Faculty of Physics and Astronomy

University of Heidelberg

Bachelor thesis

in Physics

submitted by

Dimitri Probst

born in Zatobolsk, Kazakhstan

July 2011

Analysis of the Liquid Computing Paradigm on a Neuromorphic Hardware System

This bachelor thesis has been carried out by Dimitri Probst at the

KIRCHHOFF INSTITUTE FOR PHYSICS

RUPRECHT-KARLS-UNIVERSITÄT HEIDELBERG

under the supervision of

Prof. Dr. Karlheinz Meier

Analysis of the Liquid Computing Paradigm on a Neuromorphic Hardware System

In the context of this thesis a highly accelerated mixed-signal neuromorphic hardware system has been utilized to perform real-time classification with Liquid State Machines. A self-stabilizing neural architecture implements the liquid, while the readout is realized by a modified Tempotron. To this end, the analog spiking neurons are trained to behave like a Tempotron. The original learning rule is modified to account for conductance-based synapses, discrete weights and limited hardware parameter ranges. Moreover, a software study analyzes potential classification improvements achievable for better synaptic input distributions. To test the computational power of the combined liquid-Tempotron setup, a particular task is devised which challenges the abilities of a Tempotron classifier. It is shown that the liquid enables the system to perform beyond the characteristic scope of the readout alone. In a further experiment, handwritten digits from the MNIST database are discriminated, thus showing the generality of the hardware Liquid State Machine in a real world application.

Analyse des Liquid Computing Paradigmas auf einem neuromorphen Hardwaresystem

Die vorliegende Arbeit verwendet ein beschleunigtes, analog-digitales neuromorphes Hardwaresystem um Klassifizierung in Echtzeit mit Hilfe von Liquid State Machines durchzuführen. Ein selbststabilisierendes neuronales Netzwerk realisiert dabei das Liquid, während als angepasstes Tempotron trainierte, analoge und aktionspotentialbasierte Neuronen die Auslese übernehmen. Die ursprüngliche Lernregel wird verändert um konduktanzbasierte Synapsen, diskrete Gewichte und beschränkte Hardwareparameter besser zu berücksichtigen. Darüber hinaus wird in Software untersucht, ob eine Verbesserung der Klassifizierung basierend auf einer besseren Gewichtsverteilung der Synapsen zu erreichen ist. Um die Rechenleistung des Liquid-Tempotron Aufbaus zu testen wird eine spezielle Aufgabe entwickelt, die das Klassifizierungsvermögen des Tempotrons herausfordert. Es kann gezeigt werden, dass das Liquid das System befähigt über den charakteristischen Geltungsbereich der alleinigen Ausleseeinheit hinaus zu agieren. In einem weiteren Experiment werden handgeschriebene Ziffern aus der MNIST Datenbank unterschieden und somit die Universalität der Liquid State Machine in einer Anwendung des täglichen Lebens demonstriert.

Contents

1 Introduction

The brain is the most complicated organ originating in nature. For centuries scientists and philosophers rack their brain to clear up the mystery of this marvel. How do thoughts and feelings emerge? How do we perceive the world and what are the associated neural mechanisms? To come to grips with this questions the modeling of neural networks has become an essential building block in neuro-scientific research. Modeling allows for flexibly studying neural network in different spatial and temporal scales.
The computation of artificial neural networks is commonly carried out by digital computers, which use numerical simulations in order to calculate the network dynamics. However, the computational power consumption and the experimental duration scale superlinearly with the network size. Thus a complete simulation of such a complex model like the human brain on a computer becomes an inconceivable endeavor, regarding all available computational means of today.
In order to provide a technological approach in which the experimental time does not depend on the network size, the *Electronic Vision(s) Group* [1] at the University of Heidelberg in cooperation with the TU Dresden has specialized on the development of neuromorphic hardware systems. Within the scope of the *FACETS Project* [2] and the followup *BrainScaleS Project* [3], a hardware device is being designed which can be used for the emulation of programmable neural networks. Owing to the mixed-signal Very-large-scale integration (VLSI)[1] technology and the intrinsic time constants of neuromorphic devices built in a 180 nm process, an acceleration factor of up to 10^5 compared to biological real time is achieved [4]. Due to this speedup factor, neuromorphic hardware systems can be utilized for time-consuming experiments as detailed parameter sweeps for optimizing novel model architectures, long-term processes in the brain and for reasons of statistics.
Just like any other technology, this approach has some disadvantages. Above all, the maximum network size and maximum connection density are limited by the size of the silicon chips. Regarding this aspect, the present FACETS chip-based system with its 384 built-in neurons would not suffice for being utilized in sophisticated neuro-scientific experiments. To counteract this constraints the FACETS wafer-scale system is being developed, which allows for the emulation of neural networks containing up to 180,000 neurons, 40 million synapses and thus facilitates the modeling of e.g. cortical columns. Since the FACETS wafer-scale system is in its early test phase, all experiments in this thesis have been carried out on the FACETS chip-based system.

[1]Combination of 10^3–10^5 transistor-based electronic circuits on a single chip

Outline

Chapter 2 describes the technological and mathematical prerequisites. The concept of *liquid computing* and the two utilized classification algorithms are described. Chapter 3 illustrates the two developed learning concepts which are employed throughout this thesis. Thereafter, Chapter 4 provides the experimental results. In Section 4.1 a parameter study regarding the properties of a self-stabilizing liquid architecture is performed. The realization of a Liquid State Machine consisting of a liquid and a spike-based Tempotron classifier on the FACETS chip-based hardware system is presented in Section 4.2. Furthermore, a task is defined which demonstrates the classifier's dependence on the existence of a liquid to solve classification tasks (Section 4.3). Last of all, a handwriting digit recognition task is run on the FACETS chip-based hardware system and potential constraints are outlined (Section 4.4).

2 Methods and Materials

This chapter describes the hardware emulation back-end, its operating software and the theoretical background underlying the computational paradigms which have been utilized throughout this thesis.
Section 2.1 refers to the utilized neuromorphic hardware system. It introduces the FACETS chip-based hardware system and the recently designed FACETS wafer-scale system, which is already in its early test phase.
Hereafter, in Section 2.2, the current software framework is presented with all its layers from the abstract description of neural networks in PyNN [5] down to the hardware level.
Section 2.3 states the theoretical principles this thesis is built on: A compact characterization of the Liquid State Machine (LSM) and classification concepts as well as an introduction to the employed synapse models.
In Section 2.4, the preliminary results from [6] are summarized with a particular focus on the constraints they were subjected to.

2.1 Hardware Framework

In the following, the two neuromorphic hardware systems developed during the FACETS and the BrainScaleS project are characterized.

2.1.1 FACETS Chip-Based Hardware System

The mixed-signal *Spikey* chip (see Figure 2.1) is the heart of the FACETS chip-based system. It was fabricated in a 180 nm Complementary Metal Oxide Semiconductor (CMOS) process on a 5x5 mm^2 die and is currently available in its 4th version. The chip allows for modeling up to 384 analog leaky integrate-and-fire neurons with conductance-based synapses (see Section 2.3), in which an acceleration factor of up to 10^5 can be reached, while the spike times can be recorded with a temporal resolution higher than 30 μs biological real time [4].
Each of the neurons can be connected to at most 192 synapses which amounts to 98,304 synapses on the whole *Spikey* chip. The communication via spikes is handled by the digital part of the *Spikey* chip which transmits them as digital pulses. These spikes are converted to analog signals by the so-called *synapse drivers*. Two synaptic plasticity mechanisms are implemented: First, a depressing and facilitating Short-Term Plasticity (STP) mechanism, which depends on the history of pre-synaptic spike times and models the limitation of resources needed for synaptic transmission [7]. Second, a long term plasticity mechanism by means of Spike-Timing Dependent Plasticity (STDP) [8]. On the chip, the maximal synaptic conductances $g_{max}(t)$ are varied by STP, while the synaptic weights are changed

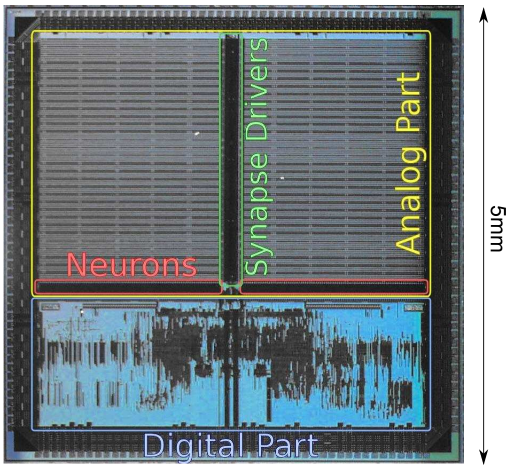

Figure 2.1: The *Spikey* chip, which is the centerpiece of the FACETS chip-based hardware system. It is divided into an analog part and a digital part. The analog part consists of two blocks, which contain the synapse arrays, the synapse drivers and the neurons. The digital part transmits spikes as digital pulses which are converted to analog signals by the synaptic drivers. Picture taken from [4].

by long-term plasticity. Due to hardware limitations, the synaptic conductances are constrained to a 4-bit resolution.
Further information about the FACETS chip-based hardware system can be found in e.g. [4]. A detailed list of chip parameters is listed in [9].
Figure 2.2 shows the entire experimental setup of the FACETS chip-based hardware system. The so-called *Backplane* is a printed circuit board, which provides power as well as a global clock signal and establishes a connection to the host computer. One *Backplane* contains 16 slots for the so-called *Nathan Carrier Boards*. Each of the *Nathan Carrier Boards* supplies one *Spikey* chip with power and provides the memory and connectivity infrastructure. In addition, multiple *Nathan Carrier Boards* can be interconnected [10].

2.1.2 FACETS Wafer-Scale Hardware System

Since the FACETS wafer-scale hardware system is still in production at the time of this thesis, it is only briefly outlined in the following. For a detailed description the reader is referred to [11].
The FACETS wafer-scale hardware system is based on the *wafer scale integration* of analog neural networks. Here, the wafer, which contains the mixed-signal Application Specific Integrated Circuits (ASICs), is left uncut after the fabrication. During the so-called *post processing* i.a. the single chips within the wafer are connected to each other ending up at a very high connection density [12]. In Figure 2.3, the composition of the main building blocks is shown. The so-called High Input Count Analog Neural Network (HICANN) chip hosts 114,688 synapses connecting up to 512 neurons. A full wafer consists of 44 so-called reticles, each of which comprises 8 HICANN chips. Thus, each wafer contains

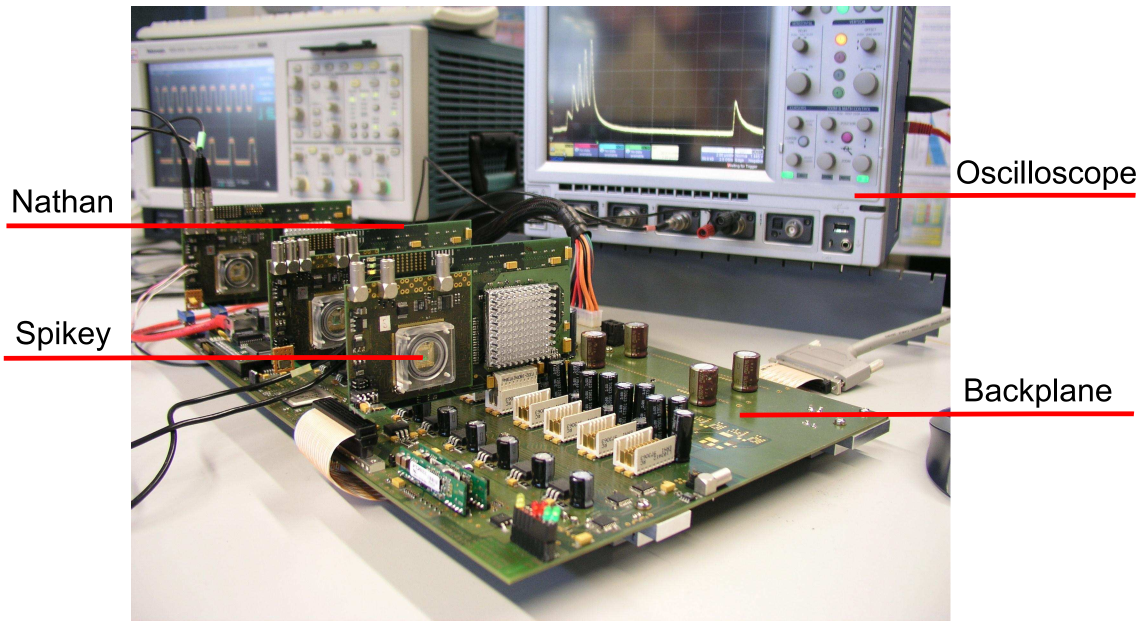

Figure 2.2: The entire infrastructure of the FACETS chip-based hardware system. Picture taken from [10].

about 180,000 neurons and more than 40 million synapses.
The membrane dynamics follow those of the adaptive exponential integrate-and-fire (AdEx) neuron model with conductance-based synapses [13]. The FACETS wafer-scale hardware system reaches an acceleration factor of up to 10^5 compared to biological real time.

2.2 Software Framework

To control and calibrate the hardware systems as well as to run experiments on them, the Application Programming Interface (API) *PyNN* [5] is used, which is based on the programming language Python [14]. *PyNN* allows for setting up neural network models and running them on numerous simulation and emulation back-ends. *PyNN* supports the software simulators NEST [15], NEURON [16], PCSIM [17], Brian [18] and the FACETS hardware systems.
PyNN is implemented as a Python package which contains a functionality common to all simulator back-ends [5]. In case of the FACETS hardware systems the Python interpreter *PyHAL* wraps the original interpreter language *SpikeyHAL* which is based on a low-level C++ API [19]. Thus *PyHAL* ensures a correct interpretation of the *PyNN* code by the lowest abstraction level of the FACETS hardware. *SpikeyHAL* uses an Automatic Repeat reQuest (ARQ) protocol to communicate with the hardware systems [20].
In order to create, connect and record neural networks, *PyNN* contains a low-level procedural API, which consists of the functions *create()*, *connect()* and *record()*, as well

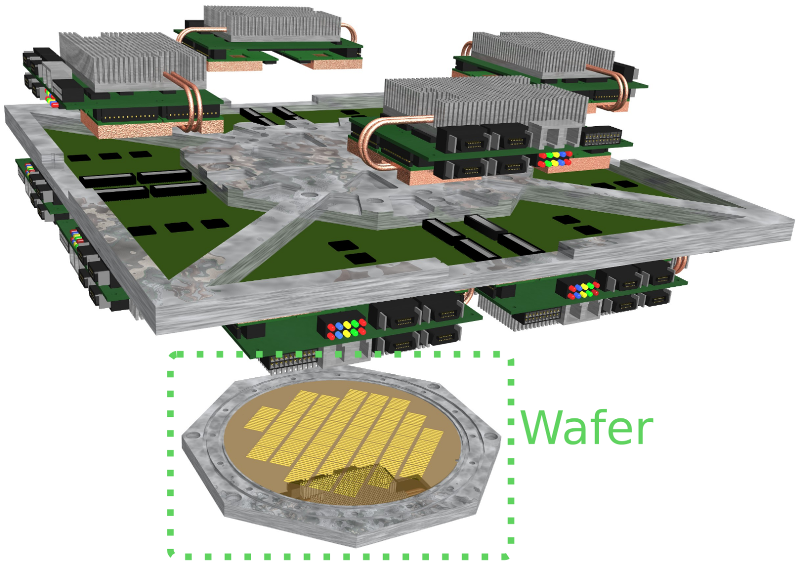

Figure 2.3: The FACETS wafer-scale hardware system. Here, the centerpiece is the wafer, which consists of about 180,000 neurons and more than 40 million synapses.

as a high-level object-oriented API, which provides classes like *Population()*, *Projection()* and different connection classes.
As listed in [5], the main advantages of the *PyNN* language are its large flexibility, an active community, the human readable code and its object-orientation.

2.3 Theoretical Framework

The subsequent section introduces the main theoretical approaches applied in the course of this thesis. At first, an insight into the mathematical description of synaptic dynamics is given. After that, the liquid computing paradigm and classification methods are presented.

2.3.1 Leaky Integrate-and-Fire Neuron Model

A leaky integrate-and-fire neuron can be modeled as a simple RC circuit (see figures 2.4a and b) [21]. The membrane voltage V behaves as follows:

$$C_m \frac{\mathrm{d}V(t)}{\mathrm{d}t} + I_{syn}(t) + g_m(V(t) - V_l) = 0, \quad \text{if} \quad V \geq V_{thresh}, \quad \text{then} \quad V = V_{reset} \,. \tag{2.1}$$

C_m represents the membrane capacitance, I_{syn} is the synaptic input current, V_l is the membrane resting potential and g_m denotes the membrane leakage conductance. The membrane time constant τ_m is determined by g_m according to $\tau_m = \frac{C_m}{g_m}$. If a certain

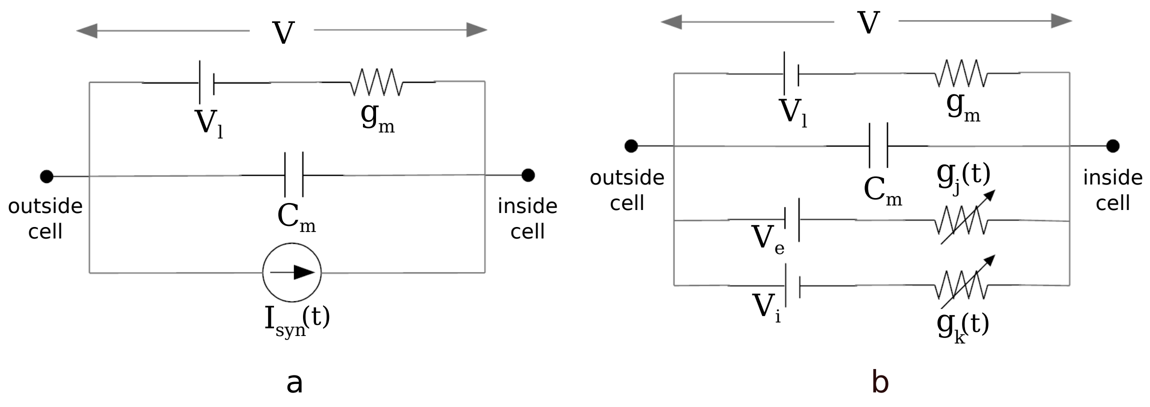

Figure 2.4: Electrical representation of a chemical synapse (inspired by [22]). **(a)** Current-based synaptic model. **(b)** Conductance-based synaptic model.

threshold voltage V_{thresh} is exceeded, the neuron emits a spike and is reset to a so-called reset voltage V_{reset}.

2.3.2 Synapse Models

This paragraph discusses two mathematical descriptions of membrane dynamics. The simple current-based synapse model is compared to the conductance-based synapse model. The information is taken from [4], [21] and [22].

Current-Based Synaptic Model

In case of the current-based synaptic model, I_{syn} from Equation 2.1 amounts to the sum of the synaptic input currents (see Figure 2.4a). Thus, Equation 2.1 shows a linear differential equation and can be solved analytically as long as the membrane voltage remains in the sub-threshold domain.
Upon arrival of a spike event at the time t_j^i from the ith afferent neuron, the post-synaptic membrane potential is determined by

$$V(t) = \sum_i \omega_i \sum_{t_j^i} K(t - t_j^i) + V_{rest} \quad . \tag{2.2}$$

V_{rest} is the membrane resting potential, which is often chosen equal to the leakage potential, and ω_i the synaptic weight of the ith afferent neuron. Each of the current-based exponential synapses adds to the neuron potential a PSP of

$$K(t - t_i) = V_0 \cdot (\exp(-\frac{t - t_i}{\tau}) - \exp(-\frac{t - t_i}{\tau_s})) \quad , \tag{2.3}$$

where V_0 is a normalization factor and τ as well as τ_s denote the membrane time constant and the synaptic current time constant, respectively.

Conductance-Based Synaptic Model

The utilized neuron model in the FACETS chip-based hardware system is a standard leaky integrate-and-fire neuron with conductance-based synapses (see Figure 2.4b). Here, the synaptic current according to Equation 2.1 amounts to:

$$I_{syn}(t) = \sum_j g_j(t)(V(t) - V_e) + \sum_k g_k(t)(V(t) - V_i) \quad , \tag{2.4}$$

where $g_j(t)$ and $g_k(t)$ are the synaptic conductances of the excitatory and inhibitory ionic channels. Furthermore, V_e and V_i represent the excitatory and inhibitory synaptic reversal potentials.
Upon arrival of a spike at time t_i from the ith input with synaptic efficacy ω_i, the change of the membrane voltage is determined by

$$\Delta V_{PSP}(t) \ \sim \ \omega_i \cdot (V_{syn} - V(t)) \cdot K(t - t_i) \quad . \tag{2.5}$$

V_{syn} is the synaptic reversal potential and K denotes the *kernel function* (see Equation 2.3). Equation 2.5 is only an approximation in case of small synaptic weights. Regarding the voltage course in response to synaptic input, Excitatory Postsynaptic Potentials (EPSPs) and Inhibitory Postsynaptic Potentials (IPSPs) have different membrane amplitudes due to individual reversal potentials (see Equation 2.5).

2.3.3 The Liquid State Machine Concept

The *liquid computing paradigm* was introduced independently by [23] and [24]. In contrast to the Finite State Machine, which requires stable states before the actual computation may continue, the LSM allows for an on-line computation on continuous input without any convergence preconditions.
As known from Support Vector Machines (SVMs) in pattern recognition [25], the classification correctness improves if the input is projected into a higher dimensional space before presenting it to the classifier. The LSM exploits this so-called *kernel trick* on computation on continuous input.
Figure 2.5 shows a schematic model of the LSM. A time-continuous input function $\mathbf{u}(\mathrm{t})$ is translated into an output function $\mathbf{y}(\mathrm{t})$. At first, arbitrary liquid filters (or operators) L^M provide a mapping of $\mathbf{u}(\mathrm{t})$ onto the, in general higher dimensional, *liquid state* $\mathbf{x}(\mathrm{t})$. After this projection, a memoryless readout f^M (for more detail see Section 2.3.4) makes a decision on the input and its history only by looking at the current liquid state. This is possible due to the liquid's intrinsic *fading memory.*
Altogether, an LSM has to satisfy two properties:

Separation property: The liquid does not have to be a neural network. However, it needs to be sufficiently complex that at any time t the distance (see [23]) between the liquid states $\mathbf{x_u}(\mathrm{t})$ and $\mathbf{x_v}(\mathrm{t})$ of two different input functions $\mathbf{u}(\mathrm{t})$ and $\mathbf{v}(\mathrm{t})$ increases.

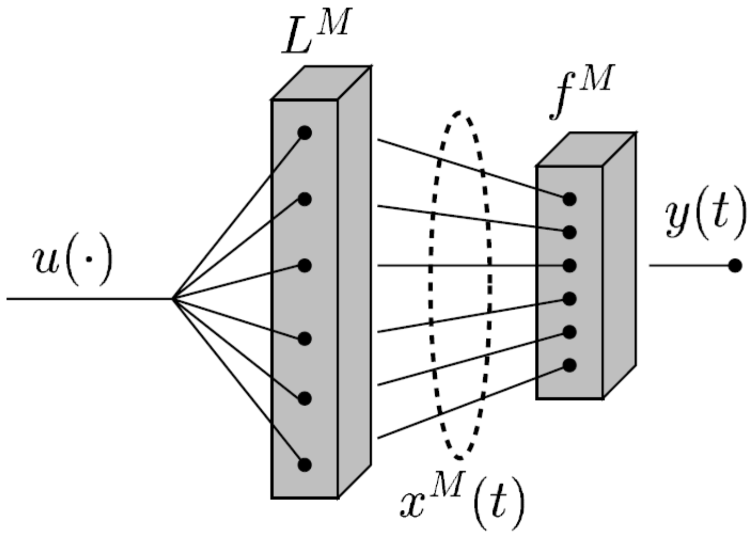

Figure 2.5: Schematic model of the *Liquid State Machine*. For a description, see Section 2.3.3. The figure is taken from [23].

Approximation property: The readout needs to provide a sufficient resolution to model any target output-function, which means that the readout's resolution needs to extract the target from any liquid state.

In any case, the complexity of the liquid should not provoke a chaotic drift of the two input functions $\mathbf{u}(t)$ and $\mathbf{v}(t)$.
For further information, [23] offers a rigorous mathematical description of the LSM and proves its inherent *universal power in computation with fading memory on functions in time*. Concerning the biological relevance of the presented concept, [26] suggests the reasonable assumption that the wiring in local volumes of the mammalian cortex is statistical, similar to the wiring in the *liquid*.
The liquid architecture utilized during this thesis is a modified version of a self-stabilizing neural network, which is proposed in [27] and employed in [28] on the FACETS chip-based system. The liquid structure is shown in Figure 2.6. It contains one excitatory and one inhibitory neuron population with recurrent connections and connections between both populations. Apart from the neuron populations themselves, the self-stabilizing feature is provided by the STP functionality [4], by which recurrent connections lead to an attenuation of too strong activity and facilitate too weak activity.
On the chip, its synapse driver limitations lead to the fact that either facilitating or depressing STP can be realized on a single driver. It is assumed that depressing connections within the excitatory population are of the greatest utility concerning the liquid's memory capacity [23]. Therefore, the constellation shown in Figure 2.6, in which only excitatory synapses are dynamic, is utilized in the course of this thesis. Apart from that, depressing connections within the excitatory population and a static connection to a hardware

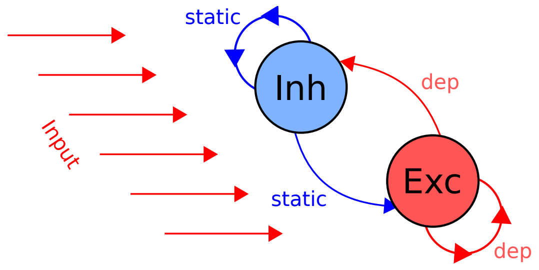

Figure 2.6: Applied self-stabilizing liquid structure. For a description, see Section 2.3.3. Figure taken from [6]

Tempotron (see Section 2.3.4) are not realizable at the same time since synaptic plasticity can only be switched on or off for all synapses of similar type. To ensure a non-chaotic liquid behavior for every utilized task in the course of this thesis, the depressing STP functionality is applied to every excitatory synapse, i.e. for the hardware Tempotron's afferent connections, which might effect the Tempotron's learning to an unexamined extent. Moreover, due to a hardware bug, only one core of the Spikey chip (see Section 2.1.1) is accessible. Consequently, the maximum liquid size is limited to 192 neurons.

2.3.4 Output Classification

In the following, two binary classifiers are presented: the *Perceptron* and the *Tempotron*. The development of both algorithms has been inspired from neural circuitry.

Perceptron Classifier

The Perceptron classifier [29] is a widely used linear discriminant model. An input vector $\vec{x}$ is linearly transformed via $\vec{\omega}^T \cdot \vec{x} + c_{thresh}$ at first. Here $\vec{\omega}$ stands for the weights vector and c_{thresh} serves as a bias. The output classification is determined by

$$f(x) = \begin{cases} 1, & \text{if } \vec{\omega}^T \cdot \vec{x} + c_{thresh} > 0 \\ 0, & else \end{cases} \tag{2.6}$$

During the learning process, the weights between the ith neuron and the Perceptron are modified by e.g. the so-called *delta learning rule*, which is introduced in [30]:

$$\omega_i^{n+1} = \omega_i^n + \Delta\omega_i^n \quad .$$

In the nth learning step, the total weight change amounts to

$$\Delta\omega_i^n = \alpha(n) \cdot (t - o) \cdot x_i \quad ,$$

with target output t, current output o and a learning rate modulation $\alpha(n)$. Since the originally proposed Perceptron (see Equation 2.6) does not account for spike time information implementation it is only implemented in software. Therefore, spiking data from the hardware or simulators of spiking neurons need to be translated by e.g. a convolution with an exponentially decaying function according to

$$\delta(t-t_i) * c \cdot \exp(-\frac{t}{\tau}) = \int_0^t \delta(t'-(t-t_i)) \cdot c \cdot \exp(-\frac{t'}{\tau})\,\mathrm{d}t' = c \cdot \exp(-\frac{t-t_i}{\tau}) \quad ,$$

leading to a time-continuous function. The summation of all spike contributions amounts to

$$x_i(t) = \sum_{t_i<t} c \cdot \exp(-\frac{t-t_i}{\tau}) \quad .$$

Here t_i denotes the spike times, c is a scaling factor and τ is the time constant of the exponential decay. A significant advantage of the Perceptron compared to the Tempotron is the rapidity with regard to the computational complexity of the learning process, which is exploited in Section 4.1.

Tempotron Classifier

The Tempotron was introduced in [31] as a classifier, whose decision is based on the spatiotemporal characteristics of the input. A common leaky integrate-and-fire neuron as implemented on FACETS chip-based hardware is utilized to carry out the classification. The afferent synaptic weights of the originally proposed Tempotron in [31] have to be trained for each new task in such a way that the Tempotron spikes exactly once while being exposed to a certain pattern A and that it does not spike while being subjected to another pattern B. However, since the originally proposed shunting of all incoming spikes after a first post-synaptic spike is on the one hand not realizable on hardware, and on the other hand not biologically plausible, the effect of this condition is neglected in the following. It is stated in [31] that the learning rule is still quite stable without this feature.
As known from pattern classification, the maximum classification correctness is achieved by minimizing the overall cost, which is here denoted by a cost function

$$E_{\pm} = \pm(V_{thresh} - V(t_{max})) \cdot \Theta(\pm(V_{thresh} - V(t_{max}))) \quad , \tag{2.7}$$

in which t_{max} specifies the time of maximal $V(t)$ and Θ is the Heaviside step function. The learning rule for the synaptic efficacies follows the so-called *gradient-descent* method: Weight changes are chosen such that $\Delta\omega_i \sim -\frac{\mathrm{d}E_{\pm}}{\mathrm{d}\omega_i}$, so that after the nth learning step, a weight increment amounts to

$$\Delta\omega_i^n = \begin{cases} \alpha(n)\sum_{t_i<t_{max}} K(t_{max}-t_i), & \text{in case of an erroneous decision,} \\ 0, & \text{in case of a correct decision.} \end{cases} \tag{2.8}$$

Here, $\alpha(n)$ describes a learning rate, which can be chosen to decay, thus allowing the convergence of synaptic weights after a finite amount of learning steps. $K(t_{max}-t_i)$ is a exponential decaying function, in which t_i denotes the spike time.

As already mentioned in [6], the implementation of the original Tempotron algorithm onto the FACETS hardware is constrained: due to hardware limitations of the maximum weight as well as the 4-bit weight resolution (see Section 2.1), the original learning rule needs to run offline on the host computer using the software simulator NEURON [16] in order to provide continuous weight changes. The resulting weights are clipped to the 16 discrete values and transferred to hardware.
A further constraint arises from the fact that the implemented model of the FACETS chip is a leaky integrate-and-fire model with conductance-based synaptic dynamics. Thus, the learning rule needs to be adapted since Equation 2.2 is no longer valid, leading Equation 2.8 only as an approximation of the *gradient-descent* method. Furthermore, EPSPs and IPSPs contribute differently to the post-synaptic membrane potential (see Section 2.3.2), which leads to a discontinuity when regarding the flipping of weights from excitatory to inhibitory and vice versa.

2.4 Preliminary Experiments

This thesis is mainly based on [6], in which a Tempotron classifier has already been implemented in software and hardware via *PyNN*. A hardware implementation of the self-stabilizing liquid (see Figure 2.6) can be found e.g. in [28].

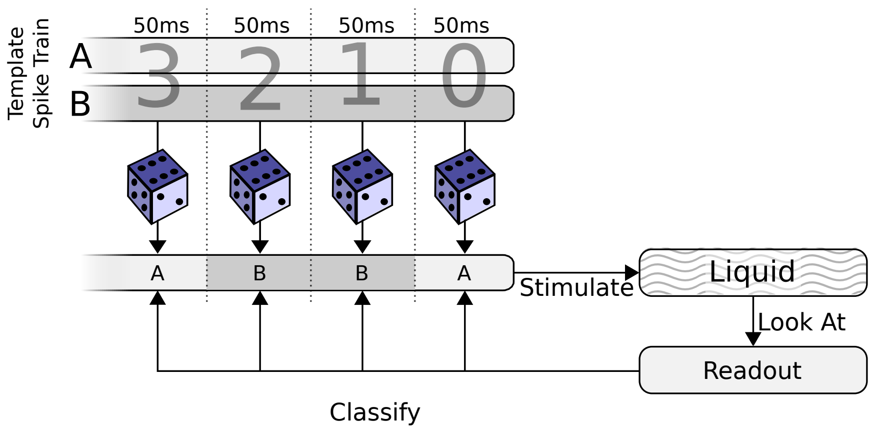

Figure 2.7: Illustration of the preliminary task to test the LSM and the liquids memory. A more detailed description can be found in the text (see Section 2.4). Figure taken from [6].

To test the quality of the LSM, a special task illustrated in Figure 2.7 has been generated. With inspiration from [32], two N dimensional Poisson process spike sequence vectors with a mean rate of 30 Hz and a temporal length of 1250 ms in biological real time are cut into 50 ms slices. Subsequently, those slices are randomly shuffled together to create new spike trains. The time of occurence of each slice is preserved during the shuffling. Additionally, spike times are varied by a Gaussian jitter with a standard deviation of $\sigma = 2\,\text{ms}$.

The shuffled and jittered spike trains are injected into N randomly chosen but fixed target neurons in the liquid, and the liquid state (see 2.3) is recorded. Ultimately, the last 50 ms of the liquids response are presented to Tempotrons, where each is supposed to classify the pattern for a specific time slice. In doing so, they implicitly measure the liquids memory capacity, because the classification of slices further in the past becomes increasingly difficult.
On the basis of the described task, a parameter study concerning the Tempotron's learning characteristics has already been performed in [6]. The mapping of the software-trained Tempotron to the hardware system has shown promising classification results regarding the current time slice (referred to as *Frame 0*) and the first time slice in the past (referred to as *Frame 1*). However, the synaptic weight range in the described experiments was artificially restricted to excitatory synapses so far, even if the liquid contained both excitatory and inhibitory neurons. Furthermore, a bundling of the liquid and the Tempotron on a single chip was proposed, which could exploit the hardware acceleration factor and reduce the required host PC communication bandwidth at the same time.

3 Learning Concepts

This chapter illustrates the two developed learning rules utilized in the course of this thesis. In the outlook of [6], several proposals are made to increase the available spectrum of input efficacies for the Tempotron with conductance-based synapses. These ideas are outlined and verified in the following.
The previous classification procedure used in [6] is illustrated in Figure 3.1. Here, the input patterns (see Section 2.4) are presented to the liquid and the resulting liquid state is recorded. In the training process as well as during the evaluation of the Tempotron's performance, every liquid neuron is connected via an excitatory synapse to the Tempotron, since the algorithmic flipping of synapse types from excitatory to inhibitory or vice versa might induce irreparable discontinuities in the learning process.
In order to accomplish the transfer of a Tempotron model which is closer to the original one (see Equation 2.8) onto the FACETS hardware, the training procedure has to become a multi-level procedure due to the restrictions for conductance-based (COBA) synapses, which arise from the fact that EPSPs and IPSPs have different amplitudes due to individual reversal potentials (see Section 2.3). Initially, the task is performed by a liquid consisting of 140 excitatory and 51 inhibitory neurons and the liquid state is buffered (see Figure 3.1). Subsequently, the simulator NEURON [16] is used to train the Tempotron's weights and to perform a validation in software. Here, two different training methods are implemented which are presented in the following.

3.1 Unconstrained Method

This training method consists of two consecutive training runs each containing 5000 training steps. First of all, the Tempotron's synapses are implemented as current-based (CUBA) synapses, whose weights are initialized following a Gaussian distribution with mean $\mu = 0\,\mu$S and standard deviation $\sigma = 0.5\,$nS. In an exponentially decaying training process with the initial rate $\alpha_0 = 0.0003$ and the learning constant $\tau_{learn} = 4000$, which are chosen in order to provide the synapses with sufficient opportunity to change, the weights are varied by following the CUBA learning rule, which can be found in appendix A of this thesis. Here, a possible flipping of weights from excitatory to inhibitory and vice versa is allowed. This is necessary because the tendency whether a synapse would become excitatory or inhibitory is generally not known *a priori* [6]. Thus the synapse type is *unconstrained.* Moreover, the maximum weight size is restricted to $0.003\,\mu$S for excitatory and inhibitory synapses to assure a later linear mapping onto hardware weights. This first training run with CUBA synapses offers a tendency towards the weight evolution in a subsequent training trial with conductance-based synapses. Consequently, the main

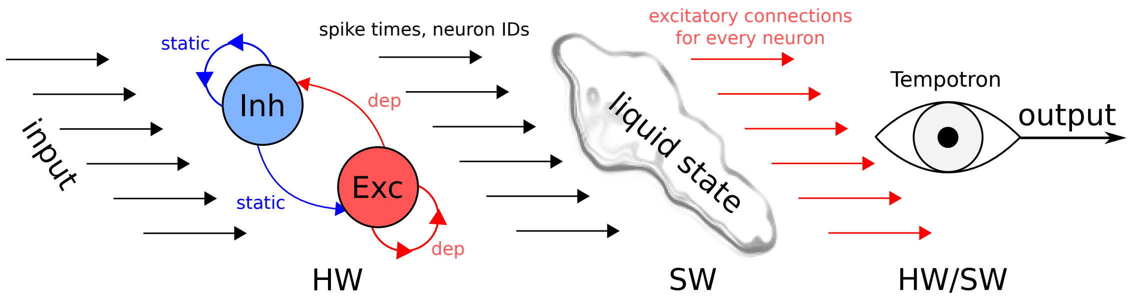

Figure 3.1: Classification procedure using the FACETS chip-based hardware. The input pattern is employed on the liquid, the spike times and corresponding neuron ID's are recorded and presented to the Tempotron readout. Due to different contributions of excitatory and inhibitory synapses to the PSP, the previous classification procedure in [6] only allowed excitatory connections to the Tempotron. In the course of this thesis, the synaptic input range was extended to excitatory and inhibitory synapses.

complicacy in a pure training with conductance-based synapses, which is finding the optimum initialization of weights, is eluded.
In a second training run, the resulting synapse efficacies in the CUBA model are used as initial weights for the training with COBA. Here, the PyNN neuron type specialized for the FACETS chip-based hardware is used. The learning rule for COBA synapses can be found in the appendix of this thesis. This learning rule uses the same learning parameters as the CUBA one. Additionally, to avoid irreparable discontinuities in the learning process, a weight flipping from excitatory to inhibitory and vice versa is prevented by setting all weights which tend to flip to zero. This discontinuity is due to the different reversal potentials $V_e = 0\,\mathrm{mV}$ and $V_i = -80\,\mathrm{mV}$. Following Equation 2.5, the dependencies of an EPSP and an IPSP respectively can be described by

$$\Delta V_{EPSP}(t) \sim g_e(t) \cdot (V(t) - V_e) \tag{3.1}$$
$$\Delta V_{IPSP}(t) \sim g_i(t) \cdot (V(t) - V_i) \tag{3.2}$$

To prevent further discrepancies for the weight updates caused by the different reversal potentials, the weight update $\Delta\omega$ (see Equation 2.8) of inhibitory synapses has to be multiplied by the factor

$$\frac{V_e - V_{mean}}{V_{mean} - V_i} \quad . \tag{3.3}$$

This step assumes (see Figure 3.2) that the Tempotron's membrane voltage change during a PSP is small compared to the distance between V_{mean} and V_e or V_i. The mean membrane voltage V_{mean} is determined by the mean value of the recorded membrane voltage course and amounts to $V_{mean} = -58.2\,\mathrm{mV}$ in the case of the described task in Section 2.4. Thus, the term 3.3 results to 2.67 and is inserted in the COBA learning rule attached in appendix A. In general, this term has to be redetermined for each new task.

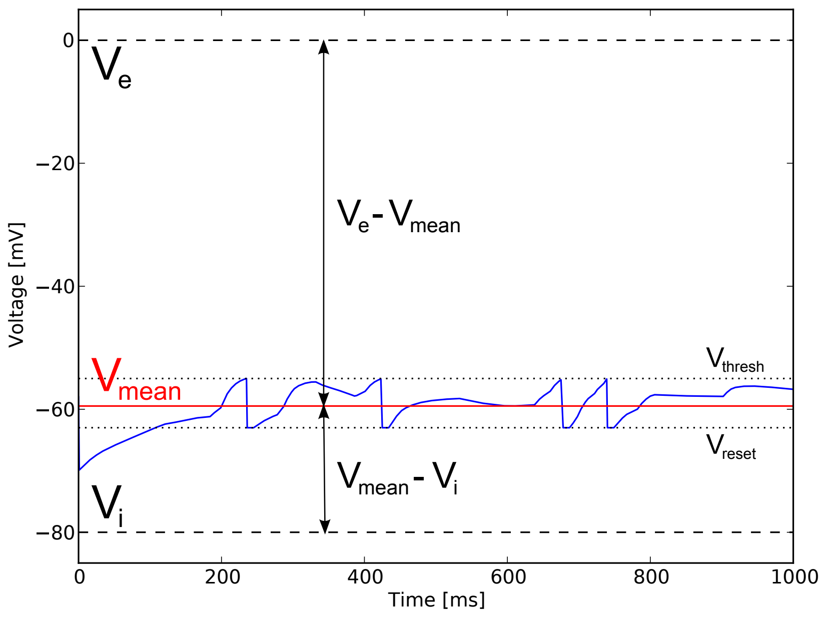

Figure 3.2: Sample voltage course of an integrate-and-fire neuron with exponential conductance-based synapses. The diagram contains the reversal potentials (dashed lines), the threshold and reset potentials (dotted lines) as well as the mean (post-synaptic) membrane voltage (red line).

3.2 Constrained Method

The second applied learning method consists of a single training run. Here, the Tempotron's synapses are directly implemented as conductance-based synapses with an initial weight of 0.1 nS for excitatory and inhibitory neurons. As opposed to the previous method, the excitatory synapses are forced to stay excitatory and the inhibitory ones inhibitory during the entire training process in which the initial decay rate is $\alpha_0 = 0.0003$ and the learning constant amounts to $\tau_{learn} = 4000$. Because the synapse type may not change during learning, this method can be described as being *constrained*. In the following, the synaptic efficacies are trained according to the above mentioned learning rule for COBA synapses which can be found in appendix A.

4 Experimental Results

This chapter summarizes the experimental results achieved during this thesis as well as conclusions drawn therefrom. In Section 4.1, the self-stabilizing liquid architecture (see Section 2.3.3) is tested in detail on the existing task described in Section 2.4. The computational power in relation to the size of the network and the inner liquid weights is examined, in particular. Thereafter, Section 4.2 depicts the necessary steps to form an any-time computing Liquid State Machine and demonstrates the performance of the classification in software and hardware. Section 4.3 illustrates a new, intricate task, which shows the impact of the liquid on the Tempotron's classification ability. The last section offers a real-world application for the Liquid State Machine. Here, handwritten digits are translated into spike patterns, which are then discriminated in a pairwise fashion by individual Tempotrons.

4.1 On the Lookout for a Suitable Liquid

Before delving into the construction of an entire Liquid State Machine on a neuromorphic hardware substrate, a parameter study has to be accomplished concerning the importance of different liquid quantities, especially with regard to later tasks. In the course of this thesis, a self-stabilizing network as described in Section 2.3 is used since it has been proven to work robust on hardware [28]. Besides, a column-based liquid model as proposed in [23] and realized on the FACETS chip-based hardware in [33] turned out to be very difficult to control on imperfect hardware.
In the following the task described in Section 2.4 is applied to the self-stabilizing liquid (Figure 3.1). The resulting liquid state is classified by a software Perceptron, whose weights are modified during a run of 1800-3000 training steps, in which the initial weight vector is chosen as $\vec{\omega}^0 = (1, ..., 1)$. As the classification of the latest 50 ms frame (*Frame 0*) is not decisive for demonstrating the fading memory property, the following classifications are based on the first time window in the past. Note that, classification on past frames is also carried out by only observing the latest 50 ms. This has been described in more detail in the corresponding task description. A complete parameter study of the liquid would go beyond the scope of this thesis, thus only selected parameters are examined.
The first analyzed quantity is the *distribution of neurons* in the liquid. So far, an arrangement of $N_e = 144$ excitatory and $N_i = 48$ inhibitory neurons had been used [6]. In Figure 4.1 one can see the impact of a small variation of the present network distribution. Conspicuously, each of the studied distributions reaches a classification rate of about 100 %, among which the version containing $N_e = 140$ and $N_i = 52$ seems to vary the least. Diagram 4.2 shows the implications of a vaster change in the network distribution. Each of the examined liquid arrangements leads to a classification rate of at least 90 %, but both

extremes (96:96 and 176:16 excitatory to inhibitory neurons) exhibit stronger fluctuations with a magnitude of approximately 10 %. Those fluctuations could arise in the case of the 96:96 network from a too weak liquid response, while in the 176:16 network case the strong excitation can support chaotic behavior and therefore result in a violation of the Separation Property (Section 2.3.3).
Another liquid property which needs to be analyzed for future classification tasks (see Section 4.4) is the *network size*. Considering a task with more than two input classes one could solve the problem by decomposing it into a set of binary classifications and consequently use multiple binary classifiers [34]. As the FACETS chip-based system is limited to 192 neurons at the time of writing, the liquid's size needs to be reduced compared to the previous studies. Figure 4.3 reveals the effect of a liquid shrinkage. Even for a network size of 8 neurons, the classification rate converges to about 80-90 %, which is only due to the simplicity of the task. Nevertheless, actual computation can be carried out with significantly less neurons, thus rendering multi-class tasks feasible. Not unexpectedly, a liquid consisting of only a single neuron results in chance-level classification, due to the irreversible compression of the input state space onto the comparably small state space of the liquid.
Additionally, the impact of changes in the strength of inter and intra-liquid connections on the classification ability has been analyzed. As described in Section 2.3, the hardware implementation of the self-stabilizing liquid contains only the depressing STP functionality for excitatory synapses. In Figure 4.4 the effect of static synapses can be observed, which manifests itself by an increase of classification fluctuations due to a more chaotic behavior. Nevertheless, the correct classification converges to about 90 %.
The effect of changing the synaptic efficacies for excitatory stimuli is displayed in Figure 4.5. Here, for a lower weight of $g_{stim,e} = 1\,nS$ the limit of a too weak liquid excitation is exceeded. The setup with $g_{stim,e} = 3\,\mathrm{nS}$ provides the most stable results.
Another study of synaptic efficacies has been performed on the intra-liquid connections. After 2000 training steps each of the configurations shows about the same result, as can be seen in Plot 4.6. Certainly, the fluctuations of the blue curve ($g_{e,e} = 2\,\mathrm{nS}$) seem to be the most attenuated after 1000 training steps.
To sum up, the self-stabilizing liquid remains stable under a broad range of parameter variations. For the subsequent experiments, the parameters shown in Table 4.1 are used. Here, the liquid size needs to be shrunk to at most 191 neurons, because at least one neuron has to be left for the adjoining implementation of the hardware readout. The remaining parameters remain unchanged to the ones used in [6], since they have shown good-natured properties already. In the following sections, the Perceptron readout is replaced by a Tempotron to finally move the LSM implementation, apart from the weight update during the training phase, to hardware.

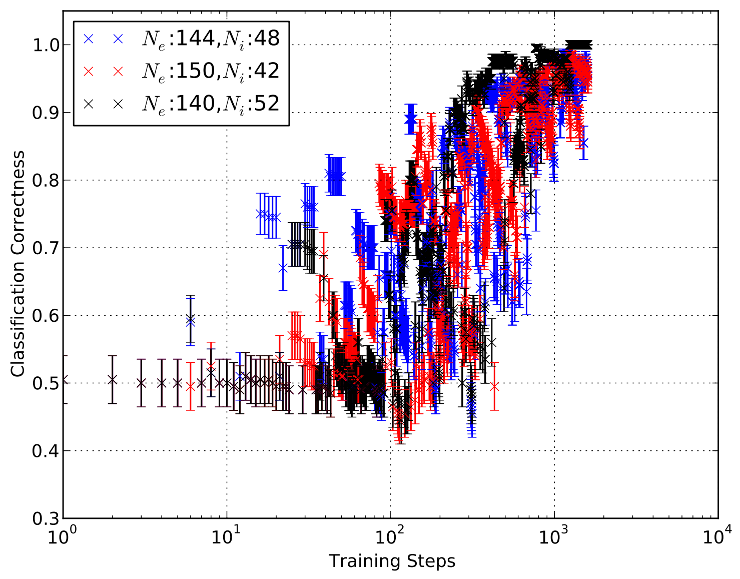

Figure 4.1: The Impact of a small variation int the liquids ratio of excitatory to inhibitory neurons on the classification ability of the Perceptron in the case of *Frame 1* classification (see Section 2.4). The depressing STP functionality is enabled.

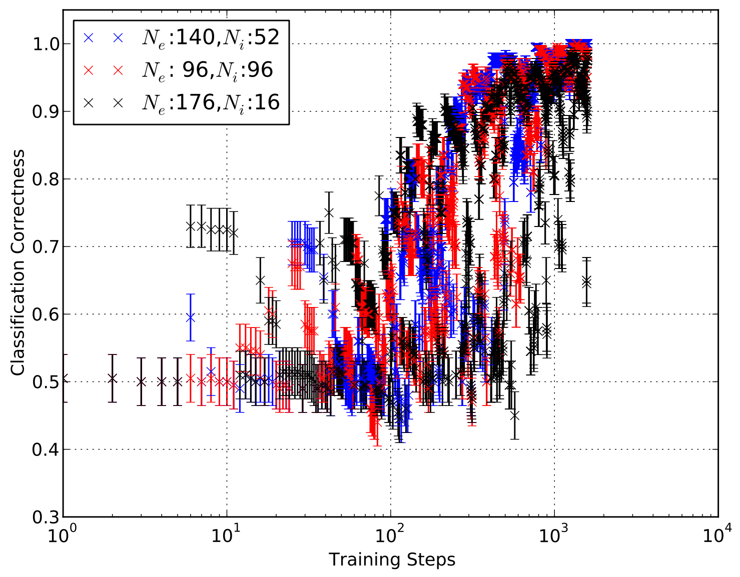

Figure 4.2: Effect of a vaster shift in the ratio of excitatory to inhibitory neurons in the case of *Frame 1* classification (see Section 2.4). Both extremes, the 96:96 and the 176:16 configurations, show slightly stronger fluctuations. The depressing STP functionality is enabled.

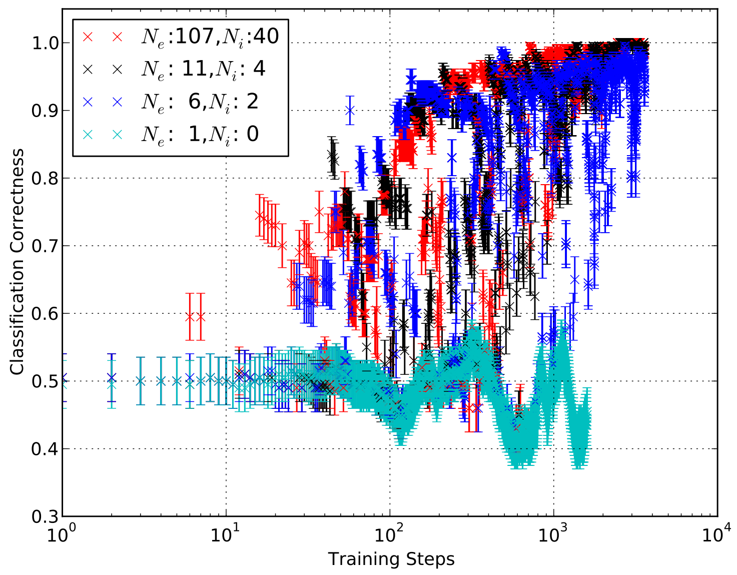

Figure 4.3: Relation between classification correctness of the Perceptron and the number of liquid neurons in the case of *Frame 1* classification (see Section 2.4). The LSM shows already acceptable results for a liquid size of only 8 neurons. The depressing STP functionality is enabled.

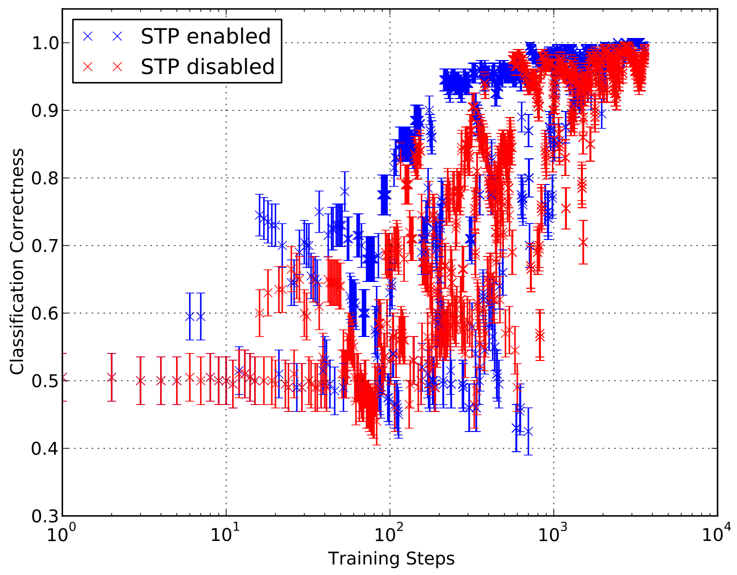

Figure 4.4: The impact of the depressing STP-functionality in hardware on the classification ability in the case of *Frame 1* classification (see Section 2.4). The utilized ratio between excitatory and inhibitory neurons was: $N_e = 107\,,\ N_i = 40\,.$

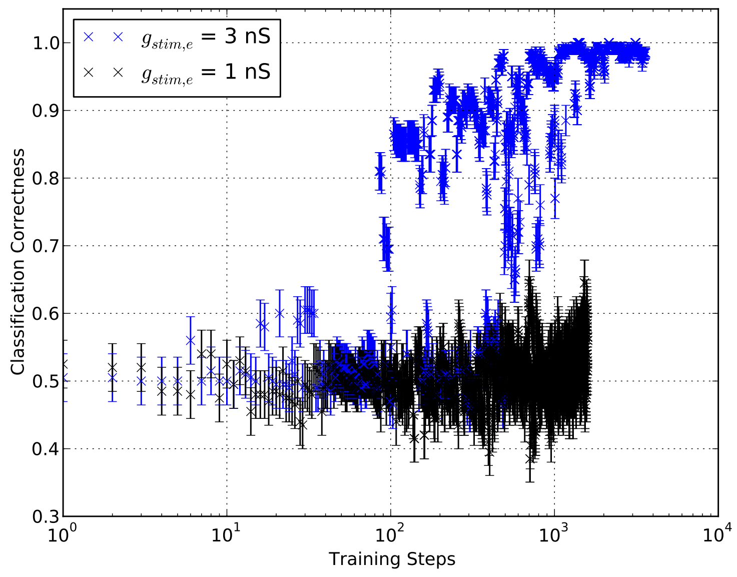

Figure 4.5: Effect of changing the synaptic efficacies of the excitatory stimuli on the Perceptron's classification ability when classifying *Frame 1* (see Section 2.4). With the given parameters, conductances of 1 nS lead to a too weak liquid activation. The utilized ratio between excitatory and inhibitory neurons was: $N_e = 107$, $N_i = 40$. The depressing STP functionality is enabled.

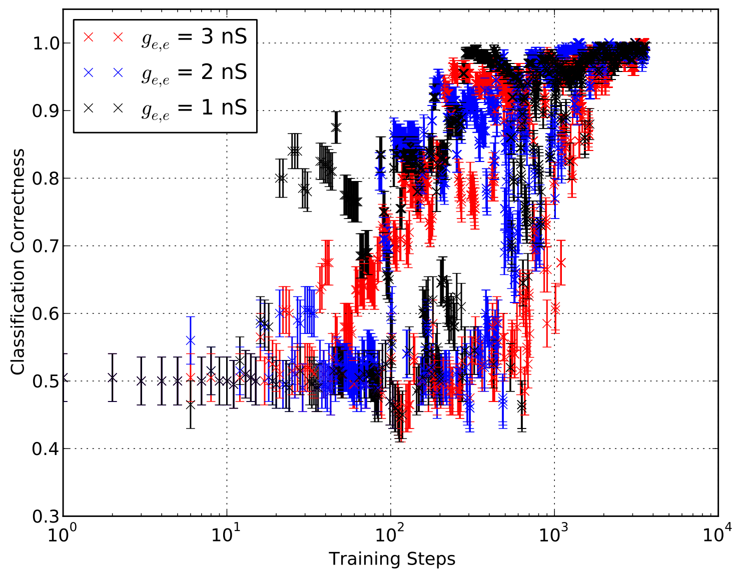

Figure 4.6: Relation between the classification correctness and the synaptic efficacies of the intra-liquid connections in the case of *Frame 1* classification (see Section 2.4). Small weight variations have hardly an impact on the classification and are therefore validating the network self-stabilizing properties. The utilized ratio between excitatory and inhibitory neurons was: $N_e = 107$, $N_i = 40$. The depressing STP functionality is enabled.

$N_{\text{excitatory}}$	140
$N_{\text{inhibitory}}$	51
$p_{\text{ee}}/p_{\text{ei}}/p_{\text{ie}}/p_{\text{ii}}$	0.05/0.1/0.1/0.2
$g_{\text{ee}}/g_{\text{ei}}/g_{\text{ie}}/g_{\text{ii}}$ [μS]	0.002/0.003/0.0015/0.002
STP (depressing)	enabled

Table 4.1: Parameters for the self-stabilizing network architecture used in the experiments in Sections 4.2 and 4.3. Here, N denotes the number of neurons, while p_{xy} and g_{xy} refer to the connection probabilities and synapse weights for the connections from population x to population y.

4.2 Hardware Implementation of a Liquid State Machine

This section describes the implementation of a Liquid State Machine with both the liquid and the readout on the FACETS chip-based hardware system (see Section 2.1.1). From here on, the previously used software Perceptron (see Section 4.1) is replaced by a Tempotron.
The weights of the software Tempotron are trained using the NEURON simulator [16] according to the learning procedures described in Chapter 3. The utilized classification task is pointed out in Section 2.4. In the following, the learning results in software are presented and afterwards, the trained conductance-based Tempotron is moved to hardware.
Figures 4.8, 4.12 and 4.14 show learning curves of the Tempotron for the classification of the current time slice (referred to as *Frame 0*), the first time slice in the past (referred to as *Frame 1*) and the second time slice in the past (referred to as *Frame 2*), as described in Section 2.4. For each frame the classification is carried out only by evaluating the current time slice.
In the case of *Frame 0*, both learning methods result in a classification correctness of about 100 % (see Figure 4.8). The classification curve which belongs to the constrained learning method shows a fast convergence to a top-level classification. This can be explained by the fact that even the small difference between the arbitrarily as $0.001\,\mu$S initialized excitatory and inhibitory weights is almost sufficient to provide 100 % confidence. As illustrated in Figure 4.10, the weights do only change slightly during the learning period. The red histogram in Figure 4.9 visualizes the final weight distribution, which is Gaussian-shaped around zero. In contrast, the unconstrained learning method results in a slower convergence of the classification correctness (see Figure 4.8). This can be explained by the long-lasting weight adaptation from the CUBA to the COBA model (see Figure 4.11). Although in both models the same weight ranges are used, the weights adapted from the CUBA model show a too strong excitation and too weak inhibition in the COBA model. Consequently, during the first 100 training steps the weights change towards lower excitation and higher inhibition leading to a temporary chance-level classification. The blue histogram in Figure 4.9 reveals the weight distribution for the *Frame 0* learning procedure employing the unconstrained learning method. The majority of weights is Gaussian distributed around zero, but few strong excitatory and inhibitory synapses have evolved, too. This is consistent with the study of [31], which uses current-based synapses.
Figure 4.12 shows the learning curves for classifying *Frame 1*. Here, both methods need about 500 training steps until the classification correctness converges. However, the unconstrained learning method leads to better results with a classification correctness of 95-100 %, whereas the learning curve of the constrained learning method fluctuates in the range of 85-90 %. This difference can be explained by the fact that due to the forbidden flipping of weights in the case of the constrained learning method, most of the synaptic weights are unable to evolve to strong synapses. Weights which should better be inhibitory are trapped as small excitatory weights and vice versa, resulting in small weights being more frequent (see Figure 4.13).

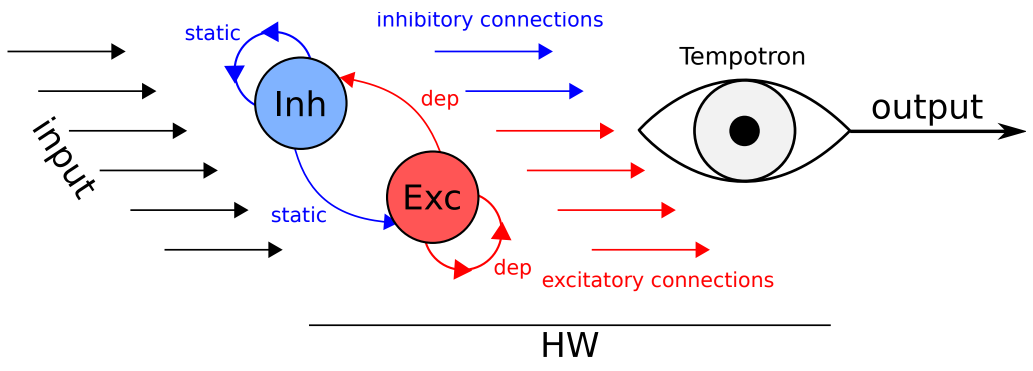

Figure 4.7: New classification procedure using the FACETS neuromorphic hardware. The input is fed into the liquid, which is directly connected to the Tempotron. The weights of the Tempotron have been trained in software according to the constrained learning method (see Section 3.2).

The histograms in Figure 4.9 and Figure 4.13 share similar weight distributions. The majority of weights is Gaussian distributed around zero and indicates inputs causing similar responses for both patterns A and B. On the other hand, the strong excitatory and inhibitory synapses emphasize the afferent neurons representing the main characteristics of the two different patterns. These strong weights are mainly responsible for the ability to distinguish between different patterns.
In Figure 4.14 the learning curves for classifying *Frame 2* are displayed. For both learning methods, the final classification correctness results in strong fluctuations around chance level. This uncertainty can be explained by the low number of strong synapses as shown by the weight distribution in Figure 4.15. This poor classification result is a consequence of the limited memory capacity provided by the liquid.
Figure 4.16 illustrates the weight evolution in the case of the unconstrained learning method. The plot displays the resulting weights after the training with COBA synapses versus the initial weights and actually shows weight flippings. As expected owing to the higher complicacy, the *Frame 1* classification causes more weight flips compared to the *Frame 0* classification. Synapses do not change their weights, if the presynaptic neurons do not fire at all (see a typical liquid response in Figure 4.17).
Comparing the outcomes of the presented learning methods, the constrained one stands out in learning rapidity. Concerning the classification correctness, the unconstrained method outperforms the constrained one in the case of *Frame 1* classification. Furthermore, the FACETS chip-based hardware system (see Section 2.1.1) does not support excitatory and inhibitory connections from the same presynaptic neuron, yet. But this feature will be available as soon as synapse mirroring is supported by the software workflow.

The results obtained by the constrained learning method suggest a realization of the liquid and the Tempotron together on the same hardware substrate. Such a setup is illustrated in Figure 4.7. Here, the liquid is directly connected to the Tempotron on the same chip so that the input spike trains can be classified immediately, without an intermediate

storage of the liquid state on the host computer. The continuous weights adapted from the software learning are mapped onto the discrete hardware weights. To avoid a mapping of all software weights onto the same hardware weight, the software weights have to be modified in a way that they occupy the entire conductance range of the synapse drivers. Therefore, the software weights with the maximum value of $0.003\,\mu$S have to be stretched by the factors `pynn.maxExcWeight`/$0.003\,\mu$S and `pynn.maxInhWeight`/$0.003\,\mu$S respectively, whereas `pynn.maxExcWeight` and `pynn.maxInhWeight` denote the maximum available conductances for excitatory and inhibitory synapses on the FACETS chip-based hardware system.
First classification trials both for *Frame 0* and *Frame 1* classifications yield promising results. Such a mapping of the software-trained weights to hardware weights leads to a classification correctness of

$$(98.4 \pm 0.4)\,\% \ \text{ for } \ \textit{Frame 0} \ ,$$

$$(85.3 \pm 1.1)\,\% \ \text{ for } \ \textit{Frame 1} \ ,$$

for a validation trial of 1000 test samples. In both cases the error is given by the standard error of the average response of the Tempotron.
Nevertheless, to find a maximum of classification correctness for *Frame 0* and *Frame 1* classifications, additional weight scaling factors stated as $p[exc] = 7.7$ and $p[inh] = 0.6$ are introduced (see Appendix A.2). The main reason for this addition arises from the fact that the mapping process from software weights to hardware weights does not work properly. E.g. the highest excitatory software weights are not mapped onto the highest of the 16 discrete hardware weights, but only onto the hardware weight *3*, which is 5 times lower than the highest weight 15. Moreover, these factors are necessary to compensate possible distortions due to the unexamined depressing STP functionality (see Section 2.3) on the Tempotron and besides, due to a lacking calibration of the synapse drivers of the utilized FACETS chip-based hardware system. This wrong weight mapping has to be investigated and resolved for future experiments. Besides, by using the second chip half to allow static connections to a dedicated Tempotron population and a more completely calibrated chip the classification results in hardware might even improve. Suggestions regarding a future implementation of the liquid and the Tempotron on the same hardware substrate are listed in the outlook of the thesis in Chapter 6.

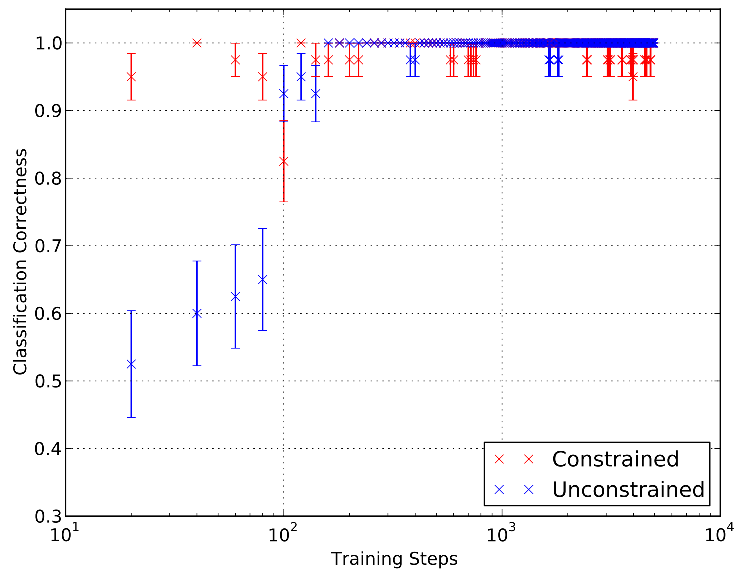

Figure 4.8: Training procedure over 5000 training steps for the classification of *Frame 0* (see Section 2.4) using a software Tempotron in the case of both learning methods (see chapter 3). Here, only training runs with COBA synapses are shown. The learning process of the unconstrained learning method takes longer to achieve top-level classification compared to the constrained learning method.

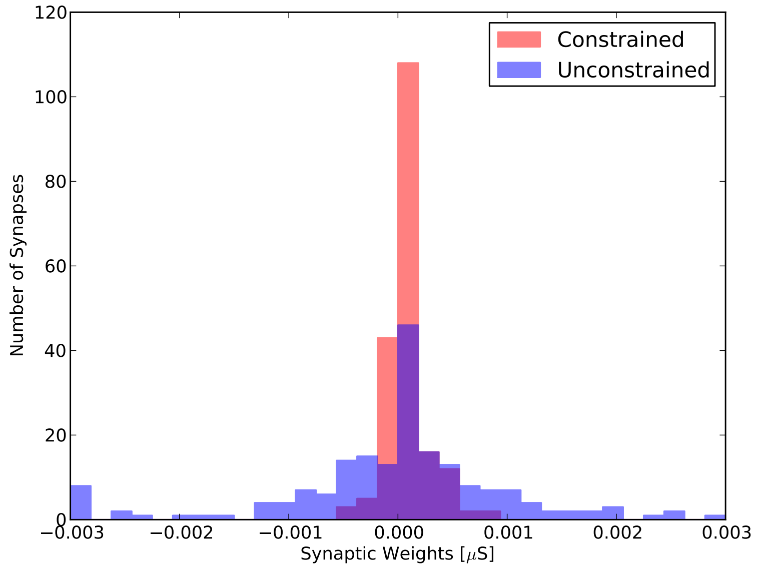

Figure 4.9: Final weight distributions of the COBA synapses after 5000 training steps of *Frame 0* classification for both learning methods (see Chapter 3). The initial weight distribution of the constrained method does not change significantly (see also Figure 4.10). The distribution of the unconstrained method ends up in a Gaussian shape around zero and several strong synapses.

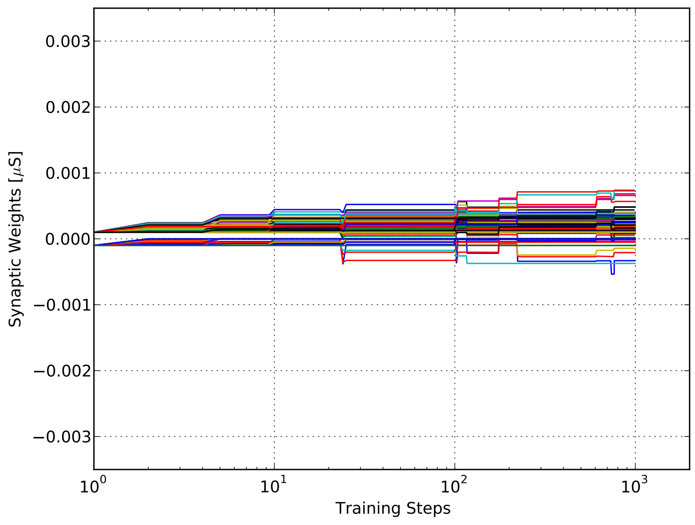

Figure 4.10: Evolution of synaptic weights during the first 1000 learning steps using the constrained learning method. Due to early successful classification results (see Figure 4.8) the weights remain close to their initial value.

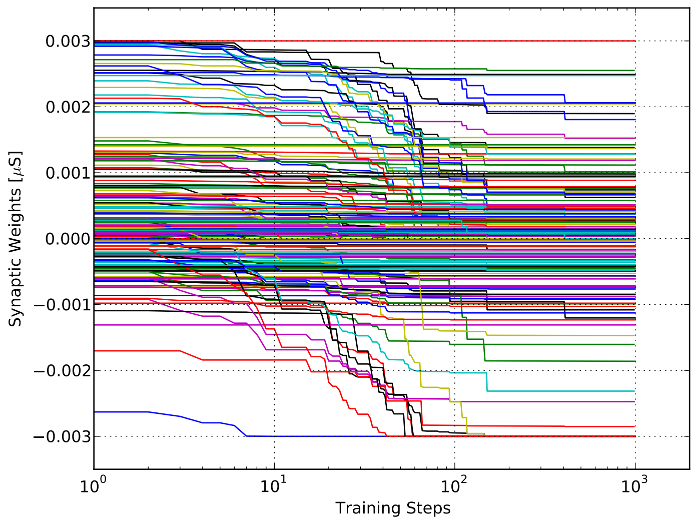

Figure 4.11: Evolution of synaptic weights during the first 1000 learning steps using the unconstrained learning method. Since the weights evolved in the CUBA model are not transferable par to par to the COBA model, quite a number of weights change towards fewer excitation or more inhibition, respectively. Thus the unconstrained learning method takes longer to reach top-level classification (see Figure 4.8).

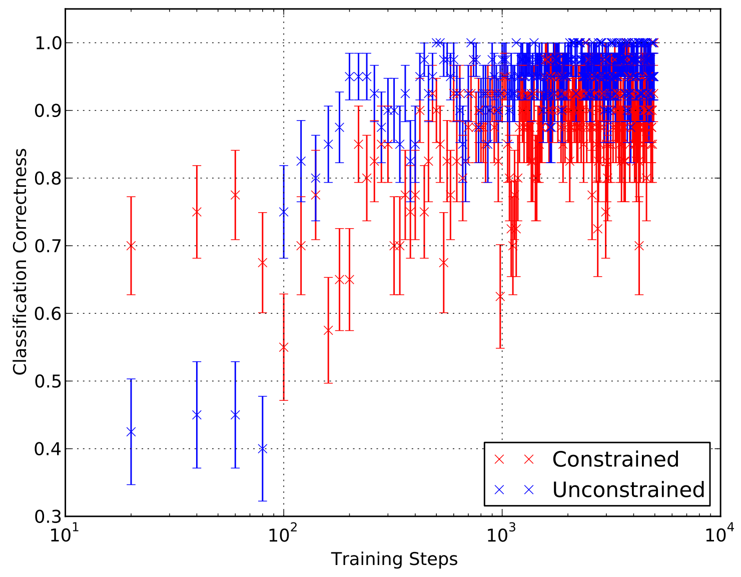

Figure 4.12: Training procedure over 5000 training steps for the classification of *Frame 1* (see Section 2.4) using a software Tempotron in the case of both learning methods (see Chapter 3). Here, only measurements for training runs with COBA synapses are shown. The classification correctness of the unconstrained learning method is better than of the constrained one.

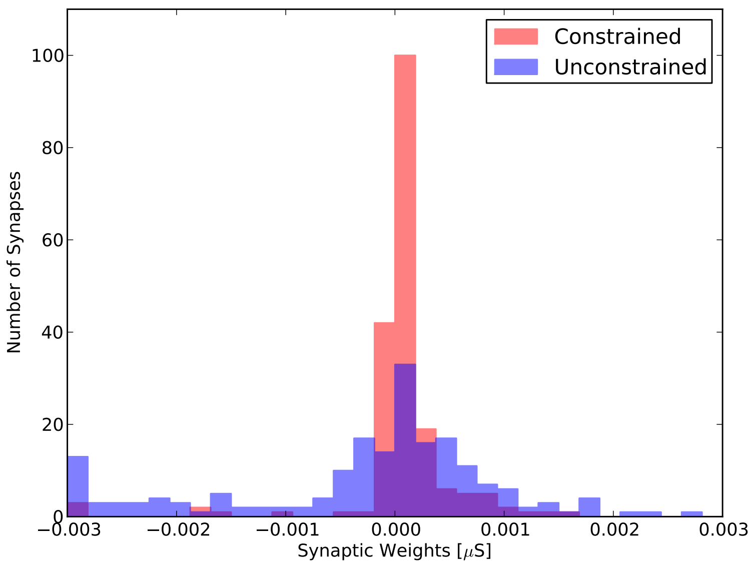

Figure 4.13: Weight distributions of the COBA synapses after 5000 training steps for *Frame 1* for both learning methods (see Chapter 3). The initial weight distribution of the constrained learning method does not change significantly. Here, weights around zero tend to flip, thus ending up in worse classifications compared to the unconstrained learning method.

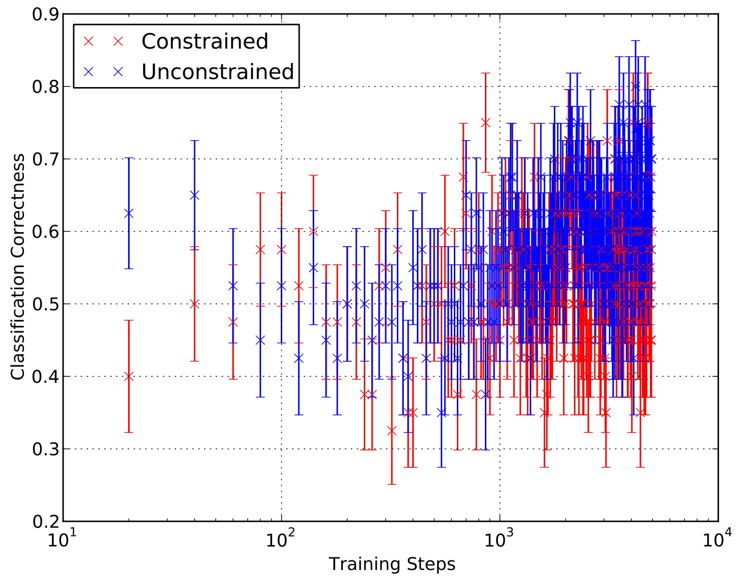

Figure 4.14: Training procedure in the case of *Frame 2* (see Section 2.4) using a software Tempotron. The measuring points show only the training runs with COBA synapses, in the case of both learning methods (see chapter 3). Here, the liquid's memory capacity does not suffice, thus the classification ends up by chance.

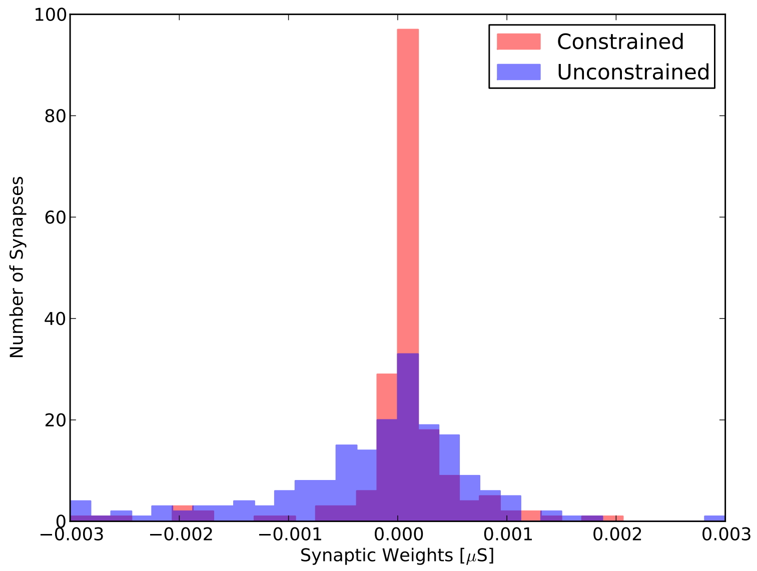

Figure 4.15: Final weight distributions of the COBA synapses after 5000 training steps for *Frame 2*. For both learning methods (see Chapter 3), the weight distribution ends up with hardly any strong synapses, which is a sign of uncertainty during the learning process. Many weights end up at medium strength compared to the previous weight distributions.

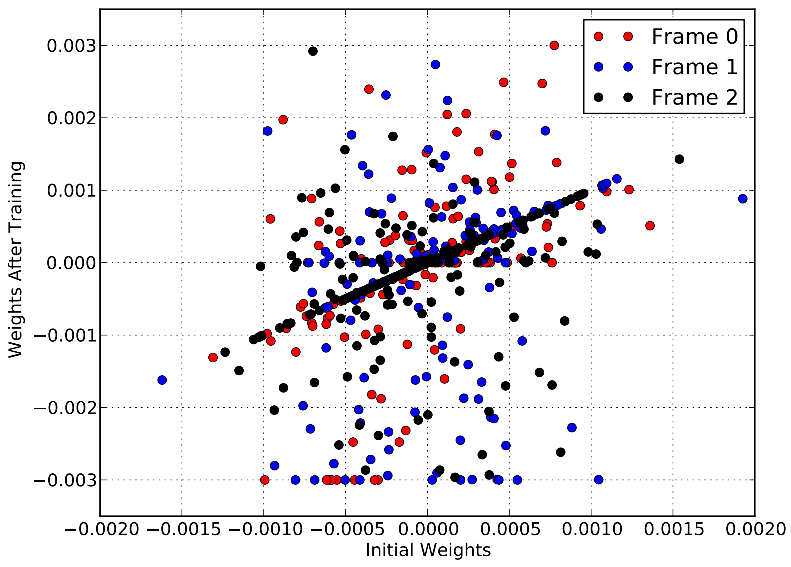

Figure 4.16: Weight changes during the unconstrained learning process (see Section 3.1). The x-axis shows the Gaussian-distributed initialized weights before the training with CUBA synapses. The y-axis reveals the weights after the final training for COBA synapses.

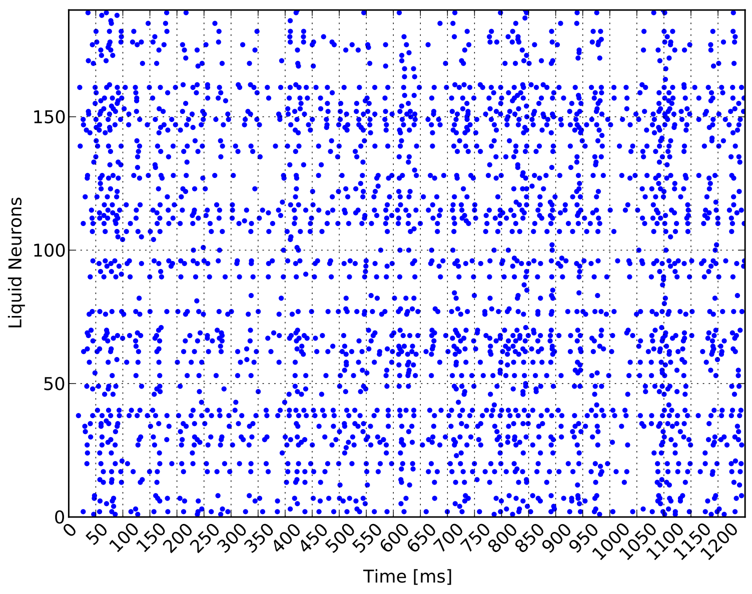

Figure 4.17: Typical liquid response for the task described in Section 2.4.

4.3 Definition of a Demanding Task

The previous Section has once more demonstrated the fading memory ability of a liquid, which manifests in the capability of the classifier to remember the past time window by only looking at the present one. Even the implementation on neuromorphic hardware has shown promising results for prospective experiments.
In the context of Liquid State Machines it is desirable to show how they can enable a classifier to perform beyond its independent capabilities, even for tasks which only consist of one time window. Hence, this Section deals with the design and application of such a task for Tempotron classifiers.
The raster plots of the two sample patterns of this task are displayed in Figure 4.18. Here, only one classification window of 100 ms in biological time is examined. The challenge is to decide between two patterns of 24 nearly coincident spikes, 12 of them are excitatory and inhibitory, respectively. The spike times are differently distributed in the classification window for the two patterns. Pattern A, which shall induce a spike as response of the Tempotron is placed at $t_A = 10\,\text{ms}$, while the spiking time $t_B = 100\,\text{ms} - \Delta t$ of pattern B is varied for each particular instance of the task. Generality is achieved by varying the afferent neurons in each training run, thus the feature to extract is only the spike time itself. Moreover, in order to further generalize the task and to avoid hardware bandwidth limitations on coincident spikes, the spike times are subject to a Gaussian jitter with a standard deviation of $\sigma = 0.7\,\text{ms}$.
Due to the low input frequency of about 30 Hz, the liquid has to be adjusted to provide a sufficiently strong but non-chaotic response on the task. Therefore, the threshold voltage of the liquid neurons is lowered to $V_{thresh} = -56.0\,\text{mV}$. More parameters of the liquid are listed in Table 4.1. The weights are trained according to both methods as described in Chapter 3.
Starting with $\Delta t = 1\,\text{ms}$ and successively increasing by 1 ms, an individual task is searched for where the sole Tempotron looses its classification ability since its only capability to distinguish patterns A and B is the delay of the post-synaptic spike in response to pattern B. An exponentially decaying function with a learning decay constant of $\tau_{learn} = 8000$ has been used for learning.
Figures 4.19 and 4.23 illustrate the classification curves for the $\Delta t = 1\,\text{ms}$-*task*. In this case, the Tempotron is still able to classify with 100 % correctness without the liquid. Regarding its weight distributions (blue steps in Figures 4.21 and 4.25), the sole Tempotron seems to avoid strong excitatory synapses in order to delay the post-synaptic spike of pattern B beyond the time window. Furthermore, inhibitory synapses are set to zero (see Figure 4.21) or even flip to excitatory (see Figure 4.25) to still spike in the event of pattern A. The classification with liquid contains erroneous trials even after 10000 training runs, but in general stays around 95 %. The weights of the constrained Tempotron with liquid stay gathered near to zero as initialized (see Figure 4.21), while the weights of the unconstrained Tempotron with liquid remain Gaussian-distributed with a few strong excitatory and inhibitory synapses since the task can be trivially solved.
The sole Tempotron's ability to delay its spiking depends on the synaptic and membrane time constants. Consequently, from a certain Δt on the maximum delay time becomes

smaller than Δt. Figures 4.20 and 4.24 reveal the training procedures for both learning methods if Δt is set to 5 ms. In both cases, the classification ability of the sole Tempotron collapses from $\Delta t = 5$ ms on to chance level, while the classification ability with liquid still remains close to 100 %. The weight distribution of the sole constrained Tempotron (blue steps in Figure 4.22) in the $\Delta t = 5$ ms-*task* indicates that the Tempotron tries to adjust the excitatory synapses in order to facilitate a spike in response to pattern A and simultaneously to provide an inhibition of the post-synaptic spike in response to pattern B since the delaying option is no more feasible. This leads to the fact that also pattern A is inhibited because both patterns contain the same amount of spikes. Thus the Tempotron alternates between spiking on both patterns or remaining silent on both of them. Regarding the results of the unconstrained method (blue steps in Figure 4.26), the weight distribution of the sole Tempotron ends up with almost all weights at zero. in the case of the Tempotron with liquid (red steps in Figures 4.22 and 4.26), the weight formations still remain similar to that of the 1-ms-task for both learning strategies, which points to the fact that the 5-ms-task is solved in the similar way as the 1-ms-task by exploiting the prolonged spike delay provided by the liquid itself.
In Figure 4.27, the classification correctness is plotted as a function of the chosen Δt. The plot outlines the results of using the Tempotron with or without liquid and following the constrained or unconstrained learning method. Besides, validation trials are performed on the hardware LSM, which is stated as constrained hardware Tempotron with liquid. Each measuring point corresponds to a validation run of 1000 test samples. As one can see, the Tempotron without liquid (black and cyan crosses) loses its classification ability from $\Delta t = 4$ ms upwards for both learning methods. On the other hand, the constrained software Tempotron with liquid (red crosses) keeps its classification ability even until $\Delta t = 10$ ms. The created LSM (purple crosses), that is the hardware analogon, classifies until $\Delta t = 6$ ms close to perfect, in spite the restrictions that are discussed in Chapter 3. Thus, it performs by far better than the sole Tempotron. The most surprising result is supplied by the unconstrained Tempotron with liquid (blue crosses). Here, the classification property still seems to remain over and above $\Delta t = 30$ ms. For this task, a classification correctness of more than 70 % can be registered, which can not be explained by a pure delaying of pattern B.
Figure 4.28 displays the weight distributions after the unconstrained learning process with liquid for $\Delta t = 20$ ms and $\Delta t = 30$ ms. Here in both cases, many strong inhibitory synapses have evolved during the learning which suggests the necessity of a strong inhibition of pattern B to solve the difficult task. It is clear that no strong excitatory synapses evolve, which would evoke a post-synaptic spike as response on each of the presented patterns.
Consequently, Figure 4.27, together with Figures 4.20 and 4.24, shows in the case of a Tempotron how a Liquid State Machine can enable the classifier to perform beyond its own characteristics.

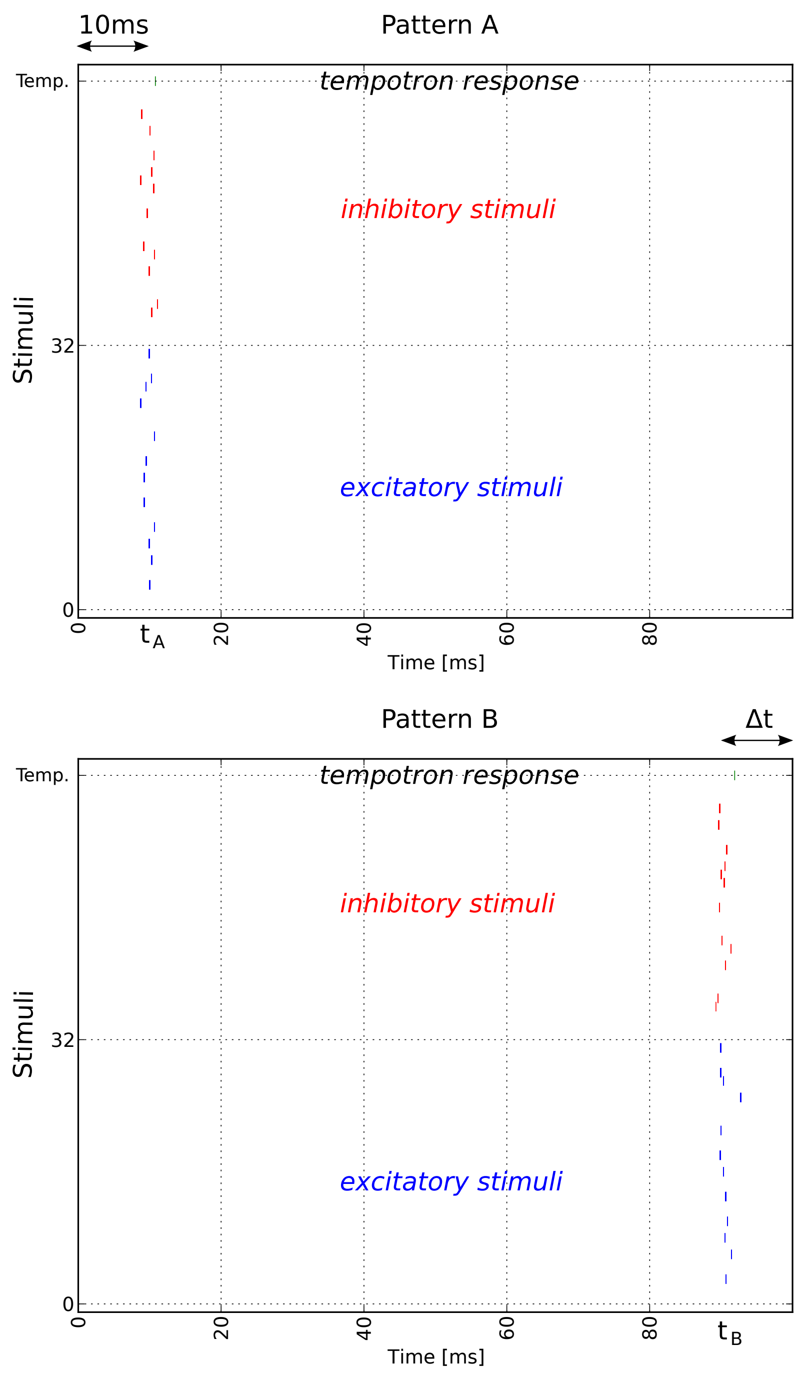

Figure 4.18: The figure reveals two sample patterns of the demanding task. Each of the patterns consists of 24-lets, 12 (of 32) excitatory and 12 (of 32) inhibitory spikes. In each training step, the afferent stimuli alternate randomly. From a certain Δt on the task to decide between pattern A and pattern B becomes a insolvable task for the sole Tempotron.

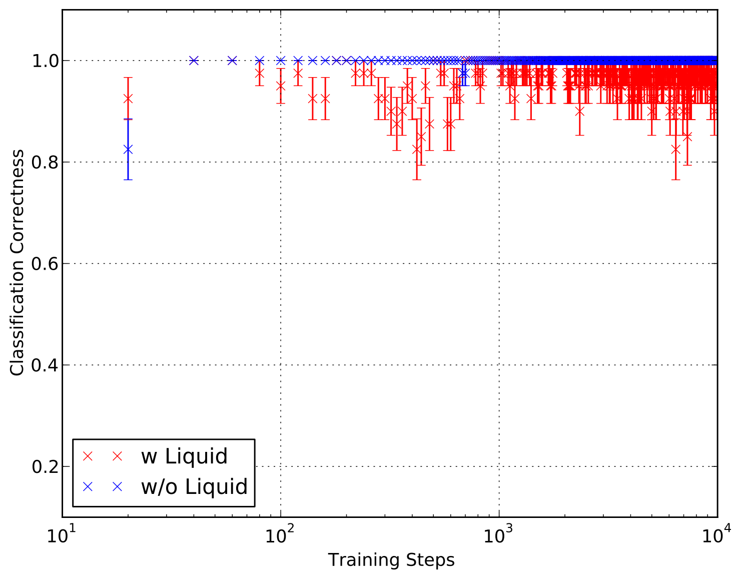

Figure 4.19: Training procedure with $\Delta t = 1\,\text{ms}$ using the constrained learning method (see Chapter 3). In both cases, with or without liquid, the classification converges to about 100 % correctness.

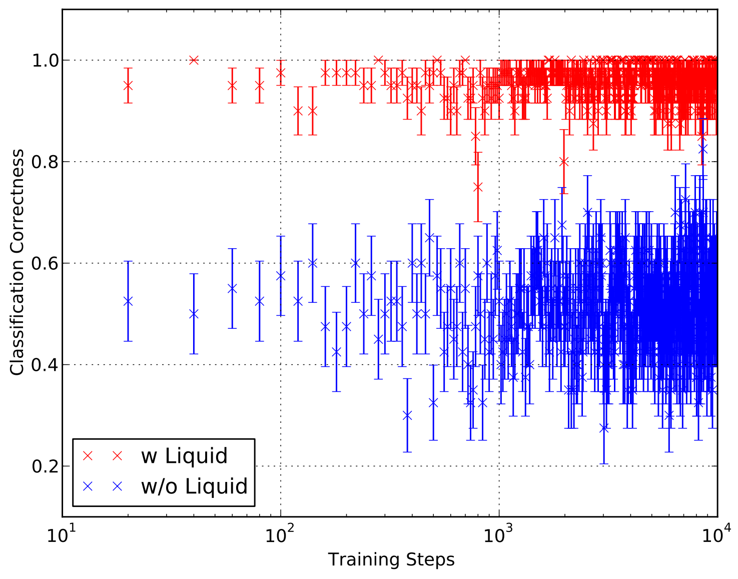

Figure 4.20: Training procedure with $\Delta t = 5\,\text{ms}$ using the constrained learning method (see chapter 3). Here, after a session of 10000 training steps the classification correctness of the Tempotron without liquid stays around chance level, while the classification with liquid converges to top level.

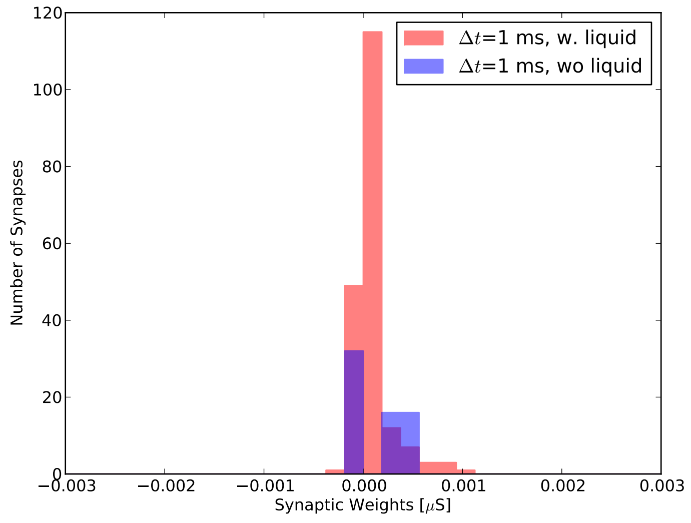

Figure 4.21: Weight distribution for $\Delta t = 1\,\mathrm{ms}$ after 10000 training steps in the case of the constrained learning method (see Chapter 3). The sole Tempotron tries to avoid strong excitatory weights in order to delay the incoming pattern B beyond the time window. Inhibitory synapses end up at zero. The weights of the Tempotron with liquid stay gathered around zero.

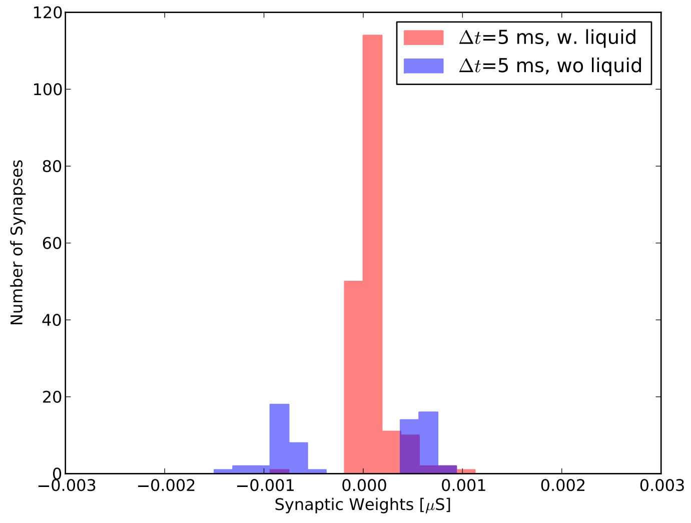

Figure 4.22: Weight distribution for $\Delta t = 5\,\mathrm{ms}$ after 10000 training steps in the case of the constrained learning method (see Chapter 3). The Tempotron tries to adjust the weights in order to spike on pattern A and remain silent in response to pattern B, which is not possible since A and B are too similar. in the case of the Tempotron with liquid, the weights remain around zero which is possible due to additional delay and the memory capacity provided by the liquid.

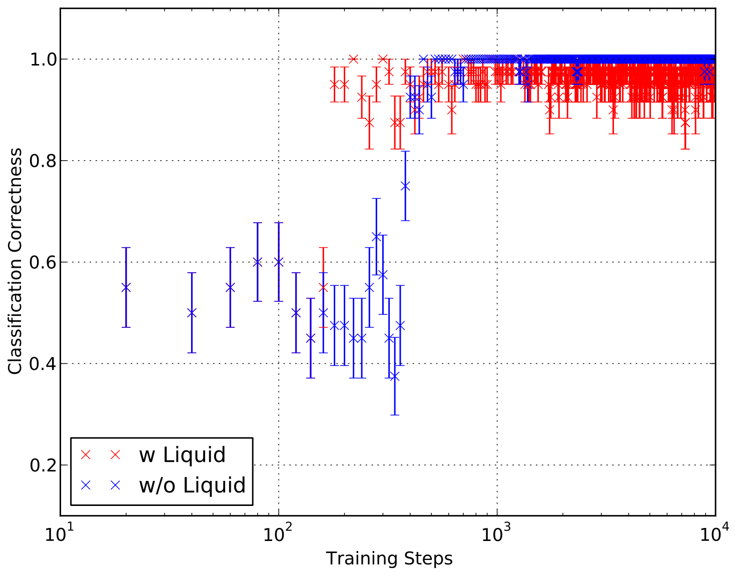

Figure 4.23: Training procedure with $\Delta t = 1\,\mathrm{ms}$ using the unconstrained learning method (see Chapter 3). In both cases, with or without liquid, the classification converges to about 100 % correctness.

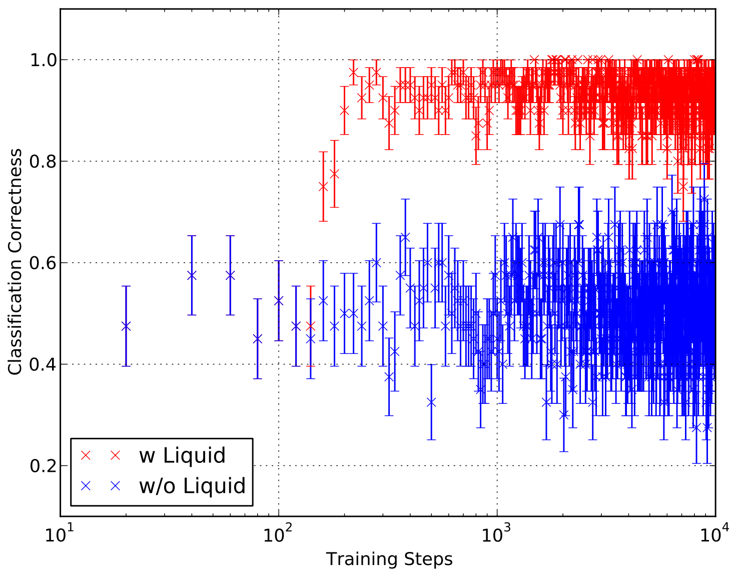

Figure 4.24: Training procedure with $\Delta t = 5\,\mathrm{ms}$ using the unconstrained learning method (see Chapter 3). Here, after a session of 10000 training steps the classification correctness of the Tempotron without liquid stays around chance level, while the classification with liquid converges to top level.

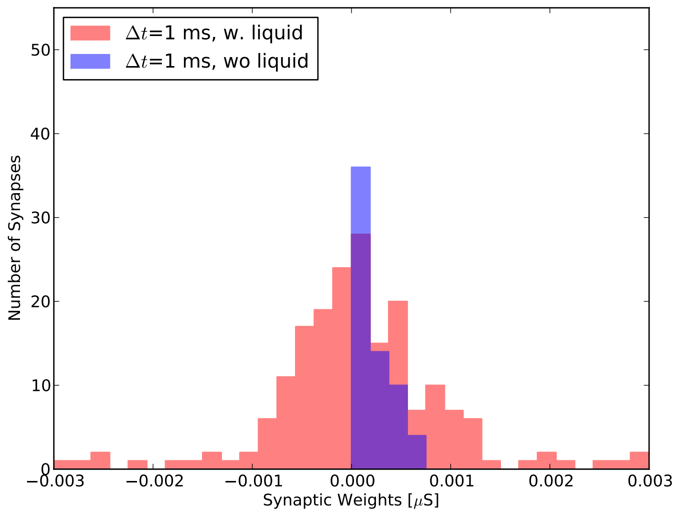

Figure 4.25: Weight distribution for $\Delta t = 1\,\text{ms}$ after 10000 training steps in the case of the unconstrained learning method (see Chapter 3) and COBA synapses. The inhibitory weights of the sole Tempotron have flipped to excitatory, but all weights stay close to zero. The Tempotron with liquid ends up with almost all weights distributed around zero and several strong synapses.

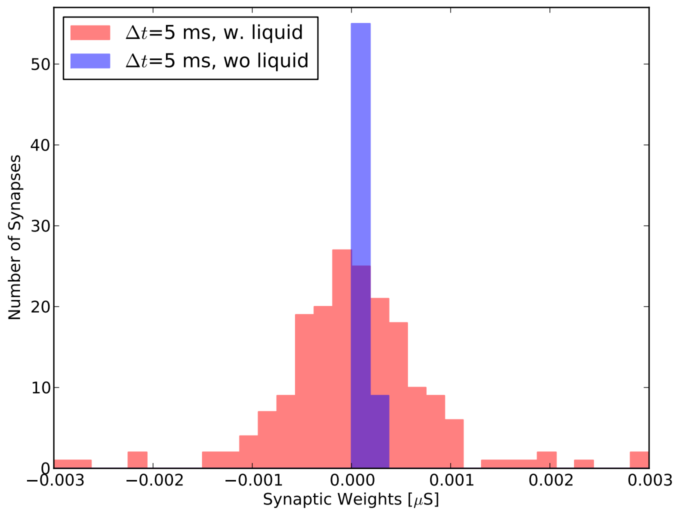

Figure 4.26: Weight distribution for $\Delta t = 5\,\text{ms}$ after 10000 training steps in the case of the unconstrained learning method (see Chapter 3) and COBA synapses. Almost all weights of the sole Tempotron evolve to zero due to uncertainty. The Tempotron with liquid is still similar to Figure 4.25.

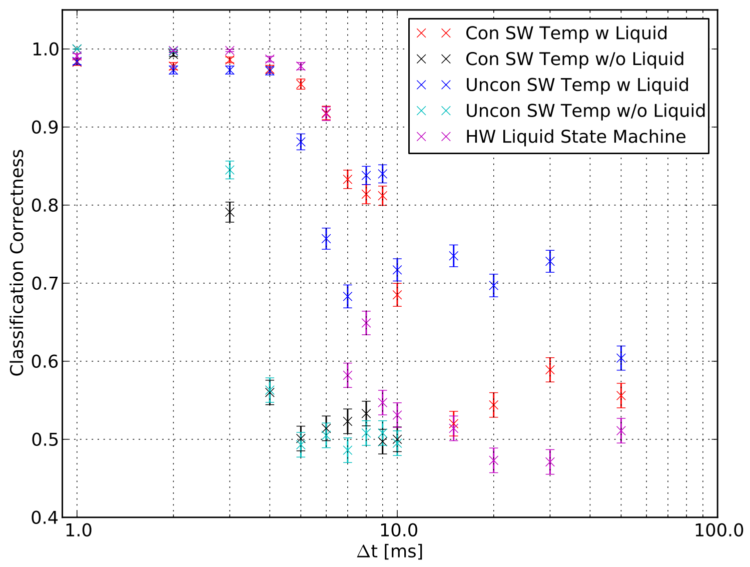

Figure 4.27: Classification correctness as a function of Δt (see Figure 4.18). The sole Tempotron already loses its classification ability at $\Delta t = 4\,\text{ms}$. The constrained Tempotron with liquid shows acceptable results until $\Delta t = 6\,\text{ms}$ in the case of hardware and $\Delta t = 10\,\text{ms}$ in the case of software. The unconstrained Tempotron (blue crosses) shows even at $\Delta t = 30\,\text{ms}$ classification results clearly beyond chance level. Each data point corresponds to 1000 validation runs using the final weights.

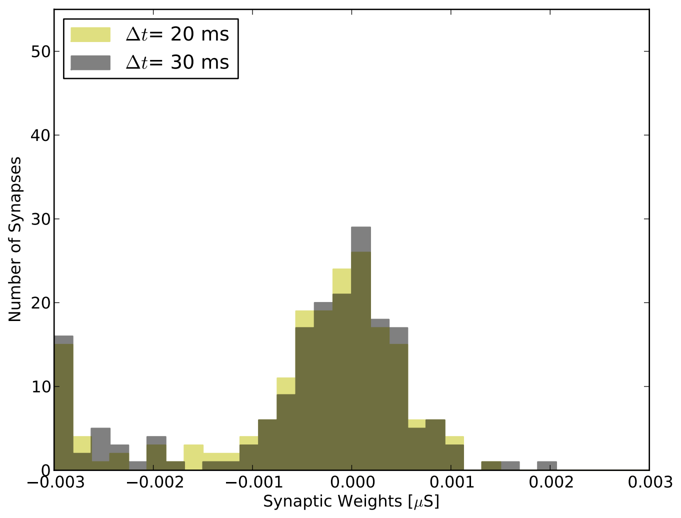

Figure 4.28: Weight distribution for $\Delta t = 20\,\text{ms}$ and $\Delta t = 30\,\text{ms}$ (see Figure 4.18) in the case of the unconstrained learning method (see Chapter 3).

4.4 Handwriting Recognition Using a Population of Tempotrons

Having the promising results of the previous experiments in mind, this section offers a possible application of the prepared Liquid State Machine in terms of a handwritten digit recognition task. The Modified NIST (MNIST) database provides the handwritten digits [35] used in this section. It contains 60000 training samples and 10000 test samples of handwritten numbers from 0 to 9 written by 250 people. Each of the digits is normalized in size and centered in a 28×28 pixel image by computing the center of mass of the pixels [36].
To make the digits obtainable for the Tempotron, the pixels have to be translated into spike patterns first. The FACETS chip-based hardware system can receive 256 inputs per core but due to the chips topology and the fact that we need all 192 neurons for either liquid and Tempotron only 64 inputs remain. Consequently, not every pixel of the 28×28 images can be translated into an individual spike train. Hence, the image needs to be compressed. Two different protocols are used in the course of this thesis to carry out the translation of pixels into spike patterns.

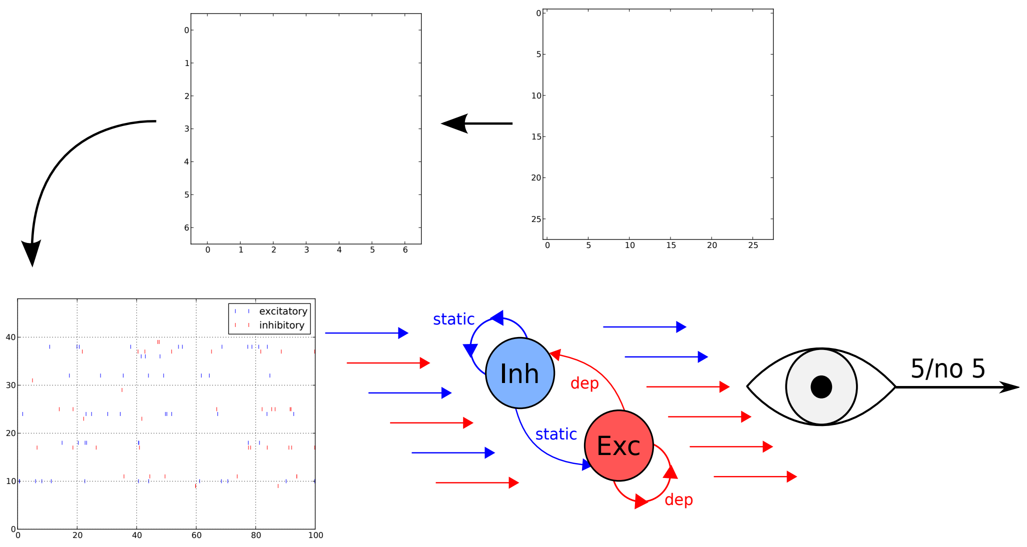

Figure 4.29: First spike encoding protocol for the handwriting recognition task. Here, the image with 28×28 pixels is initially translated into one with a resolution of 7×7 pixels, thus reducing the number of input stimuli to 49. Then, the grey value of each input is translated by means of a Poisson process into a firing pattern with a matching rate, whereby excitatory and inhibitory stimuli take turns for consecutive pixels.

Illustration 4.29 reveals the first protocol. Here, the 28×28 image is rescaled to a resolution of 7×7 pixels. The color of each pixel in the low resolution image is given

by the mean pixel color of the high resolution source. Then a Poisson process is used to linearly translate the grey values of the resulting 49 pixels into spike trains with a maximum rate of 80 Hz and a duration of 100 ms. For every two adjacent pixels, one is translates into an excitatory input pattern and the other to an inhibitory one. Ultimately, the readouts are trained in a learning process with an exponentially decaying learn rate ($\tau_{learn} = 5000$) according to the constrained and unconstrained learning methods described in Chapter 3.
Since the Tempotron is a binary classifier, it only allows for the discrimination of two different input patterns corresponding to two different digits. Therefore, to classify more than two digits, the liquid can be connected to a population of Tempotrons, with each of them trained to differentiate between two digits. Hence, the size of the self-stabilizing liquid (see Section 2.3) has to be reduced to leave sufficient space for at least $\frac{n(n-1)}{2}$ Tempotrons to distinguish between all numbers from 0 to n via pairwise classification. However, a reduction of the size does not necessarily yield a reduction in classification performance, as Section 4.1 reveals. The specifications of the used liquid is listed in the following Table 4.2.

$N_{\text{excitatory}}$	107
$N_{\text{inhibitory}}$	40
$p_{\text{ee}}/p_{\text{ei}}/p_{\text{ie}}/p_{\text{ii}}$	0.05/0.1/0.1/0.2
$g_{\text{ee}}/g_{\text{ei}}/g_{\text{ie}}/g_{\text{ii}}$ [μS]	0.002/0.003/0.0015/0.002
STP	enabled

Table 4.2: Parameters of the self-stabilizing network architecture used for the handwritten digit recognition task. Here, N denotes the number of neurons, while p_{xy} and g_{xy} are the connection probability and synapse weights for the connections between population x and y.

In the course of this thesis, classifiers for the numbers from 0 to 3 are created, whereas each of the Tempotrons is trained according to Table 4.3.
Figure 4.32 shows that the separation between the digit *one* and any other number appears to be a comparably simple task. This is not surprising, as the shape of a *one* is distinctly discernable even at a resolution of 7×7 pixels (see Figure 4.30). Thus, the classification correctness reaches high rates, fluctuating around 90% in the 1/2 and 1/3 cases and go even beyond that in the 0/1 case. The constrained learning protocol has been used to train the readout.
Figure 4.33 illustrates the discrimination between patterns 0/2, 0/3 and 2/3. In all cases the constrained learning method has been applied. For those particular patterns the lateral expansion (see Figure 4.30) causes difficulties, thus the classification tasks are subject to stronger fluctuations. The classification curves for the patterns 0/2 and 0/3 end up oscillating around 80-100 %. The separation of *two* and *three* even fluctuates around 50 and 90 %.
The Figures 4.34 and 4.35 reveal the corresponding results for the unconstrained learning method. Each of the three simple separations (see Figure 4.34) converges to a correctness

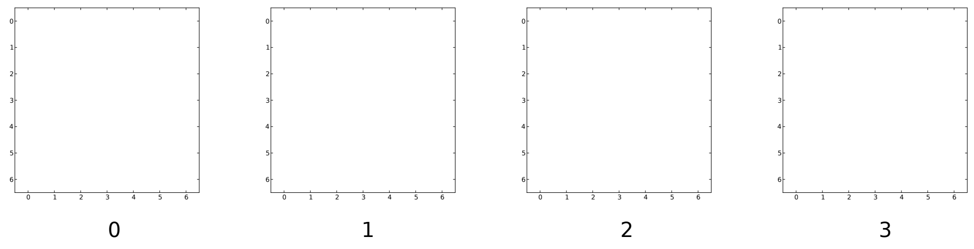

Figure 4.30: Numbers 0 to 3 in encoded via the first spike encoding protocol. Conspicuously, the shape of the *one* is clearly distinct from that of the other numbers.

Spike	No Spike
0	1
0	2
0	3
2	1
3	1
3	2

Table 4.3: Training targets utilized in the context of the handwritten digit recognition task. E.g. the first Tempotron is trained to spike in response to a 0 and remain silent in response to a 1.

of 90-100 %. Considering the more demanding tasks (see Figure 4.35), the classification of 0/3 leads to a remarkable correctness of 90-100 %, while in the 0/2 and 2/3 cases a large uncertainty remains. Here, significant fluctuations between 60 and 90 % are observable. Table 4.4 lists the classification results after a validation run of 1000 test samples of a pairwise digit classification by means of the software Tempotron and the hardware LSM using the first spike coding protocol. In the 0/1, 0/3, 1/2 and 2/3 cases the hardware results are close to those of the software prototype. Only for the distinction in the 0/2 and 1/3 cases the hardware LSM's classification correctness is off by more than 10 % compared to the software Tempotron.
The most difficult task turns out to be the distinction between the digits *two* and *three*. This case is examined more carefully, in the following. An experiment series concerning the impact of the liquid size on the classification correctness after 7000 training steps is recorded. To stay within hardware limits only liquid sizes of up to 191 neurons have been considered. The ratio of excitatory to inhibitory neurons has been kept constant at all times ($N_e/N_i = 2.75$). The results of the pure software experiment after 1000 validation steps are visualized in Figure 4.36. Up to the maximum of neurons hardly any connection between the liquids size and its performance can be drawn for either training methods. In all but one case, the unconstrained method results in up to 10 % higher classification rates than for the constrained one. However, due to the comparably low limit of 191 neurons a potential improvement in performance for even larger liquids can neither be predicted nor ruled out. In a prospective analysis of the digit recognition task, larger

Digits	0/1	0/2	0/3	1/2	1/3	2/3
SW Temp	96.2 ± 0.6 %	81.5 ± 1.2 %	87.2 ± 1.1 %	90.1 ± 1.0 %	88.7 ± 1.0 %	75.6 ± 1.4 %
HW Temp	94.0 ± 0.8 %	68.0 ± 0.5 %	85.8 ± 1.1 %	83.3 ± 1.2 %	74.7 ± 1.4 %	69.4 ± 1.5 %

Table 4.4: Comparison of the classification rates for pairwise digit recognition using on the one hand a software Tempotron and on the other hand the hardware LSM. The used spike coding scheme is pictured in Figure 4.29. The error is given by the standard error of the average response of the Tempotron over 1000 test runs.

software liquids might be used to examine the general impact of the liquids size on the classification rate.

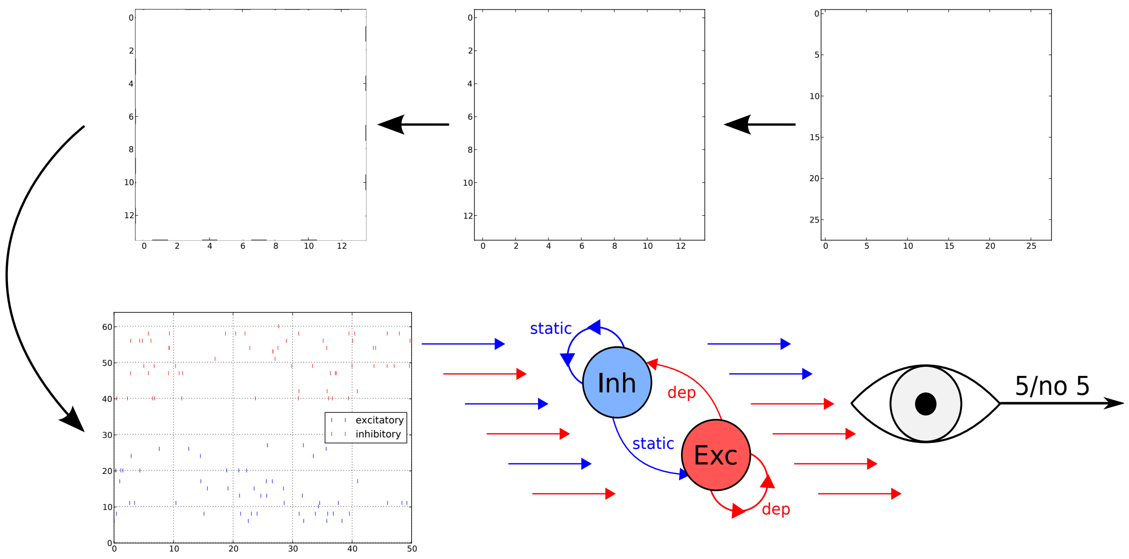

Figure 4.31: Illustration of the second spike encoding protocol used for the handwriting recognition task. Here, each third pixel of the rescaled 14 × 14 image is translated into a Poisson process spike train with a respective firing rate, whereby excitatory and inhibitory stimuli take turns for consecutive pixels. Therefore, 64 input stimuli are fed into the liquid, which equals the maximum remaining inputs on a fully occupied chip.

In Scheme 4.31 another tested protocol is shown. Here, to "increase" the resolution, the 28 × 28 images are rescaled to a 14 × 14 size, from which only every third pixel is used for coding. Each of these 64 resulting pixels is translated into a single spike train, this time of 50 ms length, since the length of 100 ms in the previous coding scheme might be a further barrier to the classification ability of the Tempotron, which does not in general rely on a firing rate. Again, the grey values of the resulting pixels are transformed to spike rates in a way, that a maximum frequency of 80 Hz is not exceeded. Regarding two

adjacent pixels, one is linearly translated to the spike train of an excitatory stimulus, the other one to an inhibitory stimulus.
Until the time of thesis submission it was only possible to carry out the discrimination for the 0/1 and 2/3 classifiers, in order to compare both presented spike encoding protocols. The results of a training with 7000 steps and an exponentially decaying learn rate ($\tau_{learn} = 5000$) are illustrated in Figure 4.37 for the constrained learning and Figure 4.38 for the unconstrained one. In the case of the constrained learning the results show fewer oscillations compared to the previous spike coding scheme, but similar classification results around 90 % for the 0/1 classifier and around 60-90 % for the 2/3 classifier. After 1000 test runs using the software Tempotron, the classification yields (96.7 ± 0.6) % correctness for 0/1 and (74.5 ± 1.4) % for 2/3. A direct mapping of the software-trained weights onto the hardware LSM led to the following outcomes: (93.4 ± 0.9) % in the case of 0/1 and (73.0 ± 1.5) % in the case of 2/3.
Regarding the results of the unconstrained learning method in Figure 4.38 a distinct increase of the classification rate concerning the 2/3 task can be observed. The classification curve of the 0/1 task does not show any improvement compared to Figure 4.37, which is no surprise as the classification of 0/1 is in both cases already flawless.
The Figures 4.37 and 4.38 indicate that the second spike coding scheme might improve the classification rates slightly if weight flipping is allowed.

After training a complete set of Tempotrons necessary to distinguish the numbers from 0 to 3, an implementation of the handwritten digit recognition system on hardware might look as Figure 4.39 indicates. In the cast, the digits are translated with the second spike coding and applied directly to the liquid, which projects onto six Tempotrons, of whom the first Tempotron may spike if a 0 is presented and not spike if a 1 is presented, while responding arbitrarily if a 2 or 3 is to be classified. For each possible input (0-3) one gets an array of six numbers as a result, whereby the most frequent number is chosen as the classification outcome. If two or more numbers occur equally frequent the winner of the direct comparison is chosen. In the rare case that still no clear winner can be determined a random guess between the most frequent candidates is performed.
In a first classification trial of 4000 digits taken from the test set of the MNIST database [36] the described hardware recognition system produced a classification rate of (50.3 ± 0.8) %. This is significantly above the chance-level of 25 %. However, to further improve the results one should try increase the classification rate of the 0/2, 0/3 and 2/3 classifiers in particular, as they contribute the largest errors (see Table 4.4).
The bundling of the liquid and the Tempotron(s) on the same hardware substrate showed promising results despite the distortions mentioned in Section 4.2. The implemented learning can account for a software bug yielding an incorrect weight mapping, the STP enabled Tempotron connections and the incompletely calibrated chip. Another challenge apart from optimizing the learning, is to find more effective spike coding schemes providing the information in a way easier to access for the Tempotron. Suggestions concerning the optimization of the digit recognition can be found in the outlook of this thesis.

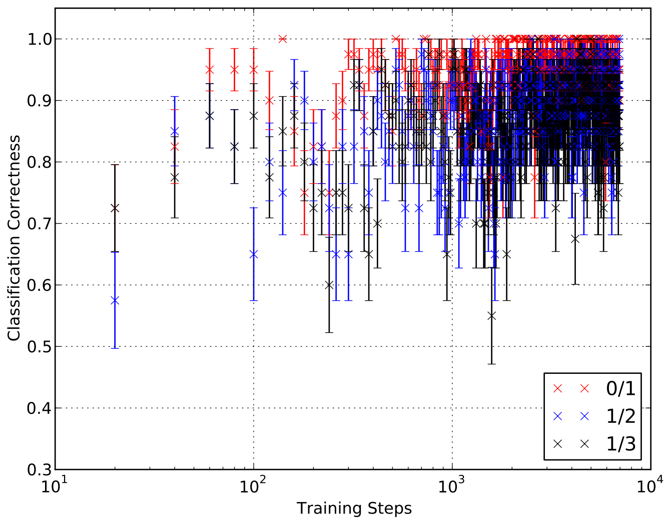

Figure 4.32: Classification curves of three easy tasks: the distinction of a *one* and another number, in the case of the constrained learning method (see Chapter 3). The software Tempotron performs with about 80-100 % confidence after 7000 training steps. Here, the first coding protocol is used (see Figure 4.29).

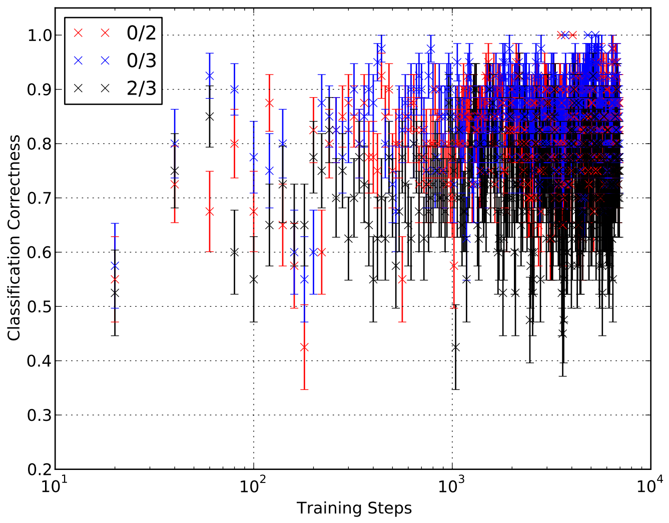

Figure 4.33: Classification curves of three difficult tasks, which are the distinctions of 0/2, 0/3 and 2/3 using the first coding protocol (see Figure 4.29). The 2/3 task is the most intricate one. In each case, one can observe large fluctuations due to uncertainty. The curves correspond to the constrained learning method (see Chapter 3).

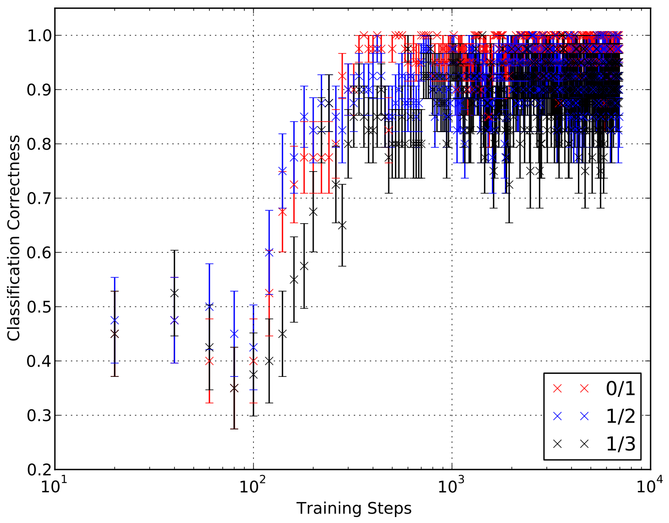

Figure 4.34: Classification curves containing 7000 training runs of three easy tasks, regarding the distinction of a *one* and another number, in the case of the unconstrained learning method (see Chapter 3). One can observe an improvement concerning fluctuations compared to Figure 4.32. Here, the first coding protocol is used (see Figure 4.29).

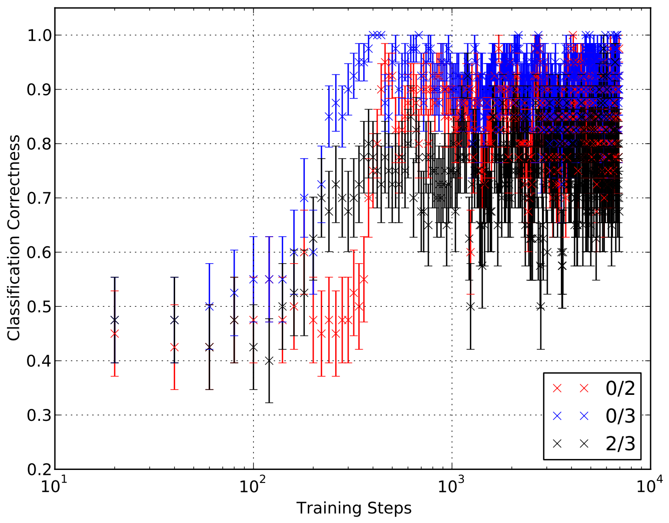

Figure 4.35: Classification curves containing 7000 training runs of three difficult tasks, which are the distinctions of 0/2, 0/3 and 2/3. The curves correspond to the training with COBA synapses using the unconstrained learning method (see Chapter 3). Nevertheless, the results still do not seem to improve significantly compared to Figure 4.33, but show less fluctuations. Here, the first coding protocol is used (see Figure 4.29).

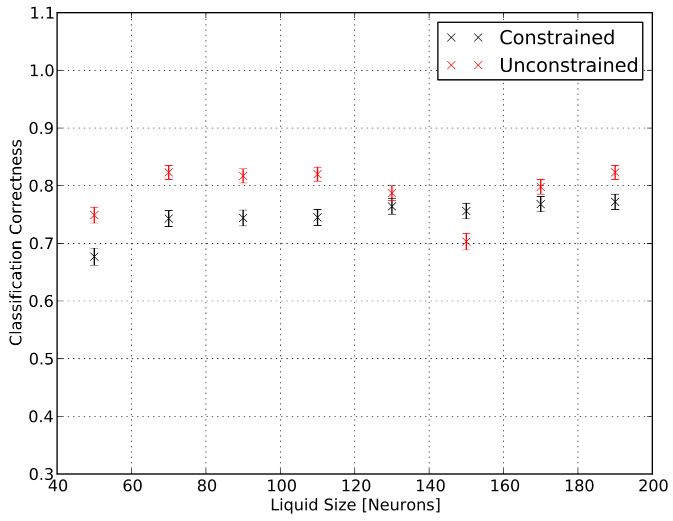

Figure 4.36: Results of classifying the 2/3 task as a function of the liquid size using the first coding protocol (see Figure 4.29). The measuring points are derived from a validation measurement of 1000 test samples. A larger hardware liquid size (within the limits of the FACETS chip-based system) does not provide any significant classification improvements. As one can observe, the unconstrained method does better than the constrained one.

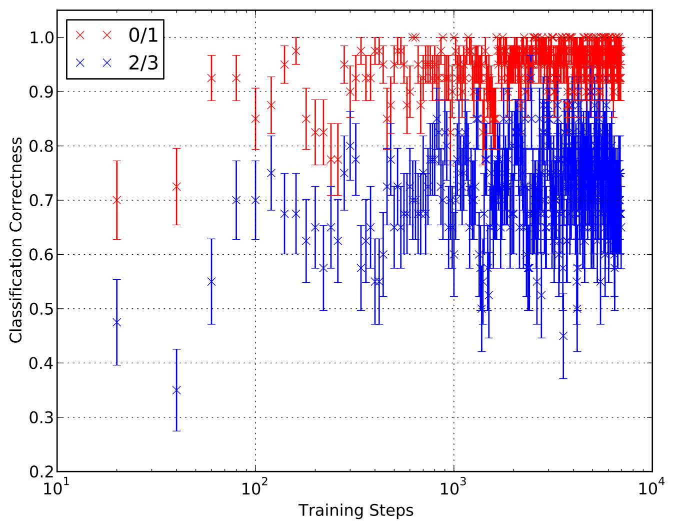

Figure 4.37: Classification curves containing 7000 training runs of classifying 0/1 and 2/3, while using the second coding protocol (see Figure 4.31). The curves correspond to the constrained learning method (see Chapter 3). The classification rate does not significantly improve compared to the results of the first coding scheme (see Figure 4.29).

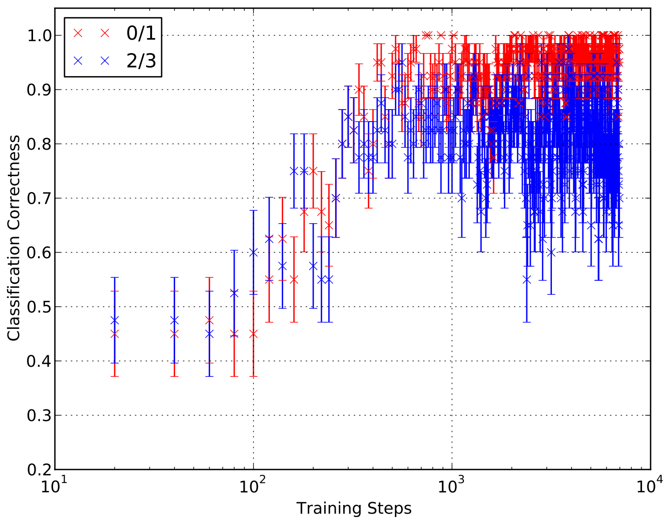

Figure 4.38: Classification curves containing 7000 training runs of classifying 0/1 and 2/3, while using the second coding protocol (see Figure 4.31). The curves correspond to the training with COBA synapses using the unconstrained learning method (see Chapter 3). Here, one can observe a slight increase of classification correctness in the 2/3 task compared to the previous figures.

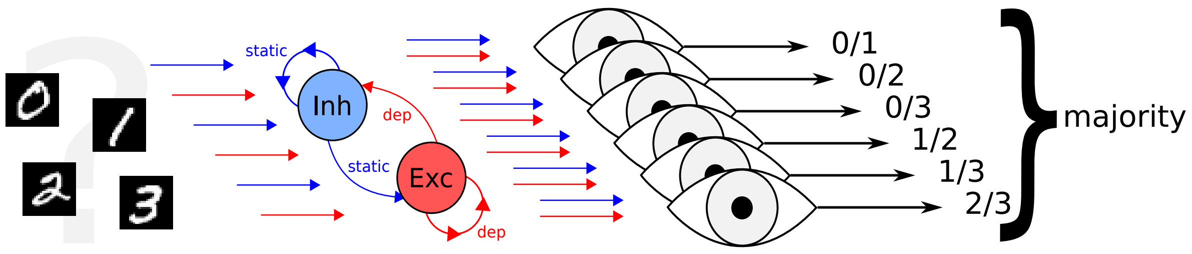

Figure 4.39: Resulting procedure to perform the digit recognition task. Arbitrary numbers are translated into a firing rate and presented to the Liquid State Machine. The LSM consists of the liquid and a population of six ($\frac{n\cdot(n-1)}{2}$ for n different digits) Tempotrons, which were trained on each specific digit recognition task. The most frequent number is returned.

5 Discussion

The results presented in this thesis have demonstrated the potential of the FACETS chip-based hardware system to be operated as a spike-based, universal classifier. The necessary steps carried out to reach this conclusion are briefly summarized in the following.
In the context of this thesis, a parameter study concerning the properties of a self-stabilizing liquid architecture has been accomplished, which demonstrated the stability of liquid under large parameter variations. An examination regarding the liquids size has pointed out that for simple tasks liquid sizes which are distinctly smaller than the dimension of the input population might still be able to provide memory and allow for close to perfect classification.
A bundling of a self-stabilizing liquid architecture together with the spike-based Tempotron classifier on the FACETS chip-based hardware system has been realized. To train the Tempotron's weights in software two hardware-adapted training methods have been implemented in order to account for the conductance-based synapse model. Therefore, it was possible to extend the dynamic input range to both, excitatory and inhibitory synapses. A direct mapping and stretching of the software weights to the available weight range on hardware showed classification results which are comparable to those of the corresponding software prototype. The refinement and enhancements in the available framework will simplify the application of the hardware system to future classification tasks.
In the further context of this thesis, a demanding task has been introduced which could be significantly better solved by a combined liquid-Tempotron setup. Thus, the ability of an LSM to enable the readout to perform beyond its own capacity could be proven. Even better results have been achieved by means of a software LSM which was allowed to change its synapse types from excitatory to inhibitory and vice versa. This system can tune itself to a more optimal weight distribution based on original Tempotron's learning rule. It is therefore expected that the synapse-driver-mirroring hardware feature or a future revision of the chip with both cores accessible can significantly improve any current classification rate. However, even without this feature the hardware LSM was able to discriminate the demanding task with more than 90 % confidence up to a level, which is by far beyond the sole Tempotron's characteristic scope.
Moreover, a real world handwritten digit recognition task has been employed on the hardware LSM. Common spike encoding protocols have been used to convert the digits into spike patters. The subsequent classification of four different digits on hardware yielded a correctness of (50.3 ± 0.8) %, which exceeds chance level (25 %) by far. Finally, the modifications proposed in the oncoming outlook are assumed to improve the classification in prospective experiments even further.

6 Outlook

This thesis has successfully demonstrated the application of the FACETS chip-based system in the field of pattern classification. In the following, possible feature extensions to the software framework and future parameter studies are proposed.
The most critical issue necessary for solve for future work on the hardware LSM is to find the software inconsistency responsible for an inconsistent mapping of the software weights onto hardware. This bug has led to too weak weights, which then needed to be stretched by a seemingly arbitrary but well chosen stretching factors in order to reach good classification performance on hardware. This bug has not yet been located, at least it doesn't seem to affect any other experiments performed on the hardware system. After solving the issue, future implementations of the LSM on hardware can neglect those factors.
Moreover, the training with current-based synapses according to the originally proposed Tempotron can be tuned to deliver more ideal weight distributions for the subsequent training with conductance-based synapses. This could significantly improve the time necessary to learn a given pattern. One attempt could be to choose lower weight maxima during the training with current-based synapses as Figure 4.10 suggests. Together with a more complete calibration of the chip, one can further improve the classification rates of the hardware LSM towards the results of the software prototype. Furthermore, a study concerning the impact of STP on the hardware Tempotron's classification ability has to be performed. This implies systematic experiments comparing the outcome of a software training with the cautious STP in place against runs without it. Access to the second half of the chip as well as synapse-driver-mirroring would both enable the hardware Tempotron population to operate without STP, while the liquid could still use it. The synapse-driver-mirroring also offers potential to increase the resolution of the discrete hardware weights, which could lead to more effective or more precise learning. Any issues related to the size of neuron populations will be gone, as soon as the FACETS wafer-scale hardware system is available. Larger liquids and higher Tempotron counts for the pairwise classification will provide an unseen liquid stability applicable in the area of neuroscience as well as industry.
In future applications of the digit recognition, a more efficient spike encoding protocol could be utilized, e.g. inter-spike time coding. Here, the grey values of the pixels determine the temporal distance between two consecutive spikes. This sort of patterns might be better suitable for the Tempotron, since its spatiotemporal structure is deterministic compared to patterns emerging from a Poisson process. However, this coding scheme does not provide extra information about the images, but merely presenting them differently. Thus, an improved classification is likely but not guaranteed. It remains to be seen if the hardware liquid's classification rate can be increased by larger liquids. A possible

approach to solve this question might be the study of the self-stabilizing liquid in pure software, here its size is only limited by the available computational power.

A Appendix

A.1 Learning Rules

Learning rule for a Tempotron with CUBA synapses:

```python
def _train(self, input, output, target, membrane, run):
  if synapse_model == 'CUBA':
    if output != target:
      for neuron_id, spike_time in input:
        if spike_time > membrane.t_max: break
        self.weights[neuron_id] += (target-output)*self.decayF(run, self.rate, self.tau)*membrane.kernel(neuron_id, membrane.t_max, spike_time)
    for neuron_id in xrange(len(self.weights)):
      if self.weights[neuron_id] > 0.003:
        self.weights[neuron_id] = 0.003
      if self.weights[neuron_id] < -0.003:
        self.weights[neuron_id] = -0.003
```

Leaning rule for a Tempotron with COBA synapses:

```python
def _train(self, input, output, target, membrane, run):
  if synapse_model == 'COBA':
    if output != target:
      for neuron_id, spike_time in input:
        if spike_time > membrane.t_max: break
        if self.weights[neuron_id] > 0.:
          self.weights[neuron_id] += (target-output)*self.decayF(run, self.rate, self.tau)*membrane.kernel(neuron_id, membrane.t_max, spike_time)
          if self.weights[neuron_id] < 0.:
            self.weights[neuron_id] = 0.
        elif self.weights[neuron_id] < 0.:
          self.weights[neuron_id] += (target-output)*self.decayF(run, self.rate, self.tau)*membrane.kernel(neuron_id, membrane.t_max, spike_time)*2.67 # additional factor
```

```
            if self.weights[neuron_id] > 0.:
                self.weights[neuron_id] = 0.
        for neuron_id in xrange(len(self.weights)):
            if self.weights[neuron_id] > 0.003:
                self.weights[neuron_id] = 0.003
            if self.weights[neuron_id] < -0.003:
                self.weights[neuron_id] = -0.003
```

A.2 Weight Initialization in Hardware

```
for i in xrange(len(weights)):
  if weights[i]<0:
    weights[i] *= pynn.maxInhWeight/0.003*p["inh"]
  else:
    weights[i] *= pynn.maxExcWeight/0.003*p["exc"]
```

A.3 Acronyms

API	Application Programming Interface
ARQ	Automatic Repeat reQuest
ASIC	Application Specific Integrated Circuit
CMOS	Complementary Metal Oxide Semiconductor
COBA	conductance-based
CUBA	current-based
EPSP	Excitatory Postsynaptic Potential
FACETS	Fast Analog Computing with Emergent Transient States
HICANN	High Input Count Analog Neural Network
IPSP	Inhibitory Postsynaptic Potential
LSM	Liquid State Machine
MNIST	Modified NIST
STDP	Spike-Timing Dependent Plasticity
STP	Short-Term Plasticity

SVM	Support Vector Machine
VLSI	Very-large-scale integration

Bibliography

[1] Electronic Vision(s) Group. `http://www.kip.uni-heidelberg.de/cms/groups/vision/home/`, 2010.

[2] FACETS. Fast Analog Computing with Emergent Transient States – project website. `http://www.facets-project.org`, 2010.

[3] BrainScaleS. BrainScaleS – project website. `http://brainscales.kip.uni-heidelberg.de/`, 2011.

[4] J. Schemmel, D. Brüderle, K. Meier, and B. Ostendorf. Modeling synaptic plasticity within networks of highly accelerated I&F neurons. In *Proceedings of the 2007 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 3367–3370. IEEE Press, 2007.

[5] A. P. Davison, D. Brüderle, J. Eppler, J. Kremkow, E. Muller, D. Pecevski, L. Perrinet, and P. Yger. PyNN: a common interface for neuronal network simulators. *Front. Neuroinform.*, 2(11), 2008.

[6] S. Jeltsch. Computing with transient states on a neuromorphic multi-chip environment. Diploma thesis, University of Heidelberg, HD-KIP 10-54, 2010.

[7] M. Tsodyks and H. Markram. The neural code between neocortical pyramidal neurons depends on neurotransmitter release probability. *Proceedings of the national academy of science USA*, 94:719–723, January 1997.

[8] S. Song, K. Miller, and L. Abbott. Competitive hebbian learning through spiketiming-dependent synaptic plasticity. *Nat. Neurosci.*, 3:919–926, 2000.

[9] Andreas Grübl. *VLSI Implementation of a Spiking Neural Network*. PhD thesis, Ruprecht-Karls-University, Heidelberg, 2007. Document No. HD-KIP 07-10.

[10] Daniel Brüderle. *Neuroscientific Modeling with a Mixed-Signal VLSI Hardware System*. PhD thesis, 2009.

[11] J. Schemmel, D. Brüderle, A. Grübl, M. Hock, K. Meier, and S. Millner. A wafer-scale neuromorphic hardware system for large-scale neural modeling. In *Proceedings of the 2010 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1947–1950, 2010.

[12] Daniel Brüderle, Mihai a Petrovici, Bernhard Vogginger, Matthias Ehrlich, Thomas Pfeil, Sebastian Millner, Andreas Grübl, Karsten Wendt, Eric Müller, Marc-Olivier Schwartz, Dan Husmann de Oliveira, Sebastian Jeltsch, Johannes Fieres, Moritz Schilling, Paul Müller, Oliver Breitwieser, Venelin Petkov, Lyle Muller, Andrew P Davison, Pradeep Krishnamurthy, Jens Kremkow, Mikael Lundqvist, Eilif Muller, Johannes Partzsch, Stefan Scholze, Lukas Zühl, Christian Mayr, Alain Destexhe, Markus Diesmann, Tobias C Potjans, Anders Lansner, René Schüffny, Johannes Schemmel, and Karlheinz Meier. A comprehensive workflow for general-purpose neural modeling with highly configurable neuromorphic hardware systems. *Biological cybernetics*, 104:263–296, May 2011.

[13] R. Brette and W. Gerstner. Adaptive exponential integrate-and-fire model as an effective description of neuronal activity. *J. Neurophysiol.*, 94:3637 – 3642, 2005.

[14] Python Software Foundation. The Python programming language - website. `http://www.python.org`, 2007.

[15] Markus Diesmann and Marc-Oliver Gewaltig. NEST: An environment for neural systems simulations. In Theo Plesser and Volker Macho, editors, *Forschung und wisschenschaftliches Rechnen, Beiträge zum Heinz-Billing-Preis 2001*, volume 58 of *GWDG-Bericht*, pages 43–70. Ges. für Wiss. Datenverarbeitung, Göttingen, 2002.

[16] M.L. Hines and N.T. Carnevale. *The NEURON simulation environment.*, pages 769–773. M.A. Arbib, 2003.

[17] Dejan A. Pecevski, Thomas Natschläger, and Klaus N. Schuch. Pcsim: A parallel simulation environment for neural circuits fully integrated with Python. *Front. Neuroinform.*, 3(11), 2009.

[18] Dan Goodman and Romain Brette. Brian: a simulator for spiking neural networks in Python. *Front. Neuroinform.*, 2(5), 2008.

[19] Bjarne Stroustrup. *The C++ Programming Language.* Addison Wesley Longman, Amsterdam, February 2000.

[20] Larry L. Peterson and Bruce S. Davie. *Computer Networks: A Systems Approach, 3rd Edition.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.

[21] Wulfram Gerstner and Werner Kistler. *Spiking Neuron Models: Single Neurons, Populations, Plasticity.* Cambridge University Press, 2002.

[22] Christoph Koch. *Biophysics of Computation: Information Processing in Single Neurons.* Oxford University Press, 1999.

[23] W. Maass, T. Natschläger, and H. Markram. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Computation*, 14(11):2531–2560, 2002.

[24] H. Jaeger. The “echo state” approach to analysing and training recurrent neural networks. Technical Report GMD Report 148, German National Research Center for Information Technology, 2001.

[25] Christopher M. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press Inc., New York, Walton Street, Oxford, 1995.

[26] Valentin Braitenberg and Almut Schüz. *Anatomy of the Cortex: Statistics and Geometry.* 1991.

[27] David Sussillo, Taro Toyoizumi, and Wolfgang Maass. Self-Tuning of Neural Circuits Through Short-Term Synaptic Plasticity. *J Neurophysiol*, 97(6):4079–4095, 2007.

[28] Johannes Bill. Self-stabilizing network architectures on a neuromorphic hardware system. Diploma thesis (English), University of Heidelberg, HD-KIP-08-44, 2008.

[29] F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–408, 1958.

[30] Albert B. Novikoff. On convergence proofs for perceptrons. In *Proceedings of the Symposium on the Mathematical Theory of Automata*, volume 12, pages 615–622, 1963.

[31] Robert Gütig and Haim Sompolinsky. The tempotron: a neuron that learns spike timing-based decisions. *Nat Neurosci*, 9(3):420–428, March 2006.

[32] T. Downarowicz. Law of series/poisson process. *Scholarpedia*, 3(11):3922, 2008.

[33] Marvin Albert. Liquid Computing mit Neuromorpher Hardware. Bachelor thesis (German), University of Heidelberg, HD-KIP 10-43, 2010.

[34] J. A. Benediktsson, J. Kittler, and F. Roli. *Multiple Classifier Systems.* Springer Berlin Heidelberg New York, 2009.

[35] Y. LeCun, L. D. Jackel, B. Boser, J.S. Denker, H. P. Graf, I. Guyon, D. Henderson, R.E. Howard, and W. Hubbard. Handwritten digit recognition: Applications of neural net chips and automatic learning. *IEEE Communications Magazine*, November 1989:41–46, 1989.

[36] Y. LeCun. The mnist database of handwritten digits.

Statement of Originality (Erklärung):

I certify that this thesis, and the research to which it refers, are the product of my own work. Any ideas or quotations from the work of other people, published or otherwise, are fully acknowledged in accordance with the standard referencing practices of the discipline.

Ich versichere, daß ich diese Arbeit selbständig verfaßt und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Heidelberg, July 12, 2011

....

(signature)