Department of Physics and Astronomy
University of Heidelberg

Master's thesis in Physics
submitted by

Markus Kreft
born in Berlin (Germany)

March 2021

Input-Induced Dynamical States in Homeostatically Regulated Neuromorphic Recurrent Neural Networks

This Master's thesis has been carried out by Markus Kreft at the
Kirchhoff-Institute for Physics
Heidelberg University
under the supervision of
Dr. Johannes Schemmel

Abstract

Information processing in recurrent spiking neural networks is of interest to both neuroscience and novel approaches to computation. The synaptic connections between neurons in these networks determine their dynamics and function. Precise control of the connections poses an acute challenge. This thesis leverages a local homeostatic plasticity rule that regulates synaptic weights based on single neuron spike rates to achieve stable network activity. In this setting the coupling strength of the network to external input allows control over the specific dynamical regime as characterised by the autocorrelation time.
Networks were implemented on the BrainScaleS-2 neuromorphic hardware system that enables fast and efficient emulation. For reference, the networks were also simulated with numerical methods. The same kind of qualitative control of network dynamics was found for both implementations. In a reservoir computing setting the spiking network was extended with a linear readout for classification tasks. This setup was analysed with various spiking datasets. Specifically, the impact of different dynamical states in the reservoir on classification performance was observed, following the general guiding principle of critical computing.

Zusammenfassung

Die Verarbeitung von Informationen mit rekurrenten spikenden neuronalen Netzwerken ist sowohl für die Neurowissenschaft, als auch für neuartige Rechenansätze mit Computern von Interesse. Die Verbindungen zwischen den Neuronen in solchen Netzwerken bestimmen deren Dynamik und Funktionsweise. Präzise Kontrolle dieser Verbindungen ist eine akute Herausforderung. In dieser Arbeit wird eine lokale homöostatische Plastizitätsregel verwendet, die die synaptischen Gewichte auf Grundlage der Feuerraten einzelner Neuronen reguliert, um stabile Netzwerkaktivität zu erreichen. Unter diesen Umständen kann der dynamische Zustand, charakterisiert durch die Autokorrelationszeit, über die Stärke der Koppelung an externen Input gesteuert werden.
Netzwerke wurden auf dem neuromorphen Hardware System BrainScaleS-2 implementiert, was schnelle und effiziente Emulation ermöglicht. Als Referenz wurden die gleichen Netzwerke auch mit numerischen Rechnungen simuliert. Für beide Implementierungen wurde die gleiche qualitative Kontrolle über die Netzwerkdynamik beobachtet. Im Rahmen des Reservoir Computing wurde das spikende Netzwerk um einen Linearen Readout erweitert und für Klassifikationsaufgaben genutzt. Der Aufbau wurde mit verschiedenen spikenden Datensätzen untersucht. Besonders wurde der Einfluss des dynamischen Zustands im Reservoir auf die Klassifikation, nach dem Prinzip des Critical Computing, betrachtet.

Contents

1 Introduction

When John von Neumann described the fundamentals of an electronic computer in his *First Draft of a Report on the EDVAC* [von Neumann 1945] he drew inspiration from McCulloch and Pitts's description of neurons in the human brain [McCulloch et al. 1943]. From this point of view, every current computer with a von Neumann architecture implements computation in a form of formal neural network. Modern artificial neural networks (ANNs) resemble biological networks much closer. These architectures — simulated on classical computers or run on specialised hardware like inference accelerators — have gained popularity for computation in recent years. However, programming them, i.e., training them to perform a specific task, has proven to be more complicated and costly. Only the wide availability of computational power allowed to leverage the backpropagation algorithm [Rumelhart et al. 1986] for the recent machine learning revolution [Schmidhuber 2015; LeCun et al. 2015].

Still, compared to actual neural substrates even these ANNs are very simplified, effectively only calculating sums of input values. In contrast, the human brain processes information in continuous time by propagating action potentials between neurons. A significant difference to classical computers is the inherently asynchronous approach of these neural spike events, which may allow for robust but energy efficient computation [Maass 1997]. As might be expected from these considerations, training of such spiking neural networks (SNNs) has proven even harder and remains a field of active research [Zenke et al. 2021a]. More advanced network architectures closer to biological models are (1) more complicated to train for tasks and (2) require vast amounts of computational resources to solve the time-continuous dynamics.

One approach that addresses the first problem of training is given by the framework of reservoir computing [Tanaka et al. 2019]. In this setting the network is only used to project inputs into a high-dimensional state space and actual learning of tasks is implemented by a simple linear readout trained with traditional regression methods. This circumvents the problems of supervised training within the SNN. Nonetheless, the connections between the spiking neurons of the reservoir need to be initialised in some way to achieve stable dynamics that are favourable for computing. Typically, weights are drawn from predefined distributions with certain parameters. However, the specific choice of these can influence network dynamics and stability, which in turn impact the performance of the reservoir.

A more flexible procedure that helps to reach stability and select desired dynamics is the active homeostatic regulation of connections inside the SNN. The rate-based plasticity rule by Zierenberg et al. [2018] dynamically adapts synaptic weights based on the spike activity of postsynaptic neurons. In this setting the synaptic connections self-organise to a dynamical state that is determined by the strength of external input. Following Cramer et al. [2020] the coupling to inputs can be controlled by the number of presynaptic connections from external units into the network. This allows fine control over the network dynamics.

In the reservoir computing setting, this control can be leveraged to chose network dynamics favourable for tasks. Dynamics close to a phase transition between ordered and chaotic states have been found to optimise abstract information theoretic measures and are believed to enable high computational performance [Bertschinger et al. 2004]. This concept, generally known as critical computing, can be investigated by classification tasks of varying complexity, where more complex tasks require more computational power. Datasets of spike patterns that allow such control over the difficulty of tasks can be generated by sampling points from high-dimensional random manifolds [Zenke et al. 2021b]. For more realistic settings, real-world data is provided by the Spiking Heidelberg Digits dataset [Cramer et al. 2019]. It contains neural spike trains generated from spoken digit audio recordings using models inspired by the human auditory system. Such auditory tasks are especially interesting because their intrinsic time dimension allows to leverage the temporal aspects of SNNs.

The second challenge with biologically inspired networks is the exhaustive computational cost of solving the equations underlying the time-continuous mechanics. Simulating these requires vast amounts of numerical calculations. Neuromorphic hardware [Mead 1990] addresses this by instead implementing neurons and synapses directly in dedicated circuits that emulate the neural substrates. One such neuromorphic hardware platform is the BrainScaleS system [Schemmel et al. 2010]. Specifically, the second generation BrainScaleS-2 (BSS-2) [Schemmel et al. 2020] system features the mixed-signal High Input Count Analog Neural Network (HICANN-X) hardware chip. In addition to analog neurons and synapses, it comprises an onboard digital processor for the implementation of synaptic plasticity in networks. Due to the analog implementation of neurons and synapses in electronic equivalent circuits the hardware runs at a 1000× speedup of temporal dynamic compared to biological counterparts. This allows for fast iteration during experiments with SNNs compared to traditional methods of simulation and also has potentially interesting applications in real time computation. Crucially, the system's architecture allows for high scalability, making the acceleration factor independent of the network size. Combined with high energy efficiency of the analog implementation this can potentially enable the emulation of systems much larger than what is currently possible with simulations.

This thesis presents a recurrent spiking neural networks with homeostatic regulation

implemented on the BrainScaleS-2 system. The strength of coupling to the external input is demonstrated to allow control over the dynamical state in the network. An aspect of interest is the comparison of the behaviour of the neuromorphic hardware to reference simulations of the underlying equations. The desired regulation of network dynamics can in principle be achieved on hardware in the same way as in simulation. Various benefits and constraints of the hardware are analysed. Additionally, first results from using the network in a reservoir computing setup are demonstrated and the impact of network dynamics on computation is analysed with different pattern detection tasks.

The thesis is structured as follows: Chapter 2 describes the network models of interest and introduces the BrainScaleS-2 system. This is followed in Chapter 3 by a description of the specific network architecture and experiment procedures, including details of the implementation on the neuromorphic hardware system. Chapter 4 demonstrates the homeostatic regulation of network activity and the control of the dynamical state in experiments on BSS-2. These are compared to simulations of similar networks in Chapter 5. Various differences between the two implementations and their impact on the network dynamics are analysed. The neuromorphic implementation of the SNN is used in a reservoir computing setup in Chapter 6, where first classification results for different stimuli are presented. Finally, Chapter 7 draws conclusion from the findings of this thesis and gives an outlook on possible future work.

> Some experiments presented in this thesis were conducted in collaboration with Benjamin Cramer and Johannes Zierenberg and will be contributed to a paper in preparation [Cramer et al. in prep].

2 Background

This chapter begins with a very short introduction to spiking neural networks and their biological terminology in Section 2.1. Subsequently, specific models for neuron dynamics are presented in Section 2.2. Section 2.3 briefly gives an overview of the general framework of reservoir computing followed by a description of critical dynamics in corresponding networks in Section 2.4. The BrainScaleS-2 hardware system that acts as a basis for neural emulation is described in Section 2.5. Section 3.4 describes datasets that are used for classification tasks and Section 3.5 presents methods for the evaluation of the network response.

2.1 Biological Principles

The human brain performs computation in a network of connected nerve cells that share information by propagating electrical pulses. (A comprehensive overview is given e.g., by Gerstner et al. [2014].) While numerous types of neurons can be distinguished, a single cell is typically divided into three main parts, the dendrites, the soma, and the axon, as schematically drawn in Figure 2.1. Dendrites are the inputs to the neuron and transmit signals originating from other neurons. The central unit of the neuron is called the soma which processes these signals in a non-linear way. The generated output is then passed through the axon to neighboring neurons. Axons can stretch over macroscopic distances leading to non-local connections in the brain.

Processing in the soma makes use of ion channels in the cell membrane that allow to build up a voltage potential depending on the charge of the ions passing through. Without input from other neurons, the potential inside the neuron is in an equilibrium state of constant polarization compared to the surroundings. If input arrives in form of an electrical charge through a dendrite, the voltage across the membrane changes from its resting state. Once the membrane potential reaches a threshold value, it exhibits a steep depolarization followed by a phase of hyperpolarization. If such an action potential is emitted, the neuron is said to fire a spike that it passes along the axon to other neurons. Sequences of single neuron spikes are referred to as spike trains.

Connections between the axon of one neuron and the dendrites of another are called synapses. Most common are chemical synapses. There, the axon of the presynaptic neuron comes close to the dendrite of the postsynaptic neuron leaving a small gap, the synaptic

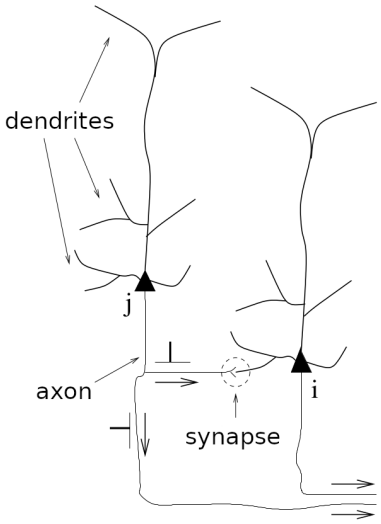


Figure 2.1: Schematic illustration of two neurons connected by a synapse. The dendrites and axon of the presynaptic neuron j are indicated with arrows, the triangle highlights the soma. The synaptic cleft is encircled. Figure adapted from Gerstner et al. [2014].

cleft (Fig. 2.1). A presynaptic action potential that arrives at the cleft leads to the release of neurotransmitters. These diffuse to the postsynaptic neuron, where they can facilitate the passage of certain ions through the channels in the membrane leading to the buildup of a postsynaptic potential (PSP). In effect, the electrical pulse is transmitted from one neuron to another.

The strength of synaptic connections is determined by the amount of neurotransmitters released in the synapse and the type and number of ion channels activated in the postsynaptic neuron. This is specified by a synaptic weight. If a presynaptic spike increases the postsynaptic membrane potential, a synapse is termed excitatory, if it leads to a decrease of the potential it is called inhibitory.

In general, the weights between neurons are not static, but can change over time. This is referred to as synaptic plasticity and plays an important role in learning of specific information. In the human brain there are many different mechanisms that regulate neuron connectivity. For this thesis, homeostatic regulation of neuron rates which maintains steady network activity is of interest.

2.2 Modelling

In this section the specific neuron and synapse models used for experiments are described. They abstract the observations from biology by describing the dynamics of voltages and currents in neurons with differential equations.

2.2.1 Neurons

The leaky integrate-and-fire (LIF) neuron is a widely used model for single neuron dynamics based on the assumption that only the timing of neural spikes is important for information processing. Thus, the dynamics of the membrane are defined only until a certain threshold potential is reached. At this point in time, a spike event is registered and the membrane is reset [Gerstner et al. 2014]. The membrane potential $u_j(t)$ of a neuron j is governed by

$$\tau_{\text{mem}} \frac{\mathrm{d}u_j(t)}{\mathrm{d}t} = -\left[u_j(t) - u_{\text{leak}}\right] + \frac{I_j(t)}{g_{\text{leak}}} \,. \tag{2.1}$$

with the membrane time constant τ_{mem}, the leak conductance $g_{\text{leak}} = C_{\text{m}}/\tau_{\text{mem}}$ and the leak potential u_{leak}, as well as the total input current $I_j(t)$. The neuron fires a spike when the membrane potential reaches the threshold u_{thres}

$$u_j(t_j^k) \geq u_{\text{thres}} \,, \tag{2.2}$$

defining the k-th spike time t_j^k of the neuron j. Subsequently, the membrane potential is clamped to the reset potential u_{reset} for the duration of one refractory period τ_{ref}.

2.2.2 Synapses

The leaky synapse model describes the flow of current between pre- and postsynaptic neurons. Currents $I_{ij}(t)$ from input units i to neurons j are modeled by

$$\tau_{\text{syn}}^{\text{exc}} \frac{\mathrm{d}I_{ij}(t)}{\mathrm{d}t} = -I_{ij}(t) + I_{ij}^{\text{in}}(t) \,, \tag{2.3}$$

$$\tau_{\text{syn}}^{\text{inh}} \frac{\mathrm{d}I_{ij}(t)}{\mathrm{d}t} = -I_{ij}(t) - I_{ij}^{\text{in}}(t) \,, \tag{2.4}$$

for excitatory and inhibitory synapses respectively, with the corresponding synaptic time constants $\tau_{\text{syn}}^{\text{exc}}$ and $\tau_{\text{syn}}^{\text{inh}}$. The input current I_{ij}^{in} is determined by the times t_i^k of the k-th spikes of input i

$$I_{ij}^{\text{in}}(t) = \sum_k I_0 \cdot w_{ij} \cdot \delta\left(t - t_i^k - d_{\text{syn}}\right) \,, \tag{2.5}$$

with synaptic weights w_{ij}, amplitude I_0 and the synaptic delay d_{syn}. The sum

$$I_j(t) = \sum_i I_{ij}(t) \tag{2.6}$$

of presynaptic input currents enters into Eq. (2.1) for the neuron membrane potentials.

2.3 Reservoir Computing Networks

Reservoir computing [Tanaka et al. 2019] is a general framework for computation with networks of connected units capable of information transfer and processing, usually using a sparsely connected recurrent neural network (RNN). Reservoir networks are also commonly known as echo state networks [Jaeger 2001] or liquid state machines [Maass et al. 2002] which describe similar approaches that were independently developed.

The setup consist of the reservoir network itself and a readout layer. A characteristic aspect of the architecture is its approach to training. Typically, synaptic weights in the reservoir are initialised by drawing random values from a certain distribution and remain fixed, while output connections are trained for a specific task. The reservoir only acts as a non-linear filter that receives input to some of its units and transforms it into a high-dimensional state space. A simple extraction mechanism like a linear classifier implements the readout and takes care of actual learning of presented data.

Specifically, the framework can be used for computation with spiking neurons. This foregoes the problem of calculating error signals from spikes times which is generally required for supervised methods of training.

2.4 Critical Network Dynamics

The concept of criticality is considered a guiding principle for selecting optimal dynamics in neural networks. Systems operating close to a critical point, a (second-order) phase transition between order and chaos, maximise various processing measures like correlation length, information transfer, and susceptibility [Barnett et al. 2013; Tkačik et al. 2015; Wilting et al. 2018a]. Therefore, criticality is believed to positively impact computational performance. Systems at the 'edge of chaos,' nearing a critical transition, are proposed to benefit complex tasks [Langton 1990; Bertschinger et al. 2004; Boedecker et al. 2012; Muñoz 2018]. However, tuning networks precisely towards such a critical point is challenging. Synaptic plasticity algorithms can enable a network to adapt its connections by itself. Ideally, the weights should self-organise to a state of desired dynamics based on local information alone.

Theoretical considerations for the design of plasticity used in this thesis follow Zierenberg et al. [2018]. The dynamical regime of a network subjected to a rate-based homeostatic plasticity rule can be controlled by the frequency of the input stimulus. This is motivated in the framework of a driven branching process [T. E. Harris 1963] with branching parameter m and external input rate h. Spike generation is modeled at discrete intervals. On average, a spike at timestep t causes m postsynaptic spikes in the next step, such that the expected

total network activity A_t becomes

$$\mathbb{E}(A_{t+1}|A_t) = mA_t + Nh\Delta t\,, \tag{2.7}$$

in a network with N units and a duration of Δt between timesteps. For $m < 1$ this process behaves sub-critically, cascades of activity fade out over time and the temporal average of the network activity converges to a stationary value

$$\langle A \rangle = \frac{1}{T}\sum_{t=1}^{T} A_t \xrightarrow{T\to\infty} \frac{Nh\Delta t}{1-m} =: A_\infty\,. \tag{2.8}$$

Assuming a homogeneous rate distribution between all neurons in the network implies a mean neuron spike rate

$$\nu = \frac{h}{1-m}\,, \tag{2.9}$$

that becomes constant for a branching parameter $m \in [0, 1)$ at finite input rate $h \in [0, \infty)$.

Plasticity in the network is designed to maintain a target firing rate ν_{target} by adapting all presynaptic connections w_{ij} of neuron j base on its local firing activity a_j, t. Weight updates at timescale T are implemented as

$$\Delta w_{ij,t} = (\nu_{\text{target}}\Delta t - a_{j,t})\left(\frac{\Delta t}{T}\right)\,. \tag{2.10}$$

For sufficiently slow plasticity ($\Delta t/T \to 0$) the updates become small $\Delta w_{ij,t} \approx 0$ and dynamics are solely determined by the average weights. These allow to estimate an effective branching parameter. With Eq. (2.9) the mean-field solution is approximated by

$$m = 1 - h/\nu_{\text{target}}\,. \tag{2.11}$$

Thus, the input strength can be used as a control parameter to determine network dynamics. For low input rates, the homeostasis strengthens recurrent connections to maintain the targeted output rate, leading to higher correlations and networks with a larger branching parameter closer to criticality.

In experimental settings, dynamics can be quantified by the autocorrelation (AC) of network activity. For a subcritical branching process, the autocorrelation function is given by [Wilting et al. 2018b]

$$C(t) = m^t\,. \tag{2.12}$$

Comparison with an exponential decay

$$C(t) = e^{-t\Delta t/\tau}\,, \tag{2.13}$$

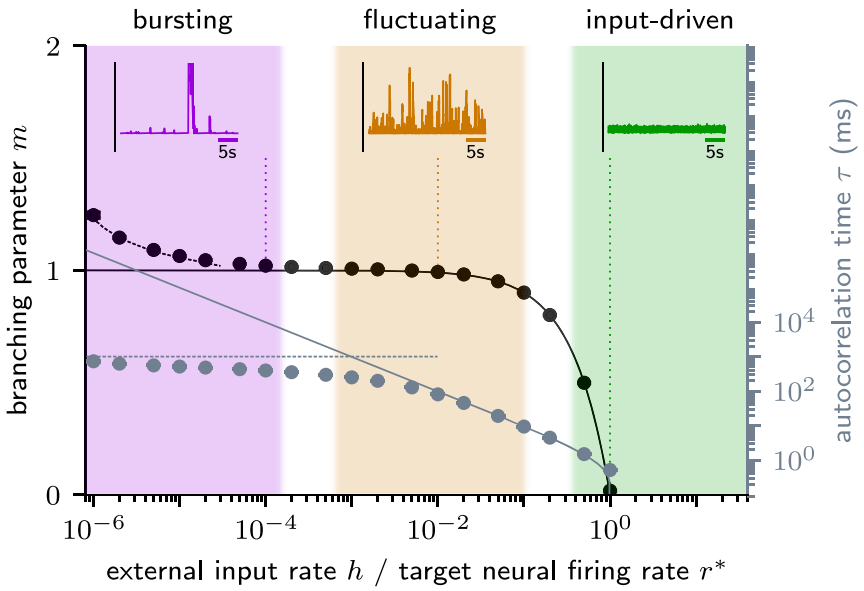


Figure 2.2: Sketch of different network dynamics characterised by their autocorrelation. The network states range from bursting over fluctuating to input-driven, depending on the external input rate. The dynamical regimes are classified by the branching parameter and the autocorrelation time (solid lines). For low inputs the homeostatic timescale leads to resonances that cause bursting behaviour. Figure taken from Zierenberg et al. [2018].

yields the autocorrelation time

$$\tau = -\Delta t / \ln(m)\,, \tag{2.14}$$

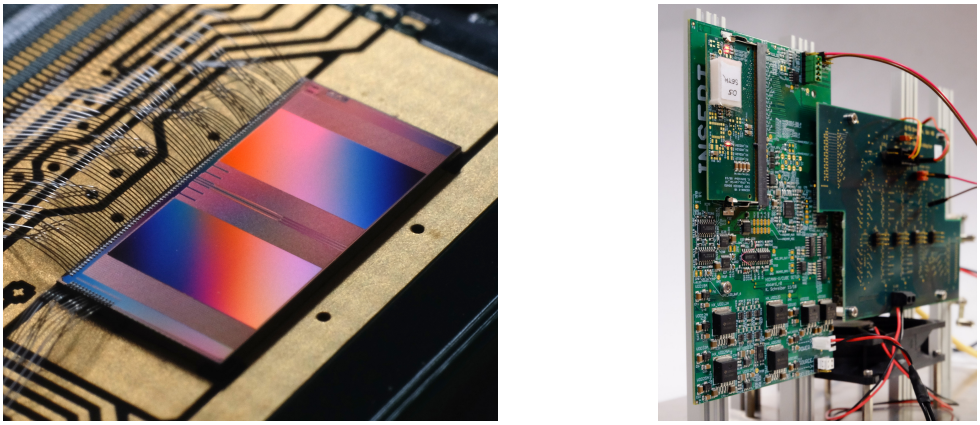
which diverges at $m = 1$ where the process has a critical phase transition.

Figure 2.2 shows three distinct dynamical regimes determined by the rate of the external input. With decreasing stimulation rate, the networks compensates missing input by generating more internal activation, tuning the system closer to a critical state. The activity passes from the input-driven regime to fluctuating dynamics. For even less inputs, dynamics are limited by the homeostatic timescale (Eq. 2.10) which leads to resonance effects that create bursts with an effective branching parameter resembling supercritical behaviour.

These mean-field consideration were also shown to function for networks of spiking neurons. Specifically, control of criticality was previously demonstrated with a different form of plasticity on neuromorphic hardware with LIF neurons [Cramer et al. 2020].

2.5 Neuromorphic Hardware

The BrainScaleS-2 (BSS-2) platform [Schemmel et al. 2020; Friedmann et al. 2017] is an accelerated mixed-signal analog neuromorphic hardware system. At its core is the HICANN-X v2 ASIC, a fully custom neuromorphic silicon chip manufactured in a CMOS



(a) HICANN-X ASIC (b) *Cube Setup*

Figure 2.3: Photographs of the BrainScaleS-2 system. (a) The die shot portrays the of HICANN-X v2 ASIC. The two reflective areas contain the synapse arrays of the upper and lower half, with the neuron blocks and capacitive memory cells in between. (b) Single-chip *cube setups* feature a HICANN-X chip under the white plastic cap (top left) bonded onto the chip-carrier PCB. An FPGA (not visible on the back) handles communication to the host computer. Photographs taken from Müller et al. [2020].

65 mm process (pictured in Fig. 2.3a). It implements neurons and synapses in analog equivalent circuits that emulate their dynamics, while spikes are digitised and routed as events. The system has characteristic timescales that are about 1000× faster than biological equivalents. This speedup allows for extremely fast and energy efficient emulation of neural dynamics.

The analog network core of HICANN-X features 512 neuron cells that are arranged in four quadrants, a left and right part of the top and bottom halves of the chip (Fig. 2.4). Single neuron circuits emulate the adaptive exponential integrate-and-fire (AdEx) model [Brette et al. 2005] by implementing the neuron membrane with a capacitor. The 131 072 synapses are placed in four grids of 256 × 128 each, where every column contains 256 synapses that pass postsynaptic currents to the neuron in that column, giving each neuron 256 possible presynaptic partners. Every row in the grid receives inputs from a synapse driver that is located on the side of the grid and provides presynaptic inputs. One synapse driver sends spike to two rows, the synapses in each of which can be configured to be either excitatory or inhibitory. The drivers receive external input spikes and can simultaneously pass the output spike events from the neurons back into the network. This way RNNs can be configured (cf. Section 3.2). Analog voltages and currents are stored in capacitive memory (CapMem) cells [Hock et al. 2013].

To reduce input/output load, HICANN-X has integrated on-chip spike generators that can produce Poisson spike trains with configurable frequency. Using these as background noise sources allows much faster experiments than when providing all spikes as external

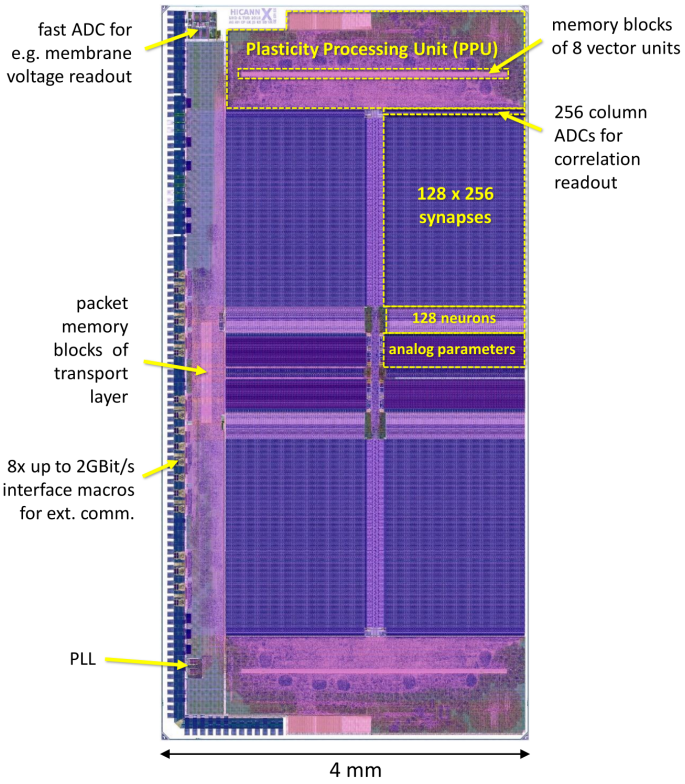


Figure 2.4: Layout of the BrainScaleS-2 full-size ASIC. The HICANN-X analog core is symmetric between the top and bottom half. Each half contains a PPU co-processor and a left and a right quadrant. A quadrant contains a row of 128 neurons and an array of 256×128 synapses. Additional components for digital control and event handling on the left side are highlighted. Figure taken from Grübl et al. [2020].

events, especially at high frequencies when many spike events would need to be transferred. The bandwidth for off-chip spike transfer can be reserved for structured input.

For observation of analog parameter values, HICANN-X has multiple analog-to-digital converters (ADCs). The fast membrane ADC (MADC) can record voltage traces of single neurons with high temporal precision. The columnar ADC (CADC) has a reduced sampling rate but provides simultaneous access to traces of 128 components in parallel.

In addition to the analog components, BSS-2 features two general purpose co-processors located in the top and bottom half of the chip (Fig. 2.4). Each of these plasticity processing units (PPUs) has access to the parameters of the neurons and synapses, allowing for closed-loop experiments implemented purely on-chip. They can also control and read the neuron rate counters that measure spike rates of each neuron over a specified interval. The PPUs implement the PowerISA instruction set architecture and feature a single instruction, multiple data (SIMD) vector extension. It can handle 128×8 bit vectors to perform calculations for multiple synapses in parallel [Friedmann et al. 2017; Grübl et al. 2020]. It also allows to configure synaptic weights in parallel. The PPUs are programmed in C++

using inline assembler instructions for efficient use of the SIMD extension [Friedmann and Pehle 2018].

In the *cube setups* (Fig. 2.3b) used for this thesis, a field-programmable gate array (FPGA) handles communication over 1-Gigabit Ethernet with a host computer. Experiment procedures are controlled by software[1] written in Python, using the hardware abstraction layers provided by the BSS-2 Operating System [Müller et al. 2020].

[1] The software for experiments is available in the `model-hw-wavy` repository hosted on the group-internal site at https://gerrit.bioai.eu/plugins/gitiles/model-hw-wavy

3 Methods

In this chapter the methods and implementation details for experiments presented in this thesis are described. Section 3.1 begins with an overview of the network architecture and experimental procedures. Details of the implementation of the recurrent network on the BrainScaleS-2 system are given in Section 3.2. Simulations of the same networks that are compared to the hardware results for validation purposes, are outlined in Section 3.3. Finally, Section 3.4 describes the datasets that are used for classification tasks and Section 3.5 provides methods for evaluation of the network response.

3.1 Experiment and Network Architecture

Experiments in this thesis can be separated into different stages. The main component is the spiking neural network implemented on the BrainScaleS-2 system. It is realised as a sparsely connected recurrent neural network with N^{exc} excitatory and N^{inh} inhibitory leaky integrate-and-fire neurons with leaky synapses. An input population of size N_{in} projects spikes into the network and contains excitatory and inhibitory units in same ratio as the network itself. However, the average number of connections from the input population into the network, the indegree K_{in} and within the network itself, the recurrent sparsity K_{rec}, are configured differently.

Tying in with the work of Zierenberg et al. [2018], synaptic weights within and between the network populations are treated equally and are dynamically adapted in the *adaptation phase*. During this time, the synapses are exposed to the homeostatic plasticity rule, while the network receives input as homogeneous Poisson noise. However, following Cramer et al. [2020] the indegree of the network is used to select the dynamical state, instead of changing the input population rate. Since postsynaptic inputs to the neurons in the network are added up, the effect for single neurons is comparable to changing the rate directly (cf. neuron layout in Section 2.5). This approach has the benefit that desired network dynamics can be controlled independently of the input stimulus. For classification tasks, adapting the stimulus rate would have the undesired effect of changing the task at hand. In contrast to Zierenberg et al., only a fraction of weights is actually changed at each plasticity update. This has a significant effect on the stability of the homeostasis.

Network dynamics are determined from static experiments with fixed weights after adaptation. The network response to Poisson noise is recorded and evaluated with statistical

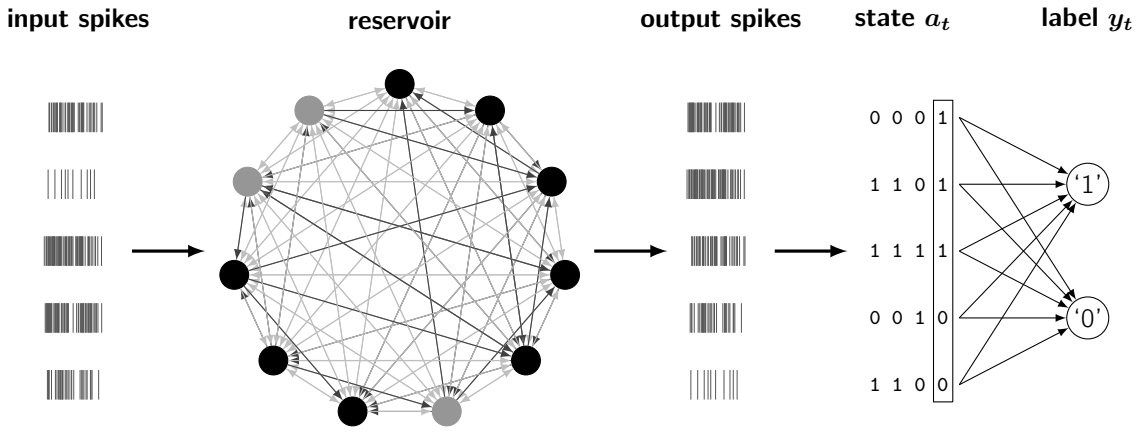


Figure 3.1: Overview of the full reservoir computing setup for classification tasks. Input spikes are created in a preprocessing stage. They are fed into the recurrent network with an average indegree of K_{in} channels per neurons. The reservoir features sparsely connected excitatory and inhibitory neurons that generate output spikes. For classification tasks, the spike trains are binned to obtain the reservoir state a_t at timestep t, which is used to train a linear classifier to predict the labels y_t. In isolation, the network dynamics are analysed with Poisson inputs.

measures (c.f Section 3.5). A specific interest of this thesis is the analysis of network dynamics for different amounts K_{in} of coupling to the input.

In the reservoir computing framework, the SNN with fixed weights is stimulated with the input spikes from specific datasets (cf. Section 3.4) and provides output to a linear classifier in the readout stage (cf. Section 3.5). A schematic overview of the full reservoir computing setup is shown in Figure 3.1.

3.2 Network Implementation

All 512 neurons circuits available on HICANN-X are used to implement one large recurrent network. The adaptive exponential integrate-and-fire (AdEx) neurons are configured to approximate LIF behaviour by disabling the exponential term. The synapse grids of the four quadrants are abstracted as a single array of size 256×512. Each synapse (i, j) receives input from the driver in its row i and generates a current that is passed to the neuron in its column j. Figure 3.2 shows a schematic view of the connectivity.

Recurrence and quadrant interconnection is achieved by using the address coding capability of the digital spike routing system [Schemmel et al. 2020]. Every spike feed to the synapse drivers is marked with a specific 6 bit integer source address label. A synapse is configured with a corresponding decoder address and only transmits spikes with matching addresses. The four most significant bits of the label are used to implement the synapse grid abstractions. The two remaining least significant bits distinguish four different kinds

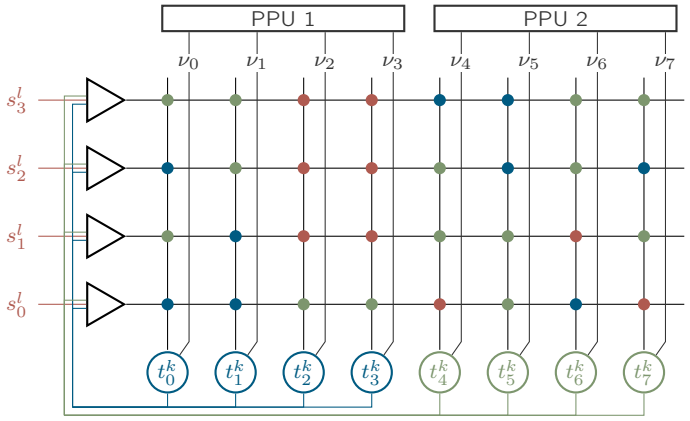


Figure 3.2: Schematic view of the abstracted synapse array on HICANN-X. Neurons (circles) from the top (blue) and bottom (green) half of the chip emit spike events t_i^k. Synapse drivers (triangles) forward both these recurrent spikes and additional external inputs s_i^l (red) to the synapses in the array (colored dots). The synapses in each row are either all excitatory or all inhibitory. Synapses are sensitive to events of one of the sources and transmit postsynaptic current to the neuron in their column. The PPUs have access to the neural firing rates ν_i of the neurons in their respective half as well as the weights of the synapses. Figure adapted from Cramer et al. [in prep].

of inputs to the synapses. The 256 output spike channels from neurons in the upper and lower half of the HICANN-X (cf. Figure 2.4), as well as 256 unique external spike channels are labeled with three separate addresses. No spikes are send labeled with the fourth one. The drivers in both the upper and the lower half receive these spikes such that connections across the two halves of the chip are made. This routing results in the recurrent neural network with $N = 512$ units and $N_{\text{in}} = 256$ inputs.
At the beginning of an experiment, the decoder address of each synapse is configured to one static value. An average number of $K_{\text{rec}}/2$ randomly chosen synapses per neuron is configured with the address of spikes from the lower and upper half each. This creates a sparse network where each neuron has on average K_{rec} recurrent presynaptic partners. A further average number of K_{in} random synapses of each neuron transmit the external spike events. This number selects the desired input strength. The rest of synapses is configured to the remaining fourth address with which no spikes are labeled, enforcing sparsity because the synapses effectively do not contribute to the network.

Following Dale's law [Dale 1934], neurons generate output that is either purely excitatory or inhibitory by configuring the synapse drivers for each row accordingly. A fixed number of $N_{\text{in}}^{\text{inh}} = 51$ randomly selected drivers in each half of the chip is chosen to be inhibitory corresponding to about 20 % inhibition in line with observations from biology.

Plasticity

The homeostatic plasticity rule motivated in Section 2.4 is implemented on the PPUs. Both run separately and are responsible for updating the synapses in their respective half

of the chip during the adaptation phase. They are triggered simultaneously by an update signal from the FPGA.

Weights are stored as 6 bit unsigned integer values that saturate at their minimal and maximal value respectively. At the beginning of an experiment, weights are initialised at zero. In intervals of $\Delta T = T_{\text{eq}} + T_{\text{meas}}$ they receive negative feedback that pushes the individual postsynaptic firing rate ν_j of neuron j towards the target ν_{target}

$$\Delta w_{ij} = \eta \cdot (\nu_{\text{target}} - \nu_j) \ , \tag{3.1}$$

with the learning rate η that acts as a scaling between firing rates and weights. Rates ν_j are measured only during a period of T_{meas}, after the network dynamics have equilibrated for a duration of T_{eq}.

Crucially, only an average fraction of p_{update} randomly chosen weights is updated. The procedure is implemented on the PPUs by drawing a random value $x_{ij} \sim \mathcal{U}(0, 1)$ for every synapse. A weight w_{ij} is only updated if $x_{ij} < p_{\text{update}}$, else the previous value is kept. Random values are obtained with the accelerated on-chip Xorshift random number generators (RNGs). The dedicated generators enable fast plasticity updates.

The driving force of the branching model is provided by the on-chip spike background sources of HICANN-X. They generate homogeneous Poisson spikes from independent processes for the 256 input channels. The average rate is set to $\nu = 10\,\text{Hz}$.

This autonomous setup with the fully on-chip implementation of homeostatic network adaptation avoids any input/output communication bottlenecks and allows for extremely fast and efficient experiments.

Parameter Noise

Due to the analog nature of circuits on BSS-2, manufacturing tolerances that allow for transistor mismatch lead to parameter fluctuations. The resulting fixed-pattern noise can be compensated to a certain degree by calibrating analog parameters to their desired target values [Brüderle et al. 2011; Neftci et al. 2011]. On HICANN-X this can be achieved by the configurability of parameters stored in the capacitive memory cells. For this thesis, the `calix` framework [Weis 2020] is used to calibrate neurons and synapses with a binary search algorithm. It makes use of the on-chip CADC for fast measurements of voltages and the MADC for precisely determining time constants.

While `calix` operates on hardware parameters, throughout this thesis all values are given in the biological equivalent domain by accounting for the 1000× speedup. Target values for time constants are chosen to be comparable to biological systems and voltages are selected to utilise the maximal dynamic range available on the hardware.

Table 8.1 in the Appendix lists all network parameters and their configured values. This includes mean values for measurements of neuron and synapse parameters presented in Chapter 4.

3.3 Network Simulations

For verification purposes the SNN is also implemented in software. In the controlled simulation environment the network configuration can be influenced in more detail and all quantities are directly available for observation. Additionally, the effect of parameter and temporal noise can be investigated by adding various levels of randomness to the parameters and the ideal solutions of equations. Comparing the network dynamics between hardware and simulation allows drawing conclusions about which aspects of the setup are important for achieving desired results. Also, unexpected behaviour of the hardware system can be detected by comparison of single neuron behaviour.

The setup is implemented in the `brian2` simulator [Stimberg et al. 2019]. It allows the simple definition of neuron populations and their synaptic connections, while still permitting detailed control of all configurations when required. For integrations, the Euler method with a timestep of $\mathrm{d}t = 10\,\mu\mathrm{s}$ is chosen. Neuron and synapse parameters are set to the same values that are measured on hardware (cf. Section 4.1). Fixed-pattern noise is realised by drawing parameters for individual units from Gaussian distributions with the same mean and standard deviation as estimated from the measurements. Additional temporal white noise with a standard deviation of $\sigma_u = 2\,\mathrm{mV} \cdot \sqrt{2\mathrm{d}t/\tau_\mathrm{mem}}$ is added to the membrane potentials to roughly corresponds to observations of the hardware behaviour.

There are slight differences in the exact implementation of the network architecture compared to the hardware. Instead of using a single sparse connection matrix to define all synaptic weights, separate populations for excitatory and inhibitory neurons and input populations are defined and each has separate synapse groups. This specifically means that, while the synapse drivers on BSS-2 are shared between external and recurrent synapses, separate input realizations are used in `brian2`. The average numbers of connections are however configured identically. Therefore, this at most impacts the realization of fixed-patten noise.

As a further validity check, the Euler integrations of neuron equations were also calculated directly using `numpy` [C. R. Harris et al. 2020]. The use of two separate simulations helped to identify and work out mismatches between implementations during development of experiments. The `numpy` simulation also allows for even more flexibility and enables implementing a network topology with a single weight matrix closer to that realised on hardware.

3.4 Datasets

In the extended reservoir computing setup for classification tasks, external spike patterns from spatio-temporal datasets are fed into the network. To analyse the effect of network dynamics on computational performance, the datasets that are described in the following have different levels of complexity.

3.4.1 Poisson Patterns

A simple spiking classification dataset is constructed by generating two classes of random Poisson spike patterns. Each class contains 1000 samples from identical Poisson patterns that differ by an amount of additional noise. The task is to discern between the two classes. The readout is trained on 75 % of the samples with the remaining ones used for testing.

The noise for the patterns is realised differently for two distinct stimulation paradigms. Either, the samples of the dataset are sent in addition to the background sources that are kept running after network adaptation. In that case all samples of a class are exactly the same and noise is determined by the background sources. This approach also means that the total input rate to the reservoir is larger than during adaptation. This case draws inspiration from biological observations, where a constant noise flow of neural activity is observed. Alternatively, the on-chip sources are disconnected from the reservoir after weight adaptation, and it only receives external input. Then, the rate of the input is selected to be the same as during adaptation and the impact of different amounts of noise added to the external spike patterns is analysed.

3.4.2 Random Manifolds

Synthetic datasets of varying complexity can be created using the `randman`[1] method by Zenke et al. [2021b]. Spikes patterns of a dataset are generated by drawing samples from smooth random manifolds. The D-dimensional manifold is defined by a smooth function $f : [0, 1)^D \to [0, \tau_{\text{rand}})^M$ that maps real-valued vectors to the M-dimensional embedding space. For each embedding dimension $i \in \{1, ..., M\}$, it is determined by the expansion in the Fourier basis

$$f_i(\vec{x}) = \prod_{j \in D} \left[\sum_{k=1}^{n_{\text{cutoff}}} \frac{1}{k^\alpha} \theta_{ijk}^A \sin\left(2\pi \left(k x_j \theta_{ijk}^B + \theta_{ijk}^C\right)\right) \right], \tag{3.2}$$

with parameters $\theta_{ijk}^L \sim \mathcal{U}(0, 1)$, $L \in A, B, C$ drawn independent and identically from a uniform distribution. The highest Fourier mode is set to $n_{\text{cutoff}} = 1000$.

[1] https://github.com/fzenke/randman

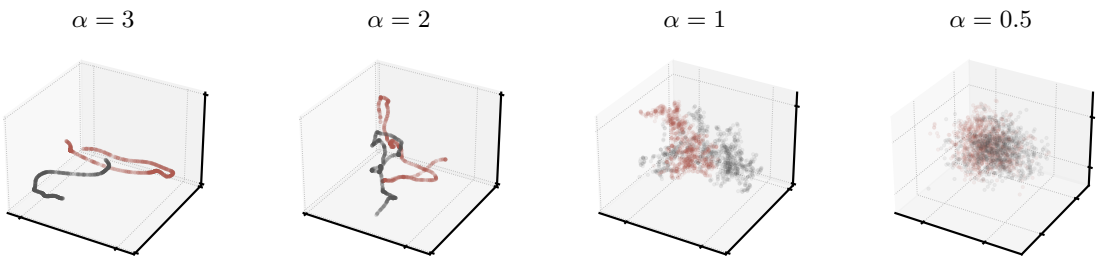


Figure 3.3: Examples from the random manifold dataset. One-dimensional example manifolds in three-dimensional embedding space are drawn for four values of the smoothness parameter α. For every smoothness, data from two different manifold is shown to demonstrate the difference between classes of the resulting datasets. From each manifold, 1000 random data points are plotted. Figures are created using the implementation provided by Zenke et al. [2021b].

To build a dataset of spike patterns, one such manifold is generated for each class c. Samples are then obtained by repeatedly sampling random points on the manifold from the hypercube $[0, 1)^D$ and mapping these into the embedding space. This results in a vector with M different entries in the range from 0 to τ_{rand}. These entries are interpreted as the single unit spike firing times of M input channels. Thus, each input channel contains exactly one spike and the embedding space dimension determines the number of unique inputs. The timescale τ_{rand} selects the effective rate of the stimulus.

The parameters D, α and c control the difficulty of the classification task. Specifically, α determines the smoothness of the spatial manifold variations, with small values increasing the high-frequency content. This effect is visualised in Figure 3.3 for sample manifolds in three-dimensional embedding space. Each plot shows samples from two distinct manifolds. Discerning between a larger number of classes c makes the task more difficult.

3.4.3 Spiking Heidelberg Digits

The Spiking Heidelberg Digits (SHD) dataset [Cramer et al. 2019] contains spike patterns generated from recordings of spoken digits. Auditory data is especially interesting because of its natural temporal dimension. With such data, SNNs can potentially leverage their time continuous nature compared to traditional ANNs. The dataset contains spikes generated from multiple utterances of the digits ‘0’ to ‘9’ in English and German language by 12 different speakers. For the creation of spike patterns in SHD, raw audio signals are processed with a chain of models inspired by the human cochlea that is depicted in Figure 3.4. The basilar membrane of the inner ear is modelled in a hydrodynamic simulation with a linear fluid that causes spatial frequency dispersion. Here, the tangential membrane velocity carries information about certain frequency segments in the audio signal. At $N_{\text{ch}} = 700$ positions with equal spacing on the membrane, transmitter based hair cells generate spikes depending on the membrane velocity. Outputs from $N_{\text{HC}} = 40$ hair cells

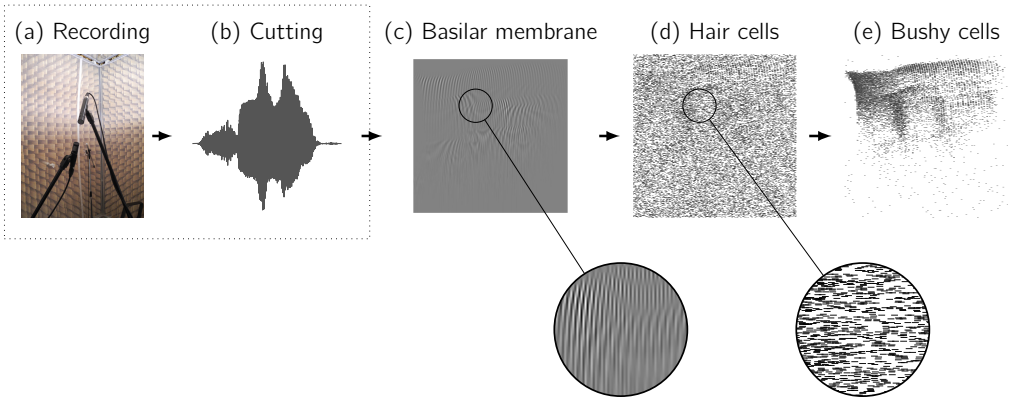


Figure 3.4: Spike generation for the SHD dataset. Audio signals (b) of digit recordings (a) are feed to a hydrodynamic basilar membrane simulation (c). Hair cells (d) that sit at certain frequency selective elements generate phase-coded spike trains. Bushy cells (e) increase phase-locking by integrating spikes from multiple hair cells at the same membrane position. Figure adapted from Cramer et al. [2019].

at each membrane position are integrated by bushy cells to increase phase-locking.

Samples of different speakers show considerable spread in their mean rate and duration. To limit the impact of this, and generally reduce the amount of data, only the samples of the first two speakers are used. These are more similar in duration. Input spike trains for the reservoir are generated by selecting 256 of the 700 channels equally spaced in the interval from 250 to 600. The higher and lower channels relate to corresponding extremes in the frequency of the audio signal and for most samples contain only few spikes (cf. Figure 3.4e). These spike trains are thresholded by neglecting spikes at times before the activity first rises above and after it last falls below a rate of 15 Hz. This filtering avoids periods of near silence. To generate stimuli of specific input rate, spike trains are further homogeneously downsampled in time by repeatedly selecting random spikes without replacement until the desired rate is reached. Ten such samples are drawn from each unique recording of the two speakers from both the training and testing sets of SHD. All resulting samples are randomly separated into custom testing and training sets with a 75 % – 25 % split. The classification task is constructed by always comparing the samples of two specific digits in the set.

3.5 Evaluation

The spiking networks produce output as neural spikes sequences. These contain neuron labels and timestamps for all events emitted during an experiment. For further processing, discrete activity traces are calculated from the spike trains. Time is binned at a step size of Δt and histograms are calculated to obtain either the neuron wise activity $a_{j,t}$, $j \in \{1, ..., N\}$ or the population activity $A_t = \sum_{j=1}^{N} a_{j,t}$ which count the number of spikes

in the interval $[t, t + \Delta t]$. If not stated differently, the bin width for single neuron activity is chosen to correspond to the refractory period $\Delta t = \tau_{\text{ref}}$. This leaves at most one spike per bin for each neuron.

Statistical quantities are generally estimated from multiple network realizations. If not stated otherwise, the median over 100 different stochastic seeds is calculated and errors indicate the 95 % confidence intervals.

3.5.1 Autocorrelation

The autocorrelation time τ is an important measure to characterise the dynamical state of the network (cf. Section 2.4). The population autocorrelation function of the network activity with a total length of T timesteps is estimated by

$$C(k) = \frac{\sum_{t=1}^{T-k}(A_t - \bar{A}_k)(A_{t+k} - \bar{A}'_k)}{\sum_{t=1}^{T-k}(A_t - \bar{A}_k)^2}\,, \tag{3.3}$$

with $\bar{A}_k = \frac{1}{T-k}\sum_{t=1}^{T-k} A_t$ and $\bar{A}'_k = \frac{1}{T-k}\sum_{t=1}^{T-k} A_{t+k}$, the mean of the observed time series and the shifted time series respectively [Spitzner et al. 2020].

The autocorrelation time is then determined by fitting $C(k)$ with an exponential decay

$$C(k) = C_0 e^{-k\Delta t/\tau}\,. \tag{3.4}$$

Fits are performed in Python with the `scipy.optimize.curve_fit` function [Virtanen et al. 2020].

3.5.2 Linear Classifier

In the reservoir computing setup for classification tasks, the network state is extracted with a liner regression readout. At every timestep t, a linear readout maps the network activity state vector a_t of size N onto the dummy coded binary label $\mathbf{y_t}$ of the n-class classification task

$$\mathbf{y_t} = M\mathbf{a_t}\,, \tag{3.5}$$

with the $n \times N$ weight matrix M. The classifier is trained with linear regression on the training data and the accuracy score is evaluated on the testing samples. In this thesis the `LinearRegression` implementation from the `scikit-learn` toolbox [Pedregosa et al. 2011] is used.

3.5.3 Mutual Information

Apart from the classification accuracy, the mutual information (MI) I between classifier predictions and input label is used. It allows a quantization of classification results that is

less specific to a certain task compared to the accuracy because it highlights the correlations between labels and predictions. The information theoretic measure is derived from the entropy H by

$$I(X;Y) = H(X) - H(X|Y)\,. \tag{3.6}$$

It quantifies the amount of information in one random variable X minus the additional information gained from observing X given observation of another variable Y. This leaves only the information already known from Y, which is common to both variables without observation of the other. Inserting the definition of entropy yields

$$I(X;Y) = \sum_{y\in Y}\sum_{x\in\mathcal{X}} p(x,y)\log\left(\frac{p(x,y)}{p(x)\,p(y)}\right), \tag{3.7}$$

with the individual and shared probability distributions p respectively. The logarithm is to base 2, generating results in binary digits (bits). The MI is calculated with the `metrics` score implementations from `scikit-learn`.

4 Control of Network Dynamics

In this chapter the general behaviour of the neuromorphic hardware and the implemented network are investigated. Characterization results of the noisy hardware parameters are presented in Section 4.1. They demonstrate the ability to configure desired behaviour of single neurons. Section 4.2 deals with the full network and the regulation of population activity with plasticity. In Section 4.3 the tuning of critical dynamics is analysed.

4.1 Characterization

Before conducting experiments with neuron populations, single neuron parameters are calibrated. Figure 4.1 shows the distribution of time constants and potentials before and after calibration with `calix`. Recordings were taken with the MADC directly from the host computer. Calibration significantly reduces the spread of parameter values. These measurements are especially necessary to determine the parameters for the reference implementations in simulation (cf. Chapter 5). For the uncalibrated measurements, parameters were configured to the mean values of the calibration results and varied by a random jitter. Average values of potentials were chosen to exhaust the dynamic range between reset, leakage, and threshold potentials available on the hardware. Time constant strike a balance between large temporal dynamics and low levels of noise.

An important parameter for network dynamics is the synaptic amplitude I_0 (cf. Eq. 2.5). Since it scales the total current that flows onto the postsynaptic membrane it has a similar effect as the indegree K_{in} that is used to tune the network state. Therefore, it is important to accurately determine this value and ensure its stability. Figure 4.2c shows excitatory postsynaptic potentials (EPSPs) for different synaptic weights. The neuron of which the membrane voltage is recorded, was stimulated with a single spike at time $t = 10\,\text{ms}$. Measurements were performed for every value of the 6 bit synaptic weight w and averaged over traces for all 512 neurons. The previously measured leak potentials are subtracted from the membrane traces.

In this single spike setting, the coupled differential equations Eq. (2.1) and Eq. (2.3) are easily solved (e.g., using the method of variation of constants). The solution for the

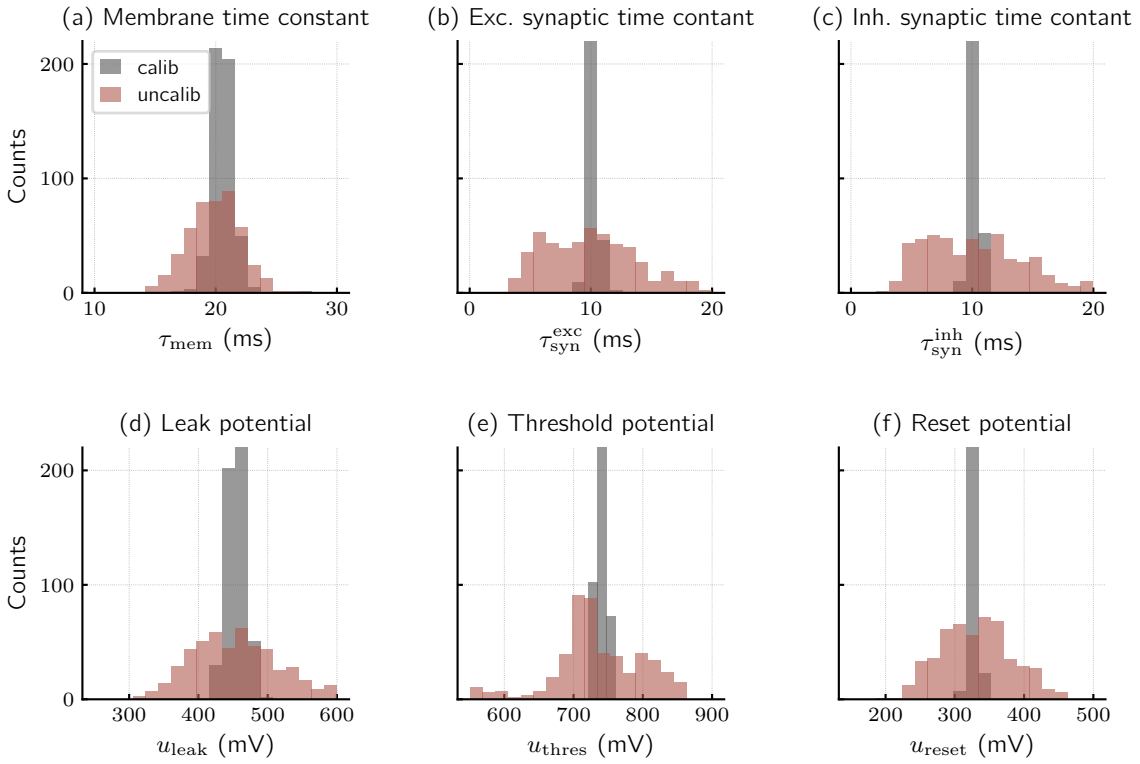


Figure 4.1: Distribution of time constants and potentials before and after calibration. Calibration significantly reduces parameter variation across synapses. All values are given in the biological equivalent time domain.

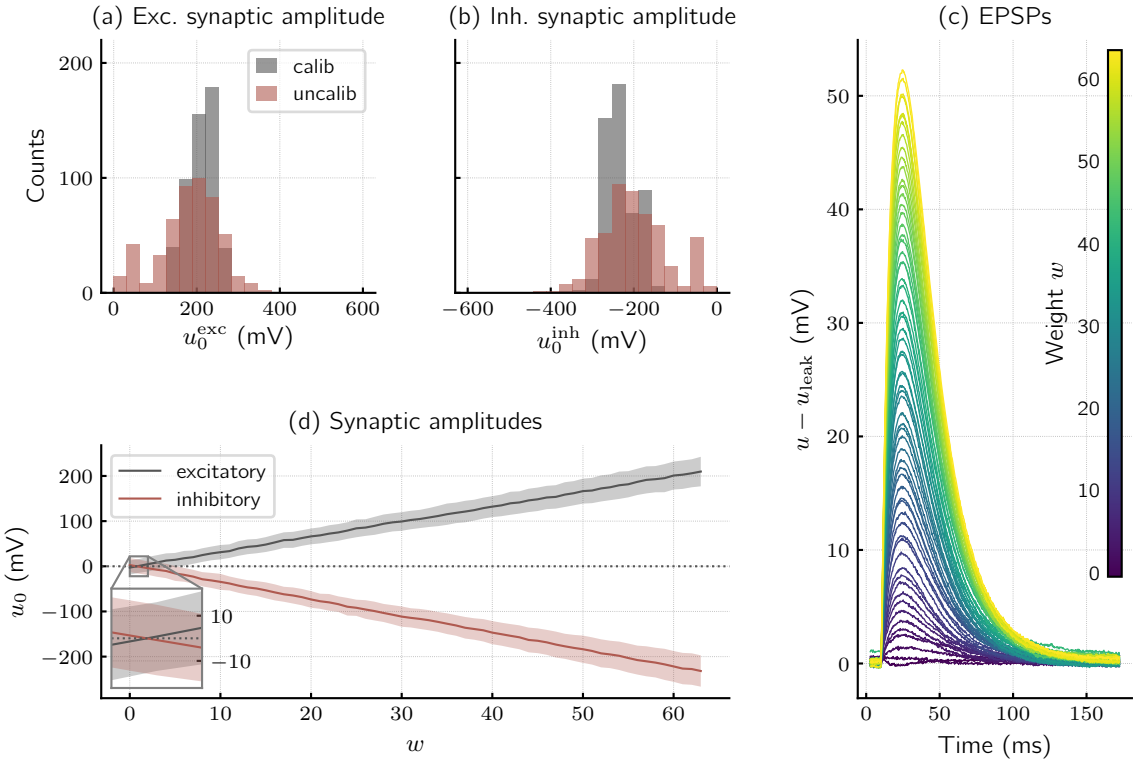


Figure 4.2: Measurement of the synaptic amplitude and its dependence on the synaptic weight. (a) The spread of u_0 across different neurons is reduced by applying calibration. (c) Stimulation of a neuron with a single presynaptic spike and different weights leads to increasing EPSPs. (d) The effective synaptic amplitude depends linearly on the weight. Errors indicate the standard deviation of the average over all neurons. The insert highlights the offset in the relation of weights and current.

membrane potential from the time of arrival of the single spike is

$$u(t) = u_0(w) \cdot \frac{\tau_{\mathrm{syn}}}{\tau_{\mathrm{syn}} - \tau_{\mathrm{mem}}} \cdot \left(e^{-t/\tau_{\mathrm{syn}}} - e^{-t/\tau_{\mathrm{mem}}} \right) + u_{\mathrm{leak}} \,. \tag{4.1}$$

The amplitude $u_0(w) = I_0/g_{\mathrm{leak}} \cdot w$ is determined by the synaptic amplitude and leak conductance and in theory scales linearly with the weight w. Equation (4.1) can be fitted to the single neuron traces (Fig. 4.2c) to determine the true relation between the current placed on the postsynaptic membrane and the synaptic weight. Here, the time constant in the equation were preselected to the target values of the calibration to achieve a more stable fit. The distribution of u_0 across all neurons is shown in Figure 4.2a, where the values are estimated from the EPSPs with maximal weight.

Average amplitudes for all weights are plotted in Figure 4.2d for excitatory and inhibitory synapses. The linear relation shows a small, negative offset for weight zero. An excitatory synapse that is configured with weight zero to not transmit spikes, can have a small inhibitory effect. (This is also visible from the slightly negative amplitude of the

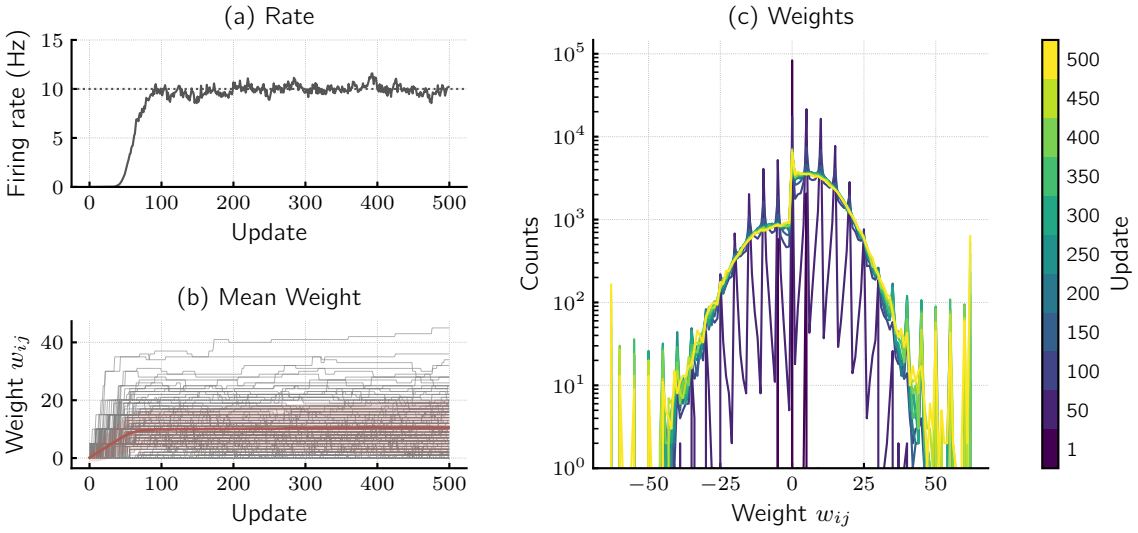


Figure 4.3: Evolution of the network rate and weight distribution during the adaptation phase. An experiment with indegree $K_{\text{in}} = 130$ is shown. (a) The network activity increases during early plasticity updates until it reaches the target value. This is caused by the homeostasis probabilistically increasing synaptic weights rapidly from their initial values at zero. (b) The mean weight (red) is increased during early updates. Individual weights (grey) are only changed at average intervals of $1/p_{\text{update}}$. Only a selection of weight traces is shown. (c) Peaks in the weight distribution lie at multiples of 5, the maximal possible positive weight update determined by $\eta \cdot \nu_{\text{target}}$.

EPSP in Figure 4.2c.) Conversely, an inhibitory synapse can have a small excitatory effect. Crucially, this violates Dale's law to an extent, as synapses are no longer purely excitatory or inhibitory. For the experiments in this chapter, the finding is ignored, but its impact is further analysed in Chapter 5. The configuration of sparsity in the reservoir is not effected by this behaviour, since it is realised using the address coding scheme and not by setting the synaptic weights to zero. Still, plastic weights controlled by the homeostatic regulation are exposed to this limitation.

4.2 Homeostatic Rate Regulation

In this section, the accurate regulation of the network rate with the plasticity rule Eq. (3.1) is demonstrated. Figure 4.3a shows the development of the average single neuron firing rate over 500 plasticity updates during the adaptation phase. After every update, a static experiment with fixed weights and a duration of $T = 80\,\text{s}$ is recorded to obtain rate averages. To that end, the on-chip spike sources are simply kept running after adaptation.

At the beginning of the experiment all weights are initialised at zero, thus no output is produced. The homeostasis therefore increases a fraction p_{update} of weights by the maximum possible value of $\eta \cdot \nu_{\text{target}} = 5$. This is also visible in Figure 4.3c, which depicts the evolution of the weight distribution. Especially during the early updates, there are

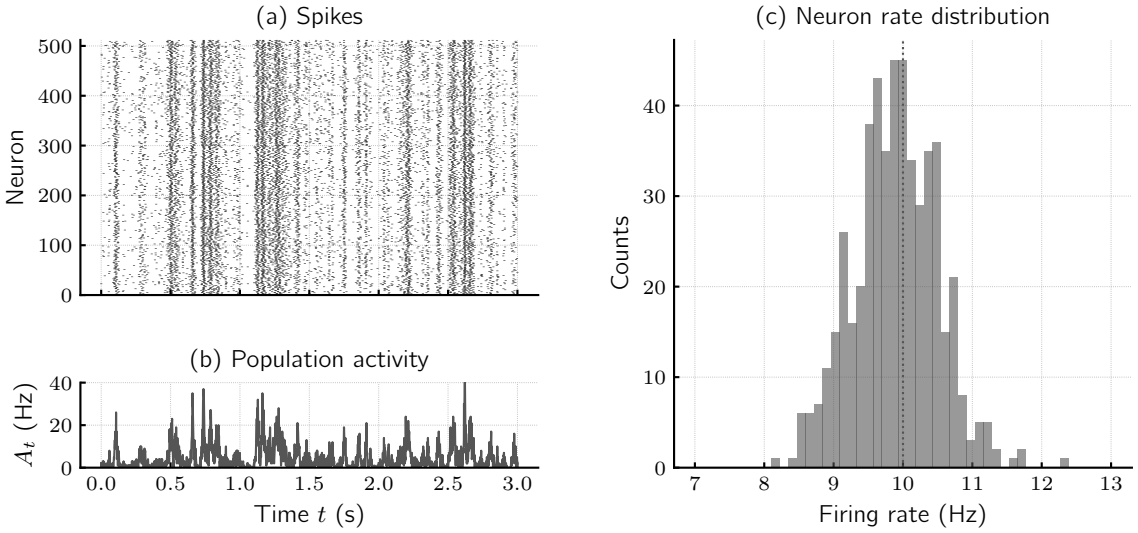


Figure 4.4: Spike raster plot and single neuron rate distribution after homeostatic regulation. (a) Spike times of all 512 units visualise the network dynamics. An extract of an experiment after adaptation with indegree $K_{\text{in}} = 130$ is shown. (b) The population activity is estimated from the spike events. (c) Average single unit spike rates estimated over the whole experiment of 80 s closely spread around the target value.

sharp peaks in the distribution at multiples of 5. The network tries to push as many of its weights as allowed by p_{update} as high as possible during these early updates.

The development of the mean weight after each update is plotted in Figure 4.3b. Here again it can be seen that during the first updates the weights increase by the maximum possible rate allowed by the homeostasis. The figure also show the evolution of a number of randomly chosen individual weights. They change more frequently during early updates, but stay constant for an average of the inverse update probability $1/p_{\text{update}} = 40$ updates.

After a few dozen updates the network starts to generate output spikes and the rate increases. A number of synapses have become strong enough to generate postsynaptic currents that can increase the membranes of some neurons above the threshold. Even though inhibitory weights are exposed to the same update rule and treated in exactly the same way, the larger number of excitatory synapses leads to a net positive current. The speed of this initial increase is determined by the update probability, a larger value leads to a faster rise in activity (data not shown). It has an effect comparable to a learning rate.

When the output rate surpasses the target, the homeostasis decreases the weights, reducing the network rate again. As a result, the rate fluctuates around the target value. A distribution of the 512 single neuron rates is shown in Figure 4.4 estimated from the final static experiment after adaptation. They closely spread around the target value. The figure also depicts an exemplary spike raster plot of the activity after the final update.
The update probability influences the stability of the final activity, larger p_{update} lead

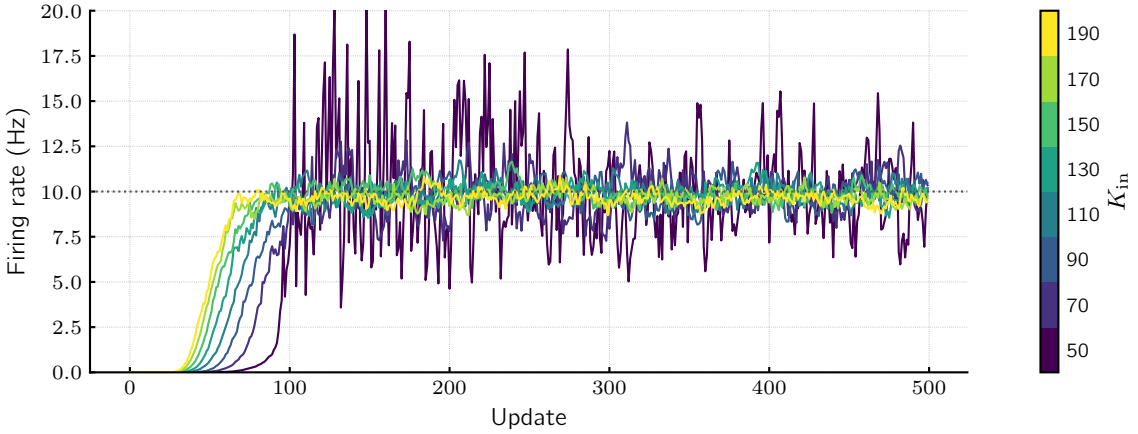


Figure 4.5: Development of the network activity for different indegrees. The network activity reaches the target value faster for higher K_{in}. In contrast, with less input at low indegrees, single synapses are more important and need stronger weights to generate the same output. This also leads to strengthening of recurrent synapses leading to self-amplifications and larger fluctuations in the mean firing rate.

more fluctuations. A homeostasis that acts on all weight simultaneously, would lead to extreme overshoots not capable of stabilizing dynamics, justifying the probabilistic approach. However, because of the integer arithmetics there is a minimum values for p_{update} below which any positive updates are floored to zero, such that weights are never updated.

The total postsynaptic current to a single neuron membrane not only depends on the value of the weights, but also the indegree K_{in}. In Figure 4.5, the rate development is plotted for different values of K_{in}. For higher values, corresponding to more input projections per neuron, the target rate is reached faster. At lower K_{in}, the network takes longer to build up activity; here weights need to be larger to compensate for the reduced input. This is demonstrated by the distribution of the weights after the final update in Figure 4.6b. For smaller K_{in} there are more large weights. At $K_{in} = 50$ the distribution even starts to saturate, such that it appears to be cut off at high weights. The impact of this will be further shown in simulation in Chapter 5.

However, the indegree also impacts the stability of the network rate. While the target rate is generally reached for a wide range of values (Fig. 4.6a), it becomes less accurate at lower K_{in}. With fewer inputs, more emphasis is placed on individual synapses and single weight updates have a larger impact on the neuron rates. Additionally, recurrent connections are also strengthened in this situation leading to self-amplifying behaviour. Thus, the activity starts to fluctuate around the target more strongly (cf. Figure 4.5).

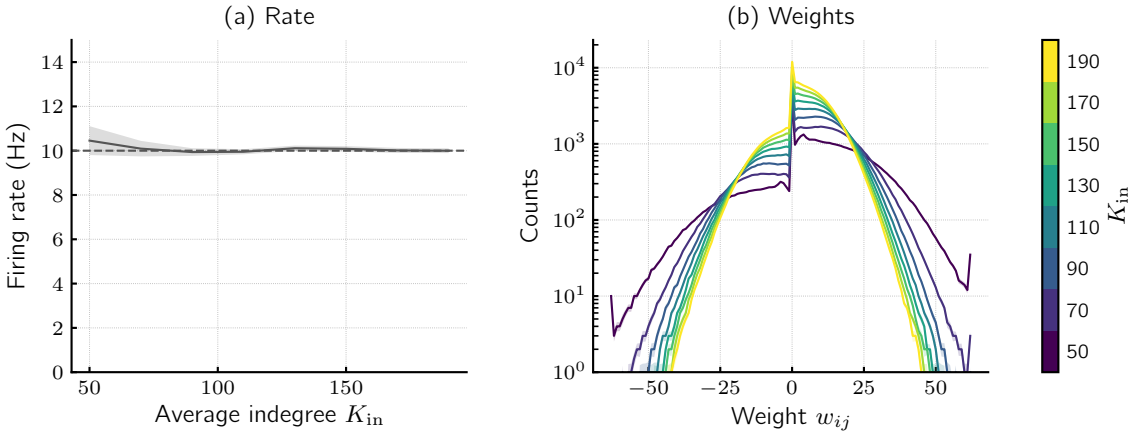


Figure 4.6: Average network activity and weight distributions for different indegrees. (a) The network rate reaches the target value of the homeostasis for a wide range of indegrees. (b) The weight distribution shows more large weights towards lower K_{in} placing more importance on single synapses. At $K_{\text{in}} = 50$ the distribution starts to saturate such that it is cut off at high weights.

4.3 Control of Autocorrelation

The autocorrelation (AC) is used to characterise the network dynamics for different indegrees. Figure 4.7a shows the autocorrelation function estimated by Eq. (3.3) from the activity of static experiments for various values of K_{in}. The dependence of the time constant as estimated by an exponential fit (Eq. 3.4) on the indegree is plotted in Figure 4.7b.

At low indegrees, recurrent connections in the network are strengthened, increasing internal correlations and maintaining activity. This manifests in an increase of AC timescale towards lower indegrees which expresses the increase in memory in the network. In the framework of a branching process as described in Section 2.4, the AC time constant is a measure for the branching parameter, and therefore the dynamical state of the network. The change of τ over more than one order of magnitude demonstrates the control of the dynamical regime in the network that is achieved with the indegree.

Figure 4.8 shows spike rasters and corresponding network activity for excerpts of the static runs from which the AC is estimated. The different dynamical regimes (cf. Figure 2.2) are clearly visibly. For low indegree $K_{\text{in}} = 50$ the activity is fluctuating, showing intervals of larger bursts. At high indegree $K_{\text{in}} = 190$ it is more homogeneous in time and synchronisation appears only on smaller timescales in very short bursts. Going to even lower indegrees would make the network unstable and lead to bistable activity with intervals of extreme spiking or near silence. This effect generates higher variation in the population rate between different trials, which is to some extend already visible in Figure 4.6a for $K_{\text{in}} = 50$.

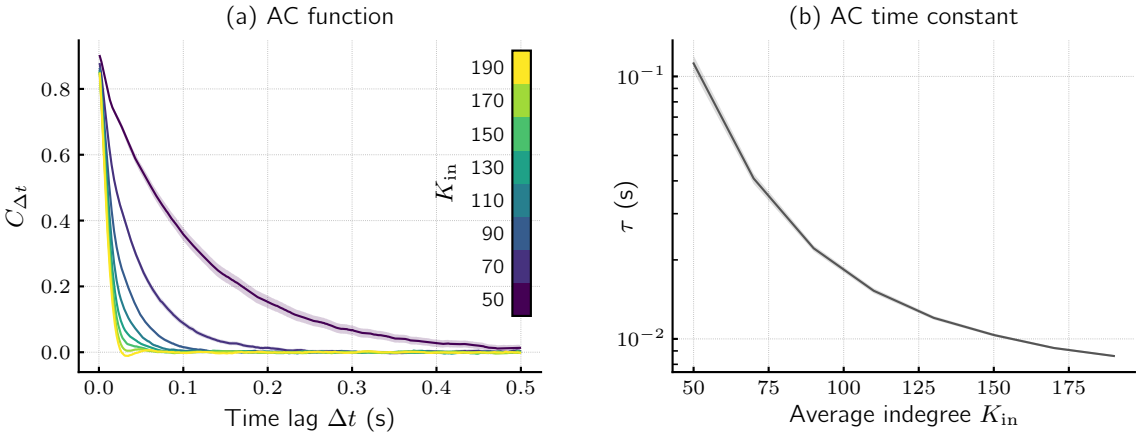


Figure 4.7: Autocorrelation functions and time constants for different indegrees. (a) The autocorrelation functions are estimated from static experiments after adaptation with various indegrees. Their exponential decay justifies the fit with a corresponding function. (b) Autocorrelation times τ monotonically depend on the indegree, which allows selecting desired network dynamics by choosing the coupling to external input.

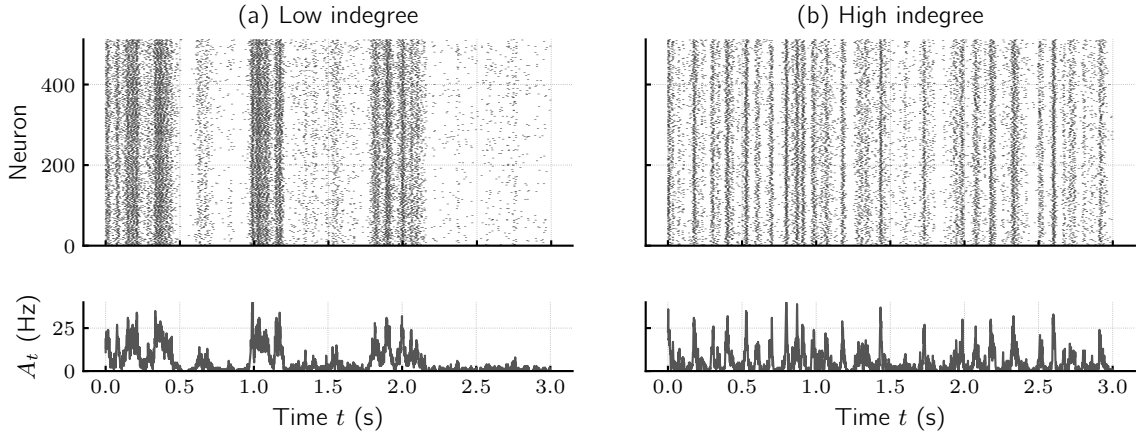


Figure 4.8: Spike raster plots and population activity of networks in different dynamical regimes. (a) At low indegree of $K_{\text{in}} = 50$ the activity shows fluctuations on diverse timescale with large bursts and more quiet intervals. (b) For high indegrees $K_{\text{in}} = 190$ activity changes more homogeneously on shorter timescales.

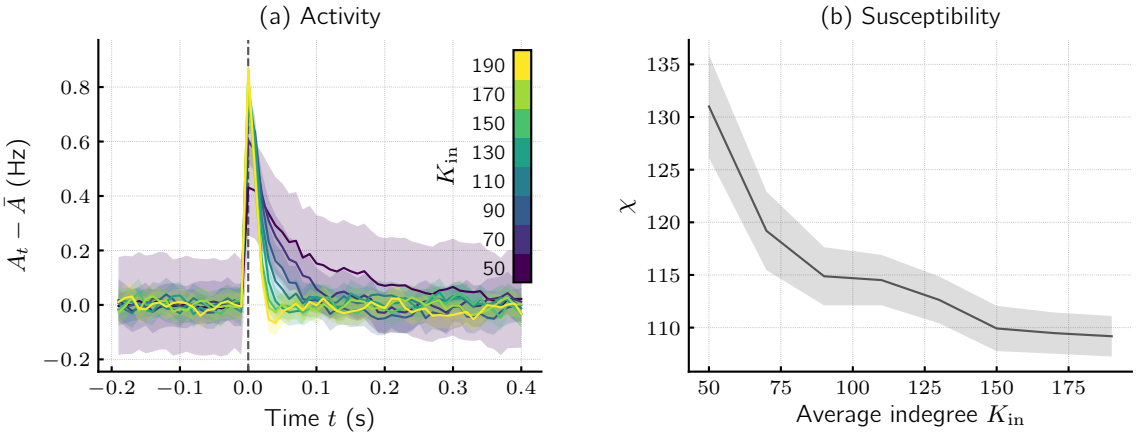


Figure 4.9: Perturbed network activity and susceptibility for different indegrees. The input Poisson noise is perturbed by an additional spike at $t = 0$. (a) The response of the network activity is visible much longer for systems at low indegree. (b) A susceptibility can be defined by the number of additional spikes caused by the perturbation. Data is averaged over experiments with a perturbation for every input neuron as well as 1000 different statistical realizations. Confidence intervals are only estimated over the statistical seeds.

4.4 Susceptibility

The different dynamical states can also be demonstrated in another way. Drawing from the model of a branching process (Sec. 2.4), the activity at one timestep will depend on the activity in a previous step by means of the branching parameter m (Eq. 2.7). For a close-to-critical network with m nearing one, this means that a single spike will result in a strong response of the network at later steps. Instead of looking at internal activity, here the input Poisson stimulus is perturbed by a single additional spike. This is achieved by sending an external spike event to the chip on top of the noise generated by the on-chip background sources. In Figure 4.9a the normalised network activity in response to a spike sent at time $t = 0$ is plotted. The spike is sent to each of the 256 input channels once and results are averaged. Additionally, each curve is averaged over 1000 network realizations with different statistical seeds. Confidence intervals are only calculated from the 1000 realizations after averaging over neurons. Despite the large number of trials, the intervals are quite big for the lowest K_{in}, hinting at less stability in the network. All graphs are corrected to the mean activity before the perturbation.

Systems with low indegree, which show higher AC, maintain the perturbation much longer in the output activity. These systems exhibit a large amount of fading memory. At high indegrees the activity promptly return to its base level. In this input-driven regime, past activity is less relevant. However, the maximum peak is higher, input-driven networks instantaneously react much more pronounced to the perturbation.

The extra activity caused by the perturbation determines a measure of susceptibility χ.

It can be estimated from the difference of average spike numbers before and after the perturbation in the intervals $[-0.2\,\mathrm{s}, 0)$ and $[0, 0.2\,\mathrm{s}]$. As indicative for a system approaching a phase transition, χ increases towards low indegrees (Fig. 4.9b).

5 Comparison of Hardware and Simulation

The results presented in the previous chapter demonstrate that it is generally possible to achieve fine control over dynamics in the homeostatically regulated networks on neuromorphic hardware. Now, specific intricacies of implementing experiments on the analog hardware system are examined. To that effect comparisons with an implementation of the reservoir in simulation are drawn. Section 5.1 gives the result from the simulation and discusses the differences to those from hardware. In Section 5.2 the influence of hardware constrains on the results are analysed by specifically removing these in simulation. The synaptic amplitude offset previously mentioned in Chapter 4 is further analysed in Section 5.3. Finally, Section 5.4 describes an undesired behaviour of the hardware synapse implementation found during experiments, that can explain some differences found between hardware and simulation.

Simulation results are only averaged over 50 network realizations with different statistical seeds as they run considerably longer than emulations on chip.

5.1 Brian Simulations

Figure 5.1 shows the population rates and weight distributions obtained from simulations with various network indegrees, in the same way this was shown in Figure 4.6 for the implementation on BrainScaleS-2. Similarly, Figure 5.2 depicts the AC functions and time constants from simulation, comparable to those on hardware (Fig. 4.7). These figures demonstrate the same kind of control of network regimes with timescales from tens to hundreds of milliseconds. However, towards low indegrees the networks in simulation start to getting unstable as seen from the rate plot. Also, the weights (Fig. 5.1b) exhibits much more saturation than previously observed on hardware. A difference is also directly visible in Figure 5.3 which compares the raster activity plots from hardware and simulation at $K_{\text{in}} = 50$. For the simulation the activity already shows extreme phases of spiking, which is not yet the case on hardware at the same indegree.

This difference in dynamics is not a result of generally lower stability in the simulated network, but rather the consequence of a global shift in the indegree regime between hardware and simulation. At $K_{\text{in}} = 50$ the AC time in simulation is considerably larger than on hardware, so much so that the AC function does not really follow an exponential decay for some of the network instantiations and estimation of confidence intervals fails

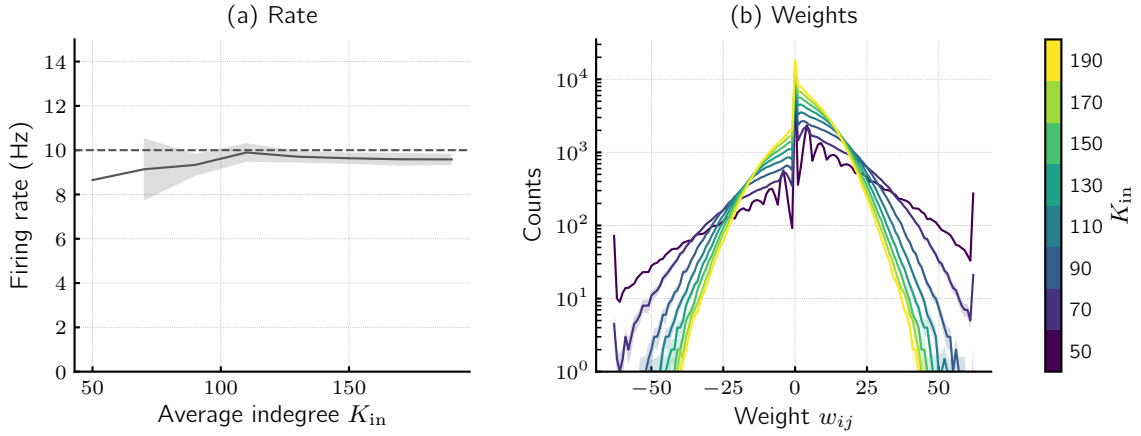


Figure 5.1: Average network activity and weight distributions for simulations with different indegrees. (a) The network firing rate approaches the target of the plasticity rule for a range of indegrees. Towards low K_{in} larger errors hint at instability in the network. In fact, for the lowest value the estimation of confidence intervals fails and errors are not shown. (b) The weight distribution shows even larger saturation at K_{in} than on hardware.

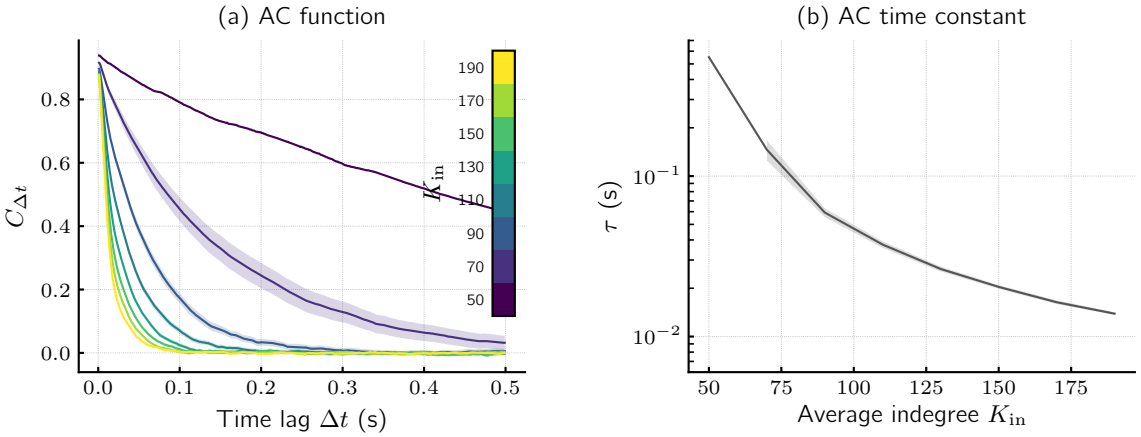


Figure 5.2: Autocorrelation functions and time constants for simulations with different indegree. Autocorrelation in simulation is generally larger than on the hardware for the same indegree range. (a) The autocorrelation function for $K_{\text{i}} = 50$ barely resembles an exponential decay. (b) The corresponding fit shows a much larger AC time compared to the hardware, but fails to estimate confidence intervals for this indegree.

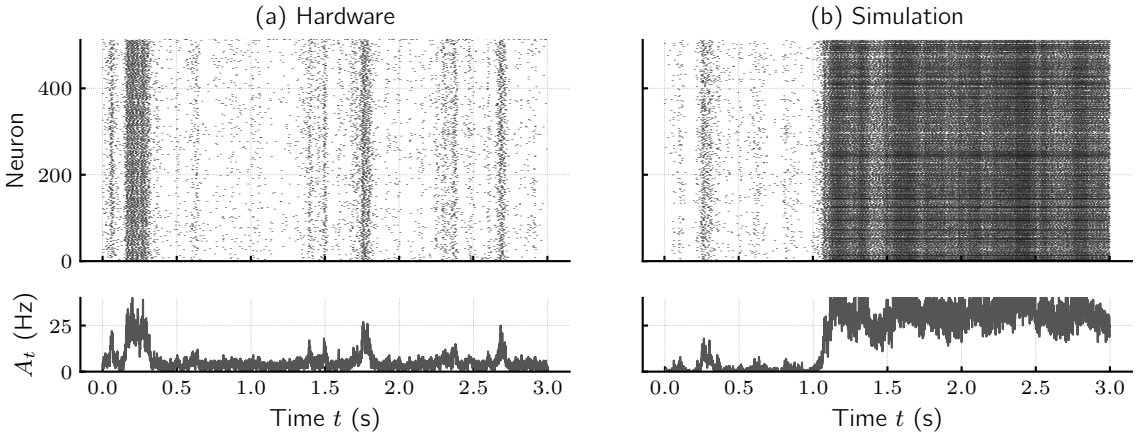


Figure 5.3: Comparison of spike raster plots from hardware and simulation. At the same indegree of $K_{\text{in}} = 50$ the dynamics in software are considerably less stable than on the hardware. The simulation already exhibits states of extreme spiking that are observed on hardware only for lower indegree. This indicates a shift of the hardware dynamics towards the input-driven regime.

(Fig. 5.2). In Figure 5.4 the AC time constants from hardware and simulation are plotted together for direct comparison. The curves seem to be shifted along the indegree axis by roughly $\Delta K_{\text{in}} = 25$.

While the exact reason for this shift is not understood in full detail with certainty at the time of writing, the following sections highlight various differences between simulation and hardware and evaluate their possible impact to obtain a reasonable explanation for the observation. In any case, the experiments presented demonstrate that qualitatively the same behaviour is observed both on hardware and in simulation.

5.2 Hardware Parameter Constraints

In simulation, the impact of parameter and temporal noise can be directly analysed by changing the stochasticity introduced into the equations. This can be used to rule out that the actual noise realised on hardware was estimated wrongly, which might impact the dynamical regime in the conducted simulations. To this end the ideal equations are solved without any fixed-pattern or temporal noise. In the comparison in Figure 5.4 the resulting network firing rates and AC time constants for the indegree range are plotted. The absence of noise does not have any apparent impact on the network dynamics. The homeostatic regulation seems to work independently from the noise levels on hardware. Therefore, it is highly unlikely that differences in parameter noise realization could cause the observed shift in dynamics.

Another specific implementation detail determined by the hardware is the use of integer weights with saturating arithmetics. In simulation this constraint is explicitly introduced

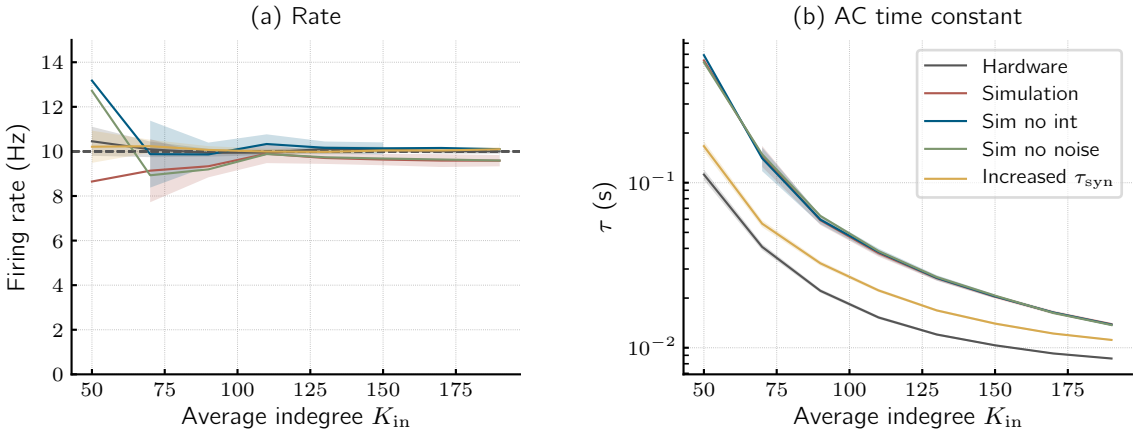


Figure 5.4: Direct comparison of average firing rates and autocorrelation time constant between hardware and simulation. (a) The average firing rates for all experiments fall around the target of the homeostatic plasticity. For simulations at lowest indegree, values vary stronger indicating less network stability. (b) Autocorrelation time constants between hardware and simulation seem to be shifted along the indegree axis by about $\Delta K_{\text{in}} = 25$. Time constant for simulations without various constraints of the hardware barely differ, indicating that they do not impact dynamics noticeably.

by integer casting. To analyse the impact of these arithmetics, this step can simply be omitted. In that case, the weight distribution shown in Figure 5.5 does not exhibit the saturation at maximal values that was previously observed on hardware (Fig. 4.6). However, this change also does not have a significant impact on network dynamics. The output rates lie close to the target within the range expected from the estimated errors and the AC times, plotted in the comparison in Figure 5.4, show no significant difference to the simulation with integer arithmetics. Thus, the restriction to integer weights on hardware does not have an impact on the working of the homeostatic plasticity.

5.3 Synaptic Amplitude Offset

The offset in the synaptic amplitude (cf. Figure 4.2) that was discovered in Section 4.1 was also incorporated into the implementation of the simulation. Reference simulations without the offset show no difference in the resulting dynamics, ruling it out as the cause of the shift in the dynamical regime between hardware and simulation. This justifies that the offset was excluded from further discussion in Chapter 4. Within a certain degree the experiments presented in this thesis do not depend on strict adherence to Dale's law.

Still, a fix for the behaviour on chip is presented which could be used for experiment where such strict adherence is required. It is achieved by simply disallowing weights with values of zero to be realised in the network. Using the plasticity mechanism implemented on the PPUs, any weight that would be set to zero is instead clipped at a value of one. Effectively, the saturation regime already present in the integer arithmetic is simply re-

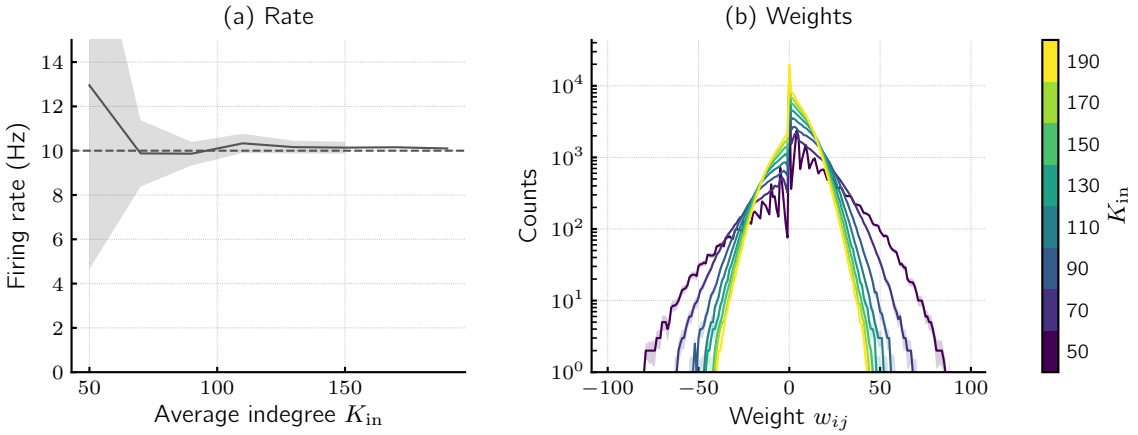


Figure 5.5: Average network activity and weight distribution for simulations without integer weights. Histograms of the weight distribution are still binned at integer values. However, the extended range reveals that the distribution does not saturate at large weights. This is in contrast to the distribution on hardware that is cut off at the maximal weight possible in saturating integer arithmetics.

duced by a single value. Because the offset does not exactly correspond to the synaptic amplitude, this leads to a slightly positive average EPSP for the lowest weight. All realised synapses will always leak a small amount of current whenever an input event is received. Still, this is potentially more desirable than violating Dale's law in certain situations. Especially, sparsity can be implemented by the address coding scheme that is not impacted by these leaks, omitting the requirement for zero valued weights. Experiment with this clipping mechanism on hardware showed no impact on the observed dynamics (thus data is not shown).

5.4 Synapse Saturation

During analysis of the shift between hardware and simulation, an undesired behaviour of the current version of the HICANN-X silicon implementation was discovered. To highlight the effect, Figure 5.6 shows simple rate tuning characterizations for a feed forward network configuration. The network is set up in the same way as in previous experiments, but no recurrent connections are realised: $K_{\text{rec}} = 0$. It receives purely excitatory Poisson input of different rate (color coded in the figure). The dependence of the network output rate on the input degree is plotted. As expected, higher input rates and larger K_{in} lead to stronger network activity. However, the rate of the network implemented on hardware starts to saturate in comparison to the simulations. A maximum possible spike rate lies at around 250 Hz. This effect goes beyond a plain saturation of the synaptic input currents.

While sustained activity of such large rates is rarely required, this can have an influence on the network dynamics in the adaptation phase of the homeostatic regulation. For input

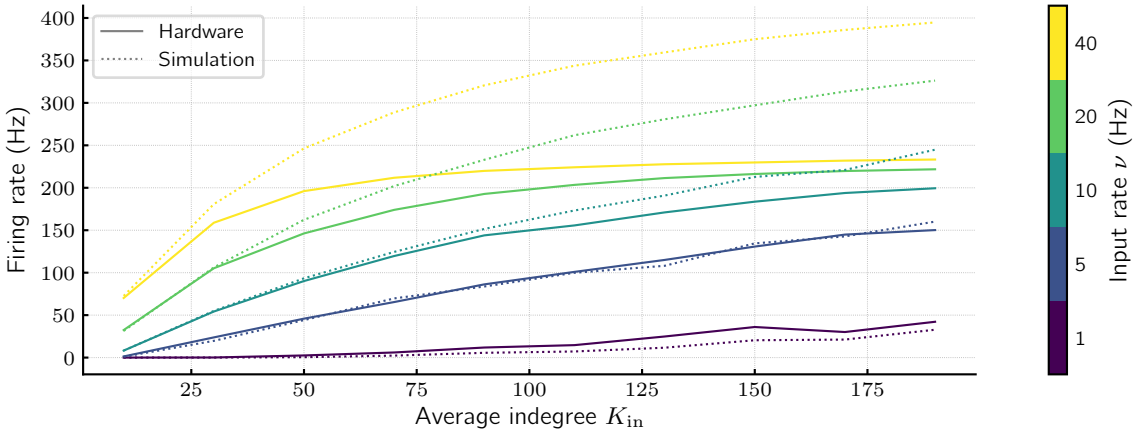


Figure 5.6: Comparison of network rate tuning curves from hardware and simulation. The dependence of the output firing rate on the indegree is plotted for different stimulation rates of a feed-forward network without recurrence. Input is purely excitatory. Saturation of firing rates from the hardware (solid lines) can be seen in comparison to the implementation in simulation (dashed lines).

spikes with a fixed rate of 10 Hz the saturation is negligible. However, the recurrent spikes can reach much higher frequencies during adaptation, especially in the regime with low coupling to the input where recurrence is generally more important (cf. Figure 4.5). A saturation of high rates at these lower indegrees means that the network depends more strongly on the activity provided by the input spikes, resulting in more input-driven dynamics. This corresponds to a network with higher indegree, which is exactly what is observed with the shift in the AC time constants.

While writing this thesis, a bug in the synaptic input circuits was discovered that leads to a rate dependence of the synaptic time constants and can explain the observed behaviour. At higher rates the effective time constant are smaller than expected, so the synaptic current decays faster. This decreases the synaptic input strength which leads to lower firing rates. To demonstrate that this can in principle explain the observed shift in the AC times, the synaptic time constants are calibrated to a higher target value of $\tau_{\text{syn}} = 15\,\text{ms}$. This setting is compared to the previous experiments with the original value of $\tau_{\text{syn}} = 10\,\text{ms}$, and indeed the shift is decreased (Fig. 5.4). While this goes to show that the effect impacts the AC timescale in the expected way, the rate dependence of time constant itself is not easily fixed by a re-calibration. This poses a problem for the dynamical adaptation of the rate. Time constants can only be tuned to correspond to the correct values for a selected rate, e.g., that of the steady state. Thus, the observation only acts as an indicator of the underlying cause of the shift. Especially, the synaptic amplitude also had to be decreased for the network with larger τ_{syn} to avoid the usual saturation of synaptic input currents that occurs for large synaptic currents, further deforming the result.

Still, the analysis demonstrates that the bug can plausibly explain the shift between hardware and simulation, though other effects can not be ruled out completely. While the hardware will be adapted in future silicon versions, for now the behaviour can be counteracted to a certain degree. The homeostatic regulation is impacted by this only quantitatively anyways, qualitatively the expected behaviour is observed regardless of the saturation.

6 Classification Tasks

This chapter presents first results obtained by using the SNN in a reservoir setup for classification tasks. The network itself is augmented by a linear readout layer as was shown in Figure 3.1. In the following sections, samples of the respective datasets (cf. Section 3.4) are injected into networks that were homeostatically adapted at different indegrees. This allows to analyse the impact of the dynamical state on classification performance.

6.1 Poisson Patterns

Sets of Poisson patterns are used to investigate the influence of signal-to-noise ratio and input rate changes on classification. The task is designed to discern between two randomly generated Poisson patterns. Two versions of signal-to-noise realizations are differentiated. In the simplest case, the stimuli are sent as external spikes in addition to the events generated by the background sources that are left running after the adaptation phase with a rate of ν_{adapt}. When presenting the patterns, the total stimulation rate for the network is then $\nu_{\text{adapt}} + \nu_{\text{ext}}$. Here, a rate scaling factor γ is defined as the relation of external and background rates $\nu_{\text{ext}} = \gamma \cdot \nu_{\text{adapt}}$, which determines the signal-to-noise ratio. In more detail, it also determines the amount of extra activity to the reservoir during pattern presentation compared to the adaptation phase. For $\gamma = 0$ the network only receives noise, just like in the previous static experiments. Increasing γ introduces additional spikes that elicit a stronger response from the network. This setting is similar to the perturbation in Section 4.4, but goes beyond a simple perturbation. Instead a linear readout is tasked to discern between different patterns of perturbations.

In Figure 6.1 the performance of classifiers trained on the single neuron activity vectors a_t at different timesteps is shown. It is quantified by the MI between predictions and labels. Beneath that, the mean network activity at each timestep is plotted. The patterns of interest start at $t = 25\,\text{ms}$ and last until $t = 50\,\text{ms}$ as indicated by the dashed lines. The rate scaling factor for this experiment is $\gamma = 4$, the network sees in total five times the activity during the stimulus than before and after.

The onset of patterns is clearly visible by an increase of the classifier performance. Simultaneously, the network activity can be seen rising considerable from the target value of 10 Hz that was adjusted by the homeostasis. During the presentation of Poisson pattern, the classification performance is similar for all network indegrees. Towards the end,

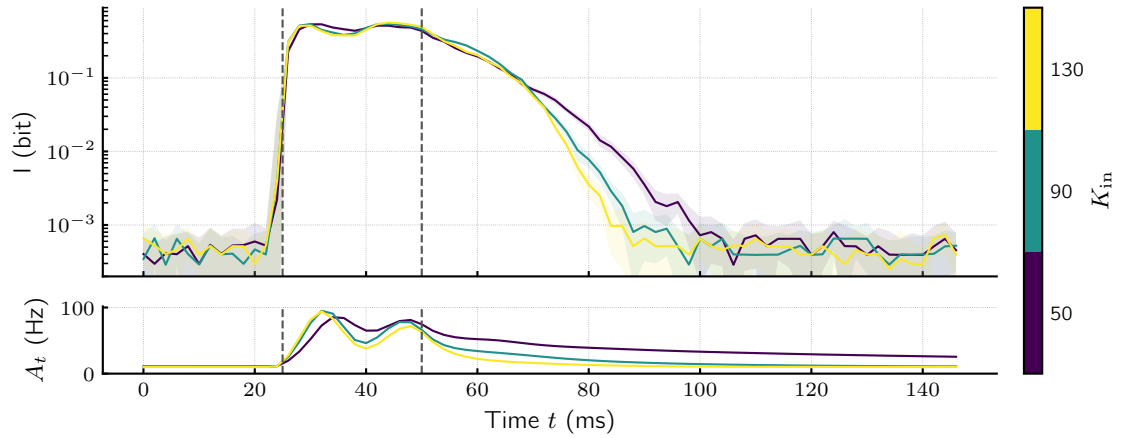


Figure 6.1: Classification performance for the Poisson pattern task during persistent background noise stimulation. The linear readout acts on the activity vectors at different timesteps in the network response. Patterns last from 25 ms to 50 ms indicated by the dashed lines. For low network indegrees the classifier demonstrates larger fading memory, while the performance is the same for all networks during the patterns (top). The fading of activity is also visible in the population rate (bottom).

the network activity starts to decrease for input-driven networks and afterwards rapidly converges back to the target rate. In contrast, for the fluctuating dynamical regime that exhibits large autocorrelations, the activity fades much slower. This is what is expected from the results for the susceptibility in Section 4.4. Here, the perturbation with the stimulus is much stronger and therefore the activity is less noisy despite averaging over a smaller number of statistical seeds. It still fades on a timescale of about 100 ms for an indegree of $K_{\mathrm{in}} = 50$.

This fading memory is also visible in the classification performance. For high indegrees the MI drops to chance level faster than for networks with lower indegree. The performance is significantly higher even 40 ms after the end of the pattern, indicating that the networks maintains usable information during that time. This is in accordance with the larger autocorrelation time observed for these networks.

The impact of the rate scaling factor is visualised in Figure 6.2. There, the dependence of the classifier performance on γ is drawn at three positions in the time development that was previously shown. At the beginning of the pattern at $t = 30$ ms, networks with high indegree perform slightly better than those with lower input coupling, especially for little additional input at low γ. These networks are highly input-driven and can react rapidly to the available information. The situation is inverted after the stimulus at $t = 80$ ms. Networks closer to criticality outperform input-driven ones, especially for high scaling factors the network response remains informative. During pattern presentation, network dynamics have little impact on the performance, but MI generally increases for larger γ with a stronger signal-to-noise ratio.

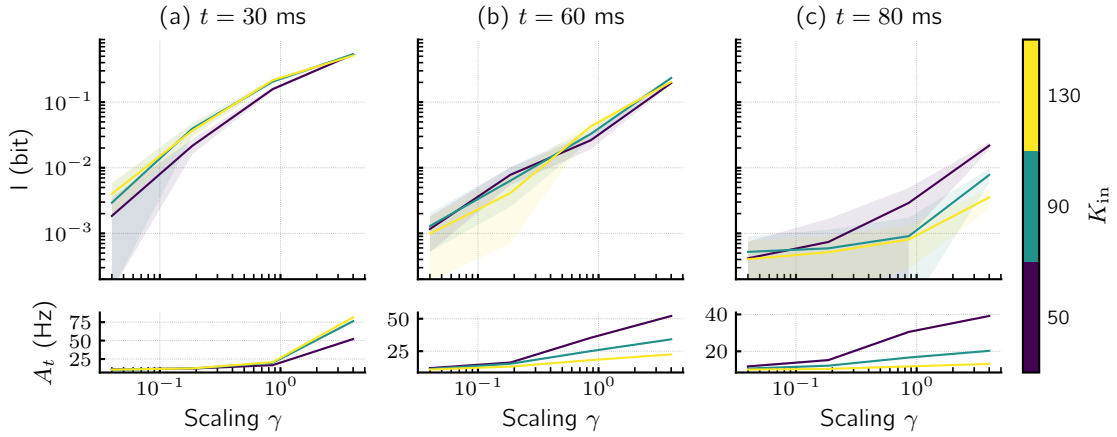


Figure 6.2: Dependence of the classification performance for Poisson patterns on the rate scaling factor. The relation is shown at three different points in times of the network activity. A larger scaling factor generally results in higher performance. (a) At the beginning of the patterns, input-driven networks demonstrate a small classification advantage. (b) Towards the end of the patterns, there is no significant difference in performance between networks of varying dynamics. (c) After the stimulation, networks with low indegree outperform input-driven ones. This is especially pronounced for large scaling factors that correspond to a strong perturbation of input activity.

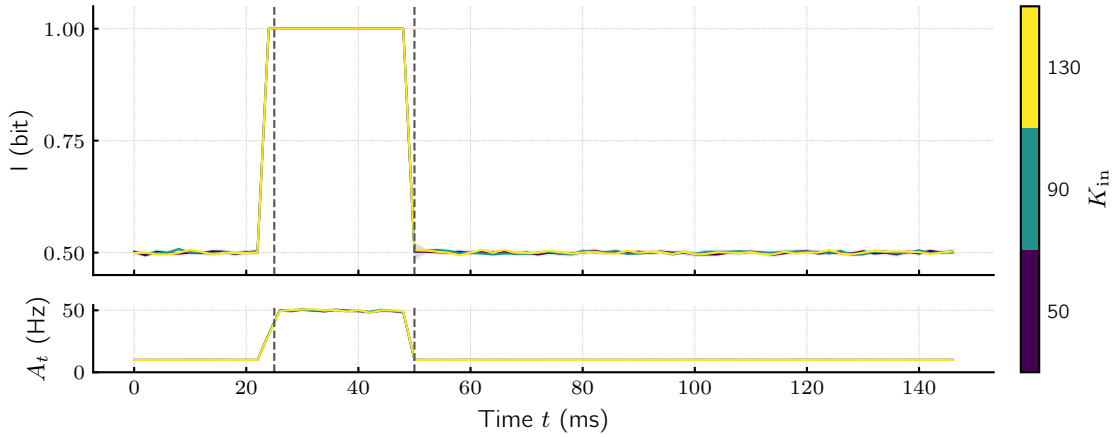


Figure 6.3: Classification accuracy on the input spikes of the Poisson patterns during persistent background noise stimulation. During pattern presentation the direct readout achieves perfect accuracy, while the predictions before and after lie at chance level (top). The activity of the input stimulus rises during the patterns from $10\,\text{Hz}$ to $50\,\text{Hz}$ indicating the scaling factor of $\gamma = 4$. Since the input stimulus to all networks is exactly the same except for noise realizations, no difference between colours is expected in this plot. (Binning is performed forward in time such that the last state before pattern onset already contains spikes from the patterns.)

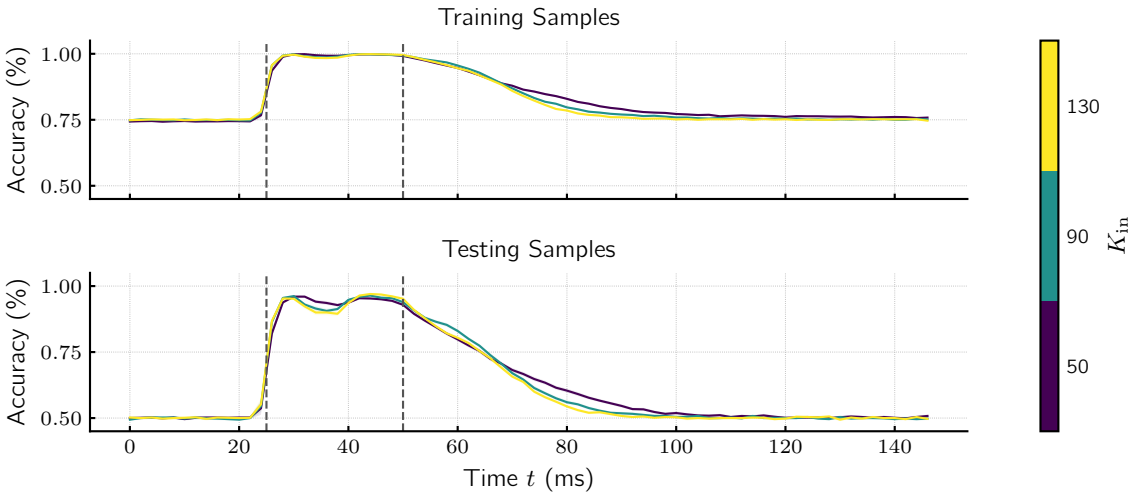


Figure 6.4: Training and testing accuracy on the Poisson patterns during persistent background noise stimulation. The readout can fit a significant portion of the training data even before and after the patterns, increasing the baseline accuracy. The classifier reaches a maximal testing accuracy of 97 % during pattern presentation, but maintains above chance level performance even after the patterns ended.

To infer an estimate of complexity of this simple task, the input spike activity is sent directly to the linear readout without processing them with the neuromorphic reservoir. In this case, the accuracy of the classifier (fraction of correctly labeled predictions) is show in Figure 6.3. During pattern presentation the classifier achieves perfect performance, but it drops immediately afterwards, as neither the input spikes nor the classifier exhibit memory on their own. The patterns themselves are fully linearly separable. The activity of the input plotted below clearly shows the increase during pattern presentation.

The low complexity of the task is also demonstrated by estimating the performance on training samples instead of the separate testing set (Fig. 6.4 top). Again the classification accuracy is shown in this situation. Quite extreme overfitting of the training data can be observed. The considerably large readout can easily separate the relatively few samples of the testing set. This demonstrates the relative simplicity of the two class task. Here the problem is addressed by working with the separate testing set, but for more complicated datasets overfitting is generally undesirable and can lead to bad performance.

For completeness, Figure 6.4 also shows the accuracy score on the testing data (bottom panel). At peak performance, the setup labels 97 % of the testing data correctly, compared to a chance level of 50 % for the given two class task. In contrast to the performance on the input patterns, the reservoir setup still achieves above chance level accuracy long after the patterns ended. The accuracy generally shows the same behaviour as the MI score and for comparisons in the following only the MI will be shown.

This analysis allows to conclude that the instantaneous performance during pattern

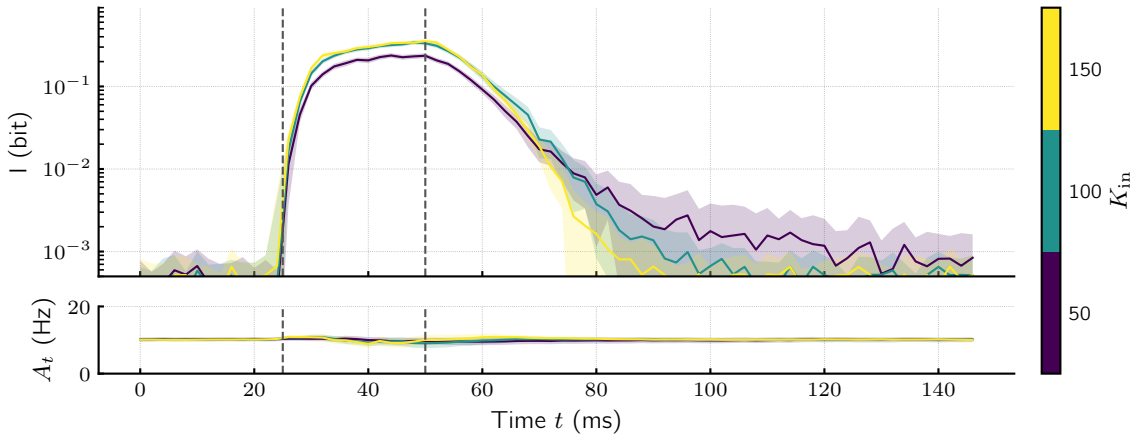


Figure 6.5: Classification performance on the Poisson patterns without background noise. Patterns in this plot have a signal-to-noise ratio of $s = 0.75$. (a) The MI between predictions and labels rises during pattern presentation and maintains fading memory afterwards. (b) Meanwhile, the network activity stays constant throughout the stimulus. Fading memory is an effect not only due to the sensitivity to rate perturbations.

presentation on the simple, linearly separable task is not impacted by the dynamical state of the network. This conforms with the experiments from Cramer et al. [2020] that found no benefit of critical dynamics for simple tasks. Nevertheless, the network closer to criticality is shown to exhibit larger fading memory for high input scaling which could be of interest in some situations.

In an alternative experiment, the spike sources are disabled during the introduction of the external stimulus. In this case the rate of the external spikes is chosen to be exactly ν_{adapt}, i.e., the network will receive the same amount of activity as during adaptation. Classification of just two Poisson patterns without any noise can lead to overfitting, since, even given the intrinsic noise levels on the hardware substrate, the network response will be very similar for identical input. Therefore, noise is now added directly to the external stimulus when it is created on the host computer. This is achieved by sampling patterns at a lower rate of $\nu_{\text{stim}} = s \cdot \nu_{\text{adapt}}$ and adding random Poisson spikes of frequency $\nu_{\text{noise}} = (1 - s) \cdot \nu_{\text{adapt}}$ to each sample individually, with the signal-to-noise ratio $0 \leq s \leq 1$.

Figure 6.5 depicts the classification performance and activity over time for a stimulus with $s = 0.75$. The MI again increases at the beginning of pattern presentation and then shows fading memory afterwards. In contrast to before, the average network activity remains nearly constant throughout the whole experiment. There is no indication of a rate perturbation. Still, the amount of fading memory is dependent on the dynamical state. The later spikes in the network output hold more information on the pattern at low indegree than at high indegree. This demonstrates that the characteristic autocorrelation in the more critical network is not only responsible for an increased sensitivity to rate

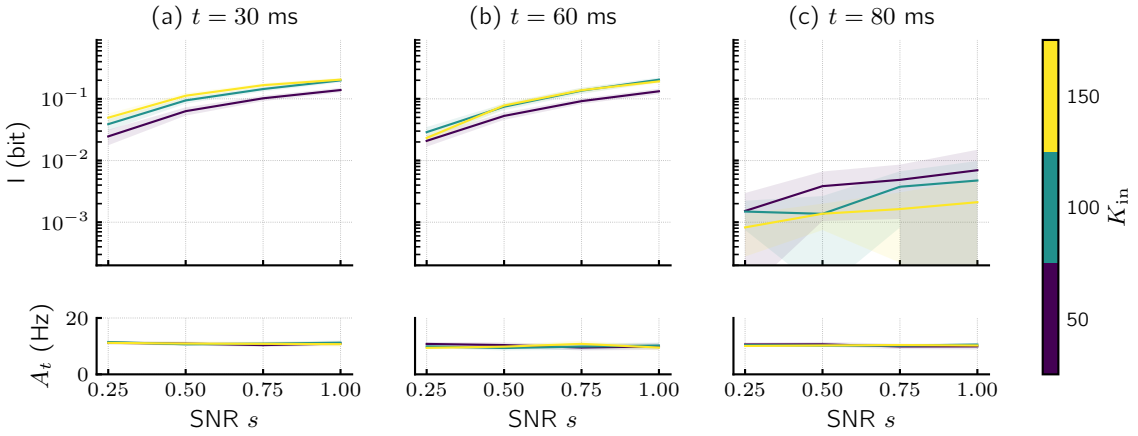


Figure 6.6: Dependence of the classification performance for Poisson patterns on the signal-to-noise ratio. The relation shown again for three different times of the network activity. A larger signal-to-noise ratio increases performance throughout the experiment. (a) At the beginning and (b) shortly after pattern presentation, the input-driven network shows a performance benefit. (c) Later on the network with low indegree demonstrates more fading memory.

changes. Another difference to the previous task with increased input rate is the behaviour during pattern presentation. While the input-driven network can immediately make use of information, delays in the network closest to criticality now degrade performance at that point in time.

The impact of the signal-to-noise ratio s is demonstrated in Figure 6.6. Again, the testing performance at three points in time is plotted against the ratio in the first row. As expected, the network activity remains constant for all values of s. The classification performance increases with growing signal-to-noise ratio. However, larger s also result in more overfitting on the training samples which can be undesired (data not shown). The dynamical state impacts performance in the same way for all signal-to-noise ratios, during the stimulus input-driven dynamics are favourable while fading memory of fluctuations can be leveraged at later times.

Observations made for these two stimulation paradigms help to design the experiment setup for the spatio-temporal patterns in the following sections. From a biological view, arguments for both forms of stimulation can be made. Baseline activity in the brain generally exhibit a background state of noisy spiking [Softky et al. 1993; Zenke et al. 2013]. Additional stimuli may be encoded on top of that background noise. But information from other brain areas can also be encoded in the seemingly random spikes.

Following the demonstrations so far, a practical consideration is made. The stimuli of the datasets analysed in the next sections are distributed less homogeneously in time than the Poisson patterns. If the background sources were instantly disconnected from the network at a specific time, and the activity of a presented pattern is at a low level

at that moment, the removal alone would lead to a rate perturbation. To avoid such phases of silence in the network input a certain amount of noise is desired in any case. Therefore, the background noise sources are kept connected to the network throughout all experiments. This also helps to forego overfitting of stimuli with low noise found in the Poisson patterns with $s = 1$.

6.2 Random Manifolds

In this thesis, the real interest for the reservoir setup is its application to tasks with different levels of complexity. Especially, the influence of network dynamics in these situations can be analysed. It was found that critical-like dynamics only benefit complex n-bit parity tasks with large history lengths n [Cramer et al. 2020]. The random manifolds (cf. Section 3.4.2) can be similarly controlled in their complexity. However, their linear separability highly depends on the embedding space dimension. For more dimensions the dataset is more easily separable by a classifier that has correspondingly larger readout. To limit the number of classes and samples needed to a reasonable number and still construct a task for many input channels that is not fully linearly separable, different samples from the same manifold are combined to create a single stimulus. Manifolds with 8 embedding dimensions are generated and the 256 input channels are constructed by stacking 32 independent samples. The task is then to discern between 8 possible classes build from different manifolds. Per class, 1000 such full size stimuli are drawn which are then split in a 75 % – 25 % training to testing ratio. The separability of the resulting dataset is tested with a linear support-vector machine (SVM) which is trained directly on the embedding space vectors before generating spikes.

By setting the stimulus timescale to $\tau_{\text{rand}} = 25\,\text{ms}$, the average rate of the spike patterns alone is $\nu_{\text{stim}} = 1/\tau_{\text{rand}} = 40\,\text{Hz}$. This activity is injected in addition to the background stimulus with $\nu_{\text{adapt}} = 10\,\text{Hz}$ leading to a scaling factor of $\gamma = 4$.

Task complexity is first controlled by the number of manifold dimensions D at a fixed value of high frequency content $\alpha = 3$. A simple task is created with $D = 1$. The SVM achieves an accuracy of $(99.9 \pm 0.1)\,\%$ (average and standard deviation over 10 datasets with newly created manifolds), indicating a highly separable, simple task. For a more complex setting $D = 4$ is used. At $(52 \pm 5)\,\%$ accuracy, SVM performance is much lower. The chance level of the 8 class task is at 12.5 %.

The MI score between reservoir predictions and labels of the testing set is plotted for both complexities in Figure 6.7. Patterns are presented in the interval from zero to 25 ms indicated by the dashed lines. As a first observation, overall performance on the simple task with $D = 1$ is indeed better, while a larger manifold dimensions degrades the classification performance. However, looking at the mean network activity over the

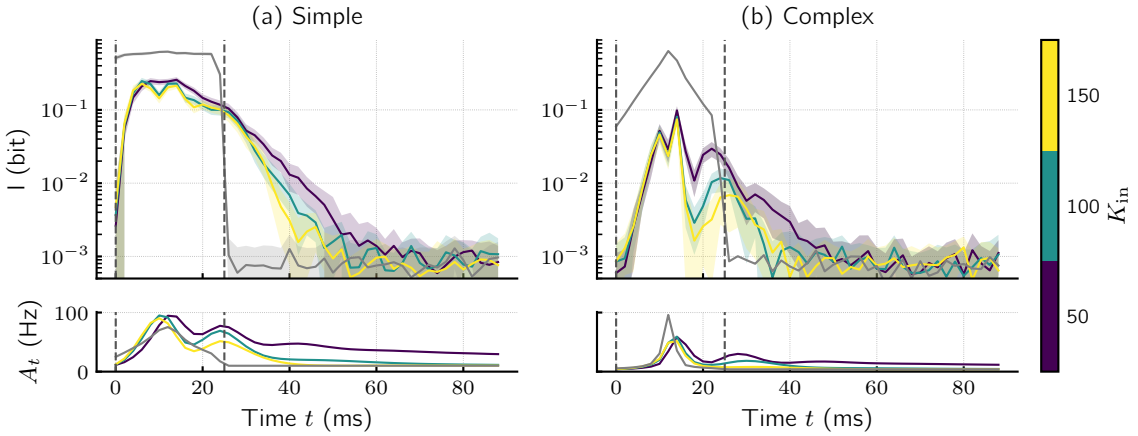


Figure 6.7: Classification performance on the random manifold task for different manifold dimensions. The MI score (top) and network activity (bottom) of reservoirs at different indegrees are plotted over time. Grey lines show the respective quantities for the input spike patterns. Stimuli are presented during the interval marked with the dashed lines. The manifold dimension does indeed impact overall classification performance, but also determines the clustering of spikes in the patterns. Networks at lower indegree respond stronger to patterns with a more pronounced peak in the activity that constitutes a stronger rate perturbation. Averages are taken over multiple instantiations of the dataset to reduce dependence on the specific manifolds chosen.

course of the stimulus, plotted in the bottom of the figure, it becomes visible that good performance is tightly coupled to the rate. The grey lines in the lower plot indicate the average input rates of the manifold spike patterns over time. In the simple task the rate is spread across the full duration of the patterns with a broad maximum towards the centre. In contrast, for the complex setting activity is much more clustered in a sharp peak at the centre of the interval. This is a direct consequence of the equations for the manifolds (Eq. 3.2) where the product is taken over more factors for larger D. Thus, the kind of complexity afforded by the manifold dimension is strongly related to the concentration of patterns in time. Different kinds of normalization are conceivable to circumvent this effect. However, simple approaches like cutting spikes off at selected stimulus borders would reduce the total rate for the spread out patterns at high D, while letting them saturate at the borders would shift the clustering to the edges. The effect of more advanced normalization techniques like sigmoidal or logarithmic scaling remains to be analysed in future experiments.

For the reservoir networks the clustering of spikes acts as a shorter but stronger perturbation. For both complexities the fading response of the network is clearly visible in the activity. Networks with lower indegree react stronger to these rate changes. Correspondingly, their performance gain is larger in the tasks with large D where activity is more clustered. Fading memory makes up some of the performance even during the stimulus, since most spikes of the patterns are already sent before the end. In contrast, for the

simple case fading memory is mostly only relevant after the end of the stimulus, before that the input activity stays at high levels.

An interesting observation to note is that in both cases the network activity initially starts to decay after reaching a first peak, but then a second smaller peak appears towards the end of the stimulus. This reaction to the sustained perturbation is already hinted at for the Poisson pattern in Figure 6.1. But for the manifolds, especially in the high-dimensional case, this can be seen to directly impact the classification. The input-driven networks show a second peak in performance at a corresponding time. This oscillating of activity during the stimulus may be related to the intrinsic timescales of the network. The second peak appears around 20 ms after the first one, which correspond to the values of the membrane time constants.

While these qualitative observations are in line with the expectations for tasks that depend on sustained memory, it is also important to take a look at absolute performance values. The average score of the linear readout trained directly on the input spikes created from the manifolds is plotted in the figure in grey. Throughout the stimulus the direct inputs are generally more informative and performance is better. It only drops to chance levels after the patterns end, as the inputs hold no memory. This indicates that the reservoir is not necessarily advantageous for tasks during the presentation of patterns.

These observations also demonstrate that the simple linear readout is very powerful on its own. In fact, the direct readout performs perfectly at a certain point in the input spike pattern even for the complex task where the SVM only achieved around 50 % accuracy directly on the embedding space vectors. The high performance on certain activity slices touches on a general problem with time-continuous tasks that is further discussed in Chapter 7. Specifically here it can be questioned in how far the dimensionality of manifolds really impacts the difficulty of classifying the resulting patterns. The spike generation approach used here produces stimuli that can clearly be linearly separated during intervals of clustered activity. Naturally, this could be addressed by increasing the number of classes in the dataset, which would however also impact experiment durations and increase the amount of data that needs to be handled.

As an alternative, task difficulty can be controlled by the scaling α of high frequency content in the manifold. The simple task is again generated with $\alpha = 3$ and $D = 1$, but for the complex case the dimension is kept fixed and more high frequency content is introduced by setting $\alpha = 0.7$. In this case, the resulting embedding space vectors can still be separated fairly well, with the linear SVM achieving (97.3 ± 1.7) % accuracy. However, the spikes generated from the vectors are much more difficult do discern. The performance is drawn in Figure 6.8. Even the direct readout that previously performed very good at least at a certain point in time, does not reach perfect performance for $\alpha = 0.7$. This

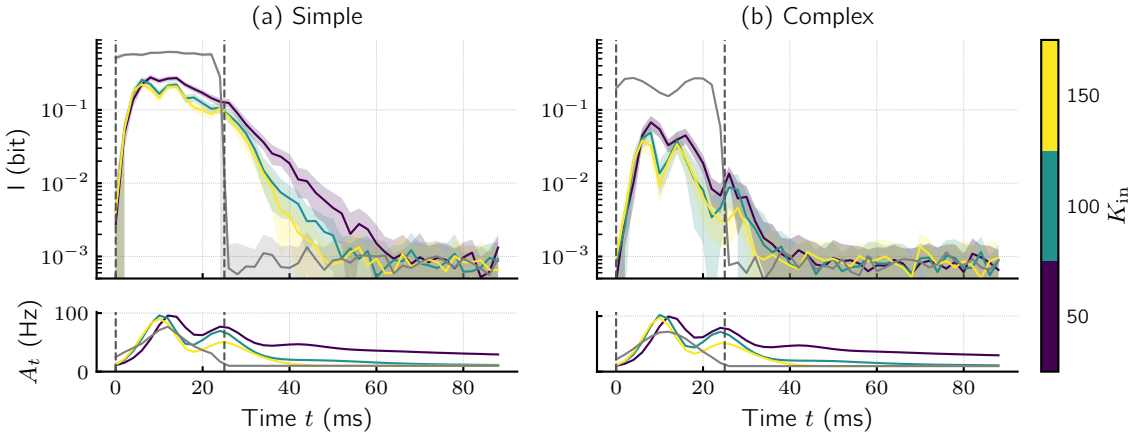


Figure 6.8: Classification performance on the random manifold task with different spatial high frequency content. The scaling of high frequency content by α can determine the complexity of the task. At low values, classification on both the input spikes and by the network is generally better. Increasing α degrades performance in all cases while not significantly changing the network activities. While fading memory is generally larger at lower indegree, it is similarly reduced by high complexity for all network dynamics.

difficulty manifests also in overall degraded performance by the reservoir. Especially, the duration of fading memory is also reduced.

The clustering effect in the activity is now avoided since both complexities are created from manifolds with the same dimension. Corresponding network activity also looks very similar for both cases. This also means that classification is not purely related to rate effects, since the networks perform much better on the simple task with large α. Interestingly, this is the case for all networks with different indegrees. While more critical network dynamics overall benefit performance in both cases, no particular impact for more complex tasks is observed.

6.3 Spiking Heidelberg Digits

As a final outlook on real world data, the performance of the reservoir setup on selected samples of the SHD dataset is considered. Digit patterns are prepared as described in Section 3.4.3 and homogeneously downsampled to the desired frequency of $\nu_{\text{stim}} = \gamma \cdot \nu_{\text{adapt}}$. The classification of two specific digit pairs is shown in Figures 6.9 and 6.10 for a scaling of $\gamma = 1$ and $\gamma = 3$ respectively. More combinations of digit pair are shown in the overview Figures 8.1 and 8.2 in the Appendix. Results are only averaged over 50 statistical network realizations to keep the amount of data needed to be saved and classified at cost. Also, because of the considerably longer duration of the stimuli, activity is binned at $\Delta t = 5\,\text{ms}$ instead of the previous value of the refractory period $\Delta t = \tau_{\text{ref}} = 2\,\text{ms}$.

For this dataset it is even more difficult to determine a measure of complexity. As

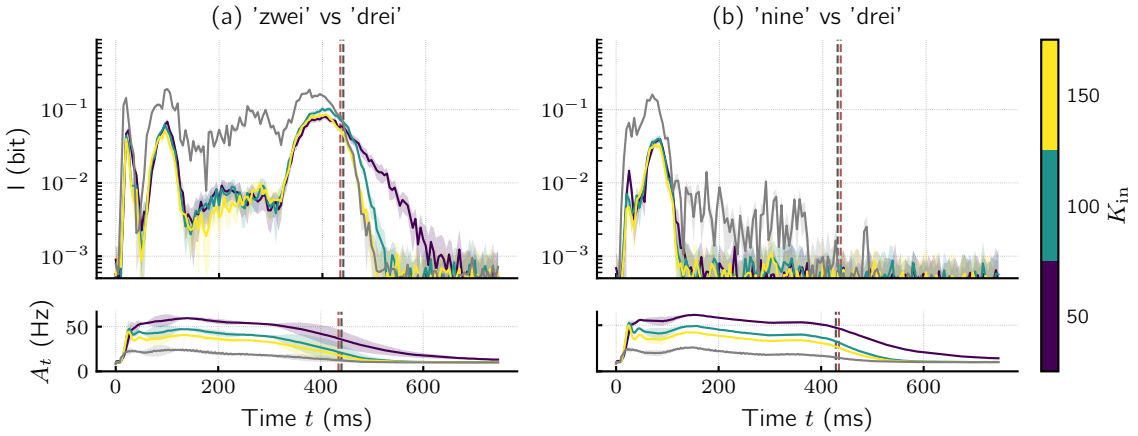


Figure 6.9: Classification performance on samples of the SHD dataset with low input scaling. The selected samples from the digits 'zwei' and 'drei' and those from 'nine' and 'drei' are classified against each other. The lower plots show the average population rates. Grey lines indicate the performance and activity for the input spike patterns. While the network generally degrades performance during the presentation of the digits, it can introduce fading memory for some cases.

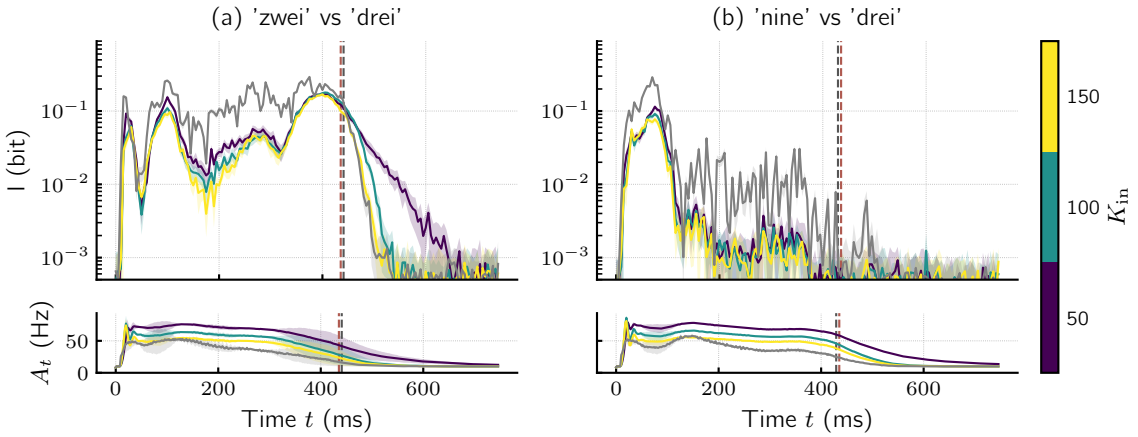


Figure 6.10: Classification performance on samples of the SHD dataset with high input scaling. Increasing the input stimulus rate significantly improves performance of the reservoir setup, pushing it closer to that achieved by the direct readout. This is especially pronounced for networks with lower indegree which starts to show an advantage beyond fading memory even during the digit duration.

the grey curves in the figures indicate, the tasks are generally better classified by the direct readout of the input spikes. This again demonstrates how powerful the ensembles of readouts at multiple points in time are. However, the reservoir can add fading memory to the output. The mean durations of the two digits being classified in each plot are drawn as dashed lines. For certain combinations of digits, like 'zwei' and 'drei,' high performance is maintained long after the end of the digits when classifying spikes from the reservoir compared to using those from the inputs directly. As would be expected, this is more pronounced for networks at lower indegrees. The same effect is also visible in the network activity which exhibits fading memory.

However, increased spike rates to not necessarily correlate with better performance of the classifiers. For one, during the duration of the digits, rates from the network are generally larger than those of the input spikes, but performance is still lower. Additionally, there are combinations of digits, for example 'nine' and 'drei' shown in Figure 6.9b, for which none of the reservoirs shows any memory at all even though fading activity is visible. In these cases the input spikes are already much less informative as indicated by the performance of the classifiers on the input, which is however also not directly represented in their rate.

Interestingly, the behaviour for specific digits does not seem to be related to any intuitive feeling about the complexity of the task. For instance one might expect that discerning 'zwei' from 'drei' might be rather difficult, especially during the ending of the digits, as they end with the same phone 'ei.' But in fact, this is one of the combinations for which the networks achieves the best performance. In this case, 'zwei' can generally be discerned rather good from the other digits (Fig. 8.1). These facts hint at the possibility that precise alignment of the samples in the dataset my be necessary to observe such effects. A quick visual glance at the digits recording patterns indeed shows that the exact positioning of phones scatters between samples.

The input scaling factor γ is overall strongly coupled to classification performance. For $\gamma = 3$ the network performance is increased closer to that achieved on the inputs. Especially, networks with lower indegree benefit from this which is again expected from the observations of the susceptibility and the previous tasks. Both, the fading memory and the performance during the digit is increased for lower indegree.

One point regarding the observations of all tasks that was not addressed so far is the possible impact of the rate saturation. In Section 5.4 it was observed that neuron firing rates in the hardware implementation start to saturate much earlier than expected. For tasks with large γ, input rates of up to 50 Hz can be reached, leading to strong network responses with activity temporarily reaching up to 100 Hz. These activities lie in the saturation regime of the neurons. While the additional non-linear behaviour resulting from this does not necessarily degrade performance, saturation is generally undesired since it limits the dynamical range of a system.

7 Conclusion and Outlook

The main results of the experiments presented in this thesis are the successful regulation of network activity with the homeostatic plasticity rule, and the control of the dynamical regime achieved in consequence. In Chapter 4 the probabilistic homeostasis was demonstrated to stabilise network activity in recurrent neural networks on a neuromorphic substrate despite the fixed-pattern deviations between neurons. The synaptic connections self-organise to an equilibrium distribution where inhibition balances excitation. Stability was achieved for networks of various couplings to the external input. Only towards very low coupling strengths the homeostasis fails to stabilise activity and a bistable state of near silence or extreme bursting was observed. Because the homeostasis only depends on local information of single neuron firing rates and it is implemented fully on the hardware chip, it is scalable to networks of larger size. As long as similar amount of presynaptic input can be provided to each neuron, the homeostasis should be able to stabilise network population activity. Particularly with neuromorphic hardware systems of larger scales becoming available, such mechanisms will be of interest.

As a second result, this thesis shows that the network indegree allows fine control of dynamics over a range of different states. Networks that are strongly coupled to the external driving force exhibit input-driven dynamics with short autocorrelation timescales. Conversely, in networks with low indegree the increased significance of recurrent connections was observed to generate fluctuating dynamics that exhibit autocorrelation on the scale of hundreds of milliseconds. These results can be compared to similar effects in biological substrates. It was proposed that autocorrelations and fluctuations hierarchically increase across the cerebral cortex from sensory to frontal areas as less and less activity is projected between areas [Murray et al. 2014]. There, timescales are suggested to reflect specializations for task-relevant computations.

The findings obtained with the BSS-2 hardware system were validated using reference simulations in Chapter 5. The comparison showed qualitatively the same kind of behaviour on hardware and in simulation. Simulations also demonstrated that parameter noise and integer weight arithmetics have no significant impact on the functionality of the homeostatic plasticity. Therefore, these properties of the hardware system have not been found to limit the possibilities for experiments. The results on noise fluctuations also indicate that precise calibration of model parameters may not be needed at the level used for

experiments in this thesis. Observations from initial experiments without explicit neuron calibration seem to confirm this. This fact could impact experiment workflows considerably, since substrate specific calibration can be time-consuming. This is especially true on larger systems.

While the qualitative behaviour was the same between hardware and simulation, a mismatch between the specific ranges of input coupling was found. This could be traced back to a saturation effect in the synapses affecting the synaptic time constants. Apart from shifting the operating point needed to achieve a specific dynamics state, it does however not degrade the functioning of the homeostasis. It is expected that a fix in the underlying synapse circuits on future version of the HICANN-X chip will at least partially remediate the observed shift by eliminating the observed saturation.

Time and Energy Considerations

One striking advantage of working with the current BSS-2 system is the ability for rapid iteration afforded by the high speedup of emulation. A precise evaluation of time and energy consumption would be of little interest for the given experiments since it highly depends on the specific computer hardware chosen. Simulations in this thesis were not optimised for performance beyond the standard behaviour of the `brian2` simulator. Furthermore, the BrainScaleS-2 system is a research platform still of limited size and availability, and is built on a completely different manufacturing process than modern processors. Thus, a direct comparison would be unfair to conventional processors in terms of their availability but unfair to BSS-2 in terms of efficiency. Nonetheless, the general behaviour of the systems as it was experienced during experimentation shall be discussed.

Owing to the high speedup of network emulation on BSS-2 and the fully on-chip implementation of the plasticity rule, a single experiment consisting of a network adaptation phase and subsequent static run is executed in a few seconds of wall-clock time. This includes experiment initialization overhead which does not need to be repeated when conducting multiple experiments in succession. As such, a parameter sweep with 1000 successive network realizations could be completed in under half an hour. In contrast, a `brian2` simulation of an individual network configuration with the same experiment setup has a runtime in the order of ten hours when executed on a single thread of an Intel Xeon 6130 CPU. The specific duration can vary by several hours depending on the regime of the network dynamics. These values were obtained with the chosen integration timestep of $\Delta t = 10\,\mu s$. On BSS-2, digital time events are handled with a precision in the order of nanoseconds, which corresponds to a biological equivalent time of around $1\,\mu s$. Thus, spike times on the neuromorphic hardware can be up to an order of magnitude more precise compared to the simulations. Seen the other way around, simulations with similar precision would need multiple days instead of seconds for a single experiment, since the timestep

scales the experiment duration fairly linearly. Though it is arguable if such precise spike times are really necessary for all experiments [Heidarpur et al. 2020]. In any case, there is a huge difference in the way experiments can be conducted with the hardware compared to simulation. Instead of being able to nearly instantly observe the impact of changes made to the network, the experimenter has to wait days before receiving feedback. Of cause, the wide availability of conventional compute power allows to easily parallelise multiple simulations. Still, the long time for completion of a single homeostatic adaptation can not be bypassed by this. Furthermore, multiplexing of experiments is in principle also possible with the hardware system, especially as more setups become available. Especially the experiments presented here should be very portable since they are not significantly impacted by fixed-pattern deviations between hardware realizations.

An obvious impact of the differences in single experiment run time is the energy consumption. While again a precise comparison is not sensible for this thesis, especially because simulations could be ported to GPUs which show considerable advantages over CPU implementations [e.g. Stimberg et al. 2020], rough estimates are presented. The processors used for simulation have a TDP of 125 W. A single threaded load can be roughly estimated to consume power in the order of tens of watts. With an approximate run time of 15 h for a single experiment this corresponds to an energy in the order of 1 MJ. On HICANN-X, the homeostatic adaptation experiment is in principle executed fully on-chip, with the background sources providing the driving force and the PPUs updating weights. This means that the host computer and FPGA are not required during experiments. Preliminary measurements indicate a power consumption of only 100 mW by HICANN-X alone. However, this does not include the full level of input/output operations and experiment control needed to record spike events for evaluation. In principle, these are not required during the network adaptation phase. Still, a more conservative estimate given by the power consumption of a *cube setup* in the order of watts. Using the above time duration for hardware experiment sweeps gives single experiment energy costs in the order of 10 J, which still puts five orders of magnitude between hardware and simulation. Especially when conducting vast amounts of experiments, but also for future systems of larger size, this difference can have an enormous impact.

Outlook on Reservoir Computing and Tasks

For a first test of the applicability of the achieved control of network dynamics, the impact of critical-like behaviour in reservoir computing networks was analysed. Finding suitable tasks with varying levels of complexity proved quite difficult. While previous work suggests that the fluctuating regime in dynamical systems is generally favourable for computation, for the tasks considered here this was found to be mostly expressed in an increased amount of fading memory in the systems. This behaviour directly corresponds to the larger auto-

correlations and increased susceptibility to rate perturbations measured for these states. In fact, classification performance of the linear readout was found to be tightly related to the rate perturbation introduced by stimuli. The time development of the network rate alone was observed to strongly correlate with the fading memory in systems of low input coupling. Still, an increased amount of memory in the fluctuating regime was also observed for homogeneous inputs that did not involve a substantial rate perturbation.

However, memory alone is not necessarily a condition for good computation. On the contrary, maximal classification performance during stimulus presentation was generally degraded by using the reservoir, compared to the best performance of a readout directly on the input spikes. For the random manifolds the direct classification of input spikes even outperformed the SVMs operating on the embedding vectors before spike generation. This demonstrates the considerable power of the linear readout alone. However, the input spikes trivially contain no memory after the end of a stimulus.

At this point, an important question for time-continuous classification tasks becomes clear: At what point in time is the prediction of the classifier wanted? The ensembles of readouts trained at successive times of the network state used in this thesis were instructive to gain insight into the working of the network, but do not present a practical system for computation in any real application. Just in the opposite, for a keyword spotting task a primary interest is the detection of the time at which information is present at all. For true time-continuous operations on longer stimuli both is needed, to realise when information of interest is available and then integrate that over time.

One way to address the readout issue could be the use of a single classifier that is trained on the activity states from all points in time. This could also significantly reduce overfitting, as such a classifier has far fewer parameters compared to the full ensemble. The settings could further be explored in future experiments with stimuli constructed from sequences of patterns by tasking the network to predict the labels of a full sequences.

Another approach to the reservoir setup that has not been explored thus far is the adaptation of the network weights during stimulation with structured inputs similar to those of the tasks, instead of using homogeneous Poisson noise. This way the network may incorporate certain features inherent to the stimuli and develop preferred paths of connectivity between neurons. This would be different to the current setting, where quite on the contrary the homeostasis homogeneously affects synapses and equalises the activity response. Going beyond reservoir computing, the plasticity rule could even be combined with supervised training methods for SNNs like backpropagation with surrogate gradients. These could benefit from the overall network stability achieved by the homeostasis. This may also be applicable to the initialisation problem of weights in deep spiking networks, where the homeostasis could help reduce training times by establishing an initial state of moderate activity.

The influence of the saturation effect observed in Figure 5.6, on the working of the reservoir setup remains to be evaluated in future version of HICANN-X that do not show this behaviour of synapses. Performing these experiments in simulation would be computationally much more challenging for the reasons discussed above. Even working with hardware, the size of the datasets had a noticeable impact mostly caused by the input/output operations of spike transfer to and from the chip. The latter of these operations may be addressed by implementing the readout mechanisms on HICANN-X directly, building on Cramer et al. [2021] for a spiking or Stradmann et al. [2021] for a non-spiking approach.

In summary, the relevance of this work for future use with tasks lies in the stability and control of network dynamics achieved by the homeostatic plasticity. These are a necessary requirement in any large recurrent network. Future experiments will show if the results can be leveraged in reservoir computing or similar approaches beyond the simple classification analysis presented here.

8 Appendix

Parameter	Symbol	Value
Leak potential	u_{leak}	$(455 \pm 11)\,\text{mV}$
Threshold potential	u_{th}	$(741 \pm 6)\,\text{mV}$
Reset potential	u_{reset}	$(325 \pm 5)\,\text{mV}$
Refractory period	τ_{ref}	$2\,\text{ms}$
Membrane time constant	τ_{mem}	$(20.2 \pm 0.6)\,\text{ms}$
Synaptic time constant	$\tau_{\text{syn}}^{\text{exc}}$	$(10.1 \pm 0.3)\,\text{ms}$
	$\tau_{\text{syn}}^{\text{inh}}$	$(10.1 \pm 0.3)\,\text{ms}$
Synaptic delay	d_{syn}	$1\,\text{ms}$
Synaptic amplitude	$I_0^{\text{exc}}/g_{\text{leak}}$	$(3.36 \pm 0.47)\,\text{mV}$
	$I_0^{\text{inh}}/g_{\text{leak}}$	$(3.74 \pm 0.52)\,\text{mV}$
Synaptic amplitude offset	$\delta I_0^{\text{exc}}/g_{\text{leak}}$	$(-1.7 \pm 2.3)\,\text{mV}$
	$\delta I_0^{\text{inh}}/g_{\text{leak}}$	$(2.4 \pm 2.5)\,\text{mV}$
Number of neurons	N	512
Number of input channels	N_{in}	256
Number of inhibitory neurons	N^{inh}	102
Number of inhibitory input channels	$N_{\text{in}}^{\text{inh}}$	51
Number of recurrent presynaptic neurons	K_{rec}	38
Initial weight	w_{ij}^{init}	$0\,\text{bit}$
Homeostatic target frequency	ν_{target}	$10\,\text{Hz}$
Homeostatic measurement duration	T_{meas}	$1\,\text{s}$
Homeostatic equilibration time	T_{eq}	$1\,\text{s}$
Learning rate	η	$0.5\,\text{s}$
Homeostatic update probability	p_{update}	$2.5\,\%$
Input frequency	ν	$10\,\text{Hz}$
Static experiment duration	T	$80\,\text{s}$

Table 8.1: List of default model parameters. Values are given in the biological equivalent time domain. Errors indicate the standard deviation of the measurements in Section 4.1.

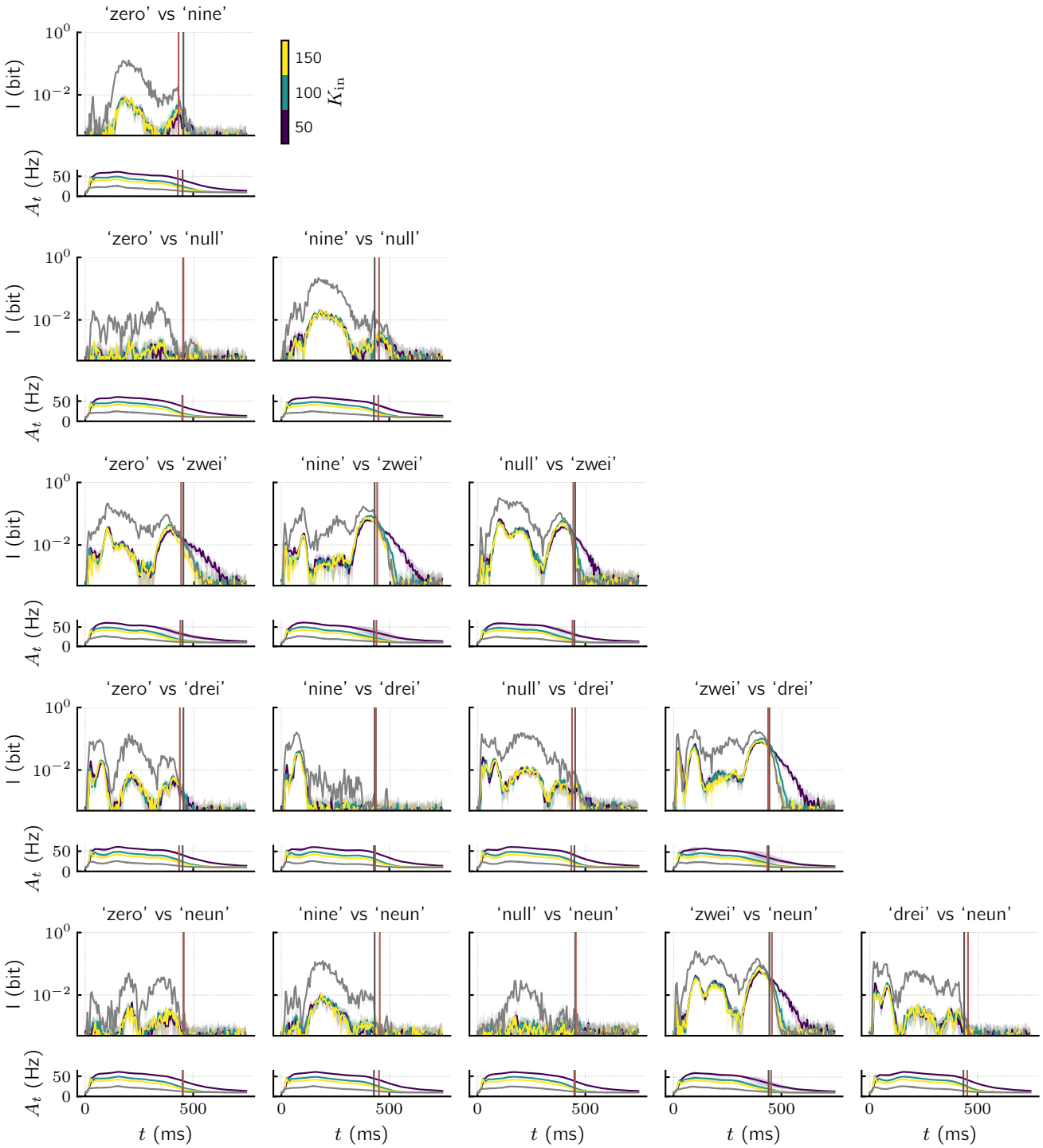


Figure 8.1: Overview of the classification performance on multiple samples of the SHD dataset with low input scaling. One-to-one classification performance and average population rates are shown for reservoirs adapted at different indegree. Grey lines indicate the same quantities for the input spike patterns directly and the mean digit duration is shown. While the reservoir setup generally degrades performance during the presentation of the digits, it can introduce fading memory in some cases. Specific digits like 'zwei' can usually be discerned quite easily, while cases that involve 'neun' are overall more difficult.

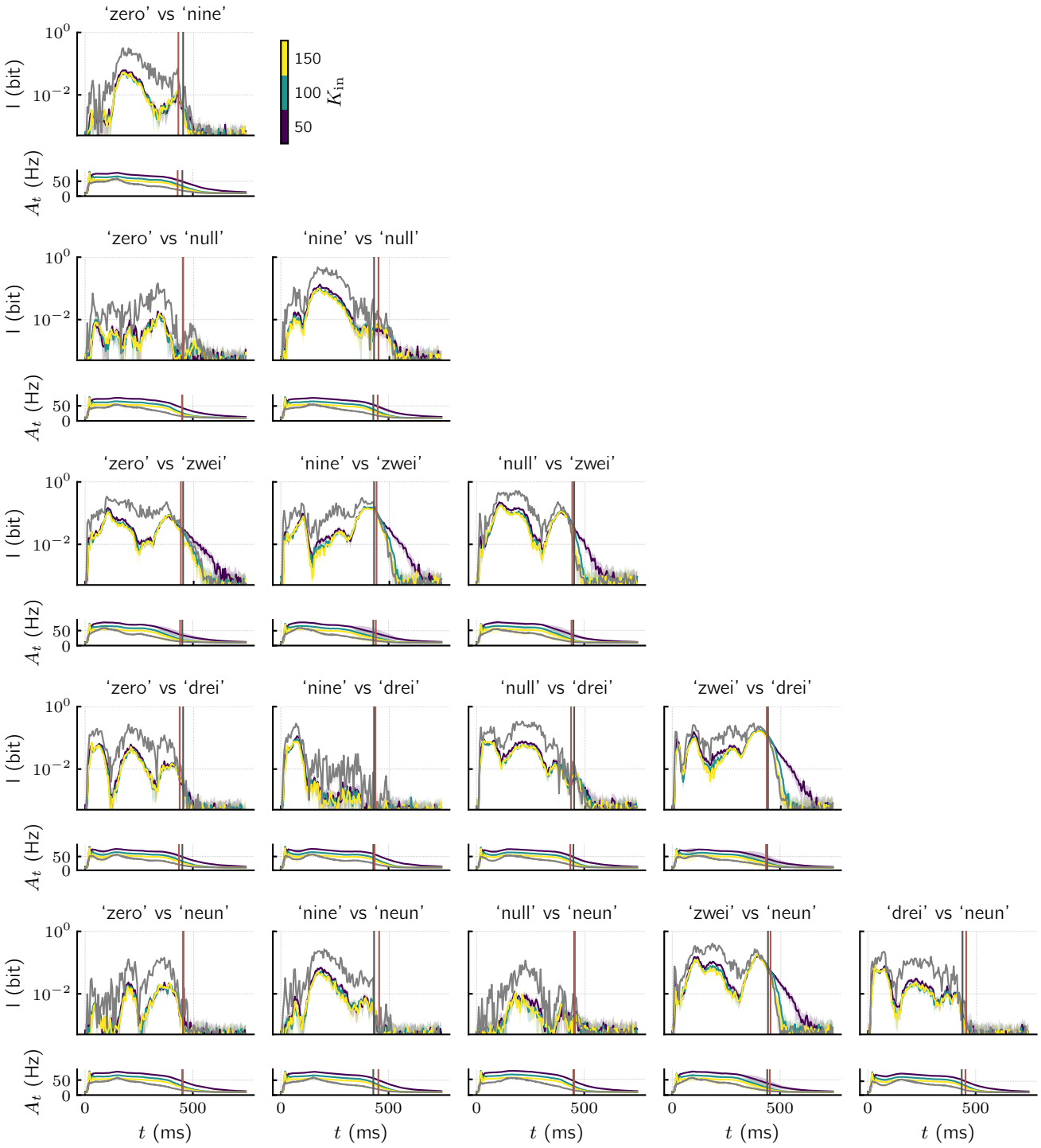


Figure 8.2: Overview of the classification performance on multiple samples of the SHD dataset with high input scaling. Increasing the input rate of the digits generally leads to better performance of the reservoir, closer to that observed directly on the input. This is especially pronounced for networks with low indegree, which start to show an advantage over input-driven ones not only in fading memory but also at points during the presentation of digits.

9 References

Barnett, L., Lizier, J. T., Harré, M., Seth, A. K., and Bossomaier, T. (2013). „Information Flow in a Kinetic Ising Model Peaks in the Disordered Phase“. *Phys. Rev. Lett.* 111 (17), p. 177203. DOI: 10.1103/PhysRevLett.111.177203.

Bertschinger, N. and Natschläger, T. (2004). „Real-time computation at the edge of chaos in recurrent neural networks“. *Neural Computation* 16 (7), pp. 1413–1436. DOI: 10.1162/089976604323057443.

Boedecker, J., Obst, O., Lizier, J. T., Mayer, N. M., and Asada, M. (2012). „Information processing in echo state networks at the edge of chaos“. *Theory in Biosciences* 131, pp. 205–213. DOI: 10.1007/s12064-011-0146-8.

Brette, R. and Gerstner, W. (2005). „Adaptive exponential integrate-and-fire model as an effective description of neuronal activity“. *Journal of neurophysiology* 94 (5), pp. 3637–42. DOI: 10.1152/jn.00686.2005.

Brüderle, D. et al. (2011). „A comprehensive workflow for general-purpose neural modeling with highly configurable neuromorphic hardware systems“. *Biological Cybernetics* 104, pp. 263–296. DOI: 10.1007/s00422-011-0435-9.

Cramer, B., Stöckel, D., Kreft, M., Wibral, M., Schemmel, J., Meier, K., and Priesemann, V. (2020). „Control of criticality and computation in spiking neuromorphic networks with plasticity“. *Nature Communications* 11 (1), p. 2853. DOI: 10.1038/s41467-020-16548-3.

Cramer, B., Billaudelle, S., Kanya, S., Leibfried, A., Grübl, A., Karasenko, V., Pehle, C., Schreiber, K., Stradmann, Y., Weis, J., Schemmel, J., and Zenke, F. (2021). *Surrogate gradients for analog neuromorphic computing*. arXiv: 2006.07239.

Cramer, B., Stradmann, Y., Schemmel, J., and Zenke, F. (2019). *Heidelberg Spiking Datasets*. DOI: 10.21227/51gn-m114.

Cramer, B., Zierenberg, J., Kreft, M., Billaudelle, S., Schemmel, J., and Priesemann, V. (in prep). „Homeostatically induced autocorrelations in networks of excitatory and inhibitory neurons inside the excitation-dominated regime“.

Dale, H. (1934). „Pharmacology and Nerve Endings“. *British medical journal* 2 (3859), pp. 1161–1163. DOI: 10.1136/bmj.2.3859.1161.

Friedmann, S., Schemmel, J., Grübl, A., Hartel, A., Hock, M., and Meier, K. (2017). „Demonstrating Hybrid Learning in a Flexible Neuromorphic Hardware System“. *IEEE Transactions on Biomedical Circuits and Systems* 11 (1), pp. 128–142. DOI: 10.1109/TBCAS.2016.2579164.

Friedmann and Pehle (2018). *Nux User Guide*. Tech. rep. available at https://github.com/electronicvisions/nux. Electronic Vision(s).

Gerstner, W., Kistler, W. M., Naud, R., and Paninski, L. (2014). *Neuronal Dynamics: From Single Neurons to Networks and Models of Cognition*. Cambridge University Press. ISBN: 978-1-107-63519-7.

Grübl, A., Billaudelle, S., Cramer, B., Karasenko, V., and Schemmel, J. (2020). „Verification and Design Methods for the BrainScaleS Neuromorphic Hardware System“. *Journal of Signal Processing Systems*. DOI: 10.1007/s11265-020-01558-7.

Harris, C. R. et al. (2020). „Array programming with NumPy“. *Nature* 585 (7825), pp. 357–362. DOI: 10.1038/s41586-020-2649-2.

Harris, T. E. (1963). *The Theory of Branching Processes*. Berlin: Springer. ISBN: 978-3-642-51868-3.

Heidarpur, M., Ahmadi, A., and Ahmadi, M. (2020). „Time Step Impact on Performance and Accuracy of Izhikevich Neuron: Software Simulation and Hardware Implementation“. *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5. DOI: 10.1109/ISCAS45731.2020.9180632.

Hock, M., Hartel, A., Schemmel, J., and Meier, K. (2013). „An analog dynamic memory array for neuromorphic hardware“. *2013 European Conference on Circuit Theory and Design (ECCTD)*, pp. 1–4. DOI: 10.1109/ECCTD.2013.6662229.

Jaeger, H. (2001). „The “echo state” approach to analysing and training recurrent neural networks-with an erratum note“. *German National Research Center for Information Technology GMD Technical Report* 148 (34), p. 13.

Langton, C. G. (1990). „Computation at the edge of chaos: Phase transitions and emergent computation“. *Physica D: Nonlinear Phenomena* 42 (1), pp. 12–37. DOI: 10.1016/0167-2789(90)90064-V.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). „Deep learning“. *Nature* 521, pp. 436–444. DOI: 10.1038/nature14539.

Maass, W., Natschläger, T., and Markram, H. (2002). „Real-time computing without stable states: a new framework for neural computation based on perturbations“. *Neural Computation* 14 (11), pp. 2531–60. DOI: 10.1162/089976602760407955.

Maass, W. (1997). „Networks of spiking neurons: The third generation of neural network models“. *Neural Networks* 10 (9), pp. 1659–1671. DOI: 10.1016/S0893-6080(97)00011-7.

McCulloch, W. and Pitts, W. (1943). „A logical calculus of the ideas immanent in nervous activity“. *Bull. Math. Biophysics* 5, pp. 115–133. DOI: 10.1007/BF02478259.

Mead, C. (1990). „Neuromorphic electronic systems“. *Proceedings of the IEEE* 78 (10), pp. 1629–1636. DOI: 10.1109/5.58356.

Müller, E., Mauch, C., Spilger, P., Breitwieser, O. J., Klähn, J., Stöckel, D., Wunderlich, T., and Schemmel, J. (2020). *Extending BrainScaleS OS for BrainScaleS-2*. arXiv: 2003.13750.

Muñoz, M. A. (2018). „Colloquium: Criticality and dynamical scaling in living systems“. *Rev. Mod. Phys.* 90 (3), p. 031001. DOI: 10.1103/RevModPhys.90.031001. URL: https://link.aps.org/doi/10.1103/RevModPhys.90.031001.

Murray, J. D., Bernacchia, A., Freedman, D. J., Romo, R., Wallis, J. D., Cai, X., Padoa-Schioppa, C., Pasternak, T., Seo, H., Lee, D., and Wang, X.-J. (2014). „A hierarchy of intrinsic timescales across primate cortex“. *Nature Neuroscience* 17 (12), pp. 1661–1663. DOI: 10.1038/nn.3862.

Neftci, E., Chicca, E., Indiveri, G., and Douglas, R. (2011). „A Systematic Method for Configuring VLSI Networks of Spiking Neurons“. *Neural Computation* 23 (10), pp. 2457–2497. DOI: 10.1162/NECO_a_00182.

Pedregosa, F. et al. (2011). „Scikit-learn: Machine Learning in Python“. *Journal of Machine Learning Research* 12, pp. 2825–2830.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). „Learning representations by back-propagating errors“. *Nature* 323 (6088), pp. 533–536. DOI: 10.1038/323533a0.

Schemmel, J., Brüderle, D., Grübl, A., Hock, M., Meier, K., and Millner, S. (2010). „A wafer-scale neuromorphic hardware system for large-scale neural modeling“. *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, pp. 1947–1950. DOI: 10.1109/ISCAS.2010.5536970.

Schemmel, J., Billaudelle, S., Dauer, P., and Weis, J. (2020). *Accelerated Analog Neuromorphic Computing*. arXiv: 2003.11996.

Schmidhuber, J. (2015). „Deep learning in neural networks: An overview“. *Neural Networks* 61, pp. 85–117. DOI: 10.1016/j.neunet.2014.09.003.

Softky, W. and Koch, C. (1993). „The highly irregular firing of cortical cells is inconsistent with temporal integration of random EPSPs“. *Journal of Neuroscience* 13 (1), pp. 334–350. DOI: 10.1523/JNEUROSCI.13-01-00334.1993.

Spitzner, F. P., Dehning, J., Wilting, J., Hagemann, A., Neto, J. P., Zierenberg, J., and Priesemann, V. (2020). *MR. Estimator, a toolbox to determine intrinsic timescales from subsampled spiking activity*. arXiv: 2007.03367.

Stimberg, M., Brette, R., and Goodman, D. F. (2019). „Brian 2, an Intuitive and Efficient Neural Simulator“. *eLife* 8. DOI: 10.7554/eLife.47314.

Stimberg, M., Goodman, D. F. M., and Nowotny, T. (2020). „Brian2GeNN: accelerating spiking neural network simulations with graphics hardware“. *Scientific Reports* 10 (1), p. 410. DOI: 10.1038/s41598-019-54957-7.

Stradmann, Y., Billaudelle, S., Breitwieser, O., Ebert, F. L., Emmel, A., Husmann, D., Ilmberger, J., Müller, E., Spilger, P., Weis, J., and Schemmel, J. (2021). *Demonstrating Analog Inference on the BrainScaleS-2 Mobile System*. arXiv: 2103.15960.

Tanaka, G., Yamane, T., Héroux, J. B., Nakane, R., Kanazawa, N., Takeda, S., Numata, H., Nakano, D., and Hirose, A. (2019). „Recent advances in physical reservoir computing: A review“. *Neural Networks* 115, pp. 100–123. DOI: 10.1016/j.neunet.2019.03.005.

Tkačik, G., Mora, T., Marre, O., Amodei, D., Palmer, S. E., Berry, M. J., and Bialek, W. (2015). „Thermodynamics and signatures of criticality in a network of neurons“. *Proceedings of the National Academy of Sciences* 112 (37), pp. 11508–11513. DOI: 10.1073/pnas.1514188112.

Virtanen, P. et al. (2020). „SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python“. *Nature Methods* 17, pp. 261–272. DOI: 10.1038/s41592-019-0686-2.

von Neumann, J. (1945). *First Draft of a Report on the EDVAC*. http://web.mit.edu/STS.035/www/PDFs/edvac.pdf. (PDF), retrieved January 8, 2021.

Weis, J. (2020). „Inference with Artificial Neural Networks on Neuromorphic Hardware“. Master's thesis. Universität Heidelberg.

Wilting, J., Dehning, J., Pinheiro Neto, J., Rudelt, L., Wibral, M., Zierenberg, J., and Priesemann, V. (2018a). „Operating in a Reverberating Regime Enables Rapid Tuning of Network States to Task Requirements“. *Frontiers in Systems Neuroscience* 12, p. 55. DOI: 10.3389/fnsys.2018.00055.

Wilting, J. and Priesemann, V. (2018b). „Inferring collective dynamical states from widely unobserved systems“. *Nature Communications* 9 (1), p. 2325. DOI: 10.1038/s41467-018-04725-4.

Zenke, F., Bohté, S. M., Clopath, C., Comşa, I. M., Göltz, J., Maass, W., Masquelier, T., Naud, R., Neftci, E. O., Petrovici, M. A., Scherr, F., and Goodman, D. F. (2021a). „Visualizing a joint future of neuroscience and neuromorphic engineering“. *Neuron* 109 (4), pp. 571–575. DOI: 10.1016/j.neuron.2021.01.009.

Zenke, F., Hennequin, G., and Gerstner, W. (2013). „Synaptic Plasticity in Neural Networks Needs Homeostasis with a Fast Rate Detector“. *PLOS Computational Biology* 9 (11), pp. 1–14. DOI: 10.1371/journal.pcbi.1003330.

Zenke, F. and Vogels, T. P. (2021b). „The Remarkable Robustness of Surrogate Gradient Learning for Instilling Complex Function in Spiking Neural Networks“. *Neural Computation*, pp. 1–27. DOI: 10.1162/neco_a_01367.

Zierenberg, J., Wilting, J., and Priesemann, V. (2018). „Homeostatic Plasticity and External Input Shape Neural Network Dynamics“. *Phys. Rev. X* 8 (3), pp. 031018–15. DOI: 10.1103/PhysRevX.8.031018.

Abbreviations

AC	autocorrelation
ADC	analog-to-digital converter
AdEx	adaptive exponential integrate-and-fire
ANN	artificial neural network
ASIC	application-specific integrated circuit
BSS-2	BrainScaleS-2
CADC	columnar ADC
CapMem	capacitive memory
CPU	central processing unit
EPSP	excitatory postsynaptic potential
FPGA	field-programmable gate array
GPU	graphics processing unit
HICANN-X	High Input Count Analog Neural Network
LIF	leaky integrate-and-fire
MADC	membrane ADC
MI	mutual information
PCB	printed circuit board
PPU	plasticity processing unit
PSP	postsynaptic potential
RNG	random number generator
RNN	recurrent neural network
SHD	Spiking Heidelberg Digits
SIMD	single instruction, multiple data
SNN	spiking neural network
SVM	support-vector machine
TDP	thermal design power

List of Figures

Acknowledgments

I thank Dr. Johannes Schemmel for supervising my thesis and all his work in developing the hardware that enabled my experiments.
I thank Professor Schäfer for taking the time from his usual areas of interest to co-examine my thesis.
Special thanks go to Benjamin Cramer for patiently and carefully overseeing my experiments and always helping to work out any problems.
Thanks go to Johannes Zierenberg for the joint work on simulations, fruitful discussions, and help with theoretical questions.

I thank Benjamin, Jakob, and Johannes for proofreading the thesis and finding mistakes from syntax to semantics.
Thanks to Oliver, Christian, and Eric for their help with any technical problems during my work.
I thank all past and present members of the Electronic Vision(s) group for designing, developing, and operating the systems that enabled my work.
Finally, I thank my family and friends for their support and inspiration throughout my studies.

The work carried out in this Master's thesis used systems, which received funding from the European Union's Horizon 2020 Framework Programme for Research and Innovation under the Specific Grant Agreements Nos. 720270, 785907, and 945539 (Human Brain Project, HBP)

Statement of Authorship (Erklärung)

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

Heidelberg, 31.03.2021

Ort, Datum

Markus Kreft