Department of Physics and Astronomy
University of Heidelberg

Master Thesis in Physics
submitted by

David Stöckel

born in Waiblingen (Germany)

November 2017

Exploring Collective Neural Dynamics under Synaptic Plasticity on Neuromorphic Hardware

This Master Thesis has been carried out by David Stöckel at the
Kirchhoff Institute in Heidelberg
under the supervision of
Prof. Karlheinz Meier

Abstract

In the human brain's resting state a stable baseline of irregular activity is observed. Facing changes in the neurons and synapses during learning and evolution these changes must be counterbalanced to maintain stability. This work proposes a long–term synaptic plasticity rule which stabilizes the activity in a feed–forward neural network regardless of the initial conditions. The rule is in particular favorable for large scale systems as it solely depends on the local pre– and postsynaptic spike time correlation. A diverse phenomenology of the neural dynamics is observed for different parameterizations of the plasticity. Applying this rule furthermore to recurrent networks the sensitivity and memory capacity of the network is shown to increase with the degree of recurrence. Last, the synaptic plasticity rule is shown to be capable of simple unsupervised pattern learning.

All measurements for this work are performed on the HICANN–DLS, a spiking, mixed–signal neuromorphic device. This work demonstrates the usability and flexibility of this system for bio–inspired neural network experiments.

Zusammenfassung

Im menschlichen Gehirn ist im Ruhezustand eine stabile Grundaktivität messbar. In seiner Entwicklung und beim Lernen verändern sich die Synapsen und Neuronen im Gehirn kontinuierlich. Diesen Prozessen muss entgegengewirkt werden um die Stabilität aufrecht zu erhalten. In dieser Arbeit wird eine Regel für synaptische Langzeitplastizität vorgestellt, welche in einem feed–forward Netzwerk die Aktivität stabilisiert. Dieser stabile Punkt wird unabhängig von den Anfangsbedingungen erreicht. Für die vorgestellte Plastizitätsregel wird eine vielfältige Phänomenologie in der Dynamik der Neuronen beobachtet je nach Parametrisierung der Regel. Werden darüber hinaus rekurrente Verbindungen erlaubt, zeigt sich, dass mit der zunehmenden Anzahl an rekurrenten Verbindungen die Sensibilität und das Erinnerungsvermögen des Netzwerkes zunehmen. Desweiteren wird demonstriert, dass diese Regel für unüberwachtes Lernen zur Mustererkennung verwendet werden kann.

Alle Messungen für diese Arbeit wurden auf dem HICANN–DLS durchgeführt, einem neuromorphen System, welches spikende neuronale Netzwerke in einer gemischt analog–digitalen Implementierung emuliert. Diese Arbeit demonstriert dessen Verwendbarkeit und Flexibilität für biologisch motivierte Experimente.

Contents

1. Introduction

Even when lying quietly with eyes closed, spontaneous activity is observed in the brain [Gusnard and Raichle, 2001]. Also, in–vitro slice preparations of cortical neurons show a self–sustained baseline of irregular firing [Plenz and Aertsen, 1996]. Maintaining such a balance of activity is not inherent to most models of neural networks and is likely to turn over to instability on changes in the synaptic connectivity [Abeles, 1991, Miller and MacKay, 1994, Abbott and Nelson, 2000, Brunel, 2000]. During learning and development however, neural circuits undergo perpetual changes in their number and strength of synapses. These possibly destabilizing influences are counterbalanced by homeostatic plasticity mechanisms, by means yet to be fully understood. Various different mechanisms of homeostatic plasticity are discussed and analyzed regarding the implications on network dynamics, especially stability [Miller and MacKay, 1994, Turrigiano, 1999, Abbott and Nelson, 2000, Sussillo et al., 2007].

To study the dynamics of spiking neural networks, software simulations have become an important tool besides analytical methods [Brette et al., 2007]. An alternative approach is the implementation of neuron and synapse models as physical units in electrical circuitry [Mead, 1990, Douglas et al., 1995]. Since all units in these neuromorphic systems operate in parallel, the speed of computation is largely independent of the system size. Several groups have made progress in this field [Indiveri et al., 2006, Merolla and Boahen, 2006, Vogelstein et al., 2007, Schemmel et al., 2008, Mitra et al., 2009]. In case of the neuromorphic Digital Learning System (DLS)[1] described by Friedmann et al. [2017] a mixed–signal approach of analog neuron and synapse circuits coupled to a digital configuration and communication interface is pursued. The system operates ≈ 1000 times faster than comparable, biological plausible networks and is designed to be scalable to very large system sizes. To provide flexibility in the implementation of different learning tasks, a hybrid approach is used: an embedded general purpose processor takes care of plasticity in parallel to the analog circuits emulating the neural network dynamics.

The aim of this work is to find long–term plasticity rules that stabilize the neural activity by exploiting the flexibility and emulation speed of the DLS. As one goal is to generate a stable baseline of irregular firing in future large scale systems, the plasticity rules are restricted to depend only on measures local to the synapse. This is done with respect to the scalability of the experiment: if the plasticity depends on observables not local to the individual synapse—e.g. the sum of all incoming synaptic weights for the postsynaptic neuron—the communication effort in large systems quickly becomes infeasible.

[1] The full name is High Input Count Analog Neural Network with Digital Learning System (HICANN-DLS). In this work it will however always be abbreviated with DLS.

1.1. Candidates for Stabilizing Synaptic Plasticity

Synaptic plasticity is a fundamental property of the central nervous system and likely to be involved in learning, memory, and other cognitive processes [Hebb, 1949, Frankland et al., 2001, Carcea and Froemke, 2013]. Sussillo et al. [2007] show that short–term synaptic plasticity can endow a network of leaky integrate–and–fire neurons with a "remarkable stability". On each spike transmitted by the synapse the synaptic strength is increased (known as short–term potentiation) or decreased (short–term depression) which is compatible with biological measurements [Abbott and Nelson, 2002]. As indicated by the name, this happens on the short time–scale of single spikes. This work however focuses on long–term plasticity where changes of the synaptic plasticity happen on the timescale of hundreds of spikes.

An important form of long–term synaptic plasticity is Spike–Timing–Dependent Plasticity (STDP), which describes the gradual change of the synaptic weight depending on the precise pre– and postsynaptic spike times [Markram et al., 1997, Bi and Poo, 1998]. A diverse phenomenology of STDP is found for different synapses, brain regions and species, reviewed by Markram et al. [2012]. This form of plasticity is in particular appealing for large scale systems as it only depends on the local information of pre– and postsynaptic spike times.

1.2. This Work

In section 2.1 the neural network model is presented and a synaptic plasticity rule proposed, where the synaptic weights are subject to three processes: decay, stochasticity and STDP. The implementation of this model on the neuromorphic system is discussed in section 2.2 while the inevitable software framework for performing neuromorphic experiments on the DLS is presented in section 2.3.

The proposed plasticity algorithm is shown to be realizable on the system (section 3.1) and to generate stable, moderate firing activity of the neurons in a feed–forward network under random spike input (section 3.2). Having this feed–forward network with stabilizing synaptic plasticity on–hand, this work further discusses the dynamics under an increasing degree of recurrence in section 3.3. When sweeping large ranges of the plasticity algorithm parameterization, the plasticity rule is shown to not only generate stable activity but also exploding, dying and diverging firing rates among the different neurons, shown in section 3.4. Last, the most thrilling question is addressed in section 3.5: is it possible to use the presented plasticity rule to learn simple patterns?

The discussion and conclusion on the presented findings, the usability of the neuromorphic system and the relevance of this work regarding biology can be found in chapter 4.

2. Neural Network Model and Neuromorphic Implementation

A neural network can be seen as an agent based complex system where the neurons are the agents and the synapses modulate the interaction which is schematically depicted in figure 2.1. To analyze the system dynamics of a small neural network under synaptic plasticity one of the most simple models of neurons and synapses is chosen, described in section 2.1. The model was chosen to be simple not only because of the hardware constraints but was also also further restricted to more clearly relate changes in the observed dynamics to changes in the models' parameters. The neuromorphic implementation of this model on the DLS is described in section 2.2.

2.1. Model Description

This section covers the description of the theoretical model: Leaky Integrate–and–Fire (LIF) neurons with current–based synapses modulated by a particular form of long–term synaptic plasticity.

2.1.1. Leaky Integrate-and-Fire Neuron Model

The studied neuron model is a LIF neuron [Gerstner and Kistler, 2002] whose dynamics are governed by the Ordinary Differential Equation (ODE) of membrane potential $u(t)$,

$$\tau_{\text{mem}} \frac{\mathrm{d}u(t)}{\mathrm{d}t} = -\left[u(t) - u_{\text{leak}}\right] + \frac{I(t)}{g}, \tag{2.1}$$

and its refractory condition

$$\begin{aligned} &\text{if} \quad u\left(t_{\text{spike}}\right) \geq u_{\text{thresh}} \\ &\text{then} \quad u\left(t\right) = u_{\text{reset}} \quad \text{for} \quad t \in \left[t_{\text{spike}}, t_{\text{spike}} + \tau_{\text{ref}}\right] . \end{aligned} \tag{2.2}$$

The parameters are named and explained in the following. In the absence of any current input $I(t)$ the membrane potential decays exponentially towards the neuron's leak potential u_{leak} with the time constant τ_{mem}. The leak conductance g scales the strength of the current input. If the membrane potential hits the threshold potential u_{thresh} at the time t_{spike} the neuron emits a spike and its potential is clamped to the reset potential u_{reset} for the refractory period τ_{ref}. The spikes can be seen as pulses which transmit information over the synapses to the connected neurons as explained in the next section. While the evolution of the membrane

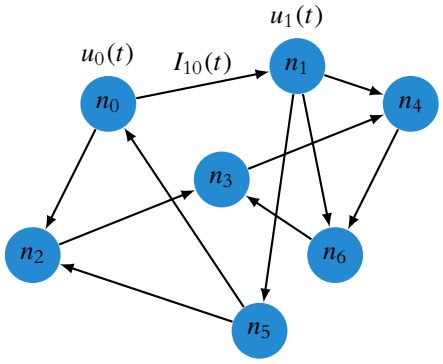

Figure 2.1.: Abstract model of a neural network. The circles represent point–like neurons, the arrows represent the directed interaction transmitted by the synapses. The neurons' state variables $u_i(t)$ are described in the section 2.1.1 and the synaptic currents $I_{ij}(t)$ in section 2.1.2.

potential in equation (2.1) is a first order linear differential equation the reset condition in equation (2.2) adds a nonlinearity. This nonlinearity is a necessary condition to allow for emergent collective phenomena in the neural network [Scott, 2007].

In the following each neuron is assigned an index. The membrane potential and input current of the i-th neuron are referred to as $u_i(t)$ and $I_i(t)$.

2.1.2. Current Based Synapse Model

The synapse model implements a directed pairwise interaction between the neurons. In figure 2.1 this interaction is represented by the black arrows. The source of the directed interaction is called the presynaptic neuron, the target is called the postsynaptic neuron.

The synaptic interaction is governed by the ODE

$$\tau_{\text{syn}} \frac{\mathrm{d}I_{ij}(t)}{\mathrm{d}t} = -I_{ij}(t) + \sum_{t_{\text{spike},j}} w_{ij} \cdot \delta\left(t - t_{\text{spike},j}\right) . \tag{2.3}$$

For any spike $t_{\text{spike},j}$ of neuron j the synaptic current I_{ij} onto neuron i is increased by the synaptic weight w_{ij}. If there are no spikes, the synaptic current decays with the synaptic time constant τ_{syn}. Equation (2.3) therefore models current–based synapses with an exponential kernel. All synaptic currents towards the i-th neuron enter the neuron dynamics in equation (2.1) as the input current $I_i(t)$ which is a linear sum:

$$I_i(t) = \sum_j I_{ij}(t) . \tag{2.4}$$

A synapse with a positive synaptic weight is called "excitatory", with a negative weight "inhibitory". The synaptic time constant τ_{syn} in general may vary from synapse to synapse.

Throughout this work the time constants of all excitatory synapses is a global constant referred to as $\tau_{\text{syn,exc}}$. To keep the model as simple as possible, there are no inhibitory synapses in any of the experiments.

2.1.3. Long–Term Synaptic Plasticity

In the investigated neural network synaptic weights may change over time. First, stating the synaptic plasticity equation, each of the terms will be motivated and explained in the following. The used model is STDP [Markram et al., 1997, Bi and Poo, 1998] together with decay, bias and stochasticity where the weights are restricted to the positive range.

$$\frac{\mathrm{d}w_{ij}(t)}{\mathrm{d}t} = -\frac{w_{ij}(t) - w_0}{\tau_{\text{decay}}} + \sigma\frac{\mathrm{d}W_t}{\mathrm{d}t} + \sum_{\text{spike pairs}} K\left(t_{\text{post}} - t_{\text{pre}}\right) \cdot \delta\left(t - \max\left(t_{\text{pre}}, t_{\text{post}}\right)\right) . \quad (2.5)$$

Negative derivatives at $w_{ij} = 0$ are clipped to zero. The constant τ_{decay} is the decay time constant, w_0 the steady state weight, σ the diffusion constant, W_t a Wiener process and $K(\Delta t)$ the STDP kernel function.

For positive τ_{decay} and σ, the differential equation

$$\mathrm{d}w_{ij}(t) = -\frac{w_{ij}(t) - w_0}{\tau_{\text{decay}}}\mathrm{d}t + \sigma\mathrm{d}W_t \quad (2.6)$$

models an Ornstein–Uhlenbeck (OU) process on the synaptic weights: a random walk of the weights, biased towards the steady state solution w_0 [Uhlenbeck and Ornstein, 1930]. For an ensemble of weights starting at time $t = 0$ with w_{init} the weights would sample after time t from a normal distribution:

$$w_{ij}(t) \sim \mathcal{N}\left(\mu_{\text{ou}}, \sigma_{\text{ou}}\right) \quad (2.7)$$

with

$$\mu_{\text{ou}} = w_{\text{init}} e^{-\frac{t}{\tau_{\text{decay}}}} + w_0\left(1 - e^{-\frac{t}{\tau_{\text{decay}}}}\right), \quad (2.8)$$

$$\sigma_{\text{ou}} = \frac{\sigma^2\tau_{\text{decay}}}{2}\left(1 - e^{-\frac{2t}{\tau_{\text{decay}}}}\right) . \quad (2.9)$$

The larger the diffusion σ and the time constant τ_{decay}, the broader the steady state distribution of an ensemble of weights becomes. The average over an ensemble of weights decays towards the steady state solution w_0 exponentially over time with the time constant τ_{decay}. Having this stochastic term allowing for exploration of the weight–space was inspired by the theory of synaptic sampling [Kappel et al., 2014] as well as for practical reasons to overcome precision issues when calculating in 8 bit arithmetic on the Plasticity Processing Unit (PPU), as explained in section section 2.2.4.

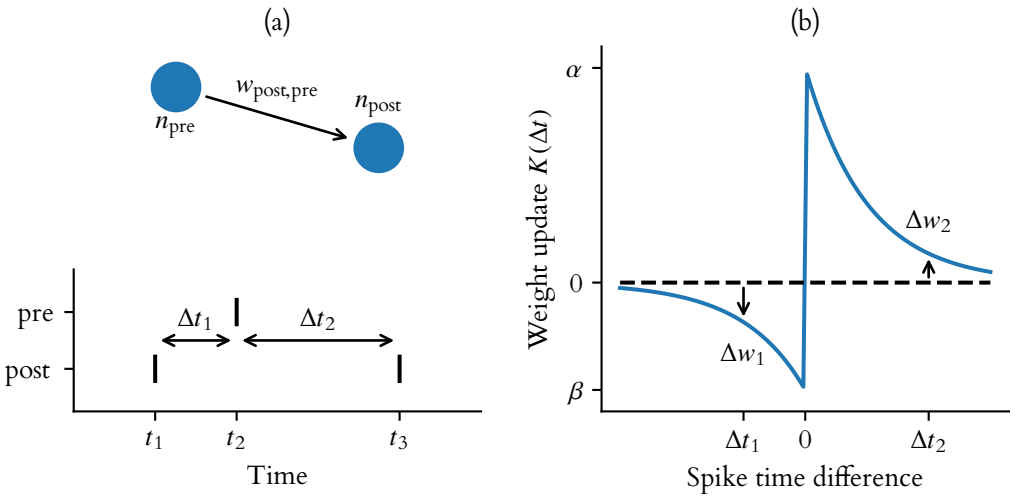

Figure 2.2.: (a) Sketch of two neurons which are connected with a synapse under the influence of synaptic plasticity. The presynaptic neuron spikes once in this example, the postsynaptic neurons twice shown at the bottom. (b) Change of the weight $w_{\text{post,pre}}$ due to STDP with an exponential kernel function. The post–pre pairing with Δt_1 yields a negative update Δw_1 for the weight, the pre–post pairing with Δt_2 a positive difference Δw_2.

The last term in equation (2.5) adds the dependency on the pre–post spike times: it is a sum over all nearest neighbor spike pairs t_{pre}, t_{post}. At the later spike of each spike pair the synaptic weight is changed by the kernel $K(t_{\text{post}} - t_{\text{pre}})$ as sketched in figure 2.2.

There are different motivations to investigate this form of synaptic plasticity of which all will be discussed in the following.

- Stability of the weights
- Robustness to different initial conditions
- Constraints of the neuromorphic computing platform
- Overcome limits of the low precision arithmetic

The stability of the ODE of the weights comes with the OU process which is one of the simplest forms to have a stochastic process with a stable fixed point. In particular, the force towards w_0 grows ever larger for large $|w_0 - w|$. The further the weights are away from the fixed point, the larger the force will be. Therefore, the OU process is also robust against different initial conditions. For any initial weights the fixed point w_0 is globally attractive for positive τ_{decay} and σ. Especially with $w_0 > 0$ an initially disconnected network ($w = 0$ for all synaptic weights) will over time establish synaptic connections. In this work only small

networks are looked into where all–to–all connectivity is a realistic scenario. For large networks one could introduce a dependence on the distance to penalize all–to–all connectivity on the scale of the whole system.

There are furthermore constraints on the plasticity imposed by the neuromorphic platform as only a limited set of observables is available to the PPU. Every synapse on the DLS has a correlation sensor which accumulates exponentially weighted pre-post spike time differences. The accumulation for positive and negative time differences is done with separate accumulation values, such that the most general form of a kernel—using the correlation sensors—is

$$K(\Delta t) = \begin{cases} \alpha \exp\left(-\frac{\Delta t}{\tau_{\text{stdp}}}\right) & \text{if } \Delta t > 0 \\ \beta \exp\left(\frac{\Delta t}{\tau_{\text{stdp}}}\right) & \text{if } \Delta t \leq 0 \end{cases} . \tag{2.10}$$

which is also sketched in figure 2.2. The implemented correlation sensors are discussed in section 2.2.3 in detail. Using these correlations sensors is highly appealing as they asynchronously measure the pre–post correlations without using the digital on–chip computing resources. This work will therefore only look into kernels of the form described in equation (2.10).

As mentioned before, the stochastic term was also introduced to overcome precision issues when calculation the plasticity rule with fixed point arithmetic. To allow for weak updates it may be desirable to have an update which is less than the precision with which the synaptic weight is implemented. Here, the approach is chosen to calculate the weight difference with a higher precision and then to round the result randomly up or down. This stochastic rounding is biased by the non–significant digits to yield the more precise weight difference on average.

2.2. The DLS Neuromorphic Computing Platform

The DLS neuromorphic chip shown in figure 2.3a is a prototype[1] system with 32 neurons and an array of 32×32 synapses [Friedmann et al., 2017]. The implemented time constants are much smaller than their biological counterparts, which leads to an emulation speed–up of $\approx 10^3$ compared to the biological domain. A general purpose processor with vector extensions—the Plasticity Processing Unit (PPU)—has not only read and write access to the complete chip configuration, but can also read the synaptic correlation data and neuron–wise spike counters.

External control as well as spike input and recording is provided by a Xilinx Spartan-6 Field Programmable Gate Array (FPGA) connected to a host computer via USB-2.0 running at 96 MHz. Similar to the PPU the external FPGA has read and write access to the complete chip configuration. The FPGA performs the chip configuration, experiment execution and spike recording. The configuration and spike input data is stored in 512 MiB DDR3-SDRAM and played back by the FPGA using cycle-accurate timing. As the FPGA and the internal chip logic both run at a clock frequency of 96 MHz, the best-case temporal

[1] The used prototype system is internally referred to as DLSv2.

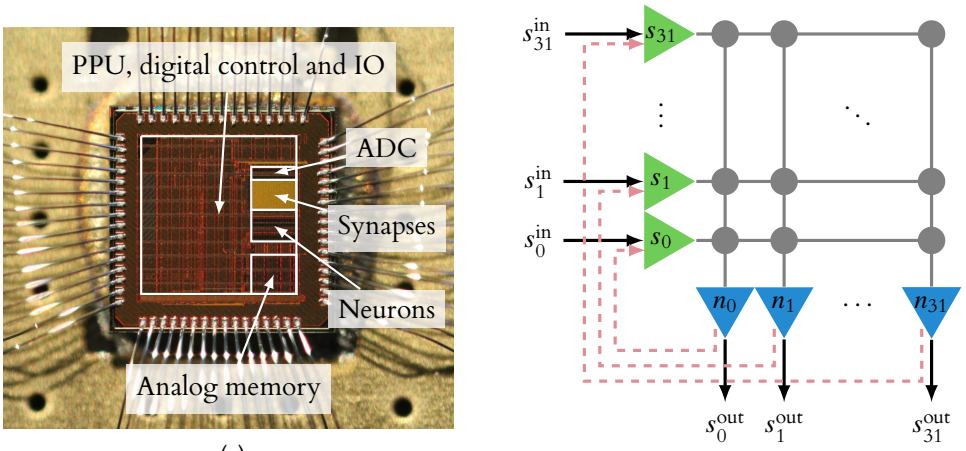

Figure 2.3.: (a) Photograph of the DLS v2 prototype chip with the layout of the functional areas highlighted. (b) Schematic view of the spike sources (green triangles), neurons (blue triangles) and synapses (grey circles) on the chip. The spike trains s^{in} and s^{out} denote the spikes sent in and recorded back from the chip by the Field Programmable Gate Array (FPGA).

precision is 10.4 ns, which equals 10 µs in biological time. Throughout this thesis, the acceleration factor is defined to be 960 which is a sensible choice to simplify the conversion between FPGA cycles measuring the wall–clock time and the biological domain:

$$t_{\text{bio}} = 960 \cdot t_{\text{wall}} \quad \Leftrightarrow \quad t_{\text{bio}} = n_{\text{cycles}} \cdot 1 \times 10^{-5}\,\text{s} \tag{2.11}$$

Measurements and time constants in the following are always given in biological time.

2.2.1. Neurons

The neurons on the prototype system are built around a 2 bit switchable membrane capacitor C_{mem} with a maximum capacitance of 2.4 pF [Aamir et al., 2016]. The leak conductance g is realized as an Operational Transconductance Amplifier (OTA) with negative feedback that pulls the membrane towards a configurable leak potential u_{leak}. Leak conductance and membrane capacity are related to the membrane time constant in the LIF equation (2.1) by

$$\tau_{\text{mem}} = \frac{C_{\text{mem}}}{g}\,. \tag{2.12}$$

Each neuron receives the accumulated input of 32 synapses via two current based synaptic inputs—one for excitatory and one for inhibitory events. Both of these inputs feature RC-integrators with individually tunable resistances which are used to generate exponentially decaying synaptic currents with adjustable synaptic time constants. Therefore, all excitatory incoming connections of one neuron share the same time constant, as well as all inhibitory connections.

Parameter	Symbol	Value
Time constant	τ_{mem}	4.8 ms
Leak potential	u_{leak}	800 mV
Reset potential	u_{reset}	600 mV
Threshold potential	u_{thresh}	1.1 V
Refractory period	τ_{ref}	4.8 ms
Excitatory synaptic time constant	$\tau_{\text{syn,exc}}$	1.9 ms
STDP time constant	τ_{stdp}	5.3 ms
Causal STDP amplitude	η_+	19 lsb

Table 2.1.: LIF parameters used for all neurons. Time constants are given in the biological time domain while voltages are not translated to biological plausible ranges.

Spikes are detected by a comparator that triggers digital events as well as the reset circuitry for the membrane. The latter allows for adjustable refractory times by implementing an analog delay cell.

All analog parameters are supplied using 17 integrated Digital to Analog Converters (DACs) per neuron—14 for currents and 3 for voltages. The reset potential is set globally for all neurons via a shared DAC. These parameters can be adjusted with a precision of 10 bit from the PPU as well as externally from FPGA. Further details can be found in [Aamir et al., 2016].

Even if all 32 neurons are configured with the same set of parameters their behavior will vary. Each neuron is implemented as a separate circuit with fixed–pattern noise on the transistors due to manufacturing variations. Furthermore, there are trial–to–trial variations due to variations in the reconfiguration of the analog parameter storage. The trials–to–trial variations measured by Hock [2014] are much less than the fixed pattern noise characterized by Stradmann [2016]. Along with the characterization a calibration database was set up to allow for an individual mapping from the LIF parameters to a set of analog parameters for each neuron. The parameters used for this work are show in table 2.1. The voltage parameters u_{leak}, u_{reset} and u_{thresh} are chosen such that the linear range of the leak term and the available dynamic range of the chip is well used. Furthermore the biological plausible order of $u_{\text{reset}} < u_{\text{leak}} < u_{\text{thresh}}$ is used. For the time constants the order $\tau_{\text{syn}} < \tau_{\text{mem}} = \tau_{\text{ref}} \approx \tau_{\text{stdp}}$ is chosen. The membrane time constants was chosen to be as large as possible, but still reliably configurable on the DLS as for this system holds: the larger the time constant the less reliable the calibration [Stradmann, 2016]. Then, the STDP time constant should be similar to the larger of the membrane and synaptic time constant in order to give a meaningful measure on how much a presynaptic spike contributed to a postsynaptic spike. Last, the synaptic time–constant was chosen to be smaller than the membrane time constant and refractory time constant, such that the memory in the synaptic currents has decayed when the refractory period is over.

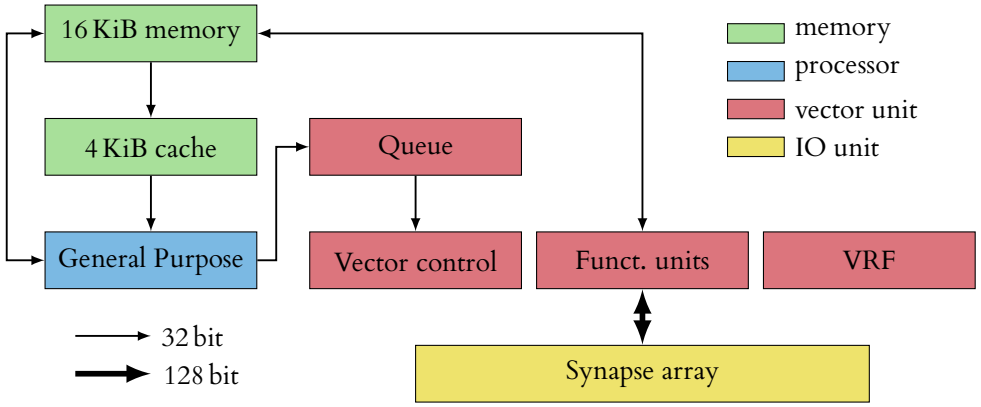

Figure 2.4.: Schematic of the PPU which is part of the plasticity sub-system and computes weight updates. It consists of a general–purpose part implementing the Power Instruction Set Architecture (ISA) and a vector unit to accelerate computations. The processor has access to 16 KiB of on-chip memory and uses a 4 KiB instruction cache. The functional units of the vector extension operate on 128 bit vectors of which 32 can be stored in the Vector Register File (VRF).

2.2.2. Synapses

The synapses are arranged in a 2-dimensional array sketched in figure 2.3b. The neurons are located along the bottom line whereas the presynaptic spikes enter the array from the left side. For each row the presynaptic signal is transmitted to all synapses in the row. Analog current pulses are generated for all active synapses and transmitted along the columns to the neurons where they are converted to synaptic currents. These pulses are scaled in height by the synaptic weights which can be configured individually for each of the 1024 synapses. The lengths of the pulses are configured globally and therefore allow for scaling all weights simultaneously. The weight has a precision of 6 bit and therefore ranges from 0 lsb to 63 lsb. The unit Least Significant Bit (lsb) will be used in the following for all digitally stored or processed integers. Each row determines whether the pulses of these synapses are send to the excitatory or inhibitory synaptic input of the neuron. In other words, all synapses along a row are of the same type: either excitatory or inhibitory.

To allow for multiplexing postsynaptic spikes from different neurons onto the same synapse row a 6 bit source address may be given to each of the spikes. Within the synaptic array a 6 bit decoder address is stored for each synapse in addition to the 6 bit synaptic weight. The synapse transmits the spike only if the source address matches the internal decoder address.

2.2.3. The Plasticity Processing Unit

Calculating new weights and actually updating the weights during operation is done by the PPU, schematically shown in figure 2.4. This is a general-purpose processor extended with a functional unit specialized for parallel processing of the synapses. It has access to 16 KiB

bit	7	6	5	4	3	2	1	0
value	-1	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}
weight	0	w_5	w_4	w_3	w_2	w_1	w_0	0

Table 2.2.: Bit representation of the weights in fractional saturation arithmetic.

of main memory with an instruction cache of 4 KiB. The special–purpose part implements an instruction set extension for vector–wise processing of synaptic properties. It operates on 128 bit wide vectors with a Vector Register File (VRF) that can store 32 of these vectors.[2]

The vector extension is organized as a weakly–coupled coprocessor. Upon encountering vector instructions the general-purpose part sends them to a queue. While the general–purpose part continues with the execution of the following instructions, this queue of vector instructions is processed in at the same time. The vector unit takes instructions in order from this queue, decodes them and distributes them to the appropriate functional units. These functional units provide operations for different precision of the vector components—8×16 bit and 16×8 bit vectors. For each of those precisions, the components may be treated with integer modulo arithmetic or fractional saturation arithmetic:

- **Integer modulo arithmetic:** The components are unsigned integers within $[0, 2^n)$ or signed integers within $[-2^{n-1}, 2^{n-1})$ where n is the number of bits for each component. Results of the arithmetic operations are stored modulo 2^n (i.e. $255+1 = 0$ in 8 bit signed integer modulo arithmetic).

- **Fractional saturation arithmetic:** The components are signed rational numbers within $[-1, 1)$ with a resolution of 2^{-n+1} where n is the number of bits for one component. A signed 8 bit integer m reinterpreted in the fractional representation maps to $m/128$. Operations resulting in values out of the valid range would saturate at -1 or $1 - 2^{-n+1}$ (i.e.: $127/128 + 1/128 = 127/128$ in 8 bit fractional saturation arithmetic).

There are two different load–store units: a 32 bit wide access to the main memory and a parallel 128 bit bus to access the synapses and the Analog to Digital Converter (ADC). The synaptic weights on the prototype chip are stored as 6 bit weights which are aligned in memory to 8 bit by filling up the two most significant bits with zeros. Therefore, the operation set for vectors of 8 bit elements is used for the synaptic plasticity algorithm. This is in particular convenient, as the ADC also digitizes the accumulation traces of spike correlations introduced below to 8 bit values. The fractional saturation arithmetic is used for all calculations as it is preferred compared to modulo arithmetics: if an update of 1 lsb is calculated to a weight which already is at its largest value, this weight stays at the current value instead of wrapping around to zero. To make use of this saturation the 6 bit weights need to be aligned to use the range from 0 to 127/128 in fractional representation. This is realized by shifting the weights to the left by one, to align the weights' most significant bit w_5 with the 6^{th} bit as shown in table 2.2. Here, w_i are the individual bits of the weight with w_5 being the most

[2]Registers are variable storages to which the compute units in the processor have a fast access.

significant bit. This bit–shift is applied after each load and before each store operation of the synaptic weights. Using this scheme, the largest weight of 63 lsb maps to 126/128 and on the reverse, the largest fractional number 127/128 maps to 63 lsb.

The correlation measurement on the neuromorphic hardware system is realized by analog circuits [Friedmann et al., 2017]. For every synapse the correlation data is stored in form of two accumulation traces:

$$a_+ = \sum_{t_{pre} > t_{post}} \eta_+ \exp\left(-\frac{t_{pre} - t_{post}}{\tau_{stdp}}\right), \tag{2.13}$$

$$a_- = \sum_{t_{pre} < t_{post}} \eta_- \exp\left(\frac{t_{pre} - t_{post}}{\tau_{stdp}}\right), \tag{2.14}$$

with the analog accumulation rates $\eta_\pm$. The summed–up pairs (t_{pre}, t_{post}) are selected according to a reduced symmetric nearest neighbor pairing rule defined in [Morrison et al., 2008]. The accumulation traces $a_\pm$ are digitized by an ADC. A characterization of the correlation measurements and their digitization is given in [Wunderlich, 2016].

2.2.4. Implementation of the Synaptic Plasticity Algorithm

To approximate the learning rules defined by equation (2.5) the vector unit of the PPU performs discrete updates

$$w'_{ij} = \max\left\{\frac{1}{2}\left[2w_{ij} + \frac{1}{4}\left(2w_{ij}\lambda_{decay} + \frac{a_+}{2}\lambda_{stdp} + n_{ij}\right)\right], 0\right\} \tag{2.15}$$

at every time step T, where the update rule can be parameterized by the 8 bit signed fractional factors λ_{decay} and λ_{stdp}. The term n_{ij} represents biased random numbers which are supposed to approximate the Wiener process. They furthermore implement stochastic rounding of the decay and STDP update when cropping the update to the significant bits of the weights. In appendix B the implementation of the update equation with in–line assembly embedded in C is listed. Some important notes on equation (2.15):

- Divisions are realized by bit shifts and therefore the results are biased towards lower values as all results are rounded to the next lower integer. A division by 8 for example yields

$$\frac{1}{8} = 0, \quad \frac{2}{8} = 0, \quad \cdots, \quad \frac{7}{8} = 0, \quad \frac{8}{8} = 1\,. \tag{2.16}$$

- Saturation may happen at any of the multiplications and additions.
- The factor of 2 attached to the weight accounts for the bit shift by one to match the fractional representation as shown in table 2.2.
- The correlation measurement a_+ has 8 significant bits. With the division by 2 the range is scaled to $[0, 128]$ such that the sign bit in the fractional representation is always cleared.

- The update term in the round brackets is calculated with three non significant bits compared to the weights (i.e. $w' = w + \frac{1}{8}\Delta w$). These non-significant bits are taken into account by the stochastic rounding. Therefore, the smallest change in the average update is $1/8$ lsb in units of the weights least significant bit.

The decay factor translates to the time constant τ_{decay} with

$$\tau_{\text{decay}} = \frac{4T}{\lambda_{\text{decay}}}, \tag{2.17}$$

where T is the update cycle. The fixed point of the plasticity algorithm without pairwise correlations ($\lambda_{\text{stdp}} = 0$) is given by

$$w_0 = -\frac{\langle n_{ij} \rangle - 4}{2\lambda_{\text{decay}}}. \tag{2.18}$$

The -4 in the enumerator accounts for the biased rounding of the update as noted before.

The general–purpose part of the PPU executes the update loop of the synaptic weights. The synapses are updated in row–major order of the synapse array. As 16 synapses are processed in parallel, 64 iterations are necessary to update all synapses of the chip. To provide control over the update cycle T of the synaptic weights the PPU is triggered with precise timing by the FPGA. The general–purpose part waits for the trigger signal until the next update loop is executed. This synchronizes the update cycle of the synaptic weights with the timing of playback and recording of spikes.

2.3. The Firmware and Software Framework

The software on three different systems is involved in order to perform the presented experiments. First, the `frickel-dls` program library on the host computer for the chip configuration and experiment description. Second, the FPGA firmware which handles the communication with the chip and the precisely timed experiment control. The last system involved is the on-chip PPU which is programmed for the continuous reconfiguration, in particular, implementing synaptic plasticity. The host and the PPU software are explained in more detail in the following two sections.

2.3.1. The Host Software

The host software `frickel-dls` is a C++ library with a Python interface named `pydlsnew`. Every configurable module on the chip and the baseboard is abstracted as a container holding the module's configuration and a coordinate which addresses this component. For example the `Neuron` class holds the digital configuration for one neuron while the `Neuron_index` addresses one of the 32 implemented neurons on the chip in a type–safe manner. These containers are nested such that the `Setup` configuration itself is a hierarchical container containing all configurable parts of a physical setup in the laboratory. The hierarchy of the

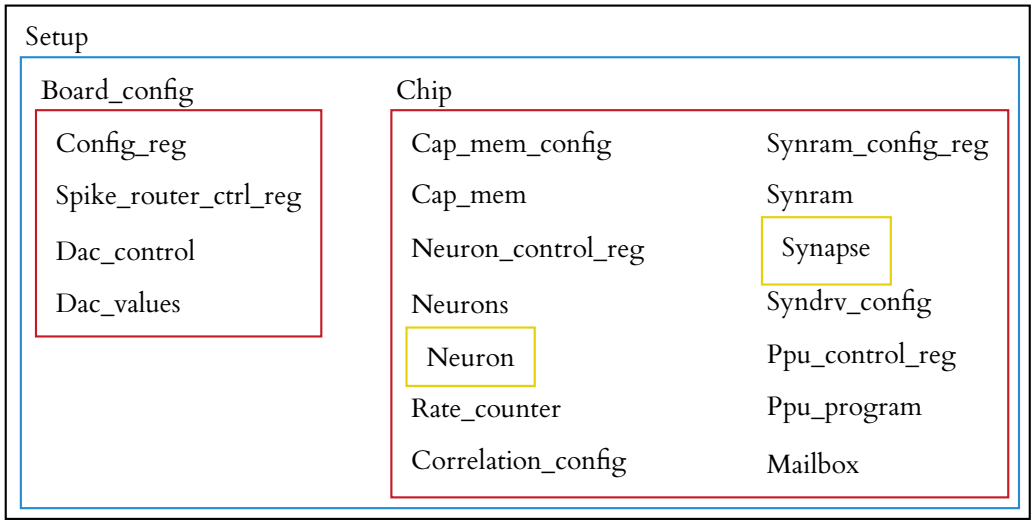

Figure 2.5.: Hierarchical container structure in the host software of the full setup configuration. Every leaf item is a configurable module either on the chip or on the experiment baseboard.

```
1: t ← 0                                  ▷ Reset the global time counter to zero
2: WAIT_UNTIL(t > 10^4)                   ▷ Wait until the counter reaches 10^4 cycles
3: SEND_SPIKE(address=1, row=0)           ▷ Send the 1st spike
4: WAIT_UNTIL(t > 2 · 10^4)
5: SEND_SPIKE(address=1, row=0)           ▷ Send the 2nd spike
6: STOP                                   ▷ Stop execution and recording of spikes
```

Figure 2.6.: Playback program which sends two spikes with an inter–spike interval of 10^4 FPGA cycles which corresponds to 100 ms biological time. The global time counter t increases with every FPGA clock cycle.

containers is shown in figure 2.5. The containers of the on–chip modules also implement the bit–formatting which transforms the configuration into a stream of 32 bit words which are understood by the chips' digital IO interface. For the containers holding the configuration of the baseboard modules the settings are transformed into control–register settings for the FPGA. The content of the containers and configuration options of the individual containers will not be discussed here.[3]

Another task of the `frickel-dls` host software is the composition of playback programs. These are precisely timed instructions to the chip which are for example external input spikes and synchronization signals to the PPU. A minimal example for a playback program is listed in figure 2.6. The instruction set of the playback program named Universal Neuromorphic Instruction set (UNI) has timing, read and write instructions. It furthermore describes the

[3]There exists an internal tutorial written by the author on how to get started with the software: `https://brainscales-r.kip.uni-heidelberg.de/projects/symap2ic/wiki/dls-testing-cookbook`.

data format of recording experiment results, in particular spike events and the responses of chip configuration read requests e.g. requesting the current synaptic weight $w_{0,0}$. The instructions are executed in sequential order as no commands to alter the control flow exist, for example no `if`– or `while`–statement is defined. Documentation on the instructions and the execution model is provided in the repository of the host implementation [Friedmann, 2017]. During the execution of a playback program the spikes from all neurons on the chip are recorded.[4]

The third responsibility of `frickel-dls` besides experiment configuration and playback program composition is the communication with the FPGA over USB. This includes transferring the configuration and experiment playback programs to the FPGA, trigger the experiment execution and fetch recorded spikes and data from the FPGA memory.

2.3.2. PPU Software Tools

Generating binary programs for the on–chip processor is not straight forward, as the custom vector extension has its own, non–standard instruction set. While the Power–Instruction Set Architecture (ISA) is supported by the GNU Compiler Collection (GCC), vector instructions cannot be used with the standard compiler. A patched version is publicly available which supports vector variable declarations, automated register allocation and wrapping of the assembly instructions as built–in functions [Heimbrecht, 2017, Stallman and the GCC Developer Community, 2017]. Register allocation is the process of mapping the variables in the source code to registers of the processor and save and fetch them from the main memory if necessary. Built–in functions of a compiler are an extension to the programming language that allow for calling the instructions of the instructions set with syntax of the programming language. Further information on this topic can be found for example in Aho et al. [2006]. These features simplify programming the vector extension and prevent a large range of programming errors, e.g. overriding register values which are later supposed to be used. The patched cross compiler together with the `libnux` library allows for the implementation of programs in the C programming language that can be executed by the embedded plasticity processor, the PPU. While the patched GCC can cross compile different programming languages the implemented execution startup routines are implemented C–specifically. Furthermore this library provides various low–level features frequently used when programming the plasticity processor.

- Chip specific constants and addresses.
- Read and write functions to the mailbox, a predefined address space of the memory for communication with the FPGA.
- String output using the mailbox.
- Linear congruential random number generation with constants from Press et al. [2007].

[4] While writing this thesis there may be more spikes recorded by error which is known as work package #2373. This issues does not distort the presented measurements as “wrong” spikes can be filtered reliably by their timing.

- A conveniently usable basic testing framework.

Though, the vector instructions are available as built–in functions when programming in C, the vector extension was programmed during the work for this thesis in assembly syntax. The assembly is embedded in the C code using the GCC extended assembly syntax and also taking advantage of the automated vector register allocation. This was considered to be the only option to have fast and efficient updates of the synaptic weights.

Summarizing, the GCC cross compiler together with the `libnux` library provide a low–level interface to implement plasticity algorithms for the PPU. Yet, effort needs to be taken to simplify this process and to hide the current complexity of programming the vector extension efficiently.

3. Experiments and Results

3.1. The Implemented Plasticity Algorithm

In order to find any errors in the implementation as well as unexpected interference between the analog and the digital part of the chip, the synaptic plasticity updates as performed by the PPU are analyzed in the following. The figures in this section further illustrate the plasticity algorithm with measurements on the hardware.

3.1.1. The Biased Random Walk of the Weights

First, the approximation of the OU process

$$\Delta w_{ij} = \frac{2 w_{ij} \lambda_{\text{decay}} + n_{ij}}{8} \tag{3.1}$$

to which the weights are subject without spike correlations is tested. This only approximates the OU process as the random numbers are uniformly distributed and not from a continuous Wiener process. The description of the constant λ_{decay} and the noise n_{ij} is given in section 2.2.4. As for the full plasticity algorithm in equation (2.15) the result of the division by 8 is not rounded but floored onto an integer value. The data for the following figures are not simulated but measured on the DLS including the full configuration as it is also used for later experiments. In particular, the updates are measured with the full plasticity algorithm and enabled neurons and the synapses. The correlation term in equation (2.15) is turned off by setting λ_{stdp} to zero.

The average update of the synaptic weights due to decay and biased noise is shown in figure 3.1. The slope of $\langle \Delta w \rangle(w)$ is determined by $\lambda_{\text{decay}} = -4/128$ (cf. equation (3.1)). The random numbers n_{ij} are drawn uniformly from -2 lsb to 13 lsb which results in a bias of 3/16 lsb:

$$\langle \Delta w \rangle(w = 0) = \frac{\langle n_{ij} \rangle - 4}{8} = \frac{3}{16}, \tag{3.2}$$

where the -4 accounts for the bias introduced by the division in equation (3.1) that floors the result to the next lowest integer.

Different considerations were taken into account for this choice of the random numbers and λ_{decay}: First, the randomness should be such that in the long run all weights are reachable by chance. For any weight the probability to increase and to decrease must be nonzero, except for the boundary cases. This prohibits too large decay factors. Furthermore, the randomness should not change the weights by more than ± 1 lsb for one synaptic plasticity update in order to always ensure weak gradual changes of the synaptic weights. Last, the

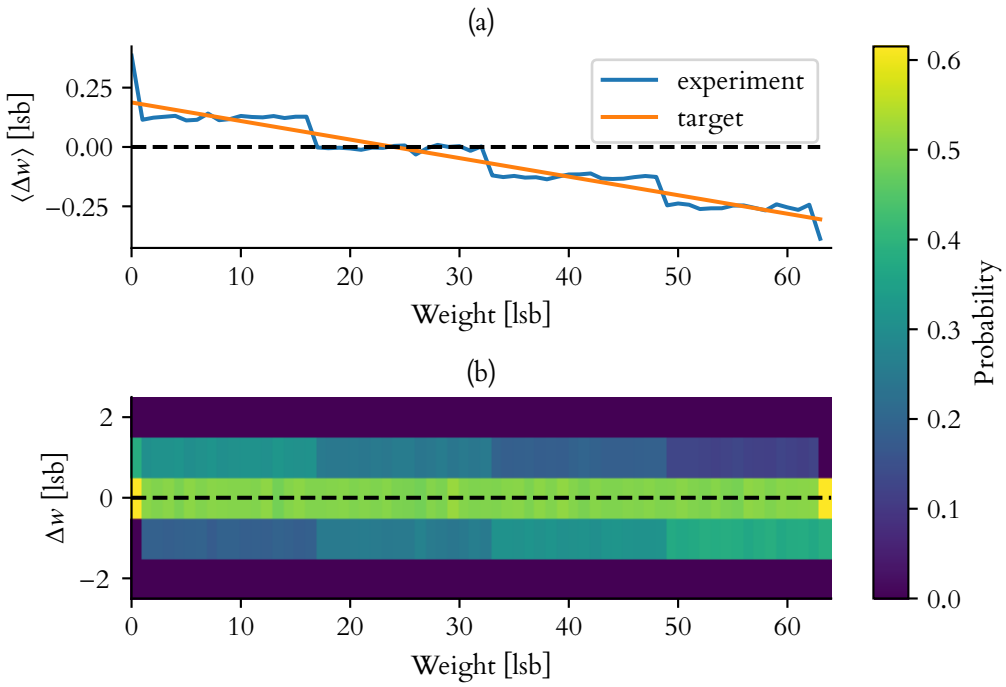

Figure 3.1.: (a) The blue line shows the average plasticity update of a synaptic weight without correlation measurements. It approximates the linear dependence drawn in solid orange where λ_{decay} determines the slope while the random noise n_{ij} accounts for the bias. (b) Probability for a weight's update. While the smallest update is ± 1 lsb, the smallest shift for the average update is $1/8$ lsb. Both figures are measured on the DLS where the PPU updates all 1024 synapses.

bias should be positive with a stable steady state which is above the weight needed to make the neurons fire. Even with initial weights of zero all neurons should in the long run start to spike.

While the decay and the STDP factor will be varied in later experiments the random numbers will be drawn from the same distribution throughout all experiments. Drawing the random numbers in the kernel code is time critical as for each update and each synapse an individual random number needs to be drawn. Therefore, the distribution is hard coded and not parameterized such that the compiler has more optimization opportunities.

The functional form

$$\langle \Delta w \rangle = -\frac{1}{128} w + \frac{3}{16}, \tag{3.3}$$

shown as an orange line is well approximated in figure 3.1. The smallest shift of the average weight update is $1/8$ lsb as the update is calculated with three non-significant bits which are taken into account by the stochasticity. As the update can only be an integer, the lower

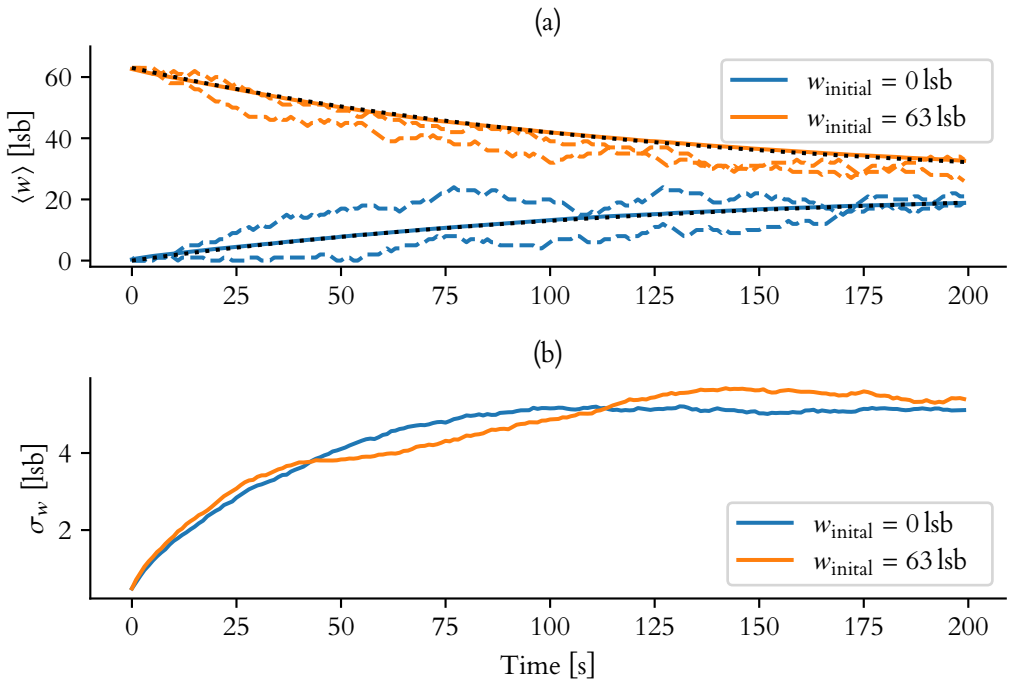

Figure 3.2.: Average weights (a) and variations (b) over all synapses starting at an initial weight of 0 lsb (blue) and 63 lsb (orange). The updates as shown in figure 3.1 are applied every second. In dashed blue and orange the weights of two randomly chosen synapses are shown for both experiments. The black dashed lines show the expected exponential decay with a time constant of 128 s and a stable weight of 24 lsb.

graph in figure 3.1 shows the probability for the weight to change. The largest change for any synaptic weight for this set of parameters is ±1 lsb.

This test shows that the random numbers from the linear congruential generator are not biased, but they may still be correlated which cannot be seen in this figure. This test also shows, that the boundary cases for the smallest and largest weights are handled correctly. At the lower bound of 0 lsb and at the upper bound of 63 lsb no overflow and underflow errors occur, instead, weights are clipped to the valid range.

If equation (3.1) is applied every second, the average weight over all synapses is expected to decay towards the stable fixed point of 24 lsb. This temporal evolution of the average synaptic weight is shown in figure 3.2a. The correlation term is still turned off: $\lambda_{\text{stdp}} = 0$. Two experiments are shown with initial weights of 0 lsb for the blue line and initial weights of 63 lsb for the orange line. The dashed orange and blue lines show the weights of two randomly chosen synapses as an example how the individual weights evolve. The expected exponential decay given by the solution to the OU process with a time constant of 128 s

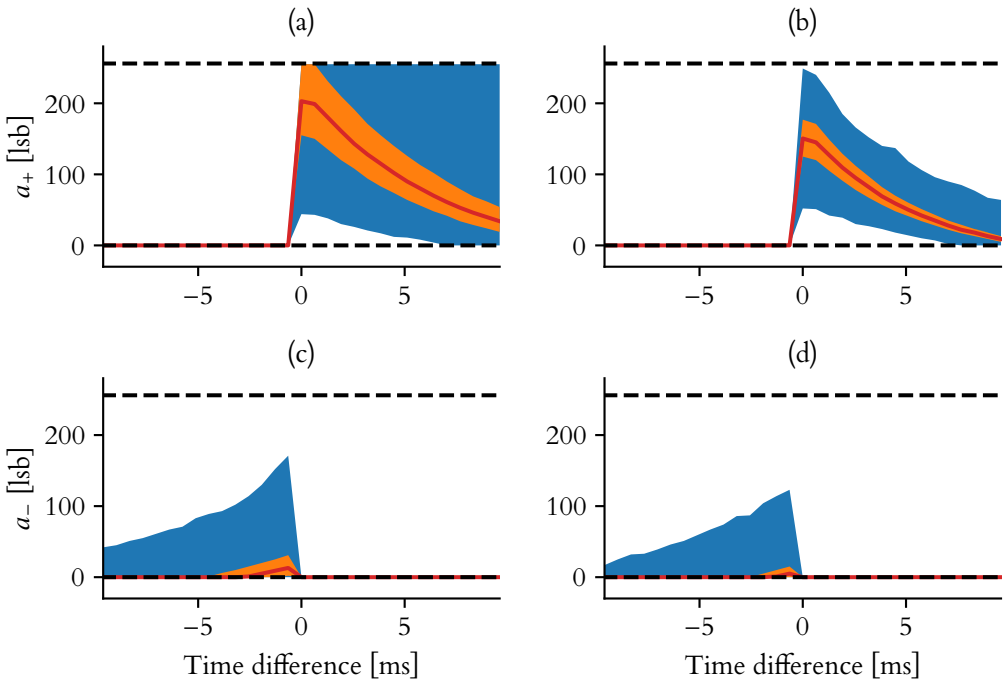

Figure 3.3.: Causal (upper row) and anticausal (lower row) correlation sensor measurements for 10 spike pairs over all 1024 synapses. In each plot the blue shaded area shows the full range of measured correlations over all synapses. The orange area shows the range of the 25 % to 75 % quantile. The median is drawn as a red line. Figure (a) and (c) show the measured correlation without the calibration, figure (b) and (d) with calibration. The range from 0 lsb to 255 lsb of the digitizer for the correlation measurements is shown by the black dashed lines.

(cf. equation (2.17)) towards the stable fixed point of $w = 24\,\text{lsb}$ (cf. equation (2.18)) is shown by the black dotted lines. The lower graph, figure 3.2b shows the variation among the individual weights over time. While the average weight decays the variation among the synaptic weights grows.

3.1.2. Correlation Sensor Measurements

As the last ingredient to the plasticity algorithm the spike correlation sensor is characterized in the following. The causal and anticausal correlation sensor measurements for 10 spike pairs are shown in figure 3.3 with and without using the per–synapse calibration bits. For each time difference plotted on the horizontal axis, all 1024 correlation sensors are read out after receiving 10 pre–post spike pairs. The blue shaded area shows the full range of measured correlations while the orange area displays the range of the 25 % to 75 % quantile. The time constant fitted to the median of the measurements with the per–synapse calibration is 5.3 ms.

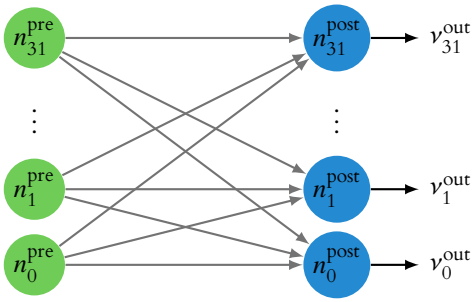

Figure 3.4.: Abstract view of an emulated feed forward neural network. The green neurons represent randomly spiking spike sources which are connected to the implemented neurons shown as blue circles.

Figures 3.3a and 3.3b show the measurements where all per–synapse calibration bits are set to the same default value. For the measurements in figures 3.3b and 3.3d the calibration bits are configured individually such that the targeted time constant and amplitude is matched as closely as possible. The synapse–to–synapse variation of the measured kernel functions can be decreased significantly. The spread in both columns is dominated by fixed pattern synapse–to–synapse variations and not trial-to-trial variations. The per–synapse calibration of the correlation sensors is limited by the coarse control one has over the amplitude and time constant with the calibration bits [Wunderlich, 2016, Friedmann et al., 2017].

If the causal correlation is within the sensor's dynamic range—as shown in figure 3.3—the anticausal branch will mostly measure $a_- = 0$. This is a problem which is supposed to be fixed in later chip revisions. As both branches are scaled with the same global constant, this implies that the causal and anticausal branch of the STDP rule cannot be taken into account for kernel functions $K(\Delta t)$ both at once. Configuring the anticausal sensor to yield measurements in its dynamic range comes at the cost of having the causal measurement saturating at $a_+ = 255$. Hence, for all experiments only the causal branch will be used.

3.2. Weight Dynamics Towards Stability

As motivated in the introduction in chapter 1 one of the main aims is to generate a connectivity matrix by local plasticity rules that yield moderate spiking activity for all neurons and therefore acting as a homeostatic plasticity mechanism. This moderate spiking activity cannot be achieved with equal parameter settings as every neuron and every synapse varies in its dynamics due to transistor mismatch. If all weights in a feed–forward network with random spike input as sketched in figure 3.4 are set to the same weight one neuron might be bursting while another neuron does not fire at all even though the same input spikes are received.

The sketched feed–forward network with Poissonian distributed input spikes from 32 spike sources at 30 Hz projecting onto the 32 neurons on the DLS is chosen for the experiments in this section. All 1024 synapses are subject to synaptic plasticity. The fixed point of the weights shown in the previous section 3.1 is such that every neuron fires close to

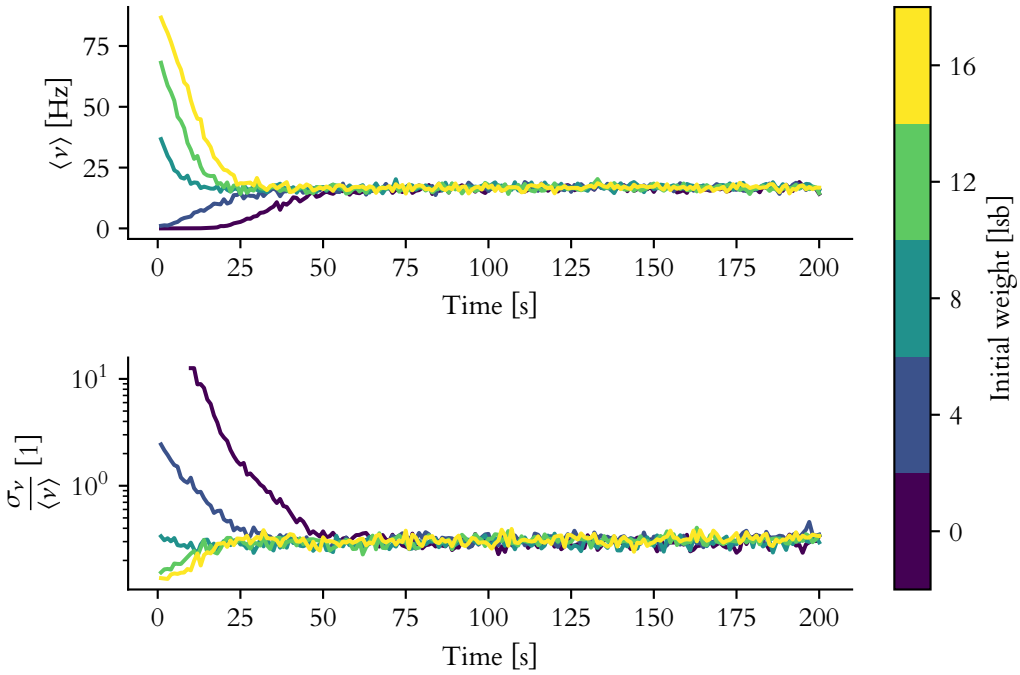

Figure 3.5.: Evolution of the average firing rates and the relative variations over all postsynaptic neurons for anti-Hebbian plasticity. The relative variations are considered to be the better measure for how different the firing rates are as explained in the text.

$\nu_{\max} = 1/\tau_{\text{ref}}$ (cf. figure 3.16a later in the experiment section). By setting the STDP scaling factor to $\lambda_{\text{stdp}} = -16/128$, correlated spiking is penalized. Every postsynaptic spike contributes to a decrease of the synaptic weight if there was a presynaptic spike close before. The decrease of the weight for correlated spiking will be referred to as anti–Hebbian plasticity.[1] It is expected that every synaptic weight constantly grows at a small rate. As soon as the weight is strong enough to make the postsynaptic neuron spike the weight is decreased again. The growth and this penalty to the synaptic weight are both configured to be weak such that a plasticity update would not be larger than ±2 lsb. This is supposed to ensure that the updates happen softly and no update iteration may strongly change the weight matrix.

The evolution of the average firing rate of the postsynaptic neurons during an emulation is shown in figure 3.5. The different colors encode different initial weights and therefore different initial firing rates. For each initial weight five repetitions with different random seeds for the spike sources and synaptic random term are averaged. The relative variations of the firing rates among all neurons of all repetitions are shown in the lower graph as they are

[1] This naming is commonly used and relates to [Hebb, 1949] which is often summarized as “Cells that fire together wire together”.

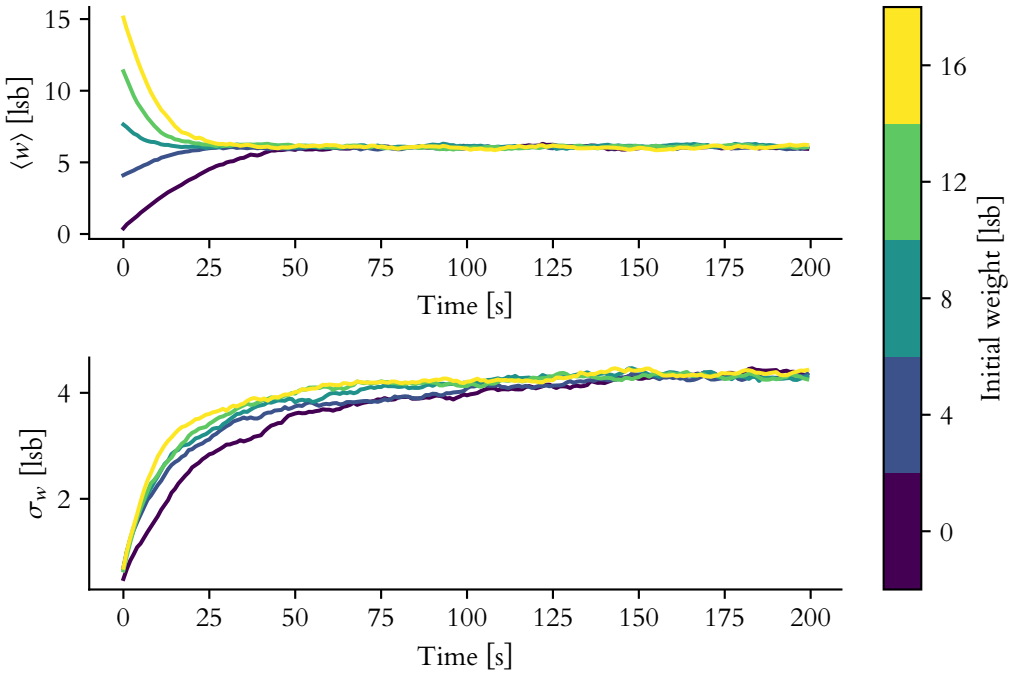

Figure 3.6.: Evolution of the average weights and the variations over all synapses for anti-Hebbian plasticity.

considered to better show how different the firing rates are. For the initial weight of 16 lsb, initially all neurons fire at a very high rate such that the absolute spread for the firing rates among the neurons is large. Still, all neurons show a similar behavior as all firing rates are at the same order of magnitude. Contrary, for low initial weights, the firing rates vary among the neurons within a factor of 10 showing a much more diverse behavior from neuron to neuron even though the absolute variations are small.

As expected, due to the plasticity updates the system reaches a steady state where all neurons fire at a moderate rate of 17 Hz. All firing rates of the individual neurons in the last 10 s are within 3.9 Hz to 45 Hz. Neither are neurons firing close to the maximum firing rate of 210 Hz nor are any neurons not firing at all. Furthermore, the same steady state of firing rates is reached regardless of the initial conditions. In particular, an initially disconnected neural network also reaches the same steady state of moderate firing activity.

The evolution of the average weight over all synapses and five repetitions is shown in figure 3.6. As for the firing rates, the same stable average weight of 6 lsb is reached for long emulation times regardless of the initial conditions. Though the stable average weight is quickly reached, the variations show slower dynamics towards a stable state. In comparison to the evolution without STDP in figure 3.2, the steady state average weight is much lower with the anti-Hebbian STDP term. This shows that indeed the positive bias and the anti-Hebbian STDP term balance far below the steady state solution of 24 lsb. For large initial

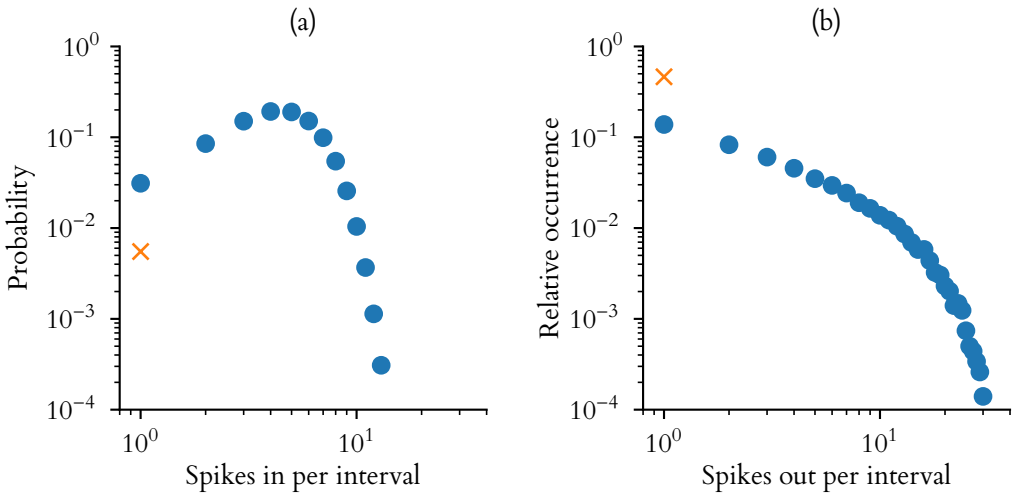

Figure 3.7.: (b) Probability of the number of input spikes over all 32 spike sources within intervals of 5 ms. There are on average 4.80 input spikes per interval. (a) Relative occurrence of the number of output spikes summed up over all neurons within the same interval size. There are on average 2.72 spikes in the intervals 5 ms. For both plots the orange cross marks the probability for zero spikes within a time window.

weights and therefore large initial firing rates the decrease of rates and weights happens much faster than in the free case (cf. figure 3.2). In this region the dynamics are dominated by the STDP term. For low initial weights on the other hand, the rates and weights increase slowly with the same speed as without STDP. Here, the positive bias dominates the dynamics.

In figure 3.7a the probability of the number of input spikes across all 32 spike sources within intervals of 5 ms is shown. It is a binomial distribution with $p = 30\,\text{Hz} \cdot 5\,\text{ms}$ and $n = 32$. The same statistics over the number of spikes over all postsynaptic neurons is shown in figure 3.7b. The time interval of 5 ms was chosen to match the neurons refractory periods such that each neuron contributes with at most one spike to the number of spikes within an interval. Therefore, there are at most 32 spikes within one time frame for both graphs in figure 3.7. The orange crosses mark the probability for zero spikes within a time window, as this cannot be correctly placed on a logarithmic scale.

It is remarkable how much the distributions of input and output spikes differ, though the average rates of 30 Hz for the input and 17.0 Hz for the output are at the same order of magnitude. While it is highly unlikely to have all 32 source neurons fire within a period of τ_{ref}, this is still observed for the output neurons. This indicates that the output neurons tend to fire together. As all neurons may connect to the same spike sources by means of the same plasticity algorithm this is expected. The only asymmetry between the synapses in the theoretical model is the contribution of the random term to the plasticity algorithm.

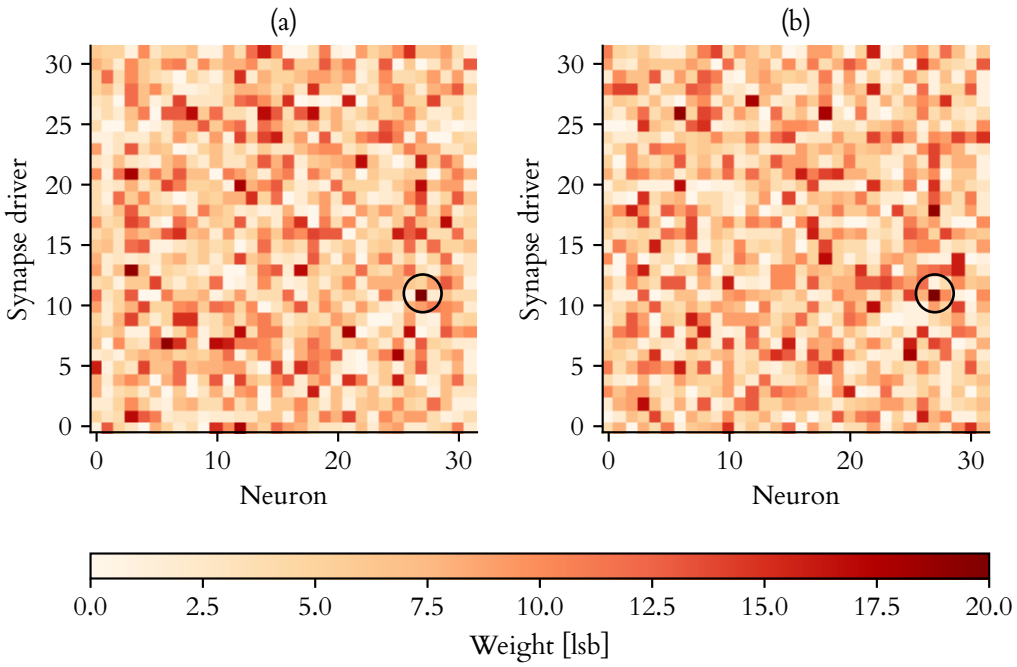

Figure 3.8.: Weight matrices after the emulation of 200 s for two different random seeds for the synaptic random term and the random spike sources. The circles are drawn for reference.

In practice there also is asymmetry due to different strengths of the synapses, correlation sensors and sensitivity of the neurons.

To qualitatively show this asymmetry, figure 3.8 shows two weight matrices at the steady state for different random seeds. Different random seeds for the random term in the synaptic plasticity and for the spike sources generate different weight matrices. These two examples of resulting weight matrices do not show, but indicate that the steady state solution is not too biased by the synapse–to–synapse variations. Though, the synapse $s_{11} \rightarrow n_{27}$, highlighted with the black circle, is probably biased for example, as the final weight is quite large in both trials. A more thorough analysis on a bias of the steady state by fixed pattern noise is given in figure 3.9. The correlation between the amplitude of the correlation sensor and the steady state weight is shown in figure 3.9a. Each synapse has its own correlation sensor with an individual sensitivity (the spread of the correlation sensor amplitudes is measured in section 3.1.2). For the same correlation of pre–post spikes, a synapse with a more sensitive correlation sensor decreases the weights stronger than with a less sensitive correlation sensor. A synapse with a strong correlation sensor therefore contributes to the histogram in figure 3.9a in the lower right as the steady state weight is likely to be low. All synapses of 25 repetitions with different seeds are taken into account individually for the histogram.

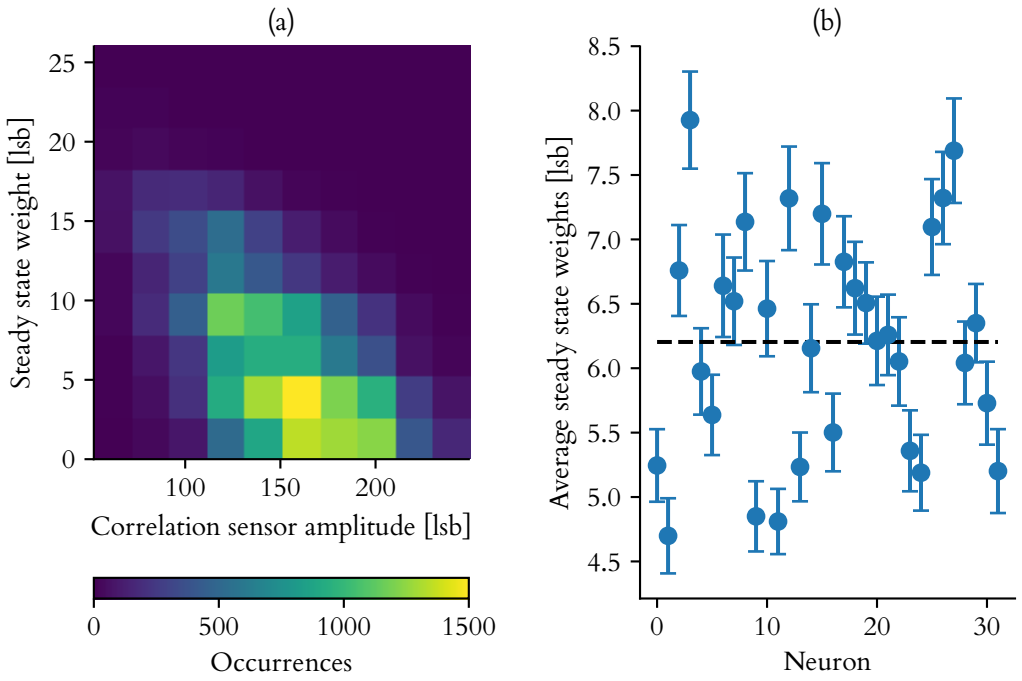

Figure 3.9.: (a) Two-dimensional histogram over the final synaptic weight and the synapses' correlation sensor amplitude measured in section 3.1. (b) Average incoming synaptic weights for each individual neuron. The error bars show the uncertainties of the average weights. The data from all five repetitions over the five different initial conditions is included in the histogram and the graph of the average final weight per neuron.

Figure 3.9b shows the average steady state weight of all incoming synaptic connections with their respective uncertainties for each neuron. The neuron–to–neuron variation of this average weight is much larger than the expected uncertainty. In other words, there are neurons that systematically have higher or lower incoming synaptic weights than the expected average. This shows a form of fixed pattern noise from neuron to neuron which is believed to be due to varying strengths in the synaptic inputs.

3.3. Recurrence in Stable Networks

With the plasticity algorithm, a robust tool is available to generate neural networks with a stable moderate activity. This allows for studying how recurrent connections among the neurons would change the dynamics in a spiking neural network with all of its neurons at a favorable working point. In this section two questions are addressed: Do recurrent connections add some kind of measurable memory to the network and does the sensitivity

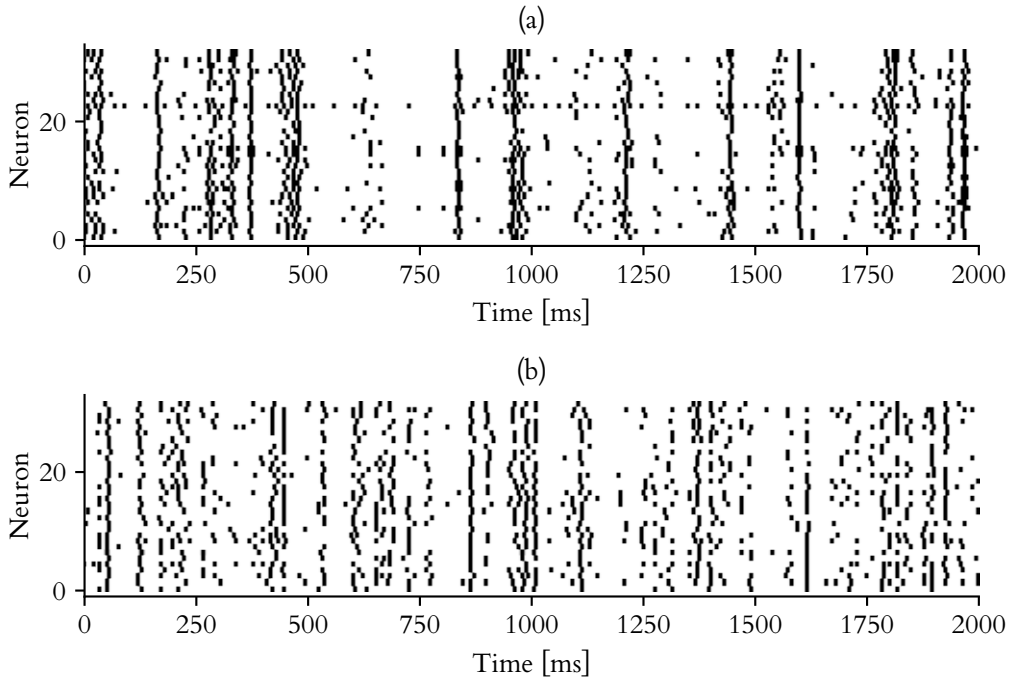

Figure 3.10.: (a) Spike pattern after 198 s for a network with 16 recurrent synapses and 16 synaptic connections from external spike sources for each neuron. (b) The spike pattern for a pure feed–forward network.

to perturbations increase? The first question is a natural question to ask as the recurrent connections allow for the information packages, as which the spikes may be seen, to re–enter the system at multiple different locations. The second question whether the sensitivity increases arose when trying to hand–craft a connection matrix that yields stable activity with recurrent synapses. The more recurrence the harder it gets to make the network neither excite itself to exploding activity nor to have the activity die out. Analytical calculations with balancing excitatory and inhibitory populations support this finding [Brunel, 2000].

For the following experiments the feed–forward network with stable plasticity parameters is modified to also have recurrent connections. The degree of recurrence d_{rec} is used to control the amount of connections among the neurons: each neuron has $32 - d_{\text{rec}}$ Poisson spike sources as presynaptic partners and d_{rec} recurrent connections from randomly chosen neurons, also allowing for synaptic connections with itself. Throughout this section the plasticity parameters are the same as in the previous section ($\lambda_{\text{stdp}} = -16/128$ and $\lambda_{\text{decay}} = -4/128$). Also, the random input spikes are at a rate of 30 Hz except otherwise noted.

The spikes of a network with plastic synapses and a degree of recurrence of 16 is shown in figure 3.10a. Below in figure 3.10b the spike pattern of a network without recurrent connections but only the feed–forward connectivity is shown. Both spike patterns show

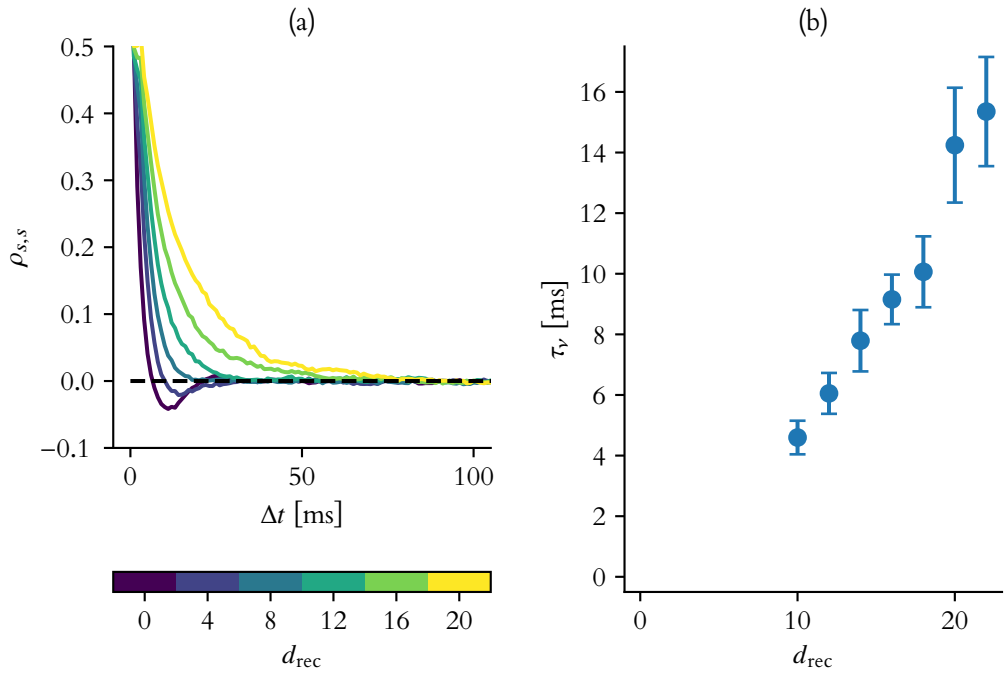

Figure 3.11.: (a) Autocorrelation of the total activity for different degrees of recurrence. (b) Time constant of the decay of the autocorrelation.

the steady state recorded after 198 s and therefore 198 synaptic plasticity updates. The network with recurrence shows distinct periods of high activity and low activity, whereas the spike pattern without recurrence is much more homogeneous. With increasing number of excitatory recurrent connections a high spiking activity will further excite the neurons. Contrary, a low spiking activity will keep the network at a low activity. Broadly speaking, the recurrence adds a form of short term memory to the networks' activity.

3.3.1. Time Constant of the Collective Neural Activity

To quantify the previous qualitative observation—the more recurrence the longer the network stays in a high or low activity state—the autocorrelation of the summed up spiking activity of all neurons is shown in figure 3.11a. The autocorrelation $\rho_{s,s}(\Delta t)$ of the spiking activity $s(t)$ is defined as commonly used in statistics

$$\rho_{s,s}(\Delta t) = \frac{1}{\sigma_s^2} \int_{-\infty}^{\infty} [s(t) - \mu_s]\,[s(t + \Delta t) - \mu_s]\,\mathrm{d}t\,, \tag{3.4}$$

where μ_s is the average and σ_s the standard deviation of $s(t)$. The spiking activity s is the total number of spikes, binned within bins of 1 ms. As $s(t)$ can only be recorded over a finite duration, the integral limits are cropped to the valid range where both $s(t)$ and $s(t + \Delta t)$ are

given. The autocorrelations are calculated for spike trains of 10 s duration during which the weight matrices were kept static. For each degree of recurrence 10 weight matrices were generated with different random seeds. The correlations are averaged over those 10 weight matrices and further over 10 trials with different random spike input.

A positive autocorrelation for some time lag Δt indicates, that if s is large at time t it's likely to be large at time $t + \Delta t$, too. Contrary, a negative autocorrelation for Δt indicates, that s is likely to be small at time $t + \Delta t$ if it was large at time t.

For small degrees the autocorrelation quickly drops below zero. This is expected: if many neurons were active at time t, all of them are in the refractory state shortly after and may not spike again. For large degrees the autocorrelation does not drop below zero, instead slowly decreases. A large firing activity at time t will excite the non spiking neurons via the recurrent synapses such that the activity stays high. Furthermore, if there is low activity at time t is is likely to still be low after a short time lag: then the neuron mostly get spikes from the external spike sources, but only few from the other neurons within the network.

To quantify the memory how long such a high or low activity state is stable, the time constant of the decay of the autocorrelation is fitted. Figure 3.11b shows these time constants. The more recurrence, the longer the network keeps its activity.

With these findings at least a very simple form of memory is shown. The next section addresses the question on how sensitive the system responds to perturbations.

3.3.2. Sensitivity to Perturbations

The sensitivity of the neural network towards perturbations is analyzed with the following three measurements:

- The trial–to–trial variation between two experiments is measured where the chip configuration and the input spikes are exactly the same. This shows the influence of the analog parameter reconfiguration variations, thermal noise or jitter due to the spike routing and recording on the FPGA.
- The input spikes are perturbed by adding a pulse of spikes at a specific point in time. The neurons' responses are compared to their responses if there are the same random input spikes without a pulse.
- The last measurement for analyzing the sensitivity is the decrease and increase of the spiking activity, if the frequency of the random spike input is changed.

The trial–to–trial variations in the resulting spike patterns are measured by the squared difference between the Gauss–convolved spike trains:

$$\Delta_{\text{spikes},m,n} = \frac{\sum_i \int_{-\infty}^{\infty} \left(s_i^m(t) - s_i^n(t)\right)^2 \mathrm{d}t}{\sum_i \int_{-\infty}^{\infty} \left(s_i^m(t) + s_i^n(t)\right)^2 \mathrm{d}t}, \tag{3.5}$$

with

$$s_i^m(t) = \int_{-\infty}^{\infty} \exp\left[-\frac{1}{2}\left(\frac{t-t'}{\sigma_t}\right)^2\right] \sum_{t^m_{\text{spike},i}} \delta\left(t^m_{\text{spike},i} - t'\right) \mathrm{d}t' . \tag{3.6}$$

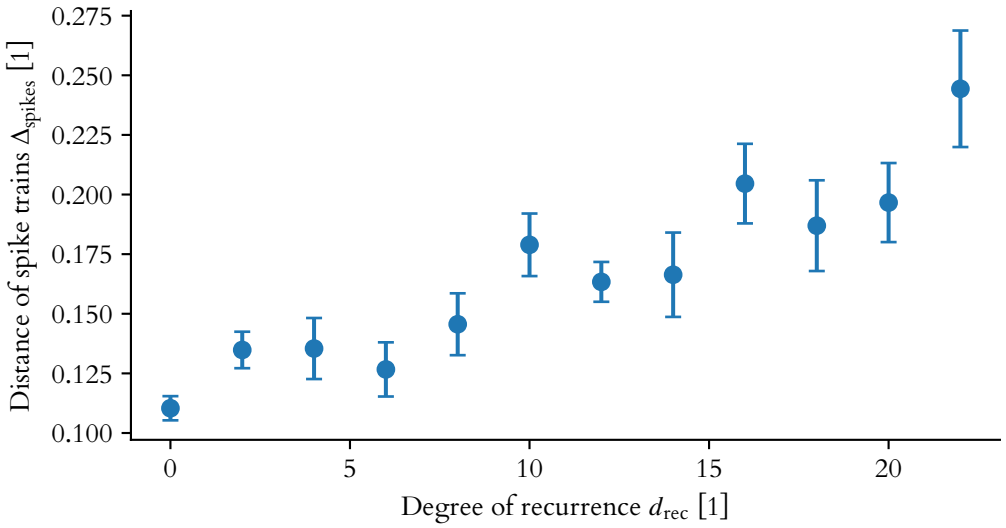

Figure 3.12.: Average distance between the spike trains of repetitions of the experiment with exactly the same random spike input and configuration. The distance measure Δ_{spikes} is explained in the text.

The index i indexes the different neurons, m and n the different trials and σ_t denotes the width of the Gaussian kernel. The kernel of the convolution is not normalized, as the measure in equation (3.5) is invariant under scaling all $s_i^m(t)$ with the same factor. The measure is within $[0, 1]$ where zero codes for a perfect overlap between the Gauss–convolved spike trains, while one stands for no overlap at all. Not only the number of spikes is accounted for by the spike measure but also the precise timing. This measure is similar to the van Rossum distance [van Rossum, 2001] where the squared distance of the spike trains convolved with an exponential decay for $t > 0$ is calculated. Here, the proposed measure is considered to be easier to interpret as its boundaries have intuitive interpretations.

For each degree of recurrence the distance in the spike patterns between two trials of the same experiment is shown in figure 3.12. The distances are averaged over 10 pairs of experiments for the 10 generated matrices which were also used for measuring the autocorrelations. The width of the Gaussian kernel is 2.5 ms which is chosen to match half of the refractory period. Therefore, a spike contributes to $s_i(t)$ mostly during a duration of τ_{ref}. Each experiment is executed with a duration of 1.1 s where the spikes of the last 1 s are used for the calculation of the distance. The first 100 ms are discarded for the evaluation such that the warm-up at the beginning of an emulation does not contribute to the distance measure. Longer experiments cannot be used for calculating the distance because of a known issue that causes the times of recorded spikes to be shifted systematically.[2]

[2]This is internally documented as working package #2550.

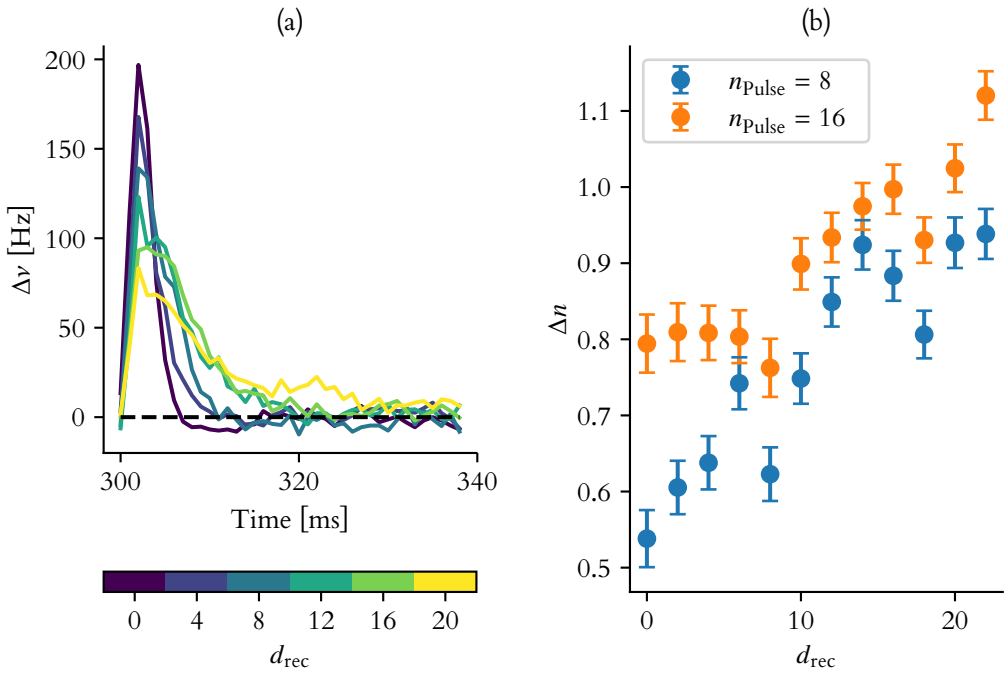

Figure 3.13.: (a) Average increase in the average spiking frequency of the neurons when introducing additional spikes on 8 of the 32 external spike sources at $t = 300$ ms. (b) Average increase in the number of spikes per neuron for different strengths of the pulses.

The measure of distance increases for an increasing degree of recurrence. Even without recurrence the resulting spikes vary from trial–to–trial, shown by the nonzero distance measure. Possible reasons are trial–to–trial variations in the analog parameter storage, thermal noise in all analog circuitry and jitter on the recording of spikes on the FPGA. The distance is interpreted to increase with an increasing degree of recurrence as these small changes further influence the spike times in the future. The postsynaptic neurons may receive different spike trains from trial–to–trial which even stronger change the future spike times. The more recurrence the more the perturbations within chip and FPGA influence the system.

Next, the neurons' responses to pulses of input spikes are tested as a perturbation of the system. Figure 3.13 shows the response as the time resolved increase of the spiking frequency as well as the total number of additionally excited spikes. In figure 3.13a the increase in the average spiking frequency is shown, if 8 out of the 32 external spike sources spike at $t = 300$ ms. The frequency increase is averaged over 10 repetitions and 10 different weight matrices for each of the recurrent degrees.

Within 5 ms after the pulse, the average firing rate increases stronger for fewer recurrence. Looking at short time lags, the feed–forward network is the most sensitive to the tested perturbation on the spike input. This may be explained by the fact that all neurons are

connected to all input sources. The spiking activity is solely determined by the input spikes and not by any recurrent spikes. Therefore, on the other hand, the feed–forward network quickly forgets the pulse, shown by the fast decrease in figure 3.13a.

For larger degrees of recurrence the increase of activity is not as strong but longer in time compared to no recurrence. As already seen for the autocorrelation in figure 3.11a the increased activity after the pulse further excites the network via the recurrent connections.

Figure 3.13b shows the average increase of the total number of spikes per neuron when comparing the perturbed network to the unperturbed. An increase of $\Delta n = 1$ means, that all of the 32 neurons spiked on average one time more with the perturbation compared to no perturbation. This increase is the integral of the curves shown in figure 3.13a. It measures the total influence the perturbation has for all times. The different colors encode different strengths of the pulse. There is the trend to a stronger increase of the number of spikes for an increasing recurrence. For fewer recurrence the system responds fast and strongly on a short timescale while with increasing recurrence the total influence of the perturbation becomes larger.

As the last measure of sensitivity of recurrent neural networks, the increase of the average spiking frequency is shown in figure 3.14 when varying the random input spike frequency. For each point of the response functions two repetitions of the experiment for each of the 10 weight matrices are averaged. All weight matrices were generated with a random spike input at 30 Hz. Interestingly all curves intersect close to an input spike rate of (30 ± 1) Hz and an output spike frequency of (14 ± 2) Hz, which is the spike rate of input spikes at which the weight matrix was generated. A larger degree of recurrence yields a steeper increase of the output frequency when varying the random spike rate. Again, the more recurrence, the more sensitively the network responds to a shift in the background spike frequency.

3.4. Exploring the Plasticity Parameter Space

All of the previous measurements are performed with the same set of plasticity parameters. However, the synaptic plasticity has a few free parameters which are the decay factor λ_{decay}, the strength of the correlation update λ_{stdp}, the time constant of the STDP kernel and the bias of the random numbers. As the values used previously are chosen with few justification, this section aims to explore the influence of the decay strength and correlation update factor. How sensitive are the resulting network dynamics to these changes of the synaptic plasticity? Due to the acceleration factor of the hardware it is possible to do many plasticity experiments within a short time. One of the repetitions for the sweep in figure 3.16 for example takes 51.6 min wall clock time emulating 16.1 h biological time in which 112×10^6 spikes are recorded and every pre–post spike pair at each of the 1024 synapses is taken into account for the synaptic plasticity. Still, the STDP time constant and bias of the randomness are fixed to limit the parameter space to a sweepable range.

The discrete synaptic plasticity update reads simplified

$$\Delta w_{ij} \propto 2 w_{ij} \lambda_{\text{decay}} + \frac{a_+}{2} \lambda_{\text{stdp}} + n_{ij} \,. \tag{3.7}$$

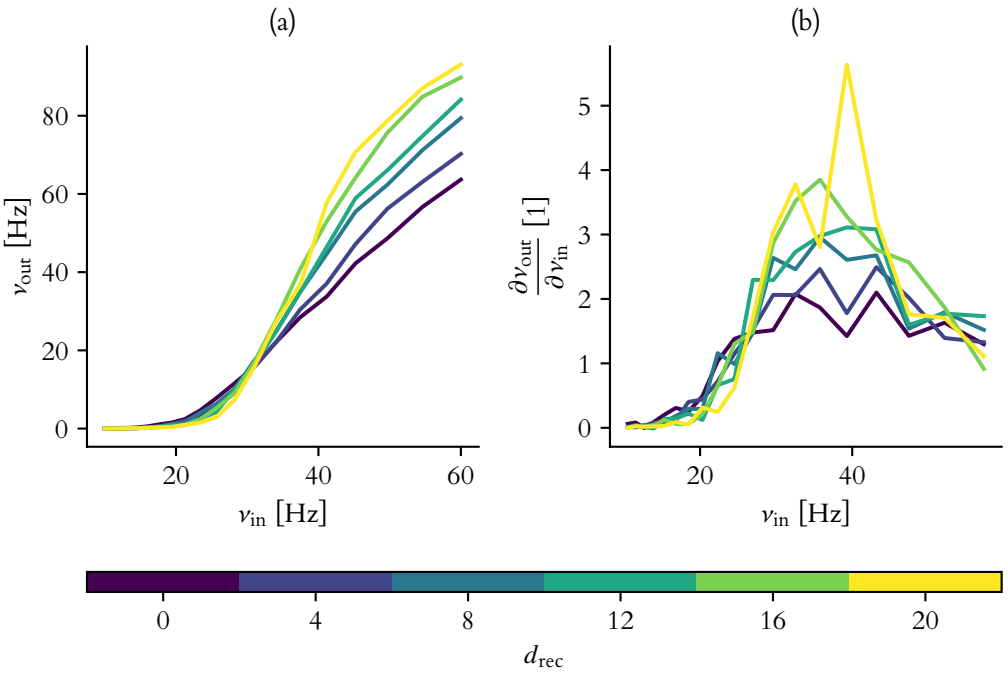

Figure 3.14.: (a) Change in the spiking frequency when varying the rate of the random input spike sources for different degrees of recurrence. (b) Slope of the change in the spike frequency. It shows the gain factor with which changing the input frequency will change the output frequency.

The change of the weights due to decay and bias in the random numbers is sketched in figure 3.15a Given a positive bias of the random numbers, a positive decay factor causes the weights to grow while a negative factor makes the weights decay towards the stable fixed point. The correlation update factor λ_{stdp} configures how strong the accumulated, exponentially weighted pre–post spike pairs contribute to the plasticity update. A positive factor causes the weights to increase, a negative factor causes the weights to decrease on causal correlated spike pairs. Hence, the swept λ_{decay}–λ_{stdp} parameter plane covers different interesting regimes:

- **Hebbian and convergent:** $\lambda_{\text{stdp}} > 0$ and $\lambda_{\text{decay}} < 0$. Causal spiking will lead to potentiation of the synaptic weights while weights decay towards the stable fixed point.
- **Anti–Hebbian and convergent:** $\lambda_{\text{stdp}} < 0$ and $\lambda_{\text{decay}} < 0$. Causal spiking will decrease synaptic weights and weights decay towards the stable fixed point. For large decay factors the activity is expected to die out because all weights will converge towards zero as the stable weight w_0 is proportional to $1/\lambda_{\text{decay}}$. The spiking activity may only speed up this decrease of weights.

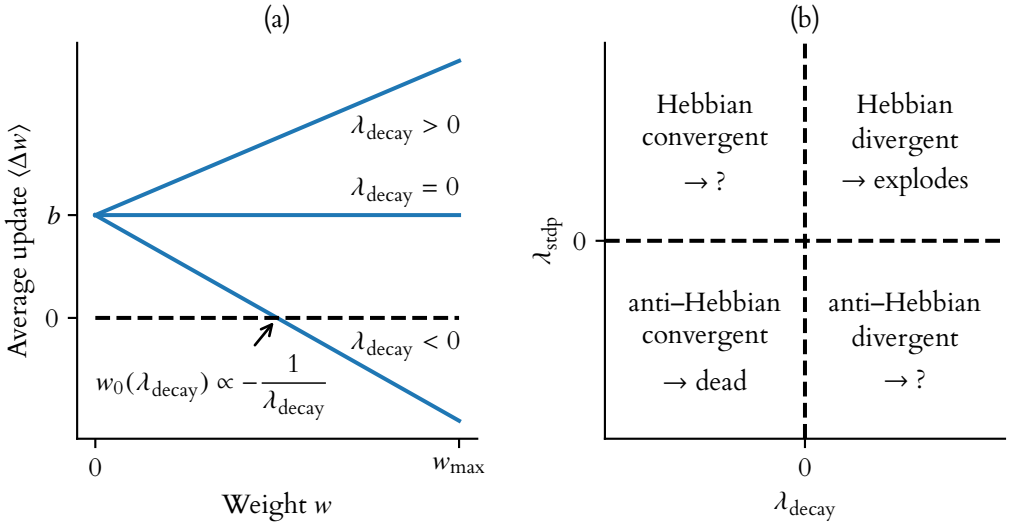

Figure 3.15.: (a) Sketch of the average change in the synaptic weights given a positive bias b in the randomness and no contributions from the correlation term. (b) The different regimes of the swept parameter range. The expected behavior for the Hebbian divergent and the anti–Hebbian convergent regime as explained in the text are also added.

- **Hebbian and divergent:** $\lambda_{\text{stdp}} > 0$ and $\lambda_{\text{decay}} > 0$. Causal spiking will increase the synaptic weights and weights are repelled from the instable fixed point which is smaller than zero. The weights may only increase and therefore spiking activity is expected to explode.

- **Anti–Hebbian and divergent:** $\lambda_{\text{stdp}} < 0$ and $\lambda_{\text{decay}} > 0$. Causal spiking will decrease the synaptic weights while weights are repelled from the instable fixed point which is smaller than zero. In this regime, the network may find a balance between growing weights due to the divergent decay and the decreasing of the weights due to the STDP kernel.

Figure 3.15b shows the different regimes and the expectations.

Using the feed–forward topology as in figure 3.4 the network activity is recorded in the last 10 s out of 200 s emulation time. The measurements from analyzing the stability in section 3.2 indicate that indeed the steady state of the weight evolution is reached after this period. The swept parameters cover mostly regions with stronger factors than for the previous measurements—especially $\left|\lambda_{\text{decay}}\right|$ is mostly larger than 4/128. Therefore the weights are expected to converge even faster. Figure 3.16a shows the average firing rate of the neurons at the end of the emulation for the different plasticity parameters. The frequencies are averaged over all neurons and over three repetitions for each measured point. The 5 % quantile over all postsynaptic neurons' spiking frequencies of all of the three trials is shown in figure 3.16b. It shows the activity of the rarely spiking neurons. Similarly, the 95 % quantile of all postsynaptic neurons' spiking frequencies is shown in figure 3.16c to show the activity of the frequently spiking neurons. The solid line shows the border where the activity of the rarely spiking neurons is larger than 1 Hz. The dashed line on the other hand shows the border where the activity of the frequently spiking neurons is larger than 50 Hz.

In the *Anti–Hebbian convergent* regime in the lower left the spiking activity goes to zero for large decay factors as expected. On the opposite, in the *Hebbian divergent* regime at the upper right all postsynaptic neurons fire with their largest possible firing rate since all weights grew to the largest value of 63 lsb. In the regimes *Hebbian convergent* and *Anti–Hebbian divergent* where STDP and decay may balance different patterns are observed and characterized in the next paragraph.

The solid and dashed lines define different phases which are shown in figure 3.16d. The phases are named:

- **Dead:** There are rarely spiking neurons which fire with less than 1 Hz and there are no neurons firing heavily at more than 50 Hz.

- **Diverging:** Both rarely spiking neurons and heavily spiking neurons exist.

- **Stable:** Neither rarely spiking neurons nor heavily spiking neurons exist.

- **Explode:** No rarely spiking neurons, but heavily spiking neurons exist.

For each of those phases the spikes within the last second of the emulation for exemplary parameter combination are shown in figure 3.17. The spikes are taken from the first of the

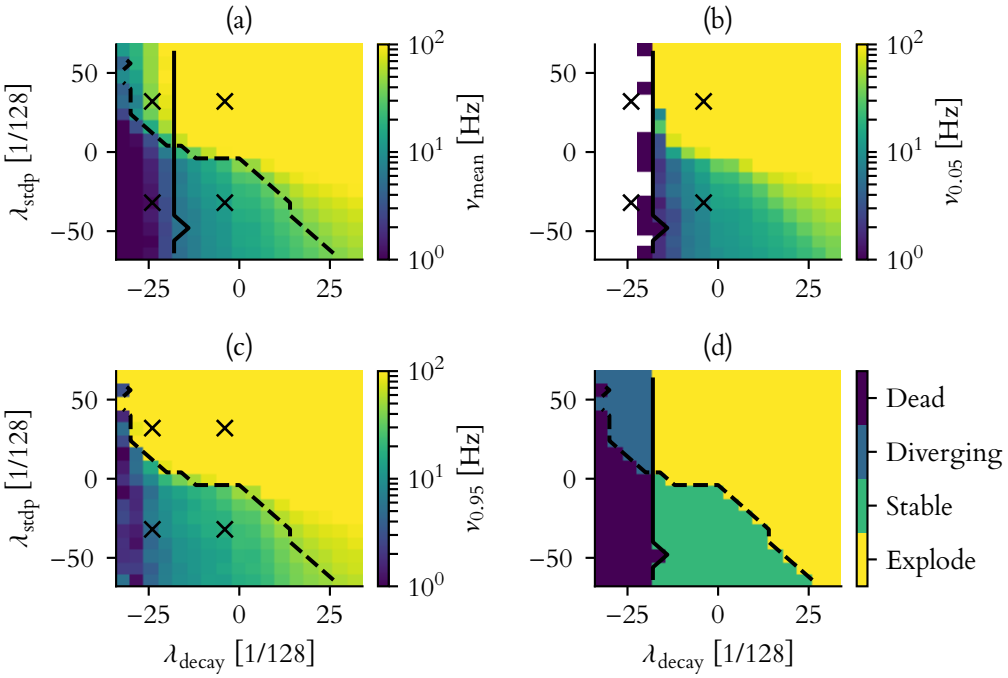

Figure 3.16.: (a) Average firing rate over three experiments for different plasticity parameters in the λ_{stdp}–λ_{decay}–plane. Figure (b) and (c) show the 5 % and 95 % quantile of the neurons' firing rates over all three repetitions for each parameter combination. The transparent data points in (b) show where the 5 % quantile of the firing rates is zero and therefore not representable on the logarithmic frequency scale. The solid and the dashed lines indicate where $\nu_{0.05}$ crosses 1 Hz and where $\nu_{0.95}$ crosses 50 Hz. These lines separate the space into the different phases—shown in (d). The black crosses are shown for later reference.

three trials with the parameter combinations marked in figure 3.16 by the black crosses. The spike patterns can nicely be related to the names of the different phases—dead, diverging, stable and explode. Still, also within the different phases the spike plots may look differently for different parameter combinations which cannot be seen from these examples. For example, the ratio from exploding to dying neurons in the diverging phase varies with respect to the λ_{stdp}–λ_{decay} parameters.

Log–spaced histograms over the spiking frequencies of all neurons in three trials for the exemplary parameter combinations are shown in figure 3.18. The outermost bins hold all occurrences of frequencies which are outside of the vertical dashed lines. In particular, a neuron that never fired would fall into the leftmost bar. For the dead case in figure 3.18a still some neurons fire rarely while most of the neurons never fire at all. With the diverging parameter combination most of the neurons either fire at a very high rate or at a very low rate

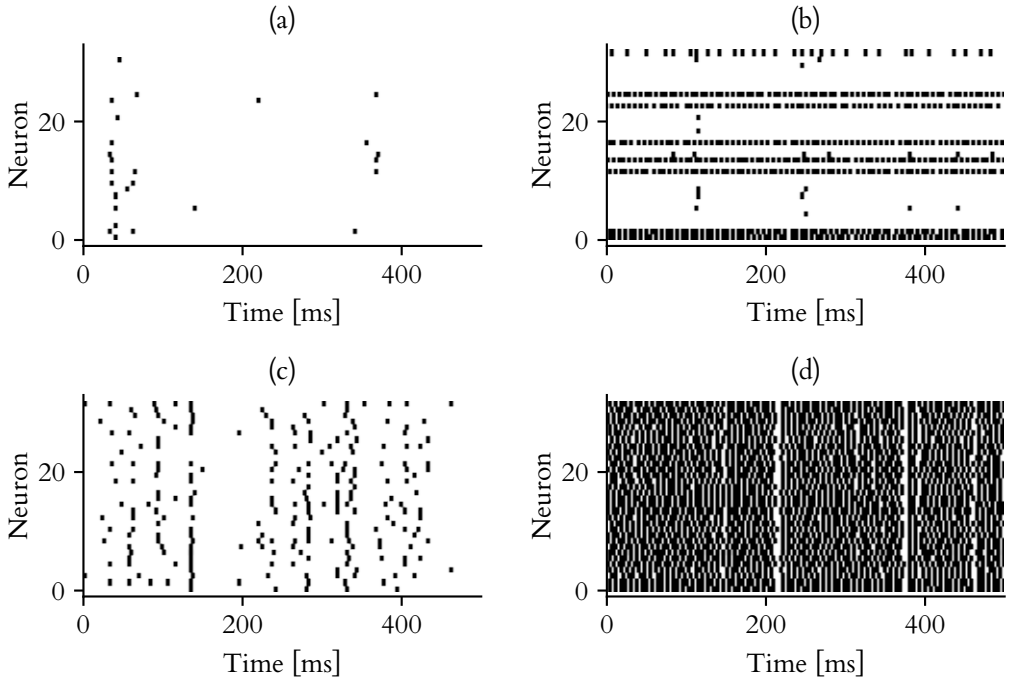

Figure 3.17.: Spikes during the last second of the emulations for the parameter combinations marked in figure 3.16 with black crosses. The spike pattern of the first of the three repetitions is displayed. Figure (a) to (d) are for the phases *dead*, *diverging*, *stable* and *explode*.

(cf. figure 3.18b). Only very few neurons spike at a rate within 1 Hz to 50 Hz. Contrary, in the histogram in figure 3.18c all neurons have a firing rate within this range. In the exploding phase (figure 3.18d) all neurons fire at a rate above 50 Hz.

As expected the dynamics change heavily depending on the parameter combinations of λ_{decay}–λ_{stdp}. Though designed to also yield stabilizing plasticity with moderate firing rates, it is remarkable how large the stable parameter space is. For strong, anti–Hebbian plasticity with $\lambda_{\text{stdp}} = -64/128$ the decay factor may vary from $-24/128$ to $16/128$ while still observing moderate activity for all neurons. In particular, the factor may even change its sign and therefore changing the plasticity from converging to diverging weights.

For Hebbian plasticity—"what fires together, wires together"—no stable, moderate firing is observed. The network either shows dead, diverging oder exploding activity dependent on the strength and sign of the decay.

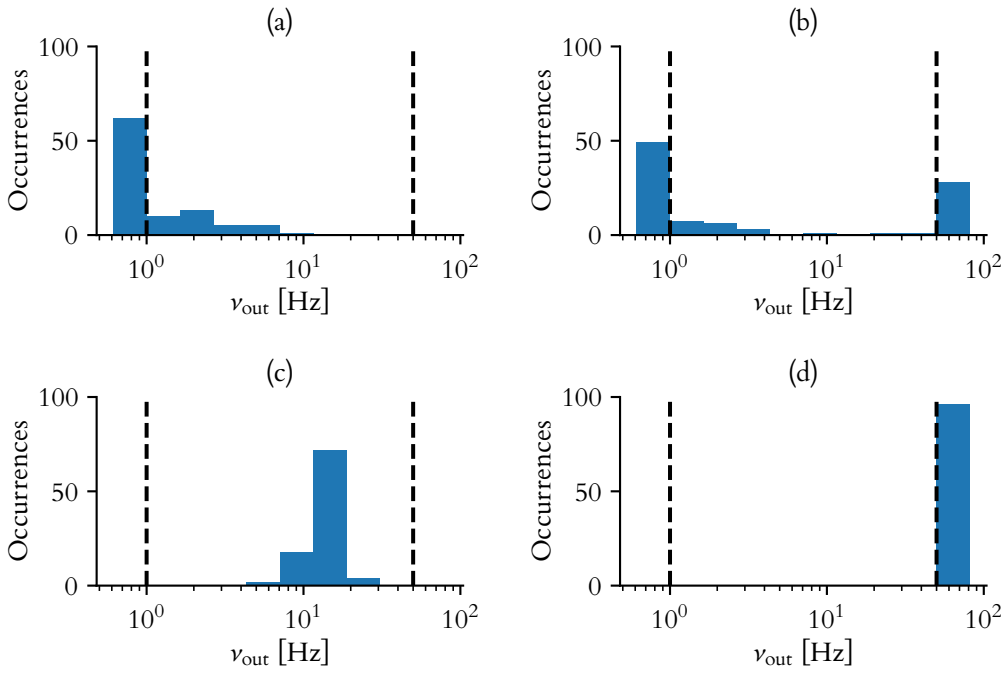

Figure 3.18.: Histogram over the measured frequencies of the postynaptic neurons for the exemplary parameter combinations. The outermost bins hold all occurrences of frequencies which are outside of the vertical dashed lines. Figure (a) to (d) are for the phases *dead*, *diverging*, *stable* and *explode*.

3.5. Unsupervised Orthogonal Pattern Learning

After observing this diversity of spiking dynamics, one of the most thrilling questions now certainly is: can this rule be used for learning? Here, one of the most simple pattern learning tasks is to be solved, where the neurons are expected to respond to either of two populations. There are two input populations of which at any point in time one is silent while the other is active. Can the neurons automatically adapt onto these two different input patterns and learn how to discriminate among them? On the one hand the plasticity algorithm is known to generate moderate spiking activity for anti–Hebbian plasticity where all neurons are at a favorable working point: neither silent nor close to the largest possible firing rate. This even holds for extremely different initial weights where all neurons are either spiking or silent. On the other hand this plasticity algorithm is known to penalize correlated spiking where the postsynaptic spikes come closely after the presynaptic spike. For pattern learning the strengthening of correlated spiking is favored. The neurons should respond quickly as soon as the learned pattern is active. The parameter sweeps of the previous section 3.4 however suggest that the resulting dynamics may not be robust for example against different initial conditions.

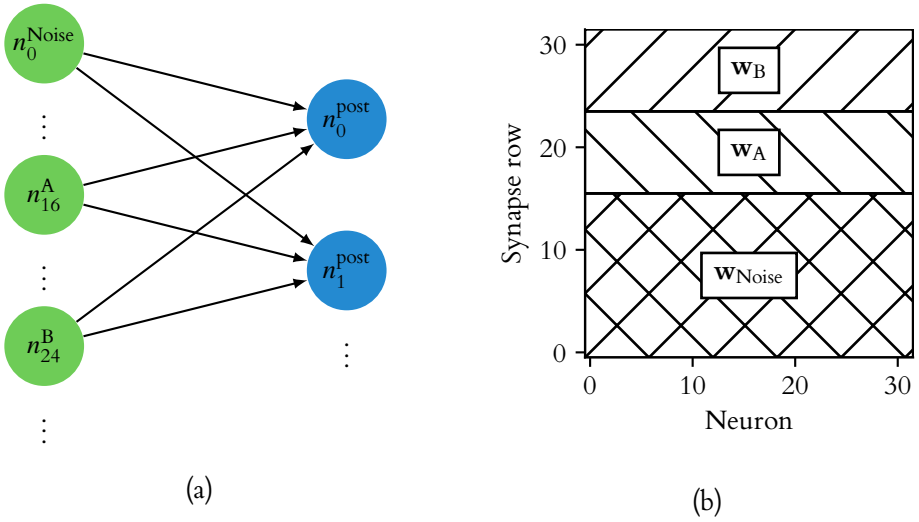

Figure 3.19.: (a) Sketch of the network for the unsupervised pattern learning experiments. There are three populations as spike sources: the noise population n^{Noise} and the A– and B–populations. (b) Partition of the synaptic weight matrix for this experiment on the DLS.

What about combining both types of plasticity? While anti–Hebbian synapses from random spike sources shift the individual neurons towards a state of responsive moderate spiking activity there are Hebbian synapses that reinforce the correlated spiking together with the input patterns. Broadly speaking, the anti–Hebbian plasticity stabilizes, the Hebbian plasticity learns. The network topology for this experiment is shown in figure 3.19a. There are three populations of external spike sources. The noise population n^{Noise} consists of 16 neurons that spike randomly with an average rate of 30 Hz. The A– and B–populations n^{A} and n^{B} are populations of 8 neurons which alternate every 250 ms to fire at a frequency of 60 Hz. If one population fires, the other population is silent. These populations A and B are in the following referred to as the *pattern populations*. Each of the 32 neurons is connected to these three populations. Figure 3.19b shows the partition of the synaptic weight matrix into the weights from the noise and the pattern populations.

While the synaptic connections from the noise feature anti–Hebbian plasticity with $\lambda_{\text{stdp}} = -16/128$ the connections towards the pattern populations evolve according to Hebbian plasticity with $\lambda_{\text{stdp}} = 32/128$. The decay of the synaptic weights from the noise population decays with $\lambda_{\text{decay}} = -4/128$. For the synapses from the pattern populations the decay is much stronger with $\lambda_{\text{decay}} = -32/128$. These plasticity parameters for the synapses from the noise population were the same as previously used when analyzing the stability in section 3.2.

The evolution of the average weight from the noise and the pattern populations per neuron is shown in figure 3.20. Initially, all synaptic weights are set to zero. Figure 3.20a shows the connection strength towards the noise population. At the beginning of the experiment

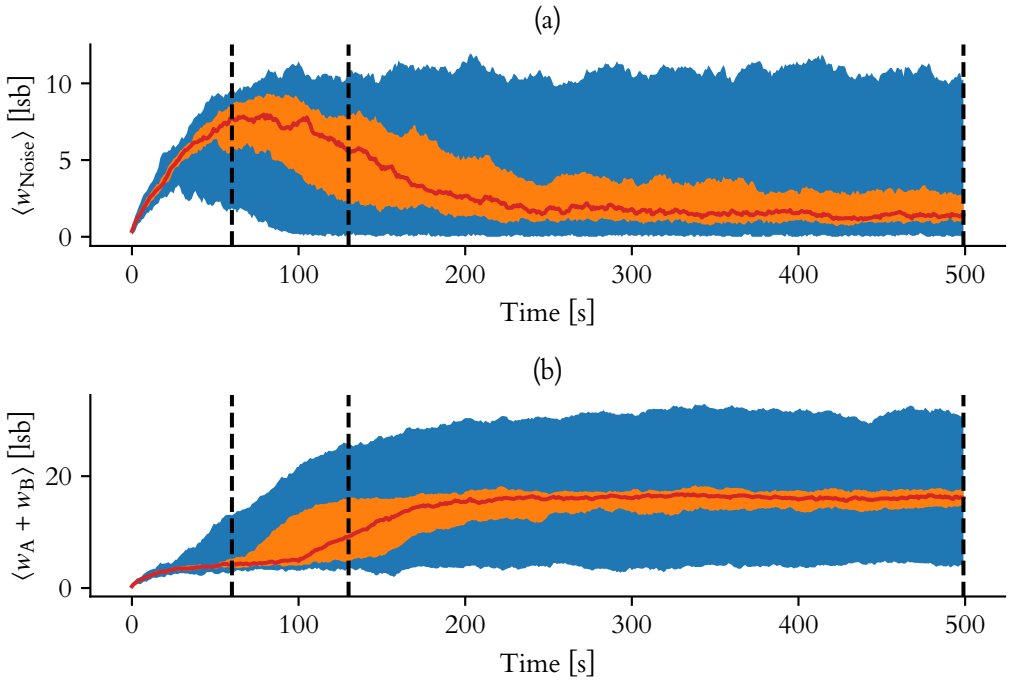

Figure 3.20.: (a) Evolution of the weights towards the noise spike sources. The red line represents the median weight, the orange range the 25 % to 75 % quantile and the blue area shows the total range. (b) Evolution of the weights towards the A– and B–population. The color code is the same as for (a). The dashed lines are for later reference.

these weights increase to stabilize the firing rate onto a moderate level, as already analyzed in section 3.2. As soon as the neurons reach a moderate firing rate the synapses from the pattern populations with Hebbian plasticity start to grow. This can be seen in the lower plot, figure 3.20b. The potentiation of these Hebbian synapses causes the neurons to fire even stronger. Therefore, the synaptic connections to the noise population decrease. At the end of the emulation synapses with Hebbian and anti–Hebbian plasticity balance onto a stable equilibrium. As later shown in figure 3.23, the neurons also fire with a moderate firing rate at this steady state.

Three exemplary weight matrices during this process are shown in figure 3.21. The first shows the weight matrix where the noise connections are still growing. Only few synapses with Hebbian plasticity have yet grown strong. The second weight matrix (figure 3.21b) shows the stage where the connections towards the pattern populations grow. The last matrix (figure 3.21c) shows the steady state at the end of the emulation. Most of the neurons—26 out of 32—formed strong incoming synaptic connections from one of the two pattern populations. Of the remaining neurons, three still have strong incoming connections from

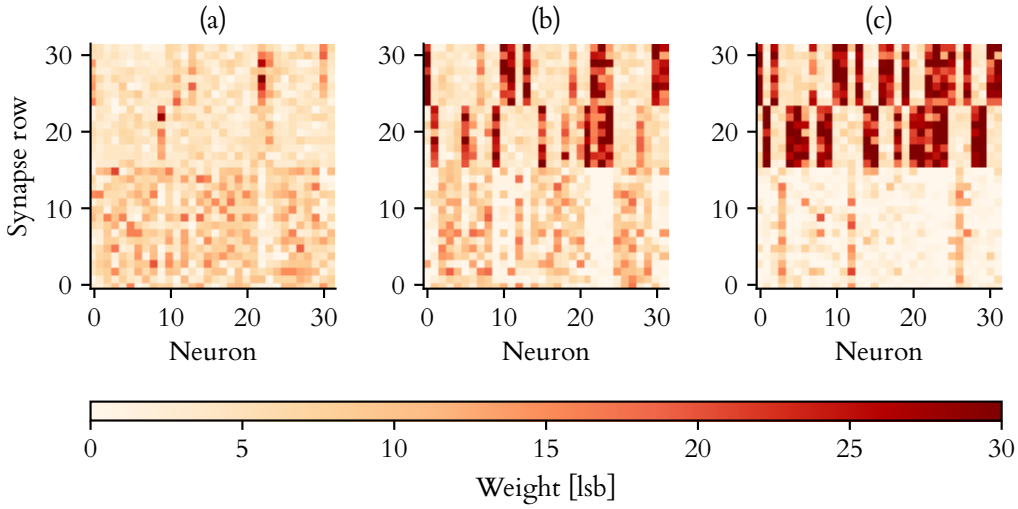

Figure 3.21.: The weight matrices after (a) 60 s, (b) 130 s and (c) 499 s. These times are also highlighted in figure 3.20.

the noise sources and no relevant connections from the pattern populations. The remaining three neurons connect strongly to both pattern populations.

This evolution of the weights in more detail is shown in figure 3.22. It covers most of the interesting aspects. The x– and y–axis show the average incoming weight from the A– and B–population for each neuron. Each line in the plot is the evolution of the weights for one of the 32 neurons. All neuron starts at $w_A = 0$ and $w_B = 0$. The change of the weights during the emulation is shown by the arrows in the background. They show the average direction of movement over time in a smoothed way, averaged over all traces. Lastly, the color encodes the average weight towards the noise population.

The movement of the incoming weights—as nicely visible in this graph—starts for all neurons at zero and moves to a plateau at high w_{Noise} and relatively low w_A and w_B. This is also seen in figure 3.20. From this plateau the incoming synaptic connections may chose one of four paths:

- Couple strongly either onto the pattern A or B. As soon as the symmetry is broken, the weights towards the preferred input quickly increase while the synaptic connections towards the noise decrease. This process is believed to be governed mostly by the stochasticity of the synaptic plasticity and the fixed pattern noise of the strength of the correlation sensors.
- Couple to both inputs.
- Stay on top of the noise plateau. This cannot be seen in this graph but is known from the figure of the final weight matrix (figure 3.21c).

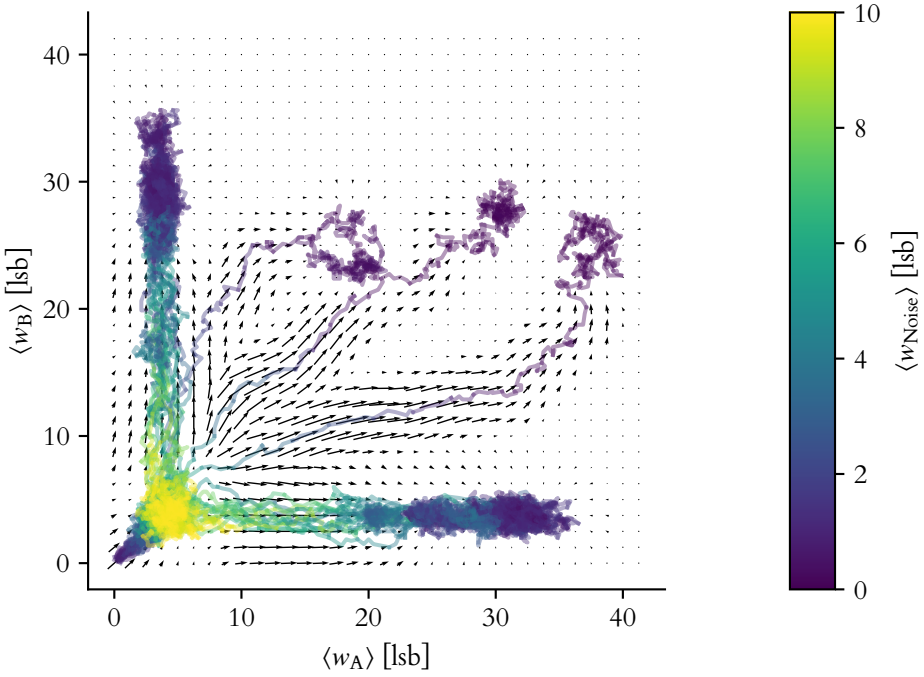

Figure 3.22.: Average incoming synaptic weight w_A, w_B and w_{Noise} for each neuron. One line represents the weights for one neuron during the emulation. The arrows in the background show the smoothed movements of the average weights during the emulation.

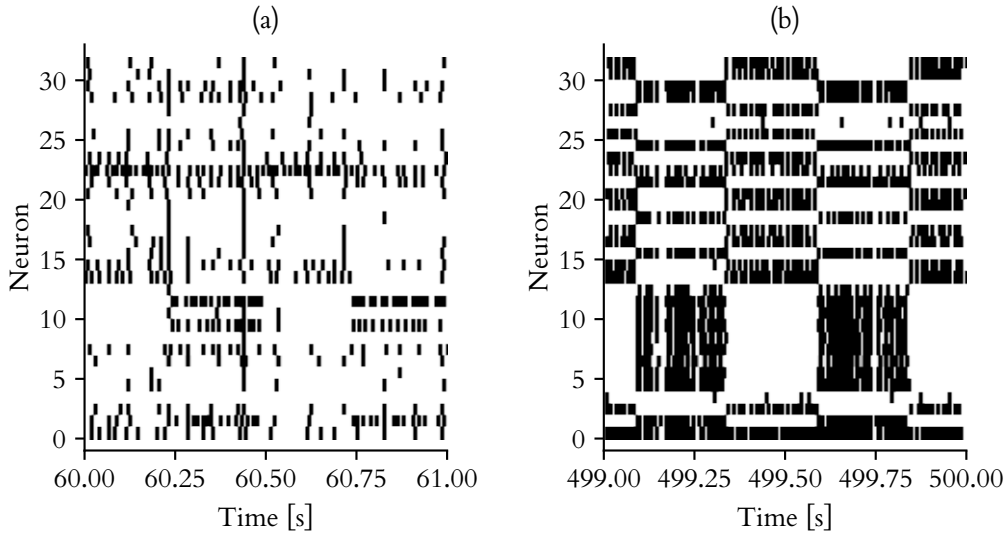

Figure 3.23.: Spike output of all postsynaptic neurons after 60 s and 499 s.

This visualization indicates that the final weights with coupling towards the pattern populations are stable while coupling solely to the noise is meta stable.

The firing patterns of the postsynaptic neurons at the beginning and the end of the experiment is shown in figure 3.23. It is recorded from a *different* trial than the previous plots analyzing the weights. This is done because of technical reasons. For reading out the synaptic weight matrix, the experiment needs to be stopped, which is done after each plasticity update for the previous measurements. The spikes, however, can be recorded without interfering with the experiment. Therefore, the spike patterns in figure 3.23 are a check that the qualitative result is the same as with the short interruptions of the experiments as for all previous figures in this section. The spike pattern in figure 3.23b is recorded after 60 s. At this time the onset of the Hebbian plasticity happens. The neurons fire at an average rate of 14.7 Hz. While most of the neurons seem to fire randomly some neurons already show a preference for one of the pattern populations, e.g. neuron 11. At the end of the experiment shown in figure 3.23b most of the neurons strongly respond to either of the pattern populations. On average—including the silent periods—the neurons fire at a rate of 44.2 Hz.

Summarizing so far, this network topology and synaptic plasticity not only results in a network with stable, moderate firing of all neurons. Furthermore, most of the neurons learn to respond to either the A or the B input pattern. This coupling to the patterns is governed by local plasticity rules without any teacher signal. The network performs unsupervised pattern learning using local plasticity rules without the need of inhibitory synapses.

It is known, that the stabilization by the noise synapses is not too sensitive to the plasticity parameters as shown in section 3.4. However, the question arises how sensitive the performance is towards changing the plasticity parameters of the learning synapses $\lambda^{\mathrm{A,B}}_{\mathrm{stdp}}$ and $\lambda^{\mathrm{A,B}}_{\mathrm{decay}}$.

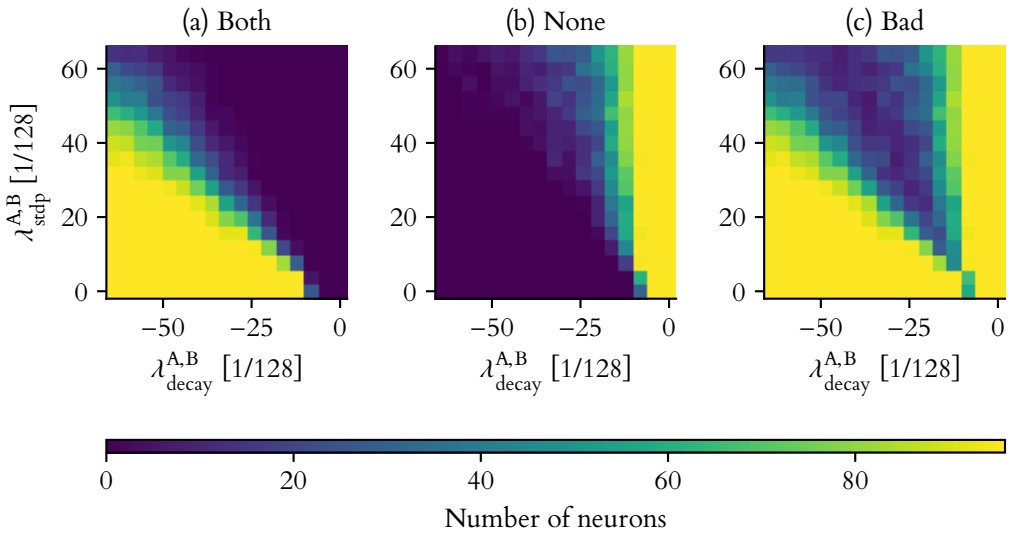

Figure 3.24.: (a) Number of neurons in three trials that couple to neither of the patterns. (b) Number of neurons that couple to both patterns. (c) Total number of "bad" neurons in three trials that either respond to both populations or to neither population.

The superscript "A,B" denotes that parameters of the synapses from the pattern populations are addressed. Figure 3.24 shows the pattern recognition performance within this parameter plane. The performance is measured by how many neurons are in "bad" final states:

- **Coupling to both patterns**, shown in figure 3.24a. The average incoming weights w_A and w_B are both above 12 lsb.
- **Coupling to neither pattern**, shown in figure 3.24b. The average incoming weights w_A and w_B are both below 12 lsb.

Lastly, figure 3.24c shows the total number of "bad" final states. The threshold of 12 lsb is chosen as in figure 3.22 these values appear to discriminate well between coupling to noise and coupling to the patterns. This measure of the pattern recognition performance is chosen because of its simplicity. For each data parameter combination the emulation is repeated with three different random seeds such that 96 examples of neurons are available.

There are three regimes observed in the parameter space. For large negative decay factors and low STDP strengths the weights from the pattern populations decay faster than the STDP term could strengthen them. The neurons still fire at a moderate rate but couple only to the random spike sources. Decreasing the strength of the decay changes the behavior such that neurons are able to form stable synaptic connections from the pattern populations. As soon as these connections become stable, there are also neurons that respond to both patterns.

With a further decreasing decay all neurons would respond to both pattern populations almost independent of the STDP strength.

There is no perfect set of parameters where all neurons connect reliably either to A or to B. Still, the parameter range where most of the neurons show the desired behavior is quite large. For example, the STDP strength of the previous measurements in this section could be doubled while still most of the neurons perform well on the classification task.

4. Discussion and Conclusion

While the initial results look promising, a critical discussion is appropriate. The individual experiment results are discussed in section 4.1. Next, advantages and disadvantages of the DLS neuromorphic computing platform and the available software framework for these experiments are discussed in section 4.2. Section 4.3 addresses the question whether the presented experiments are ready to be scaled to larger systems currently under development. The crucial question whether any predictive power or biological relevance comes with the presented findings is addressed in section 4.4.

4.1. The Experiments

Stability

The goal of finding a local synaptic plasticity rule that stabilizes the neural activity in a feed–forward network was achieved. Even for extreme initial conditions at either end of the scale (i.e. a strongly connected or a completely disconnected network), the neurons reach a stable baseline of spiking activity by this homeostatic plasticity mechanism. This stable baseline is at an average firing rate of 17 Hz for the tested plasticity parameters in section 3.2. This working point provides the dynamic range for each neuron to increase or decrease the firing rate. The average weight and firing rate at the steady–state are the same for all initial conditions. It was observed furthermore, that the steady–state weights are biased by the strength of the individual correlation sensors in the synapses and also biased neuron–wise (see figure 3.9). Synapses with a more sensitive correlation sensor tend to have a lower steady–state weight. Also the average weight of all incoming synaptic connections varies significantly from neuron to neuron from (4.7 ± 0.3) lsb to (7.9 ± 0.4) lsb. This is supposedly due to the fixed pattern noise in the strength of the synaptic inputs. Both biases are undesirable: ideally all synapses and neurons are exactly the same and there is no preference for one over another. As this contradicts the theoretical model, this questions the predictive power of the results regarding the model. This is addressed in section 4.4.

Recurrence

The influence of recurrence on the network dynamics is analyzed in section 3.3. For all experiments the recurrence changes the neural dynamics significantly. It was shown that an increased recurrence adds memory and sensitivity to the networks dynamics. These two aspects, memory and sensitivity, are however only analyzed on a very basic level, namely on the level of the total network activity. The memory was characterized by the width of the autocorrelation of the networks' activity. It therefore measures how long it takes

for the system to forget a high and low firing activity state. A more thorough analysis of the memory capacity is yet to be done for example by integrating the mutual information between any pair of network states as done by Natschlaeger and Maass [2004]. This method would also reveal information stored as repetitive occurring patterns where the overall firing activity is constant.

The sensitivity of the presented recurrent neural networks was analyzed with different methods:

- The distance in the individual spike trains of two repetitions of exactly the same experiment. This not only measures the perturbation of the total firing rate over all neurons but also on the level of individual neurons and spikes.
- The increase in the time–resolved firing rate and the total number of spikes when injecting a stimulation pulse.
- The change in the firing rate and the gain factor when varying the rate of the random input spike trains.

All measurements showed that the sensitivity of the network increases with an increasing degree of recurrent connections as expected with the exception of the short–term response onto the pulse stimulus. In the time resolved pulse–response the feed–forward network showed a faster and stronger increase in the firing rate, than the networks with higher degrees of recurrence. However, as indicated by the increasing measures of sensitivity, it was not possible to generate a self–sustained network without the input of random spikes for these specific neuron parameters. The discrete plasticity updates were found to over– and undershoot even for very weak factors in the plasticity rule as the network activity changed extremely with very small changes in the synaptic weights. A self–sustained network may be achievable in a larger system, where the spikes from distant populations may serve as independent random spike sources.

Plasticity Parameter Space

In section 3.4 the parameter space of the plasticity algorithm was partly explored and analyzed regarding the resulting dynamics of the system. The parameters chosen for the exploration are the decay and the STDP scaling factor. These parameters were considered to be the most interesting: The decay factor heavily determines the fixed point of the OU process of the weights and whether it is stable at all. The STDP scaling factor on the other hand determines to which degree the dynamics of the weights are governed by this random walk instead of the pre–post spike time correlations ranging from Hebbian to anti–Hebbian plasticity. Consequently, the observed dynamics vary strongly for those different regimes: the parameter space could be separated into regions where the activity dies out, explodes, stabilizes onto a moderate firing rate or diverges among the different neurons. It is remarkable how large the parameter space is in which stable activity is observed: For example for $\lambda_{\text{stdp}} = -64/128$ the stable region ranges from $-24/128 \leq \lambda_{\text{decay}} \leq 16/128$.

However, this was tested for a fixed bias with a fixed STDP time constant. If these fixed parameters are changed the dynamics are also expected to change. For large biases, the

weights would continuously grow unless a very strong anti–Hebbian term counterbalances. Very low—in particular negative—biases would on the other hand force the neural network into a disconnected state. It would be even more interesting to measure the effect of the STDP time constant on network dynamics. For $\tau_{\text{stdp}} \to 0$ the dynamics are expected to be the same as for $\lambda_{\text{stdp}} = 0$: with a vanishing time constant hardly any correlations would be measured at all and therefore no update due to correlated spiking would take place. On the opposite for $\tau_{\text{stdp}} \gg \tau_{\text{mem}}, \tau_{\text{syn}}$ the correlation measurement is expected to not be meaningful anymore on how much an individual presynaptic spike contributed to the postsynaptic spike. Instead, the correlation measurement is rather a measurement of the overall spiking activity $\nu_{\text{pre}} \cdot \nu_{\text{post}}$ that may relate to rate based models, e.g. by Oja [1982]. Since all software tools are readily available for this study, experimental results should be straightforward to obtain.

Pattern Learning

The application of the proposed plasticity algorithm for simple pattern learning was shown in section 3.5. The neurons are connected to input populations presenting the patterns with Hebbian synapses while the stability is provided by anti–Hebbian synapses from random spike sources. The presented patterns however are simple: Either population A or population B fires at a rate of 60 Hz. If a neurons responds to either of the pattern populations it is likely that the other synapses from the same population measure a correlated spike pair. It was observed that most of the neurons couple to either of the pattern populations, instead of coupling to neither or both. This highly desirable behavior is not yet fully understood though the attempt of an interpretation is given: If there is any spontaneous symmetry breaking due to the randomness of the synapses or the fixed–pattern noise such that a synapse would by chance strongly connect to population A or B, the other synapses would follow. As the firing rate quickly increases, the weights from the noise decrease and random postsynaptic spikes during the non–preferred pattern become more and more unlikely. The noise population can be seen as shifting the baseline of the neural activity onto a moderate rate. As soon as a neuron fires due to the growing Hebbian synapses this baseline vanishes again. This may be seen as that the shifted baseline gets inhibited when coupling to a pattern.

There are however serious limitations on this kind of pattern learning if the underlying processes are well described by the given interpretation. To couple either to the A or B pattern is explained by symmetry breaking due to the stochasticity in the synaptic strengths. This symmetry breaking becomes more difficult for an increasing number of synapses. In the presented experiment are eight synaptic connections to each of the pattern populations which is quite few. If there are much more synapses the relative variations in the average synaptic strengths over these pattern populations are much smaller. Therefore, it is expected that the neurons would then always couple to both patterns or neither of both patterns.

The parameter space in figure 3.24 shows the parameter combinations for which the two patterns are well discriminated. This region where the classification performs well is expected to change for different numbers of patterns. It is expected that for more patterns, the synaptic noise connections need to decrease even faster to not couple to more than one pattern. Ideally, there are parameters for which any number of patterns is well discriminated.

This may be realistic if there is a process which either allows for escaping the state of coupling to multiple patterns or which penalizes if many neurons couple to the same pattern. In the given interpretation none of both is present. Introducing lateral inhibition is expected to be a promising approach to prevent too many neurons from coupling to the same pattern. This may also contribute to the process of symmetry breaking in a favorable way: if any neuron coupled to one pattern, the inhibited neurons are expected to quickly connect the opposite pattern.

4.2. Usability of the DLS and the Software Tools

In addition to the measurements presented in this thesis, numerous software and hardware tests were performed on the DLS. Having learned a lot about the DLS, this section will discuss the usability of the current system and software tools.

First of all, the DLS is a viable platform for studying LIF neurons under the perpetual influence of synaptic plasticity. The calibration database by Stradmann [2016] allowed for configuring the neurons reliably with the targeted LIF parameters. Performing these experiments without any calibration turned out to be feasible, too [Stöckel et al., 2017] but comes at the cost, that the predictive power of the results is limited as the realized LIF parameters do not match the targeted parameters of the model.

The emulation speedup of ca. 1000 compared to biological time scales allowed for the large parameter sweeps presented in section 3.4. Even though the current operation software has not been designed for performance, the fast emulation speed of the hardware system can be exploited. One repetition of the sweeps in figure 3.16 takes 51.6 min wall–clock time while 16.1 h of biological time are emulated. Though not measured rigorously, the preparation of the spike trains in Python, the communication with the FPGA and the conversion of the resulting spikes into numpy–arrays are known to be the most time consuming tasks for these experiments. Especially the process of input spike preparation and conversion of the resulting spikes are inefficient for–loops in Python. For example, adding the spikes to the playback program for an experiment of 200 s length is done with a for–loop with 2×10^5 repetitions. Within this loop there is also an if–else statement. Both, for–loops and if–else statements are known to cause a substantial interpreter overhead [Python Wiki, 2017]. These time consuming functions can easily be implemented efficiently in the `frickel-dls` C++ layer.

The hybrid approach—an analog neuron and synapse implementation along with the digital PPU for synaptic plasticity—proved to be flexible in terms of plasticity rules if the rules restrict to the different observables that are available to the processor. While the presented synaptic plasticity algorithm takes the current weight, spike time correlations and randomness into account the PPU also has access to per–neuron rate counters as well as the whole chip configuration. For example, changing the neurons' individual thresholds according to the current firing rate is straight forward to implement.

The simple neuron and synapse model together with lots of on–chip debugging features allow for fine–grained testing and debugging. For example, switches are provided to enable and disable components or switching different analog signals for off–chip readout In the author's point of view, the current simplicity of the implemented model is a great advantage

as it limits the parameter space of possibly faulty configurations which are likely to occur and still hard to debug. Even given the current model, a huge amount of time was spent on testing and making the host software feature complete on supporting all of the chips functionalities. The author's vision for the future development is a system scaled up in the number of neurons and synapses keeping these simple models.

When writing this thesis, the host software only supported an explicit chip configuration at the level of functional units (see section 2.3) but no abstraction onto the neural network level as PyNN [Davison et al., 2008]. As the current chip is a prototype and still under development, this is also not planned. However, to improve the current usability and to hide complexity from the user the full host software is currently being refactored by different developers[1]. The experience collected during the experiments presented in this thesis turned out to be very valuable for this process in various ways:

- It deepened the understanding of the chip and baseboard configuration options and therefore improved the future container and coordinate naming.
- Drawbacks in the control flow and programming design patterns of the current software were identified.
- The typical use cases when performing experiments are now taken into consideration for the design of the Application Programming Interface (API).

The task of designing a convenient interface for using the PPU and its vector extension for synaptic plasticity experiments is yet to be solved. There are different competing requirements to be balanced:

- The generated kernel code should be efficient. It should in particular efficiently use the vector extension.
- Any complex synaptic plasticity algorithm that is supported by the PPU should be realizable with the interface, e.g. access to all observables.
- The complexity of the software that exposes the synaptic plasticity to the user should be low.

As a short term solution the author suggests to expose a library of hand–crafted plasticity algorithms that may be parameterized by the users. This, on the one hand, limits the users to a fixed set of plasticity rules, but on the other hand allows for plasticity experiments with minimal effort. In the long term, a code generator which transforms a computational graph on the observables into performant PPU programs would enable users limited low–level programming knowledge to implement custom plasticity rules. This goal requires a high complexity of the software which exposes the plasticity features. It is planned for PyNN to adopt to NineML [Raikov et al., 2011] for the formulation of the synaptic plasticity, hence this is also the favored solution as the front–end for the code generation.

[1] This refactoring is ongoing work by Christian Mauch and Johann Klähn with support from Eric Müller, Yannik Stradmann and the author.

In summary, Friedmann et al. [2017] presented the first device where STDP could be studied in a flexible manner. The advantages of this new technology were utilized for this work to analyze synaptic plasticity with regard to the stability of the neural activity. The fast emulation speed allowed for large systematic parameter sweeps to study the diverse phenomenology of the resulting system dynamics under different parameterizations of the plasticity rule. Performing the experiments came at the cost of a large software development effort. If the software tools also advance along with the physical devices they will together become an important tool for studying spiking neural network dynamics.

4.3. Scalability to Larger Future Systems

Scaling the experiment to larger systems was the main motivation when designing the proposed plasticity algorithm. Yet, the claim of scalability of the current experiments is yet only partly true: while the plasticity algorithm can run independently on many interconnected chips (as planned for the DLS wafer–scale system) the calibration effort for the neurons and the individual synaptic correlation sensors can become increasingly complex and time consuming. For the current BrainScaleS wafer–scale system [Schmitt et al., 2017] the calibration roughly takes 160 h [Kugele, 2017].[2] From the author's point of view however, it is also worth trying to repeat the presented experiments without calibration of the neurons and synapses. The main effort then is to find a parameter set for the neurons and synapses at which at least most of the neurons are responsive. If still a baseline of stable activity is reached in an extremely large system, this would be a major achievement as it may allow for observing collective phenomena on a much larger scale. For example slow oscillations of neural activity and slow oscillations in the synaptic strengths cannot be observed in the small network of 32 neurons but are candidates for possible large–scale, self–organized phenomena. As already discussed before, generating a network with a self–sustained activity is more promising to achieve in larger neural networks. Furthermore, does the plasticity self–organize the large–scale network topology into a state that performs interesting transformations on any structured input patterns?

4.4. Predictive Power and Biological Relevance

In the following two important questions will be addressed: To which extend are the emulation results fundamental properties of the theoretical model instead of rather being undesired side–effects of the physical model? And does any biological relevance come with the presented work?

In regard to the first question it is already shown in figure 3.9 that for this experiment the steady–state weights are biased by fixed–pattern noise in the physical implementation: For one, they are biased by the strength of the correlation sensor as synapses with a stronger correlation sensor tend to have a lower final steady–state weight. Furthermore, some neurons

[2] The single chip calibration takes for the HICANN chip ≈5 h. One wafer contains at most 384 HICANNs of which 12 can be calibrated in parallel in the current system configuration.

have a significantly higher or lower average weight over all incoming synapses than other neurons, presumably biased by different strength of the synaptic inputs. This is a clear contradiction to the theoretical model presented in section 2.1 where all synapses and neurons are equally strong and responsive, i.e. no bias is expected. For the other experiments one must therefore assume that the results are biased, too.

While—to the author's best knowledge—there are no implementation errors, it is sensible to discuss the bias introduced by possible implementation errors. Obtaining wrong or biased results did happen during the work for this thesis despite thorough testing. The control over the neuromorphic chip and the PPU happened at a very low level and therefore it is possible that some detail was overlooked. To check the implementation and the obtained results these experiments should be repeated with software simulations which was not yet done.

Even if the results were verified with software simulations the second question remains, whether these results are biologically relevant. The chosen theoretical model—LIF neurons with current–based synapses—is a very simple model where relevant mechanisms of the brain may not be covered. Furthermore the tested synaptic plasticity rule is only one of many different forms proposed in literature. However, the author suggests that this work supports the hypothesis that the dynamics on the level of multiple neurons are not subject to the exact implementation of all features on the lowest level. Instead, sweeping the large parameter space (see section 3.4) shows that the phenomenology changes when changing the key features of the plasticity algorithm: changing from Hebbian to anti–Hebbian plasticity and changing from a globally attractive fixed point of the weights to a repulsive, diverging behavior. Also the dynamics changed strongly when changing the network's topology from feed–forward to a recurrent neural network in section 3.3. Identifying such key features that lead to fundamental changes in the network dynamics can indeed contribute to the understanding of the mechanisms found in biology.

In summary, the relevance of this work regarding biology is that anti–Hebbian plasticity may act as a homeostatic mechanism. It adjusts the weights to form a stable network, while keeping the synapses' responsiveness from spike to spike, contrary to short–term plasticity. This mechanism was shown to bring the network into a ready–to–learn state and then retreating in a self–organized fashion as soon as functional plasticity takes over.

Acronyms

ADC Analog to Digital Converter.

API Application Programming Interface.

DAC Digital to Analog Converter.

DLS Digital Learning System.

FPGA Field Programmable Gate Array.

GCC GNU Compiler Collection.

ISA Instruction Set Architecture.

LIF Leaky Integrate–and–Fire.

lsb Least Significant Bit.

ODE Ordinary Differential Equation.

OTA Operational Transconductance Amplifier.

OU Ornstein–Uhlenbeck.

PPU Plasticity Processing Unit.

STDP Spike–Timing–Dependent Plasticity.

UNI Universal Neuromorphic Instruction set.

VRF Vector Register File.

Appendix

A. Frickel-DLS Software Changes

In order to perform the experiments presented in this thesis the host software had to be extended. The major extensions are:

- Integrate into the SLURM ressource management.[3]
- Integrate the spike routing configuration.[4]
- Make the chip abstraction feature complete.
- Create a full description of the experiment environment.
- Implement a serializable chip and baseboard configuration.[5]
- Add the integration of the calibration.

All measurements presented in this work were performed with the host software at the git commit `602b85ac7a8bd7c441edcc25a6bdc56bacc04ce8`. This commit is the 2nd patch set of the change id `If483fedd12abf8693eb0dfaf927059bbfa5c6877`.

[3] Thanks to Christian Mauch.
[4] Thanks to Christian Pehle.
[5] Thanks to Eric Müller.

B. Implementation of the Plasticity Update

The implemented plasticity algorithm is listed in listing 1 without further discussion in detail. It is listed as a reference on how the in–line assembly syntax of GCC is used correctly interfacing between C and assembly. Further information, especially on the different register constraints, is given in the manual by [Stallman and the GCC Developer Community, 2003].

Listing 1: Implementation of the plasticity update with GCC in–line assembly

```
void update_weights(
        uint8_t const factor_stdp,
        uint8_t const factor_decay)
{
    // Initialize vectors with the constant factors
    vector uint8_t const resets = vec_splat_u8(
            resets_causal | resets_acausal);
    vector uint8_t const factors_stdp = vec_splat_u8(factor_stdp);
    vector uint8_t const factors_decay = vec_splat_u8(factor_decay);
    vector uint8_t zeros = vec_splat_u8(0);
    vector uint8_t update_scales = vec_splat_u8(32); // = 1/4

    // Loop over all vectors of synapses
    for (uint32_t index = 0; index < dls_num_syn_vectors; index++) {
        // Create the random numbers
        uint8_t random[16] __attribute__ ((aligned (16)));
        make_random(random);

        // Declare temporary variables
        register vector uint8_t temps;
        register vector uint8_t updates;
        register vector uint8_t weights;
        asm volatile (
            // Get stdp update
            "fxvinx %[temps], %[dls_causal_base], %[index]\n"
            "fxvshb %[temps], %[temps], -1\n"
            "fxvmulbfs %[updates], %[temps], %[factors_stdp]\n"
            // Reset correlation measurement
            "fxvoutx %[resets], %[dls_causal_base], %[index]\n"
            // Load the shifted weights
            "fxvinx %[weights], %[dls_weight_base], %[index]\n"
            "fxvshb %[weights], %[weights], 1\n"
            // Add decay update
            "fxvmulbfs %[temps], %[weights], %[factors_decay]\n"
            "fxvaddbfs %[updates], %[updates], %[temps]\n"
            // Add random update
            "fxvlax %[temps], 0, %[random]\n"
```

```
            "fxvaddbfs %[updates], %[updates], %[temps]\n"
            // Divide and add the updates to the weights
            "fxvmulbfs %[updates], %[updates], %[update_scales]\n"
            "fxvaddbfs %[weights], %[weights], %[updates]\n"
            // Set to zero if the result is smaller than 0
            "fxvcmpb %[weights]\n"
            "fxvsel %[weights], %[weights], %[zeros], 2\n"
            // Save shifted weights
            "fxvshb %[weights], %[weights], -1\n"
            "fxvoutx %[weights], %[dls_weight_base], %[index]\n"
            : [temps] "=&kv" (temps),
              [updates] "=&kv" (updates),
              [weights] "=&kv" (weights)
            : [index] "r" (index),
              [dls_causal_base] "b" (dls_causal_base),
              [dls_weight_base] "b" (dls_weight_base),
              [resets] "kv" (resets),
              [zeros] "kv" (zeros),
              [factors_decay] "kv" (factors_decay_1),
              [factors_stdp] "kv" (factors_stdp_1),
              [update_scales] "kv" (update_scales),
              [random] "r" (&random)
            : /* no clobbering */);
        }
    }
}
```

Bibliography

S. A. Aamir, P. Müller, A. Hartel, J. Schemmel, and K. Meier. A highly tunable 65-nm CMOS LIF neuron for a large scale neuromorphic system. In *ESSCIRC Conference 2016: 42nd European Solid-State Circuits Conference*, pages 71–74, Sept 2016. doi: 10.1109/ESSCIRC.2016.7598245. URL https://dx.doi.org/10.1109/ESSCIRC.2016.7598245.

L. F. Abbott and Sacha B. Nelson. Synaptic plasticity: taming the beast. *Nature Neuroscience*, 2000. doi: 10.1038/nn1100_1178. URL https://dx.doi.org/10.1038/nn1100_1178.

L. F. Abbott and S.B. Nelson. Temporal dynamics of biological synapses. In Michael A. Arbib, editor, *The Handbook of Brain Theory and Neural Networks*. MIT Press, Cambridge, MA, USA, 2nd edition, 2002. ISBN 0262011972.

M. Abeles. *Corticonics: Neuronal Circuits of the Cerebral Cortex*. Cambridge University Press, Cambridge, England, 1st edition, 1991.

Alfred V. Aho, Monica S. Lam, Ravi Sethi, and Jeffrey D. Ullman. *Compilers: Principles, Techniques, and Tools (2nd Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2006. ISBN 0321486811.

Guo-qiang Bi and Mu-ming Poo. Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type. *Journal of neuroscience*, 18(24):10464–10472, October 1998.

Romain Brette, Michelle Rudolph, Ted Carnevale, Michael Hines, David Beeman, James M Bower, Markus Diesmann, Abigail Morrison, Philip H Goodman, Frederick C Harris, et al. Simulation of networks of spiking neurons: a review of tools and strategies. *Journal of computational neuroscience*, 23(3):349–398, July 2007.

Nicolas Brunel. Dynamics of sparsely connected networks of excitatory and inhibitory spiking neurons. *Journal of Computational Neuroscience*, 8(3):183–208, 5 2000. ISSN 1573-6873. doi: 10.1023/A:1008925309027. URL https://dx.doi.org/10.1023/A:1008925309027.

Ioana Carcea and Robert C. Froemke. Cortical plasticity, excitatory–inhibitory balance, and sensory perception. *Progress in brain research*, 207:65–90, January 2013.

Andrew Davison, Daniel Brüderle, Jochen Eppler, Jens Kremkow, Eilif Muller, Dejan Pecevski, Laurent Perrinet, and Pierre Yger. PyNN: A common interface for neuronal network simulators. 2:11, 02 2008.

Rodney Douglas, Misha Mahowald, and Carver Mead. Neuromorphic analogue VLSI. *Annual review of neuroscience*, 18(1):255–281, 1995.

Paul W Frankland, Cara O'brien, Masuo Ohno, Alfredo Kirkwood, and Alcino J Silva. α-CaMKII-dependent plasticity in the cortex is required for permanent memory. *Nature*, 411(6835):309–313, May 2001.

Simon Friedmann. *Universal Neuromorphic Instruction set*. Universität Heidelberg, 2017. URL `ssh://git@gitviz.kip.uni-heidelberg.de/uni.git`.

Simon Friedmann, Johannes Schemmel, Andreas Grübl, Andreas Hartel, Matthias Hock, and Karlheinz Meier. Demonstrating hybrid learning in a flexible neuromorphic hardware system. *IEEE Trans. Biomed. Circuits and Systems*, 11(1):128–142, 2017. doi: 10.1109/TBCAS.2016.2579164. URL `https://dx.doi.org/10.1109/TBCAS.2016.2579164`.

Wulfram Gerstner and Werner Kistler. *Spiking Neuron Models: An Introduction*. Cambridge University Press, New York, NY, USA, 2002. ISBN 0521890799.

Debra A. Gusnard and Marcus E. Raichle. Searching for a baseline: Functional imaging and the resting human brain. *Nat Rev Neurosci*, 2:685–694, 10 2001. doi: 10.1038/35094500. URL `https://dx.doi.org/10.1038/35094500`.

Donald O. Hebb. *The organization of behavior: A neuropsychological theory*. Wiley, New York, 6 1949. ISBN 0-8058-4300-0.

Arthur Heimbrecht. Compiler support for the BrainScaleS plasticity processor. Bachelorarbeit, Universität Heidelberg, March 2017.

Matthias Hock. *Modern semiconductor technologies for neuromorphic hardware*. PhD thesis, Universität Heidelberg, July 2014.

Giacomo Indiveri, Elisabetta Chicca, and Rodney Douglas. A VLSI array of low-power spiking neurons and bistable synapses with spike-timing dependent plasticity. *IEEE transactions on neural networks*, 17(1):211–221, January 2006.

David Kappel, Bernhard Nessler, and Wolfgang Maass. STDP installs in winner-take-all circuits an online approximation to hidden markov model learning. *PLOS Computational Biology*, 10(3):1–22, 03 2014. doi: 10.1371/journal.pcbi.1003511. URL `https://dx.doi.org/10.1371/journal.pcbi.1003511`.

Alexander Kugele. Constraint satisfaction problem solved on the BrainScaleS system. Masterthesis, to be published, Universität Heidelberg, 2017.

H Markram, W Gerstner, and P J Sjöström. Spike-timing-dependent plasticity: A comprehensive overview. *Frontiers in Synaptic Neuroscience*, 4(2), 2012. doi: 10.3389/fnsyn.2012.00002. URL `https://dx.doi.org/10.3389/fnsyn.2012.00002`.

Henry Markram, Joachim Lübke, Michael Frotscher, and Bert Sakmann. Regulation of synaptic efficacy by coincidence of postsynaptic APs and EPSPs. *Science*, 275(5297):213–215, Janurary 1997.

Carver Mead. Neuromorphic electronic systems. *Proceedings of the IEEE*, 78(10):1629–1636, October 1990.

P. A. Merolla and K. Boahen. Dynamic computation in a recurrent network of heterogeneous silicon neurons. In *2006 IEEE International Symposium on Circuits and Systems*, pages 4539–4542, May 2006. doi: 10.1109/ISCAS.2006.1693639. URL `https://dx.doi.org/10.1109/ISCAS.2006.1693639`.

Kenneth D. Miller and David J. C. MacKay. The role of constraints in hebbian learning. *Neural Computation*, 6(1):100–126, 1994. doi: 10.1162/neco.1994.6.1.100. URL `https://doi.org/10.1162/neco.1994.6.1.100`.

S. Mitra, S. Fusi, and G. Indiveri. Real-time classification of complex patterns using spike-based learning in neuromorphic VLSI. *IEEE Transactions on Biomedical Circuits and Systems*, 3(1):32–42, Feb 2009. ISSN 1932-4545. doi: 10.1109/TBCAS.2008.2005781. URL `https://dx.doi.org/10.1109/TBCAS.2008.2005781`.

Abigail Morrison, Markus Diesmann, and Wulfram Gerstner. Phenomenological models of synaptic plasticity based on spike timing. *Biological cybernetics*, 98(6):459–478, April 2008.

T. Natschlaeger and W. Maass. Information dynamics and emergent computation in recurrent circuits of spiking neurons. In S. Thrun, L. Saul, and B. Schoelkopf, editors, *Proc. of NIPS 2003, Advances in Neural Information Processing Systems*, volume 16, pages 1255–1262, Cambridge, 2004. MIT Press.

Erkki Oja. Simplified neuron model as a principal component analyzer. *Journal of Mathematical Biology*, 15(3):267–273, Nov 1982. ISSN 1432-1416. doi: 10.1007/BF00275687. URL `https://doi.org/10.1007/BF00275687`.

D Plenz and A Aertsen. Neural dynamics in cortex-striatum co-cultures: 2. spatiotemporal characteristics of neuronal activity. *Neuroscience*, 70(4):893–924, 2 1996. doi: 10.1016/0306-4522(95)00405-X. URL `https://dx.doi.org/10.1016/0306-4522(95)00405-X`.

William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. Cambridge University Press, New York, NY, USA, 3 edition, 2007. ISBN 0521880688, 9780521880688.

Python Wiki. Performance tips, 2017. URL `https://wiki.python.org/moin/PythonSpeed/PerformanceTips`. Online; accessed 03-Nov-2017.

Ivan Raikov, Robert Cannon, Robert Clewley, Hugo Cornelis, Andrew Davison, Erik De Schutter, Mikael Djurfeldt, Padraig Gleeson, Anatoli Gorchetchnikov, Hans Ekkehard Plesser, Sean Hill, Mike Hines, Birgit Kriener, Yann Le Franc, Chung-Chuan

Lo, Abigail Morrison, Eilif Muller, Subhasis Ray, Lars Schwabe, and Botond Szatmary. NineML: The network interchange for neuroscience modeling language. In *BMC Neuroscience*, volume 12, pages 1–2, 07 2011.

J. Schemmel, J. Fieres, and K. Meier. Wafer-scale integration of analog neural networks. In *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, pages 431–438, June 2008. doi: 10.1109/IJCNN.2008.4633828. URL https://dx.doi.org/10.1109/IJCNN.2008.4633828.

Sebastian Schmitt, Johann Klähn, Guillaume Bellec, Andreas Grübl, Maurice Güttler, Andreas Hartel, Stephan Hartmann, Dan Husmann, Kai Husmann, Vitali Karasenko, Mitja Kleider, Christoph Koke, Christian Mauch, Eric Müller, Paul Müller, Johannes Partzsch, Mihai A. Petrovici, Stefan Schiefer, Stefan Scholze, Bernhard Vogginger, Robert Legenstein, Wolfgang Maass, Christian Mayr, Johannes Schemmel, and Karlheinz Meier. Neuromorphic hardware in the loop: Training a deep spiking network on the BrainScaleS wafer-scale system. *Proceedings of the 2017 IEEE International Joint Conference on Neural Networks*, 2017. doi: 10.1109/IJCNN.2017.7966125. URL https://dx.doi.org/10.1109/IJCNN.2017.7966125.

Alwyn C. Scott. *The Nonlinear Universe: Chaos, Emergence, Life*. Springer Publishing Company, Incorporated, 1st edition, 2007. ISBN 3540341528, 9783540341529.

Richard M. Stallman and the GCC Developer Community. *Using the GNU Compiler Collection*, 2003. for GCC version 4.9.4.

Richard M. Stallman and the GCC Developer Community. GNU compiler collection, 2017. URL https://github.com/electronicvisions/gcc. Electronicvisions Fork.

Yannik Stradmann. Characterization and calibration of a mixed-signal leaky integrate and fire neuron on HICANN-DLS. Bachelorarbeit, Universität Heidelberg, 2016.

David Stöckel, Benjamin Cramer, Andreas Hartel, Arthur Heimbrecht, Eric Müller, Christian Pehle, Yannik Stradmann, Johannes Schemmel, and Karlheinz Meier. Flexible synaptic plasticity on accelerated analog neuromorphic hardware. To be published., 2017.

David Sussillo, Taro Toyoizumi, and Wolfgang Maass. Self-tuning of neural circuits through short-term synaptic plasticity. *Journal of Neurophysiology*, 97(6):4079–4095, 2007. ISSN 0022-3077. doi: 10.1152/jn.01357.2006. URL https://dx.doi.org/10.1152/jn.01357.2006.

Gina G. Turrigiano. Homeostatic plasticity in neuronal networks: the more things change, the more they stay the same. *Trends in Neurosciences*, 22(5):221 – 227, 1999. ISSN 0166-2236. doi: 10.1016/S0166-2236(98)01341-1. URL https://dx.doi.org/10.1016/S0166-2236(98)01341-1.

G. E. Uhlenbeck and L. S. Ornstein. On the theory of the brownian motion. *Phys. Rev.*, 36:823–841, Sep 1930. doi: 10.1103/PhysRev.36.823. URL https://dx.doi.org/10.1103/PhysRev.36.823.

M. C. W. van Rossum. A novel spike distance. *Neural Computation*, 13(4):751–763, 2001. doi: 10.1162/089976601300014321. URL https://dx.doi.org/10.1162/089976601300014321.

R. J. Vogelstein, U. Mallik, J. T. Vogelstein, and G. Cauwenberghs. Dynamically reconfigurable silicon array of spiking neurons with conductance-based synapses. *IEEE Transactions on Neural Networks*, 18(1):253–265, Jan 2007. ISSN 1045-9227. doi: 10.1109/TNN.2006.883007. URL https://dx.doi.org/10.1109/TNN.2006.883007.

Timo Wunderlich. Synaptic calibration on the HICANN-DLS neuromorphic chip. Bachelorarbeit, Heidelberg University, 2016.

Danksagung

Vielen Dank an dieser Stelle an Prof. Karlheinz Meier und Dr. Johannes Schemmel für ihr Engagement und ihre Führung dieser prima Gruppe. Danke, dass ich an diesem selbst gewählten Projekt mit Ihrer Unterstützung arbeiten durfte. Dieser Dank gilt auch meinem Betreuer Andreas Hartel.

Prof. Kurt Roth verdanke ich die Wahl des Themas durch seine motivierende Vorlesung. Durch die Zeit die Sie sich für Vorlesung und persönliche Diskussionen genommen haben, haben Sie diese Arbeit mitgeprägt.

Den vielen Helfern bei Problemen und Fragen bin ich sehr dankbar, sonst gäbe es die Arbeit nur in vielen Nummern kleiner. Bei der Software und Firmware sind das Eric, Arthur, Christian Pehle, Christian Mauch und Johann. Bei technischen Fragen hatte ich Hilfe von Yannik, Andi, Gerd, Korbi und Sebastian. Die Diskussionen mit Dr. Viola Priesemann, Mihai, Benjamin und Oliver sind in weiten Teilen für den Inhalt verantwortlich. Unter den Genannten sind auch viele, die beim Korrigieren geholfen haben, wobei Ákos, Sebastian Schmitt und Daniel noch nicht genannt wurden. Danke euch allen.

Liebe Container-Kollegen, ihr seid super. Danke fürs Kaffee kochen und wichtigen Diskussionen zu Musik, Kaffeegeschmack, Build-Systemen und anderen diskussionswürdigen Fragen.

Zuletzt an meine Liebsten: Kathrin, danke, dass du dabei bist. Freunde und Familie, mit eurem Interesse und der Freizeit mit euch macht es definitiv mehr Spaß.

Erklärung

Ich versichere, dass ich diese Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.