Dissertation

submitted to the

Combined Faculty of Natural Sciences and Mathematics
of Heidelberg University, Germany

for the degree of

Doctor of Natural Sciences

Put forward by

Benjamin Cramer

born in Karlsruhe

Oral examination: December 7, 2021

Optimizing Spiking Neuromorphic Networks for Information Processing

Referees: **Dr. Viola Priesemann (University of Göttingen)**
Professor Dr. Daniel Durstewitz (Heidelberg University)

Optimizing Spiking Neuromorphic Networks for Information Processing

The human brain efficiently processes information by analog integration of inputs and digital, binary communication. This fundamental design is captured in spiking neural network models that aim to harness the brain's processing power and energy efficiency. Within this thesis, we contribute to the manifold optimization of these models for information processing. To that end, we first consider strategies for the quantification of the ability to process information. Second, we optimize our network implementations for efficiency by exploiting the analog emulation of neuro-synaptic dynamics on neuromorphic hardware, which aims to tie on the energy efficiency of its biological archetype by mimicking key architectural principles. The actual optimization for information processing is targeted in a third step by exploiting two orthogonal approaches: Specifically, we utilize gradient-based methods in supervised learning scenarios and, moreover, we deliberately exploit collective dynamics for information processing that emerged under local unsupervised plasticity. In the last step, we consider overarched optimization strategies to cope with constraints imposed by the neuromorphic implementation. In particular, we alleviate the ubiquitous issue of limited synaptic resources via local structural reconfigurations. By considering all of the termed stages, we highlight the potential processing capabilities of spiking neural networks implemented on analog neuromorphic hardware.

Optimierung von Spikenden Neuromorphen Netzen für die Informationsverarbeitung

Das menschliche Gehirn verarbeitet Informationen effizient durch analoge Integration von Eingangssignalen und digitale, binäre Kommunikation. Dieser grundlegende Aufbau wird in Modellen für spikende neuronale Netze erfasst, die darauf abzielen, die Verarbeitungsleistung und Energieeffizienz des Gehirns zu imitieren. Im Rahmen dieser Arbeit leisten wir einen Beitrag zur vielfältigen Optimierung dieser Modelle für die Informationsverarbeitung. Zu diesem Zweck betrachten wir zunächst Strategien zur Quantifizierung der Fähigkeit zur Informationsverarbeitung. Zweitens optimieren wir unsere Netzwerkimplementierungen auf Effizienz, indem wir die analoge Emulation der neurosynaptischen Dynamik auf neuromorpher Hardware nutzen, die durch Nachahmung wichtiger Architekturprinzipien an die Energieeffizienz ihres biologischen Vorbilds anknüpfen soll. Die eigentliche Optimierung der Informationsverarbeitung erfolgt in einem dritten Schritt durch die Nutzung von zwei orthogonalen Ansätzen: Insbesondere betrachten wir gradientenbasierte Methoden in überwachten Lernszenarien und darüber hinaus nutzen wir die gezielte Anpassung von kollektiven Dynamiken für die Informationsverarbeitung, die unter lokaler unbeaufsichtigter Plastizität entstanden sind. Im letzten Schritt verwenden wir übergeordnete Optimierungsstrategien, um Beschränkungen durch die neuromorphe Implementierung zu überwinden. Insbesondere schwächen wir das allgegenwärtige Problem der begrenzten synaptischen Ressourcen durch lokale strukturelle Rekonfigurationen. Indem wir alle genannten Bereiche berücksichtigen, zeigen wir die potenziellen Verarbeitungsmöglichkeiten von spikenden neuronalen Netzen und deren Implementation auf analoger neuromorpher Hardware.

Contents

List of Figures

List of Tables

Acronyms

ADC analog-to-digital converter.

AdEx adaptive exponential leaky integrate-and-fire.

AI artificial intelligence.

AIS active information storage.

ALU arithmetic and logic unit.

ANN artificial neural network.

AP action potential.

API application programming interface.

ASIC application-specific integrated circuit.

BC bushy cell.

BM basilar membrane.

BPTT backpropagation through time.

CADC column-parallel analog-to-digital converter.

CDF cumulative distribution function.

CMOS complementary metal oxid semiconductor.

CNN convolutional neural network.

CPU central processing unit.

CU control unit.

DAC digital-to-analog converter.

DRAM dynamic random-access memory.

DVS dynamic vision sensor.

FFT fast Fourier transformation.

FLAC Free Lossless Audio Codec.

FPGA field-programmable gate array.

GPU graphics processing unit.

H entropy.

HC hair cell.

HD Heidelberg Digits.

HDF5 Hierarchical Data Format 5.

HICANN-X High Input Count Analog Neural Network X.

HICANN-DLS High Input Count Analog Neural Network with Digital Learning System.

HMM hidden Markov model.

I mutual information.

ISA instruction set architecture.

ITL in-the-loop.

LIF leaky integrate-and-fire.

LSTM long short-term memory.

LVDS low voltage differential signaling.

MADC membrane analog-to-digital converter.

MC memory capacity.

MOSFET metal oxide semiconductor field-effect transistor.

NARMA nonlinear auto-regressive moving average.

NMDA N-methyl-D-aspartic acid.

NRMSE normalized root-mean-square error.

OTA operational transconductance amplifier.

PADI parallel driver interface.

PCB printed circuit board.

PID partial information decomposition.

PLL phase-locked loop.

PPU plasticity processing unit.

PSC post-synaptic current.

PSP post-synaptic potential.

RBF radial basis function.

ReLU rectified linear unit.

RMS root mean square.

SC Speech Commands.

SDRAM synchronous dynamic random-access memory.

SERDES serializer/deserializer.

SHD Spiking Heidelberg Digits.

SIMD single instruction, multiple data.

SNN spiking neural network.

SNU spiking neural unit.

SODIM small outline dual inline memory module.

SRAM static random-access memory.

SSC Spiking Speech Commands.

STDP spike-timing dependent plasticity.

STP short-term plasticity.

SVM support vector machine.

TE transfer entropy.

TTFS time-to-first-spike.

USB universal serial bus.

VLSI very-large-scale integration.

VRF vector register file.

XOR exclusive-OR.

1 Introduction

Artificial intelligence (AI) has become a mature element in our daily life. Applications thereof range from the field of healthcare (Jiang et al., 2017) to virtual assistants (Hoy, 2018) present in most modern devices. The name artificial intelligence (AI) was coined during the Dartmouth conference in 1956 which initiated the nowadays broad area of AI and is often used to describe machines that mimic human cognitive functions such as learning, understanding, reasoning, or problem-solving (Russell & Norvig, 2016).

Machine learning is one prominent and successful subfield of AI (Figure 1.1a). It covers algorithms that learn through experience, i. e. through past information (Mohri et al., 2018). Within *supervised learning* scenarios, these methods build a model based on a set of labeled training data to learn the desired input to output mapping without being explicitly programmed to do so (Figure 1.1b). In more detail, the training of the model is often framed as an optimization problem: An error-quantifying loss function is minimized with respect to the model parameters to yield the desired information processing. This is in direct contrast to classical algorithms which would solve a problem like image recognition by a sequence of rules to extract key features from an image (Figure 1.1c).

Neural networks are a prominent example of a type of model considered within the field of machine learning. Originally inspired by the architecture of the brain, artificial neural networks (ANNs) are build of artificial neurons connected to form deep networks (LeCun et al., 2015). These neurons work by sending out analog signals to other neurons. Each neuron weights and aggregates all incoming signals, applies non-linear transformations, and then passes the signal to downstream neurons. In the last few years, there were several outstanding achievements in the field of machine learning with neural networks like *DeepRL* that mastered a diverse range of *Atari* games to superhuman level (Mnih et al., 2016), *AlphaGo* that defeated the world's best player in the game of *Go* (Silver et al., 2016) and *GPT-3* an autoregressive language model that produces human-like text (Brown et al., 2020). Despite their success, the solutions found by machine learning require huge amounts of data as well as processing power which comes at the expense of the energetic footprint (Strubell et al., 2019). In addition, there is still a large gap between current implementations and a truly human-like intelligence, albeit AI system like *Google Duplex* approached to pass the Turing test proposed in 1950 (Turing & Haugeland, 1950). Due to these shortcomings, it might be promising to once again draw inspiration from biological brains to bridge the gap between the achievements of current AI solutions obtained with neural networks and brain-like performance.

Despite similarities, biological neurons differ in important aspects from their artificial equivalents. Most notably, they do not communicate analog signals, but transmit sequences of precisely timed binary events so-called spikes (Bernstein, 1868; Perkel & Bullock, 1968). Nevertheless, the integration of input signals at a given neuron is based on analog signals. This combination of analog integration and digital processing renders the brain very special. Furthermore, the inherently sparse coding with binary events guarantees that each spike carries high information. As a result, the communication is efficient as the main energy cost comes from the generation of spikes. These key properties of

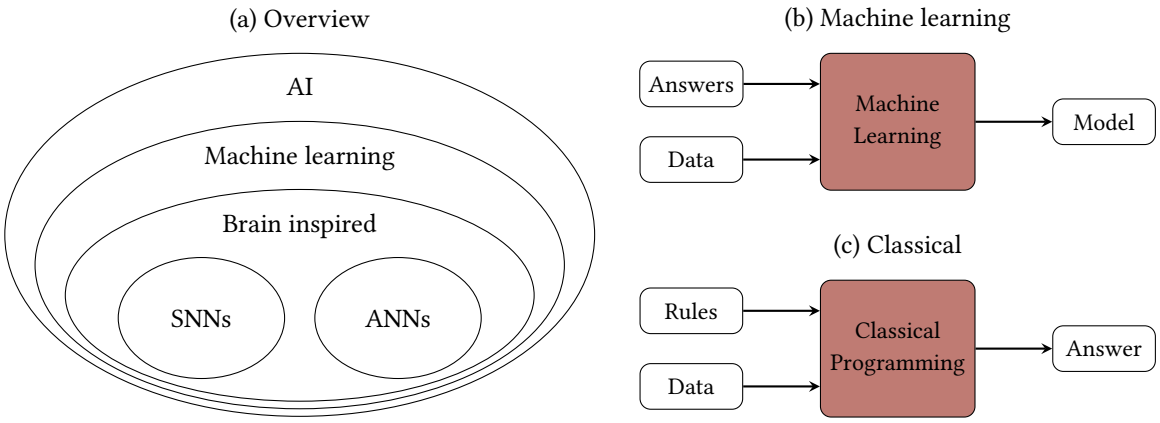


Figure 1.1: Overview of the field of AI. AI is a broad discipline which addresses very different subfields with distinct methods. **(a)** Machine learning represents a subfield of AI. It contains the branch of brain inspired approaches which encompasses SNNs as well as ANNs. The classical programming paradigm differs significantly from the machine learning approach. **(b)** Machine learning algorithms build models based on a set of training data to make further predictions or decisions without being explicitly programmed to do so. **(c)** Classical algorithms, in contrast, solve a problem by a specific sequence of rules. Figure inspired by Annoni et al. (2018).

biological neurons are captured within spiking neural network (SNN) models (Hodgkin & Huxley, 1952; Lapicque, 1907; Gerstner, 2001; Brette & Gerstner, 2005).

Most current ANNs as well as SNNs models are implemented on conventional computing systems which do not capture the intrinsic parallelism of both architectures. Specifically, current computer systems feature distinct components to store and process information sequentially (Von Neumann, 1993). The speed at which data can be transferred from memory to the processor and vice versa is often a limiting factor referred to as *von Neumann bottleneck*. Currently, there are efforts underway to engineer hardware systems that mimic the human brain more closely (Schuman et al., 2017). Computation in the brain is performed locally, asynchronously, and most notably in parallel: Memory, learning, and processing in the brain are all located alongside in the neurons and synapses. The latter constituents are slower (Hildebrand et al., 1993) and less reliable (Allen & Stevens, 1994; Czanner et al., 2015) than the substitutes of conventional processors. However, this drawback is overcome by redundancy: In any computation, a large number of neurons and synapses is involved in a highly parallel manner. Learning is accomplished by continuous adjustments of the excitability of single neurons, the coupling strengths between neurons as well as the network topology over time to finally perform information processing based on sensory input data (Huttenlocher, 2009). It is noteworthy that in contrast to machine learning applications, often neither the task nor the exact form of the optimization is known for biological brains. Neuromorphic engineering tries to mimic key architectural as well as functional properties of biological tissue with the aim to yield novel computing paradigms (Schuman et al., 2017).

Despite all advantages, SNNs pose several challenges when put to function (Figure 1.2). For example, the performance of SNNs on standard machine learning classification benchmark tasks as MNIST (LeCun et al., 1998) and CIFAR (Krizhevsky et al., 2009) is not competitive with their machine learning counterparts. A first reason for this discrepancy is the necessity of conversions from the input samples to spikes (Figure 1.2a). Second, there is a lack of training algorithms that fully exploit the quoted advantages of SNNs (Figure 1.2c). Hence, the problem of optimizing SNNs for

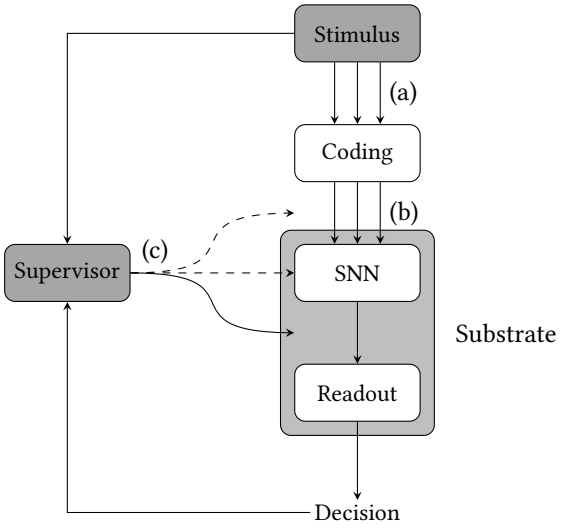


Figure 1.2: Information flow from stimulus to decision. The optimization of SNNs for information processing can act at all levels ranging from input coding and injection to the downstream classification network. **(a)** Withing the coding step, the stimulus is transformed to comply with the alphabet of the SNN implemented on an arbitrary compute substrate **(b)**. For classification, the substrate furthermore houses a classifier making a final decision. Both, the network and the classifier may have access to a supervisor providing a stimulus-specific target signal needed to adapt the model parameters during learning. The methods described within this thesis differ in the availability and application of this signal within the scope of network optimization **(c)**.

information processing does not only require suitable learning algorithms, but involves the whole processing chain from stimulus pre-processing to decision making.

Brains processes a broad range of different temporal inputs ranging from visual, auditory, and chemical to tactile stimulation (Marzvanyan & Alhawaj, 2019). These stimuli stand in stark contrast to classical benchmark tasks – e. g. the processing of static images – which often do not challenge or even exploit the temporal processing capabilities provided by SNNs. To apply more natural types of temporal stimuli to SNNs, one has to likewise mimic parts of the sensory system to transform time series data into spike trains (Figure 1.2a). The resulting models obviously impact the information processing capabilities of subsequent SNNs (Davies et al., 2018; Cramer et al., 2020b). Moreover, they often act on high-dimensional data or even increase dimensionality. However, in most scenarios, the input signal is still sparse and not all features are relevant for the actual task. As the processing of these irrelevant features requires additional computational resources, it is highly desirable to exclude them prior to injection into an SNN in a higher-level optimization. The actual optimization required to reach high performance on a given task is often accomplished by adjusting the coupling strength to and in between the neurons constituting the SNN as well as the connections between the SNN and a readout. In general, we will distinguish between two different scenarios: In *supervised learning* tasks, the network parameters are optimized to yield a specific input to output mapping. In contrast, the adaptation of parameters is done in a task-unspecific manner within *unsupervised learning* experiments. In the latter scenario, the input to output mapping is often only imposed to optimize connections from the SNN to the readout while the connections to and within the SNN are adjusted in a task-independent way.

Within this thesis, we aim to contribute to all the stages of SNN optimization shown in Figure 1.2. To that end, we first provide the required theoretical background in Chapter 2. The optimization of

SNNs also encompasses their efficient implementation. Here, we consider the emulation of SNNs on analog neuromorphic hardware. Specifically, we introduce the BrainScaleS-2 architecture in Chapter 3 which builds the target platform for many of the experiments presented within this thesis. The optimization of SNNs for information processing is first visited in the scope of Chapter 4. Here, we present a novel, publicly available collection of spoken words as well as a spike-based version thereof and finally establish a first set of benchmarks with SNNs. By directly providing spike trains to the modelers, we not only minimize computational overhead, but render the coding step in Figure 1.2 universal and hence pave the way toward well-profounded comparisons. Regarding the SNN optimization for task processing on neuromorphic hardware, we visit two orthogonal approaches. First, we present a supervised learning method in Chapter 5 relying on surrogate gradient learning for the training of SNNs. Second, we present an unsupervised learning approach that exploits collective phenomena to adapt SNNs to task requirements in Chapter 6. While the first approach ties on the success of gradient-based learning techniques with backpropagation through time (BPTT), the second one is more plausible from a biological perspective and represents a step toward fully local learning paradigms. Last, we consider higher-level optimizations for the efficient use of synaptic resources on neuromorphic substrates. To that end, we consider synaptic rewiring – a process also found in biological tissue – and present an efficient implementation thereof on the neuromorphic hardware system BrainScaleS-2. All result chapters start with a short abstract thematically classifying the content in the respective stage of SNN optimization.

2 Background

In this chapter, we will introduce several basic aspects of neuroscience and neuromorphic computing. Both of the aforementioned disciplines focus on the brain which builds the center of the nervous system of all vertebrate and most invertebrate animals. It acquires, stores, and processes information in vastly complicated networks. These networks consist of neurons (Section 2.1.2) which are locally and directionally coupled by synapses (Section 2.1.3). Computationally, the human brain has several desirable properties: It processes information inherently parallel, has a power consumption on the order of tens of watts for operation, and dynamically adapts to its environment on slow time scales by means of plasticity (Section 2.1.4). Hence, adopting computation and design principles from the highly optimized brain for the development of novel computing paradigms is a promising approach when complicated tasks need to be solved. Nowadays, detailed knowledge about the neuro-synaptic dynamics is available allowing to build complex models which in turn can be simulated on conventional computers (Section 2.2.1). However, the latter operate in an entirely different manner which is prohibitive when trying to exploit the full benefits of brain-inspired computing. To overcome these limitations, one may again draw inspiration from the brain to build dedicated compute systems, so-called neuromorphic systems, which have the potential to constitute the hardware for novel brain-inspired computing paradigms (Sections 2.2.2 and 2.2.3).

2.1 Biology

Within this section, we draw on a top-down approach: First, we highlight the basic architectural principles of the brain, before moving on to the functionality of neurons and synapses. To that end, simplified models of neuro-synaptic dynamics will be introduced as well as principles of synaptic plasticity. For further details regarding the neuroscientific background and modelling, the reader is referred to Dayan & Abbott (2001) and Gerstner & Kistler (2002) which also served as a guideline for this background chapter.

2.1.1 Nervous System

Over the past years, we have gained increasing knowledge about the structure and function of biological brains. Santiago Ramón y Cajal – one of the pioneers of neuroscience – first noticed that distinct cells form the central processing element of the brain (Figure 2.1a). The discovery of these neurons was groundbreaking, since at that time the prevailing hypothesis was that of a continuum of nerve cells, rather than individual elements. Despite different sizes and shapes, neurons form the universal processing element of the nervous system. The description of the existence of dendritic spines constitutes Cajal's second major contribution (Figure 2.1b). His interpretation of spines as possible targets of axons further underpinned his neuron doctrine (Cajal, 1890). Within the scope of his last articles he utilized the term *synapses* to refer to the aforementioned connections (Cajal, 1933).

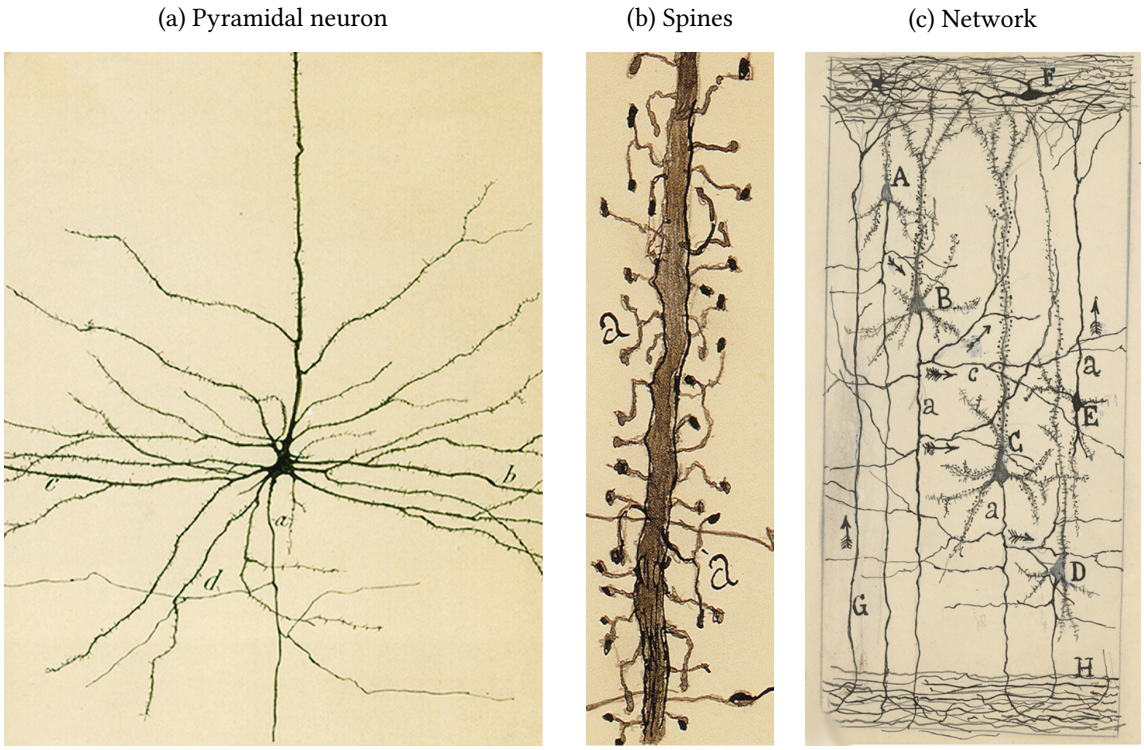


Figure 2.1: First illustrations of the nervous system drawn by Cajal. (a) Drawing of a pyramidal neuron of the human motor cortex (Cajal, 1899). The legend states: an initial part of the axon; b dendrites; d axonal collaterals. **(b)** Illustration of dendritic spines of pyramidal cells in the cerebral cortex of a 2 month old child (Cajal, 1933). The drawing in (a) highlights that the surface of the dendrites are covered with spines. **(c)** Schematic depiction of synaptic connections and information flow in the cerebral cortex (Cajal, 1933). The legend states: A small pyramid; B,C medium and giant pyramids, respectively; a axon; c, nervous collaterals that appear to cross and touch the dendrites and the trunks (apical dendrites) of the pyramids; H white matter; E, Martinotti cell with ascending axon; F special cells of the first layer of cerebral cortex; G fiber coming from the white matter. The arrows mark the supposed direction of the nervous current. Figures and descriptions are taken from DeFelipe (2015).

Cajal manufactured a schematic drawing in his article (Cajal, 1933) which illustrates the connections by contact of the collateral axonal branches of the pyramidal cells with dendritic spines (Figure 2.1c). This early drawing already hints towards the complexity of densely connected neurons: A brain of an adult male human has about 86 billion neurons and 85 billion non-neuron cells at a weight of 1.5 kg (Azevedo et al., 2009). The number of synapses present in the brain is known much less precisely, but is estimated to be 0.15 quadrillion (Pakkenberg et al., 2003). In the remainder of this thesis we will focus on spiking neurons, although there are analog neurons and supporter cells in the brain as well.

Neurons constitute electrical excitable cells which are specialized for generating and forwarding of electrical responses. They propagate signals over long distances by short electrical pulses – so-called action potentials (APs) or spikes – first observed by Bernstein (1868). These APs are fluctuations in the electrical potential across the cell membrane with an amplitude of approximately 100 mV lasting for a duration of about 1 ms. Neurons process and propagate information by firing sequences of these spikes in diverse temporal codes. Hodgkin and Huxley studied in their collaboration the fundamentals of nerve cell excitability (Hodgkin & Huxley, 1952). They not only provided

an understanding of how voltage-gated ion channels traversing the cell membrane of neurons give rise to propagating APs, but also developed a framework for studying and analyzing ion channel kinematics. While the analog subthreshold potentials of neurons are usually attenuated over relative small ranges of 1 mm or less, the active regeneration of APs facilitates the propagation over long distances.

A neuron can be mechanistically divided into three distinct parts (Figure 2.1). First, the *dendrites* – a tree-like structure – which serves as input site. Second, the *soma* hosts the whole bio-chemical machinery similar to other cells. Further, it also forms the central unit for information processing by implementing a non-linear transformation: Only if the input strength exceeds a certain threshold an AP is generated. This signal propagates along the *axon*, the third element, which terminates at the *synapses*. While neurons send out dendrites with an average length of 3 mm, the length of axons varies between 1 mm to 17 mm for pyramidal neurons in the mouse cortex (Braitenberg & Schüz, 2013). In the human brain, myelinated nerve fibres span approximately 150 000 km to 180 000 km in total (Pakkenberg et al., 2003).

Neurons are connected to networks via synapses. In the vertebrate cortex, neurons often target more than 10^4 postynaptic neurons. These connections are often expressed to their neighbouring neurons, but also extend over larger distances to different brain areas (Braitenberg & Schüz, 2013). Most of these connections in the vertebrate cortex are of chemical type which are characterized by a close gap between the pre- and the postsynaptic membrane called the synaptic cleft (Südhof & Malenka, 2008). Here, the prefix pre refers to the transmitting site, whereas the receiving partner is termed postsynaptic. At a synapse, an arriving AP causes the release of neurotransmitters into the synaptic cleft. These transmitter molecules diffuse to the postsynaptic membrane where they cause an influx of ions and thereby lead to a electrical response, the post-synaptic potential (PSP). Apart from chemical synapse, there are also gap-junctions which allow for a direct exchange of ions between neurons (Evans & Martin, 2002).

2.1.2 Neurons

As already mentioned in the previous section, neurons are specialized to communicate via electrical pulses. The key morphological features for the generation of these pulses are ion channels traversing the cell membrane (Figure 2.2a). The latter constitutes a lipid bilayer which is impermeable for ions and separates the extracellular space from the interior of the neuron. Embedded ion channels control the flow of specific ions either statically or mediated by a variety of signals. We consider the membrane potential – the electrical potential between the interior and the surround of a neuron – as the relevant electrical signal. On the one hand, the magnitude of this potential is small enough such that neurons are able to exploit the thermal energy of ions for their transport across the membrane. On the other hand, the magnitude is large enough so that thermal fluctuations do not disturb the electrical communication of the neuron.

Let us first shed light on the processes impacting the equilibrium potential. In general, there are two distinct forces that drive ions through ion channels: First, there are electrical forces and second, there is diffusion caused by concentration differences maintained by ion pumps (Figure 2.2a). We consider an ion with electric charge ne where e denotes the charge of one proton. At membrane potential u, its thermal energy has to exceed $-neu$ to cross the membrane. The Boltzmann factor $\exp(neu/k_\mathrm{B}T)$ denotes the probability of having an energy greater or equal to $-neu$ at temperature T. In case of an opposing concentration gradient, the effect of the membrane potential can be overcome.

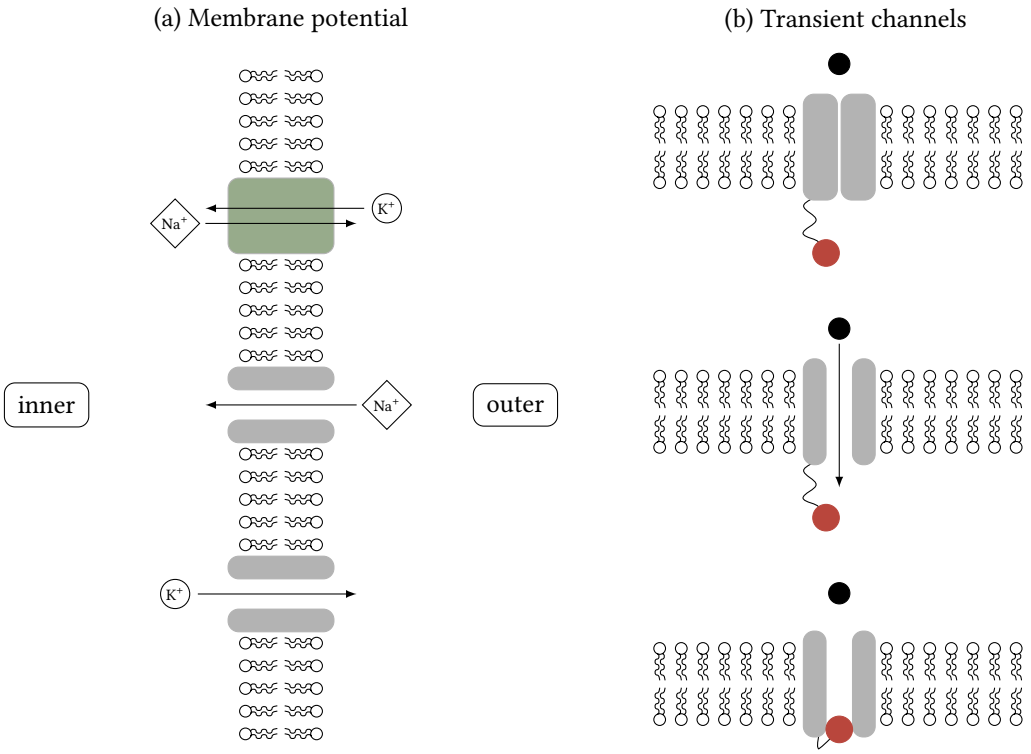


Figure 2.2: Electrophysiology of neurons. **(a)** The cell membrane is formed by a lipid bilayer which is traversed by ion channels. The electrical potential between the interior and the surround of the neuron defines the membrane potential. The equilibrium potential is affected by concentration gradients and diffusion: Na^+-K^+ pumps (green) cause concentration gradients and selectively permeable ion channels allow specific ion types to constantly diffuse across the membrane. **(b)** In contrast to the channels in **(a)**, transient ion channels open transiently when the membrane potential is depolarized. In equilibrium, the channel is closed (top). If the membrane potential rises, the channel opens (middle), thereby allowing ions to traverse the membrane. Afterwards, the channel is inactivated by a second gate (bottom).

Here, ions flow with a rate proportional to their concentration in the extracellular space $[\text{Conc}]_{\text{out}}$ into the cell. In the opposite direction, the flow is proportional to the concentration within the cell $[\text{Conc}]_{\text{in}} \exp(neu/k_{\text{B}}T)$, since a sufficient amount of thermal energy is required to pass the cell membrane. In equilibrium, these flows balance and we write for the associated potential u_{x}:

$$u_x = \frac{k_{\text{B}}T}{ne} \log\left(\frac{[\text{Conc}]_{\text{out}}}{[\text{Conc}]_{\text{in}}}\right), \tag{2.1}$$

where x refers to the considered ion type. This is the so-called Nernst equation which holds true for ion channels that allow only a single ion type to pass the cell membrane (Nernst, 1889). In a biological setting, the already addressed concentration gradients across the cellular membrane are maintained by ion pumps. As a result, there is an excess of Na^+ ions in the extracellular medium which translates to a reversal potential of $u_{\text{Na}} = 50\,\text{mV}$. In contrast, the concentration of K^+ ions inside the neuron exceeds the outer one resulting in $u_{\text{K}} = -70\,\text{mV}$ to $-90\,\text{mV}$ (Dayan & Abbott, 2001).

All different ion types at hand impact the total current flowing across the membrane. Specifically, the contribution of type x channels with conductance g_x is given by $g_x(u - u_x)$. The exterior and the interior of the neuron can be modelled as conductors, whereas the cell membrane constitutes an

insulator. By definition, the membrane has a capacitance which we denote by C_{mem}. With this, the net currents due to type x ions equal:

$$C_{\text{mem}} \frac{\mathrm{d}u}{\mathrm{d}t} = \sum_x g_x(u - u_x)\,, \tag{2.2}$$

where the sum extends over all present ion types.

Up to this point, we only considered the equilibrium potential induced by passive ion channels. However, the generation of APs as well as their propagation is caused by active conductances which introduce nonlinearities. Therefore, we will introduce additional equations that determine the characteristics of the corresponding conductances g_x. To that end, we assume g_x to be the product of a maximum conductance $\bar{g}_x$ and the probability of the gate being open P_x. Here, we focus on two types of conductances: Persistent conductances for which P_x is a function of the voltage and transient channels which only open transiently.

Let us first consider persistent channels which we assume to behave like a simple swinging gate. The opening of these gates involves a series of conformational changes. Let us assume that k independent and identical changes promote an opening of the channel, each of which occurring with a probability n. Then, the probability of the gate to open is given by $P_x = n^k$. The K^+ conductance is an example of a persistent gate for which Hodgkin and Huxley suggested the form $P_K = n^4$ (Hodgkin & Huxley, 1952).

For transient channels, in contrast, two processes with opposing voltage dependency are involved (Figure 2.2b). As a result, they only open temporarily in case the membrane potential is sufficiently depolarized. The first process behaves similar to a persistent gate which is why we can describe it by m^k. In addition, we denote the probability of the second gate being non-blocking by h. The variables m and h are characterized by opposite voltage dependencies; a depolarization of the membrane potential entails an increase of m, whereas h decreases and vice versa. One example of a transient channel is the Na^+ conductance for which Hodgkin and Huxley predicted the form $P_{\text{Na}} = m^k h$ with $k = 3$ (Hodgkin & Huxley, 1952).

For each subunit of each of the discussed channels, we can describe the rate at which the gating transition from open to close occurs by a voltage-dependent function $\alpha_x(u)$ and the reverse transition by a rate $\beta_x(u)$. Thus, we write for the probability of any gating event y to occur:

$$\frac{\mathrm{d}y}{\mathrm{d}t} = \alpha_y(u)(1 - y) - \beta_y(u)y, \tag{2.3}$$

with $y \in \{n, m, h\}$. This equation can be rewritten:

$$\tau_y(u)\frac{\mathrm{d}y}{\mathrm{d}t} = y_\infty(u) - y, \tag{2.4}$$

where we have defined the time constant $\tau_y(u)$ as well as the steady-state value $y_\infty(u)$:

$$\tau_y(u) = \frac{1}{\alpha_y(u) - \beta_y(u)}\,, \qquad y_\infty(u) = \frac{\alpha_y(u)}{\alpha_y(u) + \beta_y(u)}\,. \tag{2.5}$$

The form of the functions $\alpha_y(u)$ and $\beta_y(u)$ can be motivated by thermodynamic arguments and have been fitted to experimental data obtained in voltage clamping experiments by Hodgkin and Huxley

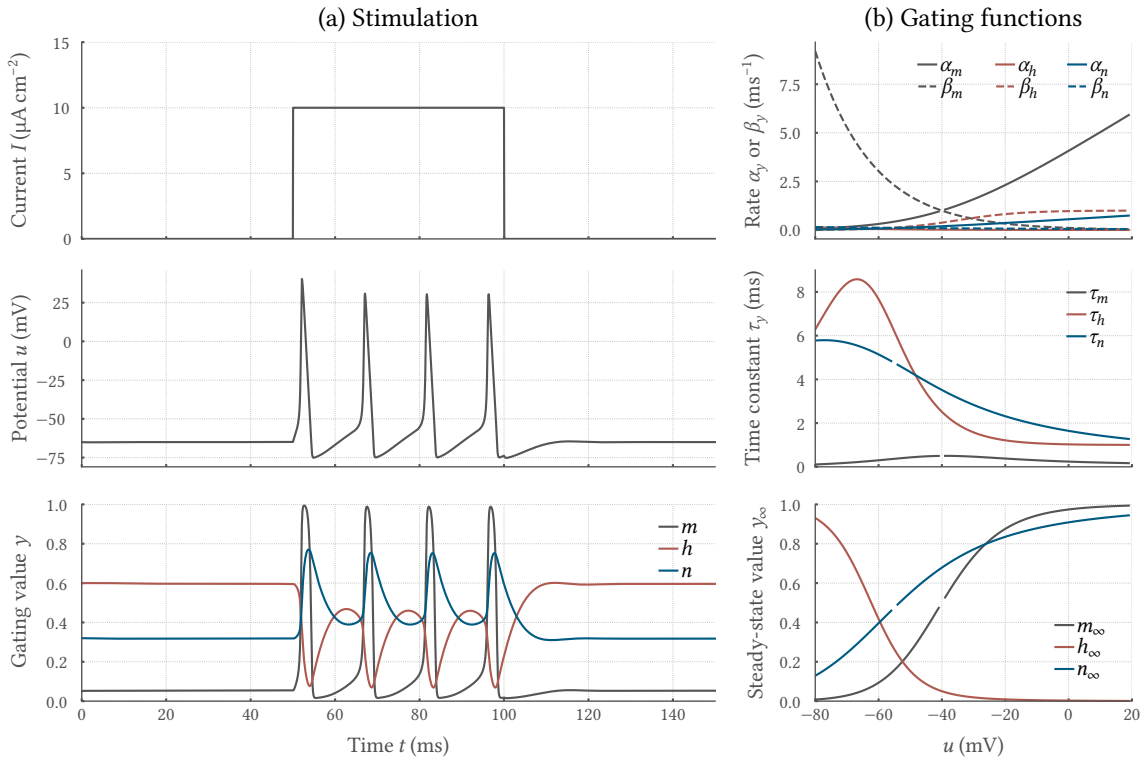


Figure 2.3: Illustration of the functionality of the Hodgkin-Huxley model. (a) We consider the model in response to a rectangular stimulus current I initiated at $t = 50$ ms (upper panel). After stimulus onset, a sequence of APs is triggered as long as the input current persists (middle panel). The initial rapid rise of the membrane potential u constituting each AP is caused by a sudden change in the Na^+ conductivity $g_{Na}m^3h$ (lower panel). In turn, the rise in u causes the inactivation of the gating variable h thereby shutting off the Na^+ current. Moreover, high values of u activate the K^+ conductance $g_K n^4$, pulling u back to negative values. In a final recovery process, all variables are readjusted and the processes recur as long as the stimulus persists. **(b)** The rate functions α_y and β_y (upper panel), the time constants τ_y (middle panel) as well as the steady-state values y_∞ (lower panel) are functions of u for all gates.

(Hodgkin & Huxley, 1952):

$$\alpha_n(u) = \frac{0.01(u+55)}{1-\exp\left(-0.1(u+55)\right)}, \qquad \beta_n(u) = 0.125\exp\left(-0.0125(u+65)\right), \tag{2.6}$$

$$\alpha_m(u) = \frac{0.01(u+40)}{1-\exp\left(-0.1(u+40)\right)}, \qquad \beta_m(u) = 4\exp\left(-0.0056(u+65)\right), \tag{2.7}$$

$$\alpha_h(u) = 0.07\exp\left(-0.05(u+65)\right), \qquad \beta_h(u) = \frac{1}{1+\exp\left(-0.1(u+35)\right)}. \tag{2.8}$$

These functions as well as the resulting time constants τ_y and steady-state values y_∞ are shown in Figure 2.3.

Combining the terms for constant, persistent and transient conductances results in the Hodgkin-Huxley model (Hodgkin & Huxley, 1952) capturing the generation of APs:

$$C_{\text{mem}}\frac{\mathrm{d}u}{\mathrm{d}t} = -g_{\text{leak}}(u-u_{\text{leak}}) - g_K n^4(u-u_K) - g_{Na}m^3h(u-u_{Na}) + I, \tag{2.9}$$

where the constant factors are grouped together in a so called leakage current $g_{\text{leak}}(u-u_{\text{leak}})$. Figure 2.3 shows the temporal evolution of the dynamic variables of the Hodgkin-Huxley model during

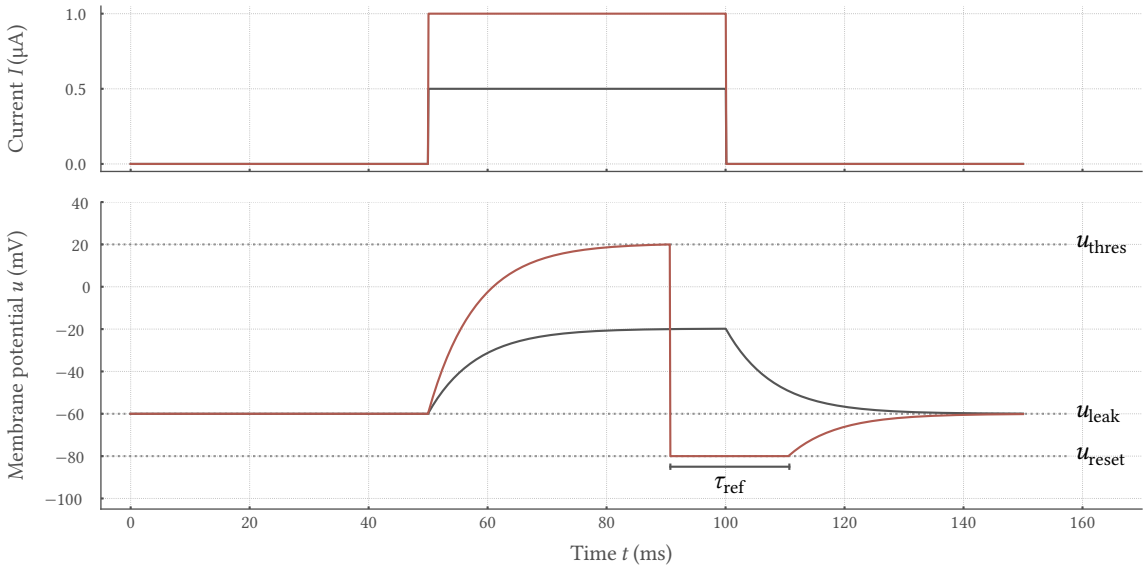


Figure 2.4: Illustration of the functionality of the LIF neuron model. The LIF neuron integrates its current input I on its membrane potential u upon reaching threshold u_{thres}. When reaching u_{thres} at time t, the neurons emits a spike at t and u is clamped to the reset potential u_{reset} for the duration of the refractory period τ_{ref}. After expiration of τ_{ref}, u is released and again evolves freely. In the absence of any stimulation, u decays to the leak potential u_{leak}. The model dynamics in response to stimulus currents with different amplitudes is depicted by different colors.

stimulation with a positive step current. First, the injected current leads to a rise of the membrane potential, thereby driving the m variable to a value of one. Since the h variable initially deviates from zero, the Na^+ conductivity m^3h increases which promotes an influx of Na^+ ions. As a result, the membrane potential u is rapidly driven to around 25 mV. Note the positive feedback effect of both u and m: A depolarized membrane potential lets m increase and in turn activates the Na^+ conductance causing u to increase. Transiently, the rising membrane potential drives h towards zero. As a result, the Na^+ conductance gets inactivated, thereby suspending the Na^+ current. In addition, the depolarized u drives n towards one and activates the K^+ conductance. This promotes an increased K^+ current, which then hyperpolarizes the membrane potential. Finally, the m, h, and n variables are restored and u decays to its resting value.

The four-dimensional Hodgkin Huxley model can be reduced to two dimensions. Figure 2.3b shows that the time scale of the gating variable m is substantially faster than the one of h, n, and u. Therefore, we replace m by its steady-state value $m_\infty(u)$. In addition, the time constants $\tau_n(u)$ and $\tau_h(u)$ are rather similar, independent of u. Furthermore, the course of $n(u)$ and $1 - h(u)$ are roughly similar allowing to replace both gating variables by an effective variable w. By setting $(b - h) \approx an$ with constants a, b and $w = b - h$ we get:

$$C_{\text{mem}} \frac{\mathrm{d}u}{\mathrm{d}t} = -g_{\text{leak}}(u - u_{\text{leak}}) - g_{\text{K}} \left(\frac{w}{a}\right)^4 (u - u_{\text{K}}) - g_{\text{Na}} m_0^3(u)(b - w)(u - u_{\text{Na}}) + I, \tag{2.10}$$

$$\frac{\mathrm{d}w}{\mathrm{d}t} = \frac{1}{\tau_w}(an(u) - w)\,, \tag{2.11}$$

with time constant τ_w. In case $\tau_w \gg \tau$, $\mathrm{d}u/\mathrm{d}t$ is larger than $\mathrm{d}w/\mathrm{d}t$ except around the nullcline of u. Hence, the model can be assumed to have a firing threshold u_{thres} and spikes become stereotypical

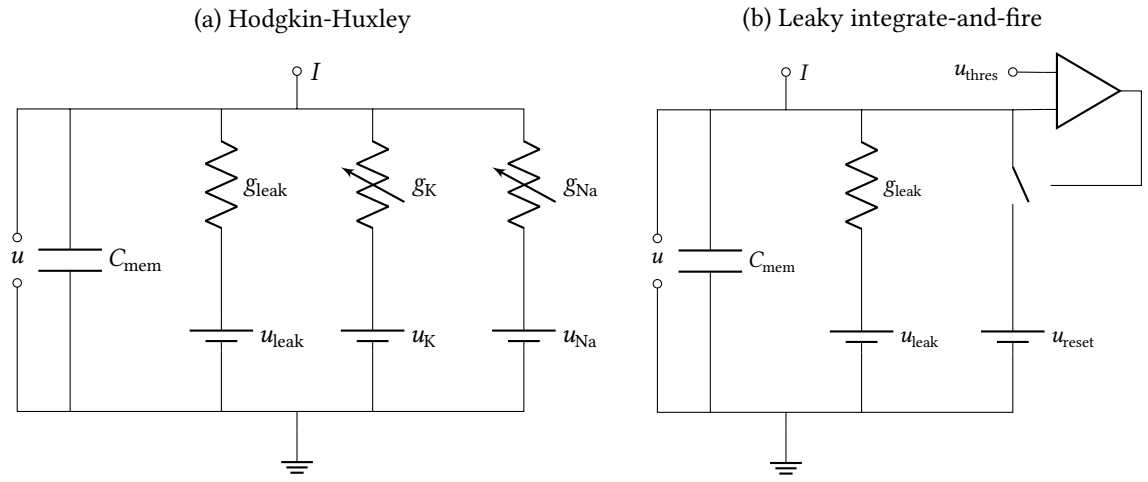


Figure 2.5: Equivalent electrical circuits of the Hodgkin-Huxley and LIF neuron model. The lipid bilayer is emulated by a capacitor with capacitance C_{mem}. All constant contributions to u are grouped together in the conductivity g_{leak} and the potential u_{leak}. **(a)** Each ion channel for the ion type x corresponds to a variable conductivity g_x and specific reversal potential u_x given by a battery. **(b)** In the LIF circuit, the ion channel blocks for Na^+ and K^+ are replaced by a comparator which triggers a switch in case the potential u exceeds a threshold u_{thres}.

events (Figure 2.4). This motivates the leaky integrate-and-fire (LIF) neuron model:

$$\tau_{\text{mem}} \frac{\mathrm{d}u}{\mathrm{d}t} = -\left[u(t) - u_{\text{leak}}\right] + \frac{I}{g_{\text{leak}}}\,, \tag{2.12}$$

with time constant $\tau_{\text{mem}} = C_{\text{mem}}/g_{\text{leak}}$ (Lapicque, 1907). Here, the spikes are described formally by a threshold criterion:

$$t^k : u_j(t^k) \geq u_{\text{thres}}\,, \tag{2.13}$$

with the k-th firing time t^k and the threshold potential u_{thres}. Immediately after t^k, the membrane potential is clamped to the reset potential $u(t) = u_{\text{reset}}$ for $t \in \left(j^k, t^k + \tau_{\text{ref}}\right]$, with the refractory period τ_{ref} capturing the readjustment of gating variables after an AP. There are also extensions to the described LIF neuron model. One prominent example is the two-dimensional adaptive exponential leaky integrate-and-fire (AdEx) model which is based on an exponential spike mechanism and an additional adaptation equation to cover a broad range of activity patterns (Brette & Gerstner, 2005; Naud et al., 2008).

The electrical properties of both presented models can be translated into equivalent electrical circuits (Figure 2.5). Here, the reversal potentials correspond to batteries, connected in parallel to the capacity C_{mem} as formed by the membrane. The ion channels are included in form of variable resistances in combination with their respective ion-conducting reversal potential. For the leakage current, the resistance is constant. Additional input currents could either charge the capacitor or flow through the resistances, depending on their actual value. We will further comment on the relevance of these equivalent circuits in the context of neuromorphic hardware depicted in Section 2.2.3 and Chapter 3.

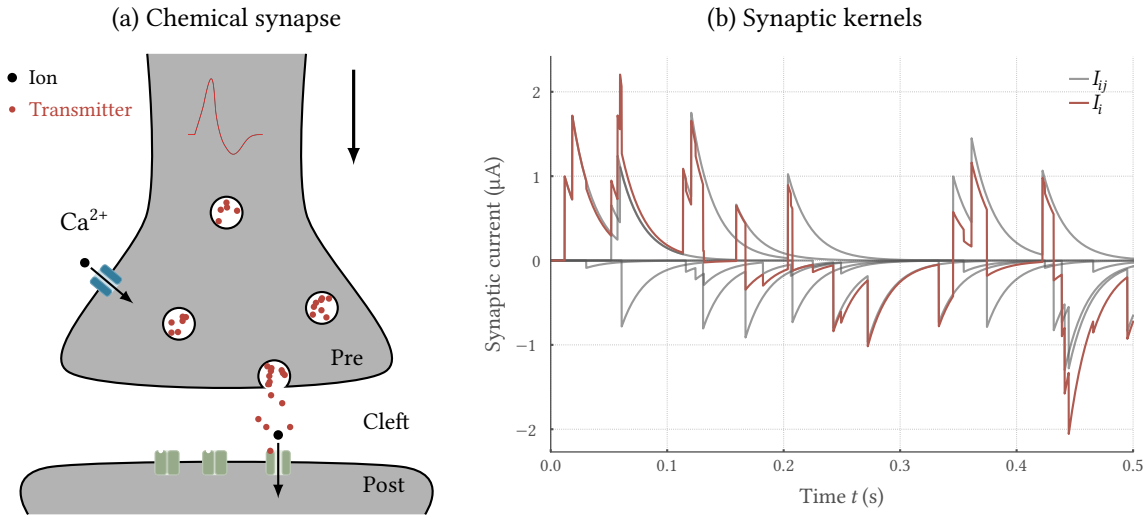


Figure 2.6: Spike-mediated chemical synapse. **(a)** Schematic drawing of a chemical synapse. Vesicles filled with neurotransmitter are released into the synaptic cleft in case an AP arrives at the bouton. Transmitter molecules diffuse to the postsynaptic membrane where they cause the opening of ion channels. **(b)** Dynamics of current-based synapse model. Shown are six synapses with random weights and time constant $\tau_{\text{syn}} = 20\,\text{ms}$ which transmit Poisson distributed spikes at a rate of 10 Hz (gray). The resulting synaptic currents I_{ij} are summed by the postsynaptic neuron j, resulting in the total synaptic current I_i (red).

2.1.3 Synapses

Synapses build the connections between neurons. They either directly transmit electrical signals or relay chemical messengers to the postsynaptic neuron (Pereda, 2014). At a spike-mediated chemical synapse, an arriving AP leads to the opening of Ca^{2+} channels and an associated influx of Ca^{2+} into the presynaptic terminal (Figure 2.6a). This rise in concentration induces the release of neurotransmitters in the synaptic cleft. The transmitter molecules diffuse to the postsynaptic side of the synapse where they undergo a binding reaction with receptors, thereby opening ion channels and shifting the PSP. Because of this, we can model synapses by a time-dependent conductivity $g_{\text{syn}}(t)$ that opens whenever a presynaptic spike occurs. The synaptic current is then given by:

$$I_{\text{syn}}(t) = g_{\text{syn}}(t)(u - u_{\text{syn}})\,. \tag{2.14}$$

This expression can be directly incorporated into Equation (2.9) or Equation (2.12), respectively. Depending on the ion type and the respective reversal potential u_{syn}, synapses are either excitatory if they increase the postsynaptic membrane potential or inhibitory in case they decrease the potential.

Let us assume that the transmitter directly interacts with a channel in a binding reaction: While the binding of transmitter molecules causes the receptor to open, the unbinding promotes the closing. In this setting, we are able to use the same formalism as for voltage-dependent channels to describe $g_{\text{syn}}(t) = \bar{g}_{\text{syn}} P_{\text{syn}}(t)$ with a conductance $\bar{g}_{\text{syn}}$ and an opening probability P_{syn} obeying the form:

$$\frac{\mathrm{d}P_{\text{syn}}}{\mathrm{d}t} = \alpha_{\text{syn}}(1 - P_{\text{syn}}) - \beta_{\text{syn}} P_{\text{syn}}\,. \tag{2.15}$$

Here, we introduced the opening rate α_{syn} as well as the closing rate β_{syn}. The opening rate is a function of the transmitter concentration available for the binding reaction with the receptor. In more detail, an AP arriving at the pre-synaptic terminal causes the transmitter concentration and

in turn α_{syn} to rise which results in an increase of P_{syn}. Subsequently, the transmitter concentration in the cleft is rapidly decreased by diffusion, degradation and reuptake mechanisms.

Up to this point, we only considered a single synapse which is artificial compared to biological systems where a single neuron receives input spikes from up to 15 000 synapses, each of which transmitting multiple spikes. To capture this situation, we first need to model the total synaptic current stimulating neuron i by summing over all presynaptic partners j:

$$I_i(t) = \sum_j I_{ij}(t)\,. \tag{2.16}$$

In the following, we will denote the spike train emitted by neuron j, $S_j(t) = \sum_k \delta(t - t_j^k)$ where the sum extends over all emitted spikes. Since most of the synapses are located far away from the soma of the neuron and the change in conductance described by Equation (2.14) only occurs locally, we can utilize the simplified model of a current pulse reaching the soma through passive propagation. In addition to this current-based approximation, we consider the release and diffusion of transmitter molecules to happen instantaneously. Further, we assume β_{syn} to be constant and set $1/\beta_{\text{syn}} = \tau_{\text{syn}}$. With these simplifications, the synaptic currents $I_{ij}(t)$ can be obtained from Equation (2.15):

$$\tau_{\text{syn}} \frac{\mathrm{d}I_{\mathrm{ij}}(t)}{\mathrm{d}t} = -I_{ij}(t) + \bar{I}_{\text{syn}} w_{ij} S_j(t)\,, \tag{2.17}$$

where w_{ij} denotes the weight of the synapse connecting neuron i and j. The latter impacts the amplitude of an elicted PSP which has been shown to correlate with the spine head size (Matsuzaki et al., 2001). Figure 2.6b shows an example of the current-based synapse model introduced above. The resulting exponential shape is a common first-order approximation (Gerstner & Kistler, 2002). We will stick to this exponential form throughout the remainder of this thesis, albeit more detailed approximations for the solution of Equation (2.15) are available.

2.1.4 Plasticity

One fascinating property of the mammalian brain is the modification of neural circuit function based on past experience (Kolb, 2013). This process is called plasticity and has the potential to impact all future behavior. Synaptic plasticity in particular refers to the activity-dependent modification of the synaptic weights w_{ij} at existing synapses (Citri & Malenka, 2008). Many different forms of synaptic plasticity have been described in the past. The associated changes can either be enhancing or depressing depending on the activity and can occur on time scales from milliseconds to hours, days, and even longer. Furthermore, almost all excitatory synapses in the mammalian brain simultaneously exhibit different forms of synaptic plasticity. In the following, we will shortly discuss short- and then move on to long-lasting forms of synaptic plasticity.

Different forms of short-term plasticity (STP) have been observed in almost all synapses in organisms ranging from invertebrates to mammals (Zucker & Regehr, 2002). The resulting effects last from milliseconds to several minutes and are caused by a dependence of the transmitter release probability on the spike history at the synapse (Markram & Tsodyks, 1996). There are two principle modes of STP (Markram et al., 1998): In case the postsynaptic potential decreases in amplitude in response to repeated presynaptic activation, the effect is termed depressing. In contrast, for a facilitating synapse, the pulse amplitude increases. Both effects can be modeled by a presynaptic transmitter release probability comparable to Equation (2.15). Here, the release probability

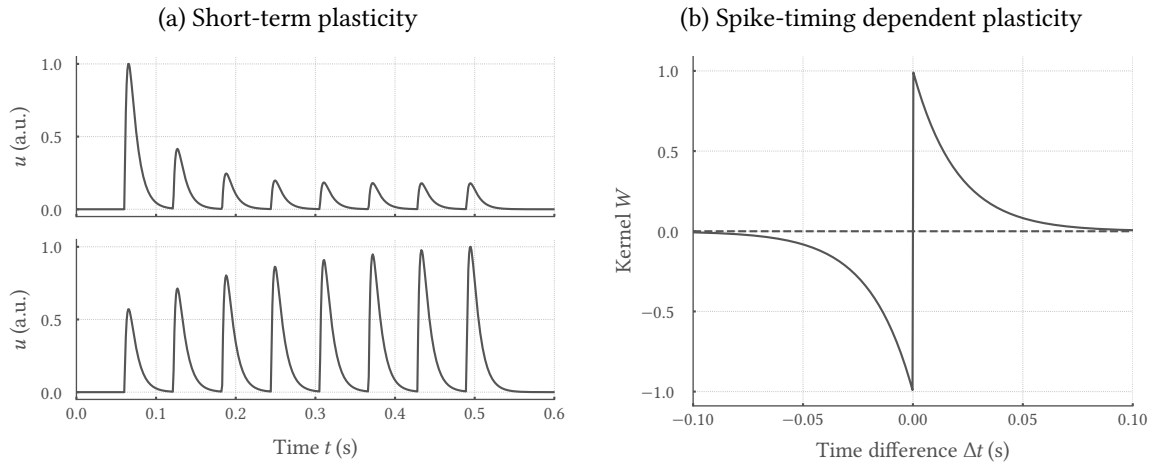


Figure 2.7: Illustration of synaptic plasticity mechanisms. (a) Short-term plasticity affects the amplitude of PSPs depending on previous activation. Shown are modelled membrane potentials for a neuron stimulated by an excitatory synapse for both depression (top) and facilitation (bottom) (Fuhrmann et al., 2002). **(b)** Spike-timing dependent plasticity is a mechanism which changes the synaptic weights based on the relative timing of pre- and postsynaptic spikes. Here, anticausal spike pairing ($\Delta t < 0$) cause depression, whereas causal ones ($\Delta t > 0$) lead to a potentiation of the respective synaptic weight which is often modeled by an exponential kernel (Bi & Poo, 1998). The time constants are chosen to be $\tau_{\pm} = 20\,\text{ms}$ and the amplitudes $A_{\pm} = 1$.

is changed by presynaptic activity (Figure 2.7a): For the facilitating mode the release probability is increased upon presynaptic activation whereas it is decreased for the depressing mode (Markram et al., 1998; Tsodyks et al., 1998; Fuhrmann et al., 2002). Since STP has a direct impact on the synaptic efficacy it alters neural information transmission, especially it has been reported to cause temporal filtering and gain control (Abbott et al., 1997; Tsodyks et al., 1998; Fuhrmann et al., 2002).

Aside from these short-term changes, there are also long-lasting activity-dependent modifications of synaptic weights w_{ij} induced by experience which are assumed to be crucial for learning. Already Santiago Ramon y Cajal advanced the idea of associative memories which are formed by these synaptic modifications in the brain (Cajal et al., 1893). This idea was put forward by Donal Hebb in the late 1940s, resulting in Hebbs's famous postulate which inspired many subsequent experiments (Hebb, 1949). Based on theoretical considerations, this postulate makes a statement about the modification of the connection between a presynaptic neuron A and a postsynaptic neuron B:

> "When an axon of cell A is near enough to excite cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased." (Hebb, 1949)

In the context of spiking neural networks (SNNs), this postulate is often rephrased in terms of temporal correlation between pre- and postsynaptic activity.

Experiments have demonstrated that synaptic changes are indeed driven by pre- and postsynaptic activity. Specifically, the amplitude and direction of these changes depend on the relative timing of presynaptic spike arrival and postsynaptic response. In paring experiments with cultured hippocampal neurons, the dependence of synaptic modifications on the exact presynaptic and postsynaptic spike times, t_j^k and t_i^k, has been examined. Most notably, the change of synaptic efficacies Δw_{ij} crucially depends on the spike time differences $\Delta t = t_i^k - t_j^k$ and is therefore termed spike-timing dependent plasticity (STDP) (Figure 2.7). In more detail, a synapse is strengthened in case spike

pairings appear to be causal ($\Delta t > 0$) and weakened for anticausal pairings ($\Delta t < 0$) (Bi & Poo, 1998; Markram et al., 1997). These findings are indeed a manifestation of the Hebbian postulate as presynaptic neurons which were activated slightly before the postsynaptic neuron emits an AP are those which "take part in firing it". STDP is likely to bio-chemically rely on N-methyl-D-aspartic acid (NMDA) receptors which act as coincidence detectors (Bliss & Collingridge, 1993). Similar to the discussed receptors involved in the generation of PSPs, these NMDA receptors lead to the opening of ion channels in the postsynaptic membrane. However, NMDA controlled channels are additionally blocked by magnesium ions which are only released in case of sufficient postsynaptic depolarization. Only if both factors are present, the channels open and cause Ca^{2+} influx into the postsynaptic terminal which in turn triggers a complex biochemical reaction chain ultimately inducing a change of the synaptic strength (Nevian & Sakmann, 2006).

Albeit detailed bio-physical models of STDP are available, we consider a phenomenological approach (Shouval et al., 2001). In that process, the total weight change Δw_{ij} induced by STDP is often modeled by weighting the spike time difference Δt according to a kernel and an accompanied summation over all pre- and postsynaptic spike pairs:

$$\Delta w_{ij} = \sum_{t_i^k, t_j^k} W(t_i^k, t_j^k)\,. \tag{2.18}$$

A common choice for the kernel function $W(t_i^k, t_j^k)$ is an exponential:

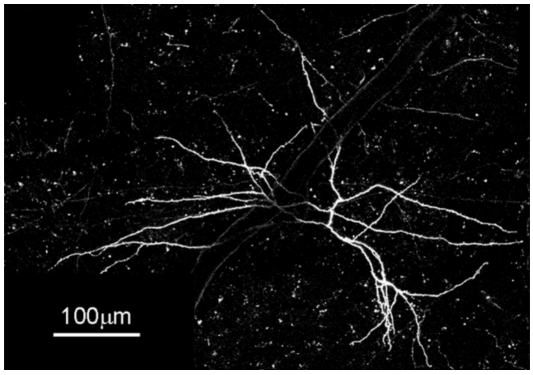
$$\Delta W(t_j^k, t_i^k) = \begin{cases} A_+ \exp\left(\frac{t_j^k - t_i^k}{\tau_+}\right) & \text{for } t_j^k - t_i^k < 0\,, \\ A_- \exp\left(\frac{t_i^k - t_j^k}{\tau_-}\right) & \text{for } t_j^k - t_i^k > 0\,, \end{cases} \tag{2.19}$$

with causal amplitude A_+ and time constant τ_+, as well as anticausal amplitude A_- and time constant τ_- (Figure 2.7b). It is noteworthy that there are also exceptions contradicting this simplified model: Synapses between parallel fibers and Purkinje-cells show the opposite dependence on spike timing and therefore obey an anti-Hebbian form of plasticity (Bell et al., 1997).

The increase in synaptic strength in response to activity represents a positive feedback process: The activity inducing synaptic changes is reinforced by Hebbian plasticity, which in turn causes more activity and further modification. Hence, neural networks subject to activity-dependent plasticity would be driven to epileptic or silent behavior in the absence of additional stabilizing mechanisms. In biology, synaptic scaling – a form of homeostatic plasticity – counteracts the aforementioned processes by globally affecting the transmission through all afferent synapses of a given neuron (Turrigiano & Nelson, 2004). This process occurs in case the network activity dramatically deviates for prolonged periods of time: Reduced activity levels promote an increase in the strength of excitatory synapses onto excitatory neurons, whereas increased activity reduces the overall strength of all excitatory synapses (Turrigiano et al., 1998). Most notably, the relative strength of individual synapses is maintained even though the total synaptic input is altered.

Plasticity is not only limited to the change of synaptic weights. Experiments suggest that there is a continuous rewiring of synapses in the cortex (Lamprecht & LeDoux, 2004; Trachtenberg et al., 2002; Yasumatsu et al., 2008; Holtmaat & Svoboda, 2009; Loewenstein et al., 2011). These mechanisms are centered around dendritic spines on which most excitatory synapses in the mammalian neocortex project onto (Yuste, 2011). Although most of the dendritic spines are maintained over long periods of time (Grutzendler et al., 2002), a small fraction of them is likely to disappear (Loewenstein et al.,

(a) Apical dendrite



(b) Tip of Dendrite

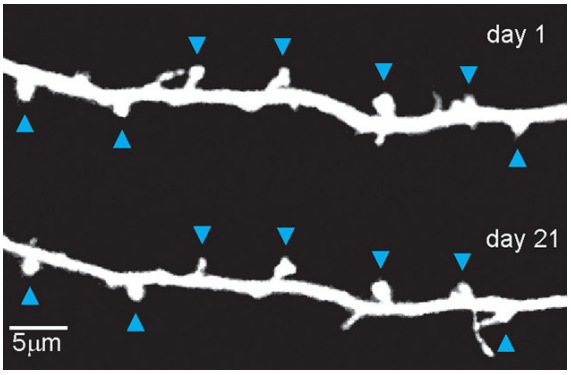


Figure 2.8: Biological neural networks are prone to structural reconfigurations. (a) Two-photon laser scanning microscopy allows to monitor dendritic spines over time spans of weeks. Shown is a maximum intensity projection of an apical dendrite of a neuron in the auditory cortex of a mouse. **(b)** Spine sizes exhibit substantial volatility at time scales ranging from days to months. The upper part displays the tip of a dendrite at the first imaging day, whereas the bottom panel visualizes the very same dendrite on the last imaging day. The spines being present at both measurement days are highlighted by blue triangles. Figure taken from Loewenstein et al. (2011).

2015). Specifically, it has been observed that small spines are removed on time scales from seconds to weeks and new spines are formed (Holtmaat et al., 2005, 2006; Yasumatsu et al., 2008; Holtmaat & Svoboda, 2009). From a modeling perspective, this means that synapses with small weights are more likely to be pruned since the spine head size correlates with the amplitude of an elicited PSP (Matsuzaki et al., 2001). Further, cortical spines show remarkable dynamics after lesioning of input pathways (Trachtenberg et al., 2002; Holtmaat et al., 2006). In particular, after the trimming of whiskers, the fraction of newly formed spines in the somatosensory cortex that become persistent increases strongly (Holtmaat et al., 2006).

Besides the presented plasticity mechanisms, there are also higher-order forms which are referred to as metaplasticity. This metaplasticity can be understood as the *plasticity of plasticity* (Abraham & Bear, 1996; Bear et al., 1987). One example is the shift of the threshold for which long-term potentiation and depression occurs mediated by previous activity (Huang et al., 1992; Wang & Wagner, 1999). In general, metaplasticity is a regulator of plasticity thresholds and plays a key role in keeping the expression of plasticity functional.

Within this chapter, we gave a glimpse of the diverse mechanisms summarized under the term plasticity. Especially synaptic plasticity is assumed to be a basic mechanism underlying learning and memory. More coarsely, learning mechanisms can be divided into three categories: For *unsupervised learning*, the elicited response solely depends on a combination of the stimulus and the structure and/or dynamics of the underlying network. In these scenarios, the input as well as the learning rule lead to self-organization of the network, as in the case of STDP. In contrast, in *supervised learning*, a network is trained according to an input to output mapping imposed by a supervisor. This learning scheme is met by most machine learning applications (Goodfellow et al., 2016b). Last, there is an intermediate form of the aforementioned scenarios called *reinforcement learning* (Sutton & Barto, 2018). Here, the network only obtains feedback about its current performance during the process of training which is provided in form of a reward signal.

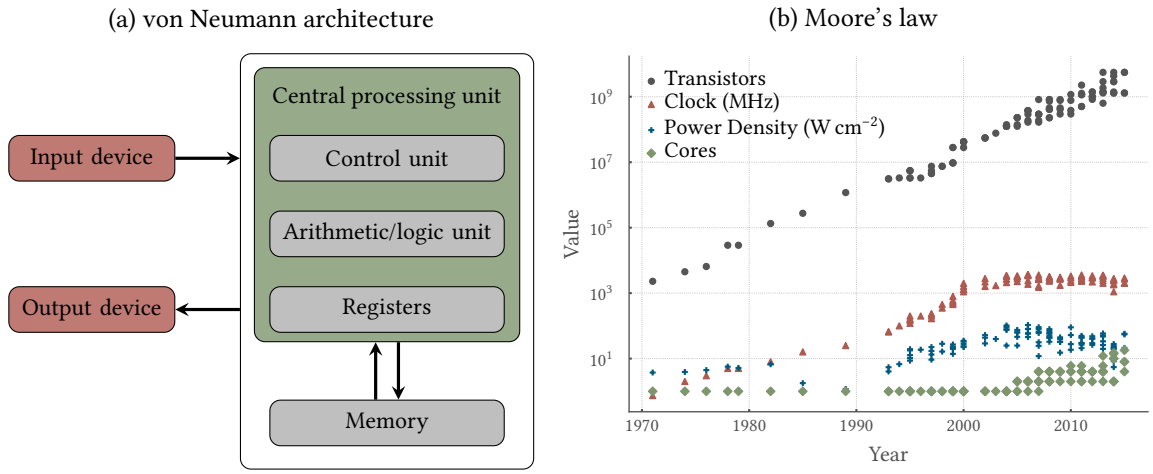


Figure 2.9: Illustration of the von Neumann architecture and Moore's law. **(a)** The von Neumann architecture is a computer architecture based on the stored-program concept; a single shared memory holds both the program as well as the instruction data. The central processing unit executes the instructions of a computer program. It contains a control unit, a arithmetic and logic unit and registers. Further, the central processing unit is coupled to a memory unit as well as in- and output devices. **(b)** The transistor count of microprocessor almost doubles every two years (Moore et al., 1965). Shown are key processor features as well as their dates of introduction. Data taken from Wallentowitz (2021).

2.2 Neuromorphic Systems

In the previous section, we have introduced the basic working principles of spiking neurons and networks thereof. Despite this detailed knowledge, simulating large-scale networks of spiking neurons remains a major challenge. To understand the associated hurdles that prevent scaling up neural networks on conventional hardware, we need to take a look at the von Neumann architecture. Thereby, we will naturally motivate the ideas of neuromorphic engineering.

2.2.1 Von Neumann architecture

Most of the contemporary computers are based on the *von Neumann architecture* (Von Neumann, 1993). This electrical digital computer architecture consists of a control unit (CU), an arithmetic and logic unit (ALU), registers, a memory unit, and in- as well as outputs (Figure 2.9a). The ALU, CU and a set of register make up the central processing unit (CPU) which constitutes the central element for the execution of the instructions of a computer program. The accompanying arithmetic as well as logic operations are executed by the ALU, whereas the CU takes care of the operation of the ALU, memory as well as input and output devices. In more detail, the CU controls the response to program instructions fetched and interpreted from the memory unit. Further, the timing and control signals required by other components are supplied by the CU. High-speed storage is provided in form of registers which have to hold all data before the actual processing. This architecture composed of a centralized CU with memory access is what is typically referred to as von Neumann architecture, albeit several variations and extensions have been proposed and implemented in the past.

The von Neumann architecture constitutes a powerful design that is only limited by the amount of memory and the available computation time. In particular, the shared bus between program and data memory creates a bottleneck, usually termed the *von Neumann bottleneck*. This shared

bus prevents an instruction fetch and an arbitrary operation at the same time, thereby limiting the theoretical throughput of the CPU. Especially for minimal processing of massively parallel data, this problem is aggravated and limits the effective processing speed. Here, the CPU has to continually wait for required data to move from or to the memory.

Most of the advancement in CPU development is due to the progress in semiconductor technologies. The amount of memory as well as the CPU size and speed have increased in recent years (Figure 2.9b). Especially, Moore's law still holds by doubling the number of transistors on a chip every two years (Moore et al., 1965). However, the speed of CPUs increases at a rate $60\,\%\,\mathrm{yr}^{-1}$ whereas the speed of dynamic random-access memory (DRAM) increases with only $10\,\%\,\mathrm{yr}^{-1}$ (Patterson & Hennessy, 2016). This in turn exacerbates the effect of the von Neumann bottleneck as in typical programs 20 % to 40 % of the instructions reference memory (Patterson & Hennessy, 2016). This becomes a severe problem for massively parallel processing like the simulation of large-scale bio-inspired neural networks.

Nevertheless, neuro-synaptic dynamics can be modeled on digital computers based on the von Neumann architecture. The equation presented Sections 2.1.2 and 2.1.3 that model the dynamics of neurons and synapses can be solved by numerical methods: Large networks of spiking neurons can be simulated by discretizing time and iterative update schemes of all state variables (Press et al., 2007). Recently, efforts have been made to perform large-scale simulations on conventional computing systems which are based on the von Neumann architecture. Some of these implementations were centered around a highly connected cortical microcolumn (Potjans & Diesmann, 2014). Van Albada et al. (2018) performed an optimized simulation of this microcolumn on a supercomputer. This simulation encompassed about 80000 neurons and 0.3 billion synapses, corresponding to only 0.0001 % of the entire brain. The authors reported a slowdown of 4.6 and a minimum required energy of 5.8 µJ per synaptic event. If one would linearly scale this result up to the size of a brain, this would lead to a total power consumption of 58 GW. Among other factors, this inefficiency in terms of energy and simulation speed is caused by the von Neumann bottleneck: For a single state update, the required data – like e. g. the synaptic weights – needs to be fetched from the memory prior to the actual computations. In contrast, the physical substrate in the brain that implements the memory – like the synaptic weight – directly performs the computation. This difference between the von Neumann architecture and the design of the brain significantly affects the performance of simulations of the latter.

Different changes and extensions to the von Neumann architecture have been proposed to alleviate the von Neumann bottleneck for neural network simulations in particular and to improve the parallel processing capabilities in general. Among others, strategies like caching, prefetching, multithreading, new types of random access memory and near-memory computing with a processor mingled with memory on a single chip have been developed and applied in the past (Patterson & Hennessy, 2016). An existing and pervasive class of computing devices developed for parallel processing are graphics processing units (GPUs). Originally developed for the efficient handling of graphic data, they are suitable for parallel computing in general due to their enormous amount of parallelly distributed processing cores and their associated parallel arithmetic (Brodtkorb et al., 2013). For these devices, the complexity of a single processing unit has been traded against the number of available units which renders them quite successful for a broad range of applications. Therefore it is obvious that the model of the cortical column has also been simulated on GPUs. Knight & Nowotny (2018) demonstrated that GPUs performed substantially better in terms of both energy efficiency as well as simulation speed. The best results were achieved on an Nvidia Jetson

TX2 card which only consumed 0.3 µJ per synaptic event at a slowdown of 25 compared to biological real-time. For an Nvidia Tesla V100 accelerator speedups approaching a factor of two have been reported. Despite these successes, both approaches are still orders of magnitude away from the 0.2 fJ per synaptic operation as required by the brain, thereby increasing the demand for novel computing architectures.

2.2.2 Neuromorpic hardware

Biological neural networks operate quite differently compared to conventional computers obeying the von Neumann architecture introduced in the previous section. Most notably the parallelism is deeply rooted in the structure of brains; neurons and synapses process information independent of each other. This is enabled by the asynchronous processing of information in the brain. In particular, there is no global clock and new information gets immediately processed at the time of arrival. Apart from this, there is no separation between memory and computation, instead, both are combined and distributed over the entire network. The physical substrate that implements the memory – like the synaptic weight – directly performs the computation. These desirable properties point to the beneficial role of novel computing architectures that mimic aspects of the brain.

The associated *neuromorphic computing* has emerged in recent years and was coined by Mead (1990). Originally, the term neuromorphic hardware referred to very-large-scale integration (VLSI) systems with analog electrical components that mimic parts of biological neural networks. Nowadays, implementations that are based on biologically-inspired or artificial neural networks as well as implementations on non-von Neumann architectures are also referred to as neuromorphic. However, these neuromorphic implementations share dense connectivity, parallel processing, low-power consumption and co-located memory and processing (Schuman et al., 2017). In the following, we motivate the potential benefits of a neuro-inspired computing device compared to purely analog or digital solutions based on theoretical arguments. This consideration follows the arguments of Boahen (2017).

In contrast to digital computers, the communication and computation in the brain are prone to errors (Allen & Stevens, 1994). To finally understand why this approach is favorable in terms of energy efficiency, let us consider a collection of communications channels and their associated information content. Each of these channels conveys a certain number of information bits b. According to the Shannon-Hartley theorem b depends on the signal energy E:

$$b = \frac{1}{2} \log_2 \left(1 + \frac{E}{k_\mathrm{B} T} \right) , \tag{2.20}$$

for a channel with additive white Gaussian noise with an energy of $k_\mathrm{B}T$ (Shannon, 1948; Hartley, 1928). A theoretical maximum of b with respect to E is reached for $E = (e-1)k_\mathrm{B}T$ with $b_\mathrm{max} = {}^{1}/_{\log(4)e}$. Most notably, the information decreases only logarithmically with the signal energy. In contrast, the number of signals transmitted per second rises linearly with the bandwidth B. Hence, the total channel capacity is given by $C = Bb$. Therefore, using many error-prone, low-energy channels is more energy-efficient – albeit taking up more space – than using a few pristine, high-energy channels (Figure 2.10a). In summary, communication efficiency can be maximized by trading most of the signal-to-noise ratio for energy efficiency.

Communication errors are highly probable in the regime of maximum energy efficiency, i. e. the working point of the brain (Figure 2.10a). For a binary alphabet, the error probability caused by

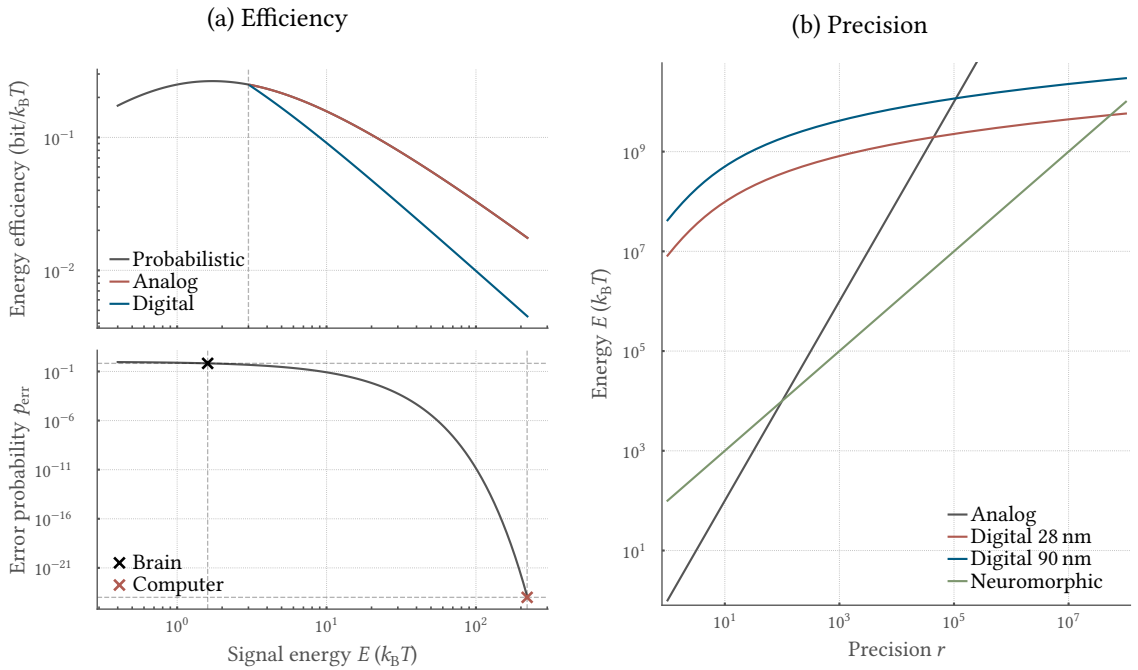


Figure 2.10: Efficient combination of analog and digital computation. The computation carried out by human brains relies on a combination of digital and analog processing. **(a)** Analog computing is characterized by a high energy efficiency $^{b}/_{(E+k_B T)}$. A signal with energy E transmits $b = {}^{1}/_{2} \log_2 (1 + {}^{E}/_{k_B T})$ bits of information (top). For $E > 3k_B T$ it conveys more than 1 bit of information. In case of a digital communication, b is clipped to 1 bit. The error probability p_{err}, however, exponentially depends on E (bottom). Conventional computers therefore operate with high energy signals to minimize the error probability, whereas the brain operates in the converse regime. **(b)** Computations with intermediate precision profit from neuromorphic solutions. To low-pass filter a signal, the energy demand depends quadratically, logarithmically and linearly on the precision r for analog, digital and neuromorphic solutions. Hence, an analog implementation is most energy-efficient for low precision, whereas digital multiply-accumulate units take over for high precision. In between, neuromorphic implementations seem to be beneficial for which the neuron count is assumed to scale linearly with the precision. Figure based on Boahen (2017).

thermal fluctuation is given by $\exp(-{}^{E}/_{4k_b T})$. To be functional, digital computers require low error rates on the order of $p_{err} = 10^{-24}$ corresponding to a signal energy of $220 k_B T$. Because of this, the computer's communication energy efficiency is 60 times lower than the theoretical maximum achieved for $E = (e-1)k_B T$. In direct contrast, the brain operations in the error-prone regime of the theoretical optimum: The transmission at synaptic contacts is unreliable, i. e. in two-thirds of the cases a spike fails to trigger ligand release at the terminals (Allen & Stevens, 1994).

The combination of digital and analog communication and computation of neurons is characterized by a low energetic footprint for intermediate precisions (Boahen, 2017). Let us consider a low-pass filter similar to the sub-threshold behavior of neurons. Here, the precision is given by $r = \sqrt{{}^{E}/_{k_B T}}$ with E being the energy to cycle from the minimum to the maximum signal value and back (Figure 2.10b). The implementation in a digital fashion requires a multiply-accumulate unit that grows quadratically with the number of bits. Here, a full cycle is divided into m equal steps. Hence, the associated energy is given by $E = mb^2 E_{mac}$ with the energy E_{mac} for a single multiply-accumulate operation per bit. Substituting $b = \log_2 (r+1)$ yields $E = m \left(\log_2 (r+1)\right)^2 E_{mac}$ (Figure 2.10b). Because of this logarithmic scaling, digital solutions consume less energy at high precision, whereas analog implementations are beneficial for low precision. With advances in semiconductor technol-

ogy during the last years, the intersection has moved to lower precision values. For example, an actual 28 nm process consumes about 2.1 pJ, whereas an older 90 nm process requires 10.9 pJ for $m = 50$ and $r = 255$ (Reyserhove et al., 2014). If the subthreshold dynamics of neurons can now be implemented with analog circuits and the asynchronous communication between them by digital logic instead, the precision can be assumed to be linearly dependent on the neuron count (Figure 2.10b). As a result, this neuromorphic implementation has the potential to outperform purely digital as well as purely analog realizations in an intermediate precision regime.

So far, the theoretical consideration discussed above does not take a specific hardware implementation into account. In general, there are many different strategies to approach the field of neuromorphic computing which resulted in vastly different computing substrates in the past years (Schürmann et al., 2005). In the following, we will present a selection of contemporary hardware devices as well as their potential field of use.

2.2.3 Neuromorpic designs

Given the broad definition of neuromorphic systems, it is not surprising that a variety of devices has been developed in the past (Schuman et al., 2017). These application-specific integrated circuits (ASICs) are made up of a vast range of basic building elements. Among others, optical (Brunner et al., 2016; Maier et al., 1999; Paquot et al., 2012) as well as magnetic (Roy et al., 2014a; Grollier et al., 2016) components have been visited previously. Moreover, recent effort in material science promoted the development of memristive components (Roy et al., 2019; Marković et al., 2020; Joshi et al., 2020; Dalgaty et al., 2021). Here, we limit our consideration to electrical devices implemented in a standard complementary metal oxid semiconductor (CMOS) technology which relies on metal oxide semiconductor field-effect transistors (MOSFETs). This technology is also used for the implementation of conventional computers. On a higher level, all implementations of neuromorphic devices can be subdivided into three major categories: digital, analog and mixed-signal devices. In the following, we will discuss some representatives of each category and highlight their basic working principles.

Digital systems, rely on boolean logic-based gates to perform computations (Harris & Harris, 2013). For these digital systems, the voltage across the gate of the underlying MOSFETs is almost always driven between its minimum and maximum value. By this, the transistor is either fully conducting or fully closed, thereby mostly suppressing leakage currents as well as device-specific variations. Since the energy required to charge the gate and make the substrate conductive scales quadratically with the applied voltage, the energy efficiency of digital CMOS devices also decreases with the squared applied voltage. However, the effect of thermal noise is minimized by operating at the binary states of a closed or open gate. Like in conventional computers, this digital logic can be used to calculate the neuro-synaptic dynamics. In contrast to conventional computers, they often use dedicated accelerators to speed up the simulation of specific model components. Examples of digital systems are IBM's *TrueNorth* chip (Akopyan et al., 2015), the *SpiNNaker* system (Furber et al., 2014), and Intel's *Loihi* chip (Davies et al., 2018). These exemplary digital systems already show the extent to which the design choices being pursued differ in terms of flexibility. The SpiNNaker system relies on many small integer cores and a custom interconnection, optimized for the communication of spike-events. Hence, it provides a high degree of flexibility in terms of the implementable neuron, synapse as well as plasticity models. In contrast, the TrueNorth chip implements LIF neurons – i. e. a fixed neuron model – in combination with a programmable network connectivity, but

without support for on-chip plasticity. This reduced flexibility, however, provides a high energy efficiency: While TrueNorth is reported to require on average 25 pJ to pass one spike through one static synapse, the SpiNNaker system consumes 10 nJ for the same operation (Furber, 2016). This already gives a glimpse towards the trade-off of flexibility and energy efficiency in contemporary neuromorphic solutions.

For analog devices, flexibility is traded for energy efficiency. The transistors of *analog systems* are not bound to the two extrema of digital systems and instead utilize the physical characteristics of these electronic devices to emulate a certain behavior. In more detail, the transistors can be operated in arbitrary configurations, only limited by the signal-to-noise ratio as described in the previous section (Sansen, 2007). This facilitates the representation of state variables as actual physical observables within the system in direct accordance with biological neural networks. As a result, these devices emulate fixed neuron, synapse as well as plasticity models with their analog circuits. In so-called *mixed-signal devices*, these analog circuits are accompanied by digital logic implementing the spike traffic. These ASICs capture the digital behavior of spikes as well as the analog nature of synapses and neurons in the sub-threshold regime of the membrane potential. With analog and mixed-signal systems being most relevant for this thesis, we wish to focus on them in the following.

Depending on the operating point of the used transistors, analog neuromorphic systems could be divided further into sub- and supra-threshold devices. In the sub-threshold region, the gate voltage is below the threshold voltage required to activate the source-drain channel. In this regime, the drain-current exponentially depends on the difference of the gate-source voltage and the threshold voltage (Gray et al., 2001). This behavior lets the system operate at extremely low currents ensuring high energy efficiency. Systems build to work in this regime have been reported to require only on the order of pJ per spike (Livi & Indiveri, 2009). At the same time, the low currents allow these systems to operate in real-time, opening a wide range of applications by the possibility to directly interface with neuromorphic sensors (Corradi & Indiveri, 2015; Ros et al., 2015; Corradi et al., 2014; Chicca et al., 2014; Rongala et al., 2015). On the downside, the drain current also depends exponentially on the temperature of the environment (Gray et al., 2001). Therefore, the temperature impacts the behavior as much as the signal itself in the absence of explicit control mechanisms. In contrast, this dependence is much less pronounced for devices exploiting the ohmic or even the saturation region.

Transistors operating above the threshold voltage cannot achieve low drain currents. As a result, the associated circuit dynamics are faster, since these currents are directly proportional to the voltage changes across the capacitors. One way to exploit the dynamics in the supra-threshold regime is to design all circuits such that all time constants are scaled down by a constant *acceleration factor*. For neuromorphic implementations, this means that the parameter defining the time scale of the underlying model dynamics like the membrane and synaptic time constants are scaled by this constant factor. Hence, the relevant time scales are compressed while the relative dynamics are preserved.

The impact of these accelerated dynamics is twofold. Technically, the interfacing of accelerated systems with external hardware can be difficult. In particular, the input and output data bandwidth increases linearly with the acceleration factor. At the same time, the tolerated latency decreases linearly. Further, most of the conventional hardware is build for human interaction and is therefore not directly applicable. These technical problems, however, can, in general, be mitigated by additional engineering effort (Schreiber, 2000).

The acceleration opens up new application areas. Most notably, the human-relevant time scales of our environment coincide with the time scales of the dynamics of the brain (Kiebel et al., 2008). By

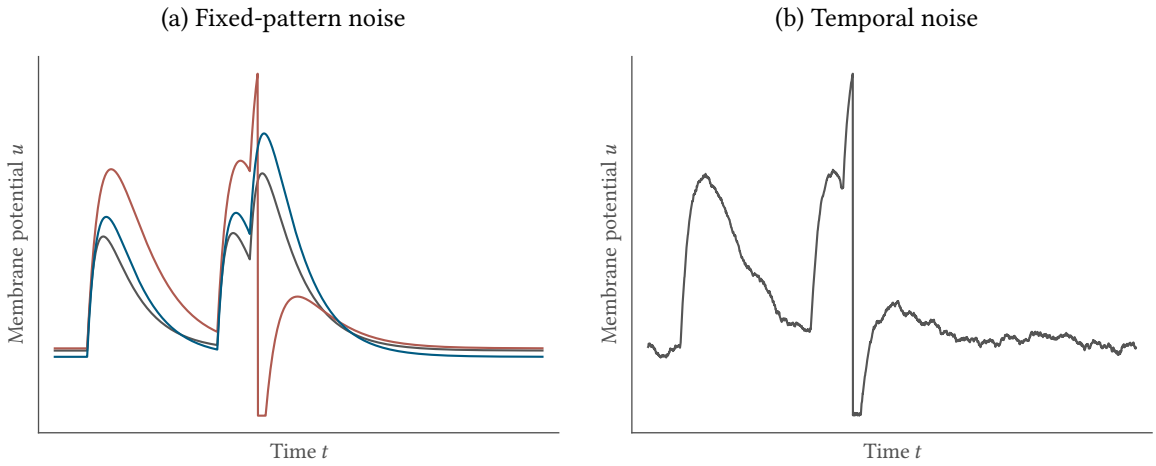


Figure 2.11: Impact of noise on the emulation of neuro-synaptic dynamics on analog neuromorphic hardware. Due to the analog implementation of neurons and synapses, their dynamics are subject to noise. Here, we distinguish between two sources of noise. **(a)** Circuit-to-circuit variability induced by imperfections in the production process results in parameter deviations of the emulated models. This fixed-pattern noise can be mitigated by dedicated calibration routines. **(b)** Temporal noise affects time scales relevant for emulation by e. g. thermal noise. Effects are exemplarily shown for LIF neuron dynamics.

accelerating the dynamics of a neuromorphic device we, therefore, shift the potential use cases by precluding a direct interaction with the environment on "human-perceptible" time scales. Especially medical applications and in general brain-machine interfaces involving human interactions are often precluded (Corradi & Indiveri, 2015; Ros et al., 2015; Corradi et al., 2014; Chicca et al., 2014; Rongala et al., 2015). However, the speedup does not only preclude applications, but also enables new ones. While for example, real-time applications with audio signals covering a frequency range of 10 Hz to 20 000 kHz (Rossing, 2007) is not possible, the speedup directly translates to high-frequency signals. Depending on the actual acceleration factor even applications in the radio frequency range are possible (Beasley, 2010). Moreover, if no real-time processing of inputs is required, the systems could either run on hold or can just be used to evaluate pre-recorded data.

The acceleration promotes experiments with high throughput. In particular, the development of the nervous system and its involved plasticity mechanisms cover vast time scales. For example, the changes induced by structural plasticity occur on time scales ranging from days to months (Loewenstein et al., 2011). Here, the acceleration factor comes to rescue by emulating the underlying network dynamics in minutes to hours assuming an acceleration factor of 1000. On a larger scale, the development of a human brain starts in the third gestational week and extends at least through late adolescence (Stiles & Jernigan, 2010), thereby covering multiple decades. Again, these scales translate to months and are therefore in general feasible to study when drawing on accelerated systems. High throughput is also desirable for any hyperparameter optimization by dedicated learning algorithms (Thrun & Pratt, 2012), evolution strategies (Hansen et al., 2015) or parameter sweeps. Last but not least, large amounts of statistical data could be gathered in relatively short timespans. Hence, the accelerated emulation of neuro-synaptic dynamics renders these devices a promising tool for neuro-scientific simulations.

Analog systems come with the promise to process information faster and more energy-efficient compared to their digital counterparts. However, these potential advantages come with drawbacks as analog circuits are prone to noise. Here, we would like to distinguish between two sources of

noise. First, there are substantial circuit-to-circuit deviations induced by variations in the manufacturing process which we would like to term *fixed-pattern noise* (Shyu et al., 1984). While digital systems are also subject to this form of variation, its effect is usually less pronounced for these systems due to the binary operating mode as well as the established error-correcting protocols. For analog neuromorphic circuits, in contrast, fixed-pattern noise results in parameters deviations of the emulated model for every circuit (Figure 2.11a). These variations can, however, be mitigated by dedicated calibration routines, resulting in a mapping from configured parameters to actual behavior (Neftci & Indiveri, 2010; Neftci et al., 2011a; Brüderle et al., 2011). Second, there is *temporal noise* which affects time scales relevant for emulation (Figure 2.11b), e. g. thermal noise (Wunderlich et al., 2019; Cramer et al., 2020a; Pfeil et al., 2013).

The BrainScaleS architecture is an example of an accelerated approach to the design of neuromorphic hardware (Schemmel et al., 2010; Friedmann et al., 2016; Schemmel et al., 2020). In the past years, it has been applied to a variety of problems, most of them explicitly exploiting its intrinsic acceleration. First of all, in-the-loop (ITL) learning has been utilized to port deep artificial neural networks to the wafer-scale BrainScaleS-1 system (Schmitt et al., 2017). In recent work, the extensions for vector-matrix multiplication and accumulation of BrainScaleS-2 have been used to implement artificial neural networks (ANNs) (Weis et al., 2020). Directly exploiting the spiking mode, neural sampling – a spike-based implementation of Bayesian computing – has been demonstrated on all versions of BrainScaleS on a variety of tasks (Kungl et al., 2019; Billaudelle et al., 2020; Czischek et al., 2020). Moreover, multi-layer SNNs employing a time-to-first-spike coding scheme have been trained on BrainScaleS-2 (Göltz, 2019). In addition, surrogate gradient learning has been applied to train multi-layer as well as recurrent SNNs on several benchmark tasks (Cramer et al., 2021). Likewise, a neuromorphic agent has been trained with a learning-to-learn framework in a maze runner task which directly exploiting the hybrid plasticity approach (Bohnstingl et al., 2019). Using the flexible plasticity of BrainScaleS-2, SNNs have been optimized to task complexity by exploiting collective dynamics (Cramer et al., 2020a). Also exploiting the hybrid plasticity of BrainScaleS-2, reinforcement learning has been applied to train a virtual player in the game of Pong (Wunderlich et al., 2019). Finally, limitations in the number of synaptic resources have been surmounted by structural reconfigurations on BrainScaleS-2 (Billaudelle et al., 2021). In the following chapter, we highlight the basic components of the BrainScaleS-2 system facilitating this broad range of applications.

3 BrainScaleS-2

The spiking neural networks (SNNs) considered within the scope of this thesis were not only optimized for information processing, but likewise, their energetic footprint was minimized. To that end, the networks were emulated on two different mixed-signal neuromorphic chips of the BrainScaleS-2 family. The BrainScaleS-2 architecture is centered around an analog neural network core implementing neuron and synapse models in electrical circuits that emulate the dynamics of their biological archetypes in continuous time (Sections 3.1.1, 3.1.2, 3.2.1 and 3.2.2). Consequently, the state variables like the membrane potentials and the synaptic currents are physically represented in the respective circuits. Compared to biological measurements, all time constants of neurons and synapses are accelerated by a factor of 1000, determined by the intrinsic capacitances and conductances of the semiconductor technology. This acceleration facilitates the emulation of time-consuming experiments, such as the investigation of learning on various time scales, the performing high-dimensional parameter sweeps, or the gathering of large volumes of data in statistical experiments (Cramer et al., 2020a; Bohnstingl et al., 2019; Billaudelle et al., 2021). Facilitated by the 65 nm process, BrainScaleS-2 features a freely programmable on-chip processor (Sections 3.1.3 and 3.2.3). This processor is specialized towards the implementation of plasticity rules and together with the named speedup allows to fully exploit the benefits of neuromorphic computing in terms of energy efficiency and execution speed. Although the ultimate target of the BrainScaleS-2 architecture is wafer-scale integration, smaller versions based on single application-specific integrated circuits (ASICs) are needed to develop and debug the final design. The experiments described within this thesis were conducted on two different chips within this development. Early experiments were implemented on the small prototype device High Input Count Analog Neural Network with Digital Learning System (HICANN-DLS), whereas later work focused on the full-size ASIC High Input Count Analog Neural Network X (HICANN-X), both of which were used in their respective second revision.

3.1 HICANN-DLS prototype

HICANN-DLS represents a small prototype chip within the BrainScaleS-2 family (Figure 3.1). It covers an area of $1.7 \times 1.7\,\mathrm{mm}^2$ and features 32 neurons and an array of 32×32 synapses. In the following, the analog neuromorphic network core is depicted in Section 3.1.1. It houses the neurons and synapses, the synapse drivers, as well as an analog parameter storage and a 64 channel analog-to-digital converter (ADC). This core is surrounded by a digital backend as well as a digital coprocessor whereof the latter is described in Section 3.1.3. We close by highlighting the event routing in Section 3.1.2 and the basic components of the experimental setup in Section 3.1.4. For further details, the reader is referred to Friedmann et al. (2016), Aamir et al. (2017) and Hock et al. (2013).

(a) Neuromorphic system

(b) Neuromorphic chip

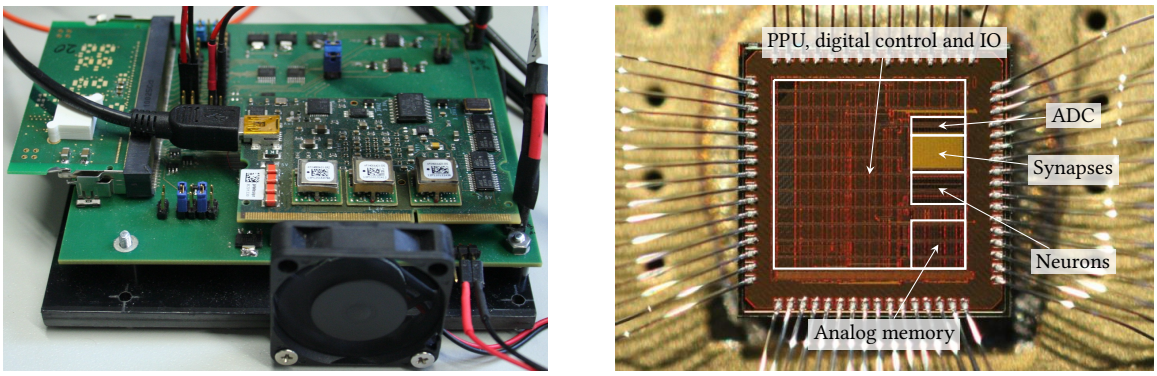


Figure 3.1: Photograph of the BrainScaleS-2 prototype system. (a) The full system is designed as a small tabletop setup. The neuromorphic chip HICANN-DLS (left, below white cover) is mounted on the chip carrier board. The latter is housed together with a FPGA (right foreground) on the baseboard. **(b)** The HICANN-DLS chip features an analog neural network core as well as an on-chip general-purpose processor, the PPU. Shown is a close-up of the neuromorphic chip with highlighted key functional units. Annotations taken from Cramer et al. (2020a). Photographs adapted from Aamir et al. (2017).

3.1.1 Analog neuromorphic network core

Synapse array: The analog network core constitutes the central component of the BrainScaleS-2 architecture (Figure 3.2a). In case of the HICANN-DLS prototype, it features an array of 32 silicon neurons (Aamir et al., 2018) emulating leaky integrate-and-fire (LIF) dynamics according to Equation (2.12). The connectivity is physically implemented by the synapse array: Every neuron receives input from a column of 32 current-based synapses. Events enter this 2D array of synapses from the left via synapse drivers (Figure 3.2a). The synapse drivers forward the digital spike events to a single synaptic row and control the timing by means of a short digital voltage pulse. In more detail, events injected into a single row of the synapse array are tagged with 6 bit addresses denoting their presynaptic origins and are then forwarded to each synapse in the respective row. Likewise, each synapse holds a 6 bit label alongside a 6 bit weight in the synapse-local static random-access memory (SRAM). It allows filtering afferent spike trains by their addresses (Figure 3.2b): Only if the address of an incoming event matches the locally stored one, an address comparator generates a presynaptic enable signal which in turn causes the creation of a current pulse with amplitude proportional to the stored synaptic weight. Hence, the total amount of charge of each pulse linearly depends on both the synaptic weight and the pulse length which is derived from the digital voltage pulse. Two vertical lines – one excitatory and one inhibitory – sum and forward the resulting current pulses to the synaptic inputs of each neuron where the actual conversion to the synaptic current according to Equation (2.17) takes place (Figure 3.3). On BrainScaleS, the sign of a synaptic current is determined by the synapse driver and is, therefore, a row-wise property in the synapse array: Each driver can be configured to be either excitatory or inhibitory.

In addition to the described routing mechanisms, each synapse implements an analog circuit for measuring pairwise correlations between pre- and postsynaptic spike events (Friedmann et al., 2016). These sensors have access to the current pulses generated in each synapse which constitute the presynaptic side. Dedicated signaling lines carry up the postsynaptic digital spike event from the neuron to each column of synapses (Figure 3.3). In total, there are two measurement units per

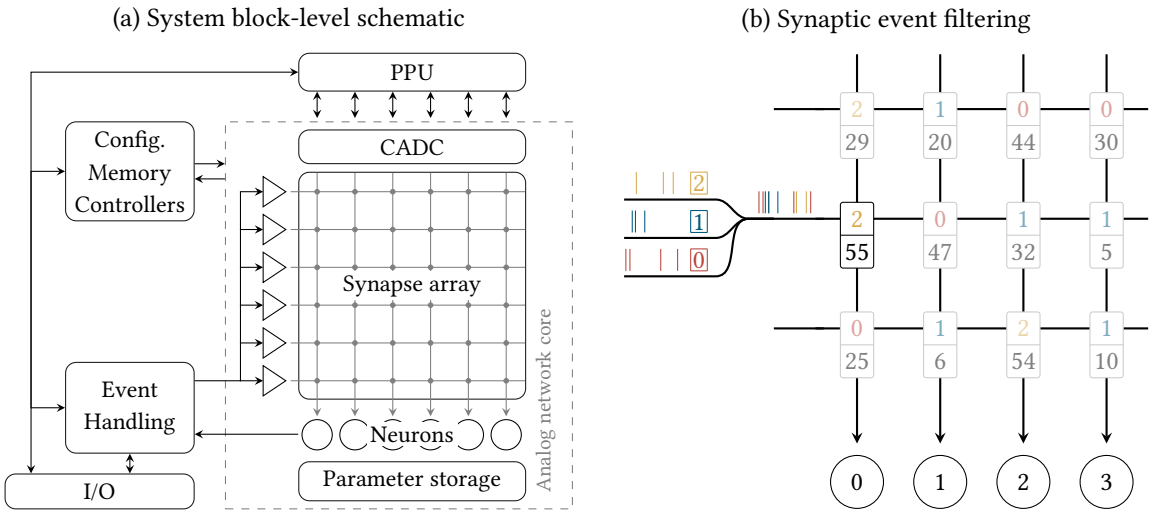


Figure 3.2: Block-level schematic of the BrainScaleS-2 prototype ASIC. (a) The neural and synaptic circuits make up the analog neuromorphic network core which is most notably accompanied by an analog parameter storage and the CADC for digitizing synaptic correlation measurements. Surrounding digital logic allows to interface the full-custom circuits by taking care of configuration data as well as event traffic. The close attachment of analog neural network core and PPU facilitates the processor to access synaptic weights and address labels, digitized data from the CADC as well as configuration data. **(b)** Events injected into a row of the synapse array are identified with an address denoting their source (numbered and marked by color). Spike trains from different origins can be overlayed and injected into a single synapse row. Synapses filter afferent events by comparing the source address to a label stored in their local SRAM. Only in case of matching addresses, a spike is forwarded to the postsynaptic neuron. Figures adapted from Billaudelle et al. (2020, 2021).

synapse, one for causal and one for anticausal correlations between pre- and postsynaptic spike times. The measurements are represented as analog voltages stored on two capacitors and obey the following equations:

$$f_+\left(t_i^k, t_j^l, t, t+T\right) = \sum_{\substack{k \\ t_i^k, t_j^k \in [t, t+T]}} \eta_+ \exp\left(-\frac{t_i^k - \max_l\left[t_j^l < t_i^k\right]}{\tau_+}\right), \tag{3.1}$$

$$f_-\left(t_i^k, t_j^l, t, t+T\right) = \sum_{\substack{k \\ t_i^k, t_j^k \in [t, t+T]}} \eta_- \exp\left(\frac{t_i^k - \max_l\left[t_j^l > t_i^k\right]}{\tau_-}\right), \tag{3.2}$$

where f_+ accounts for causal and f_- for anticausal correlations in the time interval $[t, t+T]$. The parameters $\eta_\pm$ and $\tau_\pm$ correspond to the freely configurable amplitudes and time constants of the respective spike-timing dependent plasticity (STDP)-kernel. Note that the correlation measurements in Equations (3.1) and (3.2) follow a reduced nearest-neighbor pairing rule (Morrison et al., 2008). The availability of correlations measurements on the system allows access to various forms of learning rules based on STDP (cf. Section 2.1.4).

Neurons and synaptic input: The LIF neurons are implemented in silico and feature current-based synaptic inputs (Figure 3.3). The latter emulate the exponential dynamics according to Equation (2.17). To that end, the current pulses relayed by each of the aforementioned columns of 32

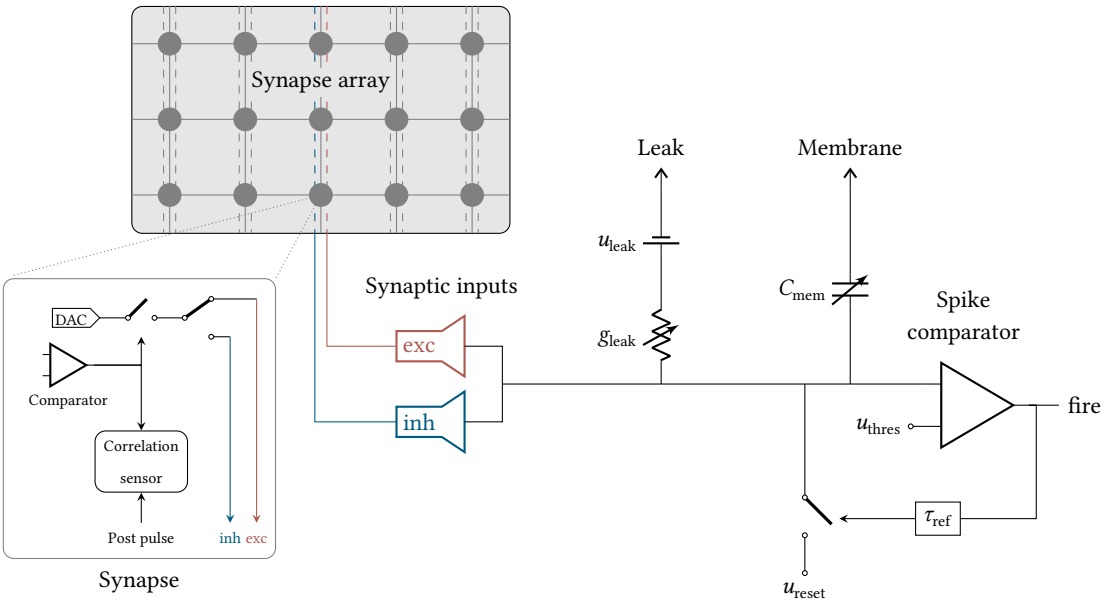


Figure 3.3: Block-level schematic of the HICANN-DLS neuron. Each term in the current-based synapse and LIF neuron model is emulated by an equivalent circuit. The current pulses generated by the synapses (bottom left) are summed along the columns of the synapse array on the excitatory (red) and the inhibitory (blue) line respectively (top left). These signals are finally converted to proportional singed currents by the synaptic inputs. LIF dynamics are emulated by a leak conductance g_{leak} and an associated leak potential u_{leak} as well as a capacitor C_{mem} (top middle). Further, a spike comparator (bottom right) implements the spike condition: In case the membrane potential exceeds a threshold u_{thres}, a fire signal is emitted to the digital backend. Further, the potential is clamped to the reset potential u_{reset} by means of a switch for the duration of the refractory period τ_{ref}.

synapses on the excitatory as well as the inhibitory line are integrated separately onto two RC integrators per column. Each incoming nanosecond-wide current pulse causes a drop in the respective voltage as they discharge the aforementioned capacitor. This decline is proportional to the strength of the incoming current pulse and recovers with the synaptic time constant τ_{syn}, determined by two configurable resistors, one for each line. As a result, we obtain two voltage signals with similar polarity, one excitatory and one inhibitory. Both of which are accordingly transformed into proportional currents by the operational transconductance amplifiers (OTAs) of the synaptic inputs. While signals on the excitatory line lead to positive currents, the input on the inhibitory line causes negative currents on the membrane capacitance C_{mem}.

The leak term in Equation (2.12) is emulated by another OTA. This tunable resistor with conductance g_{leak} is connected to the leak potential u_{leak} and the membrane capacitance C_{mem}. Hence, the membrane time constant τ_{mem} of the emulated neuron is determined by the quotient of C_{mem} and g_{leak}. Despite being digitally configurable, we fixed C_{mem} to its maximum value of 2.3 pF for the experiments conducted within this thesis.

The circuits described so far emulate a leaky integrator with synaptic input (Equation (2.12)). A comparator and an associated switch complete the model by implementing the spike generating mechanism according to Equation (2.13) (Figure 3.3). In case the membrane potential u exceeds the threshold voltages u_{thres}, the comparator triggers a fire signal which marks the actual spike. In turn, this signal is forwarded to the digital backend via outgoing digital lines and in addition, is registered by circuits that count the number of emitted spikes for each neuron with a precision of

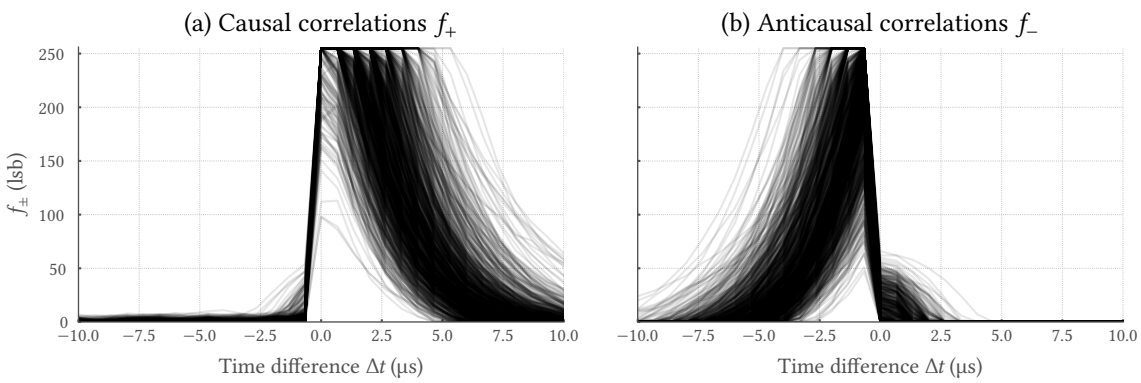
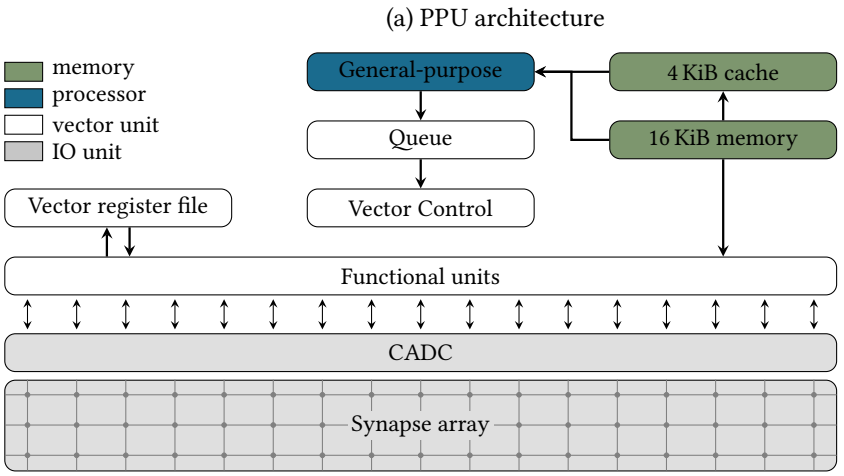


Figure 3.4: HICANN-DLS supports parallel recordings of synapses-local spike-time correlations. Each synapses on HICANN-DLS features sensors measuring the correlation of pre- and postsynaptic spike times. Here, a STDP-like paring experiment is conducted: The measurement of each sensors is assessed in parallel via the CADC after stimulation with a pair of pre- and postsynaptic spikes with time difference $\Delta t = t_i^l - t_j^k$. To facilitate STDP-like plasticity experiments, the sensors are designed to emulate an exponential correlation kernel. Each synapse features two circuits, one for the measurement of causal spike-time correlations f_+ with $\Delta t > 0$ **(a)** and a second one for the assessment of anticausal spike pairings with $\Delta t < 0$ **(b)**. Shown are the respective measurements for all 32×32 synapses present on HICANN-DLS.

10 bit. Moreover, the fire signal leads to the closing of the switch, thereby shorting C_{mem} to u_{reset}. After the expiration of the duration of the refractory period τ_{ref}, the switch is again released. The parameter τ_{ref} is configurable by a refractory bias current.

Column-parallel analog-to-digital converter: The analog neuromorphic core is augmented by a column-parallel analog-to-digital converter (CADC) which forms the connecting line between analog and digital processing. Most notably, it can be used to read out the analog correlation voltages (Figure 3.4). The CADC is located at the top of the synapse array and extends over its full width (Figure 3.2a). Every column of this parallel ADC features two channels; one for the digitization of the causal and one for the anticausal correlation measurement. Both of which can be used simultaneously resulting in a total of 64 inputs with a resolution of 8 bit. The CADC is built upon a ramp-compare architecture with a centralized ramp generator: This global circuit generates a linearly rising voltage ramp which is distributed to all channels. The actual value of this ramp is compared to the applied input voltages of each channel. The time difference between the start of the ramp and the match of ramp and input voltage is proportional to their difference. Channel-wise counters measure this time difference. Running at a clock speed of 100 MHz, the theoretical sample rate of the CADC is determined by the number of counts, i. e. its precision, and therefore given by approximately 390 kHz. In the design of BrainScaleS-2, the CADC plays a central role as it constitutes the connecting link between the analog emulation and the digital processing with the plasticity processing unit (PPU).

Analog parameter storage: For all of the aforementioned circuits to work properly, supply voltages and bias currents are required. An on-chip analog capacitive memory generates all voltages and currents necessary to individually configure all components of the neuromorphic chip (Hock et al., 2013). Like the synapse array and the neurons, this parameter storage is organized in columns with additional extensions for global currents and voltages. The analog behavior of each neuron is



(b) Arithmetic

Bit	Value	Weight
0	2^{-7}	0
1	2^{-6}	w_0
2	2^{-5}	w_1
3	2^{-4}	w_2
4	2^{-3}	w_3
5	2^{-2}	w_4
6	2^{-1}	w_5
7	2^{-0}	0

Figure 3.5: The architecture of the PPU is optimized for the implementation of plasticity rules. (a) The PPU consists of a general-purpose part implementing the Power ISA and a SIMD vector unit to accelerate computations. The general-purpose part comes with 16 KiB of on-chip memory and a 4 KiB instruction cache. Aside from performing arbitrary computations, it can send vector instructions to a queue which is then processed by the SIMD vector unit. The functional units of the vector extension operate on 128 bit vectors of which 32 can be stored in the VRF. A 128 bit wide bus facilitates a close attachment to the analog neural network core and in turn provides access to the synaptic SRAM as well as the CADC. **(b)** The SIMD vector unit implements a fractional saturation arithmetic. The 6 bit synaptic weights are aligned with the 8 bit fractional saturation arithmetic by a bit-shift. By shifting the weight values by one bit to the left, the most significant bit w_5 of each weight is aligned with the 6-th bit of the fractional saturation arithmetic, thereby allowing to exploit saturation effects. Accessing synaptic weights hence requires this bit shift after each load and in the reverse direction in advance of each store operation.

individually configurable by 17 integrated digital-to-analog converters (DACs). Each of these DACs comes with a precision of 10 bit admitting room for detailed calibration routings equalizing out systematic parameter variability induced by the analog implementation (Pfeil et al., 2013; Brüderle et al., 2011; Neftci & Indiveri, 2010; Neftci et al., 2011b; Aamir et al., 2018).

3.1.2 Routing

External spike events can be injected into the analog neuromorphic core by a single-ended serializer/deserializer (SERDES) link from the field-programmable gate array (FPGA). Since the prototype ASIC HICANN-DLS lacks on-chip routing logic, the event routing is also implemented on the FPGA. In general, the FPGA features two different routing modes. Here, we rely on one of these modes of routing which is used throughout this thesis. This mode is characterized by a short roundtrip time d_{syn} of recurrent spikes which comes, however, at the cost of less flexible routing possibilities. In more detail, all accruing recurrent events are tagged with the same address label and injected back into the chip. Specifically, the events emitted by neuron i are transmitted to synapse driver i. Together with excitation and inhibition being a row-wise property, this choice enforces the adherence to Dale's law (Dale, 1934), i. e. a neuron only forms excitatory or inhibitory synapses with its postsynaptic partners, but not both simultaneously.

3.1.3 Plasticity processing unit

The incorporation of a freely programmable embedded microprocessor into the BrainScaleS-2 architecture significantly expands its versatility (Friedmann et al., 2016). In particular, the latter is tailored to the implementation of synaptic plasticity and hence enables the execution of custom programs interacting with the analog emulation of neuro-synaptic dynamics. In more detail, the PPU comprises of a general-purpose part and a single instruction, multiple data (SIMD) vector unit which constitutes a weakly coupled coprocessor (Figure 3.5a). The general-purpose part implements the Power instruction set architecture (ISA) and features a total of 16 KiB of main memory with an instruction cache of 4 KiB. Further, the processor is able to reconfigure all of the aforementioned components of the neuromorphic system during experiment execution by accessing the chip-internal configuration bus. The analog neural network core can be efficiently interfaced by the SIMD vector unit. In particular, the tight coupling of the SIMD vector unit to the synapse array's SRAM controller and the CADC promotes massively parallel updates of the synaptic SRAM. The PPU can thus be used for a vast array of applications such as near-arbitrary learning rules (Bohnstingl et al., 2019; Billaudelle et al., 2021; Grübl et al., 2020; Cramer et al., 2020a), on-line circuit calibration (Leibfried, 2021), or the co-simulation of an environment capable of continuous interaction with the network running on the neuromorphic core (Wunderlich et al., 2019; Schreiber, 2021).

The SIMD vector unit is optimized for the parallel calculation of plasticity updates. In more detail, the SIMD unit constitutes a weakly coupled coprocessor (Figure 3.5a): The general-purpose part of the PPU can send vector instructions to a queue which is processed alongside by the SIMD unit. Specifically, the instructions sent to this queue are processed in order by the vector unit and in turn decoded as well as distributes to the corresponding functional units. The vector unit operates on 128 bit wide vectors and has access to the vector register file (VRF) which can store up to 32 vectors. In general, the functional units units can either process vectors with a precision of 8×16 bit or 16×8 bit. For each precision, the vector elements can either be treated with integer modulo or fractional saturation arithmetic. In the following, we stick to the fractional saturation arithmetic and a precision of 16×8 bit as this setting is most relevant for the results presented in this thesis. In this arithmetic, the vector elements are interpreted as signed rational numbers in the range $[-1, 1)$ with a resolution of 2^{-7} (Figure 3.5b). Hence, a signed 8 bit integer i maps to $i/128$ if interpreted in the fraction saturation arithmetic. Most notably, operations resulting in values out of the dynamic range clip at -1 and $1 - 2^{-7}$ which can be exploited to efficiently prevent overflows within weight update calculations.

A tight coupling of the PPU to the analog neural network core is established by two different load-store units. First, a 32 bit unit guarantees access to the main memory as well as the on-chip configuration bus from the general-purpose part. This unit enables the PPU to reconfigure all other components of the neuromorphic system during experiment execution. Second, a 128 bit bus provides access to the synaptic SRAM and the CADC from the SIMD vector unit. The choice of a precision of 16×8 bit is not only convenient because of the 8 bit precision of the CADC, but also synaptic weights can be easily converted to profit from the fractional saturation arithmetic. First, the 6 bit weights on the prototype ASIC are aligned to the 8 bit resolution of the SIMD vector unit by filling the most significant bits with zeros (Figure 3.5b). To finally profit from the saturation arithmetic, this conversion is augmented by shifting the weights one bit to the left, resulting in an alignment of the most significant bit of the weights with the 6-th bit of each vector element. This bit-shift is applied after each load operation and in the reversed direction before each store opera-

tion (Listing 1). As a result, the largest weight of 63 lsb corresponds to a value of 126/128 on the vector unit, whereas the largest fractional number 127/128 is translated back to a weight of 63 lsb. Thereby, we prevent wrap-around effects of weights already being at their maximum value.

Scalability of the hybrid plasticity approach implemented by the incorporation of the PPU in the BrainScaleS-2 architecture is guaranteed by the parallelism of the SIMD vector unit. With the choice outlined above, the SIMD unit operates in parallel on slices of 16 synapses. Hence, two accesses are required to update all synapses residing within the same row (Listing 1). In case the width of the SIMD vector unit is scaled linearly with the number of neurons, the provided parallelism lets plasticity algorithms scale $\sim \mathcal{O}(m)$ with the indegree m, but $\sim \mathcal{O}(1)$ with the number of postsynaptic neurons. In addition, the clock speed of the PPU can be increased to further facilitate scalability. On the considered prototype system, the PPU runs at a clock speed of 100 MHz.

All kernel codes shown within this thesis are embedded into an update loop comparable to Listing 1. To ensure precise temporal control of synaptic updates, the PPU is programmed to continuously poll the FPGA memory for update signals which can, in turn, be triggered from the FPGA via the host system (lines 2 to 4). In case this signal was sent, the PPU iterates over all slices of the synapse array (lines 7 and 8). For STDP-based update rules, the processing of each slice starts with the assessment of the causal and anticausal correlation measurements via the CADC (lines 11 and 12). The resulting data is then shifted by one bit to the right in order to correctly interpret and treat the unsigned 8 bit measurements with the signed 8 bit fractional arithmetic of the vector unit (lines 13 and 14). Afterwards, the correlation sensors are reset (line 16). The synaptic weights can also be directly loaded into the SIMD vector unit (line 18). Prior to the update calculations, the weights are aligned to the 8 bit representation by a bit-shift to the left as outlined previously (line 19). After the calculation and application of weight updates, a compare operation is utilized to clip all negative elements of the weight vector at zero to prevent underflows (lines 22 and 23). It is noteworthy that the weights do not need to be checked for overflows due to the saturation arithmetic. Before actually writing back the weights, the initial bit-shift is reversed (lines 25 and 26).

3.1.4 Experimental setup

The neuromorphic chip HICANN-DLS is bonded on a chip carrier board (Figure 3.1a and b). The latter mainly contains passive components for supply voltage stability. Further, it features some test pads for direct signal readout. A standard small outline dual inline memory module (SODIM) connector establishes the connection between the carrier board and a baseboard. This baseboard provides the required supply voltages for all functional components as well as debug pins. Moreover, it houses a custom printed circuit board (PCB) on which a Xilinx Spartan-6 FPGA, memory, a differential input ADC, multiple digital general-purpose I/O lines and universal serial bus (USB) connectivity is mounted.

The Xilinx Spartan-6 FPGA serves as an interface with a host system. On the one hand, the FPGA communicates with a host computer via USB 2.0. On the other hand, the communication between the neuromorphic chip and the FPGA is established by four digital data signals and a clock line where the accruing data on both sides is serialized and deserialized by a SERDES. The FPGA provides the sequencing mechanisms for experiment control and spike handling including the configuration, the experiment control, the spike recording and in case of the HICANN-DLS ASIC the spike routing. Due to the accelerated physical implementation, configuration and spike data are first stored in 512 MiB of DDR3 synchronous dynamic random-access memory (SDRAM) and then played back

```
1  ; wait for run signal
2  while(command != HALT) {
3    if(command == RUN) {
4      command = NONE;
6      ; iterate over synapse array to update weights
7      for(size_t half = 0; half < dls_num_vectors_per_row; half++) {
8        for(size_t row = 0; row < dls_num_rows; row++) {
9          asm volatile(
10           ; load and shift correlation measurements
11           inx ca_meas, %[ca_base], %[offset]
12           inx ac_meas, %[ac_base], %[offset]
13           shiftb ca_meas, ca_meas, -1
14           shiftb ac_meas, ac_meas, -1
15           ; reset correlation sensors
16           outx %[select], %[ca_base], %[offset]
17           ; load and shift weights
18           inx weights, %[weigh_base], %[offset]
19           shiftb weights, weights, 1
20           ; calculate updates
21           ; set to zero if result is smaller than zero
22           compareb weights
23           select weights, weights, %[zeros], LT
24           ; Save shifted weights
25           shiftb weights, weights, -1
26           outx weights, %[weight_base], %[offset]
27           : [weights] "=&qv" (weights)
28           : [zeros] "qv" (zeros),
29             [weight_base] "b" (dls_weight_base),
30             [offset] "r" (2*row + half));
32       }; end for
33     }; end for
34   }; end if
35 }; end while
```

Listing 1: Control loop for the execution of plasticity kernels. PPU updates are triggered from the FPGA. The general-purpose part of the PPU continuously polls the FPGA memory for update signals. In case a run signal is detected, a sequence of updates is initiated by iterating over the synapse array. On HICANN-DLS slices of 16 synapses can be updated in parallel. Hence, two accesses are required to update all synapses residing within the same row, implemented by the outer loop. The inner loop iterates over all rows. Most plasticity kernels start with the triggering of CADC measurements to assess the causal and anticausal correlation measurements. The unsigned 8 bit measurements are aligned with the signed 8 bit arithmetic by a bit-shift to the right. Afterwards, the correlation sensors are reset. Next, the 6 bit synaptic weights are loaded into the vector unit. To profit from the 8 bit saturation, the weights are shifted by one bit to the left. After the calculation of updates, the resulting weight values only have to be checked for lower saturation before reverting the bit-shift and the write to the synaptic SRAM. The kernel code is shown using NASM syntax.

(a) Neuromorphic system (b) Neuromorphic chip

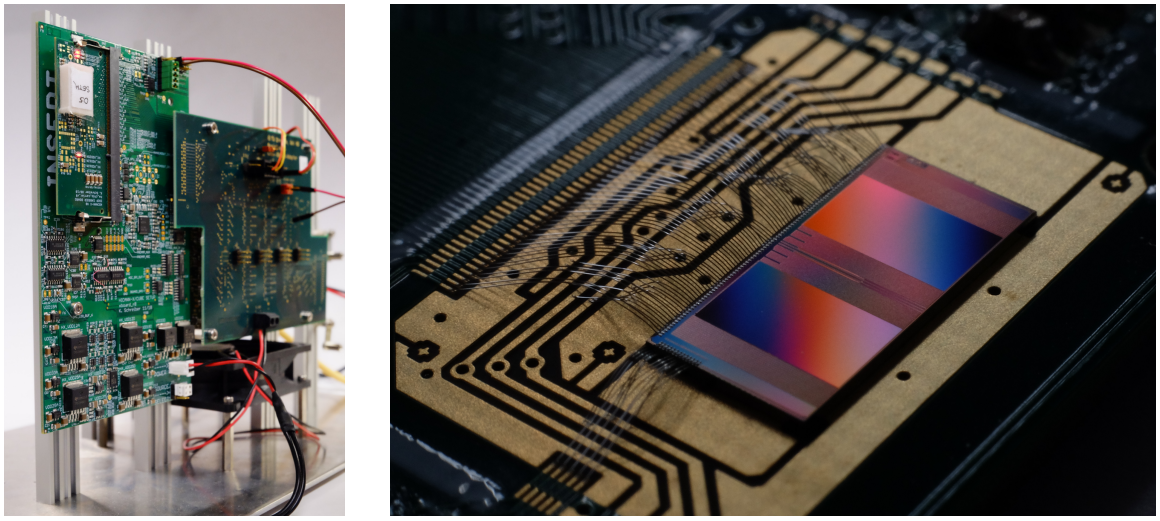


Figure 3.6: HICANN-X is the first full-size single chip system implementing the BrainScaleS-2 architecture. (a) Several PCBs connect the neuromorphic chip (top left, below white cover) to an FPGA which serves as an interface to the host system. **(b)** The neuromorphic chip HICANN-X is mounted on a carrier board. The chip covers an area of $4 \times 8\,\mathrm{mm}^2$ and houses 512 AdEx neuron circuits with 256×512 synapses. Photographs taken from Müller et al. (2020).

by the FPGA using cycle-accurate timing. Both, the FPGA and the internal chip logic run at a clock frequency of 100 MHz. Therefore, the best-case temporal precision is 10.4 ns corresponding to 10.4 µs in biological time when accounting the constant speedup factor of 1000.

3.2 HICANN-X

In the following, the HICANN-X chip is described by highlighting the key differences to the prototype system HICANN-DLS. Most notably, HICANN-X represents the first full-size neuromorphic chip implementing the BrainScaleS-2 architecture. The ASIC covers an area of $4 \times 8\,\mathrm{mm}^2$ (Figure 3.6). This section is structured exactly like its predecessor by starting with an overview of the analog neural network core in Section 3.2.1, followed by a description of the event routing in Section 3.2.2. Alterations affecting the PPU are detailed in Section 3.2.3. Again, we close with a description of the full experimental setup in Section 3.2.4. For further details, the reader is referred to Schemmel et al. (2020).

3.2.1 Analog neuromorphic network core

Being at full chip size, the analog network core of HICANN-X physically implements four interconnected blocks of neurons and synapses (Figure 3.7). Each of these blocks contains its own synapse array and the associated neuron compartments as well as a capacitive memory and a CADC instance.

Synapse arrays: Each of the four synapse arrays features 256×128 current-based synapses (Figure 3.7). The rows in these synapse arrays are fed with spike events by synapse drivers which are located between two adjacent synapse arrays. In contrast to HICANN-DLS, each synapse driver is

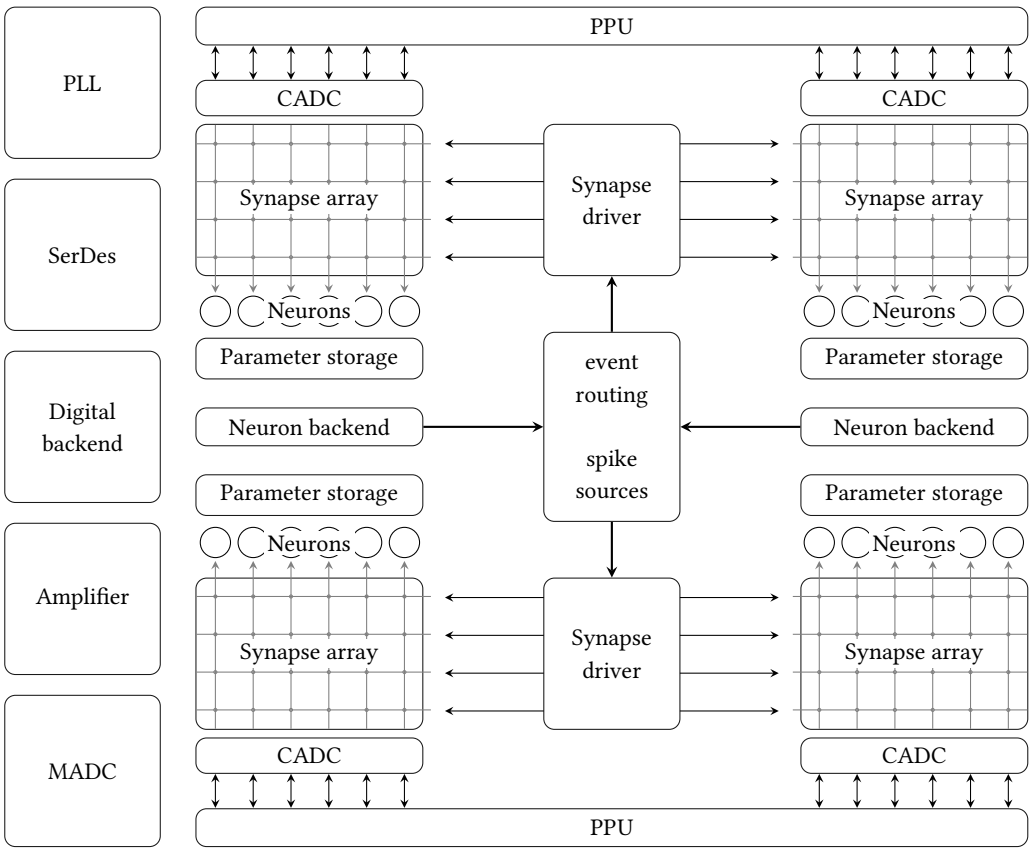


Figure 3.7: Block-level schematic of the BrainScaleS-2 full size ASIC. The analog network core comprises four blocks. Each block contains 128 AdEx neuron compartments, a synapse array of size 256×128, an instance of the CADC and an associated analog parameter storage. Two adjacent blocks of neuron compartments share a digital neuron backend, synchronizing and serializing neural events onto digital output buses. A central event router drives the event buses on each half. Two closely attached PPUs can access neurosynaptic observables by the CADC as well as the synapse array on their respective chip half. The analog network core is surrounded by digital logic, most notably the PLL, eight SERDES high-speed links, the digital backend, readout amplifier and the MADC.

associated with two rows, resulting in a total of 128 drivers per chip half. Spike events are represented by a 6 bit address within a synapse array. Together with the synapse driver being individually addressable, a total of 16 386 distinguishable presynaptic partners are available, i. e. at least 8 presynaptic resources are shared among a single synaptic row.

Aside from their increased count, the synapse drivers of HICANN-X also feature new circuits providing additional modelling capabilities. As a first extension, a simplified Tsodyks-Markam model for short-term plasticity (STP) is implemented by dynamically modulating the pulse width within the synapse drivers. The state variables of the model are kept on an array of capacitors within each driver (Schemmel et al., 2007; Billaudelle, 2017). Another important innovation is the ability of synapse drivers to not only handle single- but also multi-valued input signals. With this, analog vector-matrix multiplication can be performed which in turn promotes the implementation of artificial neural networks (ANNs) on HICANN-X (Weis et al., 2020; Stradmann et al., 2021).

Neurons and synaptic input: On HICANN-X, the physical LIF neuron model is augmented by additional circuits to provide adaptive exponential leaky integrate-and-fire (AdEx) functionality.

Specifically, each block houses 128 of these AdEx neuron compartments. Each of the latter receives input via two lines from the aforementioned columns of 256 synapses. In contrast to the prototype ASIC, the neurons can be connected to form structured neurons or to enlarge the synaptic fan-in (Schemmel et al., 2017; Aamir et al., 2017). This can be achieved by connecting the membranes and the digital spike signals of each neuron either vertically or horizontally.

Analog parameter storages: HICANN-X provides an adjacent analog parameter storage for each row of neuron compartments. Like for its predecessor, this storage is organized in columns. In more detail, the storage now provides 24 values for each AdEx neuron and 48 global parameters. Again, this manifold of controllable parameters paves the way for detailed calibration routines, equalizing out potential circuit-to-circuit variations.

Column-parallel analog-to-digital converters: Each block on HICANN-X houses its own instance of the parallel CADC with two channels per column. The CADC has also been scaled up with a total of 256 channels per instance, i. e. 1024 channels per chip. In contrast to HICANN-DLS, the CADCs on HICANN-X do not only provide access to synaptic correlation measurements, but moreover are able to digitize the synaptic inputs as well as the membrane potentials. Especially the availability of the latter is key for many learning rules discussed in the literature (Urbanczik & Senn, 2014; Zenke & Ganguli, 2018a). In addition, digital calibration circuits have been added to the CADC to eliminate inter-channel variability.

3.2.2 Routing

HICANN-X comes with 16 low voltage differential signaling (LVDS) lines for host communication (Karasenko, 2020). These links have a total I/0 bandwidth of 16 Gbit when driven with a 500 MHz clock, i. e. 1 Gbit per line. With the current spike encoding on HICANN-X, this translates to 250×10^6 spikes s^{-1} per direction. In other words, each neuron can emit spike events at a rate of 488 kHz which is significantly higher than average firing rates reported for biological tissue. All eight physical links present on the neuromorphic chip can not only be used simultaneously for the transfer of neuron event data, but also for slow control and PPU as well as global memory accesses.

With scaling up the ASIC, not only the number of neuron and synapse circuits increases, but also the number of required and accruing events rises. Thereby, the I/O bandwidth is further challenged. To address this concern, spike sources have been included in the central event routing unit (Figure 3.7). These sources can either produce spikes with regular inter-spike intervals or random Poisson distributed spikes times. By relying on these sources, a dramatic relief of the input bandwidth can be achieved, since the associated events are generated on-chip and can hence be directly fed into the synapse array via the synapse drivers. This approach allows to save bandwidth resources and gives the modeler more flexibility to stimulate the silicon neurons with externally generated input spike trains comprising a complex structure.

The routing of external, internal as well as spike source events is implemented by a crossbar (Figure 3.8a). In general, there are two layers of communication: first a real-time address-event layer (L1) and second a layer using time-stamped event packets (L2). Both types of events are converted into each other in the digital core logic driving the crossbar. In total, this crossbar provides two symmetric sources and sinks for event data: first, the analog network core featuring eight input and

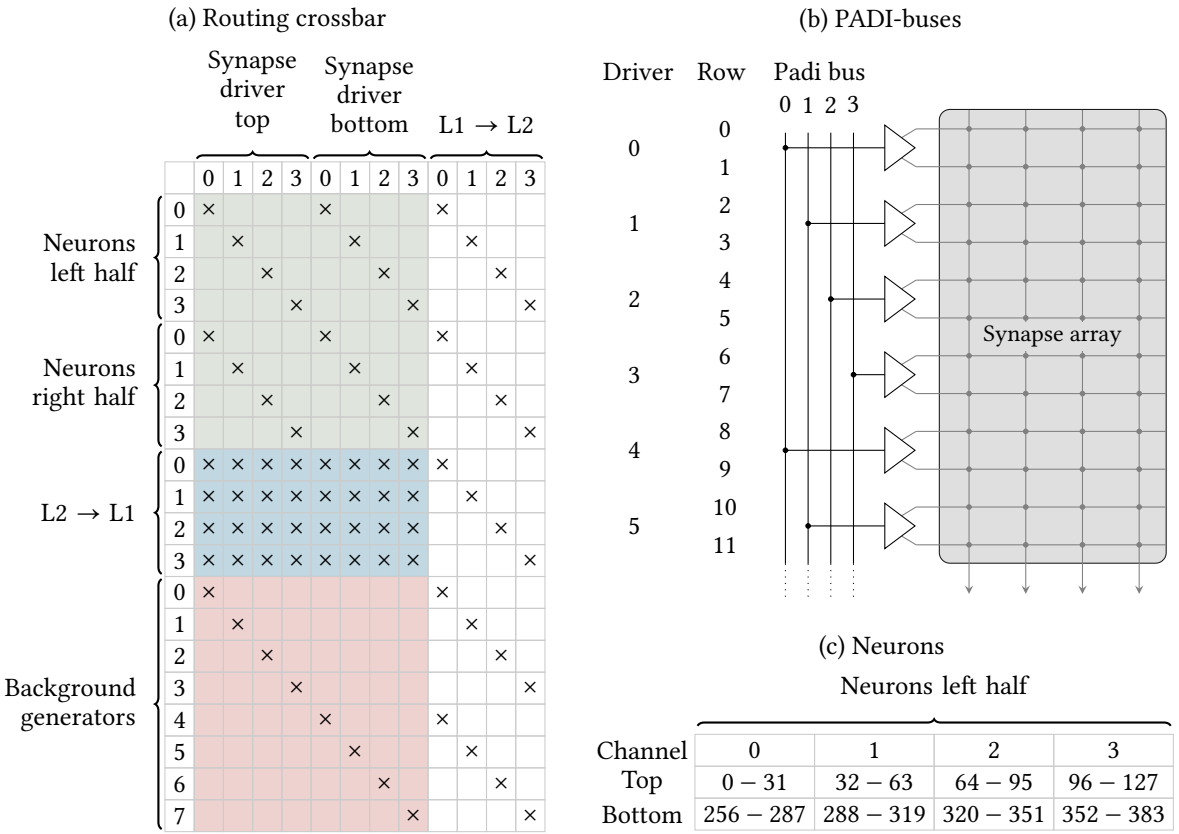


Figure 3.8: Illustration of the event routing on HICANN-X. (a) A crossbar implements the event routing of external, internal as well as spike source events. All input sources are shown on the left, whereas the output channels are illustrated at the top. At the position of crosses, a programmable routing element is located. Chip-external events enter via L2 → L1 connections, whereas accruing event data leaves the chip by L1 → L2 nodes. Besides external events, the neurons of each chip half in combination with potential events emitted by the background generators constitute event sources of the crossbar. All of this event data can be injected into the synapse drivers of each chip half. **(b)** The synapse drivers are statically connected to the first eight output channels of the crossbar via four PADI buses per chip half. Within each half, the PADI buses are connected to the synapse drivers in an alternating fashion. **(c)** Closing the event chain, the neurons on the left half connect to the first four input channels of the crossbar, whereas the right half is attached to the second four input channels. Each neuron is uniquely assigned to a single crossbar input channel.

eight output event buses as well as an event link layer with four links in each direction to insert external events and to conduct away events to other chips and/or the host system. The background generators constitute an additional source of event data. All sources and sinks result in a total of 20 distinct input and 12 output channels.

Prior to injection and after leaving the crossbar, external events are 16 bit wide. In more detail, an external 16 bit event is assigned to one of the four input channels of the crossbar by taking the value of the two most significant bits. The latter then directly corresponds to the respective crossbar input channel. For further treatment, these two bits are discarded within the crossbar, resulting in 14 bit events. In the reverse direction, the 14 bit wide crossbar events are extended by 2 bit characterizing the respective output channel. The crossbar operates on 14 bit wide events which are all treated equally and remain unchanged. Each crossbar node connecting an input to an output channel filters these events. In more detail, an event is only forwarded to a crossbar output channel in case the masked event ID matches a target, i. e. event & mask == target. Here, mask and

target are configurable 14 bit values.

The synapse drivers are connected to the crossbar output channels by four parallel driver interface (PADI) buses per hemisphere in a one-to-one correspondence. These buses handle 11 bit events which are obtained by only taking the least significant bits of the crossbar events. The resulting PADI events are forwarded to the 128 synapse drivers per hemisphere. Each of them is connected to a single PADI bus in an alternating fashion (Figure 3.8b). Hence, each PADI bus statically connects to 32 synapse drivers. Every synapse driver filters incoming PADI events based on a freely configurable 5 bit mask in combination with the index of the synapse driver on its PADI bus. Specifically, an event is only forwarded to the synapse array in case the masked five most significant bits of the PADI event match the masked index on the PADI bus. In case this condition is fulfilled, the lower 6 bit of the PADI event are forwarded as a synaptic event to the two rows of the synapse array connected to the respective synapse driver. At this point, the routing within a single block of synapses is comparable to the one of the prototype ASIC.

For inter-block communication, two adjacent neuron blocks share a digital neuron backend (Figure 3.7). The latter synchronizes accruing neural events to the digital system clock of 125 MHz and serializes them onto digital output buses (Kiene, 2017). In case a neuron emits a spike, an event is generated which is again injected into the routing crossbar. To that end, each neuron is connected to exactly one channel of the crossbar (Figure 3.8c). The generated event is again 14 bit wide and configurable via the digital neuron backend. In more detail, the lower 8 bit can be freely set per neuron, whereas the six most significant bits are clamped to zero. Besides event addressing, the digital neuron backend further extends the capabilities of the neurons and replaces former analog circuits like the implementation of the refractory time by digital circuits.

3.2.3 Plasticity processing unit

Like HICANN-DLS, HICANN-X provides digital extensions for the flexible implementation of plasticity. In contrast to HICANN-DLS, HICANN-X contains two independent PPUs within each chip half (Figure 3.7). Both of which can in general access the same set of variables and configuration as their predecessor. Parallel access to the synapse array and the CADC, however, is only provided within the associated chip half. Nevertheless, the PPUs are capable of exchanging data via a shared FPGA memory with a total size of 128 KiB. This memory can also be used to communicate data between the PPUs and the host system which builds a central element of the framework discussed in Chapter 5.

Scalability of the hybrid plasticity approach is guaranteed by enlarging the size of the SIMD vector unit as well as a higher clock speed. On the full size ASIC, the SIMD vector units operates on 1024 bit wide vectors and feature an appropriate bus to access the synaptic SRAM as well as the CADC. It, therefore, allows processing 128 synapses in parallel on each chip half. In addition, the clock speed is increased to 250 MHz. Both provisions promote scalability of plasticity implementations for the increased system size.

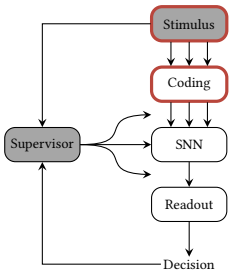
Scalability is further facilitated by the incorporation of pseudo-random number generators. While for the prototype chip HICANN-DLS, random numbers had to be drawn on the general-purpose part and then transferred to the SIMD vector unit of the PPU, HICANN-X provides specialized hardware acceleration. Most notably, these accelerators can be directly employed from the SIMD vector unit. This massively parallel access to random numbers improves the update frequency of the synaptic

SRAM in the case of plasticity rules involving stochastic terms. It is noteworthy that the fixed-point arithmetic often requires stochastic rounding. By drawing on the aforementioned hardware accelerators, rounding mechanisms could be efficiently incorporated into update calculations on the SIMD vector unit.

3.2.4 Experimental setup

The periphery of the analog neural network core of HICANN-X has been augmented by two novel components. First, a fast ADC has been included to digitize any signal originating from the analog neural network core at high speed and precision. Specifically, this membrane analog-to-digital converter (MADC) allows to digitize analog signals with a resolution of 10 bit and achieves a sample rate of up to 62.5 MSample s^{-1} when driven by a 750 MHz clock signal. Most notably, this MADC renders the substrate more autonomous by making external measurement equipment superfluous, thereby simplifying the calibration of all analog circuits. Second, a phase-locked loop (PLL) has been included which converts a reference clock to different output frequencies. While the PLL relies on a 50 MHz signal, it is able to generate different clock signals with up to 1.5 GHz. It is this module that generates the high-speed clock signals for the MADC, the PPU and further off-chip communication infrastructure.

As for the prototype system, HICANN-X is bonded to a chip carrier board. The latter is attached to a Cube Setup which has been originally developed for the BrainScaleS-1 system (Figure 3.6). Essentially, this setup is composed of six PCBs: four Kintex-7 FPGA boards, one I/O board and the xBoard (Kleider, 2017; Güttler, 2017; Schreiber, 2021). Like for HICANN-DLS, the FPGA provides the required sequencing mechanisms for experiment control and spike handling. Connectivity with the host system in form of Ethernet and USB is established by the I/O board. The xBoard, in contrast, interfaces the Cube setup to the HICANN-X carrier board. Most notably, the system also comes with INA219 current and voltage monitors which provide easy access to the system's power consumption.



4 Benchmarks

The context of this chapter has been published in Cramer et al. (2020b) in collaboration with Yannik Stradmann under the supervision of Dr. Friedemann Zenke. In the following, I will follow the structure of the publication, but with a more detailed description of the generation of the described benchmark dataset.

In the previous chapters, we introduced spiking neural networks (SNNs) as well as strategies for their efficient implementation. Irrespective of the implementation on either conventional computers or neuromorphic systems, the assessment of computational capabilities of SNNs builds an essential step within the development of novel optimization strategies. In this context, classification is one of the often pursued approaches. Here, the agreement on a common set of problems within the community is indispensable for well-defined comparisons which foster the development of new algorithms. In contrast to the artificial neural network (ANN) community, the SNN community has not agreed on a set of canonical benchmarks. Moreover, the frequently available time series data used for the performance quantification of ANNs cannot directly be used for SNNs: Here, the samples have to be converted to the spiking domain prior to stimulation. Since the data and the conversion are often treated separately, there is a deficiency of spike-based benchmarks. This constitutes a fundamental problem since the pre-processing often involves complex processing steps that complicate the dissection of the impact of the conversion and the neural processing of a downstream SNNs on performance and hence harm comparability. Here, we aim to tackle these problems by introducing two spike-based datasets for classification with SNNs implemented on conventional as well as neuromorphic hardware. Both benchmarks are based on collections of spoken words, namely the novel high-fidelity Heidelberg Digits (HD) and second Google's Speech Commands (SC). We finally convert the raw audio signals into spikes by a chain of models capturing key features of the ascending auditory pathway. Based on the converted data, we establish the first set of baseline performances by training non-spiking as well as spiking classifiers and demonstrate that the spike times contain inevitable information for the classification of both datasets.

4.1 Introduction

One prominent way of benchmarking neural networks is to assess their performance on a common classification problem to finally compare different approaches and methods. In contrast to ANNs, there exists no general set of tasks to benchmark SNN implementations (Davies, 2019; Roy et al., 2019). This lack of suitable benchmarks has the potential to slow down the development within the SNN research community. Often, only specific properties of an SNN are investigated by *private* benchmarks which are hence tailored to a specific problem at hand. Despite their necessity, these tasks are not sufficient to foster comparability among different architectures and optimization

approaches. Ideally, the community as a whole needs to agree upon a set of shared benchmark datasets.

The lack of common spike-based benchmark datasets for SNNs is partially due to the variety of different learning schemes and visited architectures. Approaches directly tying on the success of conventional ANNs by employing steady-state rate-coding schemes do not require spike-based datasets. In this scenario, the input and output rate remain constant during the stimulation with a single input pattern (Zylberberg et al., 2011; Neftci et al., 2017; Pfeiffer & Pfeil, 2018). Often, the inputs enter as Poisson distributed spike trains with a rate proportional to the stimulus intensity, i. e. the gray value of an image. Similarly, the network signals its decision in form of firing rates for each output unit. Hence, networks employing steady-state rate-coding can often easily be trained by network translation (Pfeiffer & Pfeil, 2018) on standard machine learning datasets like MNIST (LeCun et al., 1998), CIFAR10 (Krizhevsky et al., 2009), or SVHN (Netzer et al., 2011), eliminating the need for spike-based benchmarks. However, these rate-coding approaches neither allow to naturally process time series data nor to exploit the temporal processing capabilities provided by SNNs.

Temporal coding schemes constitute an elemental approach to efficient information processing with SNNs. Here, the input as well as the output activity vary during the presentation of a single stimulus sample. Therefore, the information is coded in the timing and outputs can be either individual spikes (Gütig, 2014; Bohte et al., 2002; Mostafa, 2017; Comsa et al., 2020), spike trains with predefined firing times (Memmesheimer et al., 2014) or varying firing rates (Gilra & Gerstner, 2017; Nicola & Clopath, 2017; Thalmeier et al., 2016). Furthermore, continuously varying quantities derived from output spikes have been considered, typically in form of linear combinations of low-pass filtered spike trains (Eliasmith & Anderson, 2004; Denève & Machens, 2016; Abbott et al., 2016; Nicola & Clopath, 2017; Gilra & Gerstner, 2017). In general, the time series data constituting the input side can be injected into an SNN in form of time-varying currents. However, this is often not possible for the processing on neuromorphic hardware: Most current systems do not support the stimulation with time-varying currents evolving on time scales of neuro-synaptic dynamics. And even if systems support current injection, they do not allow to stimulate a large set of neurons by different inputs simultaneously. Nevertheless, all contemporary devices implementing spiking neurons support the injection of precisely timed input spike trains. Hence, the time series data constituting a given benchmark dataset needs to be transformed to spike trains prior to the actual stimulation. This process and preceding methods for feature selection usually involve complex processing chains, which leave fundamental design decisions to the modeler and violate comparability. Especially, the dissection of the SNN's performance and the effect of conversion becomes increasingly challenging. Because of this, spike-based benchmark datasets are required for the systematic evaluation of the performance of SNNs in general and their hardware equivalents in particular.

Artificially generated spike patterns have been used to quantify the information processing capabilities in the past. The temporal exclusive-OR (XOR) task and variations thereof represent a simple and standardized approach to the assessment of the performance of an SNN employing temporal coding (Bohte et al., 2002; Huh & Sejnowski, 2018; Abbott et al., 2016). Within this task, a network is trained to solve a boolean XOR where the logical *on* and *off* states correspond to early and late spike times respectively. Often this task is used to demonstrate hidden layer learning since it can not be accurately solved by an SNN without a hidden layer, which is in direct accordance with the perceptron's inability to solve the regular XOR task. However, this benchmark is of limited applicability as it is already saturated due to its low dimensionality. In more general scenarios, pattern generation tasks have been used to assess the performance of SNNs, directly exploiting their sparse

spatio-temporal coding capabilities (Ponulak & Kasiński, 2009; Pfister et al., 2006; Florian, 2012; Mohemmed et al., 2012; Memmesheimer et al., 2014; Gardner & Grüning, 2016). Here, the SNNs are trained to generate a specific target spike train when stimulated with a set of regular (Gardner & Grüning, 2016) or random input spike trains of variable length and statistics (Memmesheimer et al., 2014). This general idea has also been visited by Gütig & Sompolinsky (2006) who performed binary classification of random input spike patterns. While these approaches already make use of spatio-temporal patterns, they lack non-random structure, which is, however, present in many real-world stimuli. Aside from the aforementioned random synthetic datasets, the classification of spike patterns with additional spatio-temporal structure has been considered. One example is based on smooth random manifolds from which spike times can be sampled (Zenke & Vogels, 2021). These synthetic datasets have the advantage of controllable dimensionality as well as sample and class counts and hence provide an excellent opportunity for an initial investigation of computational capabilities. However, this well-behaved formulation is in direct contrast to most real-world applications.

One last category of classification benchmarks stems from neuromorphic sensors. Prominent examples encompass datasets generated with a dynamic vision sensor (DVS) (Lichtsteiner et al., 2008) and a silicon cochlea (Anumula et al., 2018). One instance of a vision dataset is the Neuromorphic MNIST, which was derived from the MNIST benchmark by projecting the associated images onto a screen (Orchard et al., 2015). The digits were moved in order to elicit spikes within the DVS's response. The resulting spike trains can then be directly used in classification scenarios and are hence very successful in the SNN community. Since the dataset is based on MNIST, it is already nearing saturation, i. e. it is almost solved perfectly with current methods. The DVS128 gesture dataset is another vision dataset that has been released more recently by IBM under a Creative Commons license (Amir et al., 2017). This benchmark consists of DVS recordings of 11 unique hand gestures performed by different subjects under various lighting conditions. The stimuli are available in form of a continuous data stream and hence entail extensive preprocessing. Furthermore, the high dimensionality of 128×128 pixels renders this dataset computationally challenging. Last, the DASDIGIT benchmark naturally exploits the intrinsic temporal processing of SNNs in a lower dimension (Anumula et al., 2018). It was generated by processing the spoken words of the TIDIGTS dataset with a 64 channel silicon cochlea. However, the original samples are released under a proprietary license and the provided word sequences of the TIDIGITS go beyond the capabilities of many current SNN implementations.

Based on this assessment of previous work, which is notably far from complete, we formulate a minimum set of requirements for a benchmark dataset. First, it has to be free to use and needs to be published under a permissive public domain license. Second, the benchmark should be easily applicable and self-explanatory. Especially, it should not require extensive preprocessing. Third, the data has to be general enough and not explicitly tailored to a specific problem at hand to attract broad interest within the community. And last, the task should not already be saturated, i. e. solved with almost perfect accuracy by existing methods. Otherwise, improvements upon existing methods would be hidden.

In this chapter, we present two general SNN benchmark datasets designed to require modest computational overhead for preprocessing as well as classification. To this end, we use speech data, which provides a natural temporal dimension and a relatively low bandwidth compared to video data. More specifically, we recorded the novel, high-fidelity Heidelberg Digits (HD) and resort to the existing Speech Commands (SC). We converted the raw time series data to spikes by a chain

of models inspired by the ascending auditory pathway. By training a set of conventional as well as spiking classifiers on the resulting spike trains, we demonstrate that these newly created samples are not already saturated by current methods. In that process, we show that leveraging spike-timing is essential to solve these problems with high accuracy. Moreover, we suggest approaches to quantify generalization by exploiting design features of both datasets as well as the pursued partitioning of samples.

4.2 Methods

In the following method section, we describe the two audio datasets which form the basis of our spike-based benchmarks (Section 4.2.1). Next, we outline the models used for the audio-to-spike conversion (Section 4.2.2) and highlight the data format used to distribute the final benchmark data as well as its organization (Section 4.2.3). We close with a description of the spiking (Section 4.2.4) as well as the non-spiking classifiers (Section 4.2.5) used to establish the first set of benchmarks. All reported performance measures within this chapter correspond to the mean and their errors to the standard deviation of 10 independent experiments.

4.2.1 Audio datasets

Our benchmarks are based upon two distinct audio datasets, which serve widely different application areas. While our newly recorded Heidelberg Digits (HD) were optimized for word-level alignment and overall recording quality, the Speech Commands (SC) closely replicate real-world conditions of mobile devices.

Heidelberg Digits: The Heidelberg Digits (HD) cover the spoken digits ranging from *zero* to *nine* in English and German language published under a permissive public domain license[1]. In more detail, the dataset provides a collection of 10 420 digits assigned to 20 distinct classes. The age of the six female and six male speakers spanned the range from 21 yr to 56 yr with a mean of (29 ± 9) yr. The native language of all subjects was German. While making the recordings, all of these speakers were tasked to speak as clearly as possible.

All digits were recorded in a sound-shielded room at the Heidelberg University Hospital. The near and far-field were captured by three microphones; two AudioTechnica Pro37 in different positions and a Beyerdynamic M201 TG (Figure 4.2a). The signals of all microphones were digitized with a sample rate of 48 kHz and a precision of 24 bit by a Steinberg MR816 CSX audio interface and finally stored in Free Lossless Audio Codec (FLAC) format. For convenience, the digits were recorded in ascending sequences for each language and speaker (Figure 4.1a). This not only renders the dataset balanced in terms of digits within each language, but also reduced the time spent on recording (Figure 4.1b and c). However, the use of sequences requires an automated cutting framework.

First, the recorded sequences were manually pre-selected and externally mastered (Schumann, 2019). We drew on a gate with speaker-dependent thresholds and release times to finally carve the individual digits within each sequence (Figure 4.1a). The gate parameters were optimized by a black-box optimizer (Knysh & Korkolis, 2016) to yield exactly 10 single digits per sequence. Additionally,

[1]The raw audio files of HD as well as the spiking datasets SHD and SSC are available at https://compneuro.net. In addition, we provide code examples and further information.

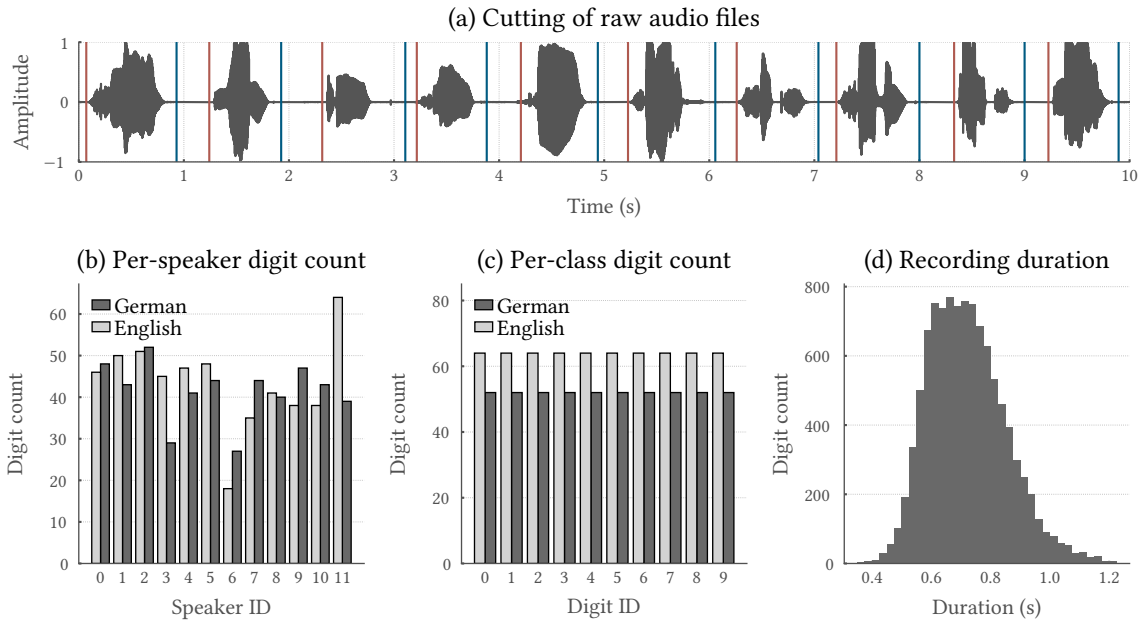


Figure 4.1: The HD have a balanced class count for each language and variable duration. The HD consist of 10 420 recordings of spoken digits ranging from *zero* to *nine* in English and German language. **(a)** The samples were recorded in sequences of 10 ascending digits for each language. The cutting times (colored) were determined by a gate and speaker specific ramp-in and ramp-out times. **(b)** Variable numbers of digits are available for each speaker and each language. **(c)** The dataset is balanced in terms of digits within each language. **(d)** The HD audio recordings were cut for minimal duration to keep computation time at bay. Panels (b) to (d) as well as their caption taken from Cramer et al. (2020b).

the threshold was constrained to be as low as possible to appropriately capture on- and offset effects. The release time was forced to be as low as possible to reduce the duration of each sample to a minimum, thereby minimizing the computational overhead of further modelling. To even further shorten audio file durations, we determined speaker-specific ramp-in and ramp-out times by visual inspection. This comprehensive processing produces samples with optimal temporal alignment and hence different sample durations due to speaker variations (Figure 4.1d). To not disturb the computation of fast Fourier transformations (FFTs) involved in the following processing stages, 30 ms Hann windows were applied to the start and end of the peak normalized audio signals respectively.

The partitioning of the HD into training and testing samples is designed to challenge generalization. To this end, we exclusively assigned all samples spoken by the speakers four and five to the test set. In order to result in a common split ratio, we further appended 5 % of the remaining recordings of each digit and language. By this, the evaluation of the test performance allows quantifying the ability to generalize to novel speakers.

Speech Commands: Recently, the TensorFlow and AIY teams published the Speech Commands (SC)[2] under a *Creative Commons BY 4.0* license (Warden, 2018). The dataset contains command words spoken by a total of 1864 speakers. Each recording is available as a single WAVE file of 1 s duration with a sample rate of 16 kHz. Our benchmark dataset is based on version 0.02 which provides a total of 105 829 audio files classified into 34 distinct categories.

[2]The raw audio files constituting SC as well as information about the pursued partitioning strategy are available at https://www.tensorflow.org

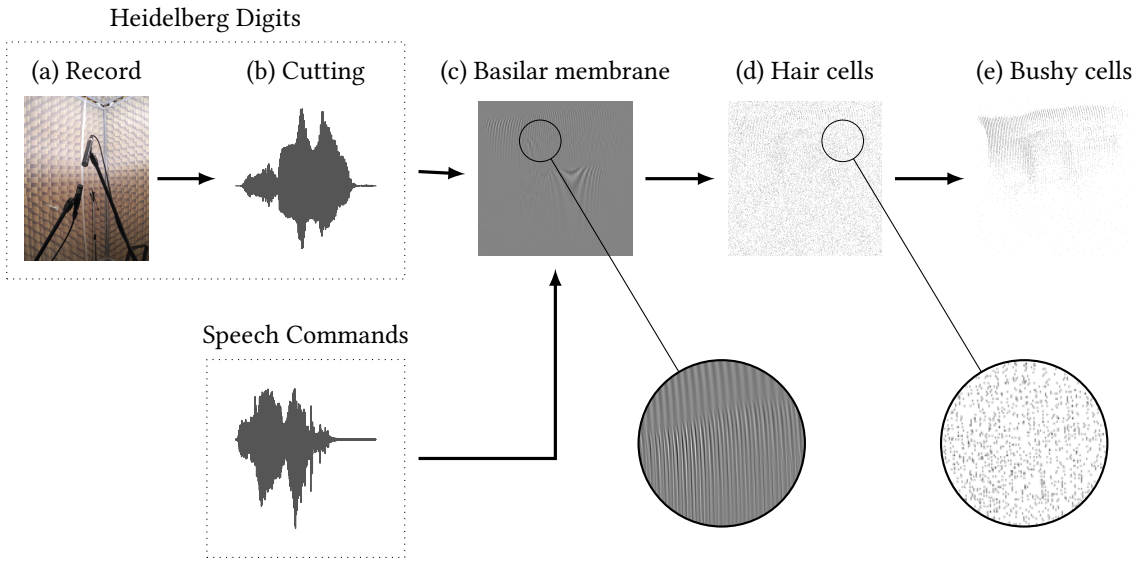


Figure 4.2: Processing pipeline for the Heidelberg Digits (HD) and the Speech Commands (SC) dataset. (a) The HD were recorded in a sound-shielded room. **(b)** Afterwards, the resulting audio files were cut and mastered. **(c)** The HD as well as the SC were fed through a hydrodynamic basilar membrane model. **(d)** Basilar membrane decompositions were converted to phase-coded spikes by use of a transmitter-pool-based hair cell model. **(e)** The phase-locking was increased by combining multiple spike trains of hair cells at the same position of the basilar membrane in a single bushy cell. Figure and caption taken from Cramer et al. (2020b).

24 of these 34 categories make up the command words *Yes*, *No*, *Up*, *Down*, *Left*, *Right*, *On*, *Off*, *Stop*, *Go*, *Backward*, *Forward*, *Follow*, *Learn*, *Zero*, *One*, *Two*, *Three*, *Four*, *Five*, *Six*, *Seven*, *Eight*, *Nine* with about five samples per speaker and word. In addition, only one sample per word and speaker is available for the ten auxiliary words *Bed*, *Bird*, *Cat*, *Dog*, *Happy*, *House*, *Marvin*, *Sheila*, *Tree*, and *Wow*. The partitioning of the samples into training and testing set was performed by the intended hashing function with a share of 80 % and 20 %, respectively (Warden, 2018). For validation purposes, we used 10 % of the samples of the training set.

As for the samples of the HD, we applied a 30 ms Hann window to the on- and offset of each sample's peak normalized waveform. In contrast to the originally intended use of the SC, we consider the top-one-classification of all 34 distinct words. This renders the benchmark more challenging compared to the proposed keyword spotting task which only requires separating the samples into 12 classes (10 keywords, unknown word, and silence). Our choice, however, does not preclude the usage of the derived spike-based benchmark dataset in the keyword spotting scenario.

4.2.2 Spike conversion

We converted the audio files of the datasets described above to the spiking domain by an artificial model[3] of the ascending auditory pathway (Figure 4.2). While this chain of models is inspired by physiological findings, its effect is comparable to commonly used language processing systems (Huang et al., 2001). As a first stage, we reached spatial frequency dispersion by a hydrodynamic

[3]The model used to convert the raw audio signals into spike trains is available at `https://github.com/electronicvisions/lauscher`

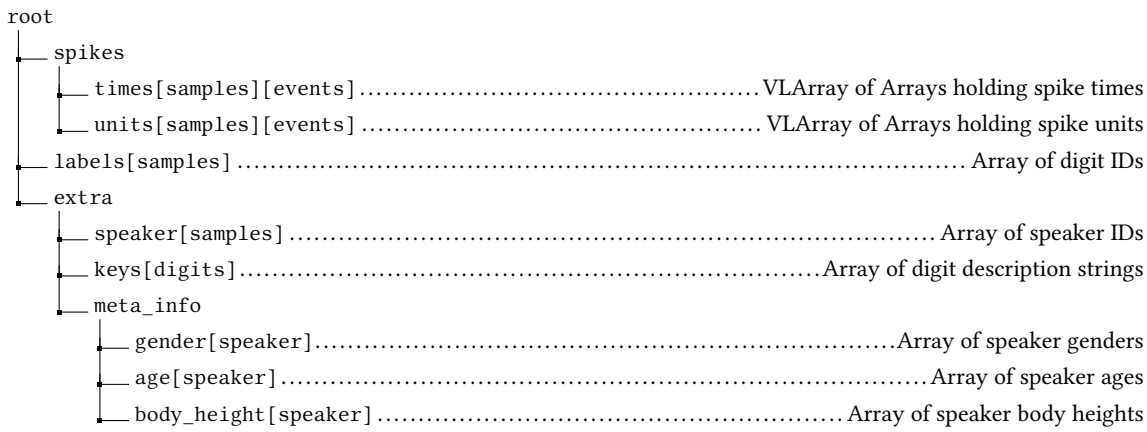


Figure 4.3: HDF5 file organization. For each partition and dataset, we provide a single HDF5 file which holds spikes, digit labels, and additional meta-information. In more detail, each element i in `keys` describes the transformation between the digit ID i and the spoken words. Further, the entry i of each array in `meta_info` corresponds to the information for speaker i. The `meta_info` is only available for SHD. Figure and caption taken from Cramer et al. (2020b).

basilar membrane (BM) model, analogous to the computation of a spectrogram with Mel-spaced filter banks. Next, the resulting channels were converted to time-varying firing rates through a transmitter-pool-based hair cell (HC) model. In an intermediate stage, we generated spike trains from these firing rates with Poisson statistics. Further, we imposed a simple refractory effect upon these spike times by denying any event occurring in a certain time interval τ_{ref} after a spike. Last, we simulated a layer of N_{ch} bushy cells (BCs) to increase phase locking and sparsity (cf. Figure 4.2). Our benchmark datasets Spiking Heidelberg Digits (SHD) and Spiking Speech Commands (SSC) provide the spikes emitted by the layer of BCs. Intuitively, the spikes emitted by each of these BCs code the stimulus information contained within a small frequency band. By resorting to biologically plausible models, all parameters were chosen according to physiological findings, which reduces the number of free parameters (see Table 4.1). All of these outlined model stages are described in detail in Appendix A.1.

4.2.3 Event-based data format

The spikes emitted by the BCs are available in event-based representation in Hierarchical Data Format 5 (HDF5). By this, the overall memory requirements are reduced and download times are kept at a minimum. Moreover, this choice facilitates the usage of the benchmark datasets in most common programming environments. Each partition of each dataset is published as a single file that contains additional meta-information aside from the actual spikes and labels (Figure 4.3).

The spike data of each sample is contained in two separate VLArrays: One for the times and the other one for the associated unit. All corresponding labels – characterizing the digit identity – are stored alongside. The `extra` group holds additional information about the speakers and labels. Specifically, we provide an additional array holding the speaker identity for each sample. Furthermore, the mapping between the label ID in `labels` and the actual spoken word is provided within the `keys` group where the i-th element holds the spoken word for ID i. The last group is only available for the SHD. This `meta_info` provides detailed information about the speakers where the i-th entry of each category holds the information for the speaker with ID i as given in the speaker labels.

Table 4.1: Benchmark model parameters. The BM parameters were taken from Sieroka et al. (2006) and the HC parameter inherited from Meddis et al. (1990). Table and caption adapted from Cramer et al. (2020b).

	Parameter	Symbol	Value
BM model	Damping constant	γ	$0.15\,\mathrm{s}^{-1}$
	Greenwoods constant	a	$35\,\mathrm{kg\,s}^{-2}\,\mathrm{cm}^{-2}$
	Stiffness constant	C_0	$10^9\,\mathrm{g\,s}^{-2}\,\mathrm{cm}^{-2}$
	Fluid density	ρ	$1.0\,\mathrm{g\,cm}^{-3}$
	Attenuation factor	α	$3.0\,\mathrm{cm}^{-1}$
	Height of scala	h	0.1 cm
	Effective mass	m	$0.05\,\mathrm{g\,cm}^{-2}$
	Number of channels	N_{ch}	700
HC model	Input scaling factor	c	$1.0\,\mathrm{s\,cm}^{-1}$
	Permeability offset	A	5
	Permeability rate	B	300
	Maximum permeability	g	1000
	Replenishing rate	y	11.11
	Loss rate	l	1250
	Reuptake rate	r	16 667
	Reprocessing rate	n	250
	Propability scaling	h	50 000
	Number of HCs at same BM position	N_{hc}	40
SNN model	Synaptic time constant	τ_{syn}	$0.5\,\mathrm{ms}^1/10\,\mathrm{ms}^2$
	Membrane time constant	τ_{mem}	$1\,\mathrm{ms}^1/20\,\mathrm{ms}^2$
	Refractory time constant	τ_{ref}	$1\,\mathrm{ms}^1/0\,\mathrm{ms}^2$
	Leak potential	u_{leak}	0
	Reset potential	u_{reset}	0
	Threshold potential	u_{thres}	1
	Number of input neurons	N_{I}	700
	Number of hidden neurons per layer	N_{H}	128
	Number of label neurons	N_{L}	$20^3/34^4$
General	Simulation time step	δt	0.5 ms
	Simulation duration (per input sample)	T	1.0 s
	Number of time steps (per input sample)	N_{T}	2000
Optimization	Training epochs	N_{E}	$300^2/100^4$
	Batch size	N_{B}	256
	Learning rate	η	0.001
	Surrogate gradient steepness	β	100
	Lower bound regularization threshold	θ_{l}	0.01
	Lower bound regularization strength	ρ_{l}	1.0
	Upper bound regularization threshold	θ_{u}	100.0
	Upper bound regularization strength	ρ_{u}	0.06
	First moment estimates decay rate	β_1	0.9
	Second moment estimates decay rate	β_2	0.999
	Stability parameter	ϵ	1.0×10^{-8}

[1] BC parameter [2] SNN parameter [3] SHD parameter [4] SSC parameter

All of these files are publicly available (Cramer et al., 2019) without additional requirements or the need for registration (Footnote 1). To reduce the barrier of entry, we furthermore provide code examples for downloading and visualization of the benchmark data.

4.2.4 Spiking network models for establishing benchmarks

We created the first set of performance references for each of the two spike-based benchmark datasets by training networks of leaky integrate-and-fire (LIF) neurons to classify word identities.

To overcome the binary nature of these SNNs, we resort to surrogate gradients to finally perform backpropagation through time (BPTT) on arbitrary loss functions. In the following section, we give an overview of the visited neuron and synapse models as well as their composition to various topologies. Moreover, we highlight the weight initialization, the supervised learning algorithm and the loss function as well as the regularization techniques.

Neuron and synapse models: We consider LIF neurons with dynamics evolving according to Equation (2.12). For the work presented within this chapter, we directly incorporate the reset term into the differential equation:

$$\frac{\mathrm{d}u_i^{(l)}}{\mathrm{d}t} = -\frac{u_i^{(l)} + u_{\text{leak}} - RI_i^{(l)}}{\tau_{\text{mem}}} + S_i^{(l)}(t)(u_{\text{leak}} - u_{\text{thres}})\,. \tag{4.1}$$

Here, we introduced an additional layer index to denote the membrane potential of neuron i in layer l by $u_i^{(l)}$.

The Equations (2.17) and (4.1) solely determine the neuro-synaptic dynamics. Both of which can be solved by numerical integration based on a regular time grid with n time steps and step size δt over a total duration $T = n \cdot \delta t$. With this, the output spike train $S_i^{(l)}[t]$ of neuron i in layer l at time step t can be expressed as a nonlinear function of the membrane potential:

$$S_i^{(l)}[t] = \Theta\left(u_i^{(l)}[t] - u_{\text{thres}}\right)\,, \tag{4.2}$$

with the Heaviside function Θ. Here, we highlighted the discrete nature of the argument by square brackets. For small time steps δt, the evolution of synaptic currents (Equation (2.17)) is solved by:

$$I_i^{(l)}[t+1] = \kappa I_i^{(l)}[t] + \underbrace{\sum_j w_{ij}^{(l)} S_j^{(l-1)}[t]}_{\text{feed-forward}} + \underbrace{\sum_k v_{ik}^{(l)} S_k^{(l)}[t]}_{\text{recurrent}}\,. \tag{4.3}$$

The first sum extends over all presynaptic partners j and $w_{ij}^{(l)}$ denotes the corresponding afferent weights from the layer below. Recurrent connections within each layer are incorporated by the second sum over the presynaptic partners k within the same layer l and the recurrent weights $v_{ij}^{(l)}$ respectively. The membrane potential in Equation (4.1) can be expressed as:

$$u_i^{(l)}[t+1] = \lambda u_i^{(l)}[t](1 - S_i^{(l)}[t]) + (1-\lambda) I_i^{(l)}[t]\,, \tag{4.4}$$

where we set $u_{\text{leak}} = 0$ as well as $u_{\text{thres}} = 1$ without loss of generality. Further, we used $R = (1-\lambda)$ and defined the decay constants $\kappa := \exp(-\delta t/\tau_{\text{syn}})$ and $\lambda := \exp(-\delta t/\tau_{\text{mem}})$.

Network model: The spike trains emitted by the $N_{\text{ch}} = 700$ BCs – in the following denoted by $S_i^{(0)}$ – were used to stimulate the actual classification network. In the context of this work, we trained both feed-forward and recurrent networks with hidden layers containing $N_{\text{H}} = 128$ LIF neurons each. For all network architectures, the last layer was accompanied by a linear readout ($l = L$) consisting of N_{L} leaky integrators which did not spike.

Weight initialization: The weights $w_{ij}^{(l)}$ and $v_{ij}^{(l)}$ of all our networks were initialized according to Kaiming's uniform scheme (He et al., 2015). Specifically, the initial values were drawn independently from a uniform distribution given by $\mathcal{U}(-1/\sqrt{N}, 1/\sqrt{N})$ with the number of efferent connections, i. e. N_{ch} or N_{H}, respectively. This choice ensures suitable activation prior to learning.

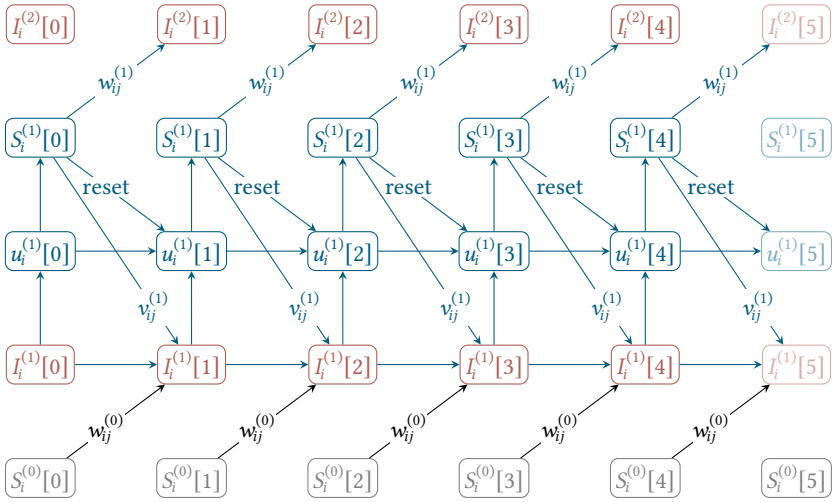


Figure 4.4: Computation graph of an SNN in discrete time. Time evolves from left to right. Input spikes $S_i^{(0)}$ enter the network from the bottom and propagate upwards to higher layers. On the one hand, the synaptic currents $I_i^{(1)}$ decay from one time step to the next and on the other hand they supply the membrane potentials $u_i^{(1)}$. The latter similarly decay over time. Spikes $S_i^{(1)}$ are generated by a threshold criterion in each time step. Upon spike emission, the membrane potential is reset within the next time step. The accruing spikes propagate to a downstream layer as well as recurrently within the same layer to build up the respective synaptic currents.

Gradient-based learning: We trained our networks by minimizing a loss function $\mathscr{L}$ formulated on the activation of the readout layer. More specifically, we modified the SNN parameters by performing gradient descent:

$$\theta \leftarrow \theta - \eta \frac{\partial \mathscr{L}}{\partial \theta} = \theta - \eta \sum_t \frac{\partial \mathscr{L}[t]}{\partial \theta}\,, \tag{4.5}$$

with a learning rate η and the network parameters θ, i. e. the feed-forward as well as the recurrent weights, $w_{ij}^{(l)}$ and $v_{ij}^{(l)}$, respectively. The interplay of the observables of multi-layer SNNs incorporating recurrence can be illustrated by a computation graph that can be constructed based on the description of neuro-synaptic dynamics given by the Equations (4.2) to (4.4) (Figure 4.4). With regard to this graph, the optimization of the loss function $\mathscr{L}$ requires to solve a spatio-temporal credit assignment problem which can be achieved by gradient descent and the application of the chain rule by drawing on the BPTT algorithm (Werbos, 1990). For the full recursive relationship of BPTT for recurrent SNNs we refer to Zenke & Neftci (2021). However, it should be noted that the calculation of the underlying partial derivatives involves the computation of $\sigma' = \partial S_i^{(l)} / \partial u_i^{(l)}$ which only deviates from zero at the time of a spike (cf. Equation (4.2)). Here, we overcame this binary nature of SNNs and the associated crucial points of spike emergence and disappearance under parameter changes by introducing a surrogate for the derivative of the spike σ' (Neftci et al., 2019a; Bellec et al., 2020; Zenke & Vogels, 2021). Specifically, we replaced the Heaviside activation function (Equation (4.2)) by a fast sigmoid (Figure 4.5) in the backward pass:

$$\sigma(u_i^{(l)}[t]) = \frac{\beta\left(u_i^{(l)}[t] - u_{\text{thres}}\right)}{1 + \beta\left|u_i^{(l)}[t] - u_{\text{thres}}\right|} + 1\,, \tag{4.6}$$

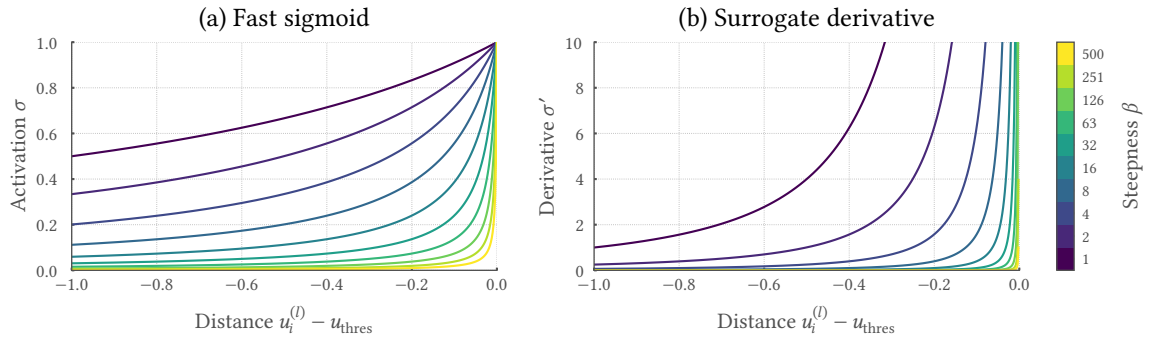


Figure 4.5: Illustration of the SuperSpike surrogate derivative To train SNNs with gradient-based methods, the true derivative of the neuronal activation function in the backward pass is replaced with a surrogate. **(a)** Specifically, a fast sigmoid σ is used as activation function within the backward pass. Here, σ is shown as a function of the difference between the membrane potential of neuron i in layer l, $u_i^{(l)}$, and the threshold potential u_{thres}. **(b)** The associated surrogate derivative σ' takes on the form of the SuperSpike non-linearity (Zenke & Ganguli, 2018b). The steepness of the activation function and in turn the surrogate derivative can be controlled by the parameter β.

where we introduced a positive steepness parameter β (Figure 4.5). Hence, the partial derivative of Equation (4.2) occurring in the recursion relation of BPTT was replaced by:

$$\sigma'(u_i^{(l)}) = \frac{\partial S_i^{(l)}[t]}{\partial u_i^{(l)}[t]} = \frac{1}{\left(\beta \cdot |\tilde{u}_i^{(l)}[t] - u_{\text{thres}}|\right)^2} . \tag{4.7}$$

This surrogate derivative was originally invented by Zenke & Ganguli (2018b).

Within this work, we performed BPTT by relying on standard auto-differentiation tools. To that end, the surrogate derivative in Equation (4.7) was implemented in PyTorch (Paszke et al., 2017) by overloading existing auto-differentiation capabilities (Listing 2). An instructive example of this method can be found online[4]. For the results shown in this chapter, we applied the Adamax optimizer (Kingma & Ba, 2014) for the minimization of the loss functions described in the following.

Loss functions: We quantified the success of our SNNs by a cross-entropy loss evaluated on the activity of the readout layer $l = L$. More specifically, after stimulation with the input spikes $S_{i,s}^{(0)}$ constituting a batch composed of N_{B} samples:

$$\left\{ (S_{i,s}^{(0)}[t], y_s) \,|\, s = 1, ..., N_{\text{B}} \,;\; y_s \in \{1, ..., N_{\text{C}}\} \right\} , \tag{4.8}$$

and N_{C} classes, we considered a negative log-likelihood loss based on the membrane potentials of the readout neurons evaluated by a softmax function:

$$\mathscr{L}\left(u_{i,s}^{(L)}, y_s\right) = -\frac{1}{N_{\text{B}}} \sum_{s=1}^{N_{\text{B}}} \mathbb{1}(i = y_s) \cdot \log \left(\frac{\exp\left(u_{i,s}^{(L)}[\tilde{t}_{i,s}]\right)}{\sum_{i=1}^{N_{\text{C}}} \exp\left(u_{i,s}^{(L)}[\tilde{t}_{i,s}]\right)} \right) , \tag{4.9}$$

with the indicator function $\mathbb{1}$. Here, we introduced an additional sample index s for both the spikes as well as the membrane potentials. For the establishment of a first set of benchmarks on SHD

[4]Instructive examples of the implementation of surrogate gradients in PyTorch can be found at https://github.com/fzenke/spytorch

```python
class SuperSpike(torch.autograd.Function):
    beta = 80.0

    @staticmethod
    def forward(ctx, v: torch.Tensor):
        ctx.save_for_backward(v)
        return torch.gt(v, torch.as_tensor(0.0)).to(v.dtype)

    @staticmethod
    def backward(ctx, grad_output):
        v, = ctx.saved_tensors
        grad_input = grad_output.clone()
        grad = grad_input / (SuperSpike.beta * torch.abs(v) + 1.0) ** 2
        return grad, None

spike_fn  = SuperSpike.apply
```

Listing 2: Implementation of surrogate gradients in PyTorch. Surrogate gradients can be implemented in PyTorch by overloading existing auto-differentiation capabilities. In the forward pass, the step function is applied as activation function. The `ctx` object is utilized to stash information with the `ctx.save_for_backward` method for the later backpropagation of error signals. In the backward pass, the neuronal activation function is replaced by the normalized negative part of a fast sigmoid.

and SSC, we trained our SNNs based on two distinct choices for $\tilde{t}_i$ (Figure 4.8). First, we used the time step with maximal membrane potential deflection for each readout unit $\tilde{t}_{i,s} = \arg\max_t u_{i,s}^{(L)}[t]$ resulting in a *max-over-time* loss function. Second, we utilized the last time step T of each readout unit $\tilde{t}_{i,s} = T$. The latter choice resulted in a *last-time-step* loss.

Regularization: The loss function in Equation (4.9) can be additively augmented by regularization terms to shape the activity of the SNNs. Here, we employed a penalty acting on the mean population activity:

$$\mathscr{L}_{\mathrm{u}}\left(S_{i,s}^{(\mathrm{l})}\right) = \rho_{\mathrm{u}} \frac{1}{N_{\mathrm{B}}} \sum_{s=1}^{N_{\mathrm{B}}} \left[\max\left\{0, \frac{1}{N_{\mathrm{H}}} \sum_{i=1}^{N_{\mathrm{H}}} \sum_{n=1}^{T} S_{i,s}^{(l)}[n] - \theta_{\mathrm{u}} \right\} \right]^2 , \tag{4.10}$$

with strength ρ_{u}, and threshold θ_{u}. This first contribution prevents pathologically high activities and hence enforces sparse spatio-temporal activity. In a second term, we rewarded spiking activity by a per neuron lower threshold regularization:

$$\mathscr{L}_{\mathrm{l}}\left(S_{i,s}^{(\mathrm{l})}\right) = \rho_{\mathrm{l}} \frac{1}{N_{\mathrm{B}}} \sum_{s=1}^{N_{\mathrm{B}}} \frac{1}{N_{\mathrm{H}}} \sum_{i=1}^{N} \left[\max\left\{0, -\frac{1}{T} \sum_{n=1}^{T} S_{i,s}^{(l)}[n] - \theta_{\mathrm{l}} \right\} \right]^2 , \tag{4.11}$$

with strength ρ_{l}, and threshold θ_{l}. This latter penalty especially prevented the problem of vanishing gradients in deep networks by ensuring activity.

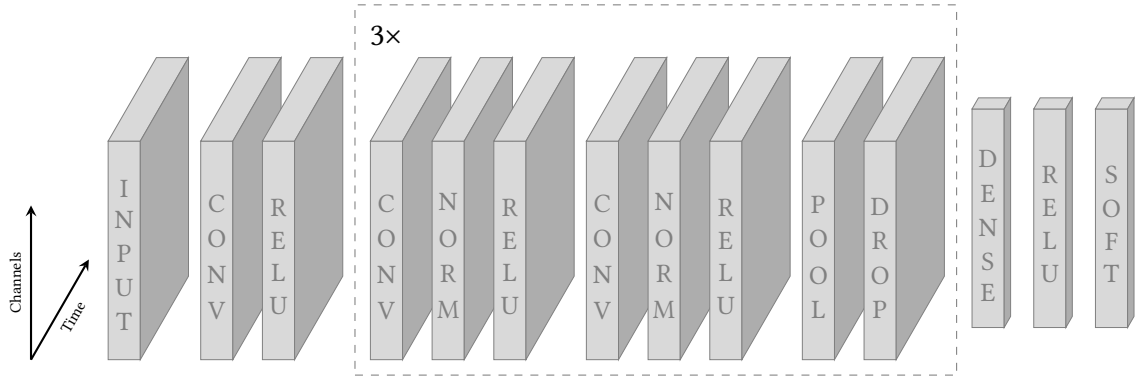


Figure 4.6: Schematic view of the deep CNN architecture. For validation purposes, we trained deep CNNs on spatio-temporal histograms of the spike times of both SHD and SSC. To that end, we formulated a categorical cross-entropy loss based on the activation of the last layer of the shown CNN which was in turn optimized with the Adamax optimizer. The CNN was implemented in *Tensorflow* with the *Keras* API.

4.2.5 Non-spiking classifiers for establishing benchmarks

We trained three standard non-spiking classifiers on our newly created benchmarks to assess their separability and to investigate the information carried by spike times. To this end, we consider results achieved by support vector machines (SVMs), long short-term memories (LSTMs), and convolutional neural networks (CNNs), which are described in the following.

Support vector machines: We trained a set of SVMs on activity vectors of both datasets. Aside from linear, we considered radial basis function (RBF) as well as polynomial kernel functions. The latter were visited up to the third degree. To inject the data into the SVMs, we generated N_{ch}-dimensional activity vectors $a_{i,s}$ for each sample by constructing spatial histograms of the spikes emitted by each BC i for sample s, $t^k_{i,s}$, i. e. $a_{i,s} = \sum_k t^k_{i,s}$. In more detail, these activity vectors no longer had a temporal dimension. All resulting $a_{i,s}$ were corrected for the mean and scaled to their unit variance. We trained SVMs using the *scikit-learn* package (Pedregosa et al., 2011).

Long short-term memories: For validation purposes, we applied LSTMs explicitly relying on temporal information within the data (Hochreiter & Schmidhuber, 1997). Specifically, we trained our LSTMs on the activity:

$$a_{i,s}[t] = \sum_k \mathbb{1}\left(t^k_{i,s} \geq t \cdot \delta t,\, t^k_{i,s} < (t+1) \cdot \delta t\right), \tag{4.12}$$

estimated with a temporal bin size of $\delta t = 10\,\text{ms}$. These activity patterns were injected into a single LSTM layer with 128 units. The dropout probabilities for both the linear transformation of the input as well as the recurrent states were set to 20 %. We connected a readout layer with softmax activation in series and formulated a categorical cross-entropy loss which was minimized with the Adamax optimizer (Kingma & Ba, 2014). Like for SNNs, the latter was formulated based on the last time step of each sample or the time step of maximal deflection respectively. For the implementation of LSTM networks we made use of *TensorFlow 1.14.0* with the *Keras 2.3.0* application programming interface (API) (Abadi et al., 2015; Chollet et al., 2015). All layers were initialized with their default values unless mentioned otherwise.

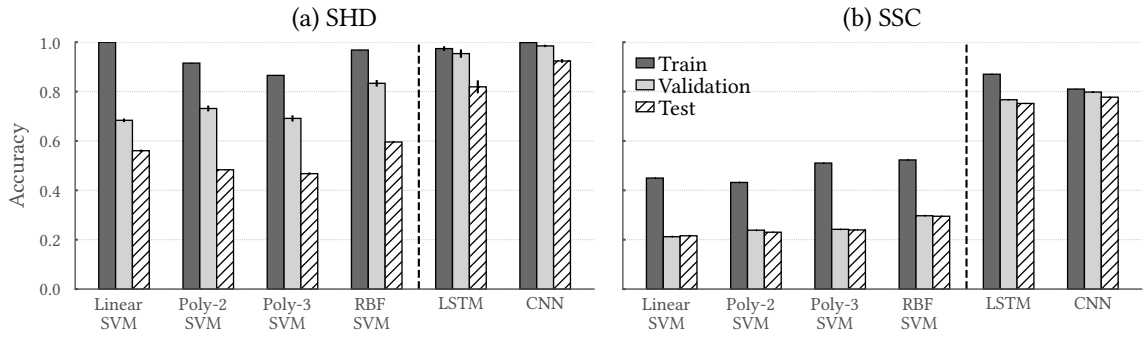


Figure 4.7: Temporal information is essential for the achievement of high classification accuracy. **(a)** Classifiers incorporating temporal information outperform rate-based architectures on the SHD. Here, SVMs are trained on spike count vectors whereas LSTMs as well as CNNs draw on spatio-temporal histograms of the original spike trains. LSTMs and CNN tend to reduce overfitting and outperform all tested SVMs by a large margin. **(b)** Temporal information also proves beneficial for the larger SSC dataset. Likewise, LSTM and CNN architectures promote highest accuracy on the SSC. Figure taken from Cramer et al. (2020b).

Convolutional neural networks: The separability of both benchmarks was further assessed by training deep CNNs. Like for the LSTMs, we stimulated the input layer by spatio-temporal activity histograms with a temporal binwidth of $\delta t = 10\,\text{ms}$. Moreover, we condensed neighboring input spike trains according to:

$$a_{j,s}[t] = \sum_{i=1}^{\delta i} a_{j\cdot\delta i+i,s}[t]\,, \tag{4.13}$$

with $\delta i = 11$. The topology of the CNN was made up of several blocks (Figure 4.6): First, the stimuli were processed by a 2D convolutional layer with 32 filters of size 11×11 and rectified linear units (ReLUs). The output of this first block was processed by a sequence of three equal blocks. Each of these blocks encompassed two 2D convolutional layers, each of them individually combined by batch normalization and ReLUs. The convolutional layers hold 32 filters of size 3×3. This sequence was completed by a 2D max-pooling layer with pool size 2×2 and a dropout layer with a dropout probability of 20 %. Finally, a dense layer with 128 ReLUs provided input to a softmax readout. We again formulated a categorical cross-entropy loss which was minimized with the Adamax optimizer (Kingma & Ba, 2014). This CNN topology was implemented and trained in *Tensorflow 1.14.0* by drawing on the *Keras* API. All layers were initialized with their respective default parameters unless mentioned otherwise.

4.3 Results

With the publication of both spike-based datasets – Spiking Heidelberg Digits (SHD) and Spiking Speech Commands (SSC) – under a *Creative Commons By 4.0* license, we already satisfy parts of our minimum requirements for a valuable benchmark dataset. By providing code snippets, we further reduce the entry hurdle and thereby hopefully attract the interest of a broader community. In the following, we establish the first set of benchmarks to test the generality of our datasets and more importantly to assess the separability by current methods. To that end, we first consider non-spiking classifiers and then move on to SNNs.

4.3.0.1 Training non-spiking classifiers

Rate-based versions of our novel datasets can not be solved accurately by current methods. Moreover, these reduced datasets allow us to investigate the impact of temporal information on classification performance. To that end, we trained a set of SVMs on spike count patterns of each sample, which by design lack temporal information. When using a linear kernel function, the SVM already suffers from overfitting resulting in a test performance of only (56.0 ± 0.4) % on the SHD (Figure 4.7a). In contrast, the same SVM trained on the SSC is less prone to overfitting, but only reached an overall test accuracy of (21.6 ± 0.0) % (Figure 4.7b). Hence, both datasets can not be accurately solved by a simple linear model. Non-linear kernel functions only slightly improve upon these results. Specifically, SVMs with RBF kernels lead to the highest performance with an overall test accuracy of (60 ± 3) % on the SHD and (29.5 ± 0.0) % on the SSC (Figure 4.7a and b). Most notably, SVMs with polynomial kernel functions are not able to improve upon the results reached with linear SVMs. In summary, for all tested kernels, we are not able to surpass the 60 % accuracy mark on the SHD and the 30 % mark on the SSCs. Hence, our newly created datasets can not be accurately classified based on plain firing rates. Particularly, the generalization to the SHD test remains a major challenge.

The design of the SHD dataset allows us to make well-founded statements about generalization to new unseen speakers. Specifically, generalization is stressed by the test set which exclusively contains all samples of two speakers. Because of this, the ability to generalize to new speakers can be quantified by comparing the performance reached on a uniformly drawn validation set with the one observed on the test set. Particularly, the SVMs with polynomial as well as RBF kernel generalize worse than the ones with linear kernels (Figure 4.7a). In contrast, there is no noticeable difference between the accuracies reached on the validation and the test set of the SSC. This is a direct consequence of the uniform partitioning of the samples, which, however, stem from a much larger pool of speakers.

Leveraging the temporal information within both datasets improves the classification performance in general and the generalization in particular. More specifically, the performance on both benchmark datasets and especially the ability to generalize can be improved when considering classifiers with explicit access to temporal information within the samples. To that end, we trained LSTMs on spatio-temporal histograms of the spikes (Section 4.2.5). Despite the relatively small amount of samples in the SHD dataset, LSTMs are less prone to overfitting and clearly outperform the results of all tested SVMs with an accuracy of (89 ± 2) % (Figure 4.7a). A similar improvement can be observed on the SSC. Here, LSTMs reach a test accuracy of (73.0 ± 0.1) %, which is more than twice as high as the best-performing classifier on the spike count data. However, the degree of overfitting on the SSC is slightly higher than on the SHD when only considering the uniform partitioning of samples within both validation sets.

CNNs generalize best to new speakers and novel samples. Since the performance of LSTMs still suffers from overfitting, we trained deep CNNs. Like for LSTMs, spatio-temporal histograms of spikes were used to stimulate the first layer of the CNN (Section 4.2.5). Indeed, these networks generalize best among all tested architectures (Figure 4.7). In more detail, the test accuracy is only 1.4 % less than the training accuracy on the samples of the SHD. Most notably, the performance on the SHD test set is comparable to the one reached on a uniformly drawn validation set. For the SSC we observe a drop of only 1.5 %. In absolute terms, we reach an accuracy of (92.4 ± 0.7) % on the SHD and (77.7 ± 0.2) % on the SSC.

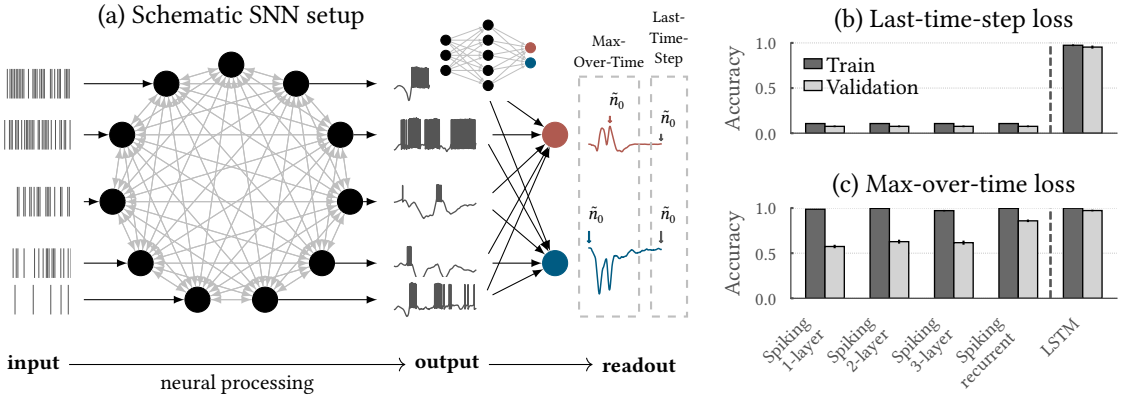


Figure 4.8: Dedicated loss functions bridge the gap between time scales of neuro-synaptic dynamics and speech signals. **(a)** The flexibility of the surrogate gradient approach allows to formulate different loss functions based on the output of the linear readout units. Here, we employ two different strategies for the evaluation of a cross-entropy loss: First, we only consider the time step with maximal activity of each readout unit (colored arrows) and second, we utilize the activation of the very last time step (gray arrows). Shown is a schematic illustration of two readout units receiving input from an SNN with a single hidden layer incorporating recurrent connections. The inset of the panel illustrates the corresponding feed-forward topology. **(b)** The performance of all tested SNN architectures suffers from training with a last-time-step loss function. Only the LSTM is able to maintain relevant stimulus information until the last time step. Shown is the performance on the SHD dataset. **(c)** The performance of all tested architectures profits from a max-over-time loss. Particularly, the accuracy on the SHD reached by SNN classifiers substantially increases compared to the aforementioned training with a last-time-step loss. Figure taken from Cramer et al. (2020b).

All of these results do not only serve as a performance baseline, but also demonstrate how temporal information present in both datasets could be leveraged to improve classification performance. For the sake of completeness, it should be mentioned that a more sophisticated hyperparameter optimization and architecture search can improve upon our results. However, both datasets seem to be useful contributions to promote quantitative comparisons between different SNNs architectures as well as training mechanisms up to at least these empirical accuracy values.

4.3.0.2 Training spiking neural networks

The results presented above demonstrate that our benchmark datasets contain essential temporal information, which is required to accurately classify the data. Next, we will investigate whether this information is also accessible to SNNs. To that end, we make use of surrogate gradient learning to overcome the binary nature of SNNs and to finally perform gradient-based learning on a supervised loss function by BPTT (Neftci et al., 2019a). The associated surrogate derivative replaces the true gradient within the backward pass and can be interpreted as a continuous relaxation of the true gradient. Thus, the forward pass remains unchanged and only the backward pass is modified (Section 4.2.4). With this, we are furthermore able to establish the first set of benchmarks for SNNs.

Inspired by biology, we consider SNNs with finite neuronal and synaptic time constants on the order of milliseconds, although this is generally not required for the framework of surrogate gradient learning. The finite time constants stay in stark contrast to the previously considered LSTMs. The main difference is that LSTMs are equipped with a memory on arbitrary time scales, which promotes the training with a cross-entropy loss based on the last time step of each sample. Specifically, this

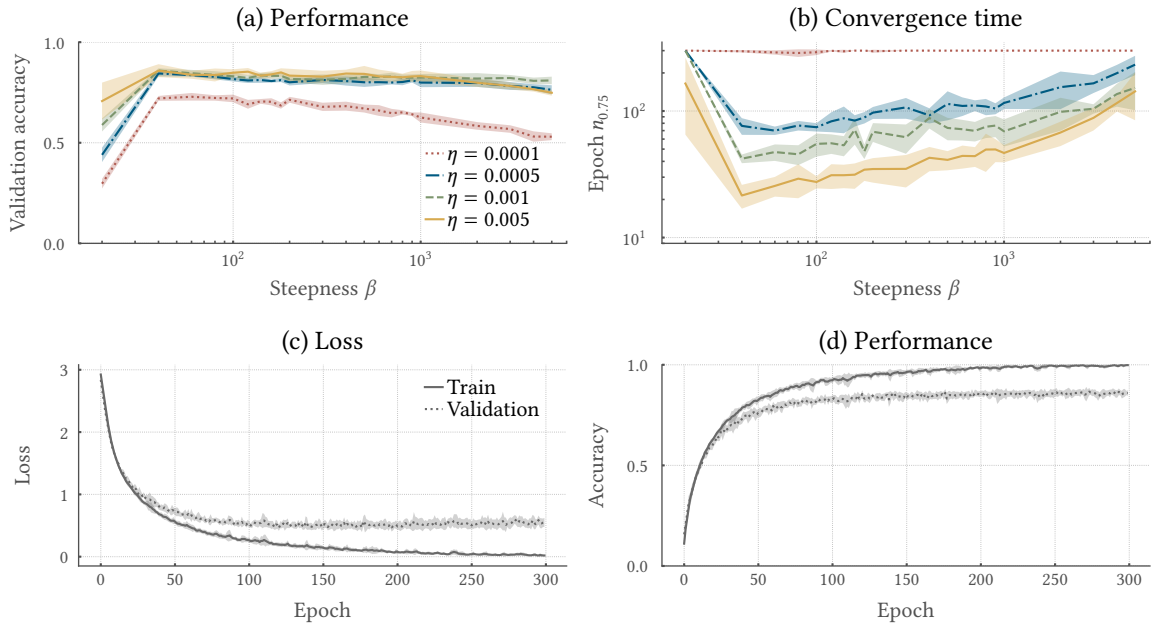


Figure 4.9: The steepness of the surrogate derivative mainly impacts convergence time. A grid search of the steepness β and the learning rate η reveals an optimal parameter combination for fast convergence to a solution with high accuracy on the SHD dataset. **(a)** The accuracy reached by a recurrent SNN on a SHD validation set is high for a broad range of β values and only shows a slight dependence on η. **(b)** Particularly, low values of β and hence shallow surrogate derivatives promote fast convergence. The number of epochs required to reach an accuracy larger than 0.75, $n_{0.75}$, is lowest for $\beta = 40$. **(c)** For the parameter combination $\beta = 40$ and $\eta = 10^{-3}$, SNNs quickly converge. **(d)** With this parametrization, the training data could be fitted and SNNs generalize well on a validation set of SHD indicated by high accuracy levels. Figure taken from Cramer et al. (2020b).

loss requires the maintenance of information over the entire course of a stimulus until the last time step (Figure 4.8a). An SNN composed of LIF neurons with finite time constants, however, needs to implement the required memory by reverberating activity and hence with its dynamics. To relax this constraint, we furthermore use a *max-over-time* loss, which considers only the time step with maximum activation of each readout unit (Figure 4.8a). This loss function is inspired by the Tempotron (Gütig & Sompolinsky, 2006), which signals its decision about the class membership by whether a readout neuron emitted a spike at an arbitrary point in time or remained silent. With this loss function, the time scales between neuro-synaptic dynamics and speech signals could be bridged.

SNNs have difficulties with the maintenance of information over long time spans. We trained LSTMs as well as various SNN topologies in combination with both aforementioned loss functions on the SHD. While LSTMs accurately classify the dataset irrespective of the chosen loss function, SNNs only perform well when being trained with a *max-over-time* loss (Figure 4.8b and c). Most notably, the recurrent SNN slightly outperform the feed-forward architectures in combination with a *last-time-step loss*, underpinning the need for memory. Within the recurrent SNN, the latter is likely to be implemented by reverberating activity through recurrent connections. In summary, all network architectures and topologies – including the LSTM – performed best when being trained with a *max-over-time* loss. Because of this, we employ this loss function throughout the remainder of this chapter.

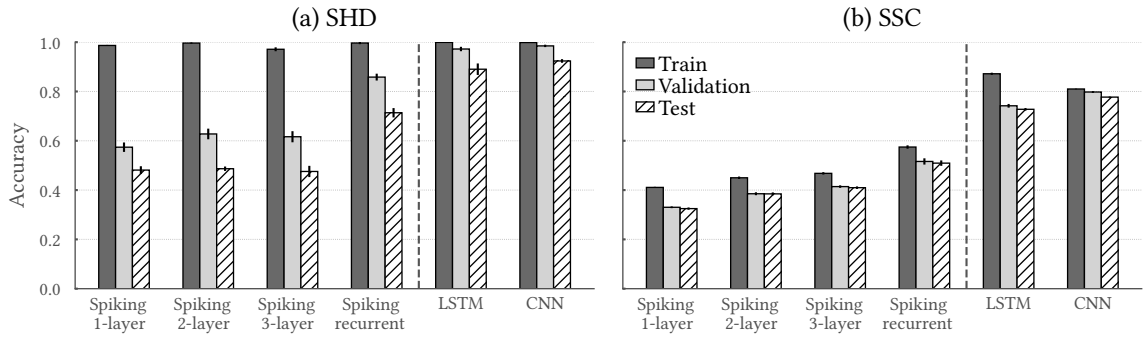


Figure 4.10: Recurrent SNNs outperform feed-forward architectures on both datasets. (a) The accuracy reached by recurrent SNNs on the SHD dataset is comparable to the performance of LSTMs with a max-over-time loss. Among all tested spiking architectures, recurrent topologies promote highest performance. Increasing the number of layers in feed-forward topologies hardly affects performance. **(b)** The performance on the larger SSC dataset reached by all SNNs is significantly lower than the one obtained by LSTMs and CNNs. In contrast to networks trained on SHD, an increasing number of layers leads to a monotonic increase of accuracy on the SSC. Figure taken from Cramer et al. (2020b).

Our surrogate gradient learning entails a new hyperparameter β, which controls the steepness of the surrogate derivative (Equation (4.7)). Particularly, β plays a crucial rule for convergence similar to the learning rate η. Most notably, a single layer recurrent SNN shows high performance on the SHD for a plethora of both parameter values (Figure 4.9a). In more detail, only a slight decrease of accuracy can be observed for high values of β, whereas a dramatic drop only occurs for very small β. The learning rate, in contrast, has a minor impact on the performance for the tested values. Aside from the peak performance, the number of epochs required for a given network to converge plays a crucial role in efficient training. Here, convergence speed heavily depends on both β and η (Figure 4.9b). The combined observations for peak performance and convergence speed let us to choose $\beta = 40$ and $\eta = 1 \times 10^{-3}$ for all SNN architectures within this chapter. For this parameter set, the peak performance is reached after about 150 training epochs (Figure 4.9c and d). After this point, additional training only improves training accuracy, but does not impact generalization. It is noteworthy that the considered recurrent SNN already leads to results comparable to the performance of LSTMs. The benchmarks for feed-forward architectures, however, have yet to be established.

Recurrent architectures outperform feed-forward networks on both benchmark datasets. With the described set of hyperparameters, we trained multi-layer feed-forward architectures with up to three hidden layers as well as a network with a single hidden layer but recurrent connections. Among all investigated topologies, the recurrent networks performs best with a final test accuracy of $(71.4 \pm 1.9)\,\%$ on the SHD and $(50.9 \pm 1.1)\,\%$ on the SSC (Figure 4.10). However, the generalization of recurrent SNNs falls back behind LSTMs as well as CNNs. Moreover, the accuracy on the SHD test set is almost independent of the number of hidden layers. A monotonic increase of performance with increasing layer count can, however, be observed for feed-forward networks trained on the SSC. Despite this rise, feed-forward architectures are not even able to surpass the benchmark of the best performing SVM on the SSC (Figure 4.7b).

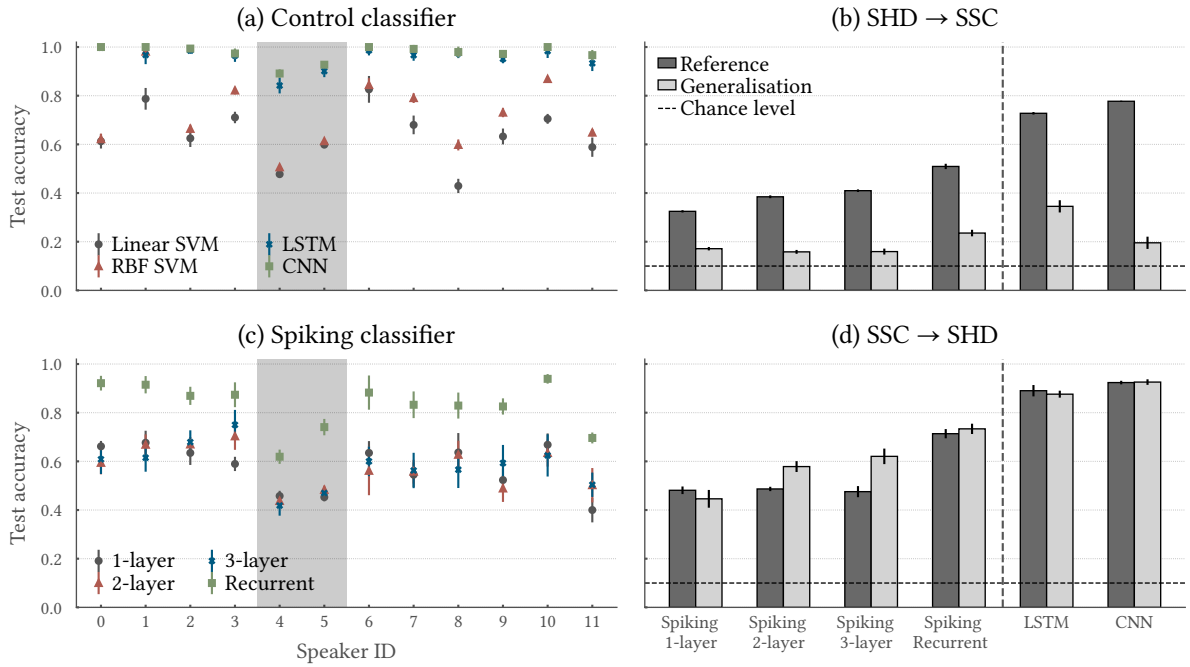


Figure 4.11: Trained networks generalize across speakers and datasets. The design of the SHD test set allows to quantify generalization to new speaker, since the samples spoken by the speakers four and five exclusively occur in the test set. **(a)** Among all tested non-spiking classifiers, CNNs generalize best on the samples of the aforementioned unseen speakers. However, a clear decrease in performance is observable for all classifiers on the samples spoken by the held-out speakers four and five (highlighted). Likewise, the generalization to new speakers can be assessed by evaluating the performance of networks trained on one dataset and evaluated on the common subset of the other one. **(b)** Networks trained on the SHD dataset do not necessarily generalize to the English digits of the SSC. For all classifiers, the accuracy on the SSC digits (generalization) is substantially lower than on the digits of the SSC test set (reference). It is noteworthy that despite of this drop in accuracy, all networks performed above change level. **(c)** Among all tested spiking classifiers, recurrent architectures lead to highest performance on the unseen speakers of the SHD test set. **(d)** Networks trained on the SSC generalize well to the English digits of the SHD dataset. Figure taken from Cramer et al. (2020b).

4.3.0.3 Quantifying generalization across speakers and datasets

Before deploying an implementation for the classification of spoken words, its ability to generalize to new data needs to be carefully investigated. Particularly, the generalization to novel previously unseen speakers is indispensable for the robust classification of spoken words in real-world scenarios, e. g. on mobile devices. To that end, a performance evaluation based on a uniform partitioning of all available samples may not be sufficient. Our newly generated benchmark datasets allow us to quantify generalization to novel speakers in a more fine-grained manner. This can be achieved most easily by resorting to the SHD: Since the test set of the SHD exclusively contains the samples of two speakers, the generalization to new speakers can be investigated by simply evaluating the test performance. By comparison to the accuracy reached on a uniformly drawn validation set, the ability to generalize can be quantified.

Our tested architectures suffer from a decline of performance on the SHD test set compared to a uniform validation set. In more detail, the accuracy on the samples of the unseen speakers four and five is systematically lower compared to the remaining samples (Figure 4.11a and c). When

considering SVMs only, the linear kernel shows the smallest decline of about 18 %. In contrast, we observe a drop of 26 % for the RBF kernel function. The best overall generalization performance is achieved by the CNNs characterized by a decrease of only 8 % closely followed by the LSTMs with reduction of 10 %. With regard to SNNs, recurrent topologies generalize best with a decline of 21 %. The feed-forward topologies, however, undergo drops in the range from 24 % to 27 % on the samples of the unseen speakers. As a result, the composition of the SHD test set does not only pose a challenge to existing implementations, but also allows the investigation of generalization across speakers of a given algorithm.

The uniform partitioning of samples in the SSC prohibits the quantitative assessment of generalization to novel speakers. In general, the SSC also provide speaker information and hence allow for a comparable assignment of samples as done for the SHD. This, however, would involve a new partitioning of samples into train and test sets and hence a deviation from the scheme proposed by Warden (2018). Nevertheless, the generalization to new speakers can still be quantified by evaluating networks trained on the SSC with the shared English digits of the SHD and vice versa. Due to the purpose of both datasets, this furthermore allows making statements about the impact of noise present during training and inference respectively. As a result, networks trained on the SSC generalize better to the samples of SHD. This result, on the one hand, hints towards the favorable role of noise and on the other hand, emphasizes the valuable impact of the high number of samples constituting the SSC. Among all tested architectures, CNNs and recurrent networks promote the highest performance. Overall, it can be stated that all networks performed significantly above the chance level when generalizing to novel datasets.

The generalization across datasets also constitutes a first step towards the quantification of the impact of noise on classification. It is noteworthy that the high-fidelity recordings of the HD allow us to investigate the influence of noise on training and inference in more fine-grained scenarios. Here, the signal-to-noise ratio can be precisely controlled by imprinting noise with well-defined amplitude to the raw audio signals. The SSC already come at a higher noise level and are hence inappropriate for this application.

4.3.0.4 Strategies to improve generalization performance

The assessment of generalization in the previous section suggests the exploration of methods to increase the overall network capacity. To that end, we consider the network size as well as the scale of model time constants. For an exemplary recurrent SNN trained on the SHD, all of the aforementioned methods improve on the result obtained previously. While even small networks are already able to almost fit the training data, an increased size nevertheless enhances test accuracy up to (76.5 ± 1.0) % for a recurrent hidden layer with 1024 LIF neurons (Figure 4.12a). To tie on the discussion of finite time constants, we scaled both neural as well as synaptic time constants simultaneously. For a scaling factor of 4 , we significantly reduced overfitting and achieved a test accuracy of (79.9 ± 2.8) % (Figure 4.12b). This choice corresponds to the time constants $\tau_{\text{mem}} = 80\,\text{ms}$ and $\tau_{\text{syn}} = 40\,\text{ms}$. Most notably, this approach allowed us to almost catch up with the results achieved with LSTMs of similar size (Table 4.2). A comparable approach could be taken by accelerating the stimuli along the temporal dimension. This method has the potential to further reduce the computational overhead since a reduced number of time steps needs to be evaluated. In summary, an enlargement of the network size as well as increased time constants allow us to improve upon previous results.

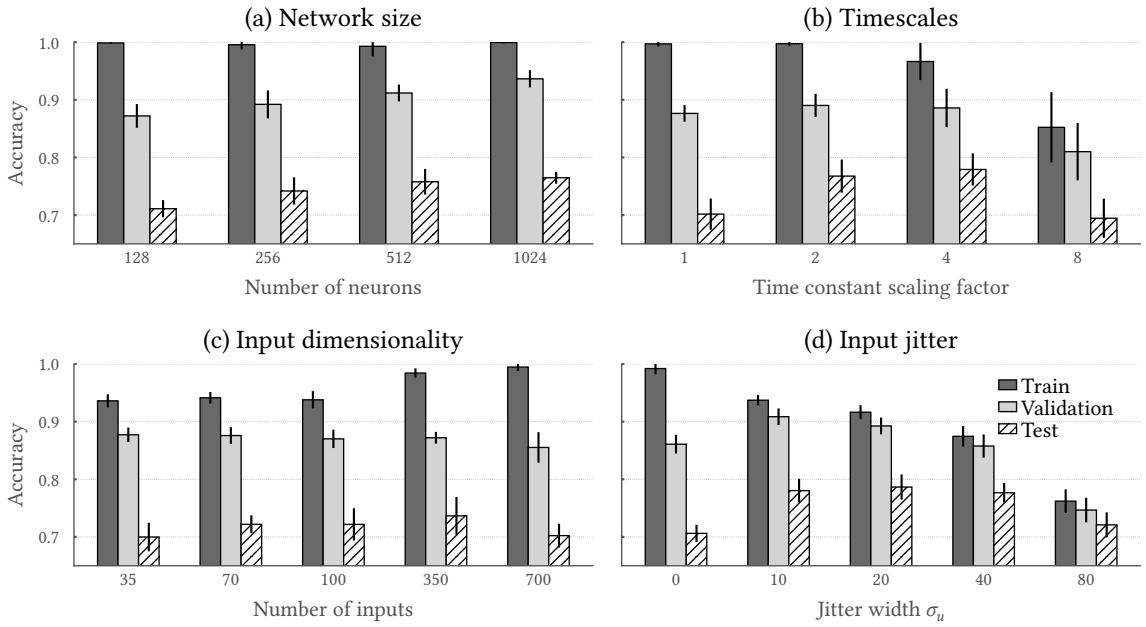


Figure 4.12: Generalization performance improves with network size, input reduction, time constants as well as data augmentation. **(a)** The performance achieved with recurrent SNNs on the SHD profits from increasing network sizes. While already small networks readily fit the training data, the test accuracy still increases with the network size. **(b)** Increased time constants further promote generalization. Highest validation performance is reached when scaling synaptic as well as membrane time constants by an expansive factor of 4. **(c)** Condensing the spikes of neighboring input channels slightly improves generalization performance. A maximum validation accuracy is achieved when the spikes of the 700 input units are compressed to only 70 units. **(d)** Likewise, generalization can be improved by augmenting the training data with Gaussian spatial noise (channel jitter). The validation accuracy peaks for a standard deviation of $\sigma_u = 20$ channels. Figure taken from Cramer et al. (2020b).

Generalization performance can be further enhanced by compressing and augmenting the input data. As already showcased for the CNN, a compression of the input promotes high performance. To likewise exploit this finding for SNNs, we merged the spikes of neighboring channels of the input spike trains. We find a plethora of increased accuracy of about 73 % when merging 2 to 10 neighboring channels for a recurrent SNN trained on SHD (Figure 4.12c). Most notably, this strategy does not only increase test performance, but it also decreases the computational overhead by reducing the original 700 down to only 70 input channels. Equally acting on the input spike trains, data augmentation allows us to improve generalization. Specifically, we investigate the impact of noise present at the input layer by implementing a spatial spike jitter. To that end, each spike carried by channel i was moved independently to a neighboring unit drawn from the integer normal distribution $\mathcal{N}(i, \sigma)$. This method effectively reduces overfitting and leads to a peak performance of (78.7 ± 2.2) % for a jitter of width $\sigma_u = 20$ (Figure 4.12d). Hence, both of the presented methods constitute simple strategies to significantly improve generalization performance by manipulating the input data.

All of the described strategies can be combined to result in a *best effort* test accuracy of (83.2 ± 1.3) % on the SHD test set (Table 4.2). Most notably, this number was obtained by utilizing the optimal parameters found for the individual experiments described above. A more sophisticated parameter search is likely to improve upon this result. However, the reported number exceeds the perfor-

Table 4.2: Benchmark performance comparison on SHD and SSC. The errors indicate the standard deviation. Table and caption adapted from Cramer et al. (2020b).

Classifier	Architecture	SHD (%)		SSC (%)	
		Train	Test	Train	Test
SVM	Linear kernel	100.0 ± 0.0	56.0 ± 0.4	50.0 ± 0.0	21.6 ± 0.0
	Polynomial kernel of degree 2	91.5 ± 0.1	48.3 ± 0.2	43.2 ± 0.0	23.0 ± 0.0
	Polynomial kernel of degree 3	86.5 ± 0.2	46.7 ± 0.5	51.0 ± 0.0	23.9 ± 0.0
	RBF kernel	96.8 ± 0.1	60.0 ± 0.3	52.3 ± 0.0	29.5 ± 0.0
LSTM[1]	1 hidden layer	99.9 ± 0.1	89.0 ± 0.2	87.2 ± 0.4	72.8 ± 0.5
CNN		99.9 ± 0.0	92.4 ± 0.7	81.0 ± 0.0	77.7 ± 0.2
Spiking[1]	1 hidden layer	98.7 ± 0.2	48.1 ± 1.6	41.1 ± 0.2	32.5 ± 0.5
	2 hidden layers	99.6 ± 0.3	48.6 ± 0.9	45.0 ± 0.5	38.5 ± 0.6
	3 hidden layers	97.1 ± 0.8	47.5 ± 2.3	46.8 ± 0.5	41.0 ± 0.5
	1 recurrent hidden layer	99.7 ± 0.4	71.4 ± 1.9	57.5 ± 0.7	50.9 ± 1.1
	1 recurrent hidden layer (best effort)[2]	99.6 ± 0.4	83.2 ± 1.3	–	–
	1 recurrent hidden layer (SNU)[2]	100.0 ± 0.0	79.0 ± 1.6	–	–

[1] Trained with max-over-time loss [2] 1024 neurons, optimized time constants and channel number combined with noise injection

mances promoted by every strategy on its own. Moreover, it significantly improved upon the test performance of (71.4 ± 1.9) % on the SHD achieved with our previous recurrent SNN implementation.

The choice of SNNs composed of LIF neurons used to establish this first set of benchmarks is somehow arbitrary. To place our results in a somewhat broader context, we finally trained recurrent networks of spiking neural units (SNUs) on our benchmark datasets (Wozniak et al., 2020). These units are related to LIF neurons, but in contrast feature delta synapses. In addition, the networks are usually trained with a different surrogate derivative. For training and evaluation, we applied the aforementioned *best effort* condition. Here, the network composed of SNUs reaches an accuracy of (79.0 ± 1.6) % and hence falls slightly behind the performance of our regular LIF SNNs (Table 4.2).

4.4 Discussion

In this chapter, we have presented two new public-domain spike-based classification datasets to foster benchmarking within the SNN community. Both pose speech classification problems and are hence based on audio recordings. The latter naturally exploit the temporal dimension of SNNs and require fewer input features for accurate classification. Hence, our datasets are computationally more feasible in terms of utilized memory as well as dimensionality. By training a set of spiking as well as non-spiking classifiers, we were able to establish the first set of performance baselines, which serve as a reference for future comparisons. We already gave a glimpse of the tremendous exploration possibilities facilitated by our benchmark data by visiting different loss functions. Moreover, we hinted towards setups for the quantification of generalization to novel speakers and showcased strategies to increase test performance.

The merits of audio data for the benchmarking of SNNs have also been addressed in the past. Here, we decided to base only one of our benchmarks on an existing dataset as only the SC satisfy our minimum requirements. One prominent example for another spoken digit collection is the TIDIGTS dataset (Leonard & Doddington, 1991). However, these digits are published under a commercial license and hence preclude open access. In contrast, the Free Spoken Digit Dataset (Zohar

et al., 2018) is available under a Creative Commons BY 4.0 license. Nevertheless, at the time of writing, it contained only 2000 samples of lower recoding and alignment quality and therefore stays in stark contrast to the HD. Other publicly available datasets encompass Mozilla's Common Voice (Mozilla, 2019), LibriSpeech (Panayotov et al., 2015), TED-LIUM (Rousseau et al., 2012) as well as the Spoken Wikipedia Corpora (Köhn et al., 2016). However, all of these datasets pose more complex classification problems since they only provide alignment information at the sentence level. Only the Spoken Wikipedia Corpora supplies word-level alignment information, but the inventors left the cutting to the end-user. Moreover, the enormous amount of classes as well as their imbalance renders classification more challenging. Because of these reasons and in view of the capabilities of current SNN implementations, we refrain from converting the discussed datasets. Aside from HD, the only public domain dataset with word level-alignment and moderate computational requirements on preprocessing at the time of writing was the SC dataset. Therefore, we decided to derive a second spike-based dataset on these SC mimicking real-world conditions and hence provide a necessary expansion to the high-fidelity recordings of the HD.

The conversion of raw audio signals into neuronal spikes was achieved by relying on established models. By directly releasing spike trains, we standardize the feature selection step for both the HD as well as the SC. As the preprocessing has a major impact on the separability of the benchmarks, we pave the way towards well-defined comparisons of SNN implementations and at the same time relieve the end-user of tedious preprocessing. The models underlying our conversion provide several advantages: First, the physical inner-ear model (Sieroka et al., 2006) and the hair-cell model (Meddis, 1988) are based upon physiological measurements and hence set all hyperparameters. Second, the layer of BCs does not only increase phase-locking, but also emits spike trains that are sparse in space and time. The latter is especially useful for the reduction of the memory footprint and thereby shortens download times of the data files.

A comparable approach to the generation of spike-based benchmark datasets has been taken in the past by Anumula et al. (2018), who played the samples of the TIDIGIT dataset (Leonard & Doddington, 1991) to a dynamic audio sensor with 2×64 frequency selective channels. Unfortunately, the raw audio files of the TIDIGTS dataset are not publicly available and therefore consolidate the contribution of the SHD and the SSC. Furthermore, we also publish the software implementing all model stages required to generate both datasets. Therefore, it is straightforward to extend both datasets by future recordings which would otherwise only be possible for the inventors of the DASDIGIT dataset due to the required dynamic audio sensor.

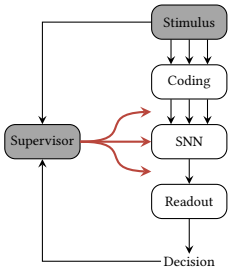
By training a set of spiking and non-spiking classifiers we were able to establish the first set of benchmarks (Table 4.2) and furthermore demonstrated that both datasets can not be accurately solved by existing methods. In this process, we were able to showcase that spike times indeed provide useful information for classifying both datasets. In addition, architectures exhibiting explicit recurrence led to the best performance. Since the audio signals carry information at various time scales, the reverberating recurrent activity most likely implements the required memory to roll out stimulus characteristics at later points in time. Hence, the exploration of additional state variables evolving on slower time scales like the ones presented by Bellec et al. (2018) represents an interesting direction of future research.

It is noteworthy that the surrogate gradient method used within this chapter is only one possibility to train SNNs. Apart from different surrogate derivatives (Bohte, 2011; Bellec et al., 2018; Shrestha & Orchard, 2018; Neftci et al., 2019a; Yin et al., 2020; Wozniak et al., 2020) different gradient-based

approaches have been investigated in the past. Other work considered network translation (Zambrano et al., 2017; Pfeiffer & Pfeil, 2018; Rückauer et al., 2019; Stöckl & Maass, 2021), single spike timing (Mostafa, 2017; Göltz et al., 2021), mean firing rates (Hunsberger & Eliasmith, 2015; Lee et al., 2019), and stochastic approximations (Bengio et al., 2013b; Rezende & Gerstner, 2014; Gardner & Grüning, 2016; Jang et al., 2019) to only name a few. Moreover, there exists a body of literature on online approximations of surrogate gradients (Zenke & Ganguli, 2018b; Kaiser et al., 2020; Bellec et al., 2020). Finally, biologically motivated spike-timing dependent plasticity (STDP)-like learning rules have been explored by Kheradpisheh et al. (2018) and Zhang et al. (2018) (Tavanaei et al. (2018) contributed an extensive review). All of these methods provide interesting starting points for detailed comparisons facilitated by the SHD and the SSC.

In summary, we introduced two open-access datasets which are easy to use and pose challenging problems for current SNN implementations. This has the potential effect to advance the field of neuromorphic computing, which lacks suitable real-world benchmarks. By providing spike-based benchmark datasets, we contribute to the standardized performance evaluation of SNN implementations even across different platforms (Davies, 2019). The latter encompass both conventional computers as well as neuromorphic hardware. Alongside the spiking data, we released our conversion software (Footnote 3) and raw audio datasets (Footnote 1), both under permissive public domain licenses. This allows to keep pace with current and future developments by refining and extending the current version of the SHD and the SSC, but also promotes the creation of novel datasets.

5 Supervised Learning



The surrogate gradient in-the-loop (ITL) framework has been published as a preprint in Cramer et al. (2021) in close collaboration with Sebastian Billaudelle and Dr. Friedemann Zenke. In the following, I will closely follow the publication, but with a more detailed description of the implementation. The work on networks employing time-to-first-spike (TTFS) coding was performed in collaboration with Julian Göltz and Laura Kriener and has been published in Göltz et al. (2021).

Throughout the remainder of this thesis, we focus on distinct optimization strategies for information processing with spiking neural networks (SNNs) emulated on analog neuromorphic hardware. While the emulation is characterized by exceptional efficiency, the training of SNNs on these devices, however, remains a challenge due to the inherent device mismatch, the intrinsic noise as well as the lack of suitable learning methods. Within the scope of this chapter, we start to approach these problems by considering a supervised learning scenario (Section 2.1.4), i. e. we train networks to learn a predefined mapping from an input to an output based on some labeled training data. To turn this into an optimization problem, the mapping is imposed by a loss function quantifying the current success of a given network based on the labeled training data. For the actual optimization, we consider successful methods from the field of machine learning. Specifically, we minimize the loss function with gradient-based methods by resorting to the backpropagation through time (BPTT) algorithm to learn the input to output mapping. Here, we apply the surrogate gradient methods already presented in Chapter 4 to overcome the binary nature of SNNs which precludes the utilization of vanilla gradient descent. While this approach has been shown to be successful for the training of SNNs implemented in software as well as on digital neuromorphic devices, the application of surrogate gradient learning to SNNs emulated on analog neuromorphic hardware has not been achieved so far. Within this chapter, we extend surrogate gradient methods to build a general in-the-loop (ITL) training framework targeting analog neuromorphic devices. The latter is applied to the High Input Count Analog Neural Network X (HICANN-X) single-chip system to showcase how learning self-corrects for device mismatch of the analog components. This property allows us to exploit the benefits of analog neuromorphic hardware and hence facilitates low-latency and high throughput classification of both standard vision datasets as well as the novel collection of spoken words presented in Chapter 4. Our neuromorphic networks exhibit sparse spiking activity and consume less than 200 mW during inference. By this, we establish new benchmarks for low-energy inference with SNNs on analog neuromorphic substrates. We compare our general approach to the specific case of networks employing a time-to-first-spike (TTFS) coding for which the parameter changes resulting from the optimization can be expressed analytically.

5.1 Introduction

Supervised learning methods achieved great successes in the field of machine learning with artificial neural networks (ANNs) (Abiodun et al., 2019). Originally inspired by the architecture of the brain, ANNs are build of artificial neurons connected to form deep networks (LeCun et al., 2015). Despite of this descent, ANNs differ in important aspects from their biological archetype. While ANNs process and communicate multi-valued signals, the processing and communication scheme of biological neurons is characterized by spatio-temporal sparseness deeply anchored in the design of biological networks (Sterling & Laughlin, 2015). The constituents of the latter are spiking neurons which receive and integrate incoming input on their analog membranes and, when reaching a firing threshold, emit action potentials (APs), so-called spikes (Section 2.1.2). These events are communicated asynchronously through the network and serve as input to other downstream neurons (Section 2.1.3). This behavior is captured by SNN models which promise energy-efficiency as well as fault tolerance just like their biological archetype (Gerstner & Kistler, 2002). It is noteworthy that already a single bio-inspired neuron – in contrast to an artificial neuron – constitutes a complex system with finite response times and membrane dynamics, as well as its spike-based communication (Gerstner, 2001; Izhikevich, 2004). The benefits of SNNs can be exploited in combination with neuromorphic hardware systems which mimic the key architectural properties of the processing and communication scheme of biological neural networks (Section 2.2). Here, the analog emulation of neuro-synaptic dynamics, drawing on physical properties and dynamics of the underlying hardware components, promises a power-efficiency as well as scalability (Ambrogio et al., 2018; Marković et al., 2020; Roy et al., 2019).

Analog neuromorphic systems are affected by device mismatch induced by variations in the production process which renders the implementation of functional SNNs challenging (Section 2.2). Hence, when emulating SNNs on these devices, the training needs to be robust and has to take into account potential hardware imperfections. ITL learning schemes have been developed to directly incorporate variations in the training process. Within each iteration of the loop, the network is emulated on the device and observables are recorded. The latter are incorporated into a model implemented in software on the host system to finally calculate weight updates. These updates are then written back to the device and the next iteration starts. By directly incorporating actual observables recorded from the neuromorphic system into the training loop, the procedure automatically corrects for potential deviations between the software model and the hardware implementation (Schmitt et al., 2017; Kungl et al., 2019). This strategy, however, requires suitable training strategies to fully exploit the temporal processing capabilities of SNNs.

Supervised learning in deep SNNs requires to solve both a spatial as well as a temporal credit assignment problem. The standard way of training neural networks in machine learning is to perform gradient descent on a suitable loss function with respect to the network parameters by resorting to the BPTT algorithm (LeCun et al., 2015). A similar approach cannot be directly taken for SNNs, since the spike mechanism renders these networks non-differentiable. Specifically, the crucial points associated with the appearance and the vanishing of spikes under parameter changes pose challenges. Hence, past work came up with the strategy of *network translation* to port the success of machine learning with ANNs to the realm of SNNs (Pfeiffer & Pfeil, 2018). Here, a non-spiking ANN is first trained and then translated into an equivalent SNN (Rückauer et al., 2019; Zambrano et al., 2017; Stöckl & Maass, 2021; Büchel et al., 2021). However, the resulting SNNs often rely on rate-coding schemes and hence do not exploit temporal processing capbabilities provided by SNNs which in

turn results in inefficent solutions. Here, it might be promising to draw again inspiration from the role model of SNNs and aim to mimic the sparseness in the communication scheme of biological neural networks far away from the rate coding limit (Tavanaei et al., 2018; Pfeiffer & Pfeil, 2018; Neftci et al., 2019b).

Temporal coding in combination with a supervised loss function requires to overcome the binary nature of SNNs which precluded the application of vanilla gradient descent. Here, several strategies have been proposed to render the dynamics of SNNs differentiable. First, *smoothing approaches* introduce either graded spikes (Huh & Sejnowski, 2018) or stochasticity (Ackley et al., 1985; Bengio et al., 2013a; Brea et al., 2013; Gardner & Grüning, 2016; Rezende & Gerstner, 2014; Guerguiev et al., 2017) to render the forward pass differentiable. However, introducing stochasticity typically requires averaging over time or space by considering populations of neuron to reduce the variance of the gradient estimates (Hunsberger & Eliasmith, 2015; O'Connor & Welling, 2016; Lee et al., 2016; Neftci et al., 2017; Guerguiev et al., 2017; Payeur et al., 2020). Second, *spike-time gradients* can be adopted in cases in which the neuronal membrane potential (Gütig & Sompolinsky, 2006; Memmesheimer et al., 2014) or firing times (Bohte et al., 2002; Mostafa, 2017; Mozafari et al., 2019; Göltz et al., 2021; Comsa et al., 2020) can be expressed analytically in closed form. This strategy has been demonstrated in multi-layer networks in conjunction with time-to-first-spike coding schemes. While this method leaves the forward model unchanged and allows to exploit temporal coding, it requires additional mechanisms to deal with neurons that remained silent or emitted multiple spikes. Finally, *surrogate gradient approaches* also leave the forward model unchanged while not relying on a specific coding strategy. Instead, such approaches directly apply approximations to the gradients to overcome the binary nature of spikes which impedes the application of vanilla gradient descent, a notion inspired by work on binary ANNs (Bengio et al., 2013a; Courbariaux et al., 2016; Neftci et al., 2019b). Most notably, they allow to successfully train SNNs with spike-time coding (Bohte, 2011; Esser et al., 2016; Zenke & Ganguli, 2018a; Shrestha & Orchard, 2018; Bellec et al., 2018; Wozniak et al., 2018; Neftci et al., 2019b).

In this chapter, we extend previous work on surrogate gradients to develop an ITL training frameworks. The latter is applied to the mixed-signal BrainScaleS-2 single-chip system to solve several challenging benchmark problems requiring precisely timed spikes instead of firing rates. Most notably, we demonstrate recurrent SNNs trained on the real-world speech dataset presented in Chapter 4. In addition, we showcase ultra-low latency classification facilitated by the spiking implementation and further boosted by the accelerated nature of HICANN-X (Section 3.2). Moreover, we highlight the self-calibrating nature of our ITL training scheme which automatically equalizes device variations and hence renders hardware calibration redundant. All of our networks perform at comparable accuracy levels as corresponding software simulations, thereby validating our hardware implementation.

5.2 Methods

Within this section, we present a general ITL learning framework which facilitates the training of multi-layer as well as recurrent SNNs emulated on analog neuromorphic hardware. Specifically, we consider loss functions on which gradient descent via BPTT is performed. To that end, we draw on a surrogate derivative to overcome vanishing gradients (Zenke & Ganguli, 2018a; Neftci et al., 2019b). Additionally, this choice provides a high degree of flexibility as surrogate gradient

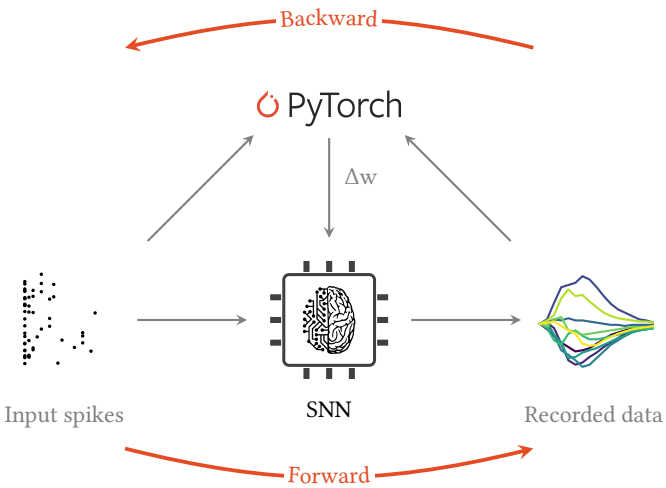


Figure 5.1: Surrogate gradient-based ITL learning for analog neuromorphic hardware. The SNN emulated on the analog device is optimized for information processing in an iterative fashion based on an approximate model on the host computer. To that end, the forward pass of the model is emulated on the neuromorphic device by stimulating the SNN with a set of input spikes. Simultaneously, observables originating from the device are recorded and injected together with the input spike trains in an approximate model of the neuro-synaptic dynamics simulated on the host computer. This approximate model allows to perform the backwards pass with gradient-based methods and the BPTT algorithm with a standard auto-differentiation library. Subsequently, the computed weight updates Δw are written back to the analog neuromorphic device and the next iteration is entered. Figure taken from Cramer et al. (2021).

learning supports arbitrary loss functions which can be tailored to either rate- or spike-timing-based coding schemes or even combinations thereof. We start with a general description of our surrogate gradient-based ITL training framework in Section 5.2.1 and then move on to the specific implementation on the HICANN-X application-specific integrated circuit (ASIC) in Section 5.2.2. We close by a description of the utilized application specific techniques like the loss functions and regularization methods (Section 5.2.3), the considered benchmark datasets (Section 5.2.4) as well as the initialization strategies (Section 5.2.5). All utilized parameters are summarized in Table 5.1.

We compare our framework to a spike-time gradient approach for which the weight updates can be calculated analytically. More specifically, we consider a TTFS coding scheme within feed-forward architectures. The methods thereof are depicted in Appendix A.2. Due to the analytical solution and the requirement of emitted spikes only, the weight updates can be directly calculated according to Equations (A.16), (A.22) and (A.23) on the host machine without relying on auto-differentiation tools.

5.2.1 In-the-loop training framework

Basically, our ITL framework comprises of three successive stages (Figure 5.1). First, the forward pass is emulated on the analog neuromorphic substrate by stimulation with samples originating from a spike-based dataset. Simultaneously, the spikes as well as the membrane traces of all neurons are recorded and transferred to the host computer. Second, this data is injected into an approximate software model of the neuromorphic SNN to render the acquired data differentiable. In more detail, we draw on BPTT using auto-differentiation libraries in combination with state-of-the-art optimizers which promotes the calculation of weight updates (Paszke et al., 2017) A single loop iteration is

then finalized by, third, transferring the calculated weight updates from the host computer back to the analog device. By incorporating the actual hardware recordings into the computation of surrogate gradients – instead of relying on simulated estimates – the software model is ensured to not deviate too far from the actual dynamics emulated on the neuromorphic substrate. As a result, the calculated updates remain accurate despite the approximate nature of the software model. Thereby, the algorithm self-adjusts for parameter mismatch of the analog components.

In the following, we describe the three central steps of our general ITL learning scheme in more detail. In that process, we start with a description of the emulated networks and discuss the observables required for weight update calculations. Next, the approximate model of neuro-synaptic dynamics as well as the injection of recordings is depicted. We close by a specification of the parameter updates on the neuromorphic system.

Network emulation: Our ITL training framework allows to optimize arbitrary networks composed of leaky integrate-and-fire (LIF) neurons. In general, this framework can be applied to any analog or mixed-signal substrate as long as access to all membrane potentials and spike times of the emulated neurons is supported. Here, we focus on multi-layer as well as recurrent SNNs with signed synaptic weights. In more detail, the latter can seamlessly transit between positive (excitatory) and negative (inhibitory) weight values. For simplicity, we consider the case of coinciding leak and reset potentials, u_{leak} and u_{reset}. Moreover, the refractory period τ_{ref} is set to zero for simplicity. The actual emulation of the forward pass encompasses the stimulation of the implemented SNNs with samples originating from a spike-based dataset. The spikes as well as the digitized membrane potentials of all neurons within the network are simultaneously recorded and in turn, transferred to the host computer to perform the subsequent update calculations.

Construction of computation graph: In the following, we highlight a method for the construction of a computation graph based on an approximate model of the emulated neuro-synaptic dynamics of the analog neuromorphic substrate, which allows us to perform gradient descent on an arbitrary loss function via BPTT to calculate weight updates Δw. In more detail, the neuronal dynamics can be iteratively *simulated* on the host computer resulting in a differentiable computation graph. Subsequently, the previously described measured observables originating from the analog neuromorphic substrate are incorporated into this computation graph. In that process, we rely on *measured* quantities whenever available and resort to simulation results for intermediate observables that can not be measured like the synaptic currents.

Within our computation graph (Figure 5.2), we assume ideal LIF dynamics in combination with current-based synapses (Sections 2.1.2 and 2.1.3) and apply exact integration based on a regular time step δt (Equation (2.12)). To that end, we iteratively estimate the membrane potentials by incorporating the decay to the leak potential u_{leak} as well as the calculated synaptic currents $\tilde{I}[t]$. The latter are based on the presynaptic spike trains of the downstream layer $\tilde{S}_j^{(l-1)}[t]$ and the recurrent ones $S_k^{(l)}$:

$$\tilde{u}_i^{(l)}[t+1] = \tilde{u}_i^{(l)}[t] \cdot \mathrm{e}^{-\delta t/\tau_{\text{mem}}} + \tilde{I}_i^{(l)}[t]\,, \tag{5.1}$$

$$\tilde{I}_i^{(l)}[t+1] = \tilde{I}_i^{(l)}[t] \cdot \mathrm{e}^{-\delta t/\tau_{\text{syn}}} + \underbrace{\textstyle\sum_j w_{ij}^{(l)} \tilde{S}_j^{(l-1)}[t]}_{\text{feed-forward}} + \underbrace{\textstyle\sum_k v_{ik}^{(l)} \tilde{S}_k^{(l)}[t]}_{\text{recurrent}}\,, \tag{5.2}$$

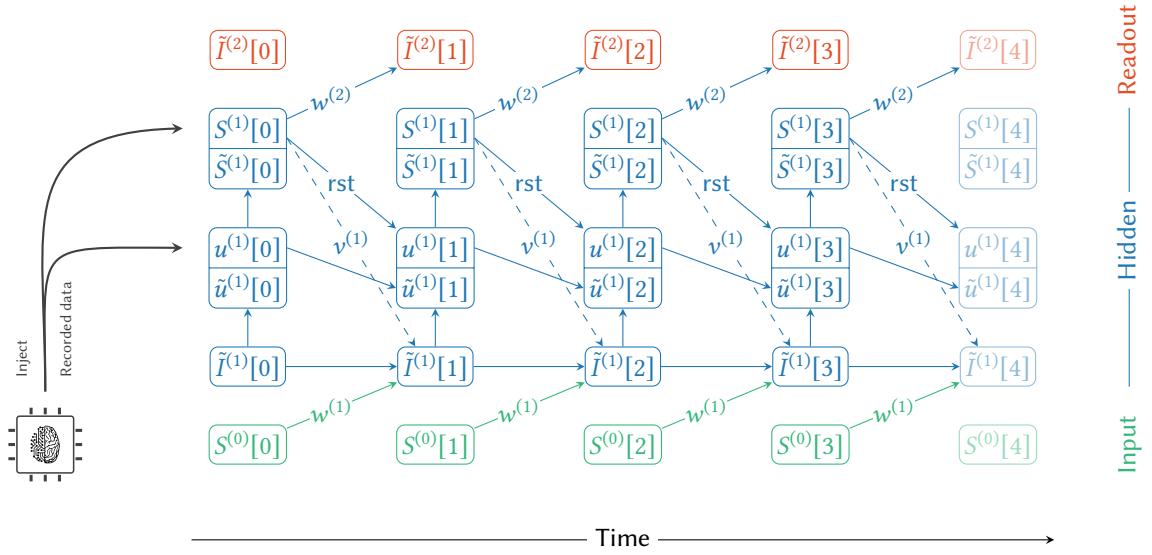


Figure 5.2: Construction of a differentiable computation graph. A computation graph is constructed by integrating the LIF equation in combination with current-based synapses. Here, all modeled state variables are highlighted by a tilde. Actual recordings stemming from the analog neuromorphic substrate are inserted into their place whenever available. Time evolves from left to right, while the information flow within a single integration step occurs from bottom to top. The synaptic currents are supplied by spikes originating from the previous layer ($S^{(l-1)}$) as well as recurrent events ($S^{(l)}$), weighted by the feed-forward ($w^{(l)}$) and recurrent ($v^{(l)}$) weights, respectively. These currents are in turn integrated on the neuronal membrane potentials ($u^{(l)}$). Upon reaching the firing threshold, neurons emit spikes ($S^{(l)}$) which in turn propagate to neurons in downstream layers as well as recurrently within the same layer. In addition, the release of a spike causes the reset (rst) of the membrane potential within the next time step. The modeled membrane potentials $u^{(l)}$ and spikes $S^{(l)}$ are continuously synchronized with data recorded from the analog neuromorphic device ($\tilde{u}^{(l)}$ and $\tilde{S}^{(l)}$). Figure taken from Cramer et al. (2021).

where $w_{ij}^{(l)}$ denote the feed-forward and $v_{ik}^{(l)}$ the recurrent weights of layer l. Here, we highlight all modelled state variables which represent estimates of the on-chip dynamics by a tilde. It is noteworthy that the simplified dynamics captured by Equations (5.1) and (5.2) lack of higher-order effects induced by the analog implementation. Hence, $\tilde{u}_i^{(l)}[t]$ and $\tilde{S}_i^{(l)}[t]$ are likely to deviate from the actual values $u_i^{(l)}[t]$ and $S_i^{(l)}[t]$ as sampled from the hardware. To not distort the resulting gradients, the normalized, *actual* recordings are inserted in their place whenever possible and we only utilize the model for the calculation of their derivatives. In more detail, we introduce an auxiliary identity function $f(x, \tilde{x}) := x$ and define its surrogate derivatives:

$$\frac{\partial f}{\partial x} := 0\,, \qquad \frac{\partial f}{\partial \tilde{x}} := 1\,. \tag{5.3}$$

With this, we are able to modify Equation (5.1) to:

$$\tilde{u}_i^{(l)}[t+1] = f\Big(u_i^{(l)}[t+1], \tilde{u}_i^{(l)}[t] \cdot \mathrm{e}^{-\delta t/\tau_{\mathrm{mem}}} + \tilde{I}_i^{(l)}[t]\Big)\,. \tag{5.4}$$

Likewise, we define $\tilde{S}_j^{(l)}[t](S_j^{(l)}[t], \tilde{u}_j^{(l)}[t]) := S_j^{(l)}[t]$ for spike times $S_j^{(l)}[t]$ with the associated surrogate derivatives:

$$\frac{\partial \tilde{S}_j^{(l)}[t]}{\partial S_j^{(l)}[t]} := 0\,, \qquad \frac{\partial \tilde{S}_j^{(l)}[t]}{\partial \tilde{u}_j^{(l)}[t]} := \Big(\beta \cdot |\tilde{u}_j^{(l)}[t] - \vartheta|\Big)^{-2}\,, \tag{5.5}$$

where β corresponds to the steepness of the surrogate derivative (Zenke & Ganguli, 2018a). Note that this is the same surrogate derivate as the one used in the context of Chapter 4.

Specifically, for the evaluation of the backward pass and the associated calculation of $\partial \mathscr{L} / \partial \theta = \ldots \partial \tilde{S} / \partial \tilde{u} \cdot \partial \tilde{u} / \partial \theta$, we resort to the sampled values of the membrane potential originating from the analog device whenever an expression containing $\tilde{u}$ is evaluated, e. g. $\partial \tilde{S} / \partial \tilde{u}$. In contrast, we draw on the modelled quantities for further derivatives $\partial \tilde{u} / \partial \theta$ within the recursion relation of BPTT.

Closing the loop: The loop is closed by transferring the calculated weight updates from the host computer to the analog neuromorphic device. To this end, the updated weights only need to be aligned with the weight resolution of the substrate. After this third step, the next iteration is entered.

5.2.2 Implementation on HICANN-X

We apply our flexible ITL training framework to the HICANN-X chip (cf. Section 3.2). In the following, we start with a description of the required mapping of multi-layer as well as recurrent SNNs with signed synaptic weights to the analog neuromorphic core of HICANN-X. Subsequently, we depict the pursued strategy for the parallel recording of membrane potentials via the CADCs simultaneously to the gathering of event data required to construct the differentiable computation graph. We close with a description of the applied weight scaling used to align the calculated updates with the weight resolution on HICANN-X as well as the software implementation of the whole framework.

Mapping of multi-layer as well as recurrent SNNs to the analog neuromorphic core: We used the HICANN-X chip to implement multi-layer as well as recurrent SNNs with signed synaptic weights (Figure 5.3). As outlined in Chapter 3, the analog core of HICANN-X is designed to emulate SNNs adhering to Dale's law. To still allow for a continuous transition between positive and negative weights during training, we configured one line of each synapse driver to be excitatory and the other line to be inhibitory. Now, a union of two synapses – one on each line of each synapse driver – can be considered as a logical synapse (Figure 5.3a). By sending each spike event to both lines and ensuring only one of the two weights constituting each logical synapse to be non-zero, the combination of both circuits can be regarded as a single synapse carrying a sign. This implementation of signed synapses reduces the number of synapses per neuron from 256 to 128. To restore the original fan-in of 256 synapses per neuron, we utilized the multi-compartment features of HICANN-X and connected always two neighboring neuron circuits to form a single logical neuron. To that end, we disabled the leak as well as the spiking mechanism of all but one of the combined neuron compartments.

The implementation of larger feed-forward as well as recurrent SNNs required to exchange spikes between all four quadrants of HICANN-X. For that purpose, we utilized the routing capabilities to unify the synapses of all four quadrants into a *virtual* synapse array. Due to the relay of two neuron compartments and the use of singed synapses, the latter is of size 256×256 for all results presented in this chapter. Within this formulation, feed-forward as well as recurrent neural networks can be mapped efficiently and easily to HICANN-X (Figure 5.3).

Here, we consider network topologies that comprise a single hidden layer accompanied by a non-spiking readout layer. The associated non-spiking leaky integrators can be implemented on

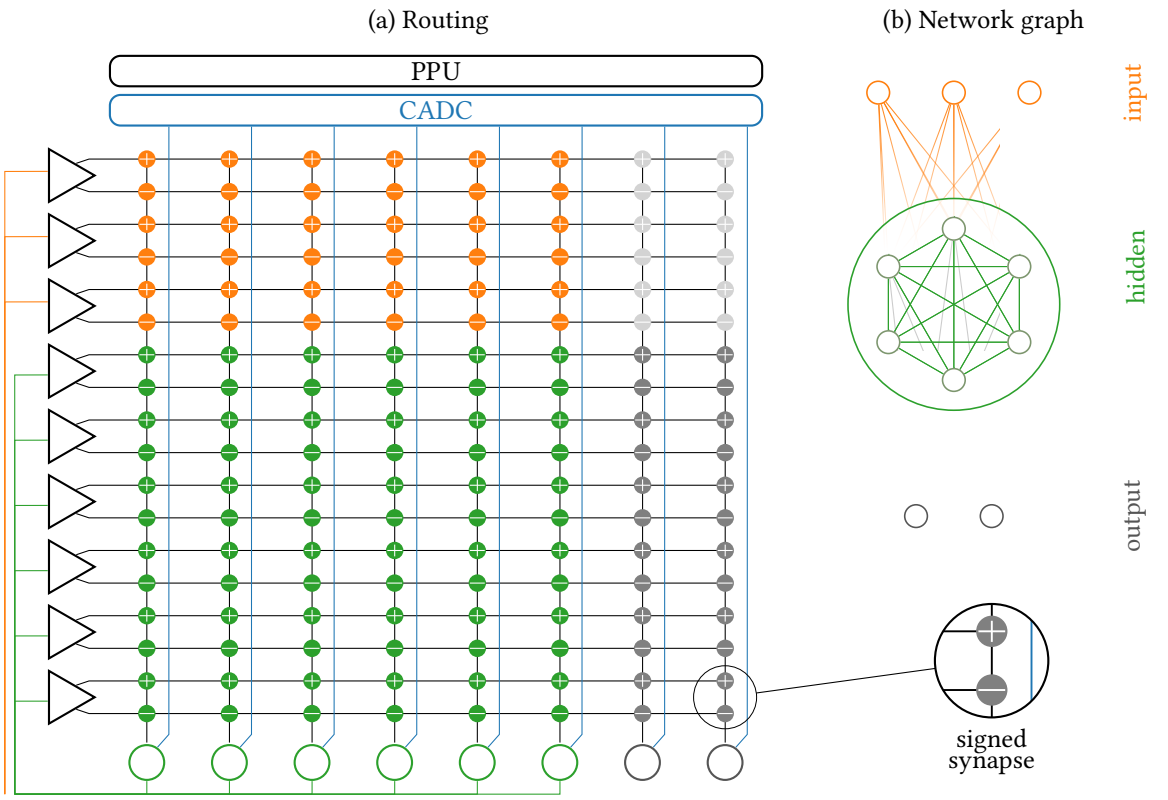


Figure 5.3: Implementation of multi-layer and recurrent networks on the HICANN-X ASIC. (a) The flexible on-chip routing capabilities of HICANN-X promote the implementation of near arbitrary network topologies on the analog neuromorphic core. The stimulating spikes (orange lines) are injected via a column of synapse drivers (black triangles) into the synapse array. A set of synapses (orange dots) relays these events to the hidden layer neurons (green circles). All hidden layer spikes are routed on-chip, either recurrently to other hidden layer neurons (green dots) or to the downstream layer (gray dots) composed of the readout units (gray circles). To implement synapses with signed weights, each logical connection is represented by a pair of excitatory and inhibitory hardware synapses. During the analog emulation, the membrane potentials are digitized in parallel via the CADCs and read out by the PPUs. **(b)** Corresponding recurrent network graph. Synaptic connections are highlighted with the same color as in panel (a). Figure taken from Cramer et al. (2021).

HICANN-X by disabling the readout neurons' firing mechanism. Thereby linear readout units – common in the field of machine learning – can be efficiently implemented on-chip.

All neuronal parameters, including reference potentials and time constants, were calibrated to the values presented in Table 5.1. The dynamical range of the membrane potential was maximized by a large distance between the leak and threshold potential $u_{\text{thres}} - u_{\text{leak}}$ to ensure a high signal-to-noise ratio. Since the model used to construct the computation graph does not include refractory effects, the refractory time τ_{refrac} was configured to a small value. Furthermore, the reset potential u_{reset} was configured to match the leak potential u_{leak} for simplicity.

The neuron-wise adjustability of parameters does not only facilitate calibration routines to mitigate circuit-to-circuit deviations induced by variations in the production process, but also allows to deliberately *decalibrate* the system. Specifically, we calibrated τ_{mem}, τ_{syn} and u_{thres} to neuron specific target values. The latter were drawn from a normal distribution $\mathcal{N}(\mu_{\text{d}}, \sigma_{\text{d}})$ with mean μ_{d} corresponding to the original calibration target. Here, the quality of the calibration is measured by

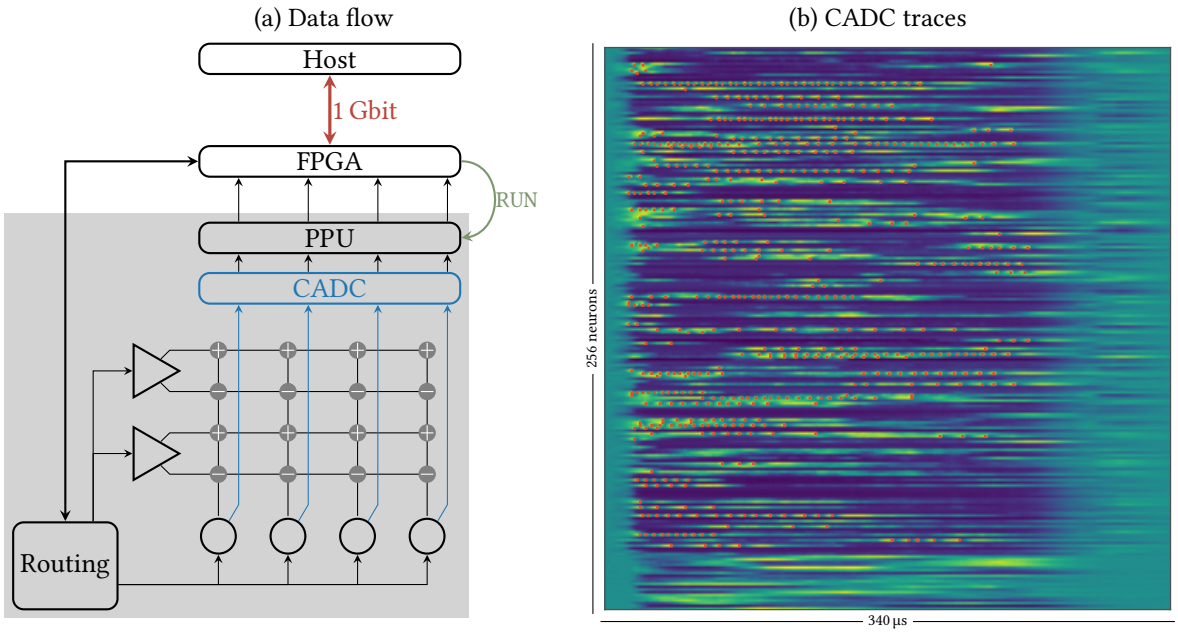


Figure 5.4: Parallel digitization of membrane potentials on HICANN-X. (a) Illustration of the applied membrane potential recording strategy. During the emulation of neuro-synaptic dynamics, the CADCs are used to digitize the analog membrane potentials of all analog neurons in parallel. To that end, the FPGA is used to trigger a sequence of CADC recordings via the PPUs. Specifically, the PPUs continuously poll for RUN signals send from the FPGA. In case this signal is detected, the SIMD vector units are used to record a sequence of samples of each neuron's membrane potential in parallel. The resulting data is immediately transferred to an intermediate memory region on the FPGA from where it is asynchronously read back by the host computer. By this approach, a data rate of 1.2 Gbit can be reached which exceeds the bandwidth achievable via a direct transfer of CADC data to the host machine. **(b)** With this strategy, the membrane potentials of 256 neurons can be digitized in parallel with a sample interval of only 1.7 µs. The spike times are annotated in red. Panel (b) taken from Cramer et al. (2021).

the normalized standard deviation σ_d which was swept in the range 0 % to 50 %. Thereby, we provide a strategy to systematically explore the self-calibration properties of our learning framework.

Recording spikes and analog membrane traces: Training SNNs with surrogate gradient methods requires knowledge of the neuronal membrane potentials. In contrast to digital systems, the latter are represented as physical voltages on analog neuromorphic systems and are therefore not directly available for weight update calculations. To be able to perform numerical calculations involving the membrane potentials on these systems, digitization is required. However, the latter is often challenging due to the inherent parallelism and time-continuous nature of the underlying substrate. This bottleneck becomes even more strained for accelerated systems like HICANN-X.

The two column-parallel analog-to-digital converters (CADCs) (Section 3.1.1), deeply anchored into the design of BrainScaleS-2, allow to digitize the membrane potentials of 2×128 neurons simultaneously and hence reduce the strain of the aforementioned bottleneck (Figure 5.4). Since we connected pairs of neurons to increase the synaptic fan-in, a single read per CADC is sufficient to acquire a sample of each neuron's membrane in parallel. This is furthermore facilitated by the interleaved row-wise parallel reading scheme of the CADCs: With a single read, the membrane potential of every second neuron is digitized. Hence, our choice of two-compartment neurons promotes a

```
while(command != HALT){
  // check if last sample in batch
  if(command == RESET_BATCH)
    batch_offset = 0;
  end if

  // wait for run signal
  if(command == RUN)
    command = NONE;
    start = get_time_base();

    // calculate memory offset for current sample
    external_offset = get_memory_offset(batch_offset, ppu_id, sample);

    // read membrane potentials and write to external memory
    for(size_t sample=0; sample < n_samples; ++sample)
      asm volatile(
        inx  potentials, %[ca_base], %[cadc_offset]
        outx  potentials, %[external_base], %[external_offset]
        : [potentials] "=&qv" (vec_splat_u8(0))
        : [ca_base] "b" (dls_causal_base),
          [external_base] "b" (dls_extmem_base),
          [cadc_offset] "r" (0),
          [external_offset] "r" (external_offset)
        :
      );
    end for
    asm volatile("sync");

    batch_offset++;

    // measure duration to estimate sample rate
    duration = get_time_base() - start;
  end if
end while
```

Listing 3: Recording of analog membrane traces with the PPUs. The PPUs continuously polls for the update signal `RUN` send from the field-programmable gate array (FPGA). In case this signal is detected, the PPUs trigger `n_samples` CADC reads via the SIMD vector units to digitize the membrane potentials of all neurons in parallel. Here, the recorded data is directly transferred to an intermediate memory region on the FPGA at offset `external_offset` also accessible by the host system. Batch support is provided by reusing the same memory region once all samples have been acquired. This cycling is implemented by the additional reset signal `RESET_BATCH` which leads to a reset of the batch index `bacht_offset`. For further processing, the time required to acquire `n_samples` is assessed. The kernel code is shown in NASM syntax.

minimal sample period while simultaneously ensuring stable timing. The latter is in particular not disrupted by multiple successive reads for the recording of a single sample for each neuron and each time step.

Instead of triggering a sequence of CADC reads from the host system, we utilized the embedded plasticity processing units (PPUs) to trigger the conversions to ensure higher and more stable sampling rates (Figure 5.4a). The implementation is centered around the kernel code shown in Listing 3. Specifically, the PPUs are programmed to continuously poll for the update signal `RUN` send in regular

time intervals from the FPGA (line 1). In case this signal is detected (line 8), the single instruction, multiple data (SIMD) vector units trigger a sequence of `n_samples` CADC reads and stores the associated digitized data in the register `potentials` (lines 16 to 27). By relying on an external update signal, both PPUs can be synchronized which in turn facilitates the assignment of CADC samples and corresponding time stamps. The digitized data in `potentials` is directly transferred to a dedicated FPGA memory region at offset `external_offset` (line 19). Notably, the latter cannot only be accessed from the SIMD vector units, but also from the interfacing host system. The traces can then asynchronously be read back by the host computer after the completion of a batch of input samples. Subsequently, the batch index `batch_offset` is reset from the FPGA via the `RESET_BACHT` signal (lines 3 to 5), leading to a cycling over the very same memory region after batch completion (line 13). For further processing of the digitized membrane data and to finally estimate the achieved sample rate, the duration `duration` required to digitize and store `n_samples` samples is assessed (lines 10 and 33).

Including the transfer of data to the external memory region, we reach a sample rate of approximately 0.6 MSample s^{-1}, which corresponds to a sampling interval of 1.7 µs. For 256 neurons, this yields a data rate of 1.2 Gbit s^{-1} which exceeds the 1 Gbit s^{-1} bandwidth between the host computer and the interfacing FPGA and hence confirms our PPU-based approach. For comparison, if the 1 Gbit bandwidth to the host system would be exclusively utilized for the transfer of digitized membrane traces, a theoretical sample rate of approximately 0.5 MSample s^{-1} could be reached which corresponds to a sample interval of 2.0 µs. In addition to the sampled membrane traces, we simultaneously record the spike events emitted by the neurons on the neuromorphic device. In this setup, the maximum recording duration is only limited by the available memory and the batch size.

For most of the presented results, we set the step size δt used to construct the computation graph to the aforementioned sample interval of the CADCs. It is noteworthy that albeit the timestamps of the spikes returned from HICANN-X feature a much higher precision, we aligned them to the same regular grid. For some tasks, a higher temporal resolution can be beneficial to better capture causal relations between successive spikes. In these cases, we applied a simple interpolation of the recorded membrane traces with base $\delta t/n$ by drawing on a piecewise constant approximation.

Aside from plain data recording, the PPUs furthermore enable the implementation of a fast inference mode. An overall lowered emulation time is a key feature of this mode which can be achieved by reducing the number of acquired CADC samples `n_samples` as well as the shortening of the temporal separation between concurrent stimuli. However, the input spikes of two concurrent stimuli can not be sent in direct succession due to the finite time constants of the analog system: Before stimulation with the next input sample, all synaptic currents as well as membrane potentials have to decay back to their respective baselines. To not have to wait for their natural decay with time constants τ_{syn} and τ_{mem}, $u_i^{(l)}$ and $I_i^{(l)}$ were artificially reset via the PPUs after the acquisition of the requested sample count. Both resetting mechanisms can be triggered in parallel from the SIMD vector unit with the kernel code shown in Listing 4. Specifically, the membrane potentials are clamped to the reset potential u_{reset} for the duration of the refractory period τ_{ref} by triggering silent neuron resets within the neuron backend (line 6). This approach effectively irradiates past states, since u_{reset} was chosen to match the leak potential u_{leak} and due to the short τ_{ref}. At the same time, the synaptic inputs are connected via debug lines to an external digital-to-analog converter (DAC) (lines 3 and 4). The latter pulls the synaptic inputs back to their baseline values and is again disconnected after the expiration of `cycles` cycles (lines 18, 22 and 23). Since the recording of the membrane potentials is done on the PPUs, this fast inference can be exploited even further by directly evaluating the

```
asm volatile(
  // Connect synapse debug lines
  outx  attach, %[debug_switch_base], %[index_0]
  outx  attach, %[debug_switch_base], %[index_1]
  // Trigger silent neuron reset of one neuron compartment
  outx  zeros, %[neuron_reset_base], %[index_odd]
  :
  : [attach] "qv" (vec_splat_u8(syn_debug_both)),
    [zeros] "qv" (vec_splat_u8(0)),
     [debug_switch_base] "b" (dls_config_odd_base),
     [neuron_reset_base] "b" (dls_neuron_reset_base),
     [index_0] "r" (512 + 0),
     [index_1] "r" (512 + 1),
     [index_odd] "r" (1)
   :
);

sleep_cycles(cylces);

asm volatile(
   // Disconnect synapse debug lines
   outx  detach, %[debug_switch_base], %[index_0]
   outx  detach, %[debug_switch_base], %[index_1]
   :
   : [dettach] "qv" (vec_splat_u8(syn_debug_cut)),
     [zeros] "qv" (vec_splat_u8(0)),
     [debug_switch_base] "b" (dls_config_odd_base),
     [index_0] "r" (512 + 0),
     [index_1] "r" (512 + 1)
   :
);
```

Listing 4: Fast inference mode on the PPUs. The temporal separation between concurrent samples can be lowered by artificially resetting the membrane potentials and the synaptic currents in parallel via the SIMD vector unit to not wait for their decay. For subsequent samples to not interfere, we reset the membrane potentials of all neurons after the presentation of the input by triggering a silent reset of the neuron backend. At the same time, the synaptic inputs are connected via debug lines to an external DAC, thereby pulling the synaptic input back to their baselines effectively irradiating the past state. After settling to the resting state, the lines are again disconnected. The kernel code is shown in NASM syntax.

network's decision on-chip via the SIMD vector unit. With this, only crucial classification results need to be transferred back to the host machine. Thereby, the I/O bottleneck can be relieved and at the same time, much larger batch sizes can be supported.

Weight scaling: The synaptic weights calculated in software have to be scaled and clipped to align with the weight resolution of 7 bit signed integers of HICANN-X. To that end, we first matched the amplitudes of single post-synaptic potentials (PSPs) in the software simulation and the hardware emulation by adjusting the analog bias currents on HICANN-X. While this scaling is crucial for all weights connecting spiking neurons, the weights of synapses targeting non-spiking layers can be scaled arbitrarily due to the absence of a threshold criterion.

For the classification of MNIST images, we adaptively scaled the weights between hidden and

```
network = strobe.nn.Network([strobe.nn.Linear(n_input, n_hidden),
                             strobe.nn.LIFLayer(n_hidden),
                             strobe.nn.Linear(n_hidden, n_output),
                             strobe.nn.LILayer(n_output)])
network.connect(...)

for epoch in range(n_epochs):
    for x, y_star in data_loader:
        optimizer.zero_grad()

        y = model(x)

        loss(y, y_star).backward()
        optimizer.step()

        with torch.no_grad():
            model.linear_hidden.weight.data.clamp_(-limit, limit)
            model.linear_output.weight.data.clamp_(-limit, limit)
```

Figure 5.5: Extension of the PyTorch library for ITL learning. Our framework extends the existing PyTorch auto-differentiation library and builds upon a custom backend that relies on the BrainScaleS-2 software stack. By this, the experiment control flow is compatible with the one used for standard machine learning applications on conventional hardware. Shown is the definition of a SNN with a single hidden layer and a readout comprising of non-spiking leaky integrators. The training loop is comparable to standard PyTorch implementations, despite the required clamping of synaptic weights prior to their upload to HICANN-X.

output neurons after each batch. More specifically, we aligned the largest absolute weight value as represented in software $\max_{i,j} |w^{\mathrm{L}}_{ij}|$ to the maximum weight possible on the neuromorphic substrate prior to writing the weights to the device.

Software environment: We implemented our ITL training framework within PyTorch's auto-differentiation library (Paszke et al., 2017). This extension builds upon the BrainScaleS-2 software stack which was used to configure HICANN-X as well as to execute experiments (Müller et al., 2020). An example of the high-level usage is shown in Figure 5.5. By deeply embedding our framework into the PyTorch library, the experiment control flow is compatible with the one of standard machine learning applications in common software implementations. The implementation of the surrogate partial derivatives given in Equations (5.3) and (5.5) was achieved in a comparable manner to Listing 2.

5.2.3 Loss functions

Our proposed framework can be combined with any differentiable loss function that can be formulated based on the data provided by the neuromorphic system. We start with a description of the specific loss functions utilized to conduct the results shown within this chapter. The flexibility provided by our framework furthermore allows us to augment these task-specific loss functions with regularizations terms. The considered choices are described afterward. We close with a summary of

the employed benchmark datasets as well as the weight initialization strategy. In close accordance with the methods presented in Chapter 4, we presented the data in batches to the SNNs which is why we again introduce a batch index s in the following.

Task loss: Within this chapter, we consider two loss functions based on the membrane potential of non-spiking leaky integrators in the readout layer ($l = L$). Hence, the network's response is directly encoded in the evolution of their membrane potentials $u_{i,s}^{L}[t]$. We trained all feed-forward architectures with the Adam optimizer (Kingma & Ba, 2014) in combination with a *max-over-time* loss which has also been visited within the scope of Chapter 4. More specifically, we applied the cross-entropy loss in Equation (4.9) to the time step containing the maximal membrane potential deflection for each readout unit $\tilde{t}_{i,s} = \arg\max_n u_{i,s}^{(L)}[t]$ (Cramer et al., 2020b). Due to the finite time constants of both LIF as well as leaky integrator neurons, we applied a loss function that considers all samples of each readout unit for our recurrent SNNs. Specifically, we replaced the max-operator with a sum, i. e. $\tilde{t}_{i,s} = \sum_{i=1}^{N_\mathrm{T}} u_{i,s}^{(L)}[t]$ to obtain a *sum-over-time* loss, incorporating all membrane samples of the readout units. This choice has been utilized in combination with the SMORS3 optimizer.

Regularization: We augmented the task-specific loss by regularization terms. Specifically, we applied activity regularization leading to sparse firing patterns, both in space and time. For the classification of MNIST digits, we applied a regularization term rewarding solutions with few spikes:

$$\mathscr{L}_\mathrm{b}\left(S_{i,s}^\mathrm{H}\right) = \rho_\mathrm{b} \frac{1}{N_\mathrm{B}} \sum_{s=1}^{N_\mathrm{B}} \frac{1}{N_\mathrm{H}} \sum_{i=1}^{N_\mathrm{H}} \left(\sum_t S_{i,s}^\mathrm{H}[t] \right)^2 . \tag{5.6}$$

Here, we introduced the strength parameter ρ_b, the hidden layer size N_H, the corresponding hidden layer spike trains $S_{i,s}^\mathrm{H}$ and the number of samples in a batch N_B. For readability and due to the use of a single hidden layer only, we dropped the layer index l in the following and denote quantities of the hidden layer by the superscript H. For the spoken digit classification, we applied a regularization which only punishes solutions in case the hidden layer spiking activity exceeds a threshold:

$$\mathscr{L}_\mathrm{a}\left(S_{i,s}^\mathrm{H}\right) = \rho_\mathrm{a} \frac{1}{N_\mathrm{B}} \sum_{s=1}^{N_\mathrm{B}} \max\left[0, \left(\left[\sum_{i=1}^{N_\mathrm{H}} \sum_{t=1}^{N_\mathrm{T}} S_{i,s}^\mathrm{H}[t] \right] - \theta_\mathrm{a} \right)^2 \right] , \tag{5.7}$$

with the strength ρ_a, the threshold θ_a and the number of time steps N_T. It is noteworthy that shaping the activity of SNNs is of particular interest in the field of neuromorphic engineering, where the overall power consumption often correlates with the spike count. Moreover, a reduced number of spikes not only relieves the bandwidth on the substrate itself, but also between chips in multi-chip systems.

Regularization terms can not only be tailored to improve generalization performance, but can also be used to incorporate hardware-specific constraints such as finite weights or limited dynamic ranges of the analog recordings. To prevent saturation of the weights on HICANN-X, we added a regularizer punishing large weight values:

$$\mathscr{L}_\mathrm{w}\left(w_{ij}^\mathrm{H}\right) = \rho_\mathrm{w} \frac{1}{N_\mathrm{I}} \sum_{i=1}^{N_\mathrm{I}} \frac{1}{N_\mathrm{H}} \sum_{j=1}^{N_\mathrm{H}} \left(w_{ij}^\mathrm{H}\right)^2 , \tag{5.8}$$

with strength parameter ρ_w and the input layer size N_I. For all feed-forward architectures, we additionally included readout regularization to better utilize the dynamic range and to prevent saturation

of the membrane potentials of the analog readout units. Here, we considered a regularizer acting on the maximum deflection of each readout neuron's membrane potential:

$$\mathscr{L}_\mathrm{r}\left(u^\mathrm{L}_{i,s}\right) = \rho_\mathrm{r}\frac{1}{N_\mathrm{B}}\sum_{s=1}^{N_\mathrm{B}}\frac{1}{N_\mathrm{L}}\sum_{i=1}^{N_\mathrm{L}}\left(\max_t u^L_{i,s}[t]\right)^2 , \tag{5.9}$$

with strength parameter ρ_r and the number of units in the label layer N_L.

5.2.4 Datasets

We utilized three different classification datasets to benchmark our implementation of SNNs emulated on HICANN-X. Specifically, we trained SNNs on the Yin-Yang, the MNIST as well as the Spiking Heidelberg Digits (SHD) dataset with our ITL learning framework.

Yin-Yang: As proposed by Kriener et al. (2021), we took the 5000 train and 1000 test samples from the Yin-Yang figure. Each point was interpreted by a pair of Cartesian coordinates $(x_1, x_2) \in [0, 1]^1$. The latter were transformed to the spiking domain by multiplication with the input time constant τ_in, i. e. $t^\mathrm{I}_1 = x_1\tau_\mathrm{in}$ and $t^\mathrm{I}_2 = x_2\tau_\mathrm{in}$, respectively. To capture the symmetry of the data and to enable training without additional bias term, the two input spike times were augmented by $t^\mathrm{I}_3 = (1 - x_1)\tau_\mathrm{in}$ and $t^\mathrm{I}_4 = (1 - x_2)\tau_\mathrm{in}$. Following the proposition of the inventors, we in addition introduced a bias spike at time $t^\mathrm{I}_5 = t_\mathrm{bias}$. These five input spikes were used to classify the $N_\mathrm{C} = 3$ classes Yin, Yang and Dot. In order to accommodate to the input strength and weight resolution of HICANN-X, each spike was multiplexed five times. The resulting 25 spike times were used to stimulate our SNNs via a set of 25 synapses, each of which transmitting a single spike per input sample. To increase the temporal resolution, we employed $n = 2$, i. e. a two-fold interpolation of the acquired CADC membrane traces.

MNIST: The size of the raw MNIST images was reduced from 28×28 to 16×16 pixels. This was achieved by first discarding the two outermost rows of pixels and second by scaling these remaining pixels. To convert the images to spike times, the normalized greyscale value of each pixel x_i was interpreted as currents to a LIF neuron. If this current is strong enough, it triggers a spike at time:

$$t^\mathrm{I}_i(x_i) = \tau_\mathrm{in}\log\left(\frac{x_i}{x_i - \theta_\mathrm{in}}\right) , \tag{5.10}$$

where τ_in corresponds to the input units time constant and θ_in to its threshold. The resulting 256 spike trains – each of which comprising a single event – were used to classify the $N_\mathrm{C} = 10$ distinct digits.

Following established methods, we employed data augmentation to the raw MNIST images by introducing random rotations up to $\vartheta_\mathrm{r} = 15°$. Furthermore, we investigated dropout to the hidden layer with a probability of $p_\mathrm{d} = 10\,\%$ to improve generalization performance.

SHD: Since the SHD dataset directly provides spike trains (Chapter 4), a custom conversion of auditory signals was not required (Cramer et al., 2020b). We, however, reduced the input dimensionality to accommodate to a manageable fan-in for HICANN-X by sampling $N_\mathrm{I} = 70$ of the original 700 input units by omitting the first 70 and then selecting every ninth. Furthermore, the time dimension was compressed by a factor of $\gamma = 2000$ to reflect the accelerated system's speedup of 1000

Table 5.1: Parameters for the neuromorphic substrate, training framework and datasets. All parameters are given in the hardware time domain. The errors given for neuronal parameters correspond to the standard deviation of all neurons. Table adapted from Cramer et al. (2021).

Stage	Parameter	Symbol	Value		
			Yin-Yang	MNIST	SHD
Neuro-synaptic dynamics	Difference threshold-leak potential	$u_{\text{thres}} - u_{\text{leak}}$	(270 ± 15) mV	(270 ± 15) mV	(270 ± 15) mV
	Membrane time constant	τ_{mem}	(8.6 ± 1.1) µs	(5.7 ± 0.3) µs	(8.6 ± 1.2) µs
	in computation graph		10.0 µs	6.0 µs	10.0 µs
	Synaptic time constant	τ_{syn}	(6.5 ± 0.1) µs	(6.5 ± 0.1) µs	(11.2 ± 0.5) µs
	in computation graph		6.0 µs	6.0 µs	10.0 µs
Network	Input size	N_{I}	5	256	70
	Hidden layer size	N_{H}	120	246	246
	Label layer size	N_{L}	3	10	20
	Mean of weight initialization	$\hat{\mu}_w$	0.0	0.0	0.0
	Stdev of weight initialization	$\hat{\sigma}_w$	0.17	0.17	0.17[1]
	Dropout probability	p_{d}	–	10 %	–
Input	Input unit time constant	τ_{in}	42 µs	8 µs	–
	Input unit threshold	θ_{in}	–	0.2	–
	Bias time	t_{b}	$0.45\tau_{\text{in}}$	–	–
	Random rotations	ϑ_{r}	–	15°	–
	Input speedup	γ	–	–	2000
	Input unit jitter	σ_{u}	–	–	15
General	Time step/sample period	δt	0.85 µs	1.7 µs	1.7 µs
	Number of time steps	N_{T}	20	20	200
Optimization	Training epochs	N_{E}	100	400	50
	Batch size	N_{B}	50	40	50
	Surrogate gradient steepness	β	50	50	50
	Learning rate	η	1.5×10^{-3}	1.5×10^{-3}	1.5×10^{-3}
	Learning rate decay (per epoch)	γ_η	0.03	0.03	0.025
	First moment estimates decay rate	β_1	0.9	0.9	0.9
	Second moment estimates decay rate	β_2	0.999	0.999	0.999
	Stability parameter	ϵ	1.0×10^{-8}	1.0×10^{-8}	1.0×10^{-8}
	Burst regularization strength	ρ_{b}	0.05	0.05	–
	Activity regularization strength	ρ_{a}	–	–	0.6×10^{-3}
	Readout regularization strength	ρ_{r}	–	–	0.5
	Activity regularization threshold	θa	–	–	600

[1] 0.34 for recurrent connections

and to further shorten the experiment duration while at the same time bridging the time scales of auditory signals and neuro-synaptic dynamics.

To improve generalization performance, we employed the same data augmentation strategy for the samples of the SHD as in Chapter 4: A spike previously originating from input channel i was reassigned to a neighboring channel drawn from a normal distribution $\mathcal{N}(\mu = i, \sigma_{\text{u}})$ with standard deviation $\sigma_{\text{u}} = 15$. This augmentation was applied prior to downsampling the inputs.

5.2.5 Weight initialization

All weights were initialized using Kaming's initialization scheme (He et al., 2015). In more detail, the synaptic weights were drawn from a normal distribution $\mathcal{N}(\mu = 0, \hat{\sigma}_{\text{w}}/\sqrt{N})$ with zero mean μ and a standard deviation of $\hat{\sigma}_w/\sqrt{N}$. Here, N corresponds to the number of afferent connections, i. e. the number of input units N_{I} or hidden layer neurons N_{H}, respectively.

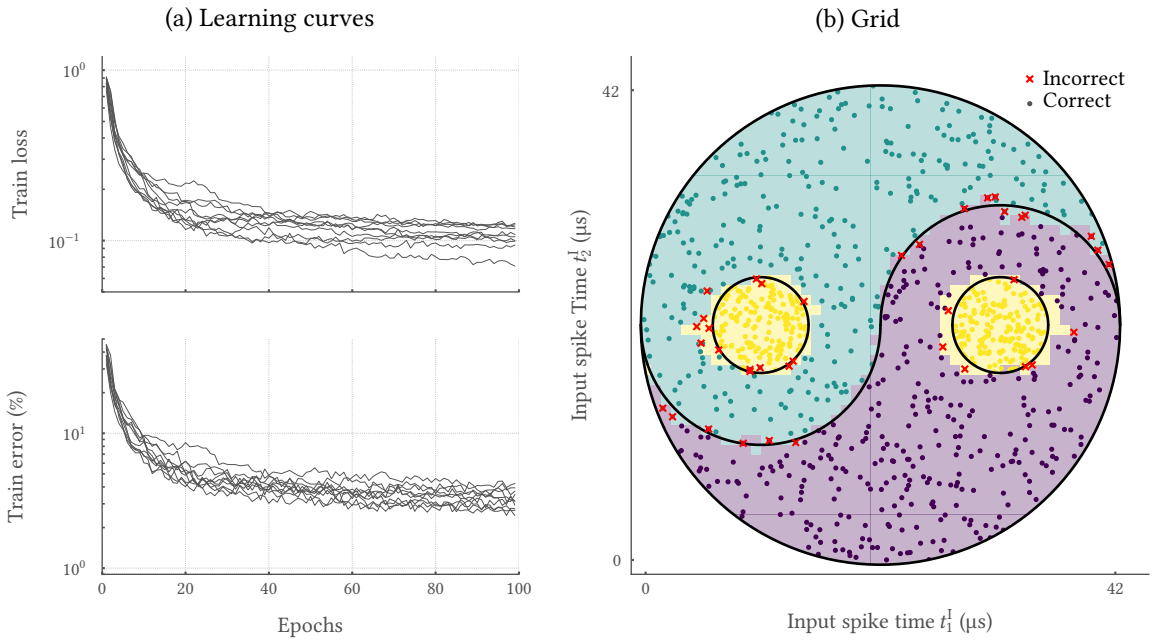


Figure 5.6: Classification of the Yin-Yang dataset with SNNs trained with surrogate gradients. Feed-forward networks with a single hidden layer composed of 120 LIF neurons are able to replicate the Yin-Yang figure. **(a)** After about 40 training epochs with the surrogate gradient-based ITL training framework, the emulated SNNs converged to a state of high performance. Shown are loss and accuracy over the course of 100 training epochs for ten initial conditions. **(b)** Trained networks closely resemble the Yin-Yang figure with errors occurring only in the vicinity of the decision boundaries. To inspect the decision boundaries, the response of the network can be evaluated on a grid spanned by the two input spike times t_1^I and t_2^I. All 1000 samples constituting the test set are overlayed and colored according to the network's response. Wrongly classified samples are marked by a red cross. The ground truth decision boundaries are highlighted in black.

5.3 Results

In the following, we discuss results achieved by training the analog neuromorphic HICANN-X chip with our ITL learning framework on a set of artificial as well as real-world benchmark tasks. The underlying datasets are chosen to challenge different aspects of SNNs as well as to mimic different application areas. Moreover, they exhibit widely different time scales and hence suggest both feed-forward as well as recurrent architectures. To put our experiments into a broader context, we also present results obtained by feed-forward networks employing a TTFS coding scheme trained with spike time gradients, likewise emulated on HICANN-X.

5.3.1 Resembling non-linear decision boundaries with feed-forward SNNs

First, we inspect the ability of emulated SNNs trained with our surrogate gradient-based framework to represent non-linear and continuous decision boundaries. To that end, we trained a feed-forward network with a single hidden layer composed of 120 neurons on spike-encoded samples drawn from the Yin-Yang figure (Kriener et al., 2021). Here, the temporal precision of trained SNNs can be assessed since samples belonging to different classes can have an arbitrarily small distance at the decision boundaries. The dataset as well as the decision of the network can be visualized in a 2D plane spanned by the input spike times (Figure 5.6). After 100 training epochs, our SNN learned

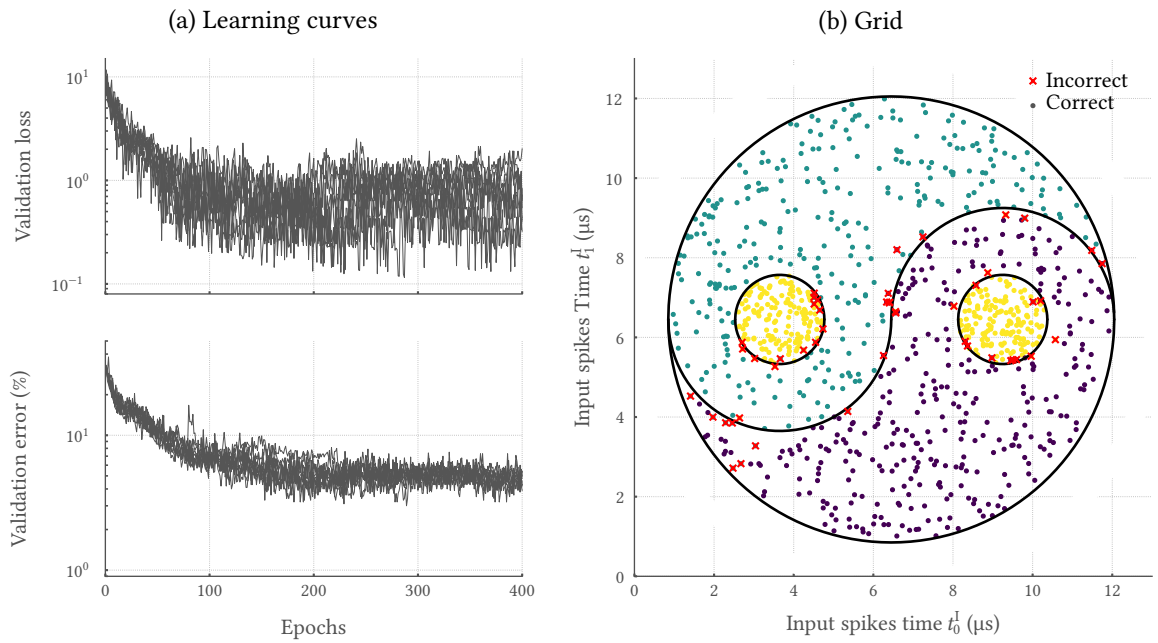


Figure 5.7: Classification of the Yin-Yang dataset with SNNs trained with spike time gradients. Feed-forward networks with a single hidden layer composed of 120 LIF neurons were able to replicate the Yin-Yang figure. **(a)** After about 200 training epochs with the spike time gradient ITL learning, the SNNs learned to closely resemble the validation data. Shown are validation loss and accuracy over the course of 400 training epochs for 10 initial conditions. **(b)** The latencies of spikes emitted by the trained SNN closely resemble the Yin-Yang figure. The classification result for all 1000 test samples is visualized in a 2D plane spanned by the two input spike times t_x^I and t_y^I. The class inferred by the network is highlighted by color for each test sample. All wrongly classified samples are marked by red crosses. The ground truth decision boundaries are highlighted in black. Figure adapted from Göltz et al. (2021).

to closely resemble the Yin-Yang figure resulting in a final test performance of $(96.7 \pm 0.8)\,\%$. Most notably, errors only occur in close vicinity to the decision boundaries. It is noteworthy that the surrogate gradient approach allows us to train the SNN starting from a quiescent hidden layer, even without a homeostatic regularization term enforcing network activity (cf. Chapter 6).

The test accuracy reached by our neuromorphic SNNs is only slightly surpassed by equivalent SNNs trained in software. Specifically, the latter reach a final performance of $(97.0 \pm 0.0)\,\%$ on the held-out test samples. Most notably, these simulations did not take into account all sources of noise as well as hardware-specific constraints like the finite resolution of the synaptic weights as well as CADC recordings. The lack of these features within simulations as well as the only minor gap between the simulation and emulation performance underpins the robustness of our ITL learning framework.

Our surrogate gradient approach outperforms the alternative hardware implementation drawing on spike time gradients for the training of SNNs. After 400 epochs, these networks learned to stably separate the three distinct classes of the Yin Yang dataset (Figure 5.7) with a slightly reduced test performance of $(95.0 \pm 0.9)\,\%$ (Göltz et al., 2021). Just like for SNNs trained with surrogate gradients, errors only occur in close vicinity to the decision boundaries (Figure 5.7b). Similarly sized networks simulated and trained in software reach an accuracy of $(95.9 \pm 0.7)\,\%$ after 300 epochs (Table 5.2). In summary, both the simulation as well as the emulation results on networks trained with spike time gradients slightly fall behind the ones reached with SNNs trained with surrogate gradient methods.

The results achieved with both training methods do not only highlight the robustness with respect to non-linear decision boundaries, but also showcases the fine-grained temporal resolution achievable with SNNs emulated on HICANN-X. For the Yin-Yang task, the maximum achievable accuracy does not only depend on the hidden layer size, but also on the temporal precision at which spikes can be injected into the SNN as well as the resolution at which the network is able to signal its decision. For our surrogate gradient-based framework, we usually align the spike times to a regular grid with N_T steps and step size $\delta t = 1.7\,\mu\mathrm{s}$ determined by the sample interval of the CADC. As a result, the temporal resolution is limited by δt despite the fact that spikes are recorded with much higher precision. Here, however, we employed an interpolation $\delta t/n$ with $n = 2$ to exploit the high temporal resolution of spike times. Similarly, N_T could be increased to improve the precision. This strategy, however, is likely to entail increased calibration targets for both synaptic and neuronal time constants to maintain the support of neuro-synaptic dynamics. In contrast, the TTFS approach also employs spike time coding in the readout layer and does not rely on CADC measurements during training. Hence, the temporal resolution is only limited by the precision at which spikes can be injected into and read out from the neuromorphic device which is way higher than the sample interval of the CADC. However, even with a two-fold interpolation, the surrogate gradient-based approach surpasses the performance achieved with SNNs employing TTFS coding. This is likely due to the limited amount of test samples provided by the Yin Yang dataset, rendering the resolution especially at the decision boundaries sufficient (Figure 5.6a). Moreover, the incorporation of the membrane potential into the training is likely to promote robustness and precision.

5.3.2 Classification of images with feed-forward SNNs

As a real-world application, we trained an SNN with our surrogate gradient-based framework on the MNIST dataset (LeCun et al., 1998). In more detail, we consider feed-forward SNNs with a single hidden layer composed of 246 LIF neurons on a spike-latency encoded version of the MNIST images with 16×16 pixels. Here, the neuromorphic SNNs learned to correctly encode the class affiliation of a presented sample in the maximally responsive output unit during training (Figure 5.8a and b). Most notably, the inhibition of all but the correct classes emerged as a byproduct and was not explicitly enforced within the loss function, Equation (4.9). After only 100 training epochs, the SNNs were optimized to almost perfectly fit the training data. On the held-out test data, the trained networks achieve an overall accuracy of $(97.2 \pm 0.1)\,\%$ (Table 5.2). By augmenting the training data by applying random rotations of up to 15°, we were able to reduce overfitting. Additionally, we incorporated dropout regularization to weights connecting the input and the hidden layer to further improve test performance. A combination of both strategies results in a best-effort accuracy of $(97.6 \pm 0.1)\,\%$ on HICANN-X (Table 5.2).

For validation purposes, we trained equivalently sized SNNs purely in software. These networks only slightly outperform the hardware emulation with a best-effort accuracy of $(97.8 \pm 0.0)\,\%$. For comparison, we in addition trained equivalently sized ANNs with rectified linear units (ReLUs) on the 16×16 pixel version of MNIST. These networks reach a test accuracy of $(98.1 \pm 0.1)\,\%$. Similar to the results presented above, dropout as well as data augmentation again allow us to improve upon this benchmark with a best-effort performance of $(98.7 \pm 0.1)\,\%$. Most notably, these numbers are on par – within their uncertainties – with the accuracy achieved by ANNs trained on the full-size MNIST images (Table 5.2). Hence, our input dimensionality reduction seems to only marginally impact performance. In turn, the accuracies reported for the reduced as well as the full-size MNIST

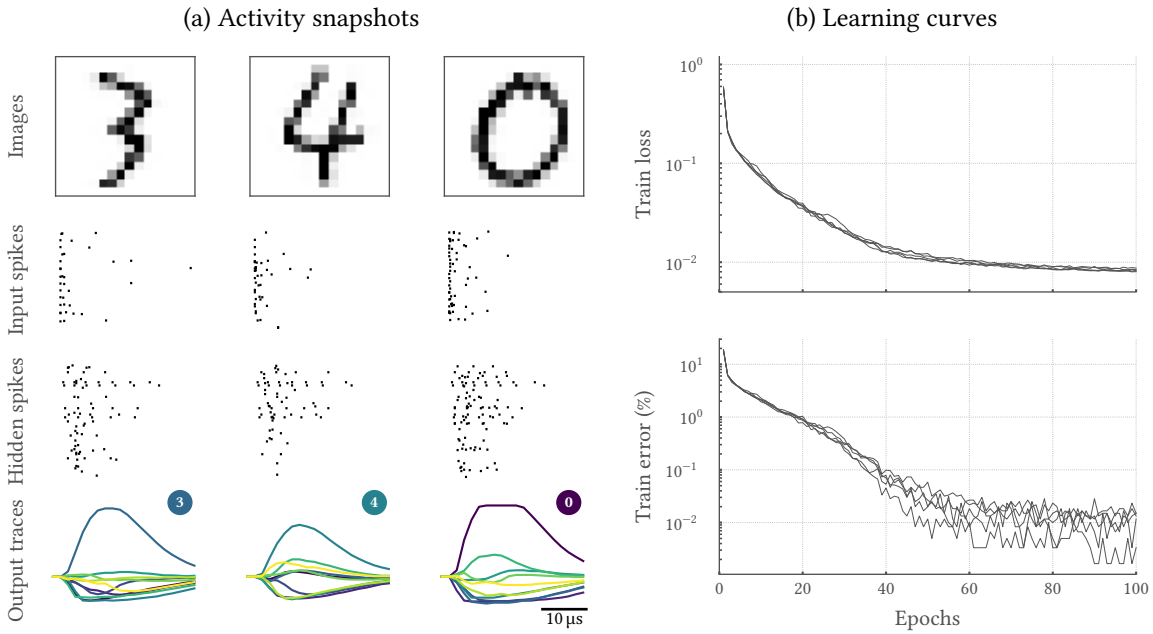


Figure 5.8: Classification of the MNIST dataset with SNNs trained with surrogate gradients. Feed-forward networks with a single hidden layer composed of 246 LIF neurons emulated on HICANN-Xs are able classify the MNIST images with high accuracy. **(a)** These emulated SNNs learned to correctly encode the class affiliation of the MNIST images in the maximally responsive output unit. Shown are three exemplary downscaled 16×16 MNIST images prior and after spike conversion (top) as well as the elicited hidden layer spiking activities (middle) and the readout neuron traces (bottom). The readout traces are colored according to their class affiliation. The ground truth of the presented digit as well as the corresponding color is highlighted inside the bottom panel of the figure. **(b)** The SNNs are able to almost perfectly fit the training data. Shown are training loss and accuracy traces over the course of 100 training epochs for five initial conditions. Figure taken from Cramer et al. (2021).

images are likely to be comparable.

Networks trained with spike time gradients are likewise able to classify the MNIST images with high accuracy. In more detail, emulated feed-forward SNNs with 246 hidden neurons reach a test performance of $(96.6 \pm 0.1)\,\%$ after 50 training epochs when resorting to data augmentation (Figure 5.10a). Equivalent software simulations surpass this result with a final test performance of $(97.3 \pm 0.1)\,\%$ (Table 5.2). On the one hand, both numbers slightly fall behind the corresponding results achieved with the surrogate gradient-based learning scheme. On the other hand, the gap between software and hardware results is slightly higher for networks employing TTFS coding. This hints towards the impact of discrepancies between the emulation in the forward pass and the software model used to calculate weight updates. The only minor difference between the performances obtained by simulation and emulation for the surrogate gradient approach stresses the favorable role of the membrane potential for the close alignment of emulation and equivalent software model on the host computer within the ITL learning scheme.

5.3.3 Low-latency classification

The spike-latency encoding of MNIST images promotes low decision times. Here, we analyze the inference latency of SNNs trained with our surrogate gradient-based ITL framework by artificially

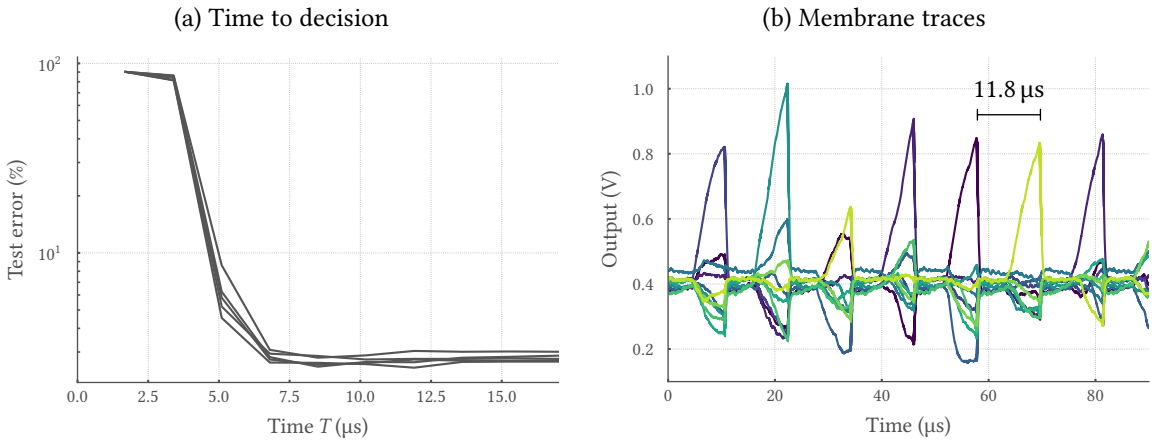


Figure 5.9: Spike-latency encoding of MNIST images promotes low-latency classification. (a) SNNs trained with our surrogate gradient-based ITL learning framework make their decision prior to 10 µs after stimulus onset. Here, we iteratively re-evaluate the max-over-time on the output traces (Figure 5.8a) restricted to a limited time intervals $[0, T]$. **(b)** The low classification latency facilitates high throughput in a fast inference mode. By artificially resetting the state of neuromorphic SNNs between successive samples, we are able to inject an image every 11.8 µs, translating to more than 85 k classifications per second. Figure taken from Cramer et al. (2021).

restricting the membrane traces of the readout layer to time intervals $[0, T]$ with $T \leq N_\mathrm{T} \cdot \delta t$ (Figure 5.8a). Based on this segment, we determine a maximally responsive output unit. By sweeping T, we find that our SNNs reach their peak performance within the first 8 µs after stimulus onset (Figure 5.9a). Hence, the emulation over the entire time span $N_\mathrm{T} \cdot \delta t$ used during training is not at all required to reach high accuracy during inference.

To ensure that this low classification latency can be exploited for high inference rates, the state of the SNN needs to be reset. Especially in feed-forward SNNs, the propagation and forgetting of information are determined by the neuronal and synaptic time constants. They determine the time scale at which the underlying state variables decay back to their respective baseline values. Because of this, the time constants pose a lower limit on the maximum rate at which stimuli could be injected into the network without interfering with each other. Thus, the synaptic as well as the neuronal state need to be reset to not wait for their natural decay to exploit the aforementioned low latency classification within a fast inference mode. To that end, we triggered an artificial neuron reset 10 µs after the injection of the first input spike of each sample and at the same time clamped the synaptic inputs to their baselines (Figure 5.9b). Both of which can be achieved in parallel by drawing on the SIMD vector units of both PPUs. By this, we are able to inject images with a temporal separation of only 11.8 µs which translates to a throughput of 85 k images per second. Within this fast inference mode, each image is classified within the first 8 µs after stimulation with the first spike.

Furthermore, we quantified the power consumption of the system during inference. For the full ASIC emulating the trained SNN, we measured approximately 200 mW. This number encompasses the current draw from the analog neuromorphic core, the plasticity processors, all surrounding periphery, as well as the highspeed communication links. It is noteworthy that the overall figure was dominated by the idle power spent on clock generation and distribution as well as the communication links to and from the ASIC. In combination with the measured throughput, this yields an energy of only 2.4 µJ per classified MNIST image.

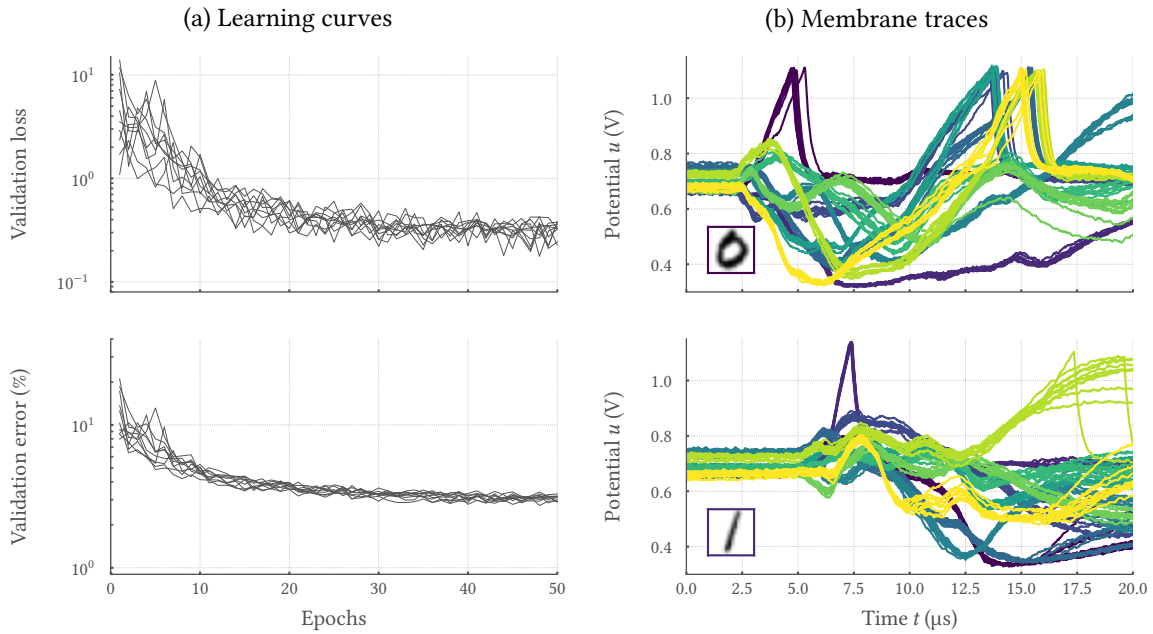


Figure 5.10: Classification of the MNIST dataset with SNNs trained with spike time gradients. Feed-forward networks with a single hidden layer composed of 246 LIF neurons emulated on HICANN-X are able classify the MNIST images. **(a)** After about 40 training epochs with the spike time gradient-based ITL approach, our networks converged to a state of high performance. Shown are loss and accuracy over the course of 50 training epochs for 10 initial conditions. **(b)** Trained SNNs robustly make their decision within the first 10 µs after stimulus onset. Learning leads to sufficiently separated spike times of the label neurons to not suffer from trial-to-trial variability induced by the analog emulation. Shown are 10 exemplary membrane traces for each readout unit recorded on HICANN-X after training. Each trace is colored according to the readout neuron's identity. The color of the box in the lower right part of the figure depicts the ground truth of the presented stimulus. The stimulating sample is shown in the inset of the box. Figure adapted from Göltz et al. (2021).

Likewise, the TTFS coding scheme in combination with the spike-latency encoding of MNIST images leads to low classification latencies on HICANN-X. Post-training membrane traces recorded from the analog substrate highlight that the correct label neuron emitted a spike prior to 10 µs after stimulus onset (Figure 5.10b). Moreover, this *first* spike is clearly separated from the spike times of all other label neurons, enforced by the TTFS loss function (Equation (A.14)). This observation explains the robust classification even in presence of substantial spike jitter elicited by the analog emulation. Note that a direct comparison to the 8 µs latency achieved with the surrogate gradient-based training should keep in mind the distinct image to spike conversion schemes: While the results presented before were obtained with LIF neurons in the receptor layer and hence a logarithmic mapping from greyscale value to spike time, the results for SNNs employing TTFS coding were acquired with a linear transformation.

The emulated networks employing TTFS coding are able to process MNIST images with a temporal separation of 48 µs which corresponds to a throughput of about 20 800 images per second. During inference, the ASIC consumed approximately 175 mW and hence slightly less compared to the surrogate gradient-based approach which is due to the omission of PPU triggered CADCs measurements. In combination with the observed throughput, this yields an energy of 8.4 µJ per classification. It is noteworthy that this figure does not include an artificial reset of the SNN's state variables. As-

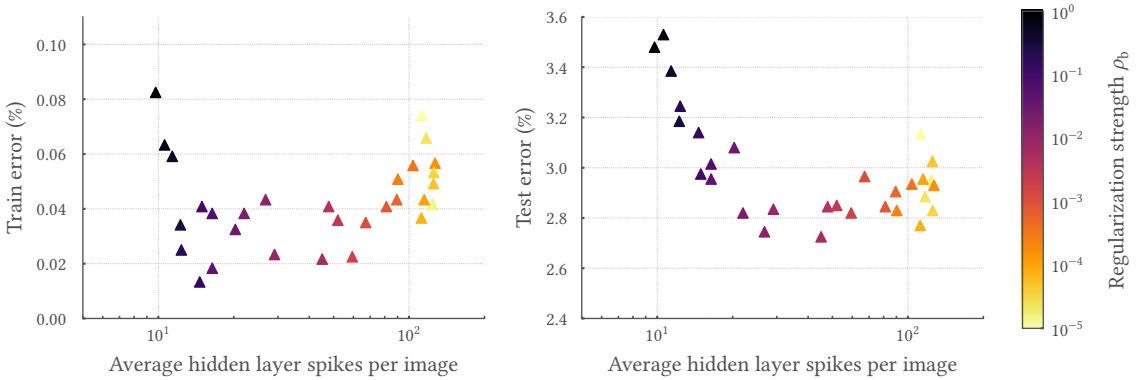


Figure 5.11: Regularization facilitates the efficient classification of MNIST images with sparse spiking activity. The generality of our surrogate gradient-based ITL training framework allows us to augment the task loss by an adjustable burst regularizer with strength parameter ρ_b to promote sparse spiking activity. When sweeping ρ_b, trained SNNs reach a constant test accuracy of about 97.2 % for activity levels down to only 20 hidden spikes per input image. Figure taken from Cramer et al. (2021).

suming a similar temporal overhead of 3.8 µs for resetting, a theoretical throughput of more than 72 k images could be reached which translates to an energy of 2.6 µJ. Here, we assumed a power consumption of 200 mW since the PPUs would be additionally required for the implementation of a fast inference mode.

5.3.4 Efficient coding through sparse spiking activity

The spiking activity of neurons in the mammalian cortex is remarkably sparse with an average of only 0.2 to 5 spikes per second (Attwell & Laughlin, 2001). This sparsity is expressed in both time and space and is assumed to be central for the low energetic footprint of biological neural tissue (Sterling & Laughlin, 2015). Like for their role model, neuromorphic systems emulating SNNs similarly profit from sparsity, especially when multiple chips need to be interfaced. Here, an information exchange characterized by spatio-temporal sparsity directly relieves the bandwidth of the associated communication channels.

The generality of our surrogate gradient-based ITL learning framework allows us to promote sparse solutions by augmenting the task loss with an adjustable regularizer. Specifically, we trained feed-forward SNNs on the MNIST images with the burst penalty defined in Equation (5.6) for a broad range of strength parameters ρ_b. Simultaneously, we assessed both the performance as well as the average hidden layer spike counts of these SNNs. For all tested ρ_b, our networks were almost able to perfectly fit the training data (Figure 5.11a). Even more important, these SNNs reach a constant test accuracy of about 97.2 % for activity levels down to 20 hidden spikes per input image (Figure 5.11b). Only for 10 spikes and below, we observe a noticeable decline of performance. For these sparse solutions, the information is carried by individual spikes and their timing which stays in strong contrast to rate coding approaches.

5.3.5 Self-calibration of the analog circuits through ITL training

The incorporation of observables measured from the neuromorphic substrate renders our approach unsusceptible to parameter deviations. All results presented so far were gathered with a calibrated

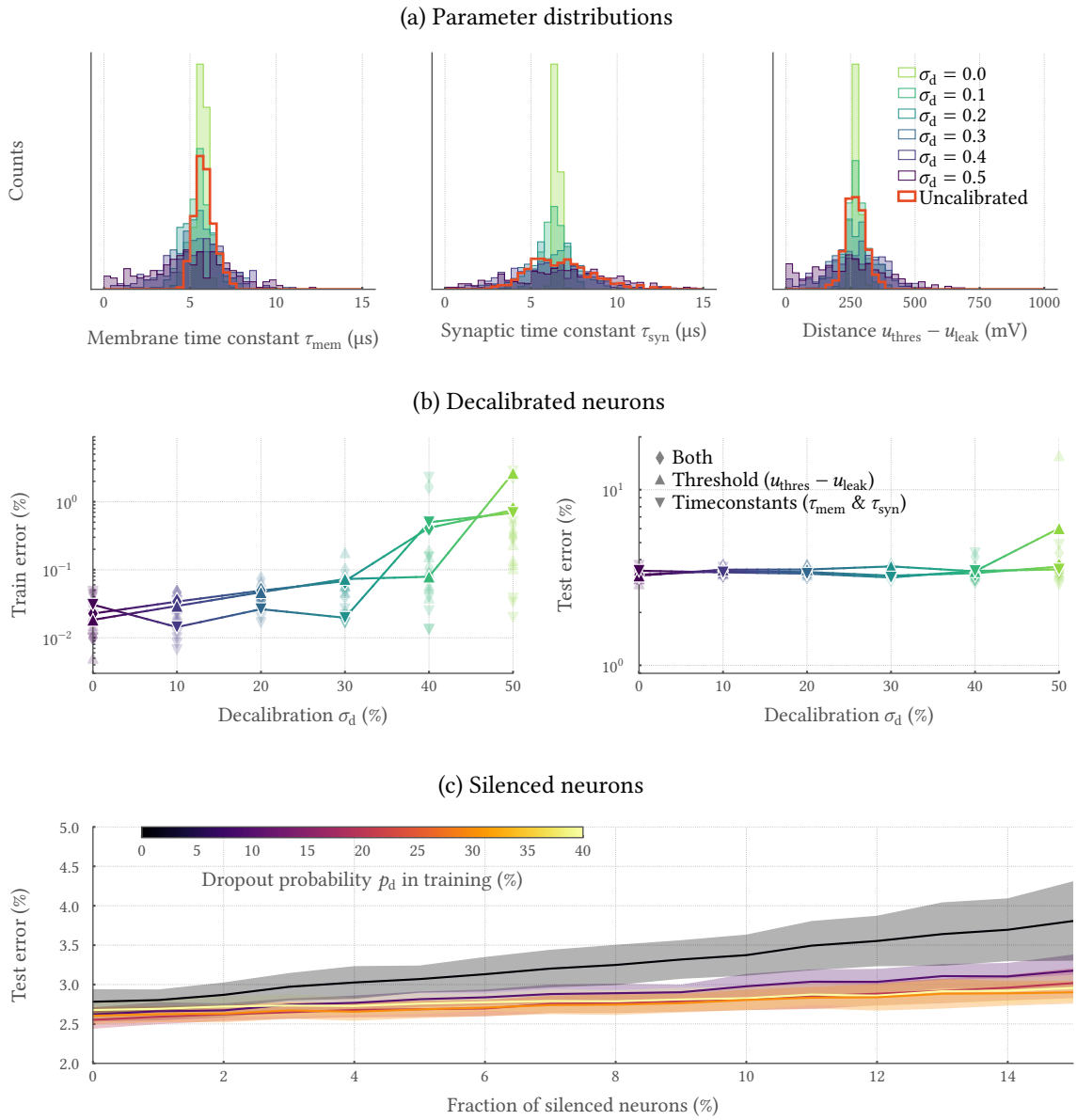


Figure 5.12: Surrogate gradient-based ITL learning promotes robust training and classification on inhomogeneous substrates. Our surrogate gradient-based ITL framework inherently compensates for parameter variations induced by the analog implementation and can be further extended to improve robustness to defects occurring after deployment. **(a)** The configurability of HICANN-X allows us to control the deviation of model parameters. For that purpose, a calibration with neuron-specific target values drawn from normal distributions of variable widths σ_d was utilized. Shown are the distributions of the measured values for the membrane time constant τ_{mem} (left), the synaptic time constant τ_{syn} (middle) and the distance between threshold and leak potential $u_{thres} - u_{leak}$ (right) for different values of σ_d (colored). For comparison, the distributions stemming from an uncalibrated HICANN-X chip are overlayed in red. **(b)** The fixed-pattern variations shown in (a) are widely compensated by ITL training, despite assuming homogeneously behaving circuits in the computation graph. Configurations with high σ_d already suffer from dysfunctional states. **(c)** SNNs can be trained to exhibit resilience against neuron death occurring after deployment. Here, dropout regularization with probability p_d during training promotes robust classification performance even for scenarios where more than 10 % of the hidden layer neurons are artificially silenced during inference. Figure adapted from Cramer et al. (2021).

HICANN-X chip for which all occurring parameter deviations were widely equalized to closely align the computation graph and the emulated dynamics. It is noteworthy that despite all effort put into calibration, a certain degree of residual parameter variations remained (Figure 5.12a). Moreover, we constructed the computation graph based on the calibration target values which deviate from the actual measured parameters (Table 5.1). Nevertheless, our ITL training leads to well-performing SNNs even in presence of deviations between computation graph and emulated dynamics.

To test the ability of our ITL training framework to compensate for these parameter variations, we deliberately decalibrated the neuromorphic circuits on HICANN-X. In more detail, we adjusted the synaptic and membrane time constants τ_{syn} and τ_{mem} as well as the distance between threshold and leak potential $u_{\text{thres}} - u_{\text{leak}}$ to individual, neuron-specific target values. The latter were drawn from normal distributions $\mathcal{N}(\mu_{\text{d}}, \sigma_{\text{d}})$ with mean μ_{d} corresponding to the original calibration target (Table 5.1). The normalized standard deviation σ_{d} was swept in the range of 0 % to 50 % (Figure 5.12a). Most notably, some of these configurations already exceed the mismatch present on an uncalibrated HICANN-X chip. To further dissect the parameter influence, we first decalibrated τ_{syn} and τ_{mem} at the same time. In a second experiment, we detuned $u_{\text{thres}} - u_{\text{leak}}$ and last we swept all parameters at the same time. For all of these conditions, we trained the SNNs starting from five different initial conditions.

Based on these configurations, we trained our neuromorphic SNNs on the MNIST dataset while still assuming ideal parameters when constructing the computation graph. Despite ignoring the introduced mismatch on the substrate, our SNNs reach a state of high performance for decalibrations of up to $\sigma_{\text{d}} = 30\,\%$ (Figure 5.12b). Most configurations even yield high performance beyond that point while some trials suffer from dysfunctional network states, observable by increased errors. Particularly leak-over-threshold configurations induced by high values of σ_{d} for the distance $u_{\text{thres}} - u_{\text{leak}}$ harm performance. However, for parameter variations comparable to the ones present on an uncalibrated HICANN-X chip and even beyond, our ITL learning framework self-calibrates the analog neuromorphic SNN. Thus our framework renders detailed calibration routines redundant.

5.3.6 Training for robustness to defects

Power efficiency as well as low-latency classification render SNNs implemented on HICANN-X particularly interesting for edge computing applications (Khan et al., 2019; Shi & Dustdar, 2016). Here, the resilience of trained SNNs to hardware defects emerging after deployment is crucial. Among others, failing neuron circuits have the potential to put functionality at risk. The latter effect can be imitated by artificially disabling a fraction of all hidden neurons during inference (Figure 5.12c). Maybe not surprisingly, the MNIST test accuracy drops with the fraction of silenced neurons.

We can, however, reduce the sensitivity to dysfunctional hidden neurons by including dropout regularization during training. Before, we already showcased the beneficial role of dropout on generalization performance (Table 5.2). Aside from reducing overfitting, it also raises the resilience to an increased fraction of silenced neuron circuits (Figure 5.12c). When disabling 15 % of all hidden layer neurons, the test error increases by only 10 % for networks trained with dropout regularization with a probability of $p_{\text{d}} = 40\,\%$. In contrast, the error rises by 37 % for SNNs trained without dropout regularization.

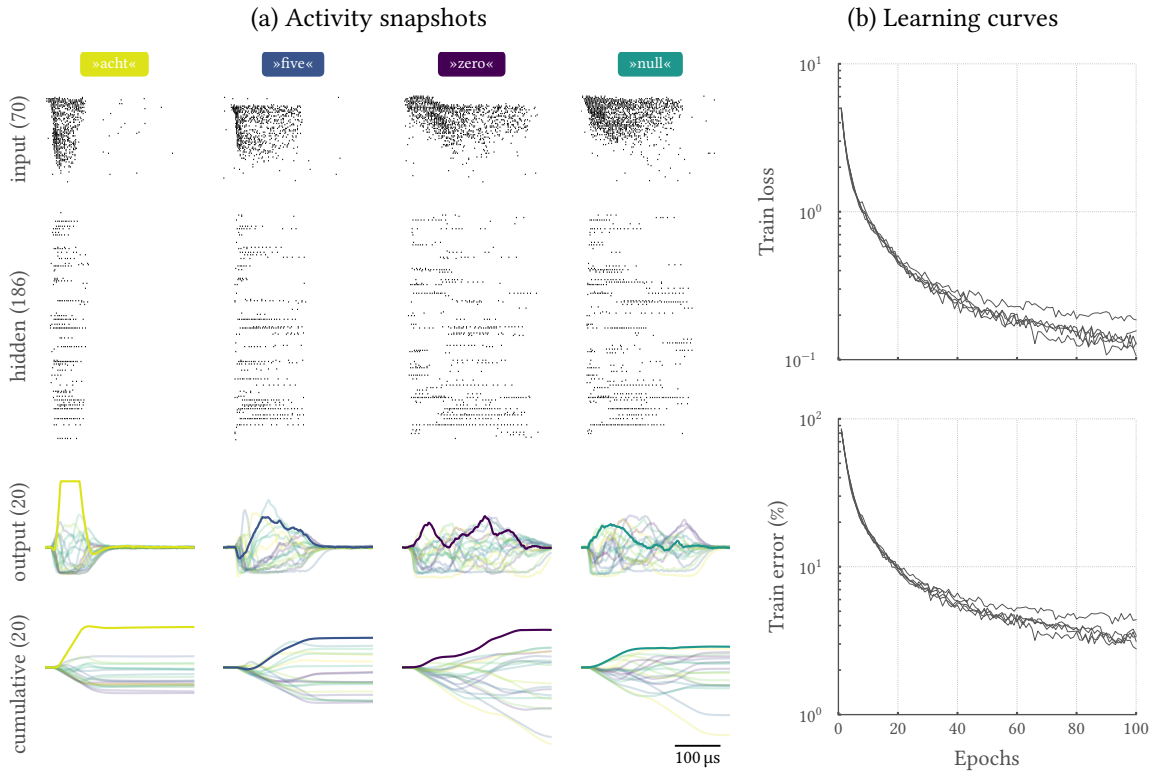


Figure 5.13: Reucurrent SNNs trained with surrogate gradients promote high performance in a natural language classification task. The surrogate gradient-based ITL framework facilitates the training of recurrent SNNs and in that process allows to us exploit a combination of rate- and temporal coding. Specifically, recurrent SNNs with a single hidden layer composed of 186 LIF neurons promote competitive accuracy levels on the SHD. **(a)** This network learned to correctly infer the class affiliation of spoken digits in the total membrane deflection of the readout neurons. Shown are snapshots of the spike converted digits (top), the elicited hidden layer activity as well as the readout neuron traces (middle). For visualization, we also show the cumulative sums of the latter (bottom) to illustrate the decision of the SNN trained with a sum-over-time loss function. Readout traces and their respective cumulative sums are colored according to their class affiliation. The color of the ground truth of the presented digit is highlighted at the top of the figure. **(b)** Only 100 training epochs are required to develop suitable representations as evidenced by a reduced training loss and error. Both of which are shown for five distinct initial conditions. Figure adapted from Cramer et al. (2021).

5.3.7 Classification of spoken words with recurrent SNNs

Up to this point, we have only considered tasks requiring to memorize and integrate information on comparatively short time scales. In the following, we will move on to benchmarks that require working memory exceeding the time scales of single-neuron dynamics. For SNNs, this memory can be implemented by activity propagating on recurrent connections. The latter are readily supported by the flexible on-chip event router of HICANN-X (Chapter 3). These topologies can be easily supported in our ITL training framework by admitting recurrent contributions to the synaptic currents in Equation (5.2).

As a potential proxy for a real-world time series classification task, we trained recurrent SNNs on the SHD (Cramer et al., 2020b). More specifically, we consider networks with a single recurrent hidden layer composed of 186 LIF neurons to classify the English and German spoken digits »zero« to »nine« into the respective 20 categories (Chapter 4). The underlying time series data naturally

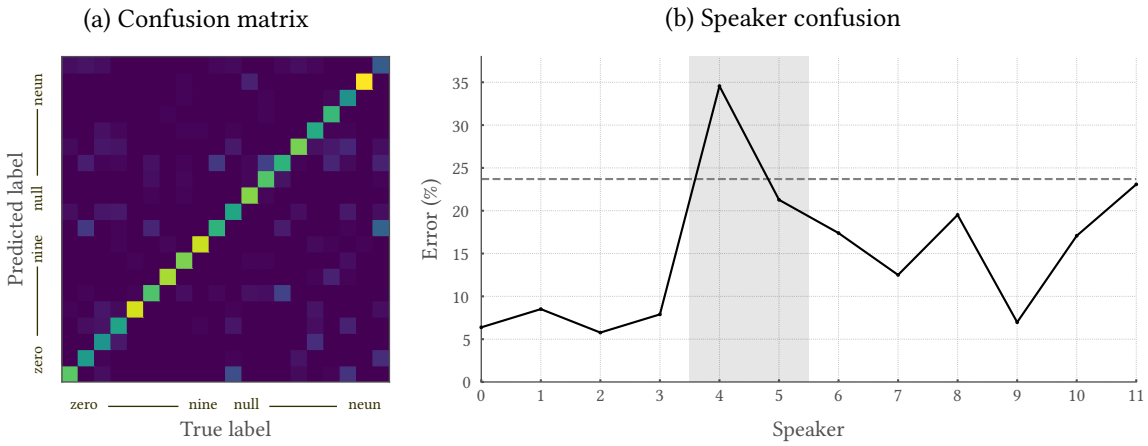


Figure 5.14: Recurrent SNNs emulated on HICANN-X generalize to novel speakers. The choice of classes as well as the composition of the SHD test set challenge generalization. **(a)** Classes exhibiting phonemic similarities like »nine« vs. »neun« are indeed harder to separate for trained SNNs. Shown is the confusion matrix for all 20 classes. **(b)** The trained SNNs generalized on previously unseen speakers. A speaker decoded error allows us to quantify the generalization to samples of new speakers since the digits of the subjects four and five (highlighted in gray) exclusively occur in the SHD test set. They constitute 81 % of all test samples and hence explain the discrepancy between training and overall test error (dashed line). Figure taken from Cramer et al. (2021).

exploits the temporal dimension of SNNs. Furthermore, we have already shown the beneficial role of recurrent connections when training SNNs on this benchmark task in software simulations (Figure 4.10). Here, we reduced the spatial dimensionality of the spike trains of SHD to accommodate to a tractable fan-in while preserving a sufficiently large hidden layer on HICANN-X. To that end, we subsampled 70 out of the original 700 original channels (cf. Methods). It is noteworthy that this choice only reduces the available information present within the samples. In contrast to Chapter 4, our neuromorphic SNNs were trained by minimizing a sum-over-time loss function (Figure 5.13a). Moreover, this task-specific loss was augmented by activity regularization to prevent pathological spiking activity in the hidden layer.

After 100 training epochs, our recurrent SNNs almost fit the training data with an accuracy of (96.6 ± 0.5) %. In this state, we reach a performance of (76.2 ± 1.3) % on the SHD test set (Figure 5.13b and Table 5.2). This rather large gap is expected due to the composition of the test set which is especially designed to challenge the ability to generalize (Chapter 4). On the one hand, the dataset comprises of classes with significant phonemic similarity like e. g. »nine« vs. »neun«. Indeed, these two classes are harder to separate for trained networks (Figure 5.14a). On the other hand, the test set of the SHD exclusively contains the samples of two of the twelve speakers. The associated samples constitute 81 % of all testing data and hence dominate the test set. When assessing a speaker decoded accuracy, an increased error on the samples of the unseen speakers can be observed (Figure 5.14b). Like already proposed in Chapter 4, we were able to reduce overfitting by the utilization of data augmentation. Specifically, we randomly reassigned input spikes originating from unit i to a neighboring channel drawn from the normal distribution $\mathcal{N}(\mu = i, \sigma_\mathrm{u})$ with width $\sigma_\mathrm{u} = 15$ prior to subsampling the spike trains. This strategy leads to a best-effort performance of (80.6 ± 1.0) % on the test data (cf. Chapter 4).

Like for feed-forward architectures, we trained equivalent recurrent SNNs in software only. When not employing data augmentation, we reach a peak performance of (71.2 ± 0.3) % on the SHD test

Table 5.2: Performance comparison of SNNs trained on BrainScaleS-2 and in software. For the downscaled 16×16 MNIST, a dropout probability of 0.1 and random rotations of up to 15° were used. In contrast, the dropout probability was increased to 0.4 for the full size MNIST images . For the SHD dataset, input spikes were randomly switched to neighboring channels according to a normal distribution with width $\sigma = 15$. In addition, we provide an ANN baseline to place our results obtained with SNNs into a broader context. Table adapted from Cramer et al. (2021).

Dataset	Implementation	Gradient	Architecture	Augmentation	Accuracy (%)	
					Train	Test
16 × 16 MNIST	BrainScaleS-2	Surrogate	256-246-10	–	100.0 ± 0.0	97.2 ± 0.1
	BrainScaleS-2	Surrogate	256-246-10	Dropout + rotation	97.3 ± 0.1	97.6 ± 0.1
	Software	Surrogate	256-246-10	–	100.0 ± 0.0	97.5 ± 0.1
	Software	Surrogate	256-246-10	Dropout + rotation	97.7 ± 0.1	98.0 ± 0.0
	BrainScaleS-2	Firing time	256-246-10	Temporal spike jitter	98.2 ± 0.1	96.6 ± 0.1
	Software	Firing time	256-246-10	Temporal spike jitter	99.2 ± 0.1	97.3 ± 0.1
	ANN	Regular	256-246-10	–	100.0 ± 0.0	98.1 ± 0.1
	ANN	Regular	256-246-10	Dropout + rotation	99.0 ± 0.0	98.7 ± 0.1
28 × 28 MNIST	ANN	Regular	784-246-10	–	100.0 ± 0.0	98.1 ± 0.1
	ANN	Regular	784-246-10	Dropout + rotation	98.0 ± 0.0	98.7 ± 0.1
Yin-Yang	BrainScaleS-2	Surrogate	5-120-3	–	96.7 ± 0.6	96.7 ± 0.7
	Software	Surrogate	5-120-3	–	98.0 ± 0.0	98.7 ± 0.1
	BrainScaleS-2	Firing time	5-120-3	–	95.3 ± 0.7	95.0 ± 1.0
	Software	Firing time	5-120-3	–	96.0 ± 0.7	96.3 ± 0.7
SHD	BrainScaleS-2	Surrogate	70-186-20	–	96.6 ± 0.5	76.2 ± 1.3
	BrainScaleS-2	Surrogate	70-186-20	Spatial spike jitter	90.7 ± 0.5	80.6 ± 1.0
	Software	Surrogate	70-186-20	–	100.0 ± 0.0	71.2 ± 0.3
	Software	Surrogate	70-186-20	Spatial spike jitter	90.9 ± 0.2	79.9 ± 0.7

set (Table 5.2). This, most notably, is far below the result achieved by SNNs emulated on HICANN-X. However, the SNNs implemented in software are able to perfectly fit the training data which is not the case for our hardware implementation (Table 5.2). This discrepancy is most likely due to the intrinsic noise induced by the analog implementation of SNNs on HICANN-X which potentially acts as a form of regularization. In contrast to the previously described feed-forward architectures, this stochasticity gets amplified by recurrent activity. Indeed, we improved the test accuracy of our simulated SNNs to $(79.9 \pm 0.7)\,\%$ when again employing data augmentation in form of the spatial spike jitter. Hence, stochasticity closes the gap between the test performances of hardware emulation and software simulation. The same holds true for the training performance which obviously suffers from an increased unit jitter width σ_u. Thus, the intrinsic noise of the analog neuromorphic substrate could indeed act as an efficient regularizer. Most notably, we only incorporated the stochastic data augmentation into our simulations and ignored further sources of noise present on the neuromorphic device which might explain the remaining gap between performances. Summing up, these results suggest that our ITL training framework is also applicable to recurrent SNNs and hence allows us to exploit application areas like the classification of time series data.

5.4 Discussion

Within the scope of this chapter, we developed a general ITL training framework relying on surrogate gradient learning. With this, we were able to train recurrent as well as multi-layer SNNs on the analog neuromorphic substrate HICANN-X. By exploiting the massively parallel recoding of

neuronal membrane potentials with the CADCs, we were able to bring the success of surrogate gradient learning to the realm of neuromorphic hardware. We benchmarked our framework based on a set of artificial as well as real-world tasks. Specifically, we visited a spike-latency encoded version of the MNIST dataset to train feed-forward SNNs and the SHD in the context of recurrent architectures. With this, we establish new benchmarks for low-latency processing, energy efficiency as well as task performance compared to other existing neuromorphic implementations. Moreover, the trained networks exhibited sparse spiking activity, both in space and time. Most importantly, the self-correcting nature of our ITL training framework compensated for device mismatch and hence renders detailed calibration routines redundant. All implementations performed at accuracy levels comparable to the ones reached by software simulations of equivalent networks. Ultimately, our approach allowed us to exploit analog neuromorphic hardware for low-latency and high throughput inference.

Most current neuromorphic hardware implementations are based on digital designs and hence often allow to simulate SNNs previously trained in software without a significant loss in performance (Esser et al., 2016; Frenkel et al., 2020). Despite the associated flexibility with regard to the training scheme (Bohte et al., 2002; Rückauer et al., 2019; Zambrano et al., 2017; Pfeiffer & Pfeil, 2018; Büchel et al., 2021; Hunsberger & Eliasmith, 2015; Lee et al., 2016; Neftci et al., 2019b; Mostafa, 2017; Bellec et al., 2020; Huh & Sejnowski, 2018) recent achievements in material science let to a refocus on analog and mixed-signal implementations (Roy et al., 2019; Marković et al., 2020; Joshi et al., 2020; Dalgaty et al., 2021). The accompanying findings have the potential to promote the development of efficient neuromorphic processors which in turn require training schemes to fully exploit their benefits. A key component to implement long-term memory on such devices are memristors, which are intrinsically analog and are subject to both drift and manufacturing variability. However, the latter exacerbates a direct mapping of software-trained models to these devices. Many studies tried to solve this problem by optimizing the implemented networks for robustness during training, but the associated strategies are of limited applicability (Büchel et al., 2021; Wright et al., 2021). As long as mature on-chip training solutions are not available, ITL has emerged as a compromise (Schmitt et al., 2017; Göltz et al., 2021). It incorporates device-specific non-idealities and heterogeneities into the training and in that process compensates for them.

Previous work on ITL learning in the context of analog neuromorphic computing mostly focused on rate-coding strategies (Schmitt et al., 2017). However, these approaches often entail higher activity levels and longer latencies and therefore do not exploit the favorable properties of SNNs. Likewise, TTFS coding schemes – constituting the extreme end of temporal coding – have been successfully applied to analog devices, but only in combination with feed-forward architectures and single spikes (Göltz et al., 2021). In contrast, surrogate gradient learning exploits and interpolates between both rate and temporal coding, and can be applied to multi-layer as well as recurrent SNNs. By extending current ITL techniques, we were able to bring the success of surrogate gradient methods to the realm of analog neuromorphic hardware. With this, we could not only train networks to reach state-of-the-art performance for SNNs on a set of benchmark datasets, but in that process also improve energy efficiency as well as throughput and latency (Davidson & Furber, 2021).

The incorporation of an additional state variable aside from spike times is likely to play a key role in the ability to self-calibrate the analog substrate. While HICANN-X supports the parallel recording of membrane traces, other analog systems only communicate spikes to the host system and if at all only feature analog-to-digital converters (ADCs) with low channel counts (Schemmel et al., 2010; Benjamin et al., 2014; Moradi et al., 2018). On the one hand, the requirement of membrane traces

Table 5.3: Comparison of MNIST benchmark results across neuromorphic platforms. Performance-wise, our emulated SNNs are only outperformed by larger networks or CNNs. The measured energetic footprint is only surpassed by optimized implementations in smaller and more energy efficient technology nodes. Our SNNs set new benchmarks in terms of both throughput as well as classification latency. Reported energy measurements are normalized to a single inference. Table taken from Cramer et al. (2021).

Platform	Reference	Architecture	Node	Energy	Accuracy	Samples/s	Latency
SpiNNaker	Stromatias et al. (2015)	784-600-500-10	130 nm	–[3]	95.0 %	–[3]	–
TrueNorth	Esser et al. (2016)	CNN	28 nm	108.0 µJ	99.4 %	1 k	–
Intel	Chen et al. (2019)	784-236-20-10	10 nm	1.0 µJ	88.0 %	6.3 k	–
Intel	Chen et al. (2019)	784-1024-512-10	10 nm	12.4 µJ	98.2 %	–	–
Intel	Chen et al. (2019)	784-1024-512-10	10 nm	1.7 µJ	97.9 %	–	–
MorphIC	Frenkel et al. (2019)	784-500-10[1]	65 nm	205 µJ	97.8 %	–	–
MorphIC	Frenkel et al. (2019)	784-500-10[1]	65 nm	21.8 µJ	95.9 %	250	–
SPOON	Frenkel et al. (2020)	CNN	28 nm	0.3 µJ[2]	97.5 %	–	117 µs
BrainScaleS-1	Schmitt et al. (2017)	100-15-15-5	180 nm	–[3]	95.0 %	10 k	–
BrainScaleS-2	Göltz et al. (2021)	256-246-10	65 nm	8.4 µJ	96.9 %	21 k	<10 µs
BrainScaleS-2	Cramer et al. (2021)	256-246-10	65 nm	2.4 µJ	97.6 %	85 k	8 µs

[1] Segmented input and hidden layers [2] Based on pre-silicon estimates [3] Estimates were given by Pfeiffer & Pfeil (2018)

represents a limitation of our work compared to e. g. the approach of Göltz et al. (2021) which only requires spike times to calculate weight updates. On the other hand, however, the incorporation of membrane traces is likely to enhance the resilience to device mismatch present on analog substrates. Hence, the investigation of the performance of SNNs trained with the rule derived by Göltz et al. (2021) under controlled decalibration – like presented within this chapter – would be an interesting direction of future research.

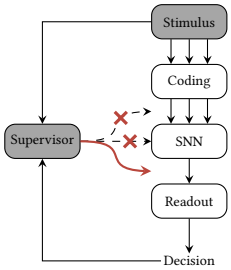
Quantitative comparisons of distinct SNN implementations on various neuromorphic substrates remain challenging. This difficulty is partially due to the lack of suitable metrics and standardized benchmarks (Davies, 2019; Cramer et al., 2020b). Especially, the inhomogeneous strategies for reporting a system's energy consumption put comparability at risk. The latter range from pre-silicon estimates of the current drawn by a neuromorphic processor to full-system lab measurements. This also explains the enormous spread of reported energy consumptions for current neuromorphic implementations (Table 5.3). Here, we nevertheless try to put our energy measurements into a broader context by comparing our results to other analog as well as digital solutions. For that purpose, we visit the MNIST dataset due to its widespread application, despite its lack of a temporal dimension and the non-standardized spike conversion scheme. When considering the energetic footprint, our implementation is only outperformed by optimized designs fabricated in much smaller and therefore more energy-efficient technology nodes (Frenkel et al., 2020; Chen et al., 2019). In terms of task performance, our emulated SNNs are only outperformed by much larger or convolutional neural networks (CNNs) (Stromatias et al., 2015; Frenkel et al., 2019; Chen et al., 2019). To draw on these results, systems with multiple HICANN-X chips or using a single chip in a time-multiplexed fashion to implement multi-layer networks represent a promising way of future research. Even with the current system, we were able to establish new benchmarks in terms of both throughput as well as classification latency (Frenkel et al., 2020; Göltz et al., 2021).

We have already demonstrated the generality of our ITL learning framework by training different network topologies as well as by visiting different task losses and regularization terms. However, our framework can be easily extended to incorporate more features like the training of neuronal parameters or the incorporation of slow varying state variables. The latter are especially promis-

ing since most real-world applications require memory that exceeds the time scale of single-neuron dynamics. We already gave a glimpse of how to extend the memory time span by recurrent connections and the associated reverberating activity for the classification of spoken words. While recurrent SNNs are often unstable and their memory time span is limited, intrinsic recurrence within spiking neurons represents a promising direction of further research (Neftci et al., 2019a). Natural candidates are spike-triggered adaption as well as short-term plasticity (STP) which both introduce additional state variables with slower temporal evolution compared to the neuro-synaptic dynamics themselves (Mongillo et al., 2008; Bellec et al., 2018). Both mechanisms are readily supported on HICANN-X and can be easily included in the existing ITL training framework.

In summary, our ITL training framework allows us to open up the extensive opportunities of SNNs emulated on analog neuromorphic hardware for energy-efficient ultra-low latency information processing. Especially the feedback provided by the construction of the computation graph based on hardware measurements leads to the compensation of device-specific imperfections. The latter feature is key to finally exploit the beneficial features of SNNs emulated on analog neuromorphic hardware for classification. ITL training is, therefore, an intermediate step toward on-chip training which will be ultimately required to exploit the full benefits of this emergent technology.

6 Unsupervised Learning



> The first part of this chapter has been published in Cramer et al. (2020a) in collaboration with David Stöckel and Markus Kreft under the supervision of Dr. Viola Priesemann. For this segment I will closely follow the publication, but with a more detailed description of the implementation. The study presented within the second part has been conducted in close collaboration with Markus Kreft and Dr. Johannes Zierenberg. Early results of the second part have been presented in Kreft (2021).

Despite the exceptional success recorded by the supervised learning approach presented in the previous chapter, the training with gradient-based methods differs from learning in biological neural networks. For the latter, global information might be costly or simply impossible to distribute which is why most biologically inspired learning paradigms draw on the concept of locality to fosters parallelism and scalability. Locality is often exploited in unsupervised learning scenarios where the associated adaptation solely depends on a combination of the stimulus and the structure and/or dynamics of the underlying network. Consequently, the learning of patterns or sequences is accomplished by self-organization. Hence, plasticity rules exploiting these concepts are likewise promising in the context of neuromorphic hardware.

Within the first part of this chapter, we consider a form of plasticity that tunes networks emulated on a prototype of the BrainScaleS-2 system to a so-called critical point – a discontinuous phase transition between order and chaos or stability and instability. In more detail, we show that the associated collective dynamics of our network can be precisely adjusted by changing the input strength under the action of spike-timing dependent plasticity (STDP)-like regulation. The associated critical dynamics maximize a set of abstract computational properties – like the sensitivity, dynamic range, correlation length, information transfer as well as the susceptibility – and were hence assumed to be optimal for task processing with recurrent neural networks. However, we demonstrate that this hypothesis is not generally valid by testing an spiking neural network (SNN) on a set of tasks of varying complexity. Specifically, we investigate the interplay of criticality, task performance and the information-theoretic fingerprint. Most importantly, the information-theoretic measures are in strong contrast to task performance: While all of these measures are maximized at criticality, only the complex, memory-intensive tasks profit from the associated dynamics, whereas the simple tasks even suffer from it. Hence, critical dynamics are not always favorable for task processing. Instead, we show that the collective network state has to be tuned to the computational requirements which can be easily achieved by adjusting the input strength under STDP-like regulation.

In the second part of this chapter, we present a homeostatically regulated system that exhibits bistable dynamics. Harnessing the transition probabilities between states, the time scales of the dynamics could be controlled by the input strength under continuous background stimulation. With

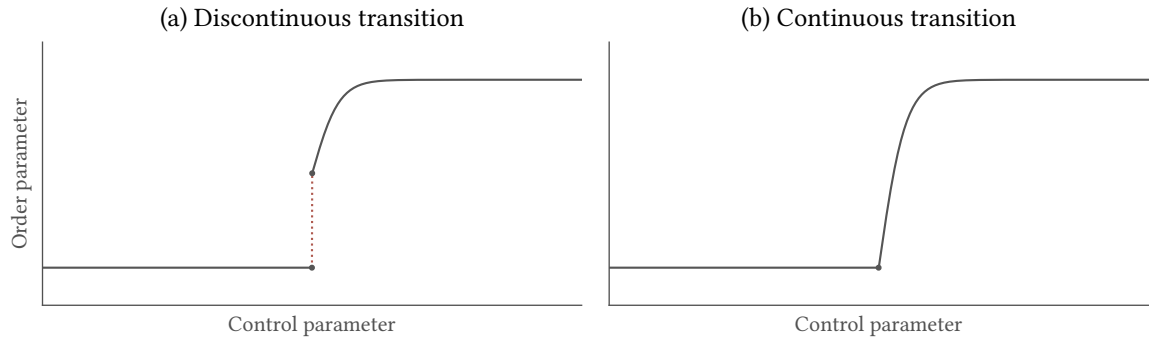


Figure 6.1: Phase diagrams of discontinuous and continuous phase transitions. Different phases are distinguished by macroscopic measurable properties of a system, so-called order parameters. Depending on their change in response to the variation of an ambient property – the so-called control parameter – phases can be classified. **(a)** For a discontinuous phase transition, the order parameter undergoes a discontinuity when the control parameter is posed at its critical value. **(b)** If, in contrast, the transition is continuous, but non-differentiable, the transition is referred to as continuous.

this system, we show how external perturbations can be used to accurately determine the autocorrelation time of the underlying dynamics. More specifically, the time scale of the decay of the population activity after the perturbation only predicts the autocorrelation time if the networks are continuously stimulated with background noise at a level used for their homeostatic adaption. Otherwise, the effect of perturbations quickly decays. Last, we show how the time scales elicited by bistable dynamics can be harnessed for fast inference and the retrieval of memory respectively. In that process, we furthermore depict a strategy to exploit bistable dynamics for classification of spatio-temporal pattern.

6.1 Introduction

The initialization of any neural network poses a central challenge as it is crucial for convergence and hence optimal performance on a given task (Yam & Chow, 2000; Drago & Ridella, 1992; Shimodaira, 1994; Wessels & Barnard, 1992; Weymaere & Martens, 1994). Different initialization strategies for feed-forward architectures have been proposed in the past (Yam & Chow, 2000; Thimm & Fiesler, 1995; Goodfellow et al., 2016a; Weymaere & Martens, 1994). The design of recurrent topologies, in contrast, could be guided by the concept of neural criticality (Boedecker et al., 2012; Bertschinger & Natschläger, 2004; Legenstein & Maass, 2007; Kinouchi & Copelli, 2006; Shew & Plenz, 2013; Del Papa et al., 2017; Langton, 1990). Here, criticality is defined as a specific type of behavior observed when a system undergoes a phase transition. In statistical physics, different phases are classified by macroscopic, measurable properties, so-called *order parameters*. Depending on their change in response to the variation of an ambient property – the *control parameter* – different phases can be distinguished (Binney et al., 1992). For a discontinuous phase transition, the order parameter undergoes a discontinuity when the control parameter is posed at a critical value (Figure 6.1a). Here, neighboring phases coexist which is tantamount to bistability in extended, noisy systems (di Santo et al., 2016). A continuous transition, in contrast, leads to a continuous, but non-differentiable dependence (Figure 6.1b). If a system features a continuous phase transition, it can exactly reside at the transition between both phases, the so-called a *critical state*.

When undergoing a continuous phase transition between order and chaos or stability and instability – at a so-called critical point – systems develop a set of desirable computational properties: Among others, the sensitivity, the dynamic range, the correlation length, the information transfer and the susceptibility all diverge (Harris, 2002; Munoz, 2018a; Wilting et al., 2018; Barnett et al., 2013; Tkačik et al., 2015). Due to these maximized capabilities, neural networks with critical dynamics are thought to be optimal for any task processing (Del Papa et al., 2017; Bertschinger & Natschläger, 2004; Boedecker et al., 2012; Kinouchi & Copelli, 2006; Shew & Plenz, 2013; Munoz, 2018a,b; Langton, 1990). Despite being widely accepted within the community, it may be questioned whether criticality is actually always beneficial for computation. Wilting et al. (2018) hypothesized that each task requires its own network state to reach optimal performance since e. g. long memory retrieval may be detrimental for problems requiring only short memory time spans. Nevertheless, the interplay of task requirements and dynamical state is not understood.

Despite the favorable role of criticality for information processing, the tuning of a neural network precisely to a phase transition in general and to criticality, in particular, is often challenging. Due to the intrinsic nature of a critical point, the adjustment has to be very precise in order to not rush into the unstable regime. It is noteworthy that the archetype of SNNs, the human brain, constitutes a large, but finite system. For such systems, phase transitions do not occur at a single value of the control parameter, but are smeared out over a small region that is technically not critical, but sustains many properties of criticality (Moretti & Muñoz, 2013). Nevertheless, control mechanisms are required to maintain this dynamical state which is why a system ideally self-organizes in an unsupervised manner and dynamically readapts its dynamics (Hesse & Gross, 2014). Recent work demonstrated that local learning rules are indeed able to tune neural networks to criticality by modifying synaptic weights only based on pre- and postsynaptic neuronal activity (Levina et al., 2007; Meisel & Gross, 2009; Stepp et al., 2015; de Andrade Costa et al., 2015; Del Papa et al., 2017; Tetzlaff et al., 2010; Munoz, 2018a; Zierenberg et al., 2018; Poil et al., 2012; Shin & Kim, 2006). Furthermore, theoretical considerations suggest that local learning even allows to sweep the entire range of collective dynamics from the subcritical to the critical regime. In more detail, the state of homeostatically regulated networks can be tuned by simply adjusting the input strength (Zierenberg et al., 2018).

Locality plays a central role in biological as well as artificial neural networks where global information like task performance error or the state of distant neurons may be unavailable. Unsupervised and local learning does not only allow to tune the collective dynamics, but also enables neural networks to *learn* patterns or sequences (Hebb, 1949; Hopfield, 1982; Bi & Poo, 1998; Markram et al., 1997). To that end, SNNs are often exposed to STDP (Section 2.1.4) which is though to be central for sequence learning involved in language as well as motor processing (Markram et al., 1997; Bi & Poo, 1998). By these unsupervised mechanisms, the networks could be preshaped in the presence of noise to strongly speed up convergence on actual pattern input (Loidolt et al., 2020). This process is in direct analogy to the development of biological neural networks by spontaneous activity during development.

The locality of learning in combination with the intrinsic parallelism of SNNs does not naturally translate to efficient software-based simulations (Section 2.2). Specifically, simulations of plastic recurrent neural networks of increasing size become inefficient on conventional hardware, since the amount of state variables that needs to be updated by numerical methods often scales super linear with the number of neurons. Further, the investigation of self-organizing networks requires comprehensive sweeps to cover the relevant time spans: First, the branching of activity within the network needs to be captured which acts on time scales of hundreds of milliseconds. In addition,

the plasticity required for self-organization evolves on the order of seconds. Last, large amounts of data have to be acquired for statistical analysis. These time scales stay in strong contrast to the neuronal dynamics: the microscopic states evolve on scales ranging from tens of milliseconds to milliseconds. Simulating these accurately is especially important for recurrent networks. Capturing this discrepancy of time scales is prohibitively expensive in software-based simulations. However, the capabilities of recurrent SNNs can be efficiently exploited by relying on the intrinsic parallelism of the physical emulation of synapses and neurons in analog or mixed-signal neuromorphic systems (Mead, 1990; Douglas et al., 1995; Schemmel et al., 2010; Indiveri et al., 2011; Moradi et al., 2018). More recently, these devices were extended for on-device learning capabilities, allowing to fully profit from local learning rules (Friedmann et al., 2016; Qiao et al., 2015). Especially for accelerated devices, the aforementioned time scales can be bridged while the learning capabilities provide the required flexibility for the self-organization of the network.

The evaluation of the performance of an SNN implementation poses challenges irrespective of the underlying substrate. In Chapter 4, we discussed the impact and benefits of classification benchmarks. Despite their favorable role, the performance of a given network crucially depends on the specific task. The quantification of processing capabilities of any local circuit in a more general task-independent way can be achieved by methods from information theory (Wibral et al., 2015). Measures from classical information theory allow us to *quantify* the information about past input, the transfer of information within a circuit as well as the storage of information (Shannon, 1948; Wibral et al., 2015; Cover & Thomas, 2012). The latter can be estimated either within the network or as read out from a single unit. More recent work focuses on partial information decomposition (PID) to disentangle the encoding of information within ensembles of neurons in a network (Williams & Beer, 2010; Bertschinger et al., 2013; Wibral et al., 2015; Lizier et al., 2018). By this, PID allows to quantify the unique and redundant contribution of a set of source variables to a target, but most notably it also quantifies the synergistic computation which is a key contributor for any information integration (Wibral et al., 2015, 2017a,b; Bertschinger et al., 2013; Williams & Beer, 2010). The collection of these information-theoretic measures promises to bridge the gap between the assessment of local processing capabilities and global task performance.

In the following, we present strategies to precisely tune the collective dynamics of SNNs. All of these approaches have in common that the dynamics can be controlled by adjusting the input strength under the permanent application of local plasticity. Within the first part of this chapter, we investigate critical-like phenomena and shed more light on the relation between criticality, task performance and information-theoretic fingerprint. In the second part, we consider SNNs exhibiting bistable dynamics which give rise to autocorrelation times significantly exceeding single-neuron dynamics.

6.2 Methods

The general results presented within this chapter are inspired by the theoretical considerations of Zierenberg et al. (2018). In this work, the framework of a driven branching process (Harris & Harris, 2013) is applied to demonstrate the control of the dynamical regime of a homeostatically regulated network by the frequency of the driving stimulus. Within this formulation, a spike at timestep t causes on average m postsynaptic spikes in the next time step, such that the expected total network activity $A[t+1]$ is given by:

$$\langle A[t+1]|A[t]\rangle = mA[t] + Nh\delta t\,, \tag{6.1}$$

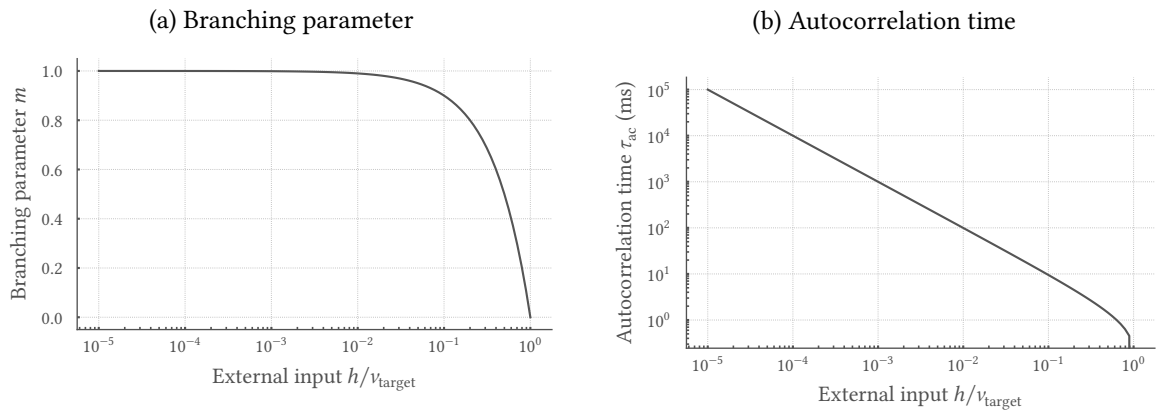


Figure 6.2: Control of collective dynamics by the input strength. Within the framework of a driven branching process, homeostatic plasticity shapes the network dynamics controlled by the ratio of the external input rate h and the target neural firing rate ν_{target}. Depending on this ratio, the network state can be bursting, fluctuating or input-driven quantified by the branching parameter m (**a**) and the autocorrelation time τ_{ac} (**b**).

in a network with N units which are externally driven at a rate h and considered at a fixed step size δt. Here, $\langle .|. \rangle$ denotes the conditional expectation. The parameter m is referred to as the branching parameter. For $m < 1$, the process is within the sub-critical regime where cascades of activity fade out over time and the temporal average of the network activity converges to the stationary value:

$$\langle A \rangle = \frac{1}{T} \sum_{t=0}^{T} A[t] \xrightarrow{T \to \infty} \frac{Nh\delta t}{1-m} . \tag{6.2}$$

Assuming a homogeneous rate distribution between all neurons in the network implies a mean neuronal firing ν rate of:

$$\nu = \frac{h}{1-m} , \tag{6.3}$$

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

Homeostatic plasticity within the network is designed to adjust neuronal firing rates to a target ν_{target} by regulating all presynaptic weights w_{ij} of a neuron i depending on its local firing activity $a_i[t]$. The associated weight dynamics evolve at a time scale τ_{hom} according to:

$$\Delta w_{ij}[t] = \left(\nu_{target} \delta t - a_j[t] \right) \left(\frac{\delta t}{\tau_{hom}} \right) . \tag{6.4}$$

For $\delta t / \tau_{hom} \to 0$ and therefore sufficiently slow weight dynamics, the updates become small $\Delta w_{ij} \approx 0$ and the network evolution is only determined by the average weights. Under this assumption, an effective branching parameter can be estimated based on Equation (6.3):

$$m = \frac{1-h}{\nu_{target}} . \tag{6.5}$$

Thus, the network dynamics can be controlled by the input strength h (Figure 6.2a). For decreasing h, the dynamics are characterized by a larger branching parameter closer to criticality. Here, the network compensates for disappearing input by internal activation, thereby tuning the system closer to a critical state. For even lower input strengths, the dynamics are limited by the homeostatic time scale which causes resonance effects. As a result, the network activity becomes increasingly bursty with an effective branching parameter resembling supercritical behavior.

In experimental settings, the dynamical state of a network can be quantified by the autocorrelation time τ_{ac} of the underlying activity. For a subcritical branching process, the autocorrelation function is governed by:

$$C(t) = m^t\,, \tag{6.6}$$

(Wilting & Priesemann, 2018). Comparison of the above expression with an exponential:

$$C(t) = \exp\left(-t\delta t/\tau_{ac}\right), \tag{6.7}$$

yields the autocorrelation time:

$$\tau_{ac} = -\frac{\delta t}{\log\left(m\right)}\,, \tag{6.8}$$

which diverges as expected at $m = 1$ where the system undergoes a critical phase transition (Figure 6.2b).

Within this chapter, we apply the theoretical concepts highlighted above to SNNs emulated on two different BrainScaleS-2 application-specific integrated circuits (ASICs). The required methods are organized in two major parts: We start with a description of the implemented network models (Section 6.2.1) and then move on to a summary of the analysis techniques (Section 6.2.2). Within the first part, we present two distinct frameworks which are both motivated by the work of Zierenberg et al. (2018): First, a STDP-based implementation on the prototype system High Input Count Analog Neural Network with Digital Learning System (HICANN-DLS) (cf. Section 3.1) is introduced where the input strength is controlled by the degree of the input. The second part highlights a realization on High Input Count Analog Neural Network X (HICANN-X) (cf. Section 3.2) that is closer aligned to the one presented by Zierenberg et al. (2018). Here, the network is controlled by homeostatic regulation based on neuronal firing rates while the input strength is adjusted via the average indegree of the network.

6.2.1 Models

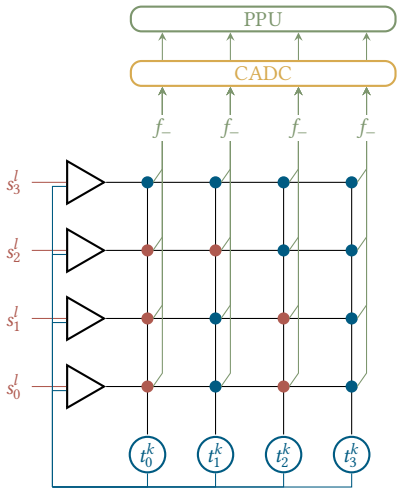
> All measures of time presented within this thesis are given in wall clock time. Due to the accelerated nature of the BrainScaleS-2 architecture, the latter are rendered 1000× faster than their biological equivalents. Hence, all time scales as well as time constants have to be scaled by a factor of 1000 to achieve a translation to the corresponding biological time domain, i. e. 1 µs wall clock time corresponds to 1 ms in the biological time domain.

The results presented in this chapter were acquired on the two different versions of the mixed-signal neuromorphic hardware system BrainScaleS-2 described in Chapter 3. While the STDP framework described in Section 6.2.1.1 was implemented on the HICANN-DLS prototype chip, the homeostatic framework depicted in Section 6.2.1.2 targeted the HICANN-X. For each implementation, a brief overview of the network topology, the programmed plasticity rule, the initialization and the simulation techniques is given.

6.2.1.1 STDP framework

Network: For the experiments executed on HICANN-DLS, we used all $N = 32$ leaky integrate-and-fire (LIF) neurons and all corresponding 32×32 synapses. The synaptic address labels were

(a) Network implementation



(b) Plasticity kernel

```
1  ; Scale anticausal measurements by factor
2  mulbfs dws, ac_meas, %[lambdas_stdp]
3  ; Apply decays to weight updates
4  mulbfs dws_decay, weights, %[lambdas_drift]
5  addbfs dws, dws, dws_decay
6  ; Load random numbers from general-purpose part
7  lax dws_rand, 0, %[rands_offset]
8  ; Add random numbers to updates
9  addbfs dws, dws, dws_rand
10 ; Scale and add update to weights
11 mulbfs dws, dws, %[scales]
12 addbfs weights, weights, dws
```

Figure 6.3: Implementation of homeostatically regulated fixed indegree SNNs on HICANN-DLS. **(a)** The synapses in the synapse array of HICANN-DLS were programmed to two different address labels. A fixed number of synapses per neuron (column) was set to the external address (red) and therefore transmits only input spikes s^l_j. The recurrent events t^k_i of neuron i were injected back into the chip to synapse drivers i (row) with a distinct address label (blue). Non-external synapses were configured to share the recurrent address and therefore only transmit recurrent events. The analog STDP measurements of every synapse were digitized by the CADC and used for the parallel calculation of weight updates on the PPU. **(b)** Kernel code for the STDP-based updates of synaptic weights in NASM syntax. By row-wise parallel iteration over the synapse array, all synaptic weights were updated equally. The basic observable for the STDP driven update rule consisted of the measurements of the correlation sensors which were digitized by the column-parallel analog-to-digital converter (CADC). A read was directly triggered from the single instruction, multiple data (SIMD) vector unit. The random numbers required for the stochastic potentiation of synaptic weights were generated in software on the general-purpose part and were then loaded into a register of the vector unit.

configured to result in fixed-indegree networks (Figure 6.3a). In more detail, we used two different address labels in the synapse array: one tagging external and the other one recurrent spikes. The LIF neurons were connected to the external input by configuring K_{ext} randomly chosen synapses per neuron – i. e. per column of synapse – to the external address. All address labels of the remaining synapses were set to the recurrent address. Therefore, each synapse either transmitted external or recurrent events. The spike router was used in the bypass mode (cf. Section 3.1.1) and programmed to send all accruing spikes of a neuron i back to the synapse driver i – i. e. to row i – with the recurrent address label. By controlling the ratio of the two address labels programmed in the synapse array and therefore by setting K_{ext}, the network can be adjusted between an input-driven and a highly recurrent dominated regime. The extreme case with $K_{\text{ext}}/N = 1$ represents a purely feed-forward topology, whereas $K_{\text{ext}}/N = 0$ corresponds to a fully recurrent network which is completely decoupled from the external input.

To approximate biological findings, our networks featured about 20 % inhibitory synapses. Specifically, we configured N_{inh} randomly chosen synapse drivers on HICANN-DLS to be inhibitory, i. e. N_{inh} synaptic rows only contained inhibitory synapses (cf. Figure 6.3). It is noteworthy that the stochastic network description led to both excitatory as well as inhibitory synapses transmitting the external input.

Table 6.1: Overview of the model parameters used within the STDP framework. All time constants are given in wall clock time. Spike-timing dependent plasticity (STDP) amplitudes as well as time constants where measured using 20 spike pairs. The errors indicate the standard deviation. Table and caption adapted from Cramer et al. (2020a).

Stage	Parameter	Symbol	Value
Neuro-synaptic dynamics	Threshold potential	u_{thresh}	(554 ± 21) mV
	Leak potential	u_{leak}	(384 ± 79) mV
	Reset potential	u_{reset}	(319 ± 18) mV
	Membrane capacitance	C_m	(2.38 ± 0.02) pF
	Membrane time constant	τ_{mem}	(1.6 ± 1.0) µs
	Refractory period	τ_{ref}	(4.9 ± 0.5) µs
	Excitatory synaptic time constant	$\tau_{\text{syn}}^{\text{exc}}$	(3.7 ± 0.5) µs
	Inhibitory synaptic time constant	$\tau_{\text{syn}}^{\text{inh}}$	(2.8 ± 0.3) µs
	Synaptic delay	d_{syn}	(1.9 ± 0.1) µs
	Weight scaling factor	γ_{w}	(8.96 ± 0.13) µA
Network	Inhibitory synapses per neuron	N_{inh}	6
	Neurons	N	32
	Degree of the input	K_{ext}	6 - 32
	Number of Poisson spike sources	N_{in}	32
	Initial weight	w_{ij}^{init}	0 mA
Plasticity dynamics	STDP time constant	τ_{-}	(6.8 ± 1.2) µs
	STDP amplitude	η_{-}	0.071 ± 0.023
	Correlation scaling factor	λ_{stdp}	11/128
	Drift parameter	λ_{drift}	1/512
	Range of random variable	n_{amp}	15/16
	Bias of random variable	$\langle n \rangle$	3/16
	Plasticity update period	T	1 ms
Input	Input rate	h	29 kHz
Experiment control	Burn-in experiment duratio	T^{burnin}	625 ms
	Static experiment duration	T^{exp}	104 ms
	Static trial experiment duration	T^{static}	1 ms
	Training experiment duration	T^{train}	104 ms
	Testing experiment duration	T^{test}	21 ms
	Perturbation experiment duration	T^{pert}	2 ms
	Perturbation time	t_{pert}	1 ms
Evaluation	Embedding dimension	l	4
	Delays steps	N_{τ}	100

The parameter fluctuations induced by the analog emulation of neuro-synaptic dynamics were not explicitly compensated by dedicated calibration routines. Especially, no explicit calibration based on single neurons and synapses was applied. Instead, all circuit instances were configured to the same shared values. The latter were chosen such that all parts behave according to the LIF equation with current-based synapses. In particular, we ensured that all neurons are sensible to stimulation, but silent in the absence thereof. This choice led to larger uncertainties in the model parameters as reported in Table 6.1, though excluded the necessity for detailed calibration routines.

Plasticity: All synapses – the recurrent as well as the stimulating ones – were subject to plasticity. Specifically, the synaptic weight updates comprised three contributions: First, a correlation sensitive part controlled by λ_{stdp}, second a weight drift controlled by the parameter λ_{drift} and third positively biased stochastic potentiation. In more detail, the plasticity processing unit (PPU) on the HICANN-

DLS chip was programmed to update synaptic weights to $w_{ij}(t+T) = w_{ij}(t) + \Delta w_{ij}$ according to:

$$\Delta w_{ij} = \underbrace{-\lambda_{\text{stdp}} f_-\left(t_i^k, t_j^l, t, t+T\right)}_{\text{specific depression}} \underbrace{-\lambda_{\text{drift}} w_{ij}(t)}_{\text{decay}} + \underbrace{n_{ij}(t)}_{\substack{\text{unspecific}\\ \text{potentiation}}} \quad . \tag{6.9}$$

While the specific depression within this update rule is comparable to vanilla STDP, the potentiation is unspecific and hence not correlation driven. It is noteworthy that despite of our choice for the anticausal accumulation trace f_- (cf. Equation (3.2)) within Equation (6.9), f_+ is an equally valid option which leads to comparable results. However, we decided to rely on f_- for weight update calculations due to its similarity to STDP with anticausal depression. Our weight updates were rendered stochastic by the last term in Equation (6.9), which adds a uniformly distributed, but biased random variable:

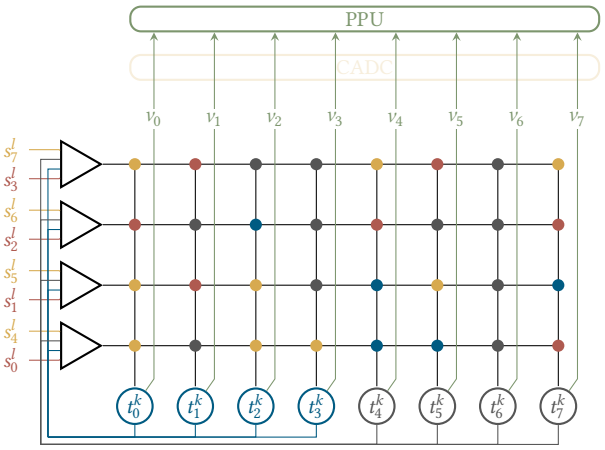
$$n_{ij} \sim \text{unif}\left(-n_{\text{amp}}, n_{\text{amp}}\right) + \langle n \rangle \,, \tag{6.10}$$

where n_{amp} specifies the range, while $\langle n \rangle$ is the bias of the random numbers. The parameters λ_{drift}, n_{amp} and $\langle n \rangle$ were chosen such that the average combined force of the decay and the stochastic potentiation was positive (Table 6.1).

To facilitate scalability, all individual terms were calculated and applied to the synaptic weights by the SIMD vector unit (Figure 6.3b). Here, the general-purpose part of the PPU was only used to implement the update loop over the whole synapse array in row-major order. For this purpose, the code shown in Listing 1 was utilized to trigger weight updates and in that process load synaptic weights and access the digitized anticausal correlation measurements via the CADC. This code was extended by the kernel shown in Figure 6.3b, implementing the individual terms of Equation (6.9). To that end, the digitized correlation measurements `ac_meas` were first multiplied by the constant scaling factor `lambda_stdp` and the result thereof was stored in `dws` (line 2). Second, the weight decay was implemented by a single multiplication of the weights in the register `weights` with the scaling factor `lambda_drift`. The result in `dws_decay` was in turn added to the weight update in `dws` (line 4 and 5). Third, the realization of the stochastic, unspecific potentiation required the availability of random numbers on the SIMD vector unit. As the prototype ASIC does not feature support for the on-chip generation of random numbers, a pseudo-random number generator was implemented by utilizing an exclusive-OR (XOR)-shift algorithm evaluated on the general-purpose part of the PPU. The drawn numbers were loaded into the register `dws_rand` accordingly and then added to the weight updates `dws` (line 7 and 9). Last, the updates `dws` were scaled by the constant factor `scales` to ensure small updates and therefore smooth convergence despite of the 8 bit arithmetic (11 and 12). Afterwards, the calculated weights in `weights` were written to the synaptic static random-access memory (SRAM) with the code shown in Listing 1.

Initialization: The network emulation starts from a silent state by initializing the synaptic weights to $w_{ij} = 0$. Later, we will show that our approach generalizes to initial conditions with $w_{ij} \neq 0$ (Figure 6.17). We then used N Poisson spike trains of rate h for stimulation via the K_{ext} synapses of every neuron. In this adaptation phase, we repeatedly applied Equation (6.9) for a duration T^{burnin} leading to synaptic weights $w_{ij} \neq 0$. To gather sufficient statistics, we emulated each network with degree K_{ext} 100 times. For each of these runs, we considered different random seeds for the input spike trains, the network topology as well as the random walk in Equation (6.9). The resulting weight matrices served as the initial condition for static experiments of duration T^{exp} with frozen weights ($\Delta w_{ij} = 0$) on which we perform the actual analysis.

(a) Network implementation



(b) Plasticity kernel

```
; Calculate weight updates
lax rates, 0, %[rates_offset]
subbfs dws, targets, rates
mulbfs dws, dws, %[lambdas]
; Stochastic acceptance
inx rands, %[rand_base], %[offset]
subbfs rands, rands, rands_offset
compareb rands
select dws, dws, %[zeros], LT
; Only update recurrent synapses
inx addresses, %[decoder_base], %[offset]
shb addresses, addresses, 6
shb addresses, addresses, -6
subbfs addresses, addresses, %[ones]
compareb addresses
select dws, %[zeros], dws, LT
addbfs weights, weights, dws
```

Figure 6.4: Implementation of homeostatically regulated fixed indegree SNNs on HICANN-X. **(a)** The routing capabilities of HICANN-X were exploited to treat the analog cores of both chip halfs as a single virtual matrix. A synapse i, j was configured to either transmit recurrent spikes from neuron j or neuron $j + 256$ respectively (blue and gray circles) or to relay a stimulating spike train j or $j + 256$ (red and yellow circles) to its home neuron. Shown is a schematic illustration for a virtual matrix of size 4×8. Homeostatic regulation was implemented on the PPU by accessing the neuronal spike counters. **(b)** Kernel code for the homeostatic regulation of synaptic weights in NASM syntax. The basic observable for this update rule consists of the spike counts which were obtained by accessing the spike counters via the general-purpose part of the PPU. Following this, the results were loaded into the SIMD vector unit. The random number generators were used to implement a stochastic acceptance of weight updates.

Simulation: We implemented idealized versions of the network in *Brian 2* to augment the hardware emulation (Goodman & Brette, 2009). In more detail, we disregarded parameter mismatch and temporal noise and instead initialized all model parameters to the mean values given in Table 6.1. Furthermore, we evaluated weight dynamics with floating-point precision, hence relaxing the constraint of integer arithmetic and discretized weights as present on HICANN-DLS. Moreover, the implemented anticausal STDP in *Brian 2* did not only consider nearest-neighbor spike pairings, but was based on an integral formulation. To stay within the population-based formulation of Brian 2, the degree of the input was implemented probabilistically, i. e. each input-neuron pair was connected with probability K_{ext}/N and each pair of neurons with probability $(N - K_{\text{ext}})/N$. This population-based formulation relaxed the hardware constraints that either the recurrent synapse between neuron i and j or the stimulating synapse between input i and neuron j could be realized.

6.2.1.2 Homeostatic framework

Network: The topology implemented on the full-size HICANN-X chip was similar to the one implemented on the prototype except for a fixed recurrent sparsity and a scaled-up network. In

more detail, the routing capabilities of HICANN-X were exploited such that the synapse arrays of the four blocks can be treated as a single larger matrix of shape 256×512 (Figure 6.4a). Here, the left half of this virtual matrix was assigned to the upper two quadrants of HICANN-X and the lower half was mapped to the two lower ones. In order to still be able to connect arbitrary inputs to neurons as well pairs of neurons among each other, the lower 2 bit of the address labels a_{ij} were reserved. Specifically, based on their configured value, one of four possible source populations could be individually assigned to each synapse: A synapse i, j could be configured to either transmit recurrent spikes from neuron j for $(a_{ij} \,\&\, 0\text{b}11) = 0$ or neuron $j + 256$ for $(a_{ij} \,\&\, 0\text{b}11) = 1$ to neuron i determined by the two lowermost values. The remaining address label values were reserved to identify a stimulating spike train j for $(a_{ij} \,\&\, 0\text{b}11) = 2$ or $j + 256$ for $(a_{ij} \,\&\, 0\text{b}11) = 3$, respectively.

The aforementioned setup was used to again implement fixed-indegree networks. In this context, K_{in} synapses of each neuron were randomly drawn and configured to transmit one of 256 possible input spike trains. Here, we relied on a fixed number of randomly chosen recurrent connections K_{rec} which was equally split into the lower and upper half of neurons. Thereby, when changing K_{in}, the network itself remained unchanged in terms of recurrent connections and only the number of synapses relaying the stimulus was changed. All remaining synapses were configured to the second input population address which could be used to inject structured input. For that purpose, we randomly selected N_{div} synapses per row – i. e. per presynaptic partner – and configured their weights to a value w_{stim} (Figure 6.11a). By relying on an extra set of synapses to stimulate our SNNs, the effective stimulus strength remained unchanged even when changing K_{in}.

Here, too, the networks featured inhibitory synapses: A randomly drawn set of $N_{\text{inh}} = 51$ synaptic rows – i. e. presynaptic partners – was configured to be inhibitory corresponding to about 20 % of all available synapses. To stay within the framework of a single large synapse array, the IDs of the inhibitory synapse drivers on the upper and lower half of HICANN-X were chosen to coincide. Again, this formulation led to excitatory and inhibitory stimulating synapses which is in direct contrast to the theoretical considerations of Zierenberg et al. (2018).

In contrast to the STDP framework, we here applied an explicit and automated calibration of the analog circuits to equalize out parameter variations (Figure 6.5). To this end, binary search algorithms were utilized to match the measured values to a desired target (Weis, 2020). More specifically, the time constants were adjusted based on membrane analog-to-digital converter (MADC) measurements to sufficiently capture the temporal aspects of neuro-synaptic dynamics. The synaptic time constants of both excitatory and inhibitory synapses were calibrated to 5 µs, whereas the membrane time constants were configured to match a target of 20 µs (Figure 6.5a to c). The configured membrane time constant builds a compromise between noise susceptibility which increases with their magnitude on the neuromorphic substrate and the desirable integration of inputs by neuro-synaptic dynamics even for the low firing rates considered within this work. In contrast, we relied on the parallel CADC for rapid calibration of all model potentials. The latter were calibrated to exhibit a high dynamic range, i. e. a high distance between the leak and threshold potential (Figure 6.5d to e and Figure 6.6a and b).

Plasticity: The homeostatic regulation implemented on HICANN-X was designed to approximate the theoretical considerations of Zierenberg et al. (2018). Here, all stimulating synapses remained unchanged with a fixed shared weight w_{in}. In contrast, the afferent recurrent synaptic weights of each neuron were homeostatically regulated depending on the neuronal firing rate. Specifically,

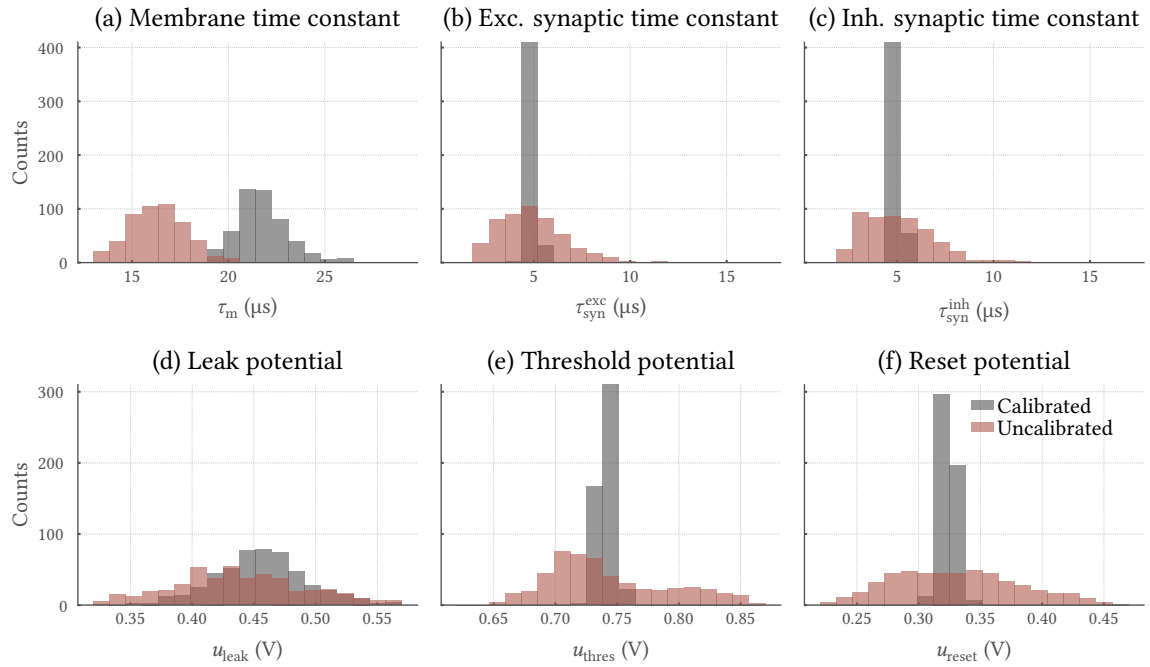


Figure 6.5: The configurability of HICANN-X allows to mitigate circuit-to-circuit variability. Dedicated calibration routines were used to compensate fix-pattern variations by exploiting the configurability of all neuromorphic circuits. The excitatory and inhibitory synaptic time constants τ_{syn}^{exc} (**a**) and τ_{syn}^{inh} (**b**) were calibrated to match a target value of 10 µs, whereas the membrane time constant τ_m (**c**) was adjusted to a target of 20 µs. To be less susceptible to temporal noise, the leak potentials u_{leak} (**d**) and the threshold potential u_{thres} were optimized for a large distance. The reset potential u_{reset} (**f**) was adjusted to a value slightly below u_{leak}. As a reference, the distribution of an uncalibrated state is shown which was obtained by configuring all circuits with the mean of the value determined by the calibration routine.

they were updated according to $w_{ij}(t + 2T) = w_{ij}(t) + \Delta w_{ij}$ with:

$$\Delta w_{ij} = \lambda \left(\nu_{target} - \nu_i \right) . \tag{6.11}$$

Here, λ denotes the learning rate, ν_i corresponds to the firing rate of neuron i in the time interval $[t, t+T)$ and ν_{target} is the target rate. The calculated weight updates Δw_{ij} were applied with a probability of only $p = 2.5\,\%$ to guarantee smooth convergence to the target rate despite integer arithmetic and finite weight resolution.

Scalability was again facilitated by updating synaptic weights via the SIMD vector units (Figure 6.4b). To this end, we again utilized the code shown in Listing 1 which was extended by the kernel highlighted in Figure 6.4b. As the homeostatic updates according to Equation (6.11) rely on ν_i, the spike counters were first evaluated on the general-purpose part of the PPUs. In general, this detour was necessary since a direct access of the spike counters is not possible on the considered version of HICANN-X. Specifically, we reset all spike counters in direct succession and triggered the associated reads after a total duration $T - T^{reset}$ from the general-purpose part, where T^{reset} denotes the duration required to reset the counters of all N neurons. To ensure a constant measurement period T for all neurons, the duration of the reset was adjusted to approximately match the time span of a read. By accessing the spike counter of the neuron backend in advance of the weight update calculations and by allowing a sufficient time interval for weight updates after triggering the next iteration of the update loop, the measurement of ν_i was not disturbed by dynamics directly elicited

Table 6.2: Overview of the model parameters used within the homeostatic framework. All time constants are given in wall clock time. The errors indicate the standard deviation.

Stage	Parameter	Symbol	Value
Neuro-synaptic dynamics	Threshold potential	u_{thresh}	(554 ± 21) mV
	Leak potential	u_{leak}	(384 ± 79) mV
	Reset potential	u_{reset}	(319 ± 18) mV
	Membrane capacitance	C_m	(2.38 ± 0.02) pF
	Membrane time constant	τ_{mem}	(1.6 ± 1.0) µs
	Refractory period	τ_{ref}	(4.9 ± 0.5) µs
	Excitatory synaptic time constant	$\tau_{\text{syn}}^{\text{exc}}$	(3.7 ± 0.5) µs
	Inhibitory synaptic time constant	$\tau_{\text{syn}}^{\text{inh}}$	(2.8 ± 0.3) µs
	Synaptic delay	d_{syn}	(1.0 ± 0.1) µs
	Exciatotry weight scaling factor	$\gamma_{\text{w}}^{\text{exc}}$	(0.55 ± 0.00) nA
	Inhibitory weight scaling factor	$\gamma_{\text{w}}^{\text{inh}}$	(0.65 ± 0.10) nA
	Exciatotry weight offset	$\delta_{\text{w}}^{\text{exc}}$	(0.59 ± 1.15) nA
	Inhibitory weight offset	$\delta_{\text{w}}^{\text{inh}}$	(0.80 ± 1.10) nA
Network	Inhibitory synapses per neuron	N_{inh}	51
	Recurrent synapses per neuron	N_{rec}	102
	Neurons	N	512
	Average indegree	K_{in}	65 to 117
	Number of Poisson spike sources	N_{in}	256
	Number of unique letter inputs	N_{stim}	10
	Number of input replications	N_{mu}	20
	Number of stimulating synapses per row	N_{div}	20
	Input weight	w_{in}	17
	Stimulating weights	w_{stim}	20 to 60
	Initial synaptic weight	w_{ij}^{init}	0
Plasticity dynamics	Learning rate	λ	0.375
	Target rate	ν_{target}	10 kHz
	Update probability	p	2.5 %
	Plasticity update period	T	1 µs
Input	Input rate	h	0 kHz to 10 kHz
	Stimulus rate	Δh	0 kHz to 30 kHz
	Number of samples	N_{sample}	500
Experiment control	Burn-in experiment duration	T^{burnin}	1000 µs
	Static experiment duration	T^{exp}	100 µs
	Letter duration	T^{letter}	200 µs
	Pattern duration	T^{pattern}	2000 µs

by rewriting the weights. In particular, the neuromorphic SNN had time to settle after applying the weight updates and prior to the next estimation of ν_i for the subsequent synaptic update. The rates measured by this approach were subsequently loaded into the register `rates` on the SIMD vector unit (line 2). In direct succession, the content of `rates` was subtracted from the target rates in `targets` and scaled by the constant factor `lambda`. The result thereof was stored in `dws` (lines 3 and 4).

We employed the hardware accelerators for the drawing of random numbers to implement the stochastic acceptance of weight updates. By this, uniformly distributed 8 bit integers were directly and most notably in parallel loaded into the register `rands` of the SIMD vector units (line 6). To finally support different acceptance probabilities, a configurable offset `rands_offset` was subtracted from the drawn values (line 7). By an elementwise comparison, the elements in `dws` were clamped to zero whenever the corresponding entry in `rands` was smaller than or equal to zero (lines 8 and 9).

To ensure that only the weights of recurrent synapses were updated, the address labels were first loaded into the register `addresses` of the SIMD vector unit to finally detect recurrent synapses (line 11). Subsequently, two bit shift operations were utilized to extract the two lowermost bits of the address labels which contain the information of the source population as described previously (lines 12 and 13). Specifically, we detected the synapses for which $(a_{ij} \,\&\, 0\text{b}11) = 0$ or $(a_{ij} \,\&\, 0\text{b}11) = 1$ holds true. This was implemented by subtracting one from the shifted addresses in `addresses` (line 14). Now, a single check for positivity was sufficient to detect recurrent synapses: For stimulating synapses, the result in `addresses` was positive and the updates in `dws` were clamped to zero, otherwise, the calculated weight updates were kept (lines 15 and 16). Lastly, the calculated updates `dws` were added to the old weights contained in `weights` and in turn written back to the synaptic SRAM with the code in Listing 1.

Initialization: At the start of each emulation, we initialized all recurrent weights w_{ij} to 0. In contrast, the stimulating weights were configured equally to w_{in}. During emulation, the network was stimulated with N_{in} Poisson-distributed spike trains of rate h by the K_{in} synapses of every neuron. For that purpose, we configured the on-chip background spike sources to inject spikes with address labels corresponding to the first input population into the emulated SNNs. The repeated application of Equation (6.9) for a total duration T^{burnin} caused the formation of recurrent weights $w_{ij} \neq 0$. For every value K_{in}, the network was evaluated 50 times, each with a different random seed for the input spike trains, the network topology and the stochastic acceptance of the weight updates. If not stated otherwise, the resulting weight matrices were used as initial conditions for experiments with frozen weights ($\Delta w_{ij} = 0$) of duration of T^{exp} on which the analysis is performed.

Simulation: For validation purposes, comparable homeostatically regulated SNNs were implemented in *Brian 2* (Goodman & Brette, 2009). Here, the characterization results shown in Figure 6.5 were used to closely align software simulation and hardware emulation. To that end, fixed-pattern noise was implemented by drawing parameter values from Gaussian distributions, parametrized by the previously identified means as well as standard deviations. Furthermore, temporal noise with a standard deviation of $2\,\text{mV} \cdot \sqrt{w\Delta t/\tau_{\text{mem}}}$ was added to the membrane potential of each neuron. This roughly matched the spread of temporal variations present on the neuromorphic substrate. To finally align the effect of a single post-synaptic potential (PSP) on hard- and in software, we characterized the PSP height as a function of the configured weight value on HICANN-X (Figure 6.6). In more detail, we determined a weight scaling factor $\gamma_{\text{w}}^{\text{exc,inh}}$ to transform the dimensionless hardware weight values w_{ij}. This was achieved by fitting the ideal solution of the LIF equation (Equation (2.12)) to MADC recordings of the membrane potential $u_i(t)$ in response to a single stimulating event t_j^0 relayed over a single synapse with weight w_{ij}:

$$u_i(t) = u_{\text{leak}} + \frac{\tau_{\text{mem}} \cdot \tau_{\text{syn}} \cdot \tilde{w}_{ij}}{C_{\text{mem}}(\tau_{\text{syn}} - \tau_{\text{mem}})} \Theta\left(t - t_j^0\right) \left[\exp\left(-\frac{t - t_j^0}{\tau_{\text{syn}}} \right) - \exp\left(-\frac{t - t_j^0}{\tau_{\text{mem}}} \right) \right]. \tag{6.12}$$

Here, we fixed all fit parameters to the calibration target values except for the PSC height $\tilde{w}_{ij}$ and the leak potential u_{leak} to ensure stable fits. A linear fit to the obtained $\tilde{w}_{ij}$ for every configured synaptic weight w_{ij} resulted in an estimate for $\gamma_{\text{w}}^{\text{exc,inh}}$ which finally allowed us to implement a comparable weight range in our Brian 2 simulations. Furthermore, the fit provided the offsets $\delta_{\text{w}}^{\text{exc,inh}}$ which systematically deviated from zero and hence led to a violation of Dale's law for low w_{ij} (Figure 6.6c to e). Their effect on the results presented within the scope of this chapter was investigated in

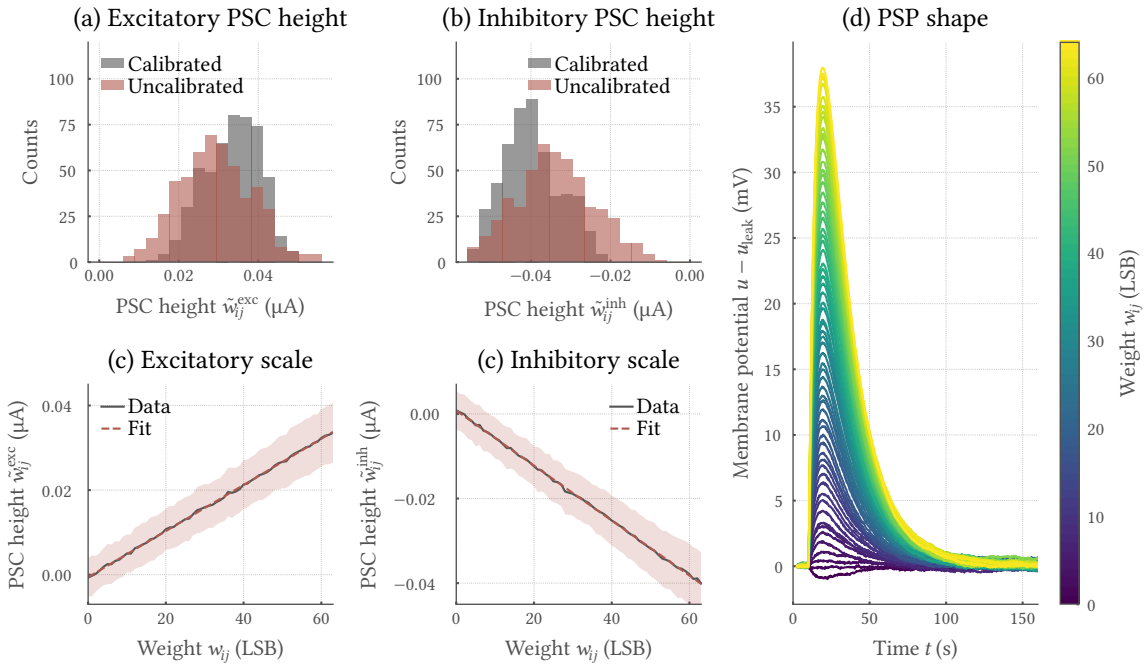


Figure 6.6: Characterization of the weight range on the HICANN-X chip. (a) The measured excitatory PSC heights $\tilde{w}_{ij}^{\text{exc}}$ show significant parameter deviations for a maximum weight. The uncalibrated state was achieved by configuring all circuit instances with the mean value determined by the calibration routine. **(b)** Same as for (a), but showing a histogram of inhibitory post-synaptic current (PSC) heights $\tilde{w}_{ij}^{\text{inh}}$. **(c)** The average excitatory PSC height $\tilde{w}_{ij}^{\text{exc}}$ is a linear function of the configured synaptic weight. Linear fits were utilized to extract the weight scaling factor essential to match the software simulations to the hardware emulation. **(d)** Same as for (c), but showing the average excitatory PSC height $\tilde{w}_{ij}^{\text{inh}}$. **(e)** Exemplary MADC recordings of an excitatory PSPs for various weight values.

simulations, but turned out to not affect the investigated collective dynamics. Because of this, the simulation results shown within this chapter were obtained with $\delta_{\text{w}}^{\text{exc,inh}} = 0$ for for simplicity.

Similar to the simulation methods of the STDP framework, the network topology was defined in a population-based manner. Because of this, the same relaxations of hardware constraints hold true for the simulations of the homeostatically regulated SNNs. In contrast to the simulations of the STDP framework, the standard time step of Brian 2 was reduced. More specifically, we relied on $\Delta t = 50\,\mu\text{s}$ approaching the time-continuous nature of the analog implementation on BrainScaleS-2 which might become more critical for the larger networks under consideration.

6.2.2 Evaluation

The networks described above were not only characterized in terms of critical dynamics, but their performance was also evaluated in a task-independent as well as task-dependent fashion. As all measures rely on an estimate of activity, we start with a description of the applied binning (Section 6.2.2.1). We proceed with the measures characterizing the distance to critical-like dynamics in Section 6.2.2.2. The task-independent measures are depicted in Section 6.2.2.3. We close the section with a description of the task-dependent performance quantification in the context of reservoir computing in Section 6.2.2.4.

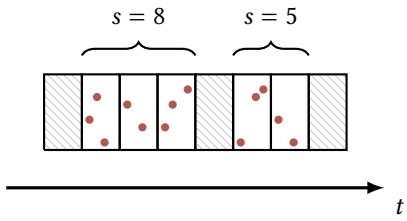


Figure 6.7: Schematic illustration of neural avalanches. Avalanches correspond to cascades of spikes in neural networks, usually quantified on a measure of population activity. The size s of a neural avalanche corresponds to the number of events (red dots) in a cascade of spikes.

6.2.2.1 Binning

All of the following measures rely on an estimate of the temporally resolved activity, either per neuron or population-wise. Because of this, we applied temporal binning to the spike times x_i^k of unit i:

$$\tilde{x}_i[t] = \sum_k \mathbb{1}\left(x_i^k \geq t \cdot \delta t,\, x_i^k < (t+1) \cdot \delta t\right), \tag{6.13}$$

with the binwidth δt, and the indicator function $\mathbb{1}$. Based on this estimate, we are able to define the binarized activity for a single process i:

$$x_i[t] = \min\left(1, \tilde{x}_i[t]\right). \tag{6.14}$$

Depending on the leading question, the variable $x_i[t]$ can represent either the activity of a neuron in the network $a_i[t]$, or of a stimulating spike train $s_i[t]$. Hence, we abbreviated the stimulating (input) and recurrent spikes correspondingly by the variable x_i^k.

The population activity $a[t]$ of the network can be obtained by summing $\tilde{a}_i[t]$ over all neurons:

$$a[t] = \sum_{i=1}^{N} \tilde{a}_i[t]. \tag{6.15}$$

From this, the population rate $\nu[t]$ can be estimated by normalization with the binwidth δt as well as the population size N.

6.2.2.2 Classical measures

Neural avalanches: In neural networks, an avalanche is defined as a cascade of spikes in direct succession. Following established definitions, we estimate avalanches based on the population activity $a[t]$ obtained by binning the spike trains with δt given by the mean inter-event interval. Hence, avalanches correspond to sequences of non-empty consecutive bins of $a[t]$ (Beggs & Plenz, 2003). Here, we only consider the sizes s of avalanches which are in general considered as the number of spikes within an avalanche (Figure 6.7). The size distribution $\mathscr{P}(s)$ is assumed to resemble a power-law in the case of critical dynamics.

The criticality hypothesis can hence be quantified by fitting different distributions to $\mathscr{P}(s)$. In more detail, we compare power-law with exponential fits and select a best-matching distribution based on the fit-likelihood (Clauset et al., 2009). To account for the system's finite size, we restrict the fit range to the interval $s \in \{4, 3 \cdot N\}$. Furthermore, we make use of a truncated power-law fit:

$$\mathscr{P}_{\mathrm{pl}}(s) \propto s^{-\alpha_s} \exp\left(-\frac{s}{s_{\mathrm{cut}}}\right), \tag{6.16}$$

with $s \geq 1$ to finally estimate the critical exponent α_s as well as an exponential cutoff s_{cut}. All fits were performed with the *power-law* Python package which provides a rich repertoire of testable distributions (Alstott et al., 2014).

Fano factor: For the quantification of variations within the population activity $a[t]$, we consider the Fano factor:

$$F = \sigma_a^2 / \mu_a \,, \tag{6.17}$$

where σ_a^2 denotes the variance and μ_a is the mean of the population activity $a[t]$. By estimating $a[t]$ with a binwidth of $\delta t = \tau_{\text{ref}}$, we are able to even capture variations within the propagation.

Trial-to-trial variability: For the estimation of trial-to-trial variations Δ_{vrd}, we stimulated each network multiple times with the very same input spike train. To this end, we obtain multiple trials which differ due to the analog emulation of neuro-synaptic dynamics. In order to finally estimate a von-Rossum like distance, we convolve the spike times $t_{i,m}^j$ emitted by neuron i in trial m with a Gaussian kernel:

$$\tilde{t}_{i,m}(t) = \sum_j \int_0^{T^{\text{exp}}} \exp\left(-\frac{(t-t')^2}{2\sigma_{\text{vrd}}^2}\right) \delta(t' - t_{i,m}^j) \mathrm{d}t' \,, \tag{6.18}$$

with the kernel width σ_{vrd} which is chosen to match the refractory period τ_{ref}. The convolution is carried out with a temporal resolution of 0.1 µs. In pair-wise comparisons, we, in turn, estimate a von-Rossum like distance from the trials m and n:

$$\Delta_{\text{vrd}} = \frac{1}{\sigma_{\text{vrd}}} \sum_{\substack{m,n \\ m \neq n}} \sum_{i=1}^{N} \int_{-\infty}^{\infty} \frac{[\tilde{t}_{i,m}(t) - \tilde{t}_{i,n}(t)]^2}{[\tilde{t}_{i,m}(t) + \tilde{t}_{i,n}(t)]^2} \mathrm{d}t \,, \tag{6.19}$$

which is then averaged over all possible combinations of trials.

Susceptibility: We define a susceptibility based on the network's sensitivity to external perturbations. To this end, we embedded a burst of N_{pert} additional spikes at time t_{pert} into the regular Poisson distributed input spike trains. The susceptibility is then defined as the difference in the populations activity $a[t]$ after and prior to this perturbation:

$$\chi = \frac{\sum_{i,k} \mathbb{1}\left(x_i^k \geq t_{\text{pert}} - \delta t,\, x_i^k < t_{\text{pert}}\right) - \sum_k \mathbb{1}\left(x_i^k \geq t_{\text{pert}},\, x_i^k < t_{\text{pert}} + \delta t\right)}{K_{\text{ext}}^2} \,, \tag{6.20}$$

where we estimate $a[t]$ with a binwidth of $\delta t = d_{\text{syn}}$, i. e. the synaptic round trip time. This allows us to only capture the effect of the perturbation and at the same time to minimize the effect of trial-to-trial variations. Here, we normalize χ to the number of stimulating synapses K_{ext}^2 in order to compensate for the decoupling from the external input with decreasing K_{ext}.

To calculate Δ_{vrd}, d_{var} and χ, each weight matrix, obtained by the application of the plasticity rule, was used as initial condition for 10 emulations with frozen weights and fixed seeds for the Poisson-distributed spike trains of duration T^{static}, T^{pattern} and T^{pert}, respectively. Additionally, a perturbation of size N_{pert} at $t_{\text{pert}} = T^{\text{pert}}/2$ was embedded into the stimulating spike trains for the estimation of χ.

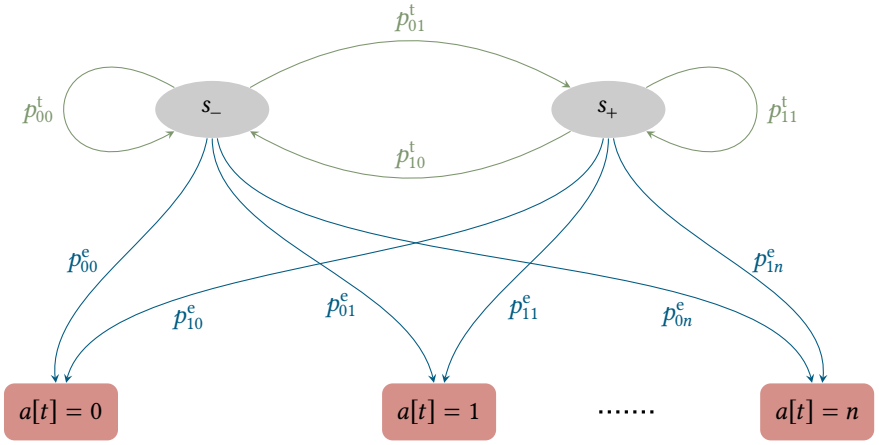


Figure 6.8: Schematic illustration of a two-state hidden Markov model model. A hidden Markov model with two hidden states s_- and s_+ (gray) is fitted to the population activity $a[t]$ (red). The transition probabilities p^{t}_{ij} (green) denote the probability of moving between the hidden states i and j, whereas the emission proabilities p_{ij}e (blue) describe the probability that the hidden state i generates the observation j.

Branching parameter: The population activity of a spiking neural network often follows a first-order autoregressive representation. In analogy to Beggs & Plenz (2003) and Wilting & Priesemann (2018), we consider the ansatz in Equation (6.1) and estimate the population activity with a binsize of $\delta t = \tau_{\mathrm{ref}}$. Here, the branching parameter m can be directly assessed, since the full network state is known and subsampling effects do not bias the estimate (Priesemann et al., 2009, 2014). Because of this, we rely on the classical estimation by computing the linear regression between $a[t]$ and $a[t+1]$ (Wei & Winnicki, 1990). For model validation pruposes, we calculate an autocorrelation time τ_{br} from m by drawing on Equation (6.8).

Autocorrelation time: However, the estimation of the autocorrelation time from the branching parameter m is only valid for vanishing external drive h. Thus, we further estimate the autocorrelation function $\rho_{a,a}$ based on the population activity $a[t]$ binned with $\delta t = \tau_{\mathrm{ref}}$:

$$\rho_{a,a}[t'] = \frac{1}{\sigma_a^2} \sum_{t=1}^{T^{\mathrm{exp}}/\delta t - t'} (a[t] - \mu_a)\,(a[t+t'] - \mu_a)\,, \tag{6.21}$$

where σ_a is the standard deviation, and μ_a the mean of the population activity. The autocorrelation time τ_{ac} is determined by fitting an exponential to $\rho_{a,a}$. We are then able to validate our model by comparing the τ_{br} and τ_{ac}.

Hidden Markov model: We fit a hidden Markov model (HMM) to the population activity $a[t]$ to identify phases of low and high activity to finally determine the dominant time scale of our neuromoprhic SNNs. To that end, we restrict our model to two hidden states s_- and s_+ (Figure 6.8). The transition and emission probabilities p^{t}_{ij} and p^{e}_{ij} are obtained by a Baum-Welch algorithm (Baum et al., 1972), whereas a Viterbi algorithm is utilized for the inference (Viterbi, 1967). We estimate the time scale of the SNNs from the transmission probabilities p^{t}_{ij}. Specifically, we determine the second largest eigenvalue of p^{t}_{ij}, λ_2. From λ_2, the time scale τ_{hm} can be estimated by applying Equation (6.8). Here, the population activity $a[t]$ is estimated with a binsize of $\delta t = 1\,\mu\mathrm{s}$ for both the HMM as well as

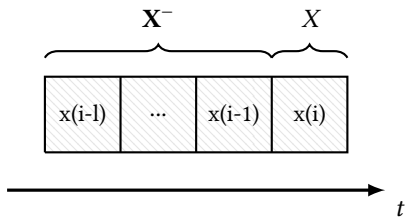


Figure 6.9: Schematic illustration of a delay embedding. For the estimation of AIS and TE as well as the PID, we reconstruct the past states $\mathbf{X}^-$ based on the activity $x(i)$. These state vectors are obtained by delay embedding with length l according to $\mathbf{X}^{\mathbf{l}}[t-1] = \{X[t-1], X[t-2], ..., X[t-l]\}$. The current value X is then simply given by $X[t]$.

the classical estimation of the autocorrelation function used for comparison. The HMM was fitted and evaluated with the *hmmlearn* Python package.

6.2.2.3 Information theory

In the following, we stick to the definitions and concepts of Wibral et al. (2015) to which we refer for further details. For the classical information-theoretic measures, we always consider pairs of neural spike trains or pairs of stimulating and neural spike trains, respectively. Their firing activity is assumed to represent two stationary random processes X_1 and X_2, composed of random variables $X_1[t]$ and $X_2[t]$, $t = 1, ..., n$, with realizations $x_1[t]$ and $x_2[t]$.

Both, the entropy (H) as well as the mutual information (I) can be directly calculated for the random variables. Hence, the mutual information (I) between X_1 and X_2 is given by:

$$\mathrm{I}(X_1 \,:\, X_2) = \mathrm{H}(X_1) - \mathrm{H}(X_1|X_2)\,. \tag{6.22}$$

To capture the fading of memory within a network, we define the lagged mutual information for time lag τ:

$$\mathrm{I}_\tau(X_1 \,:\, X_2) = \mathrm{I}_\tau(X_1[t] \,:\, X_2[t+\tau])\,. \tag{6.23}$$

Integrating this lagged I defines the memory capacity (MC):

$$\mathrm{MC}(X_1 \,:\, X_2) = \sum_{\tau=1}^{N_\tau} \delta t \left[\mathrm{I}_\tau(X_1 \,:\, X_2) - \mathrm{I}_{N_\tau}(X_1 \,:\, X_2)\right]\,, \tag{6.24}$$

with a maximal delay N_τ which we set to 100 within this work. In order to compensate for potential estimation biases, we furthermore subtract the I corresponding to N.

Since the following measures rely on past states, we start by constructing an embedding space. Specifically, the embedding vector of e. g. X_1 is constructed according to:

$$\mathbf{X_1^l}[t] = \{X_1[t], X_1[t-1], ..., X_1[t-l+1]\}\,, \tag{6.25}$$

(Figure 6.9). Here, the embedding dimension l is chosen such that the variable $X_1[t+1]$ is rendered conditionally independent of all random variables $X_1[t']$ with $t' < t-l+1$:

$$p\left(X_1[t+1]|\mathbf{X_1^l}[t], X_1[t']\right) = p\left(X_1[t+1]|\mathbf{X_1^l}[t]\right)\,, \tag{6.26}$$

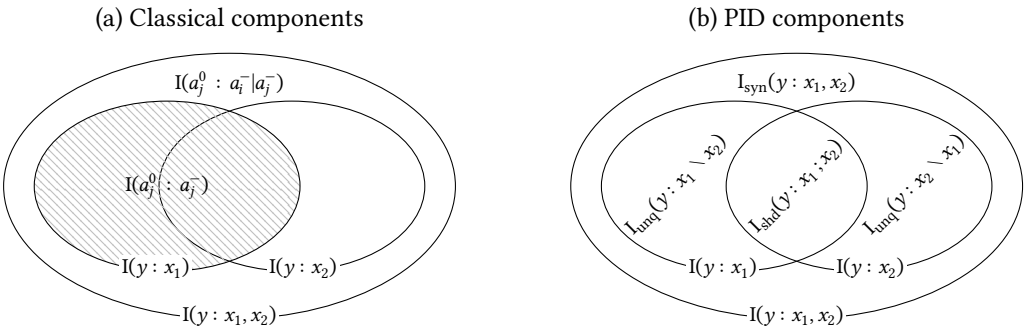


Figure 6.10: Quantification of information modfication with PID. PID allows to assess the information distribution in ensembles of agents. Here, we decompose a joint mutual information where the spiking histories $\mathbf{a_i^-}$ and $\mathbf{a_j^-}$ of two neurons correspond to the input variables, and the output variable is given by the present state a_j. **(a)** With this choice, the joint mutual information is equal to the sum of TE and AIS. **(b)** PID provides us with measures for the unique contributions of each source to the firing of the target neuron, as well as the shared (also called redundant), and synergistic contributions. Figure inspired by Wibral et al. (2017a).

where $(\cdot|\cdot)$ denotes the conditional. As a shorthand notation, we define the past state of the random process X_1:

$$\mathbf{X_1^-} \equiv \mathbf{X_1^l}[t-1] = \{X_1[t-1], X[t-2], ..., X_1[t-l]\}\,. \tag{6.27}$$

which is highlighted in bold font to clearly differentiate from the current value of the corresponding process X_1 (Figure 6.9).

The storage of information as read out from a single process is captured by the active information storage (AIS). With above's notation, the latter is given by:

$$\mathrm{AIS}(X_1) = \mathrm{I}(X_1 \,:\, \mathbf{X_1^-})\,, \tag{6.28}$$

for the process X_1. The transfer of information between a source process X_1 and a target X_2 is quantified by the transfer entropy (TE):

$$\mathrm{TE}(X_1 \to X_2) = \mathrm{I}(X_2 \,:\, \mathbf{X_1^-}|\mathbf{X_2^-})\,. \tag{6.29}$$

Alongside to these classical information-theoretic measures, we use the novel concept of PID to estimate the *information modification* (Williams & Beer, 2010; Bertschinger et al., 2013; Wibral et al., 2017a). Here, we use the same pair-wise definition as Wibral et al. (2017a). Specifically, the modification is assumed to be the information about the present state of a process, that can only be obtained by observing both the past of the own process as well as the past of a source process. Because of this, PID can be used to decompose the joint mutual information $\mathrm{I}(X_1 \,:\, \mathbf{X_1^-}, \mathbf{X_2^-})$ of a process X_1, its own past $\mathbf{X_1^-}$, as well as the past of a second process $\mathbf{X_2^-}$. This joint mutual information corresponds to the sum of the active information storage (AIS) and the TE (Figure 6.10a):

$$\mathrm{I}(X_1 \,:\, \mathbf{X_1^-}, \mathbf{X_2^-}) = \mathrm{I}(X_1 \,:\, \mathbf{X_1^-}) + \mathrm{I}(X_1 \,:\, \mathbf{X_2^-}|\mathbf{X_1^-})\,. \tag{6.30}$$

With this choice, the PID components are (Figure 6.10b):

1. The unique information $\mathrm{I}_{\mathrm{unq}}(X_1 \,:\, \mathbf{X_1^-} \setminus \mathbf{X_2^-})$ capturing the information available in the past of the own process.

2. The unique information $\mathrm{I}_{\mathrm{unq}}(X_1 : \mathbf{X_2^-} \setminus \mathbf{X_1^-})$ encompassing the information conveyed by the source process.
3. The shared information $\mathrm{I}_{\mathrm{shd}}(X_1 : \mathbf{X_2^-}; \mathbf{X_1^-})$ which is redundantly contributed by both processes.
4. The synergistic information $\mathrm{I}_{\mathrm{syn}}(X_1 : \mathbf{X_2^-}; \mathbf{X_1^-})$, which can only be obtained when observing *both* past states.

We follow the idea of Wibral et al. (2017a) and use $\mathrm{I}_{\mathrm{syn}}$ as a suitable measure for information modification. Since the decomposition can not be accomplished by the estimation of classical information terms, additional assumptions have to be made. The choices utilized within this work are outlined in Appendix A.3.

To finally evaluate all of these measures, we generate the binarized activity for every neuron and stimulus with a binwidth $\delta t = \tau_{\mathrm{ref}}$. The delay embedding vectors are constructed with $l = 4$ which builds a compromise between the required amount of data and the capturing of sufficient history. Measures incorporating more than a single process are estimated in a pair-wise scenario where the mean over all possible combinations of processes is taken. For these measures, the entropy (H) of the target process is used for normalization. All classical information-theoretic measures were estimated with the toolbox *JIDT* (Lizier, 2014). The joint mutual information was decomposed by relying on the *BROJA-2PID* estimator (Makkeh et al., 2018).

6.2.2.4 Reservoir computing

We draw on the reservoir computing framework to perform task-specific information processing with our networks (Maass et al., 2002; Jaeger, 2001). To this end, we stimulated our neuromorphic network with task-specific spike trains and consider the performance of a linear readout on the elicited neural activity trained in software. In order to not change our network by the input, we fixed all synaptic weights during the processing of inputs.

Readout: The considered readout only has access to a subset of all neurons in the network. Specifically, the activity $a_i[t]$ of a random set of neurons $\mathcal{U}$ with cardinality N_{read} is used for the training with a binary classification problem:

$$v[t] = \Theta\left(\sum_{j\in\mathcal{U}} w_j a_j[t] - \frac{1}{2}\right), \tag{6.31}$$

where $\Theta(\cdot)$ is the Heaviside function, and $v[t]$ is the predicted label. Linear regression on a set of training data with label $y[t]$ is used to determine the weight vector w_j of the classifier:

$$w_j = \arg\min_{w_j}\left(\sum_t \left|y[t] - w_j a_j[t]\right|^2\right). \tag{6.32}$$

The performance of the network is quantified by $\mathrm{I}(y[t], v[t])$ on a set of held-out test samples . Note that the multi-class scenario can be easily constructed by training multiple readout units simultaneously. For the prediction of time continuous functions, the linear readout can be modified to:

$$y[t] = \sum_{j=1}^{N} w_j a_j[t]\,. \tag{6.33}$$

To capture the continuous nature of inferred result $y[t]$, a normalized root-mean-square error (NRMSE) is used to quantify the network's performance on some test data:

$$\mathrm{NRMSE} = \sqrt{\frac{\langle x_n[t] - y[t] \rangle_t}{\sigma_y}}, \tag{6.34}$$

where we defined the standard deviation of the vote of the linear classifier σ_y. The linear regression was performed with the *scikit-learn* Python package (Pedregosa et al., 2011).

6.2.2.5 Tasks

We quantify the computational capabilities of our neuromorphic SNNs by evaluating their performance on a set of artificial tasks. The latter are designed to provide precise control of complexity or to resemble spatio-temporal input patterns.

Sum and parity: The estimation of both the temporal sum as well as the parity of the input s_i^k from the population activity $a_i[t]$ requires memorizing and combining past input. Here, we used a single Poisson-distributed spike train of rate h to stimulate all external synapses of the network, i. e. the input spike times were given by $s_i^k = s^k \, \forall \, i$. To define both functions based on these spike trains, we bin the spike times according to Equation (6.14) with a binsize of $\delta t = 1\,\mu\mathrm{s}$. The resulting activity $s[t]$ can then be directly used to calculate the n-bit sum given by:

$$z_n(s[t]) = s[t] + s[t-1] + \ldots + s[t-n+1], \tag{6.35}$$

i. e. the number of spikes that occurred in the past n time steps. Likewise, we define the n-bit parity function:

$$p_n(s[t]) = s[t] \oplus s[t-1] \oplus \ldots \oplus s[t-n+1], \tag{6.36}$$

with $p_n(s[t]) \in \{0, 1\}$ and the modulus 2 addition $\oplus$, i. e. whether an even or odd number of spikes occurred in the past n time steps.

While the classifier can be directly implemented according to Equation (6.31) for the estimation of $p_n(s[t])$, the number of readout units needs to be extended to infer $z_n(s[t])$. To that end, a winner-take-all mechanism is finally utilized to select a winning unit. To compensate for imbalance, we furthermore weighted each sample in the regression in Equation (6.32) with the respective occurrence of their corresponding class. In addition, we correct the performance I by the performance of the same classifier trained on a shuffled version of both $p_n(s[t])$ and $z_n(s[t])$.

NARMA: We also assess the performance of linear readout trained on the activity of our networks to generate continuous functions of the input. To that end, we stimulated our networks with N Poisson-distributed input spike trains of rate h and duration T^{train} and T^{test}, respectively. Based on these spike trains, we define the normalized activity:

$$\tilde{s}[t] = \frac{s[t] - \min(s[t])}{2 \cdot \max(s[t])}, \tag{6.37}$$

where $s[t]$ is obtained with a binsize $\delta t = 1\,\mu\mathrm{s}$. The nonlinear auto-regressive moving average (NARMA) can now be defined based on this non-binary observable:

$$x_n[t] = \alpha \cdot x_n[t-1] + \beta \cdot x_n[t-1] \cdot \frac{1}{n} \sum_{i=1}^{n} x_n[t-i] + \gamma \cdot \tilde{s}[t-n] \cdot \tilde{s}[t-1] + \delta, \tag{6.38}$$

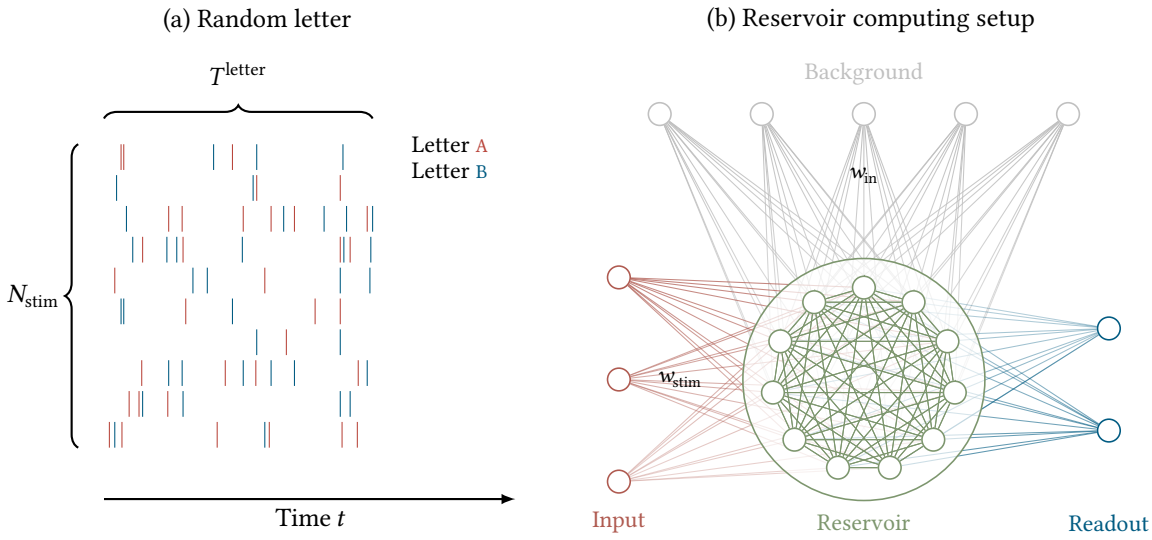


Figure 6.11: Benchmarking of SNNs with artificial letters. **(a)** Each letter consists of a set of N_{stim} Poisson distributed spike trains with duration T^{letter}. Here, we show two of these patterns A and B. The task of a network is to infer the letter identity at time t after stimulus onset. **(b)** For classification, we consider the framework of reservoir computing. Specifically, a linear readout is trained on a SNN's activity $a_i[t]$ elicited by a stimulus injected via a set of $N_{\text{in}} \times N_{\text{mu}}$ input synapses with weight w_{stim}. Simultaneously, the network is continuously stimulated with Poissonian background spikes via a set of K_{in} synapses with weight w_{in}.

with the parameters $\alpha = 0.3$, $\beta = 0.05$, $\gamma = 1.5$, $\delta = 0.1$ (Jaeger, 2003). Note that we normalized the second operand with n, since the latter is considered as a measure of complexity in close accordance to the sum and parity tasks.

Random pattern: Next, we target the classification of artificial letters. In more detail, these comprise of N_{stim} spike trains of rate $\Delta h = 30\,\text{kHz}$ and duration T^{letter} with Poisson statistics. Different classes can now be generated by considering various seeds for their generation. Here, we only discuss results for a binary classification of the letters A and B (Figure 6.11a). The associated spike trains were used to stimulate the reservoir network (Figure 6.11b). Specifically, each spike train was multiplexed N_{mu} times resulting in a total of $N_{\text{stim}} \times N_{\text{mu}}$ inputs, each of which injected into a single synaptic row of HICANN-X to elicit sufficiently high activation levels within the emulated SNN. For that purpose, we configured the two lowermost bits of the address labels of N_{div} randomly selected synapses per synaptic row to the identifier of the second input population. In addition, their weight values were set to w_{stim}. This choice allowed us to simultaneously stimulate the network with Poissonian spikes via the first input population which continuously received input from the background spike sources. By injecting the very same pattern N_{sample} times, we generated different samples. Again, we train a linear readout on the elicited activity $a_i[t]$ on 50 % of all acquired samples to finally classify the letter identity. As done previously, we quantify the performance by a mutual information based on the remaining 50 % of the samples. We utilized a binwidth $\delta t = 1\,\mu\text{s}$ for the estimation of $a_i[t]$.

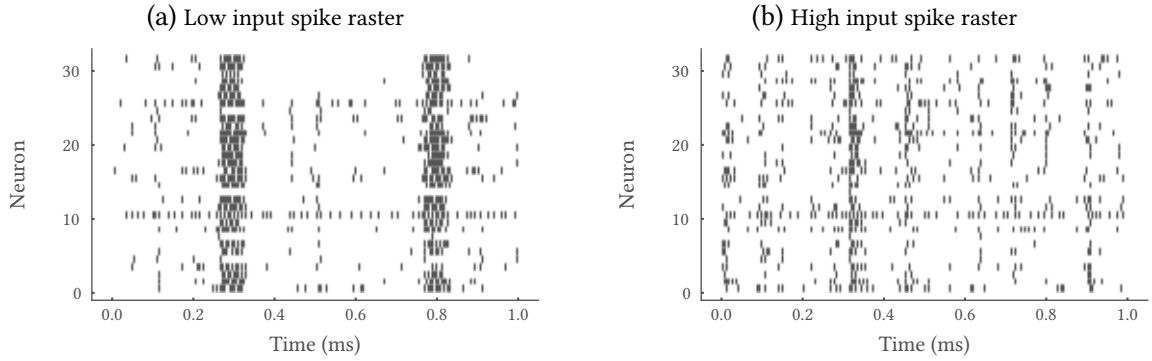


Figure 6.12: The degree of external input K_{ext} shapes the collective dynamics of the network. (a) For a low degree of the input ($K_{ext} = 0.25$), strong recurrent connections develop, and the activity shows irregular bursts, resembling a critical state. **(b)** For a high degrees of the input ($K_{ext} = 0.56$), firing becomes more irregular and asynchronous. Figure and caption taken from Cramer et al. (2020a).

6.3 Results

Throughout this work, we consider SNNs mimicking basic aspects of neural systems. In-vivo in awake animals, the latter exhibit a hierarchy of autocorrelation time scales ranging from milliseconds within sensory to hundreds of milliseconds in frontal areas (Murray et al., 2014; Hasson et al., 2015). In more detail, cortical areas associated with higher-order brain function feature a higher autocorrelation time. The functional meaning of this form of correlation within neuronal tissue is the implementation of working memory (Christophel et al., 2017). In particular, this memory is encoded in the dynamical states. Most of this knowledge is, however, only supported by data which is why comprehensive models are required. In terms of these models, the mechanisms leading to autocorrelations within the population activity could be carved out to build a fundamental understanding of time scales in SNNs and finally in biological tissue. Moreover, these models allow the investigation of the functional role of diverse time scales within a single system.

Within this section, we present two distinct mechanisms causing collective dynamics by means of correlations within the population activity of SNNs. Both underlying frameworks allow us to control the time scale of these correlations by adapting the input strength. While the dynamics of the STDP framework seem to resemble a continuous phase transition, the homeostatic framework displays bistable behavior. For both setups, we propose scenarios in which the collective dynamics can be exploited for efficient information processing without highly detailed and task-specific learning mechanisms within the network. Note that all measures of time are given in wall clock time and have to be scaled by a factor of 1000 to obtain their biological equivalent (Section 6.2.1).

6.3.1 STDP framework

Within our STDP framework, we consider plastic networks of LIF neurons emulated on HICANN-DLS. In total, we utilize all 32 neurons with the associated 32×32 synapses programmed to feature 20 % inhibitory synapses. All synapses are plastic and subject to a modified version of STDP composed of a positive drift and the negative anticausal part of STDP. In particular, this choice promotes stable network activity – i. e. no pathologically low or high firing rates – even in the presence of

hardware imperfections (Table 6.1). The synaptic plasticity is implemented on-chip, running alongside the analog emulation of neuro-synaptic dynamics. Hence, the communication with the host system is kept at a minimum which facilitates an uninterrupted emulation and in turn, allows to fully profit from the accelerated nature of BrainScaleS-2.

Here, we consider fixed indegree networks implemented on the BrainScaleS-2 prototype system HICANN-DLS which were implemented by configuring a set of K_{ext} synapses per neuron to relay Poisson-distributed distributed input spikes. All remaining synapses were used to relay recurrent spikes. This choice leads to the extreme cases of a feed-forward network for $K_{\text{ext}}/N = 1$ and a fully connected recurrent topology without stimulating synapses for $K_{\text{ext}}/N = 0$. We discover vastly different dynamics depending on the choice of K_{ext}: While the network activity is more asynchronous-irregular for low K_{ext} (Figure 6.12b), it becomes increasingly synchronous for a decreased input strength K_{ext} (Figure 6.12a) in direct accordance with the work of Zierenberg et al. (2019).

6.3.1.1 Control of critical dynamics

The change in collective dynamics evidenced by Figure 6.12 closely resembles the dynamics expected at a non-equilibrium continuous phase transition. Indeed, we find the associated critical phenomena in the avalanche distributions (Figures 6.13 and 6.14) and the branching parameter (Figure 6.15a) as well as the autocorrelation time (Figure 6.15b). Similarly, the networks become increasingly sensitive to internal as well as external perturbations as indicated by the susceptibility and trial-to-trial variability (Figure 6.15d).

We consider the framework of a branching process to test for critical dynamics (Zapperi et al., 1995; Harris, 2002; Watson & Galton, 1875; Wilting & Priesemann, 2018). Within this formulation, a spike at time t triggers on average m postsynaptic spikes at time $t + 1$. Critical dynamics emerge for a branching parameter of $m = 1$ where large discharges of spikes – so-called *avalanches* – define the population activity (Beggs & Plenz, 2003; Harris, 2002). The size s of an avalanche is given by the number of events within a cascade. Being at a critical point, the sizes are expected to be distributed according to a power law. Indeed, our network activity shows power-law distributed avalanche sizes over two orders of magnitudes for low input strengths (Figure 6.13a). Quantified by fitting exponentials and power-laws (Clauset et al., 2009), the distributions are best fitted by a power-law for low K_{ext} (Figure 6.13d). The associated critical exponents α_s are found by fitting a truncated power-law. Only for the lowermost K_{ext}, α_s closely matches the theoretically predicted critical exponent $\alpha_s \approx 1.5$ (Zapperi et al., 1995). Moreover, this fit also provides an increasing cutoff s_{cut} for decreasing input strengths. However, for the lowest value of K_{ext} the networks tend to become unstable which is why s_{cut} (Figure 6.13b) and the model comparison (Figure 6.13d) peak for higher values of the input strength. Summing up, the assessment of neural avalanches suggests a continuous phase transition for low K_{ext} and, more importantly, the distance to critical-like dynamics can be controlled by the input strength K_{ext}.

We draw on software simulations of equivalent networks to investigate finite-size scaling due to the limited size of our neuromorphic network. In more detail, we simulated an equivalent low input strength network in *Brian 2* and scaled the system size N accordingly. Just like their smaller counterparts emulated on HICANN-DLS, these low degree networks reveal power-law distributed avalanches size s (Figure 6.14a). Moreover, a truncated exponential fit suggests that the cutoff s_{cut}

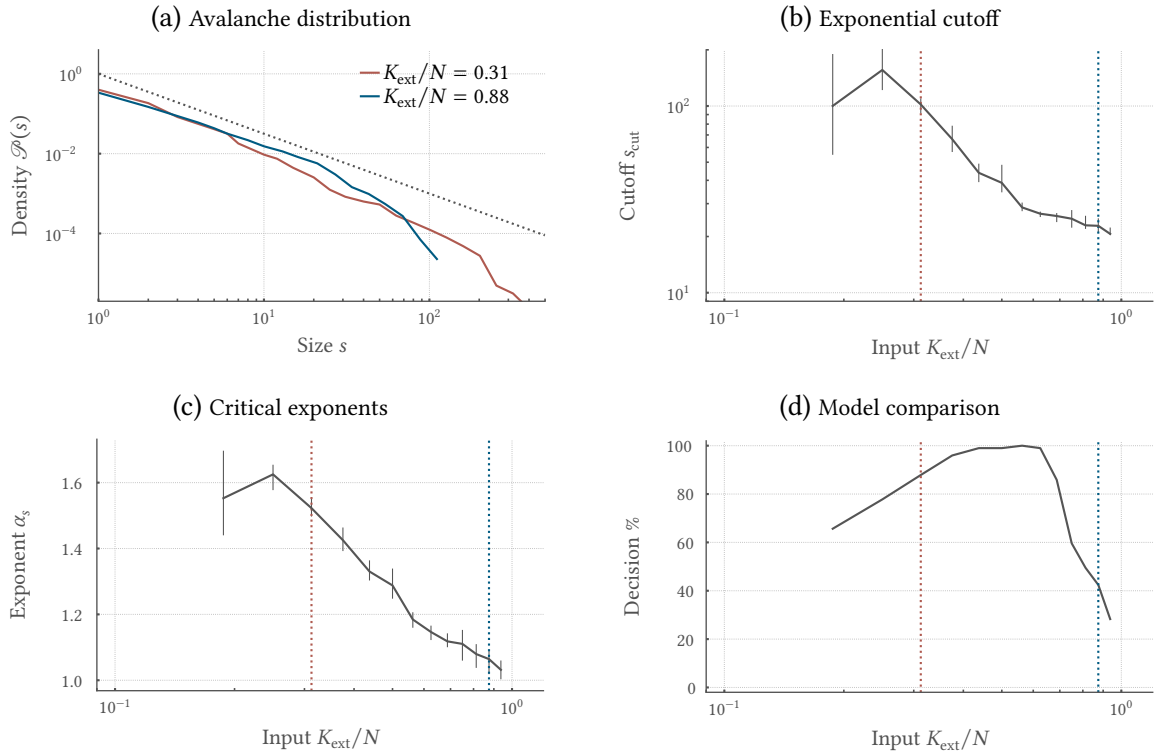


Figure 6.13: Under low degree of input K_{ext}, the network self-organizes towards a critical state, and showed long-tailed avalanche distributions. (a) Distributions of avalanche sizes s resemble power-laws over two orders of magnitude for low K_{ext}. Fitting a truncated power law, **(b)** the exponential cutoff s_{cut} peaks, and **(c)** critical exponents α_s approximate 1.5, as expected for critical branching processes. **(d)** A maximum-likelihood comparison decides for a power-law compared to an exponential fit in the majority of cases. Dashed vertical lines indicate the set of K_{ext}/N values that have been selected in (a). In this and all following figures, the median over runs and (if acquired) trials is shown, and the errorbars show the 5%-95% confidence intervals. Figure and caption taken from Cramer et al. (2020a).

of the avalanche distribution scales with N with an associated scaling exponent of 1.6 ± 0.2 (Figure 6.14b). These simulation results underpin the assumption that our neuromorphic networks self-organize to a critical point for low degrees of the input K_{ext}.

The implementation on the accelerated HICANN-DLS chip promises an efficient and fast emulation of neuro-synaptic dynamics. By taking up the aforementioned *Brian 2* simulations, we are able to directly compare the run time of emulation and simulation. Despite the small system size, the emulation on HICANN-DLS already outperforms the simulation on conventional hardware with a speedup factor of 100 , even when incorporating the overhead introduced by spike transfer. More precisely, the simulation of a single plasticity experiment with a duration of 600 s in biological time takes 570 s in *Brian 2* while the emulation requires only 6 s. In general, the simulation with detailed synaptic plasticity often scales with $\mathcal{O}(N^2)$ for all-to-all connected networks, i. e. with the number of synapses that need to be updated. In contrast, HICANN-DLS naturally captures the intrinsic parallelism of neuro-synaptic dynamics and hence renders the emulation time independent of the system size N as long as the network can be mapped to chip.

Next, we assess the branching parameter in a direct fashion to further strengthen the prevalence of the universality class of a critical branching process. The branching parameter m captures the

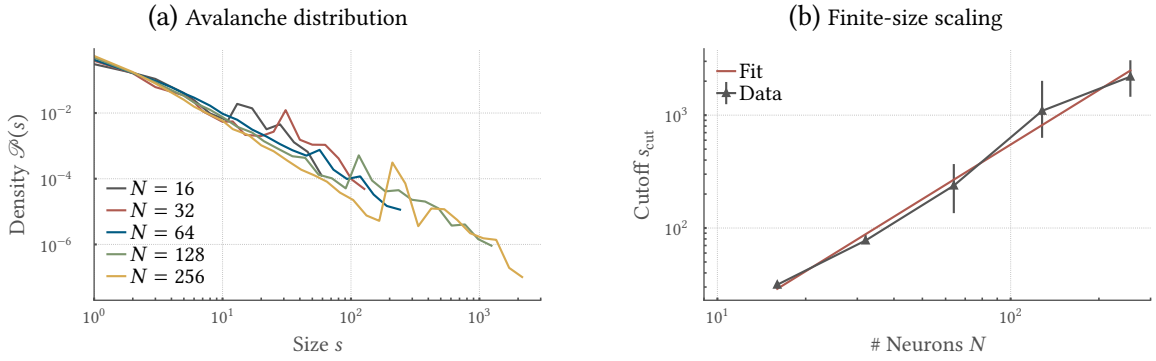


Figure 6.14: Finite-size scaling is assessed using a software implementation with varying system size N. (a) Exemplary avalanche size distributions follow a power-law for any tested N (degree of the input $K_{\text{ext}}/N = 1/4$). **(b)** As expected for critical systems, the cutoff s_{cut} scales with the system size with scaling exponent 1.6 ± 0.2. Figure and caption taken from Cramer et al. (2020a).

spread of activity within a network and is smaller than unity for sub-critical dynamics, whereas it is larger for super-critical processes. For all K_{ext}, our networks shows $m < 1$ and hence sub-critical dynamics. However, m tends towards unity for low K_{in} (Figure 6.15a). The theoretical relation $\tau_{\text{br}}(m) \sim \lim_{m \to 1}(-1/\log(m)) = \infty$ suggests that the autocorrelation time τ_{ac} diverges at a critical point (Wilting & Priesemann, 2018). In a first step, we directly estimated the autocorrelation time from the population activity τ_{ac} which indeed peaked for low K_{ext} (Figure 6.15b). Building on this result, the relation between autocorrelation time and m in Equation (6.8) allows us to validate the model used to determine m. The times obtained by both methods indeed closely match, characterized by a correlation coefficient $\rho = 0.998$ with $p < 10^{-10}$ (Figure 6.15c). As a result, this assessment of m as well as the autocorrelation time strengthens our assumption of critical-like dynamics for low K_{in}.

A major computational advantage of operating in the vicinity of a critical point is the enhancement of differences within the stimulus. This is captured by a diverging susceptibility χ for critical-like dynamics. Here, we define χ by the change in the population activity in response to a burst of $N_{\text{pert}} = 6$ additional spikes embedded into the regular Poisson spike input. Indeed, this susceptibility is highest for low K_{ext} (Figure 6.15d). Moreover, not only does the response to this external perturbation increase, but also intrinsic variations are strongly amplified. The latter are caused by the analog nature of the substrate which exhibits temporal noise within all dynamical variables of the model. Quantified by a van-Rossum distance Δ_{vrd}, these trial-to-trial variations also peak for networks closer to criticality (Figure 6.15d). Hence, not only the avalanche size distribution, but also the considered dynamical properties suggest self-organization to criticality for low input strength.

6.3.1.2 Adjusting networks to task requirements

In the following, we test the prevailing assumption that criticality is beneficial for task processing in general. Our results showcase that this general statement has to be phrased more carefully. While we already found that abstract computational properties are indeed maximized in the vicinity of a critical point this does not necessarily translate to high task performance (Shew et al., 2011; Shew & Plenz, 2013; Barnett et al., 2013; Wilting & Priesemann, 2018). Indeed, there are also tasks for which critical dynamics are detrimental. Hence, the network state needs to be carefully adjusted to task requirements, i. e. every task needs its own dynamics to be optimally solved.

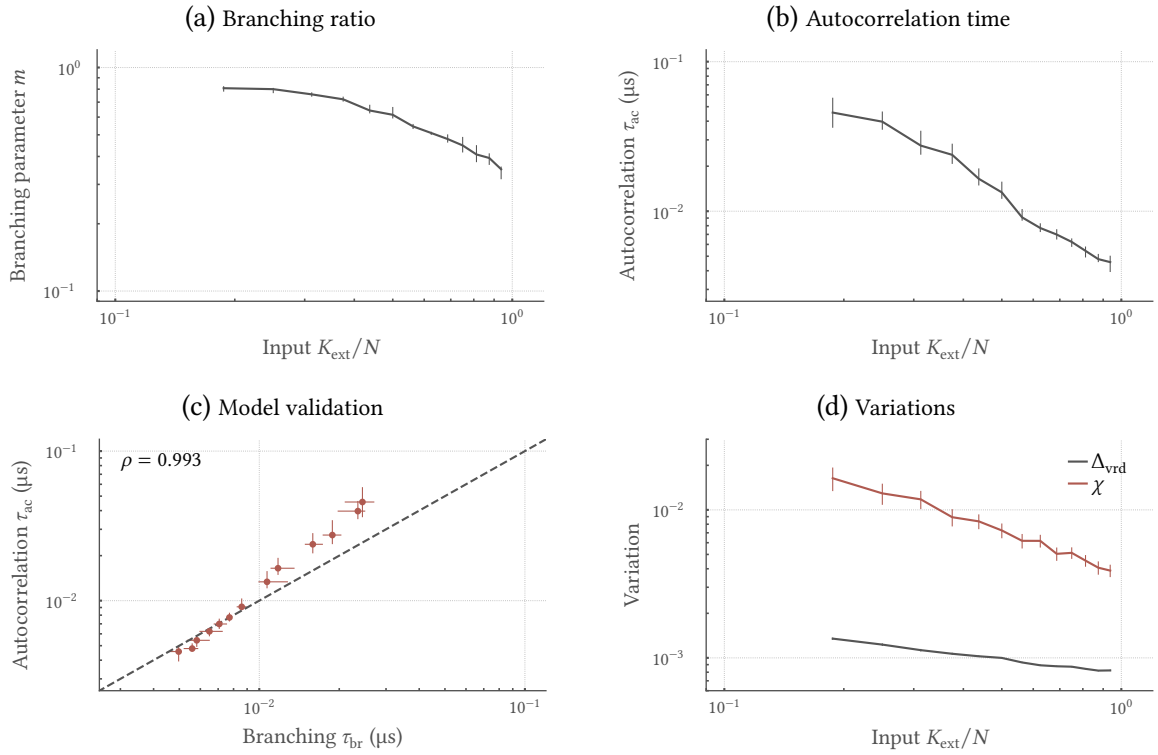


Figure 6.15: For low degrees of the input K_{ext}, the network show clear signatures of criticality beyond power-laws. Only for low values of K_{ext}, **(a)** the estimated branching parameter m tends towards unity, and **(b)** the estimated autocorrelation time τ_{ac} peaks. **(c)** The match of the τ_{ac}, and the $\tau_{br} \sim -1/\log(m)$ as inferred from m support the criticality hypothesis (correlation coefficient of $\rho = 0.998$, $p < 10^{-10}$). **(d)** Trial-to-trial variations Δ_{vrd} as well as the susceptibility χ increase for low K_{ext}. Figure and caption taken from Cramer et al. (2020a).

The performance of a recurrent SNN on a specific benchmark task can be evaluated in a *reservoir computing* framework. In more detail, a linear readout can be trained to separate different input sequences based on the activity of a recurrent network stimulated by these sequences (Maass et al., 2002; Jaeger, 2001; Schürmann et al., 2005). Aside from the ability to separate different input pattern, the fading of memory over extensive time-spans facilitate performance. The latter is especially important when past and present input need to be combined. In this scenario, additional dynamical memory might bridge the gap between neuro-synaptic time constants and the time scales involved in real-world stimuli.

Within this work, we rely on a n-bit sum and a n-bit parity to test the performance of a linear readout trained on $N_{read} = 16$ randomly selected neurons within the recurrent network. In order to achieve high performance on this benchmark, networks need to *memorize* and *combine* the input from n past time steps. Hence, the task complexity increases with n and can be further dissected by the comparison of the linear sum and the non-linear parity task. Due to the extensive memory time spans of critical networks – usually quantified by a diverging mutual information – tasks with high n are likely to profit from critical dynamics. In contrast, simple tasks with low n might suffer from critical dynamics due to dispensable memory of past input not required for task processing. In summary, we expect that – depending on the task complexity – there is an optimal input strength K_{ext} which leads to best performance.

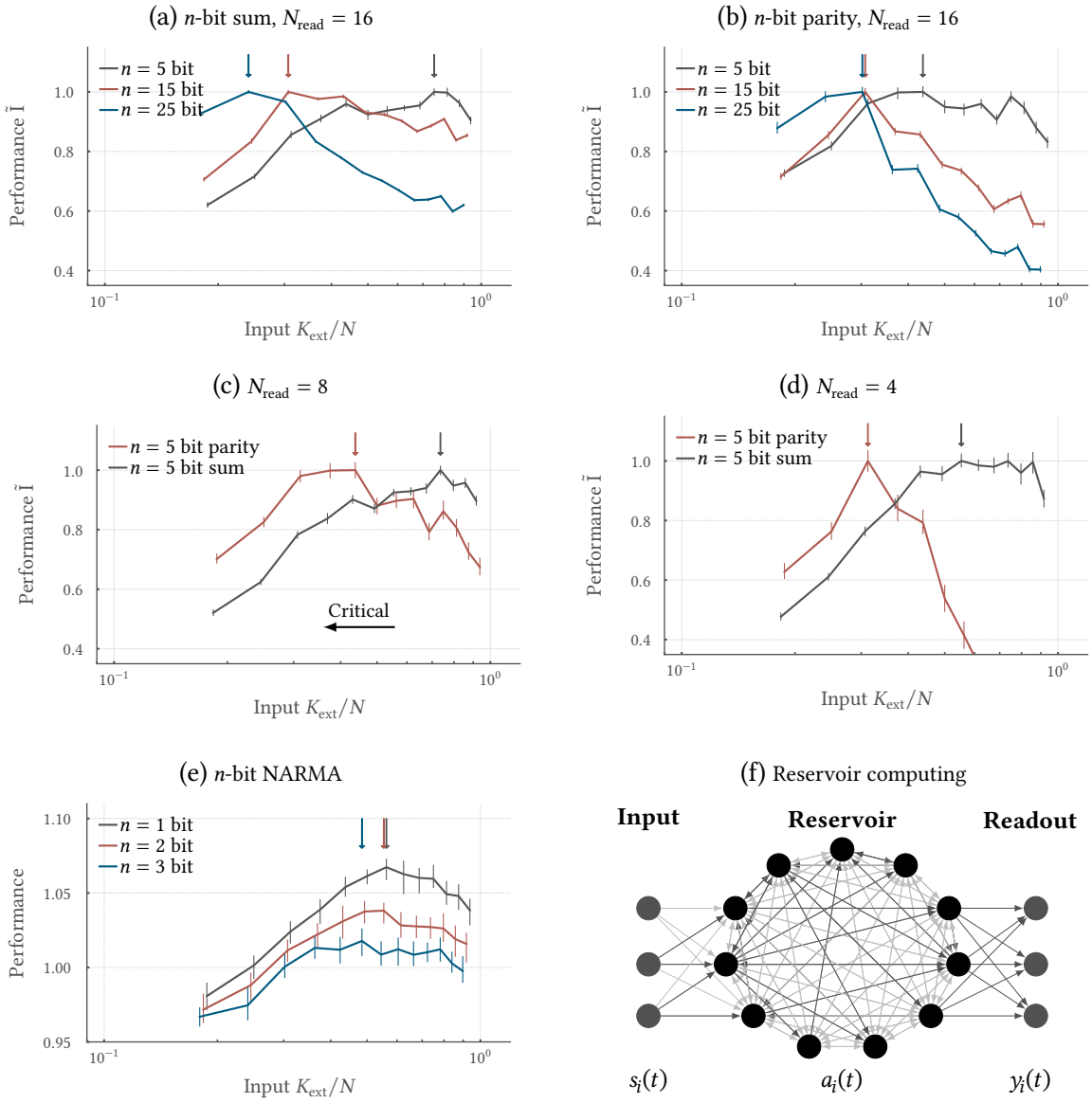


Figure 6.16: Computational challenging tasks profit from critical network dynamics (small K_{ext}) – simple tasks do not. The network was used to solve **(a)** a n-bit sum and **(b)** a n-bit parity task by training a linear classifier on the activity of $N_{\text{read}} = 16$ neurons. Here, task complexity increases with n, the number of past inputs that need to be memorized and processed. For high n, critical dynamics are beneficial for task performance, whereas simple tasks suffer from criticality. Especially, the more complex, non-linear parity tasks profit from criticality. Further, task complexity can be increased by restricting the classifier to **(c)** $N_{\text{read}} = 8$ and **(d)** $N_{\text{read}} = 4$. Again, the parity task increasingly profits from criticality with decreasing N_{read}. The performance is quantified by the normalised mutual information $\tilde{I}$ between the vote of the classifier and the parity or sum of the input. **(e)** Likewise, the peak performance moves towards criticality with increasing complexity in a NARMA task. The performance is quantified by the inverse NRMSE. Highest performance for a given task is highlighted by colored arrows. **(f)** Schematic reservoir computing setup. Figure and caption taken from Cramer et al. (2020a).

Indeed, the task performance of our networks depends on a combination of both task complexity as well as the distance to critical-like dynamics. While the simple sum task with $n = 5$ bit is solved best by sub-critical networks, the complex $n = 25$ bit sum task requires critical dynamics with as-

sociated long lasting memory (Figure 6.16a). Incorporating non-linearity in the consideration even pronounced this effect: Even a $n = 5$ bit parity task is better solved by increasingly critical networks (Figure 6.16b). Because of this, the networks need to be tuned to task requirements which can actually be done by adjusting the input strength.

A standard task for the benchmarking of reservoir networks is the NARMA task (Jaeger, 2001). In contrast to the sum and parity task, the calculation of a NARMA requires multiple computing operations. However, all tasks have in common that their complexity can be controlled by the number of incorporated past time steps n. Again, the peak performance becomes increasingly shifted into the critical regime when increasing n (Figure 6.16e).

Another task-independent approach to increase complexity is to reduce the number of neurons visible to the readout N_{read}. In general, the information for solving the aforementioned tasks can be available within the activity of a single neuron of our networks. For low N_{read}, we expect best performance for networks with low K_{ext}, since the spatio-temporal correlations associated with emerging critical phenomena render the network activity redundant (Figure 6.12a). This is tested by training the linear readout on a random subset of neurons of the network for the $n = 5$ bit sum and parity task, respectively. Here, only low N_{read} lead to highest task performance of critical reservoirs on the parity task (Figures 6.16c and 6.16d). The information required for the sum task seems to be globally available in each network and can hence be solved even with sub-critical dynamics. In general, the local availability of information is of equal importance for both large neuromorphic systems (Schemmel et al., 2010) as well as biological networks (Brette, 2019; Bernardi & Lindner, 2017). For both of which the communication of information in form of spikes between arbitrary neurons is costly or not at all possible. Hence, the redundancy provided by the discovered critical-like dynamics facilitates the local availability of task-relevant information globally within the network.

6.3.1.3 Strategies to dynamically switch between network states

As stated above, networks need to be tuned to task requirements in such a way that complex tasks require critical dynamics whereas simple ones profit from sub-critical states. In general, the adjustment of networks to computational demands by controlling the input strength can actually be implemented by different strategies. The instantiation of many networks – each with its own input strength and hence different dynamics – can be accomplished on multiple chips or a single large wafer-scale system, respectively. In this scenario, all networks operate in parallel which manifests in a fast emulation. However, this approach requires many resources. In another approach, the configuration of previous adaptation experiments for different K_{ext} can be saved. Depending on the task, the desired configuration can be readily loaded into the chip for task processing. This on the downside requires additional memory to store the configuration data and hence stresses the input bandwidth which can become a bottleneck for very large networks or fast switching of states.

A resource aware approach instead exploits the accelerated nature of HICANN-DLS by changing the input strength K_{ext} according to task requirements accompanied by the self-organization to a new state. Here, we investigate the transition from critical to sub-critical dynamics and vice versa while simultaneously assessing the performance as well as the branching parameter. Most notably, switching between previously adapted states yields final states comparable to emulations starting from a quiescent state with $w_{ij} = 0 \,\forall\, i, j$, indicated by comparable task performance as well as similar branching parameters (Figure 6.17). This dynamic adaptation of the input strength comes with minimal requirements, i. e. no additional memory needs to be allocated and the I/O bandwidth can solely

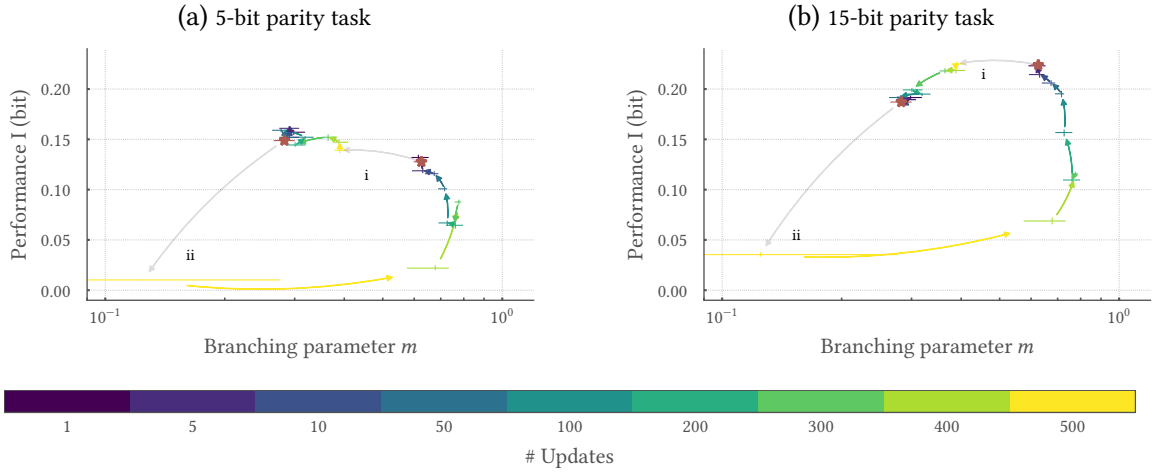


Figure 6.17: Network can be dynamically adapted by changing the degree of the input K_{ext}. After convergence of synaptic weights w_{ij}, K_{ext}/N was (i) switched from critical ($K_{ext} = 0.3$) to sub-critical ($K_{ext} = 0.8$) and (ii) vice versa. The branching parameter m and the performance I of the network on **(a)** the 5-bit and **(b)** the 15-bit parity task were evaluated after various numbers of synaptic updates. Here, the network reaches the same performance and dynamics as when starting from $w_{ij} = 0 \,\forall\, i, j$ (marked by red stars). For both tasks, the transition from sub-critical to critical dynamics require more updates. Moreover, optimal performance for (a) the 5-bit task is achieved under strong input (i), whereas for (b) the 15-bit task requires low input (ii). The performance is quantified by the mutual information I between the parity of the input and the vote of a linear classifier. Figure and caption taken from Cramer et al. (2020a).

be used for spike traffic. Hence, this approach is particularly promising for large neuromorphic systems.

For the aforementioned dynamic switching of states to be a useful strategy, the self-organization to the novel state has to be fast enough. A transition from a critical to a sub-critical state is completed with only 50 synaptic updates, whereas a sub-critical to critical transition requires on average 500 weight updates (Figure 6.17). It is noteworthy that due to the accelerated nature of HICANN-DLS, the reported numbers of updates correspond to only 50 ms and 500 ms wall clock time. These durations can most likely be even further reduced by lowering the integration time used for the measurement of the accumulation traces in Equation (3.2). Because of this, dynamical switching represents a desirable strategy for accelerated systems, especially in presence of limited resources.

6.3.1.4 Task-independent quantification of computational capabilities by information theory

The assessment of performance in classification tasks is standard (cf. Chapter 4), but poses several problems. First, it highly depends on the task at hand and second, such an approach is not applicable for measurements on biological systems, like higher brain areas or *in-vitro* preparations, due to the unknown neural code. A *task-independent* consideration of performance can instead be achieved by methods from information theory (Wibral et al., 2015). Here, the lagged mutual information between any input spike train s_i and the activity of a neuron after time lag τ, a_j predicts the performance on the parity task under standard Poisson input: Sub-critical networks only provide high information about the input on relatively short time scales and hence short τ of about 20 µs (Figure 6.18a). The forgetting on larger time scales is, however, required for erasing irrelevant past input. In contrast,

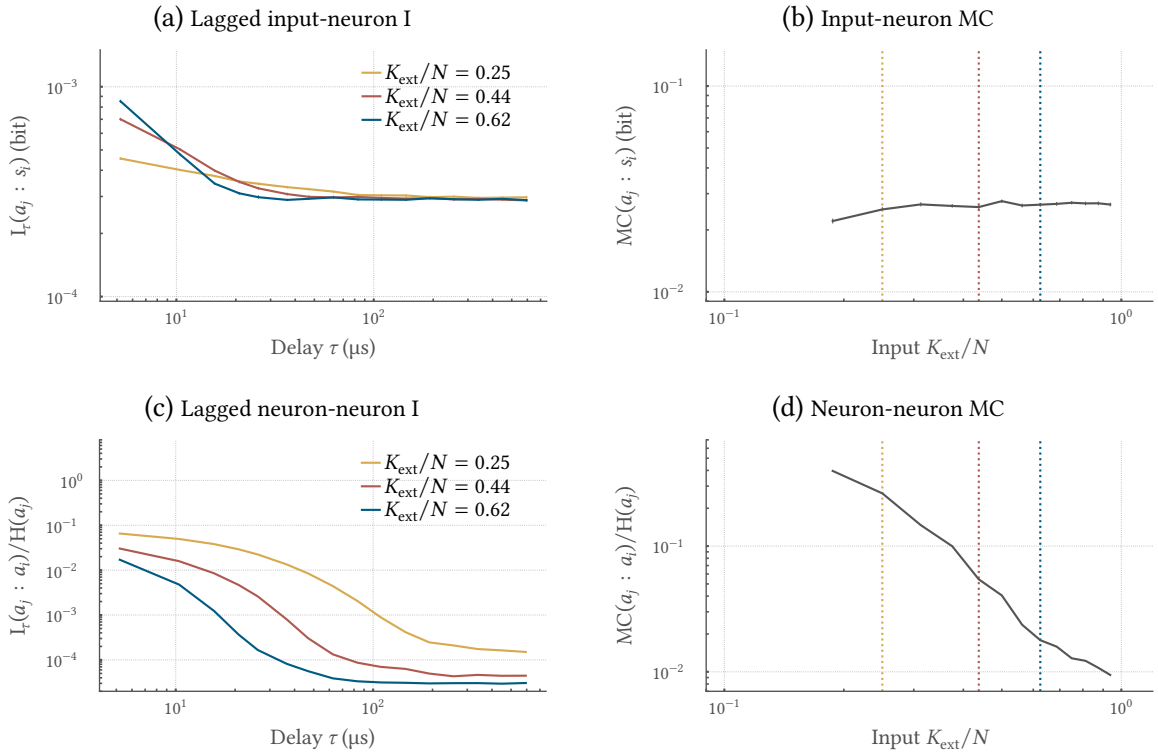


Figure 6.18: Long lasting memory accompanies critical network dynamics. (a) Memory about the input s_i as read out from neuron a_j after a time lag τ is quantified by the mutual information $I_\tau(a_j, s_i)$. Here, high degrees of the external input K_{ext} are favorable for memory on short time scales, whereas a low K_{ext} is favorable on larger time scales. **(b)** The memory capacity (MC) stays fairly constant, despite of a decreased coupling to the stimulus for low K_{ext}. **(c)** The lagged I between the activity of pairs of neurons indicates increasing memory for decreasing K_{ext}, also visible in the memory capacity (MC) **(d)**. The selection of K_{ext}/N in (a) and (c) is marked by dashed vertical lines in (b) and (d). Figure and caption taken from Cramer et al. (2020a).

critical networks with dominant recurrence provide longer memory of about 60 µs. The latter is required to combine past and novel input, a necessity for high performance on any of the previous high n tasks. However, this increased memory comes at the cost of a less reliable representation of the input within a single neuron's activity. By integrating I_τ over all delays, a measure of input representation can be obtained. Most notably, this memory capacity (MC) stays almost constant over all tested K_{ext} (Figure 6.18b). It is noteworthy that a linear classifier, in contrast, could draw on the activity of a subset of neurons which jointly provide a better representation. Here, however, we only consider the representation within a single neuron which is also easily accessible within experiments.

The memory is implemented by the recurrent network dynamics, i. e. by activity propagating through recurrent connections which is why it is often referred to as AIS (Lizier et al., 2008; Wibral et al., 2014). Dominant recurrent activity for low degrees of the input K_{ext} and hence critical dynamics leads to larger lagged I between the activity of pairs of neurons within our networks (Figure 6.18c). Because of this, the MC strongly increases over almost two orders of magnitude when lowering K_{ext} (Figure 6.18d). This internal MC boosted the performance on the high n sum and parity tasks (Figure 6.16a and b).

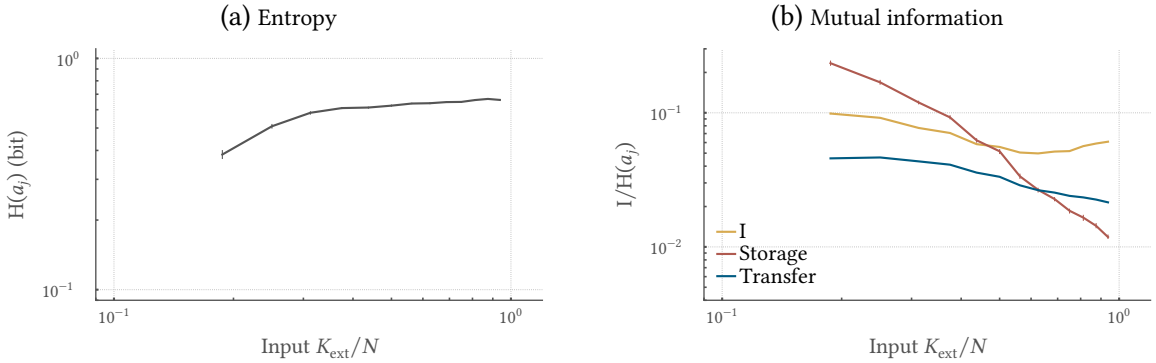


Figure 6.19: The information fingerprint changes with the degree of the input K_{ext}, thus with distance to criticality. (a) The entropy (H) of the spiking activity of a single neuron, a_j stays fairly constant, except for low K_{ext} as a consequence of decreasing firing rates. **(b)** The mutual information (I) between the activity of two units a_i, a_j increases with lower K_{ext} (i. e. closer to critical). The network intrinsic memory also grows, indicated by the active information storage (AIS) $I(a_j : \mathbf{a}_j^-)$. Likewise, the information transfer within the network increases with lower K_{ext}. The information transfer is measured as transfer entropy (TE) between pairs of neurons a_j and a_i, $I(a_j : \mathbf{a}_i^- | \mathbf{a}_j^-)$. Figure and caption taken from Cramer et al. (2020a).

Information theory does not only provide us measures for H and I, but furthermore allows to quantify the transfer and storage of information (Wibral et al., 2015, 2017a; Schreiber, 2000). Additionally, it allows disentangling unique, redundant and synergistic contributions when considering ensembles of agents (Williams & Beer, 2010; Bertschinger et al., 2013). All of these measures increase when approaching a critical point, i. e. for low K_{ext} (Figures 6.19b and 6.20b). Hence, the overall capacity of our networks increases in the vicinity of a critical point in close accordance with previous work (Boedecker et al., 2012; Barnett et al., 2013; Bertschinger & Natschläger, 2004; Langton, 1990).

Specifically, the AIS of a single neuron, the I as well as the TE between pairs of neurons increase for decreasing K_{ext} (Figure 6.19b). Most notably, the associated memory is implemented dynamically with recurrent activity and not within a single neuron's state, since the binsize used for analysis exceeds all time constants of the LIF model (refractory period τ_{ref}, synaptic τ_{syn} and membrane τ_m time scales). Hence, information theory reveals that the transfer of activity and the storage of information within the networks increase when approaching critical-like dynamics. Increasing I, AIS and TE have likewise been reported for the Ising model and reservoirs with critical dynamics (Barnett et al., 2013; Boedecker et al., 2012) and therefore support our notation that criticality indeed maximizes the abstract information processing capacity. However, this does not necessarily translate to task processing as shown above for simple tasks.

The dissection of information in ensembles of neurons is enabled by PID (Williams & Beer, 2010). Here, we considered the decomposition of a joint mutual information of the present state of a neuron a_i, its past state $\mathbf{a}_i^-$, as well as the past state of a source neuron $\mathbf{a}_j^-$. PID allows us to disentangle not only the information from each of these two past states individually, but even further provides us with redundant and synergistic information. The redundant contribution corresponds to information that can be obtained by either source respectively. In contrast, the synergistic contingent denotes the information that can only be computed when knowing both input variables (Figure 6.10b).

Not only the joint I, but all PID components increase when lowering K_{ext} (Figures 6.20a and 6.20b). Most notably, the redundant and synergistic contributions exceed the unique terms by an order of

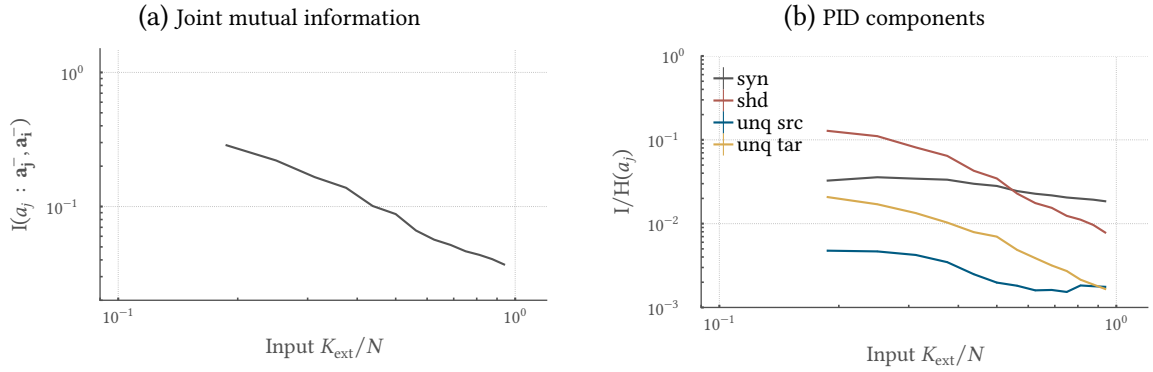


Figure 6.20: PID components increase towards criticality (i. e. with smaller input K_{ext}) (a) The joint mutual information (I) increases with decreasing K_{ext}. **(b)** All PID components increase with approaching criticality. Interestingly, the synergistic and shared contributions are always much larger than the unique contributions (note the logarithmic axis). This highlights the collective nature of processing in recurrent neural networks. Figure and caption taken from Cramer et al. (2020a).

magnitude. For critical-like dynamics, the shared information dominates, underpinning synchrony and redundancy between neurons. This, however, did not impede the performance at criticality. Nevertheless, this increase in shared information is likely to harm performance when approaching the critical point even further. Although the synergistic contribution increases towards the critical point, it exceeds all other components for sub-critical networks. The latter might be due to the fact that joint activation of multiple neurons within the network is required to lead to the emission of a postsynaptic spike.

6.3.2 Homeostatic framework

Motivated by Zierenberg et al. (2018), we consider SNNs subject to homeostatic regulation emulated on HICANN-X in the remainder of this chapter. Specifically, we study recurrent networks composed of 512 LIF neurons with the associated 256×512 current-based synapses configured to feature 20 % inhibition. A population of 256 spike sources emitting Poisson distributed spike trains at a rate of $h = 10\,\text{kHz}$ serves as input for the LIF neurons. To that end, a randomly drawn set of K_{in} out of the $N_{in} = 256$ synapses per neuron is utilized. Furthermore, $N_{rec} = 102$ randomly drawn synapses per neuron implement recurrent connections. All recurrent synapses – excitatory and inhibitory ones – are homeostatically regulated while the stimulating ones were clamped to a fixed shared weight. The homeostatic weight updates are calculated and applied by the embedded PPUs alongside the analog emulation of neurons and synapses.

The on-chip implementation of homeostatically regulated SNNs allows us to take full advantage of the accelerated emulation. Here, the I/O can be dramatically relieved by resorting to the on-chip spike sources for network stimulation and by calculating weight updates on the SIMD vector units of the PPUs. Specifically, we consider a stochastic formulation of the homeostatic regulation in Equation (6.4) which guarantees smooth convergence and an average firing rate of $\nu_{target} = 10\,\text{kHz}$ per neuron (Figure 6.21a and b). In more detail, each synaptic weight update is accepted with a probability p to cope with the available fixed-point, limited precision arithmetic. Irrespective of the learning rate, only low values of p lead to smooth convergence and the attainment of the desired target rate on the neuromorphic substrate (Figure 6.21c). With this on-chip implementation, the

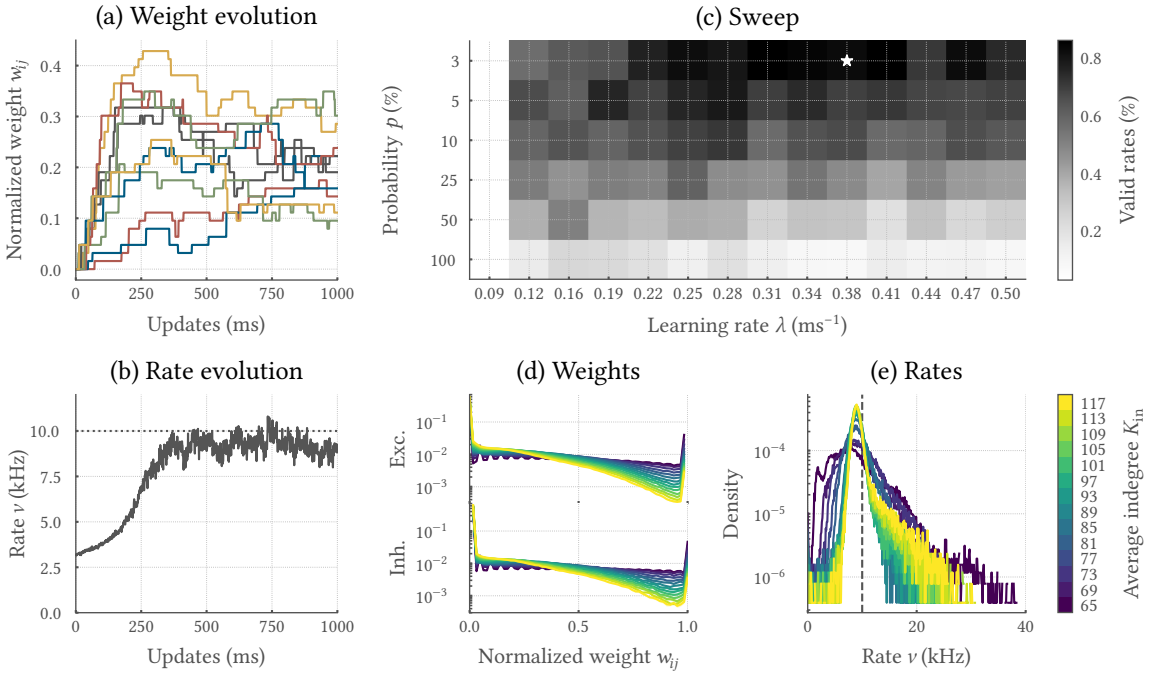


Figure 6.21: Stochastic homeostatic plasticity stabiliy regulates neuronal activity on HICANN-X. Stochastic weight updates allow to reach a stable firing rate despite of limited precision, fixed-point arithmetics as well as discrete weights for various average indegrees K_{in}. **(a)** Homeostatic regulation is applied to the afferent synapses of each neurons for a total of 1000 updates. **(b)** The adjustments leads to a smooth convergence of the firing rate ν to the homeostatic target of $\nu_{target} = 10\,\text{kHz}$ (dashed horizontal line). **(c)** On the ASIC, the latter is only reached for low update probabilities p, irrespective of the learning rate λ. Shown is the fraction of neurons for which $\nu \in [6, 14]\,\text{kHz}$. The configuration used throughout the remainder of this chapter is marked by a white star. **(d)** For high K_{in} and hence input dominated networks, the recurrent connections are regulated to low values. In contrast, the disappearing input for low K_{in} is compensated by upregulation of recurrent weights to maintain the target firing rate. **(e)** Across all neurons and experiments, the homeostatic plasticity regulates the firing rates ν to a target of 10 kHz (dashed vertical line). Only for very low K_{in}, networks tend to become unstable.

ASIC is rendered autonomous during homeostatic adaption which in turn translates to a power consumption of only 150 mW and moreover allows to fully exploit the system's speedup factor of 1000 compared to biological time scales. Hence, our emulated SNN in combination with the stochastic homeostatic plasticity is characterized by a high degree of efficiency in terms of power consumption as well as speed.

The stochastic formulation of the homeostatic weight updates allows reaching the target rate for various input strengths. To tie on the results of Zierenberg et al. (2018), the regulation has to work reliably for a range of average indegrees K_{in}, i. e. different input strengths. Indeed, the resulting networks with complex weight distributions (Figure 6.21d) promote a stable firing rate of 10 kHz for the tested K_{in} for most of the analog neurons (Figure 6.21e). It is noteworthy that the range of considered K_{in} values is naturally limited by the dynamic range of the analog neurons as well as the range and impact of the synaptic weights. By fixing all stimulating synapses and only regulating recurrent ones, the maintenance of the target rate for different values of K_{in} shifts the working point of our neuromorphic SNNs from input-driven to highly recurrent dynamics.

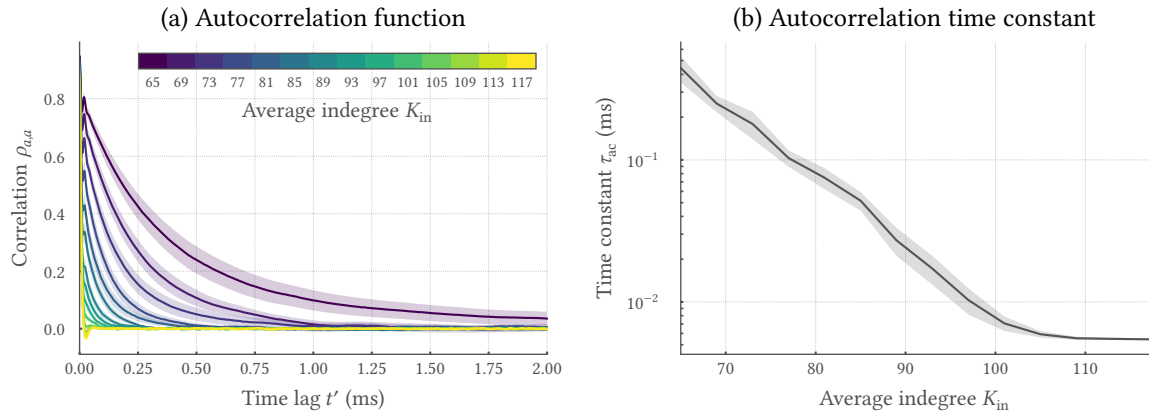


Figure 6.22: The average indegree K_{in} shapes the collective dynamics of the network. The time scale of temporal correlations can be controlled by adjusting K_{in}. **(a)** Low average indegree K_{in} lead to exponentially shaped autocorrelation functions $\rho_{a,a}$ of the population activity $a[t]$ with long time constants τ_{ac}. **(b)** The investigated range of K_{in} values allows to adjust τ_{ac} over almost two orders of magnitude. The time constants were obtained by exponential fits to $\rho_{a,a}$.

6.3.2.1 Control of time scales by the input strength

Our networks indeed show diverse collective dynamics for different input strengths as suggested by the work of Zierenberg et al. (2018). Specifically, the population activity exhibits autocorrelations with time scales controlled by the average indegree K_{in} (Figure 6.22). Therefore, the amount of dynamic memory within the network can be controlled by selecting an appropriate K_{in}. Most notably, we are able to adjust the autocorrelation time scale τ_{ac} over almost two orders of magnitude for the investigated K_{in} range. For the lowermost K_{in}, τ_{ac} exceeds the highest model time constant – the membrane time constant – by a factor of 20. Whether these correlations, however, stem from critical-like dynamics has to be investigated.

The spiking activity of our neuromorphic SNNs suggests bistable instead of critical-like dynamics. When looking at the population activity of networks with different K_{in}, the systems seem to develop bistable dynamics for low values of K_{in} (Figure 6.23a to c). This trend is also reflected in the distribution of the population activity $a[t]$ which shows a bimodal trend for low K_{in} (Figure 6.23d). Hence, we propose that the emerging autocorrelations for low K_{in} stem from the transitioning between these phases of low and high activity. In the following, we consider a HMM to estimate the corresponding phases and the associated transition probabilities.

The autocorrelation time scales determined previously are indeed consolidated by a HMM. Specifically, we fitted a two-state HMM to detect the phases of low and high firing activity within the network responses. In particular for low K_{in}, the states predicted by the model closely mimic the population activity (Figure 6.23a to c). Moreover, the model provides us with the transition probabilities between the two hidden states from which an equivalent autocorrelation time constant τ_{hm} can be estimated. This τ_{hm} closely resembles the autocorrelation time constants obtained by exponential fits to the population activity τ_{ac} (Figure 6.23e). As a result, the measured correlations seem to stem from the transitioning between the two states. Most notably, our SNNs self-organized to this bistable regime and maintained this state with frozen weights. Hence, the phases of low and high activity are not induced by ongoing weight changes or dissipative model components like short-term plasticity (STP).

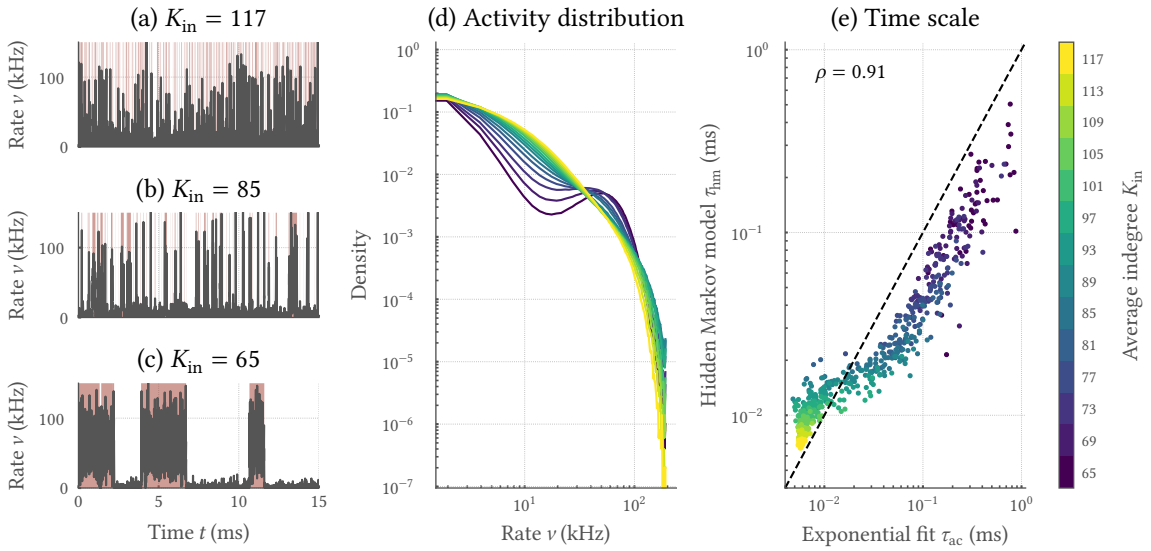


Figure 6.23: Autocorrelations emerge from transitioning between phases of low and high activity. The population rate ν shows increasingly separated phases of high and low activity when decreasing the average indegrees K_{in} from large values ($K_{in} = 117$, **a**), to intermediate ($K_{in} = 85$, **b**) to small values ($K_{in} = 65$, **c**). **(d)** Hence, the distribution of $\nu[t]$ shows a bimodal trend for low values of K_{in} which become increasingly softened for high K_{in}. These phases can be captured by fitting a two component Hidden Markov model to $\nu[t]$ (highlighted by red color). **(e)** The time scale as calculated from the transition probabilities of the Hidden Markov model τ_{hm} accurately predicts the autocorrelation time as estimated from the population activity by exponential fits τ_{ac}. Both of which are highly correlated with a Pearson correlation coefficient of $\rho = 0.91$.

6.3.2.2 Heterogeneous weight distributions promote high time scales

The connectome with diverse weight values induced by self-organization is essential for high time scales at low firing rates on the neuromorphic substrate. As demonstrated previously, the stochastic formulation of homeostatic weight updates leads to diverse weight values for the afferent synapses of each neuron (Figure 6.21d). To investigate the functional role of this diversity, we initialized all recurrent excitatory weights w_{exc} with the mean weight value determined by the homeostasis for the respective K_{in}. Likewise, we clamped all recurrent inhibitory weights w^{inh} to a fixed value which was then swept in a set of experiments. These homogeneous networks closely approximate the homeostatic target rate $r = 10$ kHz at $g = w^{inh}/w^{exc} \approx 1$ (Figure 6.24a). Like for the homeostatically regulated networks, topologies with homogeneous weights lead to non-zero autocorrelation times (Figure 6.24b). However, the networks for which $g = \arg\min_g(r - \nu_{target})$ induce systematically lower autocorrelation times compared to the homeostatically regulated connectomes with diverse weight values (Figure 6.24b). Hence, the stochastic self-organization of our neuromorphic SNNs seems to favor high time scales at low firing rates.

Our networks emit non-pathological firing activity with non-zero autocorrelation times for $g \approx 1$. The theoretical considerations of Brunel (2000) predict stable activity for $g \approx 4$, i. e. for networks with balanced excitation and inhibition. While our networks are comparable to the ones considered by Brunel (2000), there are key differences: Their model relies on delta synapses and maybe more importantly, is driven by purely excitatory input which is not shared among neurons, i. e. each individual neuron is excitatory stimulated by its own spike source. All of these factors have the potential to explain the shift in g as well as the emergence of collective phenomena at the transition.

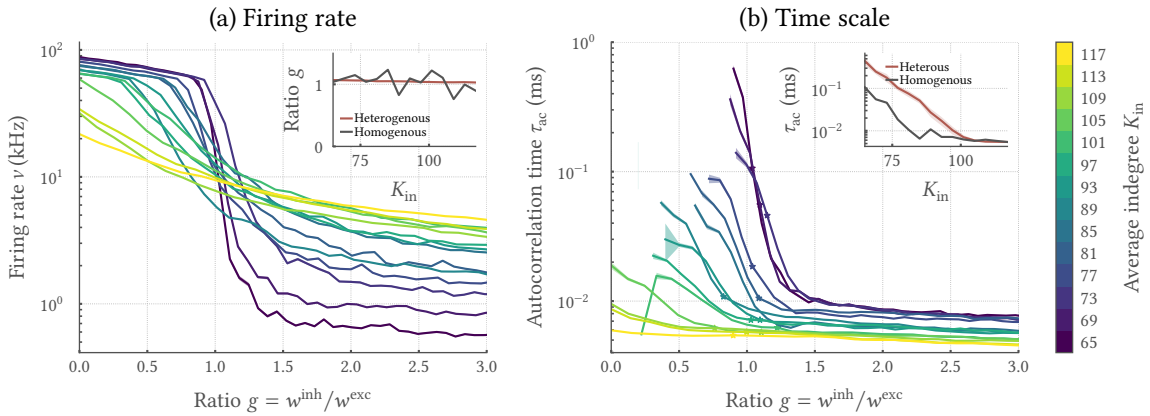


Figure 6.24: Heterogenous weight distributions induced by homeostatic regulation promote high autocorrelation time scales at low firing rates. The impact of a heterogeneous weight distribution is investigated by clamping all excitatory weights w^{exc} of a SNN to the mean excitatory weight determined by homeostatic regulation and sweeping the value of all inhibitory weights. **(a)** At a ratio of inhibitory to excitatory strength of $g = w^{\text{inh}}/w^{\text{exc}} \approx 1$, homogeneous networks approximate the homeostatic target $\nu_{\text{target}} = 10\,\text{kHz}$ with their firing rate ν for all average indegrees K_{in}. The inset of the figure displays the g values for which $\nu \approx \nu_{\text{target}}$ (gray). For comparison, the g of heterogeneous weight distributions induced by homeostatic regulation is superimposed (red). **(b)** Like for homeostatically regulated networks, topologies with homogeneous weights cause non-zero autocorrelation times τ_{ac}. The g values promoting $\nu \approx 10\,\text{kHz}$ are marked by colored stars. The associated values of τ_{ac}, however, are systematically lower (figure inset gray) than the ones estimated from a homeostatically regulated SNN with heterogeneous afferent weight values (figure inset red).

6.3.2.3 Validation of the analog emulation

Our results are validated by simulations of equivalent SNNs. Like for the emulated SNNs, our simulations show collective phenomena for low K_{in} in form of a rising autocorrelation time constant τ_{ac}. These emerging autocorrelations for simulated networks with low K_{in} equally stem from the transitioning between phases of low and high activity. In more detail, the autocorrelation time scale τ_{ac} estimated by exponential fits to the autocorrelation function of the population activity accurately predicts an equivalent time scale τ_{hm} calculated from the transition probabilities of a HMM (Figure 6.25a). Moreover, the simulations allow us to validate the emulation of neuro-synaptic dynamics in general: Both, τ_{ac} and τ_{hm} estimated from the activity of simulated SNNs coincide with their counterparts obtained from the emulated population activity (Figure 6.25a and b). These results even persist when the hardware constraints are relaxed in the software simulations: Neither the limited weight resolution nor the noise impact the collective phenomena for low K_{in} (Kreft, 2021). Summarizing, the simulation results validate the implementation on the analog neuromorphic substrate and underpin the emergence of bistable dynamics in networks with frozen weights.

6.3.2.4 Estimation of autocorrelation times in perturbation experiments

In most biological settings, neural networks are subject to external stimulation. In the following, this situation is modelled by prolonged periods of additional stimulation. In more detail, we increased the rate of the spike sources from h to $h + \Delta h$ for a duration of $T_{\text{stim}} = 1000\,\mu\text{s}$. In contrast to previous experiments, we start by considering the network response to a perturbation in the absence

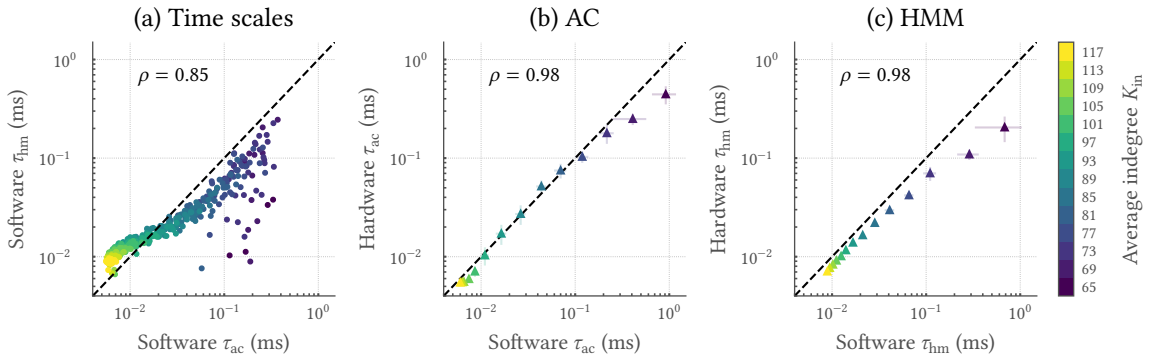


Figure 6.25: Software simulations validate the hardware emulation results. Results of equivalent SNNs obtained in a standard simulation environment validate both the emergence of collective phenomena for low values of the average indegree K_{in} as well as the accurate emulation of neuro-synaptic dynamics by the neuromorphic substrate. **(a)** The emerging autocorrelations for simulated networks with low K_{in} also stem from the transitioning between phases of low and high activity. The autocorrelation time scale τ_{ac} estimated by exponential fits to the autocorrelation function of the population activity accurately predicts an equivalent time scale τ_{hm} calculated from the transition probabilities of a HMM. Both of which are highly correlated with a Pearson correlation coefficient of $\rho = 0.91$. **(b)** Moreover, the estimated τ_{ac} of emulated and simulated networks closely match ($\rho = 0.98$). **(c)** Likewise, the obtained τ_{hm} of both implementations coincide ($\rho = 0.98$).

of background stimulation ($h = 0\,\text{kHz}$). Here, the network starts from a silent state, followed by a quick rise of the network rate ν after stimulus onset. After the perturbation, ν quickly decays back to the absorbing state (Figure 6.26a). Most notably, the initial rise is fastest for networks with high K_{in}, but the steady-state rate r during stimulus presentation is highest for SNNs with low K_{in}. Overall, the SNNs with low K_{in} show steeper response functions when only considering the steady-state value as a function of Δh. Moreover, the time scale of relaxation to the baseline after stimulus offset τ_{dc} is way faster than predicted by the autocorrelation time constant τ_{ac} for all tested values of K_{in} and Δh. However, when increasing the rate of the background noise up to $h = 10\,\text{kHz}$, the relaxation time scale τ_{dc} after a superimposed perturbation with rate Δh accurately predicts the autocorrelation time scale τ_{ac} (Figure 6.26a to c).

The results of these *in-silico* experiments have direct implications on the estimation of autocorrelation time scales. Here, a perturbation should be superimposed to existing background activity to yield an accurate estimate of τ_{ac}. Hence, measurements should be done in the presence of input and therefore highlight the importance of the latter. Moreover, these perturbation experiments prompt an input injection strategy: While many existing reservoir computing frameworks are based on stimulus injection into a previously silent neural network (Maass et al., 2002; Verstraeten et al., 2006), we have shown that for our networks the continuous presence of Poissonian background is essential to maintain input information beyond stimulus offset. In addition, we showcased that the s_+ state can be entered by sufficiently strong perturbations. This is in close accordance with the work of Legenstein & Maass (2007) where the input current was used to maintain a specific network state. The computational impact of the states $s_\pm$, however, remains to be clarified.

6.3.2.5 Classification with bistable dynamics

So far, we have presented the emergence of collective phenomena with decreasing average indegree K_{in}. In the following, we will try to exploit the associated time scales as well as the bistable dy-

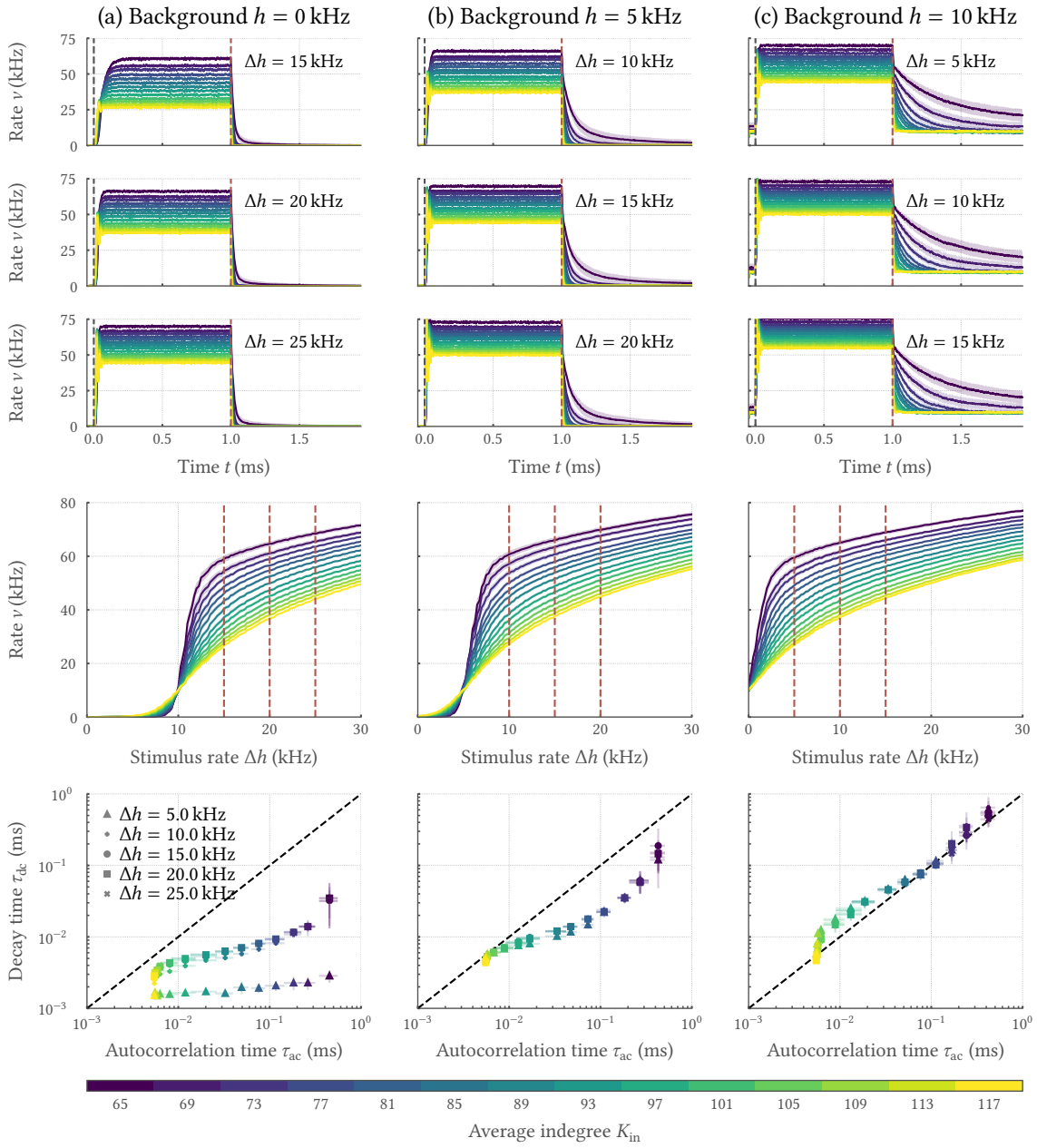


Figure 6.26: Perturbations quickly decay in the absence of background stimulation. Networks were perturbed by increasing the rate of the background spikes sources from h to $h + \Delta h$. **(a)** In absence of background stimulation ($h = 0$ kHz), the firing rate ν quickly rises after perturbation onset (first three rows). The time span of stimulus application is marked by dashed vertical lines. The steady-state ν (marked by red dashed line) non-linearly depends on Δh (fourth row). While a rising Δh leads to an increase of ν in general, especially lower average indegrees K_{in} are characterized by sharp upswings of ν. The horizontal red lines highlight the Δh values shown in the first three rows. The time scale of the decay of ν after stimulus offset τ_{dc} does not match the autocorrelation time τ_{ac} (fifth row). **(b)** With increasing background stimulation ($h = 5$ kHz), perturbations start to reverberate in the SNNs (first three rows). Hence, τ_{dc} approaches τ_{ac} for all values of K_{in} as well as Δh, but remains always smaller. **(c)** Only if the rate of the background stimulation matches the rate used during homeostatic adaptation ($h = 10$ kHz), perturbations reverberate on time scales significantly exceeding all intrinsic time constants of the SNNs (first three rows). Here, the steady-state ν only slightly depends on Δh. In this condition, τ_{dc} closely resembles τ_{ac} for all values of K_{in} as well as Δh.

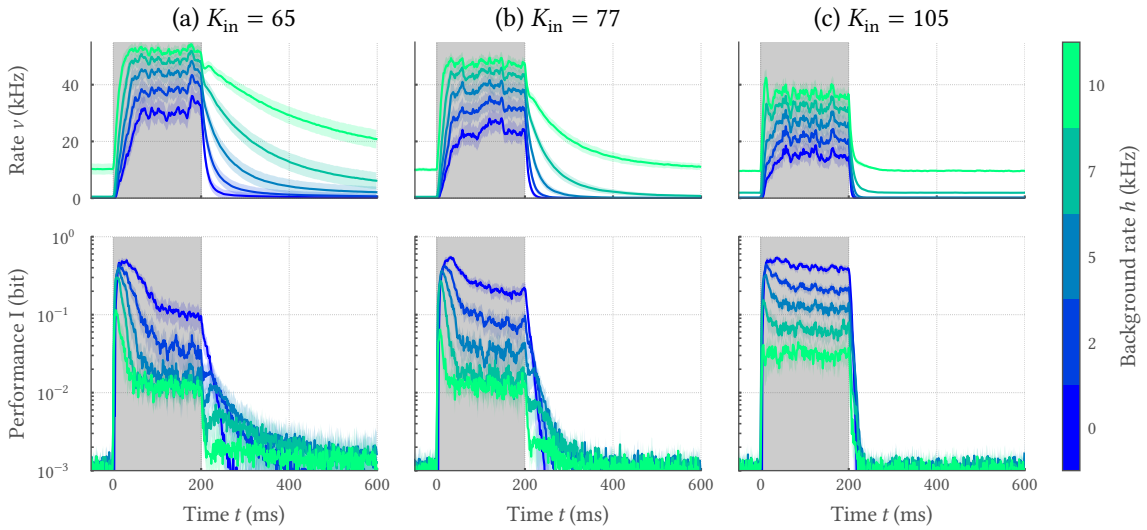


Figure 6.27: Novel task-irrelevant input interferes with task-relevant input. The SNN is used to classify spatio-temporal spike patterns with Poisson statistic by training a linear classifier on the elicited activity of all neurons. Here, the task-memory can be quantified by training and evaluating the classifier based on variable points in time after stimulus onset t. When considering the standard Poissonian background with $h = 10\,\text{kHz}$ and injecting the spike pattern with an additional set of synapses into the network, networks with short intrinsic time scales (high K_{in}) promote the highest performance I during the presentation of the stimulus (gray area). Memory beyond stimulus offset is only provided by networks with long time scales and hence low K_{in}. However, the decay of I after stimulus offset occurs on a significantly shorter time scale than both the decay of the network rate ν as well as the time scale of autocorrelations. By decreasing h, the relevance of the pattern input within the systems response could be increased. Specifically, a decrease of h leads to a monotonic increase of I during stimulus presentation. However, only intermediate values of h promote increased memory time scales while low values of h cause again a decline.

namics for classification and in that process try to approach the question of suitable stimuli as well as stimulation strategies. To that end, we again utilize the framework of reservoir computing to perform information processing with our SNNs. Specifically, we trained a linear readout based on the activity of all neurons within the network when being stimulated by patterns of random spike trains with Poisson statistics. For that purpose, an additional set of synapses was utilized, i. e. each input spike train projected onto a randomly drawn subset of neurons. Here, the task memory can be quantified by training and evaluating the linear readout on distinct time segments t after stimulus onset for various average indegrees K_{in} (Figure 6.11). With this framework, we will show that networks with low intrinsic time scales are favorable for fast inference, whereas SNNs covering long time scales are only desirable when dynamic memory is required. Moreover, we will depict that the task-irrelevant input provided by the continuous background stimulation with rate h seems to interfere with the task-relevant pattern input.

Low intrinsic time scales are particularly favorable for fast inference. In the presence of standard background stimulation with rate $h = 10\,\text{kHz}$, SNNs with high K_{in} and hence short intrinsic time scales promote the highest performance during the presentation of the stimulus (Figure 6.27). In this temporal segment, memory on relatively short time scales seems to be sufficient to lead to increased performance for the task at hand. Furthermore, even networks with low K_{in} and hence long time scales do not lead to significant memory effects for $h = 10\,\text{kHz}$. In more detail, the time scale of the decay of the performance after stimulus offset is neither on the order of the decay of the firing rate ν

(Figure 6.27a) nor comparable to the previously estimated autocorrelation time scales (Figures 6.22 and 6.23).

Novel task-irrelevant input seems to interfere with task-relevant pattern input. The continuous background stimulation with rate h in addition to the actual pattern input constitutes a source of continuous, task-irrelevant information. It is noteworthy that this background stimulation is, however, required to reach and maintain the dynamical state of our SNNs. By reducing h to intermediate values, the classification performance at all times after stimulus onset increases (Figure 6.27). Most notably, the memory time scale of networks with low K_{in} likewise increases. This reduction of h to medium values causes a reduction of the impact of the task-irrelevant background stimulation which especially facilitates the representation of stimulus information beyond stimulus offset. However, a further reduction of h decreases the memory time scale about the injected pattern after stimulus offset (Figure 6.27a), which is in direct accordance with our previous results (Figure 6.26). In contrast, the performance during stimulus presentation still monotonically increases. Here, intermediate values of h seem to build a comprise between the maintenance of activity beyond stimulus offset as well as the negative interference of pattern input and task-irrelevant background stimulation.

The network activity presumably needs to be more self-sustained to promote high classification performance on additional pattern input. The assessment of performance in Figure 6.27 suggests that our networks are still highly input-driven and hence represent input information on relatively short time scales within the network activity. Especially after stimulus offset, the still existing background stimulation is likely to dominate the information contained within the network activity. Consequently, mechanisms allowing for a further reduction of the input strength are likely to facilitate suitable stimulus representations on long time scales within the network activity. The considered reduction of h for the results in Figure 6.27 not only strengthens the impact of the pattern input, but moreover allows the exploitation of bistable dynamics for efficient information processing. While only values of h in the proximity of $h = 10\,\text{kHz}$ – the strength applied during homeostatic adaption – cause the entering of the state of high activity, intermediate stimulation strengths allow to only evoke this state in presence of additional pattern input (Figure 6.26). By this, not only the energetic footprint of our SNNs could be reduced, but moreover, the bandwidth on the neuromorphic system could be relieved. It is noteworthy that, however, high stimulus strengths only increase the likelihood of entering the state of high activity, but in addition, could lead to saturation effects on the analog neuromorphic substrate.

6.4 Discussion

Within this chapter, we considered self-organized SNNs and exploited the resulting collective phenomena for information processing. Most notably, we refrained from detailed and highly specialized training like the one presented in the scope of Chapter 5. Instead, the adaption to task requirements was achieved on an abstract level and did not take specific training samples into account.

The underlying control of time scales in SNNs has been demonstrated for two different types of collective phenomena. Specifically, we considered critical-like phenomena within the first part of this chapter, while networks exhibiting bistable behavior have been utilized in the second part. For both frameworks, an adaption can be achieved by controlling the input strength. It is noteworthy that the number of synapses connected to the external input is only one possibility to control the input strength. Likewise, the rate of the stimulus can be utilized as a control parameter. Indeed,

adjusting the input rate leads to similar results for both types of phenomena and has been shown to allow to even cross the critical point for low input rates (Appendix A.4). Both parameters in conjunction span a 2D landscape of possible configurations which provide a high degree of flexibility when tuning a network to computational requirements. Previous work has come up with analytical solutions for the optimal input strength within mean-field networks (Zierenberg et al., 2018). These results have the potential to guide the design of other networks and tasks as well.

Likewise, inhibition can act as a control parameter. Inhibition does not only shape collective dynamics and causes oscillations (Whittington et al., 2000; Buzsáki & Wang, 2012), but specific ratios of excitation and inhibition are known to cause critical phenomena in neural networks (Poil et al., 2012; Shin & Kim, 2006; Hesse & Gross, 2014; Neto et al., 2017). In close accordance, our networks featured 20 % inhibitory neurons, which were however not required for critical dynamics (Levina et al., 2007; Wilting & Priesemann, 2018). This magnitude of control parameters facilitates fine-grained adjustments even in cases where only a subset of them can be freely configured without perturbing the input coding.

Not only the configuration of a control parameter, but also the interaction with local plasticity allowed the self-organization and the associated shaping of computational properties. Most notably, this robust mechanism does not require fine-tuning of parameters. Within the first part of this chapter, we considered a modified version of STDP which implements only the negative, anticausal arm. When instead relying on full STDP, networks tend to become unstable, despite counteracting homeostasis which has also been observed in the past (Keck et al., 2017). Our plasticity mechanism is, however, similar to STDP, but with unspecific potentiation and only specific depression. Because of this, our rule may be useful for further studies, due to its stabilizing nature and its similarity to STDP. In the second part, we utilized a homeostatic regulation inspired by synaptic scaling found in biological tissue (Turrigiano & Nelson, 2004). While the latter choice allows fine-grained control of the network activity and hence facilitates usability, it lacks spike-timing aware information and hence specific features. Because of this, a combination of both rules presented within the scope of this chapter represents a promising direction for future research.

Robust adaption mechanisms with local plasticity are particularly promising for the tuning of SNNs emulated on analog neuromorphic hardware. The latter exhibit fixed-pattern variations induced by the production process and are subject to temporal noise due to the analog emulation of state variables. Here, plasticity mechanisms have the potential to equalize fixed-pattern variations which are likely to destabilize the network. A comparable approach has been demonstrated in the past for STP (Bill et al., 2010). These approaches could sidestep the role of detailed calibration routines which are especially challenging for large neuromorphic systems.

Irrespective of the compute substrate, critical-like dynamics are assumed to be beneficial for any task processing (Munoz, 2018b; Boedecker et al., 2012). However, we showcased that this assumption has to be rephrased: While criticality indeed maximizes a set of abstract computational properties like autocorrelation time (Figure 6.15b), susceptibility (Figure 6.15d), as well as information-theoretic measures (Figures 6.18 to 6.20) this does not necessarily boost task performance in a system of finite size. Here, we demonstrated that only the complex and memory-intensive tasks profit while simple ones even suffer. For the latter, task-irrelevant information is maintained by long-lasting memory effects at criticality which might erase novel, task-relevant stimuli. As a result, the network dynamics need to be tuned to task requirements, i. e. every task needs its own dynamic state to be solved optimally.

Our understanding of criticality on computation was facilitated by the task-dependent as well as task-independent considerations of performance. In the past, the network state has been characterized by classical measures like AIS, I and TE (Shew & Plenz, 2013; Mediano & Shanahan, 2017) or PID (Tax et al., 2017; Wibral et al., 2017a), but not in combination with task processing. However, only the combination of both allowed us to showcase that information capacity is indeed maximized at a critical point, but not necessarily task performance. Here, a key step to understand task processing was enabled by a direct prediction of memory time scales with a lagged I between stimulus and delayed neuronal activity. This provided us with a relation of task complexity and information-theoretic fingerprint. Because of this, our results have the potential to serve as the basis for well-founded design decisions of future artificial architectures.

Aside from critical phenomena, we analyzed SNNs exhibiting bistable dynamics within the scope of this chapter. Strong fluctuations associated with states of low and high activity have been found in experiments depending on the brain area and experimental details, however only under the influence of anesthetics or in brain slices (Cossart et al., 2003; Jercog et al., 2017; Stern et al., 1997). The associated dynamics have likewise been observed in models of neural networks exposed to STP (Holcman & Tsodyks, 2006; Benita et al., 2012) as well as in networks of adaptive exponential leaky integrate-and-fire (AdEx) neurons (Protachevicz et al., 2019). While the aforementioned models feature dissipative components, our homeostatically regulated SNNs lack thereof and instead show bistable dynamics even with frozen weights. Here, the switching between both states is likely to be driven by statistical variations of the input. The actual source of bistable dynamics is, however, a subject of ongoing research.

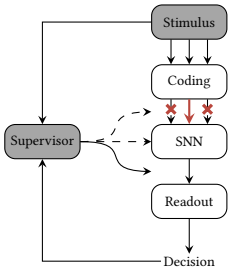
Within all experiments, we considered random networks lacking of spatial structure stimulated by uncorrelated input spike trains. In the context of reservoir computing, nearest neighbor topologies are strongly represented, especially in combination with tasks of natural language processing (Maass et al., 2002; Verstraeten et al., 2005; Legenstein & Maass, 2007). Aside, also ring, lattice, torus as well as dense connectomes have been investigated previously (Dale et al., 2021). Particularly in combination with stimuli exhibiting spatio-temporal structures, these topologies represent a promising direction for future research. Aside from the network topology, the stimulus injection constitutes a key design decision. Often a randomly drawn subset of neurons is used to inject external input into the reservoir network (Maass et al., 2002; Legenstein & Maass, 2007). We shared this idea, albeit, in the context of random network topologies.

All of the aforementioned design decisions need to be tested with regard to their area of application. Within this chapter, we utilized a set of artificial tasks to benchmark our neuromorphic SNNs. Due to their ability to precisely control the complexity and memory requirements, the computation of the n-bit parity as well as the NARMA constitute standard benchmarks within the reservoir computing community (Bertschinger & Natschläger, 2004; Jaeger, 2003; Appeltant et al., 2011). However, their applicability to larger SNNs poses problems: Especially for the computation of the n-bit parity, every single input spike is crucial for the network to make the correct decision. In the context of noise-affected SNNs implemented on analog neuromorphic hardware and also for their biological archetype – the human brain – this constitutes an artificial condition. Moreover, in their original formulation, these tasks lack spatio-temporal structure (cf. Chapter 4). In contrast, real-world benchmark data like the samples constituting the Spiking Heidelberg Digits (SHD) dataset presented in Chapter 4, feature non-random structures, but do not naturally allow to make statements about complexity or the required memory. While a partitioning of existing benchmarks in pairwise classifications according to their complexity is in general conceivable, the design of classes of artificial

spatio-temporal pattern of well-defined complexity directly in the spiking domain remains a major challenge.

Both presented strategies – the STDP-like as well as the homeostatic weight updates – revealed complex computational properties of our networks despite their small system size. A scale-up of these systems would be beneficial, as one would expect even larger processing capabilities. Moreover, larger system sizes promote more self-sustained activity (Kriener et al., 2014; Borges et al., 2020), which in turn allows to further reduce the impact of the Poisson input used for self-organization. This has the potential to induce a better representation of superimposed stimuli within the system's response as well as memory retrievals on even longer time scales. At the same time, this scale-up does not require additional parameter fine-tuning due to the self-organization with local learning rules. By this, the energy efficiency and speed of larger chips or even multi-chip systems can be exhausted with the presented frameworks.

In summary, we developed two distinct SNN models as well as their implementation on two BrainScaleS-2 chips which exhibit emerging collective dynamics within the population activity. Specifically, we showcased emerging autocorrelation times, significantly exceeding the time scale of single-neuron dynamics. The underlying dynamics promoting these time scales, however, seem to stem from different phenomena. Within the first part of this chapter, we found critical-like behavior. With this, we have shown a clear relation between criticality, task performance and information-theoretic fingerprint. Our results contradict the assumption that criticality promotes performance on any task. Although the dynamical state of the network indeed impacts performance evaluated in a reservoir computing framework, only complex tasks profit from critical states, whereas simple ones suffer. The tuning of networks to computational requirements can be achieved by the input strength under homeostatic action. With this, we provide an understanding of how critical phenomena could be exploited in the design of SNNs. Moreover, our results may give an explanation why biological neural networks do not necessarily operate at a critical point, but in its dynamically rich vicinity where they can be easily tuned to task requirements (Wilting et al., 2018; Wilting & Priesemann, 2019). In the second part of this chapter, we showcased emerging autocorrelation times induced by bistable dynamics within SNNs. Like for critical SNNs, the time scale of the system can be controlled via the input strength, although with entirely different collective dynamics. Most notably, the SNNs maintained the transitioning between both states even with frozen weights and in the absence of dissipative model components. Last, we provide a reliable strategy for the measurement of autocorrelation times within perturbation experiments as well as an approach to exploit the emerging phenomena for computation.



7 Feature Selection

The work on weight-driven structural plasticity presented within this chapter has been published in Billaudelle et al. (2021) in close collaboration with Sebastian Billaudelle. In the following, I will closely follow the publication, but with a more detailed description of the implementation. The results on STDP-driven structural plasticity have been conducted in collaboration with Markus Kreft and have already been demonstrated in Kreft (2019).

In the previous chapters, we investigated different strategies to optimize spiking neural networks (SNNs) emulated on analog neuromorphic hardware for information processing by changing the strength of synaptic weights. While we were able to make use of the associated fast and energy-efficient emulation, the neuromorphic devices also imposed limitations. One of them are finite synaptic resources which posed constraints on the fan-in as well as the overall size of the SNNs discussed in Chapter 5. This constraint does not only hold true for the BrainScaleS-2 system, but represents an ubiquitous issue for all neuromorphic substrates and is even inherited from the biological archetype, the human brain. The latter, however, does not implement a static connectome, but is known to constantly rewire its pre- and postsynaptic partners, a process referred to as structural plasticity. Therefore, it might be promising to draw again inspiration from the human brain and port similar mechanisms to neuromorphic hardware to relax the constraint of limited synaptic resources by facilitating information processing in sparse networks. Within this chapter, we discuss an overarched optimization principle by presenting structural plasticity on the mixed-signal neuromorphic chip High Input Count Analog Neural Network with Digital Learning System (HICANN-DLS). In more detail, we describe an efficient on-chip implementation facilitated by synaptic-event filtering (Section 3.1.1) and the embedded plasticity processing unit (PPU) running alongside the analog emulation of neuro-synaptic dynamics (Section 3.1.3). We characterize our concept in a supervised as well as an unsupervised learning scenario (cf. Chapters 5 and 6) with two distinct pruning conditions and demonstrate the stimulus-dependent self-organized formation of structured networks. Furthermore, we evaluate the computational efficiency of our algorithms on HICANN-DLS and showcase the comparable minimal overhead introduced by rewiring. Thereby, we highlight the beneficial role of structural plasticity on neuromorphic hardware, especially in combination with accelerated neuromorphic systems.

7.1 Introduction

Neuromorphic systems aim to transfer the efficiency of biological brains to silicon circuits thereby implementing novel computing paradigms (Section 2.2). Despite the associated advantages, these devices often also inherit limitations from their biological archetypes. With the diverse range of proposed architectures in mind (Schuman et al., 2017), the nature of the limitations as well as their

impact on the overall performance when it comes to benchmarking crucially depends on the considered device. However, all physical systems have in common that they need to operate on finite resources provided by the substrate.

One primary example of such a limitation is the synaptic fan-in. This constraint holds true for both digital as well as analog or mixed-signal implementations, albeit trade-offs are possible for various platforms. Digital systems often feature fast on-chip memory whose size poses limits on the number of synapses (Akopyan et al., 2015; Frenkel et al., 2018; Davies et al., 2018). Since these systems often use time-multiplexed update logic, they can be designed to soften the constraint of a limited fan-in by increasing the allocated memory size, however, at the cost of prolonged simulation times, a reduced precision or an overall lowered network size (Furber et al., 2013). Analog or mixed-signal devices, in contrast, implement their synapses physically and, hence, do not allow the described trade-offs. Instead, they provide a fixed number of synaptic circuits for the definition of networks (Moradi et al., 2018; Schemmel et al., 2007, 2010; Friedmann et al., 2016). Nevertheless, some of these systems provide multi-compartment neuron models which on the one hand allow the emulation of soma and dendrites, but on the other hand facilitate an extension of the fan-in by merging multiple neuron circuits – each with its associated synaptic resources – to larger logical entities (Schemmel et al., 2010; Aamir et al., 2018). Exemplary networks falling back on this strategy have been presented in the context of Chapter 5 where the fan-in had to adhere to already downscaled MNIST images. However, this comes at the cost of a reduced number of neurons available for the actual network implementation.

The investigation of current neuromorphic platforms underpins that the limited availability of synaptic resources indeed represents an ubiquitous issue. Since these devices were originally inspired by the working principle of the human brain for which similar constraints hold true, it is worth taking another look at their role model. Most notably, biological neural networks do not implement static networks, but are subject to ongoing reconfigurations by the removal and creation of synapses (Grutzendler et al., 2002; Zuo et al., 2005; Bhatt et al., 2009; Holtmaat & Svoboda, 2009; Xu et al., 2009). Nevertheless, they exhibit at all times a high degree of sparsity even at the local scale (Braitenberg & Schüz, 2013; Abeles, 1991; Hellwig, 2000). In the mammalian cortex, most of the excitatory synapses to Pyramidal neurons target dendritic spines (Yuste, 2011). As outlined in detail in Section 2.1.4, the head size of these dendritic spine varies dramatically (Trachtenberg et al., 2002) and correlates with the probability of their removal (Holtmaat et al., 2005, 2006). The lifetime of a connection is assumed to be related to the synaptic efficacy because the spine volume depends on the amplitude of the respective synaptic currents (Matsuzaki et al., 2001). While the magnitude of synaptic turnover decreases in the adult brain, a relatively small but significant fraction of synapses continuously re-adapts neural tissue to environmental stimuli (Holtmaat et al., 2005; Trachtenberg et al., 2002; Grutzendler et al., 2002). Even in the adulthood, this turnover can be reactivated after lesions or in the process of recovering from stroke (Yamahachi et al., 2009; Keck et al., 2011; Brown et al., 2007). Moreover, the process of structural plasticity is thought to be key for effective information storage, both in terms of space and energy requirements (Knoblauch et al., 2010; Chklovskii et al., 2004; Poirazi & Mel, 2001). Hence, ongoing structural reconfigurations are likely to allow sparse biological networks to maintain function at a low spatial and energetic footprint even in the presence of changing environments.

In the field of machine learning, synaptic rewiring has already been visited in the context of sparse networks. Most contemporary standard tasks like the processing of image or movie data come with high dimensionality at the stimulus level: Already for small size images like the ones provided with

the MNIST dataset (LeCun et al., 1998) a total of 28×28 pixels with a precision of 8 bit need to be processed which already challenges the synaptic fan-in of many neuromorphic implementations (Cramer et al., 2021; Göltz et al., 2021). This problem gets even more severe for the benchmark datasets CIFAR-10 and CIFAR-100 (Krizhevsky et al., 2009) where not only the amount of pixels is increased to 32×32, but also colors are included: Here, every pixel comes with three color channels resulting in a total of 3072 different inputs each with 8 bit precision. In contrast, applications like natural language processing rely on specific preprocessing steps to extract relevant features (Cramer et al., 2020b; Warden, 2018; Köhn et al., 2016; Zohar et al., 2018; Leonard & Doddington, 1991). These transformations further enlarge the input dimensionality as an additional dimension – the frequency – is introduced (Kumar, 1998). However, in most scenarios, the input signal is still sparse and not all stimulus features are relevant for task processing. Aside from the input, Denil et al. (2013) found that not all connections of multi-layer networks are required to remain functional for common machine learning datasets. Quite the opposite, these networks express many redundant connections during training. Therefore, it is highly desirable to exploit overarched optimization schemes to not even form redundant or non-informative connections as the maintenance of synapses as well as the optimization of the associated weights requires computational resources. As a result, the connectivity pattern is no longer dense, but exhibits sparsity. Hence, the community has come up with new training schemes which directly incorporate sparsity constraints into the training process (Bellec et al., 2017; Wen et al., 2016). By this, network graphs can often be dramatically compressed without a significant loss in performance. To also bring this effort to the field of neuromorphic computing with SNNs, the ability to implement structural plasticity as well as arbitrary sparse networks should be deeply anchored into the design of the associated devices.

Within this chapter, we present two structural plasticity mechanisms and their on-chip implementation on the mixed-signal HICANN-DLS chip (cf. Section 3.1). Our strategies directly draw on the fact that the connectome on HICANN-DLS is partially and most notably locally resolved within the synapses (cf. Section 3.1.1). This architecture promotes both the mapping of sparse networks as well as an on-chip implementation of structural plasticity executed by the embedded PPU (cf. Section 3.1.3). Our approaches keep the mapped connectomes sparse at all times at a constant fan-in. With our implementations of structural plasticity, we demonstrate the self-organized formation of receptive fields. Moreover, we showcase the ability to dynamically readapt these receptive fields to changing environments. Albeit of the additional structural updates by local-learning, we showcase the minimal overhead introduced by readapting the network graph compared to e. g. the computation of synaptic weights.

7.2 Methods

Within this method section, we highlight two different algorithms for the pruning and reassignment of synapses accompanied by their implementation on HICANN-DLS. As both strategies exploit the partial in-synapse resolution of the connectome enabled by synaptic event filtering on BrainScaleS-2 (Section 3.1.1), we start by shortly recapping the event-routing in the analog neuromorphic core of HICANN-DLS before moving on to our structural plasticity implementations. In that process, we introduce a general nomenclature used throughout this chapter.

Structural reconfigurations can be implemented on HICANN-DLS by synapse-local operations. The connectivity on HICANN-DLS is physically implemented by the synapse array (Figure 7.1).

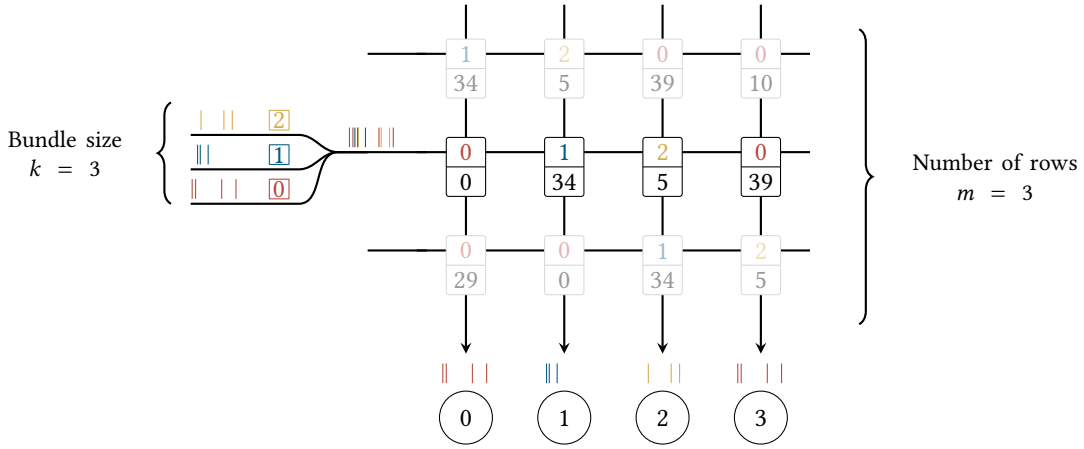


Figure 7.1: Synaptic event filtering enables the efficient implementation of structural plasticity. The connectivity on HICANN-DLS is physically implemented by a 2D array of synapses. Events enter the rows constituting this array from the left and are eventually forwarded by synapses to the neurons located at the bottom (circles). Multiple events originating from different presynaptic partners can be injected into a single row of synapses (color-coded). Each spike event carries a unique address that denotes its presynaptic origin. In addition to a synaptic weight, each synapse holds an address label in its local SRAM (color-coded). Based on this address, each synapse filters incoming events and only forwards them to its home neuron in case the event address matches the locally stored address label. Hence, structural reconfigurations can be efficiently implemented by manipulating the synapse-local address labels which can be achieved via the on-chip PPU. Here, the number of distinct spike trains injected into a single synaptic row is referred to as *bundle size k*, and the number of utilized synaptic rows is termed *m*. Figure adapted from Billaudelle et al. (2021).

Synapse drivers forward spike events to whole rows of synapses constituting this 2D array. In more detail, each event injected into a single row carries a 6 bit event address denoting its presynaptic origin. Likewise, each synapse holds a 6 bit address label alongside a 6 bit weight in its local static random-access memory (SRAM). Based on this address label, synapses filter incoming spike events and only forward them to their home neuron in case the event address matches the aforementioned address label. In general, multiple events originating from different presynaptic sources – i. e. with different event addresses – can be injected into the very same synaptic row. Here, we refer to the number of distinct presynaptic sources per synaptic row as the *bundle size k*. It is noteworthy that each synapse can only transmit the spike events of a single presynaptic partner out of this bundle to its home neuron. The number of utilized synaptic rows is termed *m*. By exploiting this synapse-local event filtering, structural reconfigurations can be implemented by dynamically manipulating the address label stored in the synaptic SRAM which effectively eliminates the previous connection and at the same time creates a new one (Figure 7.1). Hence a re-write of the synaptic SRAM is required for a single structural update which can be achieved in parallel by the single instruction, multiple data (SIMD) vector unit of the PPU (cf. Section 3.1.3). The locality and parallelism render our strategy very efficient compared to other implementations. In particular, no global access patterns – like the reordering of routing tables – are entailed which would otherwise dramatically increase computational complexity (Liu et al., 2018).

Within this chapter, we detail two distinct algorithms for the pruning and reassignment of synapses as well as their accompanying weight dynamics. First, we consider a weight-driven approach detailed in Section 7.2.1 and, second, a STDP-driven algorithm is presented in Section 7.2.2. While our weight-driven approach assumes that informative connections are characterized by a high weight,

Table 7.1: Overview of model and stimulus parameters used for the weight-driven structural plasticity implementation. The given LIF neuron parameters correspond to the target values of the calibration routines. All time constants are given in wall clock time. The errors indicate the standard deviation.

Stage	Parameter	Symbol	Value
Neuro-synaptic dynamics	Threshold potential	u_{thresh}	1000 mV
	Leak potential	u_{leak}	700 mV
	Reset potential	u_{reset}	400 mV
	Membrane capacitance	C_m	(2.38 ± 0.02) pF
	Membrane time constant	τ_{mem}	6.0 µs
	Refractory period	τ_{ref}	4.0 µs
	Excitatory synaptic time constant	$\tau_{\text{syn}}^{\text{exc}}$	4.5 µs
	Inhibitory synaptic time constant	$\tau_{\text{syn}}^{\text{inh}}$	4.5 µs
	Synaptic delay	d_{syn}	(1.9 ± 0.1) µs
	Weight scaling factor	γ	(8.96 ± 0.13) µA
Network	Number of label neurons	N	3
	Maximum input rate	$\hat{\nu}$	50 kHz
	Initial synaptic weight	w_{ij}^{init}	0
	Initial address label	a_{ij}^{init}	$\sim \text{unif}(0, k)$
	Bundle size	k	8
	Number of synase rows	m	6
Plasticity dynamics	Causal STDP time constant	τ_+	8.0 µs
	Causal STDP amplitude	η_+	0.08
	Pruning threshold	θ_{w}	10/128
	Causal STDP capping	f_{max}	9/128
	Correlation scaling	λ_{stdp}	30/128
	Regularization scaling	λ_{reg}	60/128
	Range of random variable	n_{amp}	4/128
	Plasticity update period	T	24 ms
	Structural plasticity update period	T^{struct}	120 ms
	Experiment duration	T^{exp}	4.8 s
Input	Number of receptors	n	48
	Kernel support	λ	$1.5/n$
	Maximum rate	$\hat{\nu}$	50 kHz
	Teacher rate	ν_{teach}	35 kHz
	Stimulus duration	T^{stim}	200 µs

the STDP-driven algorithm is tailored to input patterns exhibiting spatio-temporal correlation like auditory stimuli (Appendix A.1).

7.2.1 Weight-driven structural plasticity

Our first approach is motivated by two well-established biological findings, namely that first a high weight is indicative for informative synapses and that second weak synapses are more likely to be pruned (Holtmaat et al., 2005, 2006; Matsuzaki et al., 2001). While the first fact is incorporated by Hebbian learning in combination with slow forgetting, the latter is implemented by the weight-dependent pruning and reassignment of synaptic connections. This allows networks to potentially improve their performance on a benchmark task while simultaneously maintaining sparse connectivity at a limited and, more importantly, constant fan-in. In the following, we start by highlighting the classification task and the network topology as well as the plasticity algorithms and their implementation on HICANN-DLS. We end with a description of the experimental procedure and the network initialization. All model parameters are summarized in Table 7.1.

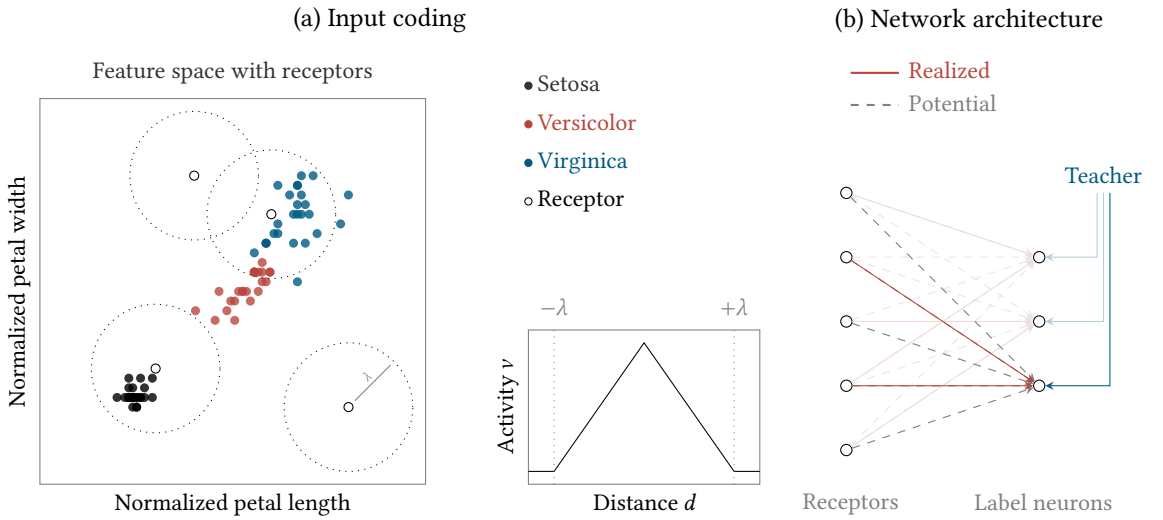


Figure 7.2: Sparse network architecture and input encoding of the Iris dataset. **(a)** The receptors are uniformly distributed on the two-dimensional feature space, which is spanned by the petal widths and lengths of Iris flowers belonging to the three classes *setosa*, *versicolor*, and *virginica*. A receptor's activity is calculated from its Euclidean distance to a data point according to a triangular kernel with radius λ. **(b)** The two-layer network consists of a group of receptors and a label population. One teacher per label neuron ensures excitation of the correct labels during learning. The inputs project onto the label layer with a *potential* all-to-all connectivity (gray), but only a subset of synapses is realized (red). Figure and caption taken from Billaudelle et al. (2021).

Task: We benchmarked our structural plasticity implementation by classifying the Iris dataset which contains petal and sepal length and width information of three different Iris flower species (Fisher, 1936). For validation purposes, we randomly partitioned the 150 samples into 120 training and 30 test data points. To accommodate the data to a manageable fan-in and to finally convert the width and length information of Iris flowers to spike trains, we reduced the four-dimensional dataset to only two distinct dimensions. In that process, we only selected the petal widths and lengths renormalized to values between 0.2 and 0.8 We uniformly placed n uniformly distributed virtual receptors in the resulting 2D feature plane (Figure 7.2a). Each of which emitted Poisson-distributed spike trains with an instantaneous rate determined by the Euclidean distance d_{ij} between the receptor i and the presented sample j. In more detail, the firing rate of receptor i in response to sample j was assigned according to a triangular kernel $\nu_{ij}(d_{ij}) = \hat{\nu} \cdot \max(0, 1 - d_{ij}/\lambda)$, with a maximal rate of $\hat{\nu} = 50\,\text{kHz}$ (Figure 7.2a). This translates to 50 Hz in the biological time domain when taking the system's acceleration into account. To adequately cover the feature plane for various receptor counts n, we scaled the radius λ inversely with $\sqrt{n}$. By this method, we generated n spike trains with a duration of $T^{\text{stim}} = 200\,\mu\text{s}$ for every sample of the Iris dataset.

Network: To classify the Iris flowers, we instantiated a label layer composed of $N = 3$ neurons on HICANN-DLS. The latter received its input spike trains from a software-based receptor layer composed of the aforementioned n virtual receptors. The routing on HICANN-DLS was realized such, that in general each label neuron could be stimulated by each receptor. However, the connectome was kept sparse at all times by ensuring that n always exceeds the number of realized synapses. Here, we considered a supervised learning framework, i. e. every label neuron was in addition stim-

ulated by a teacher spike train with Poisson-distributed spike trains of rate $\nu_{\text{teach}} = 35\,\text{kHz}$ via static synapses with frozen weights. The latter connections ensured activation of the correct label neuron whenever the presented sample corresponded to its respective class.

For all our experiments, the number of expressed synaptic connections to the readout unit, i. e. the number of utilized synaptic rows m, was always smaller or equal than the number of receptors n (Figure 7.1). To still inject all n generated spike trains into the chip, we randomly grouped them into m disjoint bundles of size k. Each of these bundles was injected into a single synapse row of HICANN-DLS (Figure 7.1). In order to uniquely identify the presynaptic origin, the spike trains constituting each bundle were assigned a unique event label in the range $[0, k)$. Similarly, the three distinct teaching spike trains were injected via a single but separate row of synapses with event labels from zero to two. The assignment of a readout unit to a stimulus class was hence determined by the configured address label within the synaptic row in which the teacher spike trains were injected. By simply configuring each of the three address labels within this row multiple times, the very same network can be instantiated repeatedly on a single HICANN-DLS chip.

The sparsity of our SNNs can be controlled by the number of utilized synaptic rows m and the bundle size k. Here, we define the sparsity of our networks as the ratio between the number of unrealized synapses $n - m$ and the count of potential synapses n, i. e. $1 - 1/k = 1 - m/n$. Hence, our implementation features two independent parameters to control the network connectivity (Figure 7.7). We exposed our SNNs to two distinct scenarios: First, the number of receptors n was increased for a fixed synapse count m which results in an enlarged bundle size k and thus increased sparsity. Second, the synapse count m was reduced while simultaneously keeping the sparsity constant $1 - 1/k$ which in turn leads to a reduced number of receptors n.

The described assignment of spike trains to synaptic rows imposes constraints on the connectome. Most notably, each label neuron is only able to connect to a single receptor within each bundle. If, however, two informative receptors by chance reside within the very same bundle, only one of them can be selected at a given point in time. Moreover, multiple projections from the same afferent to a single postsynaptic neuron are not possible. The latter is especially important in combination with a limited weight range. It is noteworthy that both of these constraints can be relaxed by a redundant injection of spike trains, i. e. by injecting the very same spike train into multiple synaptic rows.

Plasticity: For our weight-driven approach, a synapse's eligibility for pruning was determined by a threshold criterion centered around the value of its weight w_{ij}. In case w_{ij} felt below a threshold θ_{w}, the corresponding synapse was removed. Consequently, a new synaptic connection was randomly drawn out of the bundle of potential presynaptic partners of size k and was inserted in its place. By this substitution strategy, the synaptic indegree remained conserved. With regard to the BrainScaleS-2 architecture this strategy was implemented by dynamically rewriting the address label a_{ij} stored in the synapse-local SRAM according to:

$$a_{ij}(t + T_{\text{struc}}) = \begin{cases} \text{unif}(0, k-1), & \text{if } w_{ij}(t) < \theta_{\text{w}} \\ a_{ij}(t), & \text{otherwise}\,. \end{cases} \tag{7.1}$$

where the bundle size k determines the range of the drawn random numbers and in turn the amount of considered presynaptic partners. Each newly created synapse was then initialized with a low weight w_{init}.

The synaptic reconfiguration strategy described above is centered around the assumption that informative synapses are characterized by a high weight. Indeed, this requirement is met by a wide range of learning rules (Zenke & Ganguli, 2018b; Neftci et al., 2019a; Göltz et al., 2021; Urbanczik & Senn, 2014; Bellec et al., 2020). Within the scope of weight-driven structural reconfigurations, we drew on Hebbian weight dynamics based on the spike-timing dependent plasticity (STDP) measurements of HICANN-DLS which facilitated a fully local implementation of weight updates on the PPU. Moreover, the latter can be extended to yield more complex learning rules (Frémaux & Gerstner, 2016; Neftci et al., 2014). For our experiments, we calculated synaptic weight updates according to $w_{ij}(t+T) = w_{ij}(t) + \Delta w_{ij}$ where Δw_{ij} was determined by the following three terms:

$$\Delta w_{ij} = \underbrace{\lambda_{\text{stdp}} \cdot \min\left[f_{\max}, f_+(t_i^k, t_j^l, t, t+T)\right]}_{\text{specific potentiation}} - \underbrace{\lambda_{\text{reg}} \cdot \nu_i(t, t+T) w_{ij}(t)}_{\text{regularization}} + \underbrace{\eta_{ij}(t)}_{\substack{\text{random}\\ \text{walk}}} . \tag{7.2}$$

Each of the first two products was scaled by a positive factor λ_{stdp} or λ_{reg}, respectively. The first term implemented the causal branch of STDP based on the pre- and postsynaptic spike times, t_i^k and t_j^l in the time interval $(t, t+T]$ according to Equation (3.1), but cut off at a maximum value $f_{\max}$. The second term yielded slow forgetting by an exponential weight decay and at the same time homeostatic regulation based on the number of spikes emitted by neuron i in the interval $[t, t+T]$, $\nu_i(t, t+T)$. Moreover, it introduced competition between all afferent synapses of neuron i. In accordance to the work of Kappel et al. (2015), the last contribution $\eta_{ij}(t)$ built an unbiased random walk and hence led to exploration:

$$\eta_{ij} \sim \text{unif}\left(-\eta_{\text{amp}}, \eta_{\text{amp}}\right) , \tag{7.3}$$

where η_{amp} specifies the range of the random numbers. Especially on HICANN-DLS, the latter term was key to overcome local minima induced by fixed-pattern variations and the integer arithmetics of the SIMD vector unit.

All components of the weight dynamics target specialized hardware components and can be efficiently mapped to the BrainScaleS-2 architecture by drawing on the SIMD vector unit of the PPU. To that end, we utilized the update loop shown in Listing 1 to control the experiment flow and iterate over the entire synapse array in a row-wise manner. More specifically, this loop triggered the respective weight as well as structural updates with different frequencies: While synaptic weights were adjusted with a period T, the pruning condition was evaluated at a rate $T_{\text{struc}} = 5 \cdot T$, i. e. every fifth update carried out by the PPU adapted the connectome. Hence, the pruning process took place on a slower time scale than both the network dynamics as well as the weight changes, giving the synaptic weights time to develop over multiple iterations. Within the update loop we applied the kernels shown in Listings 5 and 6 implementing structural reconfigurations as well as the weight dynamics on the SIMD vector unit.

Structural reconfigurations for a slice of synapses were implemented by the kernel code shown in Listing 5. To that end, we first loaded a slice of synaptic weights and address labels into the registers `weights` and `addresses` of the SIMD vector unit (lines 2 and 3). Likewise, pseudo-random numbers were moved into the register `rands` (line 5). These uniformly distributed integer random numbers – required to update the address label – were drawn on the general-purpose part of the PPU by a *xorshift* algorithm (Marsaglia et al., 2003). The resulting 32 bit values were masked to cover the range $[0, k)$ prior to the load instruction. For most of the experiments presented within this chapter, we considered 3 bit random numbers and therefore visited $k = 8$ potential presynaptic partners

```
1  ; Load weights and addresses labels
2  inx weights, %[weight_base], %[offset]
3  inx addresses, %[address_base], %[offset]
4  ; Load uniformly distributed random numbers with range  [0,k-1)
5  lax rands, 0, %[rand_offset]
6  ; Evaluate pruning condition based on current values of weights
7  subbfs conds, weights, %[threshold],
8  compareb conds
9  select addresses, addresses, rands, LT
10 ; Save address labels
11 outx addresses, %[address_base], %[offset]
12 ; Reset weights of pruned synapses
13 compareb conds
14 select weights, weights, %[w_init], LT
15 ; Save weights
16 outx weights, %[weight_base], %[offset]
```

Listing 5: Kernel code for weight-driven structural reconfigurations. After loading the synaptic weights and address labels of a slice of synapses from the synaptic SRAM, the pruning condition is evaluated by a comparison of the weight values to a fixed threshold. In case the weights fall below this threshold, new randomly drawn address labels are assigned to the corresponding synapses thereby allocating new presynaptic partners. Further, the weights of the associated synapses are reset to a small value. Note that the random numbers are generated on the general-purpose part of the PPU and are then loaded to the SIMD vector unit. The range of these numbers determines the amount of visited presynaptic partners and is chosen to coincide with the bundle size k. Shown is the kernel code using NASM syntax.

per row of synapses. To ultimately evaluate the pruning condition, we first subtracted the pruning threshold in the register `threshold` from the current weights hold in `weights` and stored the result thereof in the register `conds` (line 7). These values in `conds` served as the basis for a subsequent compare operation (lines 8 and 9): For all elements of `conds` which were greater or equal to zero, the respective elements in `addresses` remained unchanged, i. e. the synaptic connections persisted. In the opposite case, the connections were pruned by updating the values in `addresses` to the new randomly drawn values in `rands`. Afterwards, the address labels were written back to the synaptic SRAM (line 11). An update was finalized by resetting the synaptic weights of pruned synapses. This was achieved by a second compare operation based on the previously calculated values in `conds` (lines 13 and 14): The old weight values were kept for all elements in `weights` for which the respective entry in `conds` was greater or equal to zero. Otherwise the corresponding elements in `weights` were assigned to the initial weight hold in `w_init`. Afterwards, the values in `weights` were written back to the respective slice of the synapse array (line 16). Note that the bit-shift usually carried out prior to calculations involving the synaptic weights was omitted here, as the saturation arithmetic did not matter for the evaluation of the pruning condition.

The STDP-derived term in Equation (7.2) was implemented by the kernel code in Listing 6a. The causal spike time correlations were measured in the analog synapse-local circuits and then digitized using the column-parallel analog-to-digital converter (CADC). A slice of these observables was subsequently loaded into the register `ac_meas` and aligned to the arithmetic of the SIMD vector unit by the code shown in Listing 1. Subsequently, the correlation sensors were reset. Thereafter, the values

(a) STDP kernel

```
1 ; calculate and apply update
2 subbfs thresholds, ac_meas, %[f_max]
3 compareb thresholds
4 select dws, %[f_max], ac_meas, LT
5 mulbfs dws, dws, %[lambda_stdp]
6 addbfs weights, weights, dws
```

(b) Homeostatic kernel

```
1 ; load spike counts
2 ; from general-purpose part
3 lax rate, 0, %[rate_offset]
4 ; calculate and apply weight update
5 mulbfs dws, weights, %[lambda_reg],
6 mulbfs dws, dws, rate
7 subbfs weights, weights, dws
```

(c) Random walk kernel

```
1 ; load random numbers
2 ; from general-purpose part
3 lax rands, 0, %[rand_offset]
4 ; calculate positive and negative updates
5 ; seperately omitting zero updates
6 addbfs weights_neg, weights, rands
7 subbfs weights_neg, weights_neg, %[bias],
8 addbfs weights_pos, weights_neg, %[one],
9 ; check if random update would be greater
10 ; or equal zero to restore balance
11 subbfs tmps, rands, %[bias],
12 addbfs tmps, tmps, %[one],
13 compareb tmps
14 ; select either positive or negative
15 ; weight update depending on sign
16 select weights, weights_pos, weights_neg, LT
```

Listing 6: Kernel codes for the unsupervised updates of synaptic weights accompanying weight-driven structural changes. (a) A modified STDP kernel constitutes the driving force for the weight dynamics accompanying our weight-driven structural plasticity implementation. The CADC is utilized to digitize the measurements of a slice of correlation sensors. In turn, the measurements are capped and scaled to result in the weight updates which are then applied to the old weight values. **(b)** The regularization term relies on both synaptic weights and the firing rates. The firing rates are read from the on-chip bus by the general-purpose part and then loaded into the SIMD vector unit. The product of a scaling factor, the weights and the firing rates constitutes the weight update which is subtracted from the current weight value. **(c)** Exploration is implemented by a random walk. The random numbers are drawn on the general-purpose part of the PPU and are subsequently loaded into the SIMD vector unit. To implement random updates with a mean of zero, positive and negative updates are treated separately. A compare operation is utilized to select the corresponding updates. Shown is the kernel code using NASM syntax.

in `ac_meas` were scaled by the factor in `lambda_stdp` resulting in the temporary weight updates `dws` (line 2). The capping of the result was implemented by a single subtraction. To that end, the cap value in `f_max` was subtracted from `ac_meas` and the result thereof was assigned to `thresholds` which was used for a subsequent compare operation (lines 2 and 3). For all elements in `ac_meas` for which the corresponding entries in `thresholds` were greater or equal to zero, the measurements in `ac_meas` were capped and hence the corresponding weight update `dws` was set to `f_max` (line 4). Otherwise, the actual measurements `ac_meas` were kept. Next, the calculated updates were scaled by the positive factor `lambda_stdp` (line 5). Ultimately, the weight updates `dws` were added to the old weight values `weights` (line 6) and the result thereof was written back to the synaptic SRAM by the kernel code shown in Listing 1. The capping of correlation measurements allowed us to reduce potential imbalances introduced by fixed-pattern deviations of the associated analog sensors. With regard to Figure 3.4, it becomes evident that some of these sensors systematically yield higher measurements which may result in self-amplification of the corresponding synaptic weights by increasingly dominated synchronization of the associated pre- and postsynaptic firing.

It is noteworthy that reducing the parameter λ_{stdp} could likewise counteract this self-amplification. However, the associated scaling of the STDP measurements could impact convergence due to the limited precision fixed-point calculations carried out by the PPU.

Listing 6b shows the kernel utilized to implement the second term in Equation (7.2). This homeostatic component required access to the postsynaptic spike counts. Since these observables are only available from the neuronal spike counters via the on-chip configuration bus, they were first accessed by the general-purpose part of the PPU and in turn loaded into the register `rate` of the SIMD vector unit (line 3). Subsequently, the multiplication of the synaptic weights hold in `weights` with the constant scaling factor in `lambda_reg` was carried out and the result thereof was assigned to `dws` (line 5). In direct succession, `dws` was multiplied by the rates in `rates` (line 6). Finally, the resulting updates `dws` were substrated from the current weight values `weights` (line 7) prior to writing back the result to the synapse array by the code shown in Listing 1.

Finally, the stochastic term in Equation (7.2) was implemented by the kernel in Listing 6c. Like for the structural plasticity updates, the stochasticity required for the last term was provided by a *xorshift* algorithm implemented on the general-purpose part of the PPU. Subsequently, the drawn 32 bit integer random numbers were restricted to only 3 bit and then loaded to the SIMD vector unit into the register `rands` (line 3). Hence, the random numbers covered the range $[0, 6)$. Since the associated kernel had to implement a balanced random walk with $\langle \eta_{ij} \rangle = 0$ to not induce a bias in the weight updates, the positive and negative weight updates were treated separately to guarantee a symmetrical distribution. To that end, we converted the random values to the range $[-3, 3]$ by omitting the value zero. For the negative updates, we first added the random numbers in `rands` to the current weight values in `weights` resulting in `weights_neg` (line 6). Next, the values in `bias` were substrated from the results in `weights_neg` (line 7). For the 3 bit random numbers, this bias was set to four. The positive updates were simply obtained by adding `ones` to the negative updates `weights_neg` to avoid a weight update of zero (line 8). Last, the correct update had to be determined. To that end, we utilized a compare operation to determine if the random update would be greater or smaller than zero. Hence, we first substracted the values in `biases` from the random values in `rands` (line 11). To implement a less or equal zero operation, we subsequently added a one to the results stored in `tmps` (line 12). Afterwards, a compare operation based on the results in `tmps` was utilized to select the correct update (line 13): In case these results exceeded zero, we assigned the positive update in `weights_pos` to `weights`, otherwise, we applied the negative update `weights_neg` (line 16). The random walk kernel was then finalized by writing back `weights` to the synaptic SRAM with the kernel highlighted in Listing 1.

Initialization: At the start of each experiment, we initialized the synaptic weights to zero except for the ones corresponding to the teacher synapses. The latter were chosen to ensure activation of the correct label neuron in case a sample of the corresponding class was injected. Accordingly, these synapses were initialized with maximal weight. Initially, the address labels were configured randomly according to a uniform distribution covering the range $[0, k)$. Afterwards, we stimulated the network with the randomly selected training samples drawn from the Iris dataset and an additional teacher spike train. During this phase, the synaptic weights as well as the address labels freely evolved according to Equations (7.1) and (7.2) for a total of 200 training epochs. Simultaneously, we read back the address labels as well the synaptic weights for further evaluation after each training epoch, i. e. after each presentation of the full dataset. To evaluate the test performance, we switched

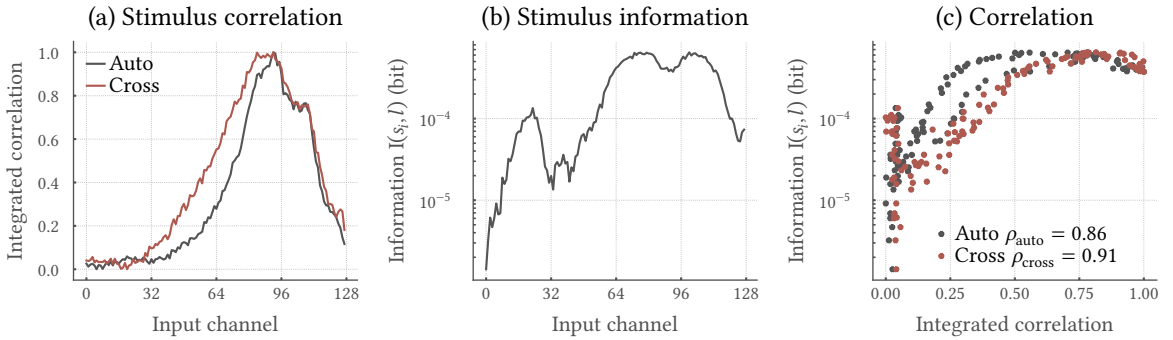


Figure 7.3: Informative auditory spike trains exhibit high correlation. (a) The auditory input channels are correlated to different degrees, visible in both the integrated auto- as well as crosscorrelation. Only in a narrow band of inputs – corresponding to the HCs sensitive to the frequencies of speech – both correlation measures peak. The crosscorrelation for channel i is obtained by estimating the correlations between the spike train of channel i and all other channels j in a pairwise fashion and a subsequent integration over all j. **(b)** The mutual information between input spike trains and digit identity likewise peaks for a narrow range of input channels. **(c)** Both, the auto- and crosscorrelation shown in (a) are highly correlated with the informative synapses as characterized in (b) with Pearson correlation coefficients of $\rho_{\text{auto}} = 0.86$ and $\rho_{\text{cross}} = 0.91$, respectively.

off the teacher spike trains and emulated our networks with frozen synaptic weights and address labels while injecting the randomly drawn test samples of the Iris dataset.

Evaluation: The decision of the network was evaluated by a winner-take-all mechanism imposed in software. In more detail, we evaluated the firing rate of each readout neuron on HICANN-DLS during stimulation with each sample. A most active, winning unit was obtained by comparison among all readout neurons.

7.2.2 Correlation-driven structural plasticity

In the following, we propose a mechanism for structural reconfigurations which is thought to augment the implementation of the critical computing framework with STDP-like weight dynamics described in Chapter 6. In more detail, we aim to adapt the network topology when being exposed to the auditory stimuli presented in Chapter 4 to induce the formation of receptive fields. When transforming audio data of speech with the methods described in Appendix A.1, the resulting spike trains are characterized by spatio-temporal correlations (Figure 7.3a). As it becomes apparent, the input channels exhibiting the highest correlation (Figure 7.3b) match the ones carrying high information about the identity of the spoken word (Figure 7.3c). Most notably, the informative channels are contained within a narrow band of input channels – the frequency band of speech. The remaining channels mostly contain noise induced by thermal fluctuations captured within the hair cell (HC) model (cf. Appendix A.1.2). Hence, the input dimensionality could be dramatically relieved by only considering the informative input channels, thereby compressing subsequent network graphs used for classification. The latter can be achieved by formulating a pruning condition that promotes a preference for correlated input spike trains. Therefore, it seems natural to formulate a STDP-driven pruning strategy which builds on the algorithm described in Section 7.2.1 to detect informative channels within the input.

Table 7.2: Overview of model and stimulus parameters used for the STDP-driven structural plasticity implementation. All time constants are given in wall clock time. The errors indicate the standard deviation. All remaining parameters are given in Table 6.1.

Stage	Parameter	Symbol	Value
Network	Number of inputs	N_{in}	128
	Bundle size	k	4
	Number of synase rows	m	32
	Initial synaptic weight	w_{ij}^{init}	0
	Initial address label	a_{ij}^{init}	$\sim \text{unif}(0, k)$
Plasticity dynamics	Causal STDP time constant	τ_+	$(7.3 \pm 0.7)\,\mu\text{s}$
	Causal STDP amplitude	η_+	0.068 ± 0.022
	Maximum pruning threshold	c_{thres}	10/128
	Structural plasticity update period	T^{struct}	0.5 s
	Experiment duration	T^{exp}	50 s
Input	Spatial correlation strength	ρ_{spat}	$\{0.0, 0.3, 0.6, 0.9\}$
	Temporal correlation strength	ρ_{temp}	$\{0.00, 0.25, 0.50, 0.75, 1.00\}$
	Mean of modulation frequency	ν_{temp}	24 kHz
	Standard deviation of modulation frequency	σ_ν	2.4 kHz
	Stimulus offset	θ	9.6 kHz
	Frequency scaling factor	A	30θ
	Phase	σ_φ	1 ms

In the following, we highlight the artificial and real-world auditory stimuli used to benchmark our implementation. After that, we describe the network topology and the plasticity mechanisms. We again close by a specification of the initialization and evaluation techniques. All model and stimulus parameters are summarized in Table 7.2.

Stimuli: Our STDP-driven structural plasticity targets auditory feature selection for classification tasks like the one provided by the Spiking Heidelberg Digits (SHD) (cf. Chapter 4). Aside from the SHD, we utilize artificial stimuli that aim to reflect the correlations present in the spoken words to finally benchmark our implementation (de Boer, 1980; Meddis, 1986). Furthermore, these artificial stimuli allow us to dissect the effect of spatial and temporal correlation inherently present in the spike trains of SHD. All stimuli are designed to feature 128 distinct spike trains thereby exceeding the fan-in of 32 synapses per neuron on HICANN-DLS by a factor of four.

We first designed a spatially correlated stimulus that exhibited correlation at level ρ_{spat} between different input channels that was chosen to be constant in time. The spiking probability of all input units was calculated from the cumulative distribution function (CDF) of a normal distribution $X \sim \mathcal{N}(\mu, \sigma^2)$ with mean $\mu = 0$ and standard deviation σ given by the correlation parameter ρ_{spat}. In more detail, a spike was generated if a value drawn from this distribution was smaller than the acceptance probability p, $\text{CDF}(X) < p$. To accommodate to a total of 128 inputs, we generated $m = 32$ bundles of inputs, each of which comprising $k = 4$ spike trains. For each bundle, we drew four spike trains with fixed $\rho_{\text{spat}} \in \{0.0, 0.3, 0.6, 0.9\}$. By changing the probability, the frequency $\nu = p\delta T$ was varied, thereby enabling fine-grained network control. Figure 7.4 shows an excerpt of the spikes in a sample stimulus created with the actual model parameters used in the experiments.

The second stimulus was designed to exhibit temporal correlation ρ_{temp} within each input channel. In close accordance with the HC model in Appendix A.1, we here generated spikes according

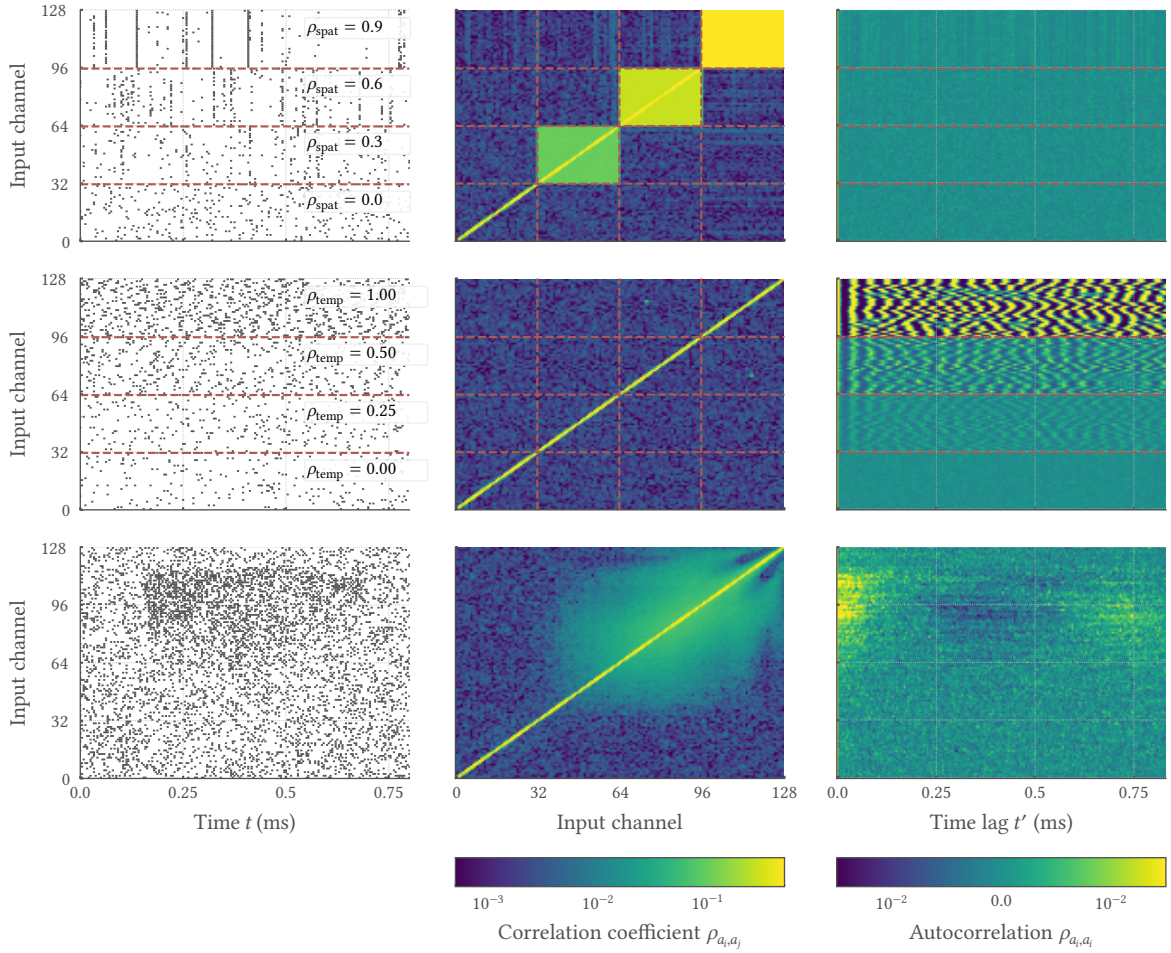


Figure 7.4: Characterization of artificial and real-world auditory stimuli. The two artificial stimuli are designed to exhibit either spatial (first row) or temporal (second row) correlations. Both of which are present within the spike trains of the SHD (third row). In the first column, exemplary raster plots for spatially and temporally correlated artificial stimuli and a real-world spoken digit are shown. The second column contains the crosscorrelation ρ_{a_i,a_j} for all combinations of input channels. In the last column, the autocorrelation for every input channel i, ρ_{a_i,a_i} is visualized. The four distinct correlation levels are visually separated by red horizontal lines. Note that the temporal dimension of the spoken digits has already been scaled to accommodate to the acceleration factor of HICANN-DLS. Panels adapted from Kreft (2019).

to a Poisson process with modulated instantaneous probability:

$$p_i(t) = \max\left\{0, A \cdot \rho_{\text{temp}} \cdot \sin\left(2\pi \cdot \nu_i \cdot (t + \varphi_i)\right) + \theta\right\}, \tag{7.4}$$

for input unit i and a fix offset θ. The frequency ν_i of the sine was drawn from a normal distribution $\mathcal{N}(\nu_{\text{temp}}, \sigma_\nu^2)$ with mean ν_{temp} and standard deviation $\sigma_\nu = 0.1\nu_{\text{temp}}$. In addition, a constant phase shift φ_i is drawn from a normal distribution $\mathcal{N}(0, \sigma_\varphi)$ for every input channel. Together, this avoided synchronous input events across all channels and therefore reduced spatial correlation. The strength of correlation was changed by scaling the amplitude of the sine with a factor ρ_{temp}. Here, A constitutes an additional scaling factor of the sine amplitude which was used to change the firing rate. Like for the spatially correlated stimulus, we generated a total of 128 independent spike trains divided into $m = 32$ bundles of size $k = 4$. Each bundle contains four spike trains with $\rho_{\text{temp}} \in \{0.0, 0.25, 0.5, 1.0\}$. An example of a temporarily correlated stimulus is shown in Figure 7.4.

As an example of real-world stimuli exhibiting both spatial as well as temporal correlation, we

applied a reduced version of the SHD. More specifically, an early version of the dataset without the additional layer of bushy cells (BCs) was used (cf. Appendix A.1). Hence, the associated spike trains exhibited larger levels of noise and a decreased phase-locking. Further, we stuck to the recordings of the English digits »zero« to »nine«. To reduce the dataset to 128 inputs only, we selected every fifth input channel. Moreover, the temporal dimension was compressed by a factor of 1000 to accommodate to the accelerated nature of HICANN-DLS. An example of a downsampled input spike raster of a single digit is shown in Figure 7.4. Again, these spike trains were assigned to $m = 32$ bundles in an alternating fashion in such a way that the first bundle contains the input channels $\{0, 32, 64, 96\}$, the second $\{1, 33, 65, 76\}$ and so on. By this assignment strategy, the highly correlated input spike trains are distributed to different bundles (cf. Figure 7.3a).

Network: We considered the very same network topology as the one described in Section 6.2.1.1 (Figure 6.3), but with 128 instead of only 32 possible external presynaptic partners. Nevertheless, only 32 of them were active at any given point in time for each neuron, all others remained dormant. In more detail, we considered $k = 128/32 = 4$ distinct event addresses to inject bundles of four spike trains into each synaptic row of HICANN-DLS (cf. Figure 7.1). These event addresses were chosen to cover the range $[0, k)$. Again, we configured the address labels of a randomly drawn set of K_{ext} synapses per neuron to a value in the range $[0, k)$ to eventually stimulate our neurons with external spike events. The remaining $32 - K_{\text{ext}}$ synapses per neuron implemented recurrent connections by setting their address label to 63. Similarly, the spike router was programmed to inject all events of neuron i back into the synaptic row i with an event address of 63.

Plasticity: Here, we only implemented structural reconfigurations for synapses transmitting the external stimuli and kept the recurrent ones fixed to not disturb the degree of the external input K_{ext} of our networks. With this choice, the network state could in general still be controlled by the parameter K_{ext} and hence adapted to task requirements. As detailed above, and in contrast to the weight-driven structural plasticity, we here relied on a STDP-based pruning condition when dealing with auditory stimuli. On HICANN-DLS, this was achieved by using the correlation sensors. In more detail, a stimulating synapse's eligibility for pruning was determined by its causal spike-time correlation:

$$a_{ij}(t + T_{\text{struc}}) = \begin{cases} a_{ij}(t), & \text{if } f_+\left(t_i^k, t_j^l, t, t+T\right) \geq c_{ij}(t) \\ \text{unif}(0, k-1), & \text{otherwise}\,, \end{cases} \tag{7.5}$$

with the causal correlation f_+ based on the pre- and postsynaptic spike times, t_i^k and t_j^l in the time interval $(t, t+T]$ according to Equation (3.1). Only if f_+ is greater than or equal to a random threshold value $c_{ij}(t)$ the respective address was kept. In the opposite case, a randomly drawn address label was assigned and the corresponding synaptic weight was reset to w_{init}. The reference values c_{ij} were drawn from an uniform distribution:

$$c_{ij} \sim \text{unif}(0, c_{\text{thres}} - 1)\,, \tag{7.6}$$

that was capped by the pruning threshold c_{thres}. Note that the parameter k determined the number of considered presynaptic partners per row of synapses and translates to a total of $32 \times k$ potential input spike trains when utilizing all synaptic resources of HICANN-DLS.

As already indicated, our STDP-driven pruning can be efficiently implemented on HICANN-DLS by drawing on the correlation measurements and the SIMD vector unit of the PPU. To that end, we

```
; Load causal measurement
inx ca_meas, %[ca_base], %[offset]
shiftb ca_meas, ca_meas, -1
; reset correlation sensors
outx %[select], %[ca_base], %[offset]
; Load weights and addresses labels
inx weights, %[weight_base], %[offset]
inx addresses, %[address_base], %[offset]
; Evaluate pruning condition
lax rand_thresholds, 0, %[threshold_offset]
subbfs conds, ca_meas, rand_thresholds
addbfs conds, conds, %[ones],
; Set conds to one if address is internal
subbfs temps, addresses, %[internal_addresses]
compareb temps
select conds, , %[ones], condsLT
; Load random data to update address labels
lax rands, 0, %[rand_offset]
compareb conds
select addresses, addresses, rands, LT
; Reset weights of pruned synapses
compareb conds
select weights, weights, %[w_init], LT
; Save address labels and weights
outx addresses, %[address_base], %[offset]
outx weights, %[weight_base], %[offset]
```

Listing 7: Kernel code for STDP-driven structural reconfiguration. After digitization of the correlation data via the CADC as well as the loading of synaptic weights and address labels, the pruning condition was evaluated by a comparison of the weight's value to a random threshold. To ensure that only stimulating synapses were updated, we utilized a second compare operation. In case a synapse was pruned, the address label was replaced by a randomly drawn value, thereby assigning a new presynaptic partner. Further, the weight was reset to a small value. Note that the random numbers had to be generated on the general-purpose part of the PPU and were then loaded to the SIMD vector unit. Here, the amount of visited presynaptic partners depends on the range of the drawn random numbers. Shown is the kernel code using NASM syntax.

utilized the update loop shown in Listing 1 for experiment control. Again, the structural plasticity evolved on a slower time scale than the actual network dynamics and weight updates: Only every 500th update triggered by this control loop comprised an evaluation of the pruning condition, i. e. $T_{\text{struc}} = 500 \cdot T$. With this choice, we gave the synaptic weights the chance to develop over multiple update periods. The update loop was augmented by the already discussed kernel shown in Figure 6.3, implementing the actual weight updates, as well as the kernel highlighted in Listing 7 which ensured structural reconfigurations of stimulating synapses.

The kernel code shown in Listing 7 was used to implement structural reconfigurations according to Equation (7.5). The evaluation of the pruning condition first required access to the causal correlation measurements. To that end, a CADC read was triggered from the SIMD vector unit and the

digitized correlation data of a slice of synapses was assigned to the register `ca_meas` (line 2). A bit shift of one was applied to correctly interpret the 8 bit unsigned measurements with the 8 bit signed arithmetic of the SIMD vector unit (line 3, cf. Chapter 3). In direct succession, we reset the correlation sensors (line 4). It is noteworthy that we accumulated pair-wise correlations not over the full course of a structural plasticity period T_{struc} to evaluate the pruning condition in Equation (7.5), but instead relied on the regular weight update duration T. This was due to the fact that the reset among causal and anticausal correlation measurement circuits is shared on HICANN-DLS. Hence, the reset of the anticausal sensors required to implement Equation (6.9) also led to a reset of the causal ones. After accessing the correlation measurements, the synaptic weights as well as the address labels were read from the synaptic SRAM and stored in the registers `weights` and `addresses`, respectively (lines 5 and 6). Since we here applied a stochastic pruning condition, we drew uniformly distributed random numbers on the general-purpose part of the PPU by a *xorshift* algorithm (Marsaglia et al., 2003). The resulting 32 bit values were masked to cover the range $[0, c_{\text{thres}})$ and were subsequently loaded into the register `rand_thresholds` (line 10). Afterwards, the pruning condition was evaluated by subtracting these random numbers from the correlation measurements in `ca_meas` and assigning the result to the register `conds` (line 11). Prior to the actual compare operation, we added ones to these results to implement a less or equal than operation (line 12). Moreover, we had to guarantee that only the address labels of stimulating synapses were updated. To that end, we manipulated `conds` for all recurrent synapses for which the address labels were configured to the maximum possible value. More specifically, we subtracted the address value of recurrent synapses in `internal_address` from the current addresses resulting in `temps` (line 14). Based on this result, we utilized a compare operation, which assigned a value of one to all entries in `conds` for which the result in `temps` was not smaller than zero, i. e. for all recurrent synapses (line 16). Now, `conds` contained only positive values for recurrent synapses as well as stimulating synapses whose weights exceeded the pruning threshold. For the update of all other synapses, we again utilized randomly drawn integer numbers which were masked to cover the range $[0, k)$ prior to loading them into the register `rands` (line 18). Based on the values in `conds`, we either assigned the old states to `addresses` in case the respective entries `conds` were smaller than zero or reassigned the corresponding synapses by assigning the respective elements in `rands` to `addresses` (line 20). Lastly, we reset the weights of pruned synapses to `w_init` by a similar compare operation (lines 24 and 25). The kernel was finalized by writing back the synaptic weights as well as the address labels to the associated slice of the synaptic SRAM (lines 25 and 26).

Initialization: The surrounding network and plasticity model was similar to the one presented in Chapter 6. For an experiment, the network was initialized with a certain K_{ext}. Since the recurrent synapses were not subject to structural plasticity, the degree of the input of our networks was conserved over the course of an experiment despite structural reconfigurations. All weights were initiated with a value of $w_{ij} = 0 \forall i, j$ and the address labels of the stimulating synapses were chosen at random according to an integer uniform distribution with range $[0, k)$. During stimulation with artificial or auditory stimuli, the synapses were subject to the modified STDP according to Equation (6.9) and structural plasticity given in Equation (7.5) allowing the network to adapt. A total of 100 structural plasticity updates were performed leading to an overall experiment duration of $T^{\text{exp}} = 100 \cdot T^{\text{struc}}$. After each evaluation of the pruning condition, the synaptic weights and addresses labels of all synapses were saved for subsequent evaluation. Each experiment was conducted 10 times with different random seeds for the involved plasticity processes as well as the

initialization and stimulation. The results stemming from the network analysis were then averaged over these 10 independent experiments. All shown errors correspond to the standard deviation.

Evaluation: For evaluation, the address labels of every row of synapses were ranked according to the correlation strength of their corresponding spike trains. While this was directly possible for artificial stimuli, we first had to sort the spike trains constituting the auditory stimuli with respect to their integrated auto- or crosscorrelation as shown in Figure 7.3. To finally quantify the performance of our structural plasticity implementation, we evaluated the fraction of all external synapses with a preference for the i-th highest available correlation within each bundle:

$$P_i = \frac{N_i}{N_{\text{ext}}}\,, \tag{7.7}$$

with the number of synapses expressed to an input with the i-th highest correlation within each bundle and the total count of inputs $N_{\text{ext}} = 128$. In other words, for networks with a preference for correlated input spike trains N_3 should be highest.

7.3 Results

In the following, we present results gathered with two structural plasticity implementations on HICANN-DLS. We start with results obtained with a weight-based pruning condition and then move on to a STDP-based pruning. Both presented strategies enable the efficient utilization of the available synaptic resources by the self-organized formation of receptive fields. While our algorithms are not explicitly tailored to the BrainScaleS-2 architecture, their implementation on HICANN-DLS is characterized by low computational overhead.

7.3.1 Weight-driven structural plasticity

To benchmark our weight-driven structural plasticity framework, we consider three label neurons emulated on HICANN-DLS which received input by a receptor layer implemented in software. Specifically, we trained the network to classify the Iris dataset under various sparsity constraints. To that end, we applied synaptic plasticity as well as structural reconfigurations (Section 7.2.1). These mechanisms allowed the emulated SNN to make efficient use of its limited synaptic resources while maintaining high performance on the task at hand.

7.3.1.1 Self-organized formation of receptive fields

In the following, we show that our implementation of structural plasticity promotes the efficient utilization of synaptic resources by the self-organized formation of receptive fields. In general, the distribution of the input data in the feature space renders each receptor informative about different features (Figure 7.2b). Our weight-driven structural reconfiguration in combination with the weight evolution as given by Equation (7.2) leads to a preference of each label neuron for the most informative receptors of its class which in turn promotes the emergence of receptive fields for every label neuron. Thereby, our reconfiguration strategy allows us to dramatically compress network graphs and facilitates learning in sparse networks.

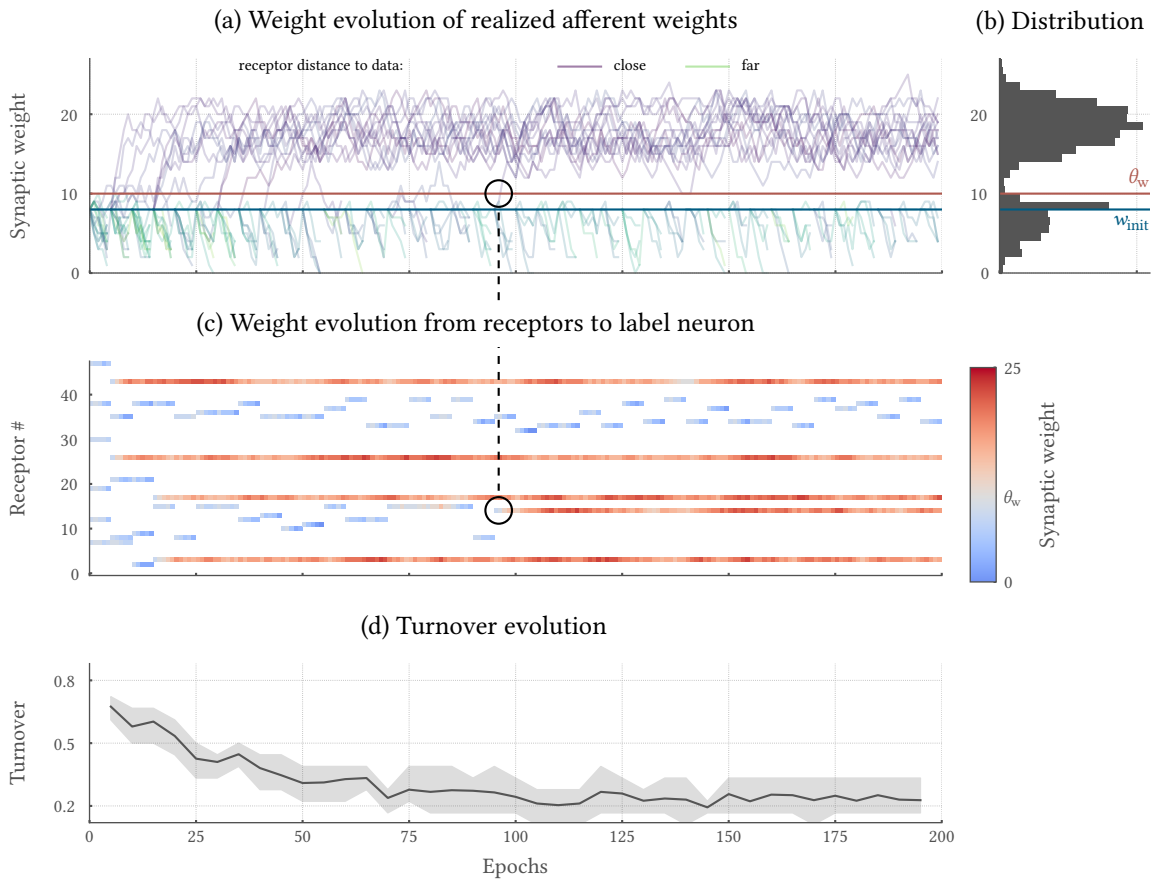


Figure 7.5: Informative synapses emerge during training. **(a)** Exemplary evolution of realized afferent weights of the *setosa* label neuron during the course of a single experiment. The line color is determined by the average feature-space distance between the respective receptor and all *setosa* data points. Synapses that receive inputs from relevant receptors (i. e. those lying close to the features that are relevant for their postsynaptic label neuron) are strengthened towards values that lie above the pruning threshold θ_w. All other, less informative synapses remain below θ_w and are pruned at regular intervals of five epochs. For each pruned synapse, a new one is initialized at w_{init}, between the same label neuron and a previously unconnected receptor. **(b)** Distribution of synaptic weights during the last 50 epochs over 20 randomly initialized runs. Note that the histogram only takes into account realized synapses, which at all times are only 18 out of 144 potential ones. **(c)** Exemplary evolution of all synaptic weights between the receptor population and the *setosa* label neuron. At all times, only $n/k = 6$ synapses are realized. The transition from blue to red marks the pruning threshold θ_w. Note how gray/blue (subthreshold) and white (non-existent) states alternate, marking the pruning of weak synapses and re-initialization of new ones. One of these reassignments is highlighted and referenced to the corresponding threshold crossing in pane (a). **(d)** Evolution of the turnover rate (fraction of all pruned synapses per epoch) for 20 runs. The solid line marks the mean and the gray area represents the 20 and 80 percentiles. As time progresses, the turnover rate converges to approximately 20 %, indicating that all relevant receptors (on average five) have been found. The remaining *free* synapses (on average one) keep switching between all other receptors, but are pruned regularly as they are not informative for the respective class. Figure and caption adapted from Billaudelle et al. (2021).

Informative synapses are potentiated by STDP-based synaptic plasticity. To understand the underlying dynamics, we start by considering the evolution of the connectome over the course of an experiment (Figure 7.5). Starting from their initial values, synapses that contribute causally to the firing of their postsynaptic neurons are potentiated by the STDP-term in Equation (7.2). In more

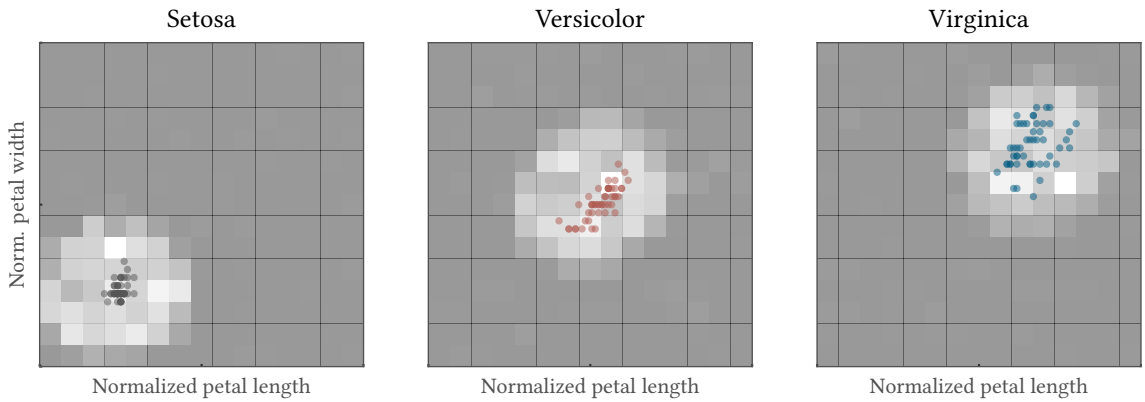


Figure 7.6: Self-organized formation of receptive fields. The probability of synapse expression depends on the location of receptors in the feature space and the class of label neurons. Each square is shaded according to the probability for a label neuron to have formed a synapse with a receptor lying within that area (lighter for higher probability); estimated from the state at the end of training in 100 experiments with random initial conditions. The size of the three emerging clusters is determined by the receptor radius λ. Figure and caption taken from Billaudelle et al. (2021).

detail, the receptors being informative about a given class are more likely to be activated during the presentation of samples belonging to their respective class which is ensured by the additional teacher spike train (cf. Figure 7.2b). The STDP term in Equation (7.2) promotes the facilitation of the corresponding synaptic weights until an equilibrium with the homeostatic force is reached (Figure 7.5a). In contrast, non-active synapses are not strengthened by STDP and hence are not potentiated.

Informative synapses are consolidated by reaching high synaptic weights, thereby persistently exceeding the pruning threshold. At the start of an experiment, the connectome did not reflect the distribution of the dataset, since the synaptic connections between the receptor and label layer were randomly initialized. Consequently, a period of increased pruning and reassignment of synaptic connections sets in. Hence, the turnover rate, given by the fraction of pruned synapses, is high at the start of an experiment (Figure 7.5c). By change, uninformative synapses are replaced by informative ones. In turn, the aforementioned weight dynamics cause the potentiation of the respective synaptic weights which in turn leads to the consolidation of these synapses in case their weights stably exceed the pruning threshold. At the end of an experiment, the pruning and reassignment of a relatively small number of synapses is still visible in the weight distribution by a pronounced peak at the re-initialization value w_{init} (Figure 7.5b). Nevertheless, the turnover rate significantly decreases over time and the connectome converges to a stable solution.

The preference of our label neurons to form informative synapses under the action of our structural configuration strategy suggests the formation of receptive fields. This can be investigated by reconstructing the topology of the emerged connectome after learning from the address labels. To this end, we calculated the probability density for a synapse to be expressed at a given point in the 2D feature plane by repeating the experiments multiple times with different seeds and hence initial conditions. The resulting map approximates the distribution of the input data (Figure 7.6). In more detail, label neurons preferentially cluster their synapses around the location of input samples of their respective class in the feature plane. It is noteworthy that the radius of these receptive fields is in general not only determined by the distribution of the stimuli in the feature plane, but also by

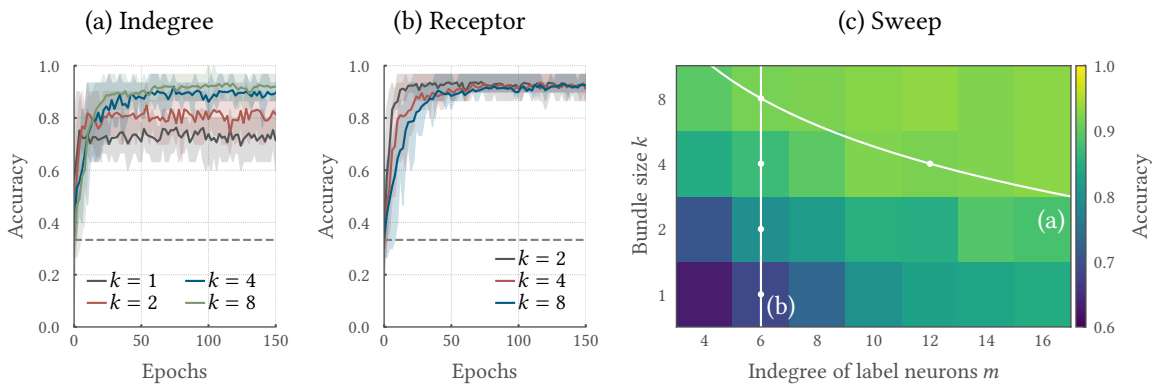


Figure 7.7: Structural plasticity improves classification performance in sparse networks. (a) Structural plasticity promotes the formation of synapses to informative receptors. For a constant indegree m of the label neurons (equivalent with the number of synapse rows on the hardware), classification accuracy improves with larger k, as the neurons gain access to an increasing number of receptors $n = km$. **(b)** Structural reconfigurations allow to utilize the available synaptic resources more efficiently. For a constant number of receptors n, structural plasticity can compensate for increased sparsity (reduced indegree m induced by a larger bundle size k) up to a certain degree. The chance level is indicated by the gray dashed horizontal line. **(c)** The results shown in the panels a and b can be embedded into a more extensive sweep over the number of the indegree m and bundle size k. The white lines highlight the sweeps shown in panel (a) and (b), respectively. Figure and caption adapted from Billaudelle et al. (2021).

the support and shape of the applied receptor kernel (Figure 7.2a). If this preference for informative synapses in sparse networks can, however, be exploited within classification tasks requires the evaluation of performance.

7.3.1.2 Coping with limited synaptic resources

Structural plasticity can be applied to maintain high performance on a given task even for increasingly sparse network topologies. In the following, we show results obtained by imposing different limitations on the connectome. Subsequently, we evaluate the performance of our SNNs by presenting the test data to the receptor layer while keeping the connectome as well as the synaptic weights fixed. In total, we trained and evaluated the networks for each limitation starting from 20 initial configurations to test the ability to generalize and furthermore reduce the impact of specific receptor positionings and initial conditions.

Structural plasticity can be used to expand the virtual fan-in of each neuron to eventually improve performance. This was shown by sweeping the bundle size k, i. e. the number of spike trains injected into a single row of synapses, while keeping the number of utilized synaptic rows on HICANN-DLS fixed at $m = 6$. By this, we increased the number of receptors $n = k \cdot m$ in the receptor layer when increasing k. In other words, each label neuron has access to an increased amount of receptors with rising k, but is only allowed to express a fixed amount of synapses m. Hence, structural plasticity is required to detect the most informative receptors. The accuracy of our networks over the course of training is shown in Figure 7.7a. For all values of k, structural reconfigurations and ongoing plasticity allow to rapidly improve classification performance. With increasing k, the classification accuracy increases with a peak performance of 92.3 % for a bundle size of $k = 8$ showcasing the ability of structural plasticity to express informative synapses. For comparison, we likewise present

results for a network with $k = 1$ corresponding to a setup without structural reconfiguration and only limited access to details in the feature plane. Maybe not surprisingly, all networks subject to structural configurations outperform this network with $k = 1$.

Furthermore, structural reconfigurations can be applied to maintain function in increasingly sparse networks. This was assessed by keeping the number of receptors n constant and sweeping the bundle size k leading to variable numbers of utilized hardware synapses m and in turn to increasingly sparse networks. Here, our networks are able to maintain a performance of about 92 % for $k \in \{2, 4, 8\}$ (Figure 7.7b). Hence our results demonstrate that structural plasticity can be used to reduce the number of required hardware resources while maintaining high performance. By reducing the overall amount of utilized resources, larger and potentially deeper network architectures can be implemented on the neuromorphic substrate (cf. Chapter 5). It is noteworthy that for decreasing numbers of synaptic rows m the connectome converges more slowly since the bundles increased in size and hence more updates are required to explore all potential presynaptic partners by sampling (Figure 7.7b). Here, the accelerated nature of HICANN-DLS facilitates the emulation of these experiments in short periods of time while simultaneously promoting learning with structural plasticity in increasingly sparse networks.

The aforementioned scenarios are a subset of a large sweep over the number of utilized synaptic rows m and the bundle size k (Figure 7.7c). Both, the expansion of the virtual fan-in as well as the increasing sparsity experiments are highlighted in Figure 7.7c by white lines. The final peak performance mainly depends on the number of receptors n and only minor on the count of utilized hardware synapses. In particular, only six hardware synapses are sufficient to reach a comparable performance otherwise only attainable with more than 32 synapses. As a result, our structural plasticity implementation can be utilized to dramatically compress the connectome.

The performance results reported so far can be put in relation by establishing a baseline performance for the very same task, but with an artificially set up network topology. This not only allows us to quantify the impact of structural reconfigurations on performance, but moreover enables the effect of Hebbian weight dynamics on accuracy. To that end, we applied the same bundling of receptors into $m = 8$ bundles of size $k = 8$, but artificially selected a synapse within each bundle. For each label neuron, the latter was obtained by choosing the receptor with the highest mean firing rate, estimated over all samples of the respective class. By evaluating the resulting receptive fields for multiple seeds, i. e. different initial conditions, we are able to compare this artificial connectome with a network topology generated by our structural plasticity implementation. Here, the expression probabilities obtained by both approaches are strongly correlated (Figure 7.8a). Hence, our implementation indeed generates connectomes relying on informative synapses. Figure 7.8a shows a slight underrepresentation of receptors exhibiting the highest expression probabilities in the learned connectome. This is likely to be a consequence of the threshold-based pruning strategy: In case a synapse gets potentiated due to the Hebbian contribution, its synaptic weight may exceed the pruning threshold and hence becomes consolidated even though a receptor carrying higher information about the class membership may be present within the corresponding bundle. As a result, further exploration of potential presynaptic partners is prevented.

For our implementation, weight dynamics were mainly governed by Hebbian potentiation as well as unspecific regularization. In contrast, the synaptic weights within the previously described artificial connectome remain as free parameters. To finally establish a baseline performance and to dissect the effect of Hebbian weight dynamics, we visited two different methods to assign weights to each synapse. First, we configured all weights to the same fixed value $w_{ij} = s$ for all synapses.

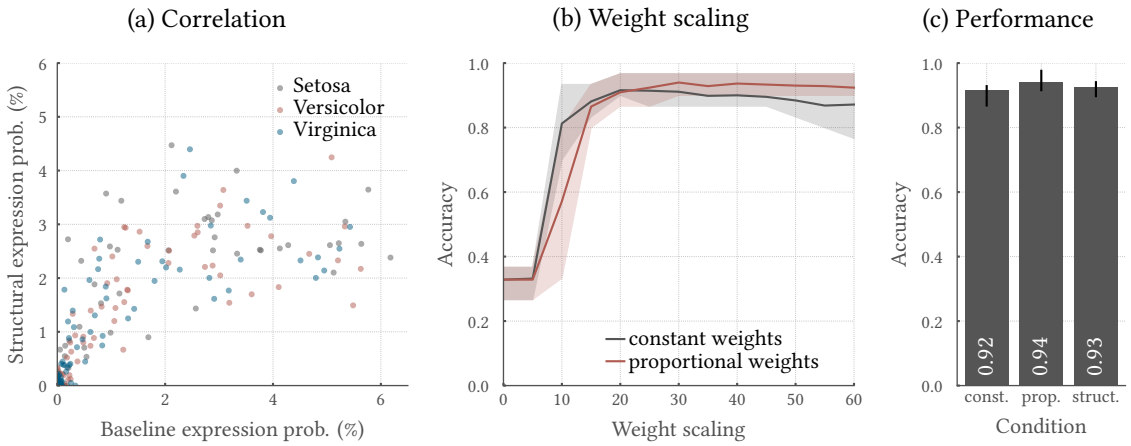


Figure 7.8: Comparison of structural plasticity to a baseline estimate. Structural plasticity yields an accuracy comparable to the one of a network obtained by artificially choosing the most active receptors for each class. **(a)** A clear correlation between the synapse expression probabilities of the trained and the artificial networks can be observed. **(b)** A proportional weight selection mimicking Hebbian potentiation outperforms connectomes with homogeneous synapses. For the proportional weight selection, the weights of the artificial connectome were scaled with the receptors' mean activations. In contrast, the weights were configured homogeneously to a constant value for the constant weight selection. Each approach was evaluated for different weight scaling factors. **(c)** Classification performance for structural plasticity is on par with the respective maxima from (b), where the proportional scaling outperforms the constant one. Figure and caption adapted from Billaudelle et al. (2021).

This scenario essentially captures the ability of a network with ongoing structural reconfigurations, but without weight dynamics. Second, we set the weights to $w_{ij} = s \cdot \nu_j / \max_j(\nu_j)$, with the average receptor firing rate ν_j, thereby mimicking the combined effects of structural reconfiguration and Hebbian plasticity. Both methods were evaluated for various scaling factors s (Figure 7.8b). As a result, we considered the respective maxima as a suitable measure for a performance comparison which is shown in Figure 7.8c. Most notably, admitting synapse-specific weight values mimicking Hebbian plasticity improves upon our baseline performance with constant weights. It should be noted that aside from the favorable role of Hebbian weight selection, the corresponding STDP term within our weight dynamics constitutes the driving force for the formation of receptive fields within our structural plasticity algorithm. Likewise, our implementation also improves upon the baseline with constant weights, but falls slightly behind the performance achieved with distinct weights. It is noteworthy that both of the presented baseline measures rely on global information of the Iris dataset and the respective receptor activities. In contrast, the connectome that emerged through structural plasticity was generated with local information only and nevertheless yielded comparable performance.

The performance of a given network depends on the hyperparameters used within the weight update rule. For our algorithm, the steady-state weight distribution is determined by the relative parametrization of the individual contributions in Equation (7.2) (Figure 7.5). Since our pruning condition is centered around the strength of synaptic weights, the selection of a suitable pruning threshold has to take into account the final weight distribution and in turn the utilized hyperparameters. In general, the pruning threshold θ_w needs to be high enough to enable the pruning of uninformative synapses, thereby facilitating the sampling of the potential feature space. Apart from

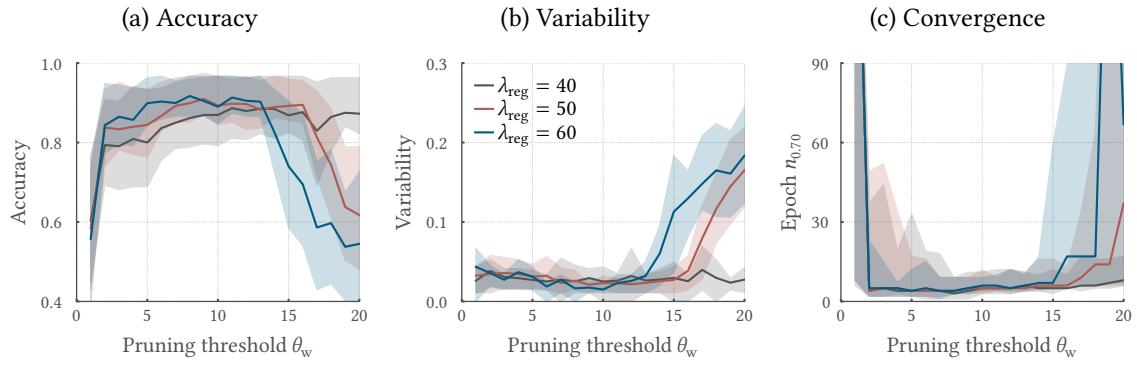


Figure 7.9: Networks promote stable performance over a wide range of hyperparameters. We varied the pruning threshold θ_w and the regularization strength λ_{reg}, which both shape the steady-state weight distribution. For different aspects of learning performance, broad plateaus with respect to variations of these hyperparameter can be observed. Solid lines and shaded areas respectively denote mean and 20-80 percentiles, measured over 20 randomly initialized experiments. The plateaus mostly coincide for **(a)** classification accuracy after learning (average over the last 20 epochs), **(b)** variability of accuracy after learning (standard deviation over the last 20 epochs), and **(c)** number of epochs until an accuracy of 70 % was reached, $n_{0.70}$. Figure and caption taken from Billaudelle et al. (2021).

that, θ_w has to be low enough to guarantee the consolidation of informative synapses. Because of its crucial role, we evaluated the performance of our SNNs by different metrics for various pruning thresholds (Figure 7.9). In addition to θ_w, we swept the regularization strength λ_{reg}, since the weight distribution after learning mainly depends on the balance of the Hebbian potentiation and the depressing effect of the regularization which in turn also impacts the choice of a suitable θ_w. For a broad range of pruning thresholds and regularization strengths not only the accuracy shows a broad plateau of high performance, but also the resulting connectomes show the least amount of variability during inference and moreover converge with the least amount of epochs. Hence our learning rule seems to be broadly applicable without precise parameter fine-tuning.

7.3.1.3 Self-organized adaptation to switching tasks

So far, we have demonstrated learning in sparse networks through structural plasticity. This was achieved by sampling the input space to eventually form informative receptive fields. All our experiments were based on uniformly drawn initial address labels and hence randomly initialized connectomes. Moreover, all synaptic weights were initially configured to the same value. In the following, we quantify the ability of our plasticity implementation to learn from different initial configurations, particularly from a previously learned state with structured connectome and non-uniform weight distribution. In more detail, we changed the task during training by rearranging the receptors in the feature plane after 200 epochs. This resulted in a misalignment of the stimuli and the receptive fields. During this *task switch*, the PPU continuously updated both the address labels as well as the synaptic weights.

Immediately after the rearrangement, the accuracy drops to approximately chance level (Figure 7.10). The associated disappearance of spike-time correlation leads to a take-over of the depression through unspecific regularization (cf. Equation (7.2)). As a result, the synaptic weights decrease and potentially fall below the pruning threshold θ_w. Hence, a period of increasing pruning

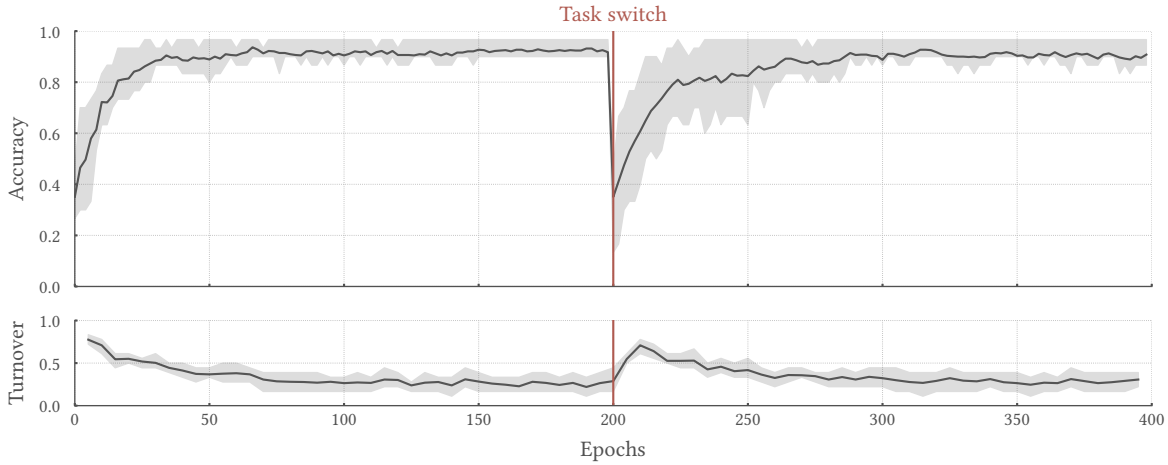


Figure 7.10: Restoration of network performance after task switch. After training for 200 epochs, the receptor layer is randomly rearranged, leading to a mismatch in receptive fields. Ongoing structural plasticity unlearns the previously established connectome and quickly starts to again explore the input space. This process can be observed in an elevated turnover rate after the task switch, similar to the initial phase of the experiment. Figure and caption taken from Billaudelle et al. (2021).

sets in, visible in a rapid increase of the turnover rate. By this mechanism, our SNNs unlearn the previous connections and newly explore the feature space to again form informative receptive fields. After about 100 additional epochs, our networks again show high performance. This behavior not only suggests that our algorithm promotes learning with different initial conditions, but highlights the ability of structural plasticity to constantly adapt SNNs to changing environments.

7.3.1.4 Efficient implementation on BrainScaleS-2

Our presented structural plasticity algorithm comes with low computational overhead, especially when implemented on HICANN-DLS. Here, the efficiency of pruning and reassignment of synapses was promoted by the filtering of events in each synapse according to their source address. Since the connectome is partially defined by the address labels stored in the local SRAM of each synapse, local operations are sufficient to effectively change the network topology.

Aside from the partial definition of the connectome within each synapse, efficiency is further promoted by the implementation of the plasticity algorithm on the SIMD vector unit of PPU. Its tight coupling to the analog neuromorphic core of HICANN-DLS is key for the combination of analog emulation and the digital implementation of weight updates as well as structural changes (Section 3.1). Essentially, our strategy of structural reconfiguration encompasses the read and write of the synaptic SRAM as well as a couple of arithmetic operations (Listing 6). All of these operations can be carried out in parallel by the SIMD vector unit of the PPU. Here, 110 clock cycles were required to apply a structural plasticity update on a single slice of 16 synapses (Figure 7.11). This corresponds to only 1.1 µs at a PPU clock frequency of 100 MHz. Calculated back to the update duration of a single synapse, this amounts to only 69 ns. For comparison, we also assessed the execution time of the individual contributions to the weight dynamics: The completion of the STDP kernel took approximately 3.8 µs for a slice of 16 synapses and hence 240 ns per synapse. The remaining regularizer and random walk contributed 69 ns and 97 ns, respectively. All three terms were implemented

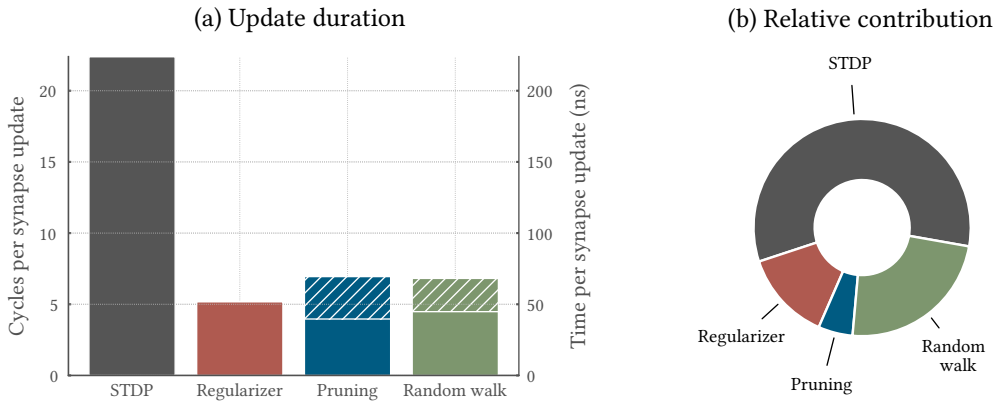


Figure 7.11: Efficient mixed-signal implementation of structural plasticity. (a) Duration of a synapse update broken down into its four individual contributions, including structural reconfiguration. The hatched areas indicate the time spent on pseudo-random number generation. **(b)** Contributions of the individual terms to the overall update duration, taking into consideration that pruning and reassignment are executed five times less often than synaptic weight updates. Figure and caption taken from Billaudelle et al. (2021).

separately and were hence not optimized for execution speed (cf. Section 7.2.1). Reducing common accesses to the synaptic SRAM or sharing intermediate calculation results between the individual contributions would further improve upon the presented results. It is noteworthy that the weight dynamics are updated five times more often compared to the pruning and reassignment of synapses and hence dominate the overall execution time. As a result, our implementation of structural plasticity is characterized by low computational overhead.

An implementation of our structural plasticity algorithm on the full-size High Input Count Analog Neural Network X (HICANN-X) chip promises to even further reduce the computational overhead. When breaking down each kernel, memory accesses as well as the generation of pseudo-random numbers contributed most to the total runtime. Access times primarily depend on the design choices of the system, but can be optimized. Likewise, the generation of random numbers can be sped up by incorporating hardware accelerators. Within our algorithm, both the random walk as well as the pruning and reassignment of synapses required the drawing of pseudo-random numbers. The portion of time spend on their generation constituted a significant amount of the overall execution time (Figure 7.11). With the inclusion of hardware accelerators into the BrainScaleS-2 architecture for the HICANN-X chip, this contribution could be reduced to negligible 0.08 clock cycles for the drawing a single 8 bit random number per synapse (Section 3.2). Hence our approach is not only able to efficiently use synaptic resources, but furthermore facilitates an implementation with low computational overhead by exploiting key features of the BrainScaleS-2 architecture.

The sampling of the potential input space associated with structural reconfigurations profits from the accelerated nature of BrainScaleS-2. Usually, the extensive time scales governed by structural plasticity are prohibitively expensive for many implementations. In this regard, we were able to benefit from the accelerated nature of the BrainScaleS-2 architecture. Here, a single epoch of 24 s in the biological time domain was performed in only 137 ms as measured from the host system. When only considering the execution time on HICANN-DLS, i. e. disregarding the on-the-fly generation of input spike trains, this number boils down to less than 50 ms and hence a measured speedup of a factor of 500. While, we only implemented the weight updates and structural reconfigurations on the

embedded PPU within this work, Wunderlich et al. (2019) demonstrated that porting the experiment control from the host and field-programmable gate array (FPGA) to the PPU further reduces the overhead by minimizing I/O. By this, the system's power consumption could be lowered to less than 60 mW with a minor dependence on network activity and plasticity evaluation (Wunderlich et al., 2019).

The speedup of BrainScaleS-2 is achieved by emulating neuro-synaptic dynamics with above-threshold analog transistor circuits (Chapter 3). Although these transistors deviate induced by variations in the fabrication process, the individual parametrization of each circuit allowed us to reduce this fixed-pattern noise (Aamir et al., 2018). More specifically, we employed calibration routines to equilibrate not only the neuron circuits, but also the synaptic correlation sensors used to implement the STDP-like term in Equation (7.2). The impact of the remaining variability on network performance was assessed by transferring a learned connectome and the associated weights to different circuits. In more detail, we trained a population of three label neurons and then replicated the learned connectome to four disjunct groups of neurons and their associated synapses on the same chip. By relying on distinct subsets of neuron and synapse circuits – each of which with their own variations – we were able to establish a measure of transferability of training results across different systems by comparing the performance achieved by each set of label neurons with the original population of neurons. Most notably, the inferred performances across all groups of neurons deviated by only 1.2 % from the originally trained population. While this result showcases the ability of a calibrated system to perform inference without device-specific training, on-chip learning furthermore enables the compensation for non-ideal calibration data underpinning its necessity in the context of analog and mixed-signal neuromorphic systems (Wunderlich et al., 2019).

7.3.2 STDP-driven structural plasticity

The results obtained so far were acquired in combination with structural reconfigurations centered around a weight-based pruning condition. Here, we adopt a STDP-driven approach to ultimately select auditory features based on spatio-temporal correlation. In the auditory system, the HCs – converting the basilar membrane (BM) decompositions to spikes by mechanotransduction – phase-lock to different frequencies depending on their position along the BM (Appendix A.1). These HCs emit uncorrelated spike trains in the absence of stimulation due to thermal fluctuations (Appendix A.1.2). In case of an incidental sound wave, the BM causes spatial frequency dispersion which in turn leads to stimulation of a set of HCs that depends on the spectrum of the impinging stimulus. Consequently, this set of HCs emits spike trains exhibiting spatio-temporal correlations. As a result, only the spike trains of these HCs provide high information about the stimulus (cf. Figure 7.3). Our STDP-based pruning condition is designed to promote the formation of synapses from these HCs to the downstream SNN by favoring presynaptic partners emitting correlated spike trains. This structural plasticity mechanism is meant to augment the critical computing framework presented in Chapter 6, since the classification of spoken words constitutes a prominent application of reservoir computing (Maass et al., 2003; Skowronski & Harris, 2007; Verstraeten et al., 2006).

To this end, we utilized artificial as well as real-world auditory stimuli which constitute the input spike trains for SNNs emulated on HICANN-DLS. In more detail, we generated a total of 128 input spike trains, but imposed a sparsity constraint on our recurrent SNNs in such a way that each neuron was only allowed to connect to $K_{\text{ext}} \leq 32$ of them at each point in time. Structural reconfigurations were restricted to synapses transmitting the external stimulus to not disturb the degree

of the input K_{ext} and occurred on a five-times slower time scale than the actual weight dynamics. The results shown in this section are meant as a demonstrator of the various opportunities provided by our structural plasticity implementation on HICANN-DLS, particularly they highlight the combinability of different weight dynamics and various pruning conditions. The investigation of collective dynamics of our networks when being exposed to correlated inputs in favor of Chapter 6 is the subject of future research.

To dissect the complex correlation patterns present in real-world auditory stimuli, we first investigate artificial stimuli, designed to exhibit either spatial or temporal correlation before considering real-world stimuli (Figure 7.4). In order to inject all 128 input spike trains into the neuromorphic substrate, they were grouped into $m = 32$ bundles with size $k = 4$ (Figure 7.1). We assigned a unique event address to each spike train in these bundles and in turn, injected them into the synaptic rows of HICANN-DLS. The PPU was programmed to update not only synaptic weights, but furthermore reassign the synapse-local address labels, thereby exploring the feature space by sampling all presynaptic partners residing within the corresponding bundle of the associated synapse. The preference of a network for a stimulus was quantified by $P_i = N_i/N_{\text{ext}}$ with the number of synapses transmitting spikes with the i-th highest correlation within their bundle, N_i and the total amount of stimulating synapses $N_{\text{ext}} = K_{\text{ext}} \cdot N$. While this ranking of input spike trains according to their correlation is directly available for artificial stimuli, the spike trains of the real-world auditory stimuli were arranged according to their auto- and crosscorrelation, respectively.

7.3.2.1 Preference for correlated input spike trains

Our correlation-driven structural reconfiguration preferentially selects inputs exhibiting high correlation for a broad range of degrees of the input K_{ext} (Figure 7.12). Since the structural reconfiguration was embedded into the reservoir computing setup presented in Chapter 6, the preference has to persists for various K_{ext} (cf. Section 6.3.1). This holds true for both artificial stimuli as well as for the real-world auditory stimulus which all show a preference for the highly correlated inputs – i. e. N_3 is the highest component – for a range of K_{ext} values (Figure 7.12). Especially for intermediate values of K_{ext}, a clear preference can be observed. Only for highly recurrent and almost completely input-driven networks, this preference degrades. In these regimes, the network is either input-driven or almost decoupled from the input and hence at its extreme values (cf. Section 6.2.1.1). Furthermore, the correlated inputs put stability at risk and most likely decrease the available range of K_{ext}. Because of this, the shown K_{ext} ranges within this chapter are slightly reduced compared to the results shown in Chapter 6.

7.3.2.2 Adjusting the pruning condition to stimulus characteristics

The time scale and sensitivity of the correlation measurements used for pruning have to match the characteristics of the stimulus. After about 50 epochs of structural reconfiguration, all networks converged to a state characterized by an excess of synaptic connections to correlated inputs irrespective of the stimulus. However, the speed of convergence was highest for the temporally correlated stimulus (Figure 7.12). For the real-world stimulus, however, the preference for the inputs revealing the highest correlation was reduced (Figure 7.12 bottom). This is likely to be a consequence of different time scales or an overall lower correlation strength within the real-world stimulus (Figure 7.4). Furthermore, the assignment of input channels to bundles plays a major role since multiple spike trains

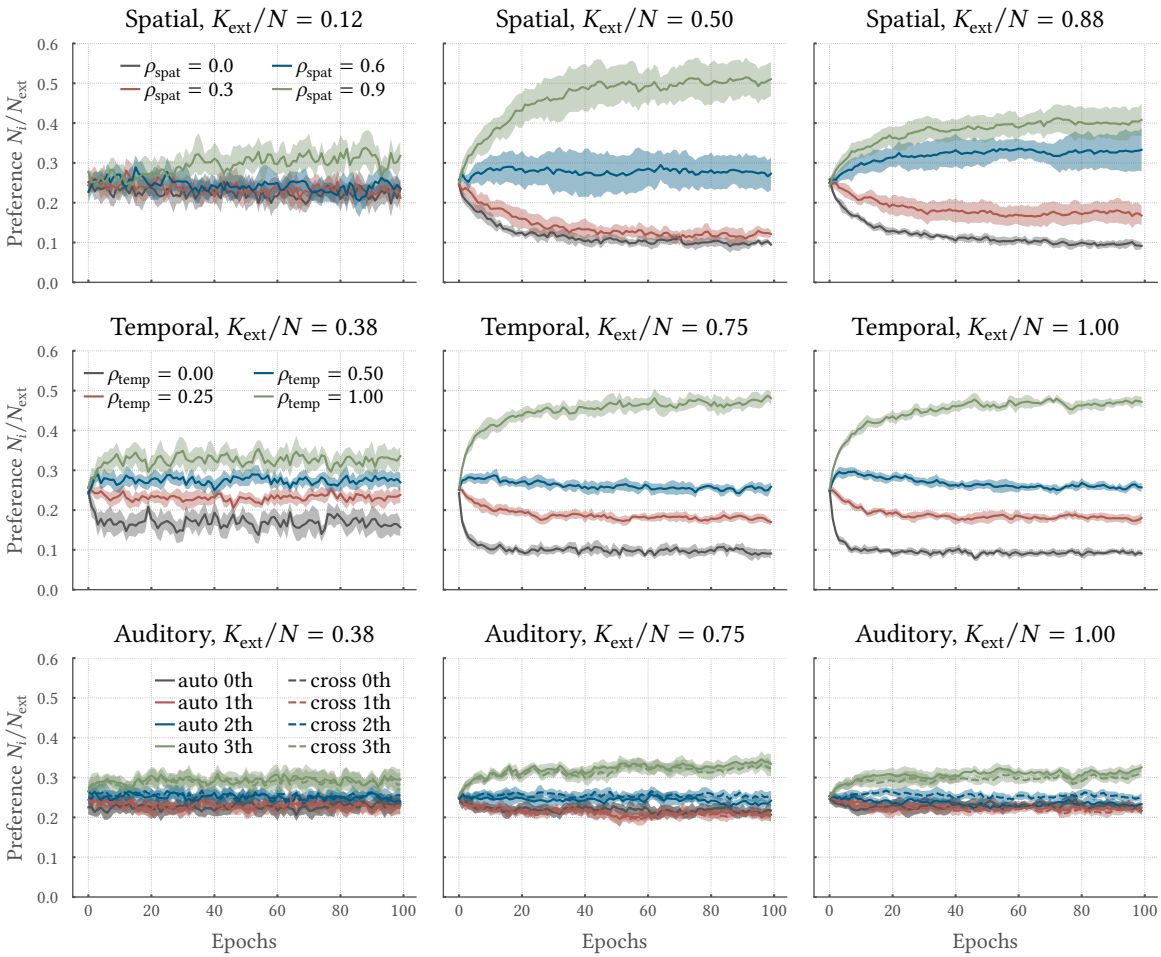


Figure 7.12: STDP-based pruning promotes a preference for correlated stimuli. The probability of synapse expression depends on the correlation present in the input spike trains. For artificial stimuli, our SNNs preferentially express synapses to stimuli exhibiting highest spatial ρ_{spat} and temporal ρ_{temp} correlation. Similarly, the input channels of the auditory stimulus with highest auto- and cross-correlation are preferentially chosen as presynaptic partners. The probability of expression is quantified by the number of synapses in the respective correlation group N_i normalized to the total amount of external synapses N_{ext}. Here, results for three distinct stimuli are shown a spatially correlated (top), a temporally correlated (middle), and a real-world auditory (bottom) stimulus. For each of them the networks are evaluated for variable degrees of the input K_{ext}/N. Panels adapted from Kreft (2019).

residing within the same bundle may by chance feature similar degrees of correlation. This again stresses the impact of suitable assignment strategies, since the amount of correlation present within auditory stimuli varies dramatically over inputs with channels exhibiting almost no correlation to very high levels of correlation. Like for the weight-driven approach, the pruning threshold has to be selected carefully to cope with this situation, i. e. to ensure that the connectome is not prematurely consolidated with inadequate synapses or dissolved by removing informative synapses.

The pruning threshold can be used to adjust the sensitivity to correlated stimuli. In contrast to the previously visited weight-based pruning condition, we here consider a stochastic pruning threshold to cope with the diverse correlation levels of auditory stimuli as well as the fixed-pattern noise of the correlation sensors on HICANN-DLS (cf. Figure 3.4). To investigate its impact, we only consider artificial stimuli and investigate the number of synapses that established a connection with

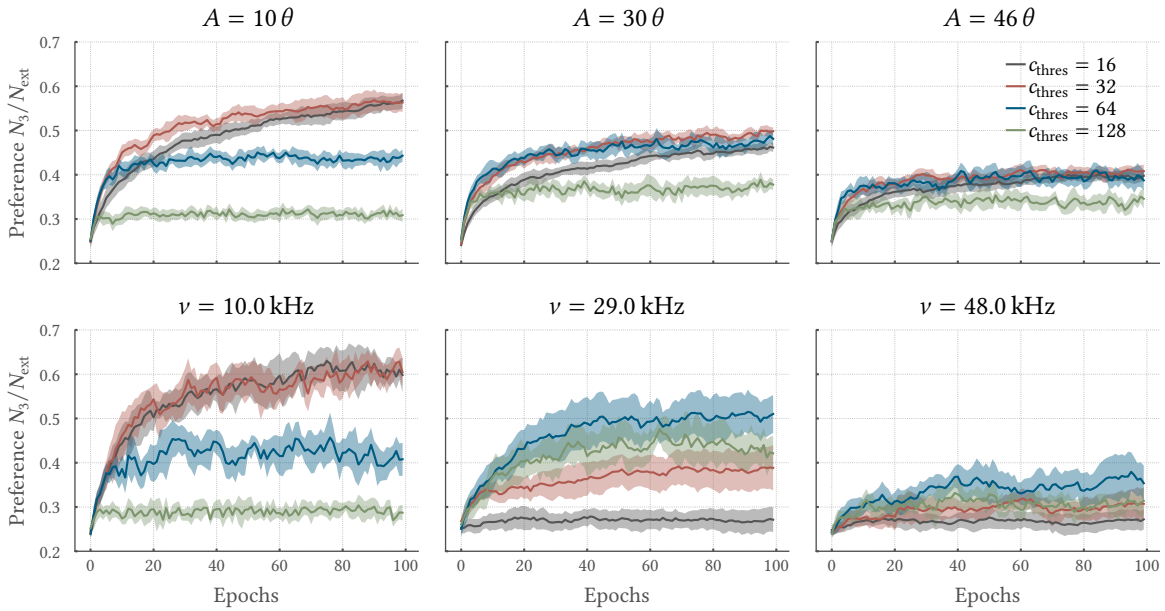


Figure 7.13: Correlation sensitivity can be adjusted by tuning the pruning threshold. The range of the stochastic pruning threshold c_{thres} impacts the sensitivity to correlation and hence the final connectome. Thus, each stimulus requires an appropriately chosen upper limit for the drawn threshold values. Receptive fields emerge for specific combinations of c_{thres} and correlation levels. Here, the temporal evolution of the number of synapses connecting to the inputs exhibiting the highest available correlation N_3 is shown for the temporally (top) and the spatially (bottom) correlated stimuli. For both types of stimuli, we controlled the exhibited correlation by the scaling parameter A and the stimulus frequency ν, respectively. Panels adapted from Kreft (2019).

the presynaptic partner exhibiting the highest correlation within each bundle. In addition to the range of the stochastic pruning threshold c_{thres}, we swept the parameters ν and A, effectively controlling the correlation magnitude of the spatially and temporally correlated stimuli, respectively. For the spatially correlated stimulus, the frequency ν for which the largest split happens gets shifted under adjustments of the pruning threshold (Figure 7.13). While for high spike frequencies around $\nu = 30\,\text{kHz}$, the highest split is achieved with $c_{\text{thres}} = 64$ it is shifted to $c_{\text{thres}} = 32$ for lower frequencies. Below $c_{\text{thres}} = 16$ no stable convergence can be observed and address labels were randomly reassigned. This behavior is also observable for the temporally correlated stimulus when adjusting A. Therefore, a reduction of the pruning threshold can be used to counteract the effect of changing correlation strengths present within the stimulus. It is noteworthy that the choice of a stochastic pruning threshold is particularly beneficial when considering non-overlapping bundles of input spike trains. As discussed earlier for the weight-based pruning, multiple informative inputs residing within the same bundle may cause the formation of a non-ideal connectome by the consolidation of unfavorable synapses. Here, the stochastic pruning threshold may by chance facilitates to again prune and reassign these synapses to further explore the feature space. This, however, comes at the cost of ongoing pruning activity over the course of an experiment.

The stimuli have to emit a certain level of correlation for a fixed range of the pruning threshold c_{thres} to cause the formation of receptive fields. To systematically analyze the behavior in response to different stimuli, we in the following summarize the preference of our networks for each of the four distinct correlation levels by averaging the preference over the last 10 structural plasticity updates (Figure 7.14). We again control the observable correlation by setting the frequency ν and the ampli-

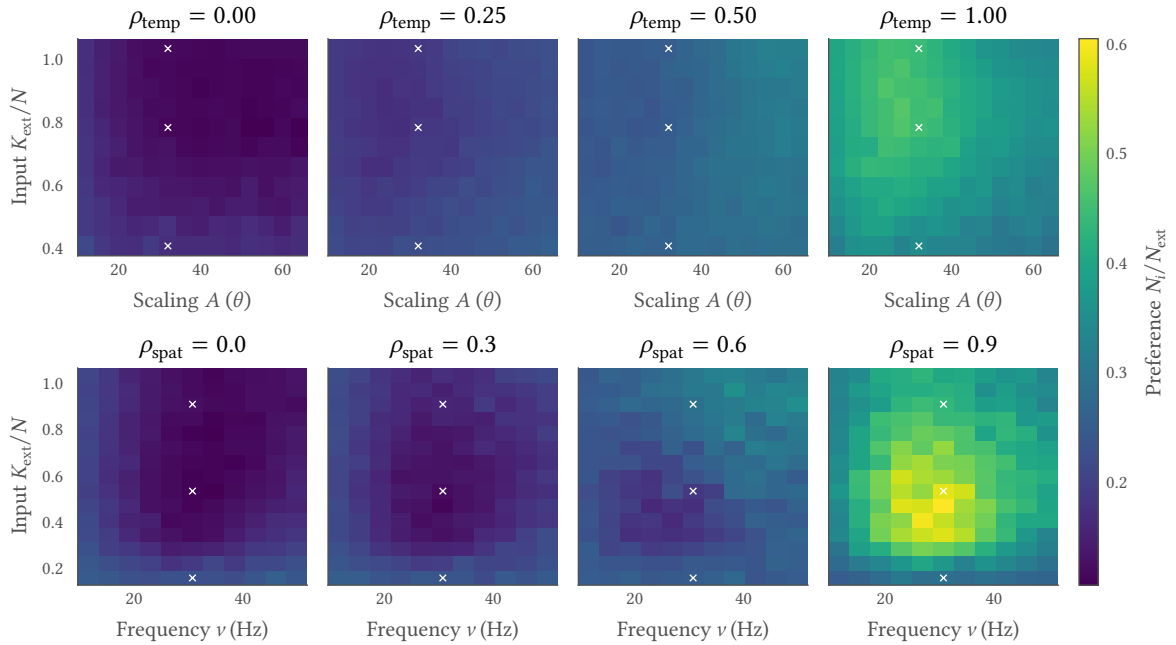


Figure 7.14: Receptive fields emerge for different degrees of the input and a magnitude of stimulus parameters. The networks show a clear preference N_i/N_{ext} for highly correlated input spike trains (high values of ρ_{temp} and ρ_{spat}) for a broad range of degrees of the input K_{ext}/N and stimulus hyperparameters A and ν. Synapse expression probabilities for every input group are shown for the temporally (top) and the spatially (bottom) correlated artificial stimulus. The preference is averaged over the last 10 structural plasticity updates. The white crosses highlight the configurations shown in Figure 7.12. Figure adapted from Kreft (2019).

tude scaling A, but for a fixed value of c_{thres} to further investigate the need for precise adjustments in c_{thres} under changing external stimulation. Despite slightly displaced regions of high correlation sensitivity, the preference for correlated input is maintained for a broad range of different correlation strengths. These regions mainly depend on the parametrization of the causal STDP-kernel used within the pruning condition (Equation (7.5)). The time constant of the kernel has to be adjusted to cover the stimulus relevant frequencies: When keeping the threshold fixed, more synapses are pruned when decreasing the frequency since less correlation per measurement interval can be detected. Hence, the time scale of the STDP kernel has to mimic the stimulus characteristics, although no fine-tuning is required.

7.3.2.3 Self-organized formation of receptive fields

The STDP-based pruning strategy causes the self-organized formation of receptive fields for real-world auditory stimuli. In the following, we consider the expression probability of a synapse as a function of the position, i. e. the frequency, of the HC on the BM. The resulting distribution resembles the correlation present within the emitted spikes of each HC and hence the information content about the identity of the stimulus (Figure 7.3a). For the tested degrees of the input, this information is indeed correlated with the synapse expression probability which confirms the ability of our STDP-driven structural plasticity to select informative auditory features (Figure 7.3b). The reduced correlation between the expression probability and the information carried by the respective input for very low and high K_{ext} is likely to be a consequence of unstable network dynamics.

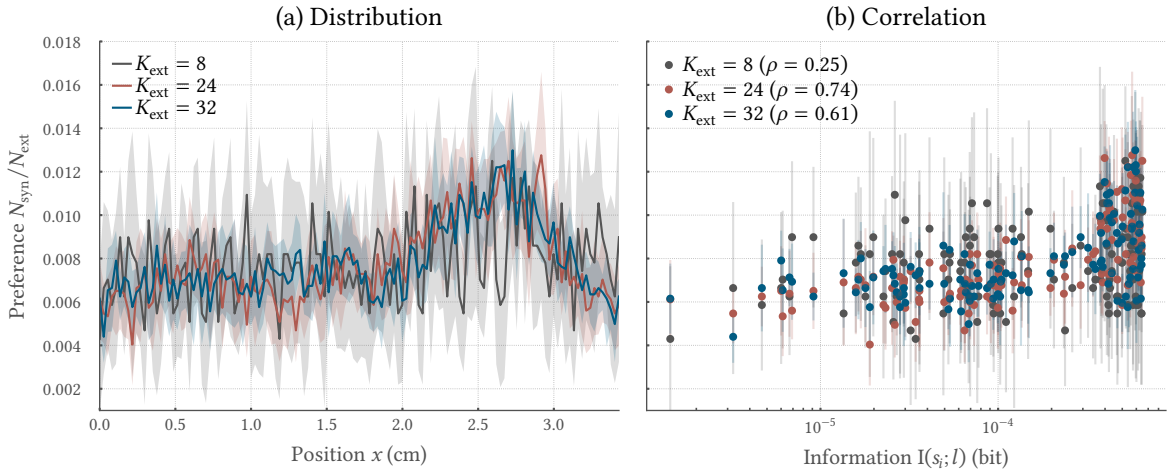


Figure 7.15: Self-organized formation of auditory receptive fields. Synapses preferentially cluster to HCs with specific frequencies covering the human voice. **(a)** The synaptic expression probability changes with the position of the HC on the BM, i. e. with the stimulus frequency. The preference N_{syn}/N_{ext} corresponds to the fraction of synapses expressed at the respective position x and the total number of available synapses N_{ext}. For each experiment, N_{syn} is averaged over the final 50 structural plasticity updates. **(b)** The STDP-based pruning strategy selects informative auditory features. Specifically, the expression probability of each input shown in (a) correlates with the information conveyed by the corresponding HC about the stimulus identity highlighted in Figure 7.3. Most notably the Pearson correlation coefficient ρ was highest for intermediate degrees of the input K_{ext}. Panel (a) adapted from Kreft (2019).

7.3.2.4 Efficient implementation on BrainScaleS-2

Like the weight-based structural plasticity, the STDP-based reconfiguration is based on synapse-local information only and hence promotes an efficient on-chip implementation on HICANN-DLS with low computational overhead. In comparison to the weight-based pruning algorithm, additional pseudo-random numbers are required for the implementation of the stochastic pruning threshold. While the associated overhead for the drawing of pseudo-random numbers has to be accepted on the prototype chip, the time spent on their generation could be reduced on HICANN-X by drawing on the specialized hardware accelerators (cf. Section 3.2). With this and the fact that structural updates occur on a way larger time scales, the time spend on the evaluation of the pruning condition on the SIMD vector unit of the PPU becomes negligible when compared to the actual weight update calculations.

The presented STDP-framework profits from the accelerated nature of the BrainScaleS-2 architecture. Particularly, the high update period for structural reconfigurations required for the long-lasting relaxation phases of the weight and network dynamics would render an implementation on many contemporary platforms prohibitively expensive (Figure 6.17). Again, the accelerated emulation comes to rescue by facilitating these long-lasting experiments thereby allowing to bridge the gap between time scales of neuro-synaptic dynamics and update intervals of structural reconfigurations. With this, the increased emulation times required to fully sample the feature space to eventually establish a suitable connectome through structural reconfigurations become feasible.

7.4 Discussion

Within this chapter, we presented an efficient on-chip implementation of structural plasticity on HICANN-DLS relying on two distinct pruning conditions. Both strategies allowed us to train sparse networks of spiking neurons by utilizing the synaptic resources more efficiently. This was especially facilitated by the self-organized formation of receptive fields without prior knowledge of the distribution of the input data. For a supervised learning task and a network with Hebbian weight updates, this led to the maintenance of high performance for increasingly sparse network topologies. Further, we augmented the critical computing framework presented in Chapter 6 with a correlation-based pruning condition which enabled the detection of informative auditory features. By replacing the pruning condition, we were not only able to give an insight into the broad applicability of structural reconfigurations, but furthermore, demonstrate the efficiency of our implementation in combination with HICANN-DLS.

The selection of informative features of the Iris dataset prior to stimulus injection into a neuromorphic hardware system has been visited in the past. Schmuker et al. (2014) proposed an off-chip preprocessing based on a principal component analysis to determine informative receptor locations and to finally satisfy the fan-in of a predecessor of HICANN-DLS. Based on the found receptor locations, they defined a static connectome which was accordingly loaded to the neuromorphic substrate for inference. In contrast to this feature selection strategy, we relied on the self-organized formation of receptive fields by structural plasticity. Most notably, our algorithm only incorporates local information and does not require additional knowledge of the distribution of data in the feature space which in turn facilitated our on-chip implementation.

The concept of structural plasticity has been applied to vastly different network topologies in combination with different learning paradigms (Butz et al., 2009; George et al., 2017; Bogdan et al., 2018; Kappel et al., 2015; Bellec et al., 2017). Aside from its role in coping with limited space and energy constraints, structural plasticity is thought to improve the overall network performance (Roy et al., 2014b; Spiess et al., 2016) and memory capacity when evaluated in combination with nonlinear multi-compartment neuron models (Poirazi & Mel, 2001; Hussain & Basu, 2016). We already gave a glimpse of the broad applicability and flexibility of our structural plasticity implementations on HICANN-DLS and expect even more possibilities when considering different network topologies and learning rules. Among others, the inclusion of modulatory reward signals into our weight-based structural plasticity algorithm seems very promising. Reward modulated STDP has already been demonstrated on HICANN-DLS and should be easily combinable with structural reconfigurations (Wunderlich et al., 2019; Grübl et al., 2020). Moreover, the flexibility provided by the embedded PPU allows to extend the structural plasticity itself by e. g. additional pruning criteria such as bookkeeping (Spiess et al., 2016), spatial information (Bogdan et al., 2018), or silent synapses (Roy & Basu, 2016). However, it should be noted that the algorithm's locality is key for its efficiency.

The combinability of our weight-based structural plasticity algorithm allows us to alleviate the ubiquitous issue of limited synaptic resources. Most notably, the latter puts constraints on the overall size of multi-layer as well as recurrent neural networks implemented on the neuromorphic device at hand. Many of the networks and models implemented on the BrainScaleS platform in the past are likely to profit from a similar weight-based pruning and reassignment strategy as the one presented in the first part of this work (Schmitt et al., 2017; Kungl et al., 2018; Cramer et al., 2021; Göltz et al., 2021). This also includes the supervised learning frameworks presented in Chapter 5, demonstrated

on HICANN-X. Both of them required an off-chip downscaling of the MNIST images to accommodate the input data to a manageable neuronal fan-in on HICANN-X without constraining the overall network size too much. Since both learning rules assign high synaptic weights to informative synapses, our structural reconfiguration should be directly applicable. Thereby, the limitations on the network topology imposed by the neuromorphic device and the input dimensionality could be relaxed.

Structural plasticity has been demonstrated on various neuromorphic platforms. On digital systems, the event handling is per se based on look-up tables. Hence, most digital implementations centered around the on-the-fly adaptation of routing tables (Bogdan et al., 2018; Yan et al., 2019). While this approach is very flexible in general, it involves the reordering of connectivity lists for each removal and insertion of a synapse to keep look-up latencies at a minimum (Liu et al., 2018). Similar approaches have also been visited in the context of analog neuromorphic systems by utilizing optimized look-up matrices (Spiess et al., 2016; George et al., 2017; Bhaduri et al., 2018). While these representations are comparable to the on-chip synapse matrix of the BrainScaleS-2 architecture, they were stored and evaluated off-chip by external FPGAs. Because of this, weight dynamics as well as structural reconfigurations were carried out off-chip. It is noteworthy that the definition of sparse connectomes requires additional memory besides the actual synaptic weight to annotate all non-zero connections. Since the overall network graphs are reduced, this memory consumption is outweighed by the overall gains. The spatial footprint can be reduced by relying on dynamic random-access memory (DRAM) instead of SRAM to externally store the look-up tables. However, the associated access latencies can be detrimental for any system in general and accelerated systems in particular. In contrast, the partial in-synapse definition of the connectome within the BrainScaleS-2 architecture enabled our efficient on-chip implementation by only incorporating and adopting synapse-local quantities.

The accelerated nature of the BrainScaleS-2 systems allows bridging the gap in time scales between neuro-synaptic dynamics and structural plasticity evolution. In biological systems, the processes participating in synaptic rewiring evolve on time scales from hours to days and are hence prohibitively expensive for simulations (Lamprecht & LeDoux, 2004). The relatively long time scale of structural reconfigurations allows synapses to process sensory input and evolve accordingly to accumulate evidence of the information content prior to pruning or consolidation. Like for synaptic plasticity, the time scale of structural reconfigurations depends on the underlying time constants of the neuro-synaptic dynamics. Hence, the acceleration of BrainScaleS-2 directly translates to the implementation of structural reconfiguration and speeds up the emulation of bio-inspired plasticity models. Notably, the accelerated nature allowed us to gather large amounts of data for statistical processing.

Despite the limited system size of HICANN-DLS, our implementation is directly designed to scale well with the system size. This is due to the incorporation of synapse-local quantities only and further facilitated by the row-wise parallel update possibilities of the SIMD vector unit of the PPU which is also scaled with the system size (cf. Section 3.2). Especially on large systems, these scaling properties will become important when dealing with more complex network structures and synapse arrays of increased size. The associated enlarged physical fan-in of each neuron – i. e. the number of available synapses – results in an even larger virtual fan-in for the BrainScaleS-2 architecture and hence allows considering complex high-dimensional tasks in future implementations.

8 Conclusion

Within this thesis, we approached the optimization of spiking neural networks (SNNs) implemented on neuromorphic hardware for information processing. In that process, we visited the key steps necessary for the emulation of functional SNNs on analog devices. We started by finding suitable real-world benchmark tasks for SNNs and then moved over to learning algorithms – being supervised or unsupervised – and their implementation on two distinct mixed-signal neuromorphic chips of the BrainScaleS-2 family. While the aforementioned optimization schemes were limited to synaptic weights only, we lastly gave a glimpse of the efficient usage of limited resources, in particular a higher-level optimization of the fan-in of a neuromorphic system with local learning capabilities.

We started by considering benchmark tasks for the quantification of an SNN's ability to perform complex information processing. In that process, we designed a dataset that exhibits a natural temporal dimension, poses a minimum set of requirements on preprocessing, and most notably is freely available to the public. We came up with a collection of spoken digits – the Spiking Heidelberg Digits (SHD) – which we directly transformed to neuronal spike trains by a bio-inspired model of the ascending auditory pathway. Most notably, our dataset is not tailored to a specific problem or implementation and can be applied to SNNs irrespective of the substrate used for simulation or emulation, respectively. Currently, the SHD started to become established within the field and have already been visited in a broad range of applications: First, the performance of sparse (Perez-Nieves & Goodman, 2021) as swell as heterogeneous (Perez-Nieves et al., 2021) networks has been evaluated based on the SHD. Aside, also networks composed of novel neuron types have been visited (Yin et al., 2020). Moreover, novel learning mechanisms (Zenke & Vogels, 2021; Zenke & Neftci, 2021; Fang et al., 2021) and computing architectures (Fang et al., 2021; Eissa et al., 2021; Kan et al., 2021) have been benchmarked with the samples of the SHDs. All of these studies build on our first set of baseline performances obtained by optimizing a range of spiking as well as non-spiking classifiers in supervised learning paradigms.

With a suitable benchmark task for the assessment of the performance of SNNs at hand, we moved on to the optimization of SNNs emulated on different chips of the BrainScaleS-2 family. In more detail, we took two different approaches to the training of these networks which were developed in parallel. While the first one draws on the success of gradient-based methods and the backpropagation through time (BPTT) algorithm, the second one aims to exploit collective phenomena in SNNs. It is noteworthy that these approaches are not compellingly orthogonal to each other, but were developed simultaneously.

The first approach was presented in Chapter 5 where we invented an in-the-loop (ITL) training framework which aims to bring the success of supervised learning with gradient-based methods to the realm of analog neuromorphic hardware. The notion of surrogate gradients allowed us to overcome the binary nature of SNNs which precludes the application of vanilla gradient descent. In combination with the massively parallel recording of membrane traces on BrainScaleS-2, this

strategy facilitates the usage of gradient descent on a suitable loss function by performing BPTT. With this framework, we were able to train analog neuromorphic SNNs on a set of challenging benchmark tasks, including the newly established SHD dataset presented in Chapter 4. In particular, we were able to showcase state-of-the-art performance as well as new benchmarks in terms of low-latency classification and energy consumption for SNNs implemented on neuromorphic hardware.

The generality of the presented ITL training framework allows to easily extend the current implementation. Here, not only the training of neuronal parameters, but also the augmentation of existing dynamics by additional slow varying state variables seems very promising. The latter are thought to further bridge the gap between the time scales of neuro-synaptic dynamics and real-world stimuli. Moreover, these dynamics even allow to surpass the memory span of the already demonstrated recurrent SNNs and are likely to further stabilize these networks. Natural candidates are spike-triggered adaption as well as short-term plasticity (STP) (Mongillo et al., 2008; Bellec et al., 2018) which are both readily supported on the High Input Count Analog Neural Network X (HICANN-X) chip and can be easily included into the existing ITL training framework.

With our ITL training framework, we contributed to current efforts which try to bring the success of machine learning strategies to the field of SNNs in general and their implementation on specialized hardware systems in particular. The emerging solutions are especially important for edge computing applications where the training needs to be done once prior to deployment. Afterwards, the optimized networks are only used to perform inference. Nevertheless, learning and inference in biological tissue are not temporally separated and moreover, significantly differ from common machine learning strategies: In biological systems, the data is not presented in batches for long time spans with thousands of samples occurring in direct succession. Moreover, the learning happens more or less continuously and maybe more importantly lasts for the whole life. Thus, it might be promising to take one step back and draw again inspiration from the brain – not only in terms of the employed neuron and synapse models – to build more efficient training algorithms, both in terms of data requirements as well as resource demands.

While the surrogate gradient-based ITL framework is indeed very powerful and allows to include a vast range of bio-inspired mechanisms, it differs in important aspects from biological neural networks. Hence, we moved on to more biologically plausible unsupervised and local methods in Chapter 6. Specifically, we considered two distinct local learning rules which in combination with the input strength shape the dynamics of SNNs. For both of which, the regulated SNNs exhibit emerging autocorrelation times with a decreasing input strength. First, a spike-timing dependent plasticity (STDP) derived plasticity allowed us to exploited critical-like phenomena for computation. Here, the distance to a so-called critical point was adjusted by the input strength. With this framework, we demonstrated that the network dynamics have to be tuned to task requirements for optimal performance: While only complex and memory-intensive tasks profited from criticality, simple ones even suffered. Thereby, we do not only challenge the common assumption that criticality would be beneficial for any task, but, moreover, provide a general understanding of how collective phenomena in SNNs could be exploited for information processing. With a second local learning rule, we likewise demonstrated the emergence of autocorrelations. However, autocorrelations developed by this homeostatic regulation are generated by a bistable population activity.

So far, the proposed critical computing framework has only been applied to small, artificial problems. However, the ultimate goal should be the application to real-world benchmark tasks like the SHD. This would, however, require a notion of complexity for the classifications necessary to solve this problem. While we have seen that words exhibiting high phonemic similarity were harder to

classify for SNNs trained with the ITL approach presented in Chapter 5, it remains to show if a system posed at a critical point performs better in classifying these words than a sub-critical one. Here, the criticality hypothesis could be tested by classifying different pair-wise word combinations. While certainly not all pairings require critical dynamics to be solved accurately, a hierarchy of networks seems promising: Each constituent within this hierarchy should thereby be tuned to a different dynamical state to cover the full range of complexity levels. This scenario can be implemented either as a chain of successive networks or by instantiating multiple networks in parallel. By this, not only the criticality hypothesis could be tested in a real-world scenario, but also multi-chip systems running the constituents of the hierarchy in parallel could be benchmarked.

While the aforementioned two approaches of supervised and unsupervised learning were developed in parallel, their union has the potential to solve a couple of problems present in many deep architectures. Specifically, a preshaping of SNNs with the proposed unsupervised methods akin to supervised learning represents a promising direction of future research. While a comparable approach has been visited in the field of machine learning by Hochreiter & Schmidhuber (1997), the application to SNNs seems likewise promising. Here, the problem of vanishing gradients and thus suitable initialization of deep SNNs could be solved by prior application of unsupervised learning (Hochreiter, 1998). Thereby, not only activation of deeper layers could be ensured, but, likewise, the adjustment to the time scales of the considered task could be done. By this, the supervised preshaping has the potential to speed up the convergence of subsequent supervised training.

Within Chapter 6, we moreover demonstrated how the optimization could be designed in a hardware-aware fashion. To further advance the field of neuromorphic computing, the development of dedicated hardware systems and novel computing paradigms should not be considered as disjoint research areas, but a co-design should be targeted. Often, the efficiency of an approach could be dramatically increased by a lively exchange between both fields, potentially resulting in the incorporation of new features into existing neuromorphic architectures.

The reverse case, in which the design of a given hardware system inspired an efficient learning algorithm was outlined in Chapter 7. Within the scope of this chapter, we do not limit ourselves to the adaptation of synaptic weights, but, likewise, adjusted the topology of SNNs to overcome typical constraints present on neuromorphic hardware. Specifically, we proposed structural plasticity as a potential mechanism to deal with the ubiquitous issue of limited synaptic resources on neuromorphic devices. This resulted in an overarched optimization strategy applicable to both supervised as well as unsupervised learning paradigms. Our particular implementation on a BrainScaleS-2 prototype comes with low overhead and exploits the fact, that the connectome on BrainScaleS-2 is partially resolved locally within each synapse. Most notably, our strategy does not require knowledge of the distribution of samples in the feature space. Instead, it exploits the accelerated nature of BrainScaleS-2 to perform synaptic sampling to find informative receptors, thereby leading to the self-organized formation of receptive fields.

The overarched nature and the fully local execution allow us to combine our structural plasticity strategy with other existing implementations emulated on the BrainScaleS-2 system. In particular, it would be very promising to combine this concept with the ITL learning framework presented in Chapter 5. Here, the ubiquitous issue of a limited synaptic fan-in necessitated the pre-processing of MNIST images as well as the merging of multiple neuron circuits to larger logical entities. In general, it is desirable to inject the full-size images and instead apply structural reconfigurations. This would not only lead to more comparable performance measures, but would furthermore render a data distribution aware preprocessing – like the discarding of the outermost pixels as well as the

zooming – redundant. Thereby, the proposed experiment builds an important step towards more autonomous and self-organizing systems.

The accelerated nature of BrainScaleS was key for all presented optimization approaches. Particularly, the acceleration led to high throughput during inference for SNNs trained with our ITL learning framework. It is noteworthy that for real-time applications the time scales of the problem should match the scale of the accelerated dynamics. The on-chip implementations profited most due to the reduction of I/O and the hence uninterrupted emulation. For neuro-scientific experiments, the speedup allowed us to gather large amounts of data or to investigate learning over long time spans, thereby bridging the gap between time scales of neuro-synaptic dynamics and slower learning processes. Hence, the temporal efficiency was visible for all presented experiments and renders our approaches fruitful.

For all topics, an efficient emulation on BrainScaleS-2 was targeted. Especially within the scope of the last chapter, we highlighted the importance of efficient hardware usage even at a small scale. The limited size of the BrainScaleS-2 prototype naturally enforced implementations taking into account hard constraints which would presumably not be the case for small-scale software simulations. When one of the ultimate goals of neuromorphic computing is the emulation of brain-scale experiments on very-large-scale integration (VLSI) systems, locality and thus scalability should be reconsidered carefully even at the small scale. In particular, the investigating of computational capabilities within small-scale networks prior to the actual scale-up might promote more efficient resource usage.

A Appendix

A.1 Auditory modelling

This summary of established auditory models is part of the appendix of the publication Cramer et al. (2020b) which has been presented in the scope of Chapter 4. For this segment I will closely follow the publication, but provide additional details about the functionality and impact of each model stage.

In the following, we depict the models involved in the conversion of audio time series data to spikes. To that end, we draw on a chain of three established models mimicking basic aspects of the ascending auditory pathway: First, we apply a hydrodynamic basilar membrane (BM) model which causes spatial frequency dispersion (Appendix A.1.1). Second, the BM movement is converted to firing rates by means of a biologically motivated hair cell (HC) model (Appendix A.1.2). Neural events are then obtained by drawing Poisson distributed spikes from these temporal firing rates and imposing a simple refractory effect. Last, a layer of bushy cells (BCs) increases phase-locking and the overall sparsity of the final spike trains (Appendix A.1.3).

A.1.1 Basilar membrane model

The physics of the inner ear are mainly determined by the interaction of the BM with the fluid within the cochlea – the perilymph – and can hence be described by hydrodynamics. A comprehensive consideration of hydrodynamic BM models and their derivation is beyond the scope of this section. For extensive reviews, the reader is referred to the work of de Boer (1980) and de Boer (1984). Here, we will closely follow the work done by Sieroka et al. (2006) and highlight the key steps required for the derivation of a long-wave analytical solution that has been applied for the conversion of both the Spiking Heidelberg Digits (SHD) as well as the Spiking Speech Commands (SSC).

The aforementioned interaction of the BM and the fluid leads to the induction of spatial frequency dispersion (Sieroka et al., 2006; de Boer, 1980, 1984). In the following, we will assume the simplified cochlea geometry depicted in Figure A.1a which captures key mechanical features. For simplicity, we assume the fluid to be inviscid as well as incompressible and the oscillations to be small so that the fluid can be described as linear. Within this framework, the BM can be described by its mechanical impedance $\xi(x, \omega)$ which is a function of the position in the x-direction (cf. Figure A.1a) and the angular frequency $\omega = 2\pi\nu$:

$$\xi(x, \omega) = \frac{1}{i\omega}\left[S(x) - \omega^2 m + i\omega R(x)\right] . \tag{A.1}$$

Here, we introduced a transversal stiffness $S(x) = C_0 e^{-\alpha x} - a$, a resistance $R(x) = R_0 e^{-\alpha x/2}$ as well as an effective mass m (de Boer, 1980). An associated damping constant of the BM is given by

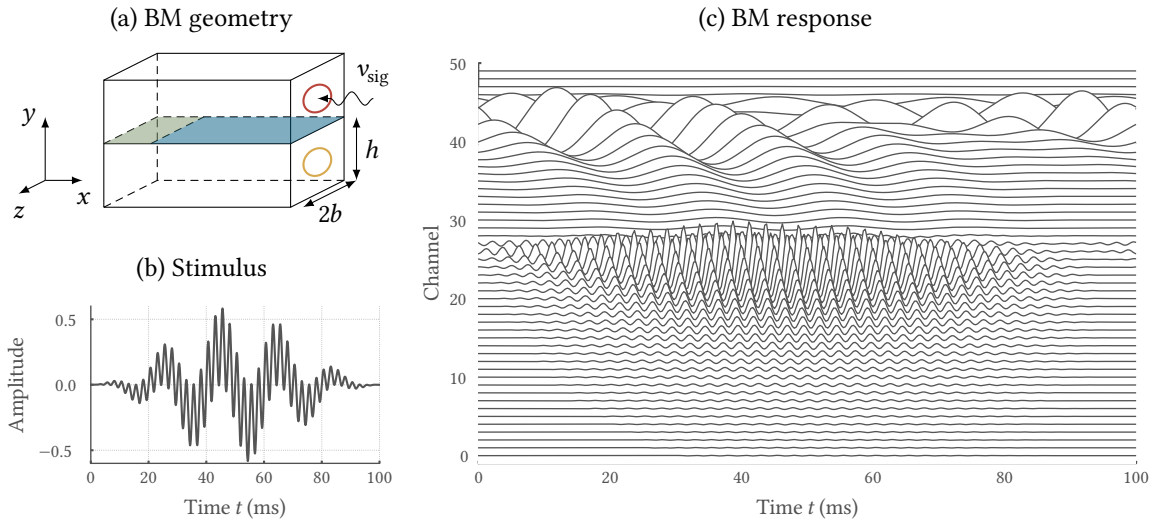


Figure A.1: Illustration of the BM model. (**a**) Schematic view of the BM model. The BM (blue) separates the *scala tympani* (lower chamber) from the *scala vestibuli* (upper chamber). At the *helicotrema* (green), the two scalae are connected. The *scala tympani* ends in the *round window* (yellow). A sound wave v_{sig} is penetrating the eardrum, applying pressure at the *oval window* (red) by moving the *ossicles*, leading to a compression and slower traveling wave. We have neglected the *scala media* (Sieroka et al., 2006) and consider a stretched form. (**b**) The BM model is exposed to a superposition of two sine waves with frequencies of 50 Hz and 400 Hz. 30 ms Hann windows are applied to the start and end of the signal, respectively. (**c**) The BM model causes spatial frequency dispersion. Being exposed to the superposition of sine waves, the velocity of the BM in y-direction clearly shows two spatially distinct areas of resonating behavior. Here, we show the response of the model at 50 distinct positions on the BM for simplicity. Panel (a) as well as its caption are taken from Cramer et al. (2020b).

$\gamma = R_0/\sqrt{C_0 m}$. The encoding of the whole range of human audible frequencies from about 20 Hz to 20 kHz (Purves et al., 2001) into BM movements is facilitated by variations of the stiffness $S(x)$ over several orders of magnitude.

In the following, we denote the difference in pressure between the upper and lower chamber in Figure A.1a by $p(x, \omega)$. The expression:

$$p(x,\omega) = \sum_n \int_0^\infty \frac{\mathrm{d}k}{2\pi} e^{-ikx} p_0(k) \left[\frac{\cosh(m_0(h-y))}{\cosh(m_0 h)} + \frac{m_0 \tanh(m_0 h)\cosh(m_1(h-y))}{m_1 \tanh(m_1 h)\cosh(m_1 h)} \cdot \cos\left(\frac{\pi z n}{b}\right) \right], \quad \text{(A.2)}$$

fulfills the boundary conditions $v_y = 0$ for $y = h$, and $v_z = 0$ at $z = \pm b$ of our assumed simplified geometry (de Boer, 1980). By resorting to the Laplace equation, we obtain the relations $m_0 = k$ and $m_1 = \sqrt{k^2 + \pi^2/b^2}$. Throughout the remainder of this section, we will only consider $n = 1$ and hence stick to the principal mode of excitation in the z-direction. Combining all of these assumptions, we obtain an expression for the y-component of the velocity in the middle of the BM by applying the Euler equation:

$$\partial_y p(x,\omega) = -i\omega\rho v_y(x,\omega) = \frac{2i\omega\rho}{\xi(x,\omega)} p(x,\omega)\,. \quad \text{(A.3)}$$

For readability, we excluded the y and z arguments. In order to obtain an analytical solution for $v_y(x, \omega)$, we restrict our consideration to the limiting case of long waves with $kh \ll 1$. Applying this

limit to the Equations (A.2) and (A.3) yields:

$$\partial_x^2 p(x,\omega) = \frac{i\omega\rho}{h\xi(x,\omega)} p(x,\omega)\,, \tag{A.4}$$

where we furthermore accomplished the replacement $\hat{p}(k) \to p(x)$, $k \to i\partial_x$ and $k^2 \to \partial_x^2$ with the Fourier transform $\hat{p}(k)$ of $p(x,0,0)$. This equation is approximately solved by:

$$p(x,\omega) = \sqrt{\frac{G(x,\omega)}{g(x,\omega)}} H_0^{(2)}\left(G(x,\omega)\right)\,, \tag{A.5}$$

with the second Hankel function $H_0^{(2)}$. The functions $g(x,\omega)$ and $G(x,\omega)$ are determined by:

$$g(x,\omega) = \omega\sqrt{\frac{\rho}{h\xi(x,\omega)}}\,, \tag{A.6}$$

$$G(x,\omega) = \int_0^x dx' g(x',\omega) + \frac{2}{\alpha} g(0,\omega)\,. \tag{A.7}$$

It is noteworthy that $G(x,\omega)$ can also be solved analytically. A closed-form solution was derived by Sieroka et al. (2006). The stimulation by an incidental sound wave with Fourier transformation $v_{\text{sig}}(\omega)$ can be done by evaluating:

$$v_y(x,t) = \int \frac{d\omega}{2\pi} i Z_{\text{in}} \frac{v_y(x,\omega)}{p(0,\omega)} e^{-i\omega t} v_{\text{sig}}(\omega)\,. \tag{A.8}$$

Here, we modelled the input impedance of the cochlea by:

$$Z_{\text{in}}(\omega) = \frac{p(x=0)}{v_x(x=0)} \approx \sqrt{\frac{2C_0}{h}} \frac{iJ_0(\zeta) + Y_0(\zeta)}{J_1(\zeta) - iY_1(\zeta)}\,, \tag{A.9}$$

where $J_\beta(\zeta)$ and $Y_\beta(\zeta)$ denote the Bessel functions of first and second kind of order β. Furthermore, we introduced $\zeta = 2\omega/\alpha\sqrt{2/(hC_0)}$.

The nature of the BM to cause spatial frequency dispersion can be illustrated by exposing the model to a superposition of two sine waves (Figure A.1b). Here, the BM movement reveals two clear resonance regions which are spatially distinct (Figure A.1c). Before applying the BM model to a stimulus, each signal $v_{\text{sig}}(t)$ was normalized to a root mean square (RMS) value of $0.3\,\text{cm}\,\text{s}^{-1}$. Afterwards, the BM model was evaluated in the spatial range $x \in (0, 3.5\,\text{cm}]$ in N_{ch} linearly spaced steps.

A.1.2 Hair cell model

The movements of the BM are converted to neural events by the inner HCs via mechanotransduction. In more detail, a sequence of N_{ch} HCs is located along the x direction of the BM (Figure A.3a). While the human cochlea contains about $N_{\text{ch}} = 3500$, we chose $N_{\text{ch}} = 700$ for our benchmark datasets to reduce the memory as well as computational footprint of our benchmarks and model, respectively. At the same time, this choice still provides an accurate representation of auditory stimuli. In the following, we highlight the key steps of the HC model invented by Meddis (1986), to which we refer for further details.

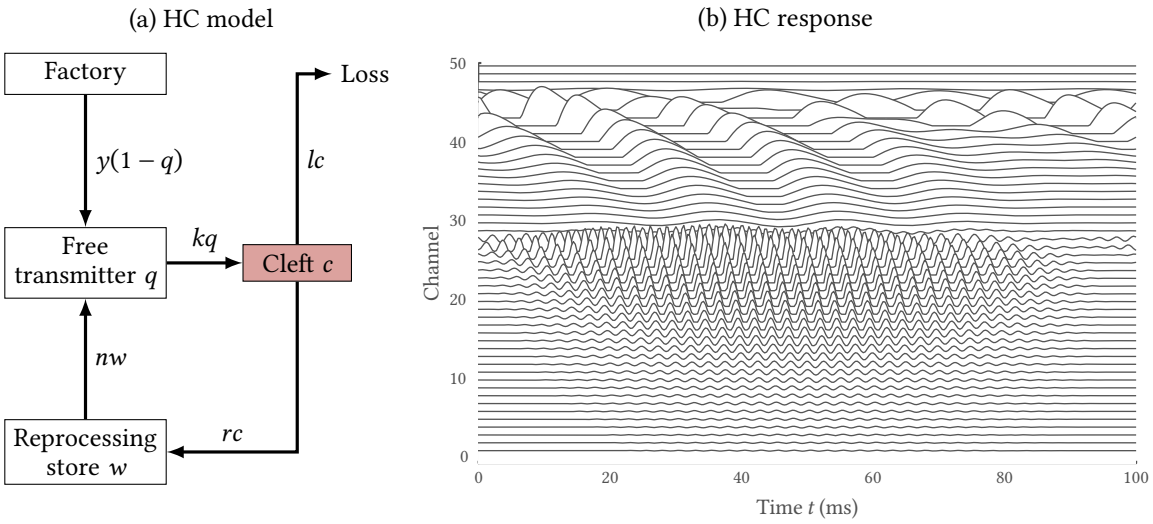


Figure A.2: Illustration of the HC model. (**a**) Schematic view of the transmitter flow within the HC model Figure adapted from Meddis (1986). The model comprises four transmitter pools which allow the description of the transmitter concentration in the synaptic cleft. (**b**) The HC model causes a half-wave rectification as well as a compression of its stimuli. Shown is the model response to the BM velocity elicited by the superposition of sine waves depicted in Figure A.1. Panel (a) as well as its captions are taken from Cramer et al. (2020b).

Within the formulation of this model, the state of a HC is determined by the dimensionless transmitter concentration in various stages. Each cell contains a specific amount of free transmitter molecules $q(x,t)$ which can be released into the synaptic cleft through a semi-permeable membrane (Figure A.2a). The permeability depends on the movement of the BM and is in our description directly based on the BM's velocity in y-direction, $v_y(x,t)$:

$$k(x,t) = \begin{cases} \frac{g\cdot[c\cdot v_y(x,t)+A]}{c\cdot v_y(x,t)+A+B} & \text{for } v_y(x,t)+A>0 \\ 0 & \text{else} \end{cases} . \tag{A.10}$$

Here, we introduced a maximum permeability g, the input scaling c, a permeability offset A as well as the permeability rate B. It is noteworthy that there is a constant leak of transmitter even in the absence of mechanical stimulation. Further, only the decomposition in positive y-direction increases the permeability, thereby capturing the half-wave rectification carried out by biological HCs. Once being released into the synaptic cleft, the amount of transmitter molecules is continuously reduced by chemical destruction or loss through diffusion $l\cdot c(x,t)$. Further, a fraction of transmitter $r\cdot c(x,t)$ is reuptaken back into the HC. Hence, the total amount of transmitter in the cleft $c(x,t)$ evolves according to:

$$\frac{\mathrm{d}c}{\mathrm{d}t} = k(x,t)q(x,t) - l\cdot c(x,t) - r\cdot c(x,t)\,. \tag{A.11}$$

The reuptaken transmitter molecules are cached in a reprocessing store from where they are continuously transferred to the free transmitter pool at a rate $n\cdot w(x,t)$. With this, the concentration in the reprocessing store $w(x,t)$ is governed by:

$$\frac{\mathrm{d}w}{\mathrm{d}t} = r\cdot c(x,t) - n\cdot w(x,t)\,. \tag{A.12}$$

Aside from reuptake, the free transmitter pool gets replenished at a rate $y[1-q(x,t)]$ by a manufac-

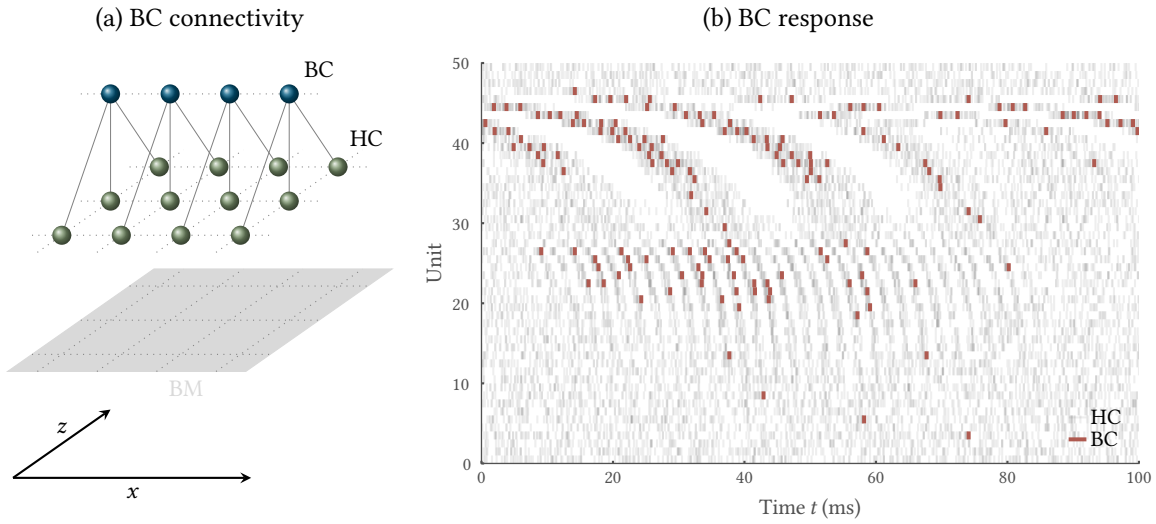


Figure A.3: Schematic view of the connectivity within the auditory model. **(a)** Multiple HCs (green circles) are located at each position x of the BM (gray area). Each of these HCs at position x projects onto a single BC (blue circles). **(b)** The integration of multiple spike trains emitted by the HCs at each position by a BC increases the phase-locking to stimulus features and at the same time reduces the overall amount of spikes in the elicited response. Shown are the spikes of a single exemplary HC at each position x in response to the BM velocity elicited by the superposition of sine waves depicted in Figure A.1 (gray). Superimposed is the response of a layer of BCs, each of which stimulated by 40 independent HCs (red).

turing base. Summing all contributions, the concentration in the free transmitter pool becomes:

$$\frac{\mathrm{d}q}{\mathrm{d}t} = y[1 - q(x,t)] + n \cdot w(x,t) - k(x,t)q(x,t)\,. \tag{A.13}$$

The emission of spikes is based on the concentration of the transmitter in the synaptic cleft. While being in the cleft, transmitter quanta could open post-synaptic ion channels to elicit post-synaptic potentials (PSPs) with a probability $p_{\mathrm{spike}} = h \cdot c(x,t)\delta t$, where we introduced the time step δt used to evaluate the model. Like for regular leaky integrate-and-fire (LIF) neurons, we impose a simple refractory period τ_{ref} by denying any spike which would occur after a previous event with a temporal separation of less than 1 ms. Revisiting the example of the superposition of sine waves, the compression and half-wave rectifying effect of the HC becomes apparent (Figure A.2).

A.1.3 Bushy cell model

We increased the phase-locking of the output spikes to stimulus features by a population of N_{ch} BCs (Figure A.3a). To that end, we simulated $N_{\mathrm{hc}} = 40$ independent HCs at each position x of the BM to capture sufficient statistics from the spike times about p_{spike}. Each of these N_{hc} HCs projects onto a single BC. In contrast to Rothman et al. (1993), we implemented the BCs as standard LIF units and not as detailed Hodgkin-Huxley neurons. All synaptic weights between the HCs and the layer of BCs were set equally to a value of $w_{ij} = 0.54/N_{\mathrm{hc}}\ \forall i,j$. Despite the noisy input spike trains emitted by each of the 40 HCs individually (Figure A.2b), the BCs accurately recover the phase relation (Figure A.3b) of the stimulus (Figure A.1a). Moreover, the spike trains emitted by the BCs are characterized by a higher sparsity level expressed in both space and time.

A.2 Time-to-first-spike coding

The information given in this section is part of the publication Göltz et al. (2021) whereof the results achieved with an implementation on HICANN-X have been presented in the scope of Chapter 5. In the following, we present the basic steps of the derivation of the learning rule as well as details concerning the implementation on HICANN-X.

The spike-timing-based code highlighted in the following ties on the ideas of Mostafa (2017). In a time-to-first-spike (TTFS) coding, the time elapsed between stimulus onset and the first firing time of a neuron signals the presence of a feature within a stimulus. In this context, an earlier spike time highlights a more manifested feature. Hence, this coding scheme enables efficiency in terms of both time-to-decision as well as energy-to-solution. In the following, we present the utilized methods for the results obtained by networks employing TTFS coding within the scope of Chapter 5. To that end, we closely follow the basic steps of Göltz et al. (2021), to which we refer for further details. We start by formulating the used TTFS loss function on which we performed gradient descent. Moreover, we highlight the key steps of the derivation of an analytical differentiable solution of the first-spike time T which facilitates the formulation of a learning rule based on the aforementioned loss function. We close by a description of the in-the-loop (ITL) training used to optimized multi-layer networks employing TTFS coding on the High Input Count Analog Neural Network X (HICANN-X) chip.

A.2.1 Loss function

To finally optimize spiking neural networks (SNNs) with gradient descent and the backpropagation algorithm, we define a TTFS loss function which is differentiable with respect to both synaptic weights as well as spike times. During the process of training, this loss function aims to maximize the temporal difference between the spike emitted by the correct and all remaining label neurons. More specifically, we consider the TTFS loss function:

$$\mathscr{L} = \log\left[\sum_i \exp\left(-\frac{t_i^{(L)} - t_{i^\star}^{(L)}}{\xi\tau_{\text{syn}}}\right)\right], \tag{A.14}$$

with the index of the correct label neuron $i^\star$ and a positive scaling parameter ξ. Here, the spike time of neuron i in layer l is denoted by $t_i^{(l)}$ with the special case $l = L$ for the readout layer. Note that we skipped the spike index since we restrict our networks to only emit a single spike per neuron. Here, we define $t_i^{(l)} = \infty$ for silent neurons which, however, renders the closed-form solution for T as well as its derivates ill-defined.

In order to enforce spiking activity, we hence increased the input weights of all silent neurons by Δw_{bump} in case the number of silent neurons exceeds a certain threshold p_{silent}. For multi-layer networks, we only applied this strategy to the first layer falling below this threshold. This weight increase was chosen exponentially in case the neurons in a layer remain silent over multiple, successive updates. To enhance training, we furthermore added a regularization term to the overall loss function:

$$\mathscr{L}_{\text{reg}} = \rho_\alpha\left[\exp\left(\frac{t_{i^\star}^{(L)}}{\beta\tau_{\text{syn}}}\right) - 1\right], \tag{A.15}$$

with the regularization strength ρ_{reg} and a positive scaling parameter β. This penalty rewards the correct label neuron to emit even earlier spike times.

A.2.2 Weight updates

Gradient descent can now be performed on the loss function in Equation (A.14) by iteratively applying the chain rule:

$$\Delta w_{ij}^{(l)} = -\eta \frac{\partial \mathscr{L}}{\partial w_{ij}^{(l)}} = -\frac{\partial t_i^{(l)}}{\partial w_{ij}^{(l)}} \underbrace{\frac{\partial \mathscr{L}}{\partial t_j^{(l)}}}_{\delta_j^{(l)}} = \frac{\partial t_i^{(l)}}{\partial w_{ij}^{(l)}} \sum_k \frac{\partial t_k^{(l+1)}}{\partial t_j^{(l)}} \delta_k^{(l+1)}, \tag{A.16}$$

with a learning rate η which was chosen to decay over the course of training with a decay rate γ_η and a step size of γ_{step}. The emerging derivatives can be calculated analytically which in turn requires an analytical solution for the first spike time of neuron i, T_i. Due to the interest in single spikes only, we can drop the spike count index to obtain a solution of the LIF equation in Equation (2.12) for the neuron i:

$$u_i(t) = \frac{1}{C_{\text{mem}}} \frac{\tau_{\text{mem}}\tau_{\text{syn}}}{\tau_{\text{mem}} - \tau_{\text{syn}}} \sum_j w_{ij}\theta(t - t_j) \left[\exp\left(-\frac{t - t_j}{\tau_{\text{mem}}}\right) - \exp\left(-\frac{t - t_j}{\tau_{\text{syn}}}\right) \right]. \tag{A.17}$$

Here, the sum extends over all presynaptic partners j and the PSPs take the form of a difference of exponentials. For the special case $\tau_{\text{mem}} \to \tau_{\text{syn}}$, we obtain with l'Hôpital's rule:

$$u_i(t) = \frac{1}{C_{\text{mem}}} \sum_j w_{ij}\theta(t - t_j)(t - t_j) \exp\left(-\frac{t - t_j}{\tau_{\text{syn}}}\right), \tag{A.18}$$

i. e. alpha-shaped PSPs. By setting $u(T) = u_{\text{thres}}$ and relying on the Lambert W function $\mathscr{W}$, we get for the first-spike time:

$$T_i = \tau_{\text{syn}} \left\{ \frac{b_i}{a_i} - \mathscr{W}\left[-\frac{g_{\text{leak}} u_{\text{thres}}}{a_i} \exp\left(\frac{b_i}{a_i}\right) \right] \right\}, \tag{A.19}$$

where we defined the functions:

$$a_i := \sum_{j \in C} w_{ij} \exp\left(\frac{t_j}{\tau_{\text{syn}}}\right), \qquad b_i := \sum_{j \in C} w_{ij} \frac{t_j}{\tau_{\text{syn}}} \exp\left(\frac{t_j}{\tau_{\text{syn}}}\right), \tag{A.20}$$

with the causal set $C = \{j | t_j < T_i\}$ (Göltz et al., 2021). In case a spike occurred, the spike threshold u_{thres} is crossed two times: first from below due to the rising of u_i caused by incoming PSPs and second from above, when u_i decays back to the leak potential u_{leak}. In general, we are interested in the earlier of the solutions of Equation (A.19). This corresponds to the branch that returns the larger $\mathscr{W}$, i. e. $\mathscr{W} = b_i/a_i - T_i/\tau_{\text{syn}} > -1$.

The above expression for T_i is differentiable with respect to both synaptic weights as well as presynaptic spike times. By adopting the nomenclature of multi-layer networks, we obtain the following derivatives (Göltz et al., 2021):

$$\frac{\partial t_i^{(l)}}{\partial w_{ij}^{(l)}} = -\frac{1}{a_i} \frac{\exp\left(\frac{t_j^{(l-1)}}{\tau_{\text{syn}}}\right)}{\mathscr{W}(z) + 1} (t_i^{(l)} - t_j^{(l-1)}), \tag{A.21}$$

$$\frac{\partial t_i^{(l)}}{\partial t_j^{(l-1)}} = -\frac{1}{a_i} \frac{\exp\left(\frac{t_j^{(l-1)}}{\tau_{\text{syn}}}\right)}{\mathscr{W}(z) + 1} \frac{w_{ij}}{\tau_{\text{syn}}} (t_i^{(l)} - t_j^{(l-1)} - \tau_{\text{syn}}), \tag{A.22}$$

$$\tag{A.23}$$

where the argument of the Lambert W function is defined by:

$$z := -\frac{g_{\text{leak}} u_{\text{thres}}}{a_i} \exp\left(\frac{b_i}{a_i}\right). \tag{A.24}$$

Hence, these expressions allow to solve the credit assignment problem and thus facilitate the exact error propagation through multi-layer SNNs employing TTFS coding. We finally obtain a synaptic update rule by inserting both derivatives into Equation (A.16).

A.2.3 Datasets

In order to stimulate the networks of LIF neurons, the input data has to be transformed to spikes. Irrespective of the dataset, this was done by defining an earliest and a latest spike time, t_{early} and t_{late}, respectively. The input values were then mapped linearly to the time interval $[t_{\text{early}}, t_{\text{late}}]$.

As a first benchmark, we consider data points sampled from the Yin-Yang figure. In more detail, each point is given by a pair of Cartesian coordinates $(x, y) \in [0, 1]^1$ and the mirrored coordinates $(1-x, 1-y)$ to capture the symmetry of the data and to enable training without additional bias term (Kriener et al., 2021). All of these coordinates were converted to spike times by the method described above and joined with an additional bias spike. The resulting five spike trains – each comprising of a single spike – were used to classify the data into the three categories Yin, Yang and Dot. In order to accommodate these stimulating spike trains to the input strength and weight resolution of HICANN-X, each logical spike train was replicated five times to finally result in a total of 25 spike trains.

The second dataset consists of downscaled MNIST images with a size of 16×16 pixels. By downscaling, we were able to accommodate the input dimensionality to a tractable fan-in on HICANN-X. Subsequently, the pixel intensities were first normalized and then linearly mapped to spike times, resulting in a total of 256 stimulating spike trains each of which containing a single spike time. Furthermore, we added Gaussian noise with standard deviation σ_t to these input spike times to reduce overfitting.

A.2.4 In-the-loop training

SNNs emulated on HICANN-X were trained ITL. To that end, the spikes emitted by HICANN-X were used to calculate the weight updates according to Equations (A.16), (A.22) and (A.23) on the host computer. Moreover, we imposed a maximum weight update Δw_{max}. The HICANN-X chip was calibrated to yield a comparable behavior as software simulations. Specifically, the PSP size as well as the produced spike times of hard- and software implementation were matched. By a high refractory period τ_{ref}, we ensured that most neurons only spiked once during the presentation of an input sample. In more detail, we configured $\tau_{\text{ref}} > \tau_{\text{syn}}$ in order to let all synaptic currents decay and to irradiate the associated information.

A.3 Partial information decomposition

Often, the information within e. g. a neural network is encoded by an ensemble of agents. Knowledge about the distribution of information in these ensembles may guide the design of both the

Table A.1: Overview of the parameters used for SNNs employing TTFS coding. All parameters are given in the hardware time domain. Mean and standard deviation of the weight initialization are given separately for the weights connecting the input with the hidden layer as well as the weights between hidden and readout layer. Table adapted from Göltz et al. (2021).

Stage	Parameter	Symbol	Value	
			Ying-Yang	MNIST
Neuro-synaptic dynamics	Leak potential	u_{leak}	550 mV	550 mV
	Threshold potential	u_{thres}	850 mV	850 mV
	Membrane time constant	τ_{mem}	6 µs	10 µs
	Synaptic time constant	τ_{syn}	6 µs	10 µs
Network	Input size	N_{I}	25	256
	Hidden layer size	N_{H}	120	246
	Label layer size	N_{L}	3	10
	Mean weight initialization	$\hat{\mu}_w$	0.1/0.075	0.01/0.006
	Stdev weight initialization	$\hat{\sigma}_w$	0.12/0.15	0.03/0.1
Input	Input noise	σ_t	–	0.3 µs
	Bias time	t_{bias}	$0.9\,\tau_{\text{syn}}$	–
	Earliest possible spike time	t_{early}	$0.15\,\tau_{\text{syn}}$	$0.15\,\tau_{\text{syn}}$
	Latest possible spike time	t_{late}	$2.0\,\tau_{\text{syn}}$	$2.0\,\tau_{\text{syn}}$
Optimization	Training epochs	N_{epochs}	400	50
	Batch size	N_{batch}	40	50
	Learning rate	η	2.0×10^{-3}	3.0×10^{-3}
	Learning rate decay step	γ_{step}	20	10
	Learning rate decay	γ_η	0.05	0.1
	First moment estimates decay rate	β_1	0.9	0.9
	Second moment estimates decay rate	β_2	0.999	0.999
	Stability parameter	ϵ	1.0×10^{-8}	1.0×10^{-8}
	Maximum allowed update	Δw_{max}	0.2	0.2
	Weight bump value	Δw_{bump}	5×10^{-4}	5×10^{-3}
	Latency regularization	ρ_α	5×10^{-3}	5×10^{-3}
	Loss scaling parameter	ξ	0.2	0.2

ensembles as well as the higher-level network. In this context, partial information decomposition (PID) allows the quantification of the unique and redundant contribution of a set of source variables to a target and, moreover, facilitates the assessment of synergistic computation. In the following, we will consider the most simple case of two input variables X_1 and X_2 as well as a single output variable Y and closely follow the steps of Wibral et al. (2015) to motivate the assumptions made for the estimation of PID components within the scope of Chapter 6.

Classical information theory provides us with a measure of mutual information which could be applied to different pairings of variables: First, each of the inputs individually provides information about the target specified by the mutual information $\mathrm{I}(Y : X_i)$. Second, both input variables jointly contain information about the target as given by the joint mutual information $\mathrm{I}(Y : X_1, X_2)$. However, classical information theory does not provide a closed-form expression for the information contained in the joint input variables about the target that cannot be obtained from each source separately. This is termed *synergistic information*. Further, we lack an expression for the information carried by only one input variable about the target that can not be obtained from the other one. The latter is referred to as *unique information*. Last, we have no notion of *shared information* that is the information contained within one input variable about the target that can also be obtained by looking at the other input variable alone. The estimation of these contributions is the subject of PID (Williams & Beer, 2010; Bertschinger et al., 2013; Lizier et al., 2018).

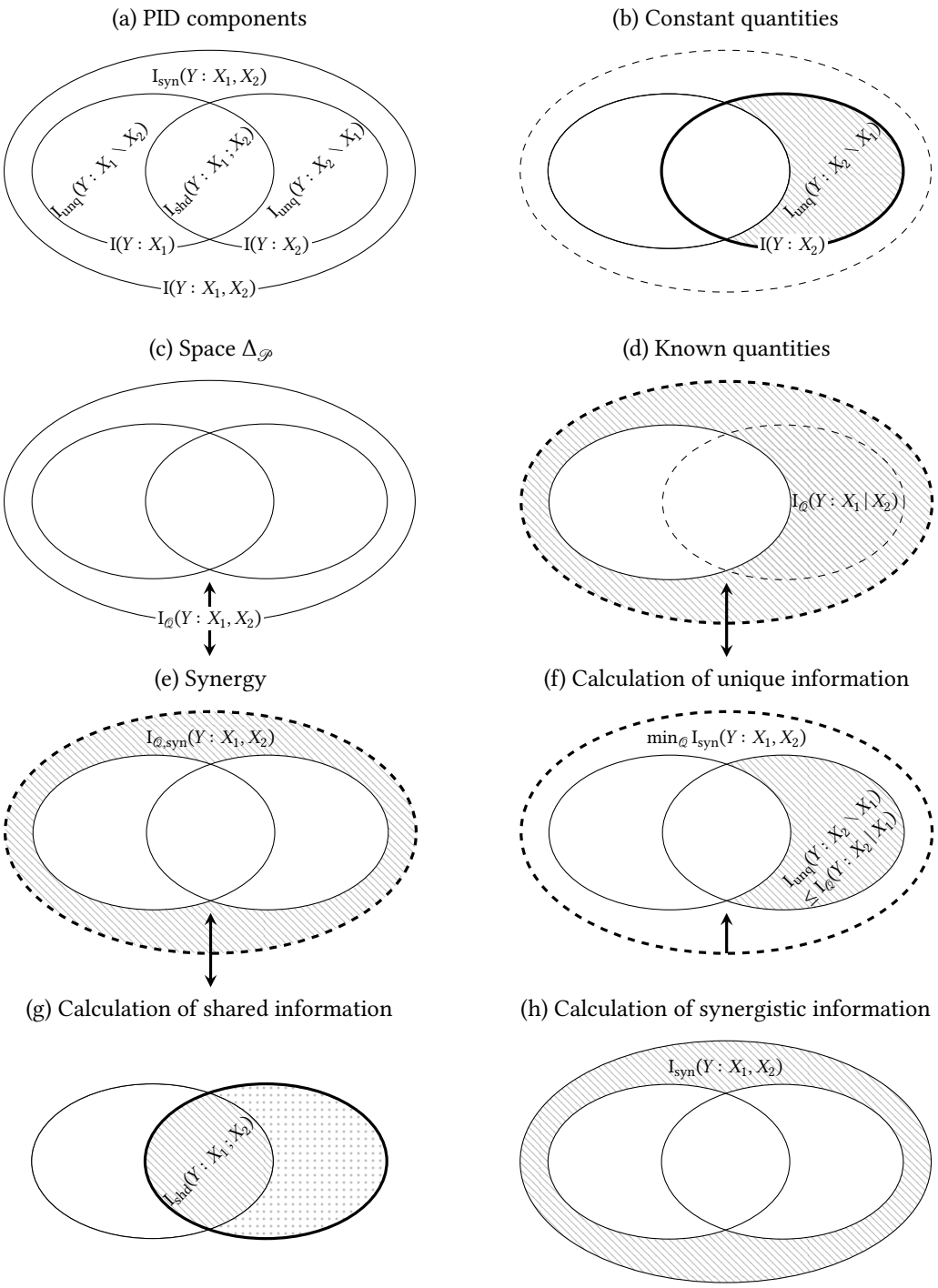


Figure A.4: Estimation of unique information. **(a)** Illustration of the constituents of a PID of the joint mutual information $I(Y : X_1, X_2)$. The synergistic information I_{syn} and the shared information I_{shd} can be calculated in case the unique information I_{unq} is given. **(b)** The mutual information $I(Y : X_2)$ is defined in classical information theory and constant on $\Delta_{\mathscr{P}}$ $I_{unq}(Y : X_2 \setminus X_1)$ is unknown, but constant on $\Delta_{\mathscr{P}}$ by assumption. **(c)** In order to estimate $I_{unq}(Y : X_1, X_2)$, $I_{\mathcal{Q}}(Y : X_1, X_2)$ is defined on $\Delta_{\mathscr{P}}$ which depends on $\mathcal{Q}$. **(d)** $I_{\mathcal{Q}}(Y_1 : X_2 \mid X_1)$ is defined in classical information theory, but depends on $\mathcal{Q}$. **(e)** $I_{\mathcal{Q},syn}(Y_1 : X_1, X_2)$ equally depends on $\mathcal{Q}$. **(f)** By minimizing $I_{\mathcal{Q},syn}(Y_1 : X_1, X_2)$ an upper estimate for $I_{unq}(Y : X_2 \setminus X_1)$ can be obtained. **(g)** With the definition of $I_{unq}(Y : X_2 \setminus X_1)$, $I_{shd}(Y : X_1 ; X_2)$ can be calculated from $I(Y : X_2)$. **(h)** Finally, $I_{syn}(Y_1 : X_1, X_2)$ can be obtained from $I(Y : X_1, X_2)$, $I_{unq}(Y : X_2 \setminus X_1)$, $I_{unq}(Y : X_1 \setminus X_2)$ and $I_{shd}(Y : X_1 ; X_2)$. Figure inspired by Wibral et al. (2015).

Let us start by considering the decomposition of the joint mutual information $\mathrm{I}(Y : X_1, X_2)$ between two input variables X_1 and X_2 and one output variable Y. First, the total information of each input variable about the target should be decomposable into the unique information $\mathrm{I}_{\mathrm{unq}}$ and the shared information $\mathrm{I}_{\mathrm{shd}}$:

$$\mathrm{I}(Y : X_1) = \mathrm{I}_{\mathrm{shd}}(Y : X_1 ; X_2) + \mathrm{I}_{\mathrm{unq}}(Y : X_1 \setminus X_2)\,, \tag{A.25}$$

$$\mathrm{I}(Y : X_2) = \mathrm{I}_{\mathrm{shd}}(Y : X_2 ; X_1) + \mathrm{I}_{\mathrm{unq}}(Y : X_2 \setminus X_1)\,. \tag{A.26}$$

Similarly, the two unique information terms, the shared information and the synergistic information $\mathrm{I}_{\mathrm{syn}}(Y : X_1 ; X_2)$ should sum up to the joint information $\mathrm{I}(Y : X_1, X_2)$:

$$\mathrm{I}(Y : , X_1, X_2) = \mathrm{I}_{\mathrm{unq}}(Y : X_1 \setminus X_2) + \mathrm{I}_{\mathrm{unq}}(Y : X_2 \setminus X_1) + \mathrm{I}_{\mathrm{shd}}(Y : X_1 ; X_2) + \mathrm{I}_{\mathrm{syn}}(Y : X_1 ; X_2)\,. \tag{A.27}$$

With regard to Figure A.4, it becomes apparent that none of these terms can be obtained by a combination of classical mutual information terms. However, if one of the components can be estimated, all the others can be computed in succession. Williams & Beer (2010) proposed a set of axioms based on a set of source variables $X_1, X_2, ..., X_n$ and a target Y to obtain a functional definition of shared information:

1. The shared information of the variables $X_1, X_2, ..., X_n$ about Y is symmetric under permutations of the variables $X_1, X_2, ..., X_n$.
2. The mutual information $\mathrm{I}(Y : X_1)$ equals the shared information that X_1 shares with itself about Y.
3. The shared information of the variables $X_1, X_2, ..., X_n$ about Y is smaller than or equal to the redundant information that the variables $X_1, X_2, ..., X_{n-1}$ have about Y.

However, none of the components of PID can be uniquely defined based on this set of axioms which is why additional assumptions have to be made.

The approach taken in this thesis applies the definition by Bertschinger et al. (2013) and is inspired by game theory. The key assumption is that the unique information terms only depend on the marginal distributions $\mathcal{P}(y, x_1)$ and $\mathcal{P}(y, x_2)$, but not on $\mathcal{P}(y, x_1, x_2)$. In more detail, the unique information should stay the same when $\mathcal{P}$ is replaced with a probability distribution $\mathcal{Q}$ from the space $\Delta_{\mathcal{P}}$ of probability distributions that share the marginals with $\mathcal{P}$. This motivates the following definition of unique information:

$$\tilde{\mathrm{I}}_{\mathrm{unq}}(Y : X_1 \setminus X_2) = \min_{\mathcal{Q} \in \Delta_{\mathcal{P}}} \mathrm{I}_{\mathcal{Q}}(Y : X_1 \,|\, X_2)\,. \tag{A.28}$$

Here, $\mathrm{I}_{\mathcal{Q}}(Y : X_1 \,|\, X_2)$ corresponds to a conditional mutual information based on to the join distribution $\mathcal{Q}(Y, X_1, X_2)$ instead of $\mathcal{P}(Y, X_1, X_2)$. It is noteworthy that only the minimum of $\mathrm{I}_{\mathcal{Q}}(Y : X_1 \,|\, X_2)$ is a measure of the unique information (Figure A.4). With the definition of unique information in Equation (A.28), the shared information $\tilde{\mathrm{I}}_{\mathrm{shd}}$ and synergistic information $\tilde{\mathrm{I}}_{\mathrm{syn}}$ are given by:

$$\tilde{\mathrm{I}}_{\mathrm{shd}}(Y : X_1 ; X_2) = \max_{\mathcal{Q} \in \Delta_{\mathcal{P}}} \left[\mathrm{I}(Y : X_1) - \mathrm{I}_{\mathcal{Q}}(Y : X_1 \,|\, X_2)\right]\,, \tag{A.29}$$

$$\tilde{\mathrm{I}}_{\mathrm{syn}}(Y : X_1 ; X_2) = \mathrm{I}(Y : X_1, X_2) - \min_{\mathcal{Q} \in \Delta_{\mathcal{P}}} \mathrm{I}_{\mathcal{Q}}(Y : X_1, X_2)\,. \tag{A.30}$$

These measures have been shown to be positive and can be obtained by convex optimization (Makkeh et al., 2018). Moreover, the synergistic, shared and unique information of any definition satisfying

the Equations (A.26) and (A.27) are bounded by the previous measures (Bertschinger et al., 2013):

$$\mathrm{I}_{\mathrm{unq}}(Y : X_1 \setminus X_2) \leq \tilde{\mathrm{I}}_{\mathrm{unq}}(y : X_1 \setminus X_2)\,, \tag{A.31}$$

$$\mathrm{I}_{\mathrm{unq}}(Y : X_2 \setminus X_1) \leq \tilde{\mathrm{I}}_{\mathrm{unq}}(y : X_2 \setminus X_1)\,, \tag{A.32}$$

$$\mathrm{I}_{\mathrm{shd}}(Y : X_1 \,; X_2) \geq \tilde{\mathrm{I}}_{\mathrm{shd}}(Y : X_1 \,; X_2)\,, \tag{A.33}$$

$$\mathrm{I}_{\mathrm{syn}}(Y : X_1 \,; X_2) \geq \tilde{\mathrm{I}}_{\mathrm{syn}}(Y : X_1 \,; X_2)\,. \tag{A.34}$$

The above definitions only hold true for considerations involving two input and one output variable. Decompositions of ensembles involving higher counts of input variables are the subject of active research.

A.4 Control of criticality by the input rate

> The results shown in this section are part of the publication Cramer et al. (2020a) which has been presented in the scope of Chapter 6. In the following, we present the results shown in the supplementary material of the publication.

For the results shown in Chapter 6, we adjusted the input strength by changing the degree of the input K_{ext} under the action of a variant of spike-timing dependent plasticity (STDP). In general, the distance to critical-like dynamics can be controlled by a variety of different parameters. Within this section, we demonstrate that the input strength could be equally adjusted by the input rate h for our framework presented in Section 6.2.1.1. Changes in h have qualitatively the same impact on the network dynamics as the degree of the input K_{ext}. Both parameters in combination shape the network dynamics and even allow to pass the critical point for low values of h and K_{ext}. The Figures A.5 to A.8 directly correspond to the ones shown in Section 6.3.1, but depict the results for varying both K_{ext} and h.

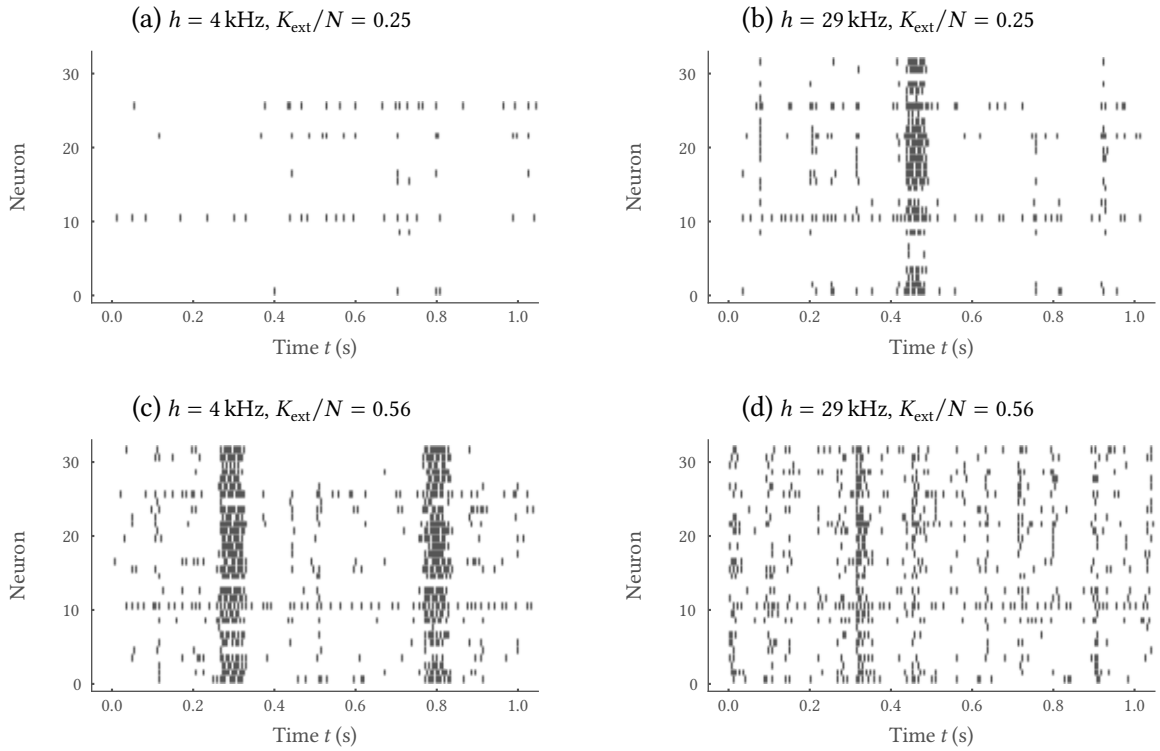


Figure A.5: The input strength shapes the collective dynamics of the network. This figure corresponds to Figure 6.12. For every degree of the input K_{ext}, there is an input rate h for which the activity shows irregular bursts, resembling a critical state. In the sub-critical case, the firing becomes more irregular and asynchronous. The input rate h increases from left to right with $h = 4\,\mathrm{kHz}$ for (a) and (c) and $h = 29\,\mathrm{kHz}$ for (b) and (d). The degree of the input K_{ext} increases from top to bottom with $K_{\mathrm{ext}}/N = 0.25$ for (a) and (b) and $K_{\mathrm{ext}} = 0.56$ for (c) and (d). Figure and caption taken from Cramer et al. (2020a).

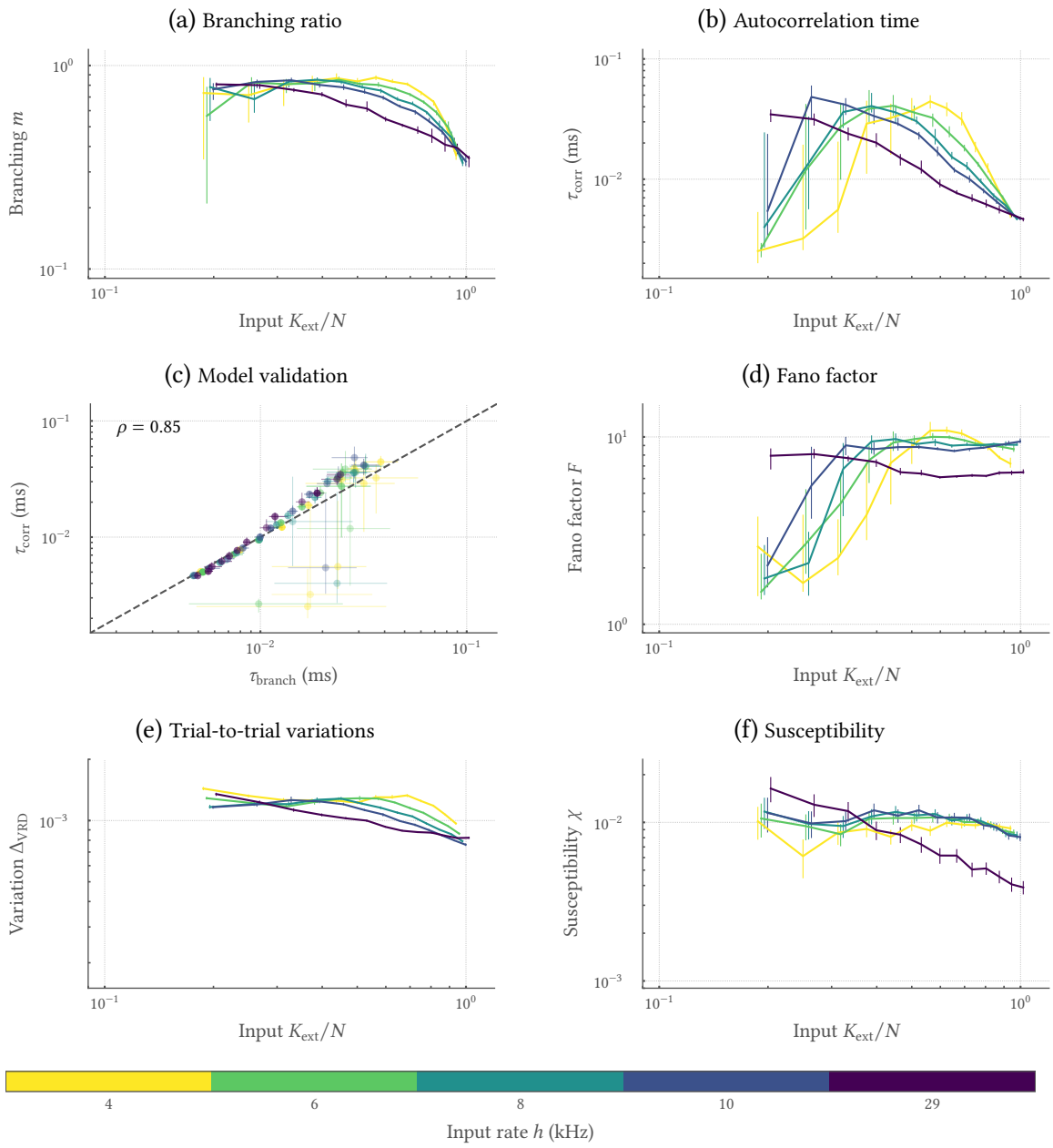


Figure A.6: Depending on the input strength, systems show clear signatures of criticality beyond power-laws. This figure corresponds to Figure 6.15. The input strength is determined by both the degree of external input K_{ext} and the input rate h (colors). Only for specific combinations of these parameters, **(a)** the estimated branching ratio m tends towards unity, and **(b)** the estimated autocorrelation time τ_{corr} peaks. **(c)** The clear match of the τ_{corr}, and the $\tau_{branch} \sim -1/\log(m)$ as inferred from m supports the criticality hypothesis (correlation coefficient of $\rho = 0.850$, $p < 10^{-10}$). Further, intrinsic variations as measured by **(d)** the Fano factor F, and **(e)** the trial-to-trial variation Δ_{VRD}, as well as **(f)** to external perturbations as measured by the susceptibility χ peak approximately for the critical input strengths. Figure and caption taken from Cramer et al. (2020a).

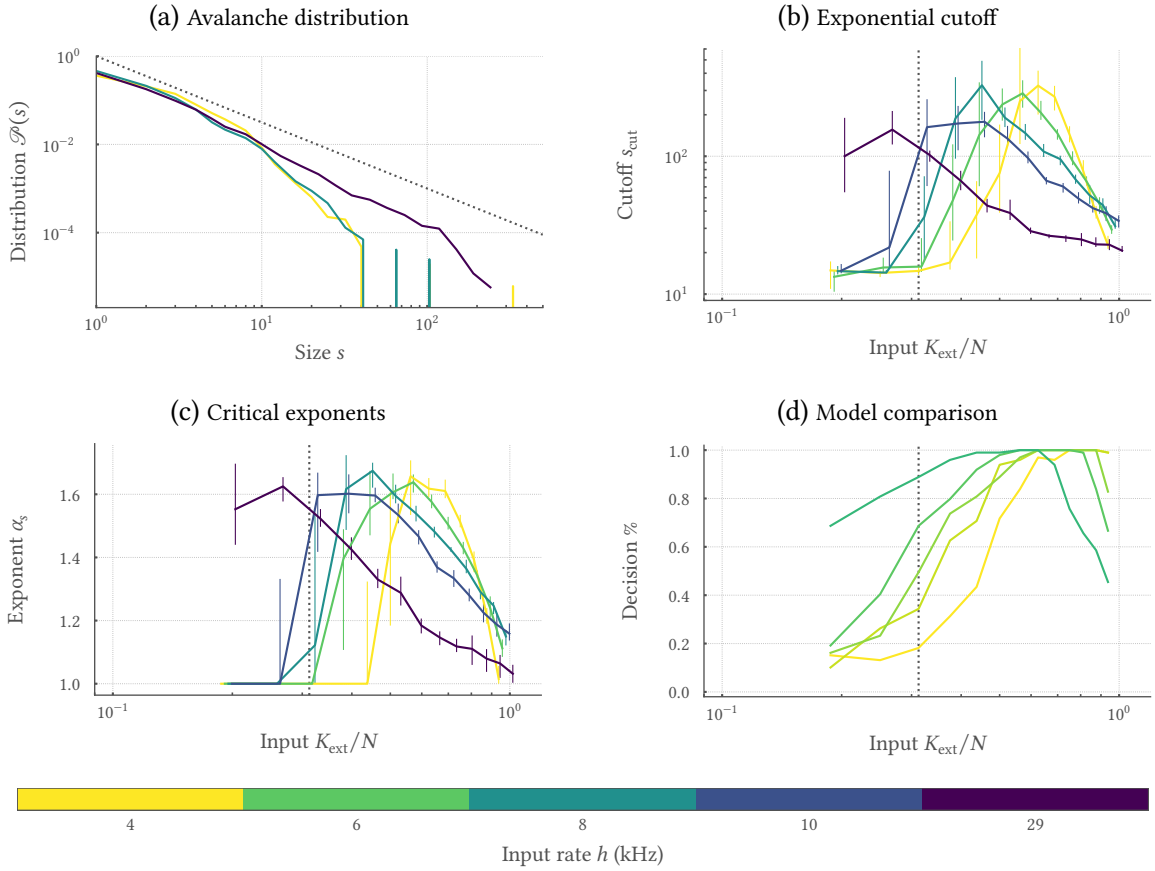


Figure A.7: Under specific input strengths, the network self-organizes towards a critical state and shows long-tailed avalanche distributions. This figure corresponds to Figure 6.13. The input strength is determined by both the degree of external input K_{ext} and the input rate h (colors). Only for specific combinations of these parameters, **(a)** power-law distributed avalanche sizes s over two orders of magnitude are observed (shown for $K_{ext}/N = 0.31$). Fitting a truncated power law, **(b)** the exponential cutoff s_{cut} peaks, and **(c)** critical exponents α_s approximate 1.5 for the critical input strengths, as expected for critical branching processes. **(d)** A maximum-likelihood comparison decides for a power-law compared to an exponential fit in the majority of cases for the aforementioned critical input strengths. The dashed vertical line in (b) to (d) highlights the K_{ext}/N that has been selected in (a). Figure and caption taken from Cramer et al. (2020a).

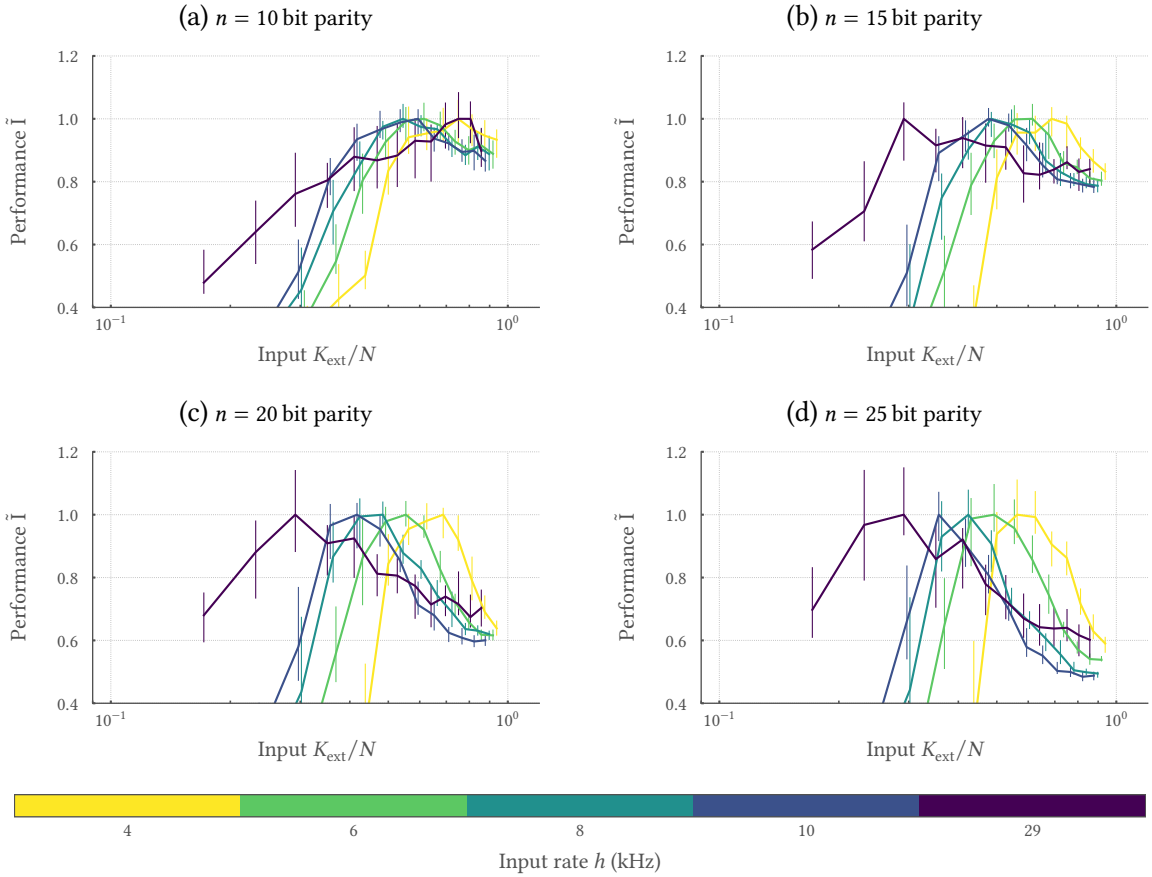


Figure A.8: Computational challenging task profit from critical network dynamics – simple tasks do not. This figure corresponds to Figure 6.16b. The network is used to solve a n-bit parity task by training a linear classifier on the activity of $N_{read} = 32$ neurons. Here, task complexity increases with n, the number of past inputs that need to be memorized. Task-complexity n increases from (a) to (d) with $n \in \{10, 15, 20, 25\}$. For high n, task performance profits from criticality, whereas simple tasks suffer from criticality. The performance is quantified by the normalized mutual information $\tilde{I}$ between the parity of the input and the vote of a linear classifier. The performance $\tilde{I}$ for high n-bit parity tasks is higher for the critical pairs of the external input K_{ext} and the input rate h. Figure and caption taken from Cramer et al. (2020a).

A.5 Software

Table A.2: Software state used for the experiments on surrogate gradient learning. The experiment code used to gather the results presented in Chapter 5 is provided in the repository `model-hx-strobe` and was used in combination with the shown software state and the container image `c10889p24_2020-11-04_1.img` with the `visionary-dls` app.

Repository	Git hash
calix	ede948267e4fb6944cee4925bf2f6d8106c6f6fe
haldls	d9d1d30bc032e214b5a59905fb964a3a1db166b9
pyhid	82c64b5569928d2ec8344b8dffbd930da23d0004
tools-kintex7	34d64ba95ac0db73c2cae2ea12cc2e3108cb0b5a
code-format	5d55a9952d4b6400fa5b2baeff9be546e45bf76d
logger	bc006238ecfdc483d5b96ce5f5bb62e5a93e99dd
halco	27ccf9d1a92dfb938f82e1de62ab0acc45cbcd33
hate	c7483cedc3d76b8e7a4a65e7bc9a423131f40ce1
fisch	087ea1930137b1e940cf841598d4f2595f5709c3
hxcomm	5114f8219973cb02e13b95965f76367ffb66697c
hmf-fpga	46844c8e5018a094b8bedbcf5c305e45f6a49091
rant	4fc2cc3689c9b141708dafbcc5f9d3c7c2b7f18d
ztl	d900ab073f6aa8df4bf7f187bdbb65f1f6cac2f6
pywrap	83ddbad8a114b4730b82d299e8bd9da2a6ca5ebb
lib-boost-patches	2d7e07d4e74827c42d9e1a51f8d180af9907f7cb
sctrltp	72a3735c606795b0b058271d4899b5f490bd88bf
visions-slurm	79de4b1754f26e77fbabb7827dc78556276c85a0
flange	fcde2aafe69805487789ca0b1a8a245caf5fb8ed
lib-rcf	5b16326ae30ee08a322a6569887ca8bd2684c252
libnux	32597fe35fa93b331de41da2adee127eb40e46e1
hwdb	754b4dbb87a76411cf2291dc34153d4199a8066a
bss-hw-params	3d88b7644bd40fc1fa0210b25062ac8866186a3b

Table A.3: Software state used for the experiments on spike time gradient learning. The experiment code used to gather the results presented in Chapter 5 is provided in the repository https://github.com/JulianGoeltz/fastanddeep and was used in combination with the shown software state and the container image `2021-02-19_1.img` with the `visionary-dls` app.

Repository	Git hash
model-hx-strobe	6fdc7b05a3334ad1c2497e563a3c4c1c41feeceb
haldls	9f34f9b1ea0e34dd1141198c61d2a0b25bfc8e92
calix	f7e9b72aae47929a5de27bd1897a042c14d10ad0
libnux	32597fe35fa93b331de41da2adee127eb40e46e1
pyhid	82c64b5569928d2ec8344b8dffbd930da23d0004
tools-kintex7	08395cc8429031281fd024e5cbf071d521f03259
code-format	be6615c28aedac9e423c5bc0cb602379ad775b18
logger	bc006238ecfdc483d5b96ce5f5bb62e5a93e99dd
halco	00ddc0f868b37be07bbf577f97168c4cef1205e6
hate	c7483cedc3d76b8e7a4a65e7bc9a423131f40ce1
fisch	2147c148af5f681453fac8e81649bc087739afcc
ztl	d900ab073f6aa8df4bf7f187bdbb65f1f6cac2f6
hxcomm	a01ba278fb4994463a9e539aaeeadb950f05256e
hmf-fpga	4c64b3f40501cb1eb2b01728a1b42a828b255102
rant	4fc2cc3689c9b141708dafbcc5f9d3c7c2b7f18d
pywrap	83ddbad8a114b4730b82d299e8bd9da2a6ca5ebb
lib-boost-patches	2d7e07d4e74827c42d9e1a51f8d180af9907f7cb
sctrltp	b5f825007b842f44f3e6401f00cf93387e5e3f3c
hwdb	04c88357dab609aef7a8d36af58d10285d5cae51
visions-slurm	5e7ea560235b068fc12f26e3f0d002d415f76cf9
flange	fcde2aafe69805487789ca0b1a8a245caf5fb8ed
lib-rcf	5b16326ae30ee08a322a6569887ca8bd2684c252

Table A.4: Software state used for the experiments on computing. The experiment code used to gather the results presented in the first part of Chapter 6 is provided in the repository `model-hw-correlation` and was used in combination with the shown software state and the container image `2019-01-09_1.img` with the `visionary-dls` app.

Repository	Git hash
logger	8355792fd3e591d08381575fcf5c4b2547b5fe3d
bitter	a1990af1e25edaa6d8dba5da979f294df7b05cfe
rant	d54908fc9e0cf9f06bace1a4910a6feddd0bdc46
pywrap	d17bf50aa098d098a1c34ebaca74493fb9153022
pyublas	85b4b20a10d16e3342b1ccc187893c2bef611899
pyplusplus	5a612a7f5372d832bebacc5377e78ffd2deabcde
uni	57efe4451fb58220ebbc47a5ee1803ffcf3b3fc8
frickel-dls	3a6ada473d7be11072a2a09cd278b35101bd2edb
pygccxml	8ae9e19ae00c4152fa5a381eb9e663561c07345f
hicann-dls-scripts	b4cad5a8366502efa8c69a13ce826a802e71fd5d
libnux	f66675bd5a95338acba45eab72eca664a269aff2
ppu-software	7110aea926bc5bdcf5d6a787611998824eadc547

Table A.5: Software state used for the experiments on networks exhibiting bistable dynamics. The experiment code used to gather the results presented in second part of Chapter 6 is provided in the repository `model-hw-wavy` and was used in combination with the shown software state and the container image `2021-09-03_1.img` with the `visionary-dls` app.

Repository	Git hash
calix	72e15b86b542cdebaae5b4c743bd36be54477d64
haldls	1b8886b3be50b5fedcd4e05bfa1b448b75ae1a2d
pyhid	82c64b5569928d2ec8344b8dffbd930da23d0004
tools-kintex7	08395cc8429031281fd024e5cbf071d521f03259
code-format	be6615c28aedac9e423c5bc0cb602379ad775b18
logger	bc006238ecfdc483d5b96ce5f5bb62e5a93e99dd
halco	307aa73f4c4ffbbee734cb8c3752f9f584ecf260
hate	f8718505f8f39d65537122bb28c59b2581e0106e
fisch	3c32848a3b1e82a60f7da7de7ea2b787b5229fc2
hxcomm	ccb630dcda2f942f28ebde6977e622ae073605d0
hmf-fpga	c816482bbe0f85468ab70a4f302a01bdd4e1ec9b
rant	4fc2cc3689c9b141708dafbcc5f9d3c7c2b7f18d
ztl	d900ab073f6aa8df4bf7f187bdbb65f1f6cac2f6
pywrap	550051ab0faad678e58cb456079b1ba45ad2230a
lib-boost-patches	2d7e07d4e74827c42d9e1a51f8d180af9907f7cb
sctrltp	be58599f60a8652b0404bf3a5f7dd3a3b4d1c303
visions-slurm	3777a9dc36a7067be3657ce06253efec32db260e
flange	2fb312fb4fc31634d3dbf74243c13a566b79810f
lib-rcf	211ea6afc811dcc9964dff3d5526048412b8e2ae
libnux	03f536b0e358ae5e473dba8c384a12f95d86b0cb
hwdb	13fadc068fc7bbaabef7ed678ef5237b423f00be

Table A.6: Software state used for the weight-driven structural plasticity experiments. The experiment code used to gather the results presented in first part of Chapter 7 is provided in the repository `model-dls-structural` and was used in combination with the shown software state and the container image `2019-01-21_1.img` with the `visionary-dls` app.

Repository	Git hash
symwaf2ic	66a4a9126fb8040fdc1849e117c61a7dd4580c82
logger	8355792fd3e591d08381575fcf5c4b2547b5fe3d
bitter	aa18d4a73a994a7e8590addbc40f6dc34a439b24
rant	8d5d65484667852ccfdb52843f0b6d4b6b067324
pywrap	85a6cfc104f52ef51310a4a61e28209daea2c1fc
pyublas	cb8db753399719cd356afa1b2c749eb54c7c420a
pyplusplus	064993baaea33e81c93655d79d9a1a6204b4acd0
uni	9336fbc68346c00fcf3e410d08a1669394fe1148
frickel-dls	2db3700557933680044cd9833d3d2c355ff5aa5d
pygccxml	8ae9e19ae00c4152fa5a381eb9e663561c07345f
flyspi-rw_api	475f99bbf2f5bd4d5b065bd39ed9704e273ec748
dls2calib	032c45cbf4d2c21bdad1aa24f6dddd52233b5c55
hicann-dls-scripts	1635a36edb01fc96f51b0ca7a95e39f6c0099844
libnux	1c1592adf6f00683e5d30e613aff4a79d9a913ac

Table A.7: Software state used for the STDP-driven structural plasticity experiments. The experiment code used to gather the results presented in second part of Chapter 7 is provided in the repository `model-hw-correlation` and was used in combination with the shown software state and the container image `2019-01-09_1.img` with the `visionary-dls` app.

Repository	Git hash
logger	8355792fd3e591d08381575fcf5c4b2547b5fe3d
bitter	aa18d4a73a994a7e8590addbc40f6dc34a439b24
rant	acd5a3cd94fe91fa942e1da727e47025f00208c8
pywrap	158e5bb2b70f8c3f9b7d45e431c6105cb4dc301d
pyublas	feaf60f2f920e30d588837dd6b0715eef23ed550
pyplusplus	064993baaea33e81c93655d79d9a1a6204b4acd0
uni	e9135459841741e446e17f514e048fff74a8441f
frickel-dls	17c2d37c3bdad0a1f3b83260edd05dc5ba501a57
pygccxml	8ae9e19ae00c4152fa5a381eb9e663561c07345f
hicann-dls-scripts	3d84f5685059e2e2d96374ff64473f124884add2
libnux	0c46980e778162a893d1ae641db4c26e38c3256c

List of coauthored publications

This chapter lists the coauthored publications and highlights their impact on this thesis. The star (*) highlights authors with equal contributions.

Peer-reviewed publications

1. **Control of criticality and computation in spiking neuromorphic networks with plasticity**

   Benjamin Cramer, David Stöckel, Markus Kreft, Michael Wibral, Johannes Schemmel, Karlheinz Meier, Viola Priesemann. *Nature Communications 11 (2020): 1-11*

   The author conceptualized, implemented and evaluated the experiment aspects of this publication under the supervision of Dr. Viola Priesemann. The content of this publication encompasses the work on critical-like SNNs presented in the first part of Chapter 6.

2. **The Heidelberg Spiking data sets for the systematic evaluation of spiking neural networks**

   Benjamin Cramer, Yannik Stradmann, Johannes Schemmel, Friedemann Zenke. *IEEE Transactions on Neural Networks and Learning Systems (2020): 1-14*

   The author conceptualized, implemented and evaluated the experiment aspects of this publication under the supervision of Dr. Friedemann Zenke. The content of this publication encompasses the benchmark dataset as well as the considerations thereof discussed in Chapter 4.

3. **Structural plasticity on an accelerated analog neuromorphic hardware system**

   Sebastian Billaudelle*, Benjamin Cramer*, Mihai A. Petrovici, Korbinian Schreiber, David Kappel, Johannes Schemmel, Karlheinz Meier. *Neural Networks 133 (2021): 11-20*

   The author conceptualized, implemented and evaluated the experiment aspects of this publication in close collaboration with Sebastian Billaudelle. The content of this publication constitutes the weight-driven structural plasticity implementation depicted in the first part of Chapter 7.

4. **Versatile emulation of spiking neural networks on an accelerated neuromorphic substrate**

   Sebastian Billaudelle*, Yannik Stradmann*, Korbinian Schreiber*, Benjamin Cramer*, Andreas Baumbach*, Dominik Dold*, Julian Göltz*, Akos F. Kungl*, Timo C. Wunderlich*, Andreas Hartel, Eric Müller, Oliver Breitwieser, Christian Mauch, Mitja Kleider, Andreas Grübl, David Stöckel, Christian Pehle, Arthur Heimbrecht, Philipp Spilger, Gerd Kiene, Vitali Karasenko, Walter Senn, Mihai A. Petrovici, Johannes Schemmel, Karlheinz Meier. *2020 IEEE International Symposium on Circuits and Systems (2020): 1-5*

This review on experiments conducted on different chips implementing the BrainScaleS-2 architecture contains aspects of the weight-driven structural plasticity implementation presented in the first part of Chapter 7.

5. **Verification and design methods for the BrainScaleS neuromorphic hardware system**.

   Andreas Grübl*, Sebastian Billaudelle*, Benjamin Cramer, Vitali Karasenko, Johannes Schemmel. *Journal of Signal Processing Systems 92 (2020): 1277-1292*

   The author conceptualized, implemented and evaluated the reward-modulated STDP experiment presented in this publication in collaboration with Sebastian Billaudelle. No content of this publication is presented within this thesis.

6. **Fast and energy-efficient neuromorphic deep learning with first-spike times**

   Julian Göltz*, Laura Kriener*, Andreas Baumbach, Sebastian Billaudelle, Oliver Breitwieser, Benjamin Cramer, Dominik Dold, Johannes Schemmel, Karlhein Meier, Mihai A. Petrovici. *Nature Machine Intelligence 3 (2021): 823-835*

   The author contributed to this publication in terms of calibration and routing routines as well as low-level software. The content of this publication encompasses the firing time gradient approach presented in Chapter 5.

Preprints

1. **Surrogate gradients for analog neuromorphic computing**

   Benjamin Cramer*, Sebastian Billaudelle*, Simeon Kanya, Aron Leibfried, Andreas Grübl, Vitali Karasenko, Christian Pehle, Korbinian Schreiber, Yannik Stradmann, Johannes Weis, Johannes Schemmel, Friedemann Zenke. *arXiv preprint arXiv:2006.07239 (in revision)*

   The author conceptualized, implemented and evaluated the experiment aspects of this publication in close collaboration with Sebastian Billaudelle under the supervision of Dr. Friedemann Zenke. The content of this upcoming publication comprises the surrogate gradient-based ITL training framework as well as the associated results presented in Chapter 5.

2. **Spiking neuromorphic chip learns entangled quantum states**

   Stefanie Czischek, Andreas Baumbach, Sebastian Billaudelle, Benjamin Cramer, Lukas Kades, Jan M. Pawlowski, Markus K. Oberthaler, Johannes Schemmel, Mihai A. Petrovici, Thomas Gasenzer, Martin Gärttner. *arXiv preprint arXiv:2008.01039*

   The author contributed to this publication in terms of calibration and routing routines as well as low-level software. No content of this publication is presented in this thesis.

Conference contributions

1. **Spike timing based learning in neural networks induces a diverse range of states**

   Benjamin Cramer, David Stöckel, Johannes Schemmel, Karlheinz Meier, Viola Priesemann. *CNS Conference 2018, BMC Neuroscience 19 (2018): P263*

The author conceptualized, implemented and evaluated the experiment aspects of this publication under the supervision of Dr. Viola Priesemann. The content of this publication constitutes the assessment of critical-like phenomena described in Chapter 6.

2. **Control of criticality in self-stabilizing neuromorphic spiking networks**

   Benjamin Cramer, Markus Kreft, Johannes Schemmel, Karlheinz Meier, Viola Priesemann. *Bernstein Conference 2018*

   The author conceptualized, implemented and evaluated the experiment aspects of this publication under the supervision of Dr. Viola Priesemann. The content of this publication encompasses the control of critical-like dynamics discussed in Chapter 6.

3. **Structural plasticity on an accelerated analog neuromorphic hardware system**

   Sebastian Billaudelle[*], Benjamin Cramer[*], Johannes Schemmel, Karlheinz Meier. *Bernstein Conference 2018*

   The author conceptualized, implemented and evaluated the experiment aspects of this publication in close collaboration with Sebastian Billaudelle. The content of this publication constitutes the weight-driven structural plasticity implementation and results thereof presented in the first part of Chapter 7.

4. **Structural plasticity on spiking neuromorphic hardware**

   Sebastian Billaudelle[*], Benjamin Cramer[*], Mihai A. Petrovici, Korbinian Schreiber, Johannes Schemmel. *Neuro Inspired Computational Elements Conference 2021*

   The author conceptualized, implemented and evaluated the experiment aspects of this publication in close collaboration with Sebastian Billaudelle. The content of this publication constitutes the weight-driven structural plasticity implementation and results thereof presented in the first part of Chapter 7.

Acknowledgements

I would like to thank

Professor Dr. Karlheinz Meier for his support and the opportunity to conduct my doctorial studies in his Electronic Vision(s) group. Dear Professor Meier, I thank you for the versatile opportunities you gave to me to present my early results that led to the collaborations and broad range of topics covered within this thesis. I would have been very pleased if you had witnessed the progress of the BrainScaleS platform over the last few years and the outcome of the projects you so enthusiastically initiated and facilitated.

Dr. Johannes Schemmel for being my supervisor. Dear Johannes, I thank you for the trust you have placed in me and the resulting freedom required to conduct the studies presented within this thesis. Your support has helped me a lot even in difficult situations.

Dr. Viola Priesemann for the external supervision and manifold support. Dear Viola, I thank you for your constantly open ear even at times of the Covid-19 pandemic and the physical distance. I always enjoyed the productive and inspiring meetings and discussions.

Dr. Friedemann Zenke for the external supervision and insight into supervised learning methods. Dear Friedemann, without your support, the SHD would probably have remained just a byproduct of this work. The friendly atmosphere in our meetings was always a pleasure.

Dr. Johannes Zierenberg for the great work and interesting discussions. Dear Johannes, Our early morning as well as late-night discussions have always given me great pleasure. I thank you for your valuable advice, scientific guidance and support during the last phase of my doctorial studies.

Professor Dr. Daniel Durstewitz, for the spontaneous willingness to write a report for my thesis and Professor Dr. Selim Jochim for participating in my oral examination.

Sebastian Billaudelle for the great collaboration and friendship. Dear Sebastian, I really enjoyed our first project and I am even more pleased that many more have following until today. I am looking forward to future exciting work as well as more tasty coffee circles and associated discussions.

David Stöckel and Markus Kreft for the joint work on collective dynamics and unsupervised learning. Dear David and Markus I really enjoyed our collaboration.

Yannik Stradmann for the support and nightly recording session. Dear Yannik, our shared audiophilia led to a really nice dataset.

Matthias Loidold for interesting discussions. Dear Matthias, I always enjoyed our scientific exchange either virtually in the early morning hours or during my visits to Göttingen. Thank you for your support during the last phase of writing.

My closer colleagues Sebastian Billaudelle, Yannik Stradmann, Korbinian Schreiber, Tobias Thommes, Hartmut Schmidt and Christian Pehle. Our joint lunch hour in the Cafe Botanik as well as our gatherings on business trips were always a pleasure.

The whole Electronic Vision(s) and the Priesemann Group for the productive environment.

Elias Arnold, Andreas Baumbach, Sebastian Billaudelle, Timo Gierlich, Markus Kreft, Matthias Loidold, Mike Schmidgen, Yannik Stradmann, Felix Weber, Friedemann Zenke and Johannes Zierenberg for proof-reading this manuscript.

My family for the guidance and support they have given to me.

My wife Sylvia Cramer for her balance, love and support. Dear Sylvia, without your love and support, especially during the last phase of my doctorial studies, this thesis would probably not have been emerged in this form. Thank you for being with me through every situation in life.

Funding statement: This work has received funding from the European Union Sixth Framework Programme under grant agreement no 15879 (FACETS), the European Union Seventh Framework Programme under grant agreement no. 604102 (HBP), 269921 (BrainScaleS), and 243914 (Brain-i-Nets), and the European Union's Horizon 2020 research and innovation programme under grant agreement no 720270, 785907 and 945539 (HBP), as well as the Manfred Stärk Foundation. This work was supported by the Novartis Research Foundation and the Max-Planck-Society. The authors acknowledge support by the state of Baden-Württemberg through bwHPC and the German Research Foundation (DFG) through grant no INST 39/963-1 FUGG (bwForCluster NEMO).

Bibliography

Aamir, S. A., Müller, P., Kriener, L., Kiene, G., Schemmel, J., & Meier, K. (2017). From lif to adex neuron models: Accelerated analog 65 nm cmos implementation. In *2017 IEEE Biomedical Circuits and Systems Conference (BioCAS)* (pp. 1–4).: IEEE.

Aamir, S. A., Müller, P., Kiene, G., Kriener, L., Stradmann, Y., Grübl, A., Schemmel, J., & Meier, K. (2018). A mixed-signal structured adex neuron for accelerated neuromorphic cores. *IEEE Transactions on Biomedical Circuits and Systems*, 12(5), 1027–1037.

Aamir, S. A., Stradmann, Y., Müller, P., Pehle, C., Hartel, A., Grübl, A., Schemmel, J., & Meier, K. (2018). An accelerated lif neuronal network array for a large-scale mixed-signal neuromorphic architecture. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 65(12), 4299–4312.

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., & Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Abbott, L. F., DePasquale, B., & Memmesheimer, R.-M. (2016). Building functional networks of spiking model neurons. *Nature Neuroscience*, 19(3), 350–355.

Abbott, L. F., Varela, J., Sen, K., & Nelson, S. (1997). Synaptic depression and cortical gain control. *Science*, 275(5297), 221–224.

Abeles, M. (1991). *Corticonics: Neural circuits of the cerebral cortex*. Cambridge University Press.

Abiodun, O. I., Jantan, A., Omolara, A. E., Dada, K. V., Umar, A. M., Linus, O. U., Arshad, H., Kazaure, A. A., Gana, U., & Kiru, M. U. (2019). Comprehensive review of artificial neural network applications to pattern recognition. *IEEE Access*, 7, 158820–158846.

Abraham, W. C. & Bear, M. F. (1996). Metaplasticity: the plasticity of synaptic plasticity. *Trends in neurosciences*, 19(4), 126–130.

Ackley, D. H., Hinton, G. E., & Sejnowski, T. J. (1985). A learning algorithm for boltzmann machines. *Cognitive science*, 9(1), 147–169.

Akopyan, F., Sawada, J., Cassidy, A., Alvarez-Icaza, R., Arthur, J., Merolla, P., Imam, N., Nakamura, Y., Datta, P., Nam, G., Taba, B., Beakes, M., Brezzo, B., Kuang, J. B., Manohar, R., Risk, W. P., Jackson, B., & Modha, D. S. (2015). Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 34(10), 1537–1557.

Allen, C. & Stevens, C. F. (1994). An evaluation of causes for unreliability of synaptic transmission. *Proceedings of the National Academy of Sciences*, 91(22), 10380–10383.

Alstott, J., Bullmore, E., & Plenz, D. (2014). powerlaw: a python package for analysis of heavy-tailed distributions. *PloS one*, 9(1), e85777.

Ambrogio, S., Narayanan, P., Tsai, H., Shelby, R. M., Boybat, I., Di Nolfo, C., Sidler, S., Giordano, M., Bodini, M., Farinha, N. C., et al. (2018). Equivalent-accuracy accelerated neural-network training using analogue memory. *Nature*, 558(7708), 60–67.

Amir, A., Taba, B., Berg, D., Melano, T., McKinstry, J., Di Nolfo, C., Nayak, T., Andreopoulos, A., Garreau, G., Mendoza, M., & others (2017). A Low Power, Fully Event-Based Gesture Recognition System. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 7243–7252).

Annoni, A., Benczur, P., Bertoldi, P., Delipetrev, B., De Prato, G., Feijoo, C., Macias, E. F., Gutierrez, E. G., Portela, M. I., Junklewitz, H., et al. (2018). *Artificial Intelligence: A European Perspective*. Technical report, Joint Research Centre (Seville site).

Anumula, J., Neil, D., Delbruck, T., & Liu, S.-C. (2018). Feature Representations for Neuromorphic Audio Spike Streams. *Frontiers in neuroscience*, 12, 23.

Appeltant, L., Soriano, M. C., Van der Sande, G., Danckaert, J., Massar, S., Dambre, J., Schrauwen, B., Mirasso, C. R., & Fischer, I. (2011). Information processing using a single dynamical node as complex system. *Nature communications*, 2(1), 1–6.

Attwell, D. & Laughlin, S. B. (2001). An energy budget for signaling in the grey matter of the brain. *Journal of Cerebral Blood Flow & Metabolism*, 21(10), 1133–1145.

Azevedo, F. A., Carvalho, L. R., Grinberg, L. T., Farfel, J. M., Ferretti, R. E., Leite, R. E., Filho, W. J., Lent, R., & Herculano-Houzel, S. (2009). Equal numbers of neuronal and nonneuronal cells make the human brain an isometrically scaled-up primate brain. *Journal of Comparative Neurology*, 513(5), 532–541.

Barnett, L., Lizier, J. T., Harré, M., Seth, A. K., & Bossomaier, T. (2013). Information flow in a kinetic ising model peaks in the disordered phase. *Physical review letters*, 111(17), 177203.

Baum, L. E. et al. (1972). An inequality and associated maximization technique in statistical estimation for probabilistic functions of markov processes. *Inequalities*, 3(1), 1–8.

Bear, M. F., Cooper, L. N., & Ebner, F. F. (1987). A physiological basis for a theory of synapse modification. *Science*, 237(4810), 42–48.

Beasley, J. S. (2010). Modern electronic communication.

Beggs, J. M. & Plenz, D. (2003). Neuronal avalanches in neocortical circuits. *Journal of neuroscience*, 23(35), 11167–11177.

Bell, C. C., Han, V. Z., Sugawara, Y., & Grant, K. (1997). Synaptic plasticity in a cerebellum-like structure depends on temporal order. *Nature*, 387(6630), 278–281.

Bellec, G., Kappel, D., Maass, W., & Legenstein, R. (2017). Deep rewiring: Training very sparse deep networks. *arXiv preprint arXiv:1711.05136*.

Bellec, G., Salaj, D., Subramoney, A., Legenstein, R., & Maass, W. (2018). Long short-term memory and learning-to-learn in networks of spiking neurons. In *Advances in Neural Information Processing Systems* (pp. 787–797).

Bellec, G., Scherr, F., Subramoney, A., Hajek, E., Salaj, D., Legenstein, R., & Maass, W. (2020). A solution to the learning dilemma for recurrent networks of spiking neurons. *Nature Communications*, 11(1), 3625. Number: 1 Publisher: Nature Publishing Group.

Bengio, Y., Léonard, N., & Courville, A. (2013a). Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*.

Bengio, Y., Léonard, N., & Courville, A. (2013b). Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation. *arXiv:1308.3432 [cs]*. arXiv: 1308.3432.

Benita, J. M., Guillamon, A., Deco, G., & Sanchez-Vives, M. V. M. (2012). Synaptic depression and slow oscillatory activity in a biophysical network model of the cerebral cortex. *Frontiers in computational neuroscience*, 6, 64.

Benjamin, B. V., Gao, P., McQuinn, E., Choudhary, S., Chandrasekaran, A. R., Bussat, J.-M., Alvarez-Icaza, R., Arthur, J. V., Merolla, P. A., & Boahen, K. (2014). Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations. *Proceedings of the IEEE*, 102(5), 699–716.

Bernardi, D. & Lindner, B. (2017). Optimal detection of a localized perturbation in random networks of integrate-and-fire neurons. *Physical review letters*, 118(26), 268301.

Bernstein, J. (1868). Ueber den zeitlichen verlauf der negativen schwankung des nervenstroms. *Archiv für die gesamte Physiologie des Menschen und der Tiere*, 1(1), 173–207.

Bertschinger, N. & Natschläger, T. (2004). Real-time computation at the edge of chaos in recurrent neural networks. *Neural computation*, 16(7), 1413–1436.

Bertschinger, N., Rauh, J., Olbrich, E., & Jost, J. (2013). Shared information—new insights and problems in decomposing information in complex systems. In *Proceedings of the European conference on complex systems 2012* (pp. 251–269).: Springer.

Bhaduri, A., Banerjee, A., Roy, S., Kar, S., & Basu, A. (2018). Spiking neural classifier with lumped dendritic nonlinearity and binary synapses: a current mode vlsi implementation and analysis. *Neural computation*, 30(3), 723–760.

Bhatt, D. H., Zhang, S., & Gan, W.-B. (2009). Dendritic spine dynamics. *Annual review of physiology*, 71, 261–282.

Bi, G. & Poo, M. (1998). Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type. *Journal of neuroscience*, 18(24), 10464–10472.

Bill, J., Schuch, K., Brüderle, D., Schemmel, J., Maass, W., & Meier, K. (2010). Compensating inhomogeneities of neuromorphic VLSI devices via short-term synaptic plasticity. *Frontiers in computational neuroscience*, 4, 129.

Billaudelle, S. (2017). Design and implementation of a short term plasticity circuit for a 65 nm neuromorphic hardware system. Masterarbeit, Universität Heidelberg.

Billaudelle, S., Cramer, B., Petrovici, M. A., Schreiber, K., Kappel, D., Schemmel, J., & Meier, K. (2021). Structural plasticity on an accelerated analog neuromorphic hardware system. *Neural Networks*, 133, 11–20.

Billaudelle, S., Stradmann, Y., Schreiber, K., Cramer, B., Baumbach, A., Dold, D., Göltz, J., Kungl, A. F., Wunderlich, T. C., Hartel, A., et al. (2020). Versatile emulation of spiking neural networks on an accelerated neuromorphic substrate. In *2020 IEEE International Symposium on Circuits and Systems (ISCAS)* (pp. 1–5).: IEEE.

Binney, J. J., Dowrick, N. J., Fisher, A. J., & Newman, M. E. (1992). *The theory of critical phenomena: an introduction to the renormalization group*. Oxford University Press.

Bliss, T. V. & Collingridge, G. L. (1993). A synaptic model of memory: long-term potentiation in the hippocampus. *Nature*, 361(6407), 31–39.

Boahen, K. (2017). A neuromorph's prospectus. *Computing in Science Engineering*, 19(2), 14–28.

Boedecker, J., Obst, O., Lizier, J. T., Mayer, N. M., & Asada, M. (2012). Information processing in echo state networks at the edge of chaos. *Theory in Biosciences*, 131(3), 205–213.

Bogdan, P. A., Rowley, A. G., Rhodes, O., & Furber, S. B. (2018). Structural plasticity on the spinnaker many-core neuromorphic system. *Frontiers in Neuroscience*, 12, 434.

Bohnstingl, T., Scherr, F., Pehle, C., Meier, K., & Maass, W. (2019). Neuromorphic hardware learns to learn. *Frontiers in neuroscience*, 13.

Bohte, S. M. (2011). Error-Backpropagation in Networks of Fractionally Predictive Spiking Neurons. In *Artificial Neural Networks and Machine Learning – ICANN 2011*, Lecture Notes in Computer Science (pp. 60–68).: Springer, Berlin, Heidelberg.

Bohte, S. M., Kok, J. N., & La Poutre, H. (2002). Error-backpropagation in temporally encoded networks of spiking neurons. *Neurocomputing*, 48(1), 17–37.

Borges, F., Protachevicz, P., Pena, R., Lameu, E., Higa, G., Kihara, A., Matias, F., Antonopoulos, C., de Pasquale, R., Roque, A., et al. (2020). Self-sustained activity of low firing rate in balanced networks. *Physica A: Statistical Mechanics and its Applications*, 537, 122671.

Braitenberg, V. & Schüz, A. (2013). *Cortex: statistics and geometry of neuronal connectivity*. Springer Science & Business Media.

Brea, J., Senn, W., & Pfister, J.-P. (2013). Matching recall and storage in sequence learning with spiking neural networks. *Journal of neuroscience*, 33(23), 9565–9575.

Brette, R. (2019). Is coding a relevant metaphor for the brain? *Behavioral and Brain Sciences*, 42.

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

Brodtkorb, A. R., Hagen, T. R., & Sætra, M. L. (2013). Graphics processing unit (gpu) programming strategies and trends in gpu computing. *Journal of Parallel and Distributed Computing*, 73(1), 4–13.

Brown, C. E., Li, P., Boyd, J. D., Delaney, K. R., & Murphy, T. H. (2007). Extensive turnover of dendritic spines and vascular remodeling in cortical tissues recovering from stroke. *Journal of Neuroscience*, 27(15), 4101–4109.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., & Amodei, D. (2020). Language models are few-shot learners.

Brüderle, D., Petrovici, M. A., Vogginger, B., Ehrlich, M., Pfeil, T., Millner, S., Grübl, A., Wendt, K., Müller, E., Schwartz, M.-O., & et al. (2011). A comprehensive workflow for general-purpose neural modeling with highly configurable neuromorphic hardware systems. *Biological Cybernetics*, 104, 263–296.

Brunel, N. (2000). Dynamics of sparsely connected networks of excitatory and inhibitory spiking neurons. *Journal of computational neuroscience*, 8(3), 183–208.

Brunner, D., Reitzenstein, S., & Fischer, I. (2016). All-optical neuromorphic computing in optical networks of semiconductor lasers. In *2016 IEEE International Conference on Rebooting Computing (ICRC)* (pp. 1–2).: IEEE.

Büchel, J., Zendrikov, D., Solinas, S., Indiveri, G., & Muir, D. R. (2021). Supervised training of spiking neural networks for robust deployment on mixed-signal neuromorphic processors. *arXiv preprint arXiv:2102.06408*.

Butz, M., Woergoetter, F., & van Ooyen, A. (2009). Activity-dependent structural plasticity. *Brain research reviews*, 60(2), 287–305.

Buzsáki, G. & Wang, X.-J. (2012). Mechanisms of gamma oscillations. *Annual review of neuroscience*, 35, 203–225.

Cajal, R. Y. et al. (1893). Nuevo concepto de la histología de los centros nerviosos.

Cajal, S. R. y. (1890). *Textura de las circunvolutiones cerebrales de los mamíferos inferiores: nota preventiva*.

Cajal, S. R. y. (1899). Estudios sobre la corteza cerebral humana. ii. estructura de la corteza motriz del hombre y mamíferos superiores. *Revista Trimestral Micrográfica*, 4, 117–200.

Cajal, S. R. y. (1933). ¿ neuronismo o reticularismo?: las pruebas objetivas de la unidad anatómica de las células nerviosas. *Archives of Neurobiology*, 13, 217–291.

Chen, G. K., Kumar, R., Sumbul, H. E., Knag, P. C., & Krishnamurthy, R. K. (2019). A 4096-neuron 1m-synapse 3.8-pj/sop spiking neural network with on-chip stdp learning and sparse weights in 10-nm finfet cmos. *IEEE Journal of Solid-State Circuits*, 54(4), 992–1002.

Chicca, E., Stefanini, F., Bartolozzi, C., & Indiveri, G. (2014). Neuromorphic electronic circuits for building autonomous cognitive systems. *Proceedings of the IEEE*, 102(9), 1367–1388.

Chklovskii, D. B., Mel, B., & Svoboda, K. (2004). Cortical rewiring and information storage. *Nature*, 431(7010), 782–788.

Chollet, F. et al. (2015). Keras. https://keras.io.

Christophel, T. B., Klink, P. C., Spitzer, B., Roelfsema, P. R., & Haynes, J.-D. (2017). The distributed nature of working memory. *Trends in cognitive sciences*, 21(2), 111–124.

Citri, A. & Malenka, R. C. (2008). Synaptic plasticity: multiple forms, functions, and mechanisms. *Neuropsychopharmacology*, 33(1), 18–41.

Clauset, A., Shalizi, C. R., & Newman, M. E. (2009). Power-law distributions in empirical data. *SIAM review*, 51(4), 661–703.

Comsa, I. M., Fischbacher, T., Potempa, K., Gesmundo, A., Versari, L., & Alakuijala, J. (2020). Temporal Coding in Spiking Neural Networks with Alpha Synaptic Function. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 8529–8533). ISSN: 2379-190X.

Corradi, F. & Indiveri, G. (2015). A neuromorphic event-based neural recording system for smart brain-machine-interfaces. *IEEE transactions on biomedical circuits and systems*, 9(5), 699–709.

Corradi, F., Zambrano, D., Raglianti, M., Passetti, G., Laschi, C., & Indiveri, G. (2014). Towards a neuromorphic vestibular system. *IEEE transactions on biomedical circuits and systems*, 8(5), 669–680.

Cossart, R., Aronov, D., & Yuste, R. (2003). Attractor dynamics of network up states in the neocortex. *Nature*, 423(6937), 283–288.

Courbariaux, M., Hubara, I., Soudry, D., El-Yaniv, R., & Bengio, Y. (2016). Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*.

Cover, T. M. & Thomas, J. A. (2012). *Elements of information theory*. John Wiley & Sons.

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

Cramer, B., Stöckel, D., Kreft, M., Wibral, M., Schemmel, J., Meier, K., & Priesemann, V. (2020a). Control of criticality and computation in spiking neuromorphic networks with plasticity. *Nature communications*, 11(1), 1–11.

Cramer, B., Stradmann, Y., Schemmel, J., & Zenke, F. (2019). Heidelberg spiking datasets.

Cramer, B., Stradmann, Y., Schemmel, J., & Zenke, F. (2020b). The heidelberg spiking data sets for the systematic evaluation of spiking neural networks. *IEEE Transactions on Neural Networks and Learning Systems*.

Czanner, G., Sarma, S. V., Ba, D., Eden, U. T., Wu, W., Eskandar, E., Lim, H. H., Temereanca, S., Suzuki, W. A., & Brown, E. N. (2015). Measuring the signal-to-noise ratio of a neuron. *Proceedings of the National Academy of Sciences*, 112(23), 7141–7146.

Czischek, S., Baumbach, A., Billaudelle, S., Cramer, B., Kades, L., Pawlowski, J. M., Oberthaler, M. K., Schemmel, J., Petrovici, M. A., Gasenzer, T., et al. (2020). Spiking neuromophic chip learns entangled quantum states. *arXiv preprint arXiv:2008.01039*.

Dale, H. (1934). Pharmacology and nerve endings. *British medical journal*, 2, 1161–1163.

Dale, M., O'Keefe, S., Sebald, A., Stepney, S., & Trefzer, M. A. (2021). Reservoir computing quality: connectivity and topology. *Natural Computing*, 20(2), 205–216.

Dalgaty, T., Castellani, N., Turck, C., Harabi, K.-E., Querlioz, D., & Vianello, E. (2021). In situ learning using intrinsic memristor variability via markov chain monte carlo sampling. *Nature Electronics*, 4(2), 151–161.

Davidson, S. & Furber, S. B. (2021). Comparison of Artificial and Spiking Neural Networks on Digital Hardware. *Frontiers in Neuroscience*, 15.

Davies, M. (2019). Benchmarks for progress in neuromorphic computing. *Nature Machine Intelligence*, 1(9), 386–388.

Davies, M., Srinivasa, N., Lin, T., Chinya, G., Cao, Y., Choday, S. H., Dimou, G., Joshi, P., Imam, N., Jain, S., Liao, Y., Lin, C., Lines, A., Liu, R., Mathaikutty, D., McCoy, S., Paul, A., Tse, J., Venkataramanan, G., Weng, Y., Wild, A., Yang, Y., & Wang, H. (2018). Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro*, 38(1), 82–99.

Dayan, P. & Abbott, L. F. (2001). *Theoretical neuroscience: computational and mathematical modeling of neural systems*. Computational Neuroscience Series.

de Andrade Costa, A., Copelli, M., & Kinouchi, O. (2015). Can dynamical synapses produce true self-organized criticality? *Journal of Statistical Mechanics: Theory and Experiment*, 2015(6), P06004.

de Boer, E. (1980). Auditory physics. Physical principles in hearing theory. I. *Physics reports*, 62(2), 87–174.

de Boer, E. (1984). Auditory physics. Physical principles in hearing theory. II. *Physics Reports*, 105(3), 141–226.

DeFelipe, J. (2015). The dendritic spine story: an intriguing process of discovery. *Frontiers in neuroanatomy*, 9, 14.

Del Papa, B., Priesemann, V., & Triesch, J. (2017). Criticality meets learning: Criticality signatures in a self-organizing recurrent neural network. *PloS one*, 12(5), e0178683.

Denil, M., Shakibi, B., Dinh, L., Ranzato, M., & De Freitas, N. (2013). Predicting parameters in deep learning. In *Advances in neural information processing systems* (pp. 2148–2156).

Denève, S. & Machens, C. K. (2016). Efficient codes and balanced networks. *Nat Neurosci*, 19(3), 375–382.

di Santo, S., Burioni, R., Vezzani, A., & Munoz, M. A. (2016). Self-organized bistability associated with first-order phase transitions. *Physical review letters*, 116(24), 240601.

Douglas, R., Mahowald, M., & Mead, C. (1995). Neuromorphic analogue VLSI. *Annual review of neuroscience*, 18(1), 255–281.

Drago, G. P. & Ridella, S. (1992). Statistically controlled activation weight initialization (scawi). *IEEE Transactions on Neural Networks*, 3(4), 627–631.

Eissa, S., Stuijk, S., & Corporaal, H. (2021). Hardware approximation of exponential decay for spiking neural networks. In *2021 IEEE 3rd International Conference on Artificial Intelligence Circuits and Systems (AICAS)* (pp. 1–4).: IEEE.

Eliasmith, C. & Anderson, C. H. (2004). *Neural Engineering: Computation, Representation, and Dynamics in Neurobiological Systems*. Cambridge, Mass.: A Bradford Book, new ed edition edition.

Esser, S. K., Merolla, P. A., Arthur, J. V., Cassidy, A. S., Appuswamy, R., Andreopoulos, A., Berg, D. J., McKinstry, J. L., Melano, T., Barch, D. R., & et al. (2016). Convolutional networks for fast, energy-efficient neuromorphic computing. *Proceedings of the National Academy of Sciences*, 113(41), 11441–11446.

Evans, W. H. & Martin, P. E. (2002). Gap junctions: structure and function. *Molecular membrane biology*, 19(2), 121–136.

Fang, H., Taylor, B., Li, Z., Mei, Z., Qiu, Q., et al. (2021). Neuromorphic algorithm-hardware codesign for temporal pattern learning. *arXiv preprint arXiv:2104.10712*.

Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2), 179–188.

Florian, R. V. (2012). The Chronotron: A Neuron That Learns to Fire Temporally Precise Spike Patterns. *PLOS ONE*, 7(8), e40233.

Frémaux, N. & Gerstner, W. (2016). Neuromodulated spike-timing-dependent plasticity, and theory of three-factor learning rules. *Frontiers in neural circuits*, 9, 85.

Frenkel, C., Lefebvre, M., Legat, J.-D., & Bol, D. (2018). A 0.086-mm^2 12.7-pj/sop 64k-synapse 256-neuron online-learning digital spiking neuromorphic processor in 28-nm cmos. *IEEE transactions on biomedical circuits and systems*, 13(1), 145–158.

Frenkel, C., Legat, J.-D., & Bol, D. (2019). Morphic: A 65-nm 738k-synapse/mm^2 quad-core binary-weight digital neuromorphic processor with stochastic spike-driven online learning. *IEEE transactions on biomedical circuits and systems*, 13(5), 999–1010.

Frenkel, C., Legat, J.-D., & Bol, D. (2020). A 28-nm convolutional neuromorphic processor enabling online learning with spike-based retinas. In *2020 IEEE International Symposium on Circuits and Systems (ISCAS)* (pp. 1–5).: IEEE.

Friedmann, S., Schemmel, J., Grübl, A., Hartel, A., Hock, M., & Meier, K. (2016). Demonstrating hybrid learning in a flexible neuromorphic hardware system. *IEEE transactions on biomedical circuits and systems*, 11(1), 128–142.

Fuhrmann, G., Segev, I., Markram, H., & Tsodyks, M. (2002). Coding of temporal information by activity-dependent synapses. *Journal of neurophysiology*.

Furber, S. (2016). Large-scale neuromorphic computing systems. *Journal of neural engineering*, 13(5), 051001.

Furber, S. B., Galluppi, F., Temple, S., & Plana, L. A. (2014). The spinnaker project. *Proceedings of the IEEE*, 102(5), 652–665.

Furber, S. B., Lester, D. R., Plana, L. A., Garside, J. D., Painkras, E., Temple, S., & Brown, A. D. (2013). Overview of the spinnaker system architecture. *IEEE Transactions on Computers*, 62(12), 2454–2467.

Gardner, B. & Grüning, A. (2016). Supervised Learning in Spiking Neural Networks for Precise Temporal Encoding. *PLOS ONE*, 11(8), e0161335.

George, R., Indiveri, G., & Vassanelli, S. (2017). Activity dependent structural plasticity in neuromorphic systems. In *Biomedical Circuits and Systems Conference (BioCAS), 2017 IEEE* (pp. 1–4).: IEEE.

Gerstner, W. (2001). What is different with spiking neurons? In *Plausible neural networks for biological modelling* (pp. 23–48). Springer.

Gerstner, W. & Kistler, W. M. (2002). *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge university press.

Gilra, A. & Gerstner, W. (2017). Predicting non-linear dynamics by stable local learning in a recurrent spiking neural network. *eLife Sciences*, 6, e28295.

Göltz, J. (2019). Training deep networks with time-to-first-spike coding on the brainscales wafer-scale system. Masterarbeit, Universität Heidelberg.

Göltz, J., Kriener, L., Baumbach, A., Billaudelle, S., Breitwieser, O., Cramer, B., Dold, D., Kungl, A. F., Senn, W., Schemmel, J., Meier, K., & Petrovici, M. A. (2021). Fast and energy-efficient neuromorphic deep learning with first-spike times. *Nature Machine Intelligence*, 3(9), 823–835.

Goodfellow, I., Bengio, Y., & Courville, A. (2016a). *Deep Learning*. MIT Press.

Goodfellow, I., Bengio, Y., Courville, A., & Bengio, Y. (2016b). *Deep learning*, volume 1. MIT press Cambridge.

Goodman, D. F. & Brette, R. (2009). The brian simulator. *Frontiers in neuroscience*, 3, 26.

Gray, P. R., Hurst, P., Meyer, R. G., & Lewis, S. (2001). *Analysis and design of analog integrated circuits*. Wiley.

Grollier, J., Querlioz, D., & Stiles, M. D. (2016). Spintronic nanodevices for bioinspired computing. *Proceedings of the IEEE*, 104(10), 2024–2039.

Grübl, A., Billaudelle, S., Cramer, B., Karasenko, V., & Schemmel, J. (2020). Verification and design methods for the brainscales neuromorphic hardware system. *Journal of Signal Processing Systems*, 92(11), 1277–1292.

Grutzendler, J., Kasthuri, N., & Gan, W.-B. (2002). Long-term dendritic spine stability in the adult cortex. *Nature*, 420(6917), 812.

Guerguiev, J., Lillicrap, T. P., & Richards, B. A. (2017). Towards deep learning with segregated dendrites. *Elife*, 6, e22901.

Gütig, R. (2014). To spike, or when to spike? *Current Opinion in Neurobiology*, 25, 134–139.

Gütig, R. & Sompolinsky, H. (2006). The tempotron: a neuron that learns spike timing-based decisions. *Nat Neurosci*, 9(3), 420–428.

Güttler, G. M. (2017). *Achieving a Higher Integration Level of Neuromorphic Hardware using Wafer Embedding*. PhD thesis, Universität Heidelberg.

Hansen, N., Arnold, D. V., & Auger, A. (2015). Evolution strategies. In *Springer handbook of computational intelligence* (pp. 871–898). Springer.

Harris, D. M. & Harris, S. L. (2013). *Digital Design and Computer Architecture*. Morgan Kaufmann, second edition edition.

Harris, T. E. (2002). *The theory of branching processes*. Courier Corporation.

Hartley, R. V. L. (1928). Transmission of information. *Bell System Technical Journal*, 7(3), 535–563.

Hasson, U., Chen, J., & Honey, C. J. (2015). Hierarchical process memory: memory as an integral component of information processing. *Trends in cognitive sciences*, 19(6), 304–313.

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision* (pp. 1026–1034).

Hebb, D. O. (1949). *The organization of behavior: A neuropsychological theory*. John Wiley & Sons.

Hellwig, B. (2000). A quantitative analysis of the local connectivity between pyramidal neurons in layers 2/3 of the rat visual cortex. *Biological cybernetics*, 82(2), 111–121.

Hesse, J. & Gross, T. (2014). Self-organized criticality as a fundamental property of neural systems. *Frontiers in systems neuroscience*, 8, 166.

Hildebrand, C., Remahl, S., Persson, H., & Bjartmar, C. (1993). Myelinated nerve fibres in the cns. *Progress in neurobiology*, 40(3), 319–384.

Hochreiter, S. (1998). The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02), 107–116.

Hochreiter, S. & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8), 1735–1780.

Hock, M., Hartel, A., Schemmel, J., & Meier, K. (2013). An analog dynamic memory array for neuromorphic hardware. In *Circuit Theory and Design (ECCTD), 2013 European Conference on* (pp. 1–4).

Hodgkin, A. L. & Huxley, A. F. (1952). A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of physiology*, 117(4), 500.

Holcman, D. & Tsodyks, M. (2006). The emergence of up and down states in cortical networks. *PLoS computational biology*, 2(3), e23.

Holtmaat, A. & Svoboda, K. (2009). Experience-dependent structural synaptic plasticity in the mammalian brain. *Nature Reviews Neuroscience*, 10(9), 647–658.

Holtmaat, A., Wilbrecht, L., Knott, G. W., Welker, E., & Svoboda, K. (2006). Experience-dependent and cell-type-specific spine growth in the neocortex. *Nature*, 441(7096), 979.

Holtmaat, A. J., Trachtenberg, J. T., Wilbrecht, L., Shepherd, G. M., Zhang, X., Knott, G. W., & Svoboda, K. (2005). Transient and persistent dendritic spines in the neocortex in vivo. *Neuron*, 45(2), 279–291.

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8), 2554–2558.

Hoy, M. B. (2018). Alexa, siri, cortana, and more: an introduction to voice assistants. *Medical reference services quarterly*, 37(1), 81–88.

Huang, X., Acero, A., Hon, H.-W., & Reddy, R. (2001). *Spoken Language Processing: A Guide to Theory, Algorithm and System Development*. Prentice Hall.

Huang, Y.-Y., Colino, A., Selig, D. K., & Malenka, R. C. (1992). The influence of prior synaptic activity on the induction of long-term potentiation. *Science*, 255(5045), 730–733.

Huh, D. & Sejnowski, T. J. (2018). Gradient descent for spiking neural networks. In *Advances in Neural Information Processing Systems* (pp. 1433–1443).

Hunsberger, E. & Eliasmith, C. (2015). Spiking Deep Networks with LIF Neurons. *arXiv:1510.08829 [cs]*. arXiv: 1510.08829.

Hussain, S. & Basu, A. (2016). Multiclass classification by adaptive network of dendritic neurons with binary synapses using structural plasticity. *Frontiers in neuroscience*, 10, 113.

Huttenlocher, P. R. (2009). *Neural plasticity*. Harvard University Press.

Indiveri, G., Linares-Barranco, B., Hamilton, T. J., Van Schaik, A., Etienne-Cummings, R., Delbruck, T., Liu, S.-C., Dudek, P., Häfliger, P., Renaud, S., et al. (2011). Neuromorphic silicon neuron circuits. *Frontiers in neuroscience*, 5, 73.

Izhikevich, E. M. (2004). Which model to use for cortical spiking neurons? *IEEE transactions on neural networks*, 15(5), 1063–1070.

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

Jaeger, H. (2003). Adaptive nonlinear system identification with echo state networks. In *Advances in neural information processing systems* (pp. 609–616).

Jang, H., Simeone, O., Gardner, B., & Gruning, A. (2019). An Introduction to Probabilistic Spiking Neural Networks: Probabilistic Models, Learning Rules, and Applications. *IEEE Signal Processing Magazine*, 36(6), 64–77. Conference Name: IEEE Signal Processing Magazine.

Jercog, D., Roxin, A., Bartho, P., Luczak, A., Compte, A., & de la Rocha, J. (2017). Up-down cortical dynamics reflect state transitions in a bistable network. *Elife*, 6, e22425.

Jiang, F., Jiang, Y., Zhi, H., Dong, Y., Li, H., Ma, S., Wang, Y., Dong, Q., Shen, H., & Wang, Y. (2017). Artificial intelligence in healthcare: past, present and future. *Stroke and vascular neurology*, 2(4).

Joshi, V., Le Gallo, M., Haefeli, S., Boybat, I., Nandakumar, S. R., Piveteau, C., Dazzi, M., Rajendran, B., Sebastian, A., & Eleftheriou, E. (2020). Accurate deep neural network inference using computational phase-change memory. *Nature communications*, 11(1), 1–13.

Kaiser, J., Mostafa, H., & Neftci, E. (2020). Synaptic Plasticity Dynamics for Deep Continuous Local Learning (DECOLLE). *Front. Neurosci.*, 14, 424.

Kan, S., Nakajima, K., Takeshima, Y., Asai, T., Kuwahara, Y., & Akai-Kasaya, M. (2021). Simple reservoir computing capitalizing on the nonlinear response of materials: Theory and physical implementations. *Physical review applied*, 15(2), 024030.

Kappel, D., Habenschuss, S., Legenstein, R., & Maass, W. (2015). Synaptic sampling: A bayesian approach to neural network plasticity and rewiring. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 28* (pp. 370–378). Curran Associates, Inc.

Karasenko, V. (2020). *Von Neumann bottlenecks in non-von Neumann computing architectures*. PhD thesis, Universität Heidelberg.

Keck, T., Scheuss, V., Jacobsen, R. I., Wierenga, C. J., Eysel, U. T., Bonhoeffer, T., & Hübener, M. (2011). Loss of sensory input causes rapid structural changes of inhibitory neurons in adult mouse visual cortex. *Neuron*, 71(5), 869–882.

Keck, T., Toyoizumi, T., Chen, L., Doiron, B., Feldman, D. E., Fox, K., Gerstner, W., Haydon, P. G., Hübener, M., Lee, H.-K., et al. (2017). Integrating hebbian and homeostatic plasticity: the current state of the field and future research directions. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 372(1715), 20160158.

Khan, W. Z., Ahmed, E., Hakak, S., Yaqoob, I., & Ahmed, A. (2019). Edge computing: A survey. *Future Generation Computer Systems*, 97, 219–235.

Kheradpisheh, S. R., Ganjtabesh, M., Thorpe, S. J., & Masquelier, T. (2018). STDP-based spiking deep convolutional neural networks for object recognition. *Neural Networks*, 99, 56–67.

Kiebel, S. J., Daunizeau, J., & Friston, K. J. (2008). A hierarchy of time-scales and the brain. *PLoS Comput Biol*, 4(11), e1000209.

Kiene, G. (2017). Mixed-signal neuron and readout circuits for a neuromorphic system. Masterthesis, Universität Heidelberg.

Kingma, D. P. & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kinouchi, O. & Copelli, M. (2006). Optimal dynamical range of excitable networks at criticality. *Nature physics*, 2(5), 348–351.

Kleider, M. (2017). *Neuron Circuit Characterization in a Neuromorphic System*. PhD thesis, Universität Heidelberg.

Knight, J. C. & Nowotny, T. (2018). Gpus outperform current hpc and neuromorphic solutions in terms of speed and energy when simulating a highly-connected cortical model. *Frontiers in neuroscience*, 12, 941.

Knoblauch, A., Palm, G., & Sommer, F. T. (2010). Memory capacities for synaptic and structural plasticity. *Neural Computation*, 22(2), 289–341.

Knysh, P. & Korkolis, Y. (2016). Blackbox: A procedure for parallel optimization of expensive black-box functions. *arXiv preprint arXiv:1605.00998*.

Köhn, A., Stegen, F., & Baumann, T. (2016). Mining the spoken wikipedia for speech data and beyond. In N. C. C. Chair), K. Choukri, T. Declerck, M. Grobelnik, B. Maegaard, J. Mariani, A. Moreno, J. Odijk, & S. Piperidis (Eds.), *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)* Paris, France: European Language Resources Association (ELRA).

Kolb, B. (2013). *Brain plasticity and behavior*. Psychology Press.

Kreft, M. (2019). Structural plasticity for feature selection in auditory stimuli on neuromorphic hardware. Bachelor thesis, Heidelberg University.

Kreft, M. (2021). Input-induced dynamical states in homeostatically regulated neuromorphic recurrent neural networks. Master thesis, Heidelberg University.

Kriener, B., Enger, H., Tetzlaff, T., Plesser, H. E., Gewaltig, M.-O., & Einevoll, G. T. (2014). Dynamics of self-sustained asynchronous-irregular activity in random networks of spiking neurons with strong synapses. *Frontiers in computational neuroscience*, 8, 136.

Kriener, L., Göltz, J., & Petrovici, M. A. (2021). The yin-yang dataset. *arXiv preprint arXiv:2102.08211*.

Krizhevsky, A., Hinton, G., et al. (2009). *Learning multiple layers of features from tiny images*. Technical report, Citeseer.

Kumar, N. (1998). Investigation of silicon auditory models and generalization of linear discriminant analysis for improved speech recognition.

Kungl, A. F., Schmitt, S., Klähn, J., Müller, P., Baumbach, A., Dold, D., Kugele, A., Gürtler, N., Müller, E., Koke, C., et al. (2018). Generative models on accelerated neuromorphic hardware. *arXiv preprint arXiv:1807.02389*.

Kungl, A. F., Schmitt, S., Klähn, J., Müller, P., Baumbach, A., Dold, D., Kugele, A., Müller, E., Koke, C., Kleider, M., et al. (2019). Accelerated physical emulation of bayesian inference in spiking neural networks. *Frontiers in neuroscience*, 13, 1201.

Lamprecht, R. & LeDoux, J. (2004). Structural plasticity and memory. *Nature Reviews Neuroscience*, 5(1), 45.

Langton, C. G. (1990). Computation at the edge of chaos: phase transitions and emergent computation. *Physica D: Nonlinear Phenomena*, 42(1-3), 12–37.

Lapicque, L. (1907). Recherches quantitatives sur l'excitation electrique des nerfs traitee comme une polarization. *Journal de Physiologie et de Pathologie Général*, 9, 620–635.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *nature*, 521(7553), 436–444.

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., et al. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278–2324.

Lee, C., Sarwar, S. S., & Roy, K. (2019). Enabling Spike-based Backpropagation in State-of-the-art Deep Neural Network Architectures. *arXiv:1903.06379 [cs]*. arXiv: 1903.06379.

Lee, J. H., Delbruck, T., & Pfeiffer, M. (2016). Training deep spiking neural networks using backpropagation. *Frontiers in neuroscience*, 10, 508.

Legenstein, R. & Maass, W. (2007). Edge of chaos and prediction of computational performance for neural circuit models. *Neural Networks*, 20(3), 323–334.

Leibfried, A. (2021). On-chip calibration and closed-loop experiments on analog neuromorphic hardware. Masterarbeit, Universität Heidelberg.

Leonard, R. G. & Doddington, G. R. (1991). A speaker-independent connected-digit database. *Instruments Incorporated, Central Research Laboratories, Dallas, TX*, 75266.

Levina, A., Herrmann, J. M., & Geisel, T. (2007). Dynamical synapses causing self-organized criticality in neural networks. *Nature physics*, 3(12), 857–860.

Lichtsteiner, P., Posch, C., & Delbruck, T. (2008). A 128x128 120 dB 15us Latency Asynchronous Temporal Contrast Vision Sensor. *IEEE Journal of Solid-State Circuits*, 43(2), 566–576.

Liu, C., Bellec, G., Vogginger, B., Kappel, D., Partzsch, J., Neumärker, F., Höppner, S., Maass, W., Furber, S. B., Legenstein, R., et al. (2018). Memory-efficient deep learning on a spinnaker 2 prototype. *Frontiers in neuroscience*, 12.

Livi, P. & Indiveri, G. (2009). A current-mode conductance-based silicon neuron for address-event neuromorphic systems. In *2009 IEEE international symposium on circuits and systems* (pp. 2898–2901).: IEEE.

Lizier, J. T. (2014). Jidt: An information-theoretic toolkit for studying the dynamics of complex systems. *Frontiers in Robotics and AI*, 1, 11.

Lizier, J. T., Bertschinger, N., Jost, J., & Wibral, M. (2018). Information decomposition of target effects from multi-source interactions: Perspectives on previous, current and future work. *Entropy*, 20(4), 307.

Lizier, J. T., Prokopenko, M., & Zomaya, A. Y. (2008). The information dynamics of phase transitions in random boolean networks. In *ALIFE* (pp. 374–381).

Loewenstein, Y., Kuras, A., & Rumpel, S. (2011). Multiplicative dynamics underlie the emergence of the log-normal distribution of spine sizes in the neocortex in vivo. *Journal of Neuroscience*, 31(26), 9481–9488.

Loewenstein, Y., Yanover, U., & Rumpel, S. (2015). Predicting the dynamics of network connectivity in the neocortex. *Journal of Neuroscience*, 35(36), 12535–12544.

Loidolt, M., Rudelt, L., & Priesemann, V. (2020). Sequence memory in recurrent neuronal network can develop without structured input. *bioRxiv*.

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

Maass, W., Natschlager, T., & Markram, H. (2003). A model for real-time computation in generic neural microcircuits. *Advances in neural information processing systems*, (pp. 229–236).

Maier, K. D., Beckstein, C., Blickhan, R., Erhard, W., & Fey, D. (1999). A multi-layer-perceptron neural network hardware based on 3d massively parallel optoelectronic circuits. In *Proceedings. 6th International Conference on Parallel Interconnects (PI'99)(Formerly Known as MPPOI)* (pp. 73–80).: IEEE.

Makkeh, A., Theis, D., & Vicente, R. (2018). Broja-2pid: A robust estimator for bivariate partial information decomposition. *Entropy*, 20(4), 271.

Marković, D., Mizrahi, A., Querlioz, D., & Grollier, J. (2020). Physics for neuromorphic computing. *Nature Reviews Physics*, (pp. 1–12).

Markram, H., Lübke, J., Frotscher, M., & Sakmann, B. (1997). Regulation of synaptic efficacy by coincidence of postsynaptic APs and EPSPs. *Science*, 275(5297), 213–215.

Markram, H. & Tsodyks, M. (1996). Redistribution of synaptic efficacy between neocortical pyramidal neurons. *Nature*, 382(6594), 807.

Markram, H., Wang, Y., & Tsodyks, M. (1998). Differential signaling via the same axon of neocortical pyramidal neurons. *Proceedings of the National Academy of Sciences*, 95(9), 5323–5328.

Marsaglia, G. et al. (2003). Xorshift rngs. *Journal of Statistical Software*, 8(14), 1–6.

Marzvanyan, A. & Alhawaj, A. F. (2019). Physiology, sensory receptors. *StatPearls Publishing*.

Matsuzaki, M., Ellis-Davies, G. C., Nemoto, T., Miyashita, Y., Iino, M., & Kasai, H. (2001). Dendritic spine geometry is critical for ampa receptor expression in hippocampal ca1 pyramidal neurons. *Nature neuroscience*, 4(11), 1086.

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

Meddis, R. (1986). Simulation of mechanical to neural transduction in the auditory receptor. *The Journal of the Acoustical Society of America*, 79(3), 702–711.

Meddis, R. (1988). Simulation of auditory–neural transduction: Further studies. *The Journal of the Acoustical Society of America*, 83(3), 1056–1063.

Meddis, R., Hewitt, M. J., & Shackleton, T. M. (1990). Implementation details of a computation model of the inner hair-cell auditory-nerve synapse. *The Journal of the Acoustical Society of America*, 87(4), 1813–1816.

Mediano, P. A. & Shanahan, M. (2017). Balanced information storage and transfer in modular spiking neural networks. *arXiv preprint arXiv:1708.04392*.

Meisel, C. & Gross, T. (2009). Adaptive self-organization in a realistic neural network model. *Physical Review E*, 80(6), 061917.

Memmesheimer, R.-M., Rubin, R., Ölveczky, B. P., & Sompolinsky, H. (2014). Learning Precisely Timed Spikes. *Neuron*, 82(4), 925–938.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., & Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. In *International conference on machine learning* (pp. 1928–1937).: PMLR.

Mohemmed, A., Schliebs, S., Matsuda, S., & Kasabov, N. (2012). Span: spike pattern association neuron for learning spatio-temporal spike patterns. *Int. J. Neur. Syst.*, 22(04), 1250012.

Mohri, M., Rostamizadeh, A., & Talwalkar, A. (2018). *Foundations of machine learning*. MIT press.

Mongillo, G., Barak, O., & Tsodyks, M. (2008). Synaptic theory of working memory. *Science*, 319(5869), 1543–1546.

Moore, G. E. et al. (1965). Cramming more components onto integrated circuits.

Moradi, S., Qiao, N., Stefanini, F., & Indiveri, G. (2018). A scalable multicore architecture with heterogeneous memory structures for dynamic neuromorphic asynchronous processors (dynaps). *IEEE transactions on biomedical circuits and systems*, 12(1), 106–122.

Moretti, P. & Muñoz, M. A. (2013). Griffiths phases and the stretching of criticality in brain networks. *Nature communications*, 4(1), 1–10.

Morrison, A., Diesmann, M., & Gerstner, W. (2008). Phenomenological models of synaptic plasticity based on spike timing. *Biological cybernetics*, 98(6), 459–478.

Mostafa, H. (2017). Supervised learning based on temporal coding in spiking neural networks. *IEEE transactions on neural networks and learning systems*, 29(7), 3227–3235.

Mozafari, M., Ganjtabesh, M., Nowzari-Dalini, A., & Masquelier, T. (2019). SpykeTorch: Efficient Simulation of Convolutional Spiking Neural Networks with at most one Spike per Neuron. *arXiv:1903.02440 [cs, q-bio]*. arXiv: 1903.02440.

Mozilla (2019). Mozilla common voice.

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

Munoz, M. A. (2018a). Colloquium: Criticality and dynamical scaling in living systems. *Reviews of Modern Physics*, 90(3), 031001.

Munoz, M. A. (2018b). Colloquium: Criticality and dynamical scaling in living systems. *Reviews of Modern Physics*, 90(3), 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., et al. (2014). A hierarchy of intrinsic timescales across primate cortex. *Nature neuroscience*, 17(12), 1661–1663.

Naud, R., Marcille, N., Clopath, C., & Gerstner, W. (2008). Firing patterns in the adaptive exponential integrate-and-fire model. *Biological cybernetics*, 99(4-5), 335.

Neftci, E., Chicca, E., Indiveri, G., & Douglas, R. (2011a). A systematic method for configuring vlsi networks of spiking neurons. *Neural computation*, 23(10), 2457–2497.

Neftci, E., Chicca, E., Indiveri, G., & Douglas, R. (2011b). A systematic method for configuring VLSI networks of spiking neurons. *Neural Computation*, 23(10), 2457–2497.

Neftci, E., Das, S., Pedroni, B., Kreutz-Delgado, K., & Cauwenberghs, G. (2014). Event-driven contrastive divergence for spiking neuromorphic systems. *Frontiers in neuroscience*, 7, 272.

Neftci, E. & Indiveri, G. (2010). A device mismatch compensation method for VLSI neural networks. In *2010 Biomedical Circuits and Systems Conference (BioCAS)* (pp. 262–265).

Neftci, E. O., Augustine, C., Paul, S., & Detorakis, G. (2017). Event-driven random back-propagation: Enabling neuromorphic deep learning machines. *Frontiers in Neuroscience*, 11, 324.

Neftci, E. O., Mostafa, H., & Zenke, F. (2019a). Surrogate Gradient Learning in Spiking Neural Networks. *arXiv:1901.09948 [cs, q-bio]*. arXiv: 1901.09948.

Neftci, E. O., Mostafa, H., & Zenke, F. (2019b). Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks. *IEEE Signal Processing Magazine*, 36(6), 51–63.

Nernst, W. (1889). Die elektromotorische wirksamkeit der jonen. *Zeitschrift für physikalische Chemie*, 4(1), 129–181.

Neto, J. P., de Aguiar, M. A., Brum, J. A., & Bornholdt, S. (2017). Inhibition as a determinant of activity and criticality in dynamical networks. *arXiv preprint arXiv:1712.08816*.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., & Ng, A. Y. (2011). Reading Digits in Natural Images with Unsupervised Feature Learning.

Nevian, T. & Sakmann, B. (2006). Spine ca2+ signaling in spike-timing-dependent plasticity. *Journal of Neuroscience*, 26(43), 11001–11013.

Nicola, W. & Clopath, C. (2017). Supervised learning in spiking neural networks with FORCE training. *Nature Communications*, 8(1), 2208.

O'Connor, P. & Welling, M. (2016). Deep spiking networks. *arXiv preprint arXiv:1602.08323*.

Orchard, G., Jayawant, A., Cohen, G. K., & Thakor, N. (2015). Converting Static Image Datasets to Spiking Neuromorphic Datasets Using Saccades. *Front. Neurosci.*, 9.

Pakkenberg, B., Pelvig, D., Marner, L., Bundgaard, M. J., Gundersen, H. J. G., Nyengaard, J. R., & Regeur, L. (2003). Aging and the human neocortex. *Experimental gerontology*, 38(1-2), 95–99.

Panayotov, V., Chen, G., Povey, D., & Khudanpur, S. (2015). Librispeech: an asr corpus based on public domain audio books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 5206–5210).: IEEE.

Paquot, Y., Duport, F., Smerieri, A., Dambre, J., Schrauwen, B., Haelterman, M., & Massar, S. (2012). Optoelectronic reservoir computing. *Scientific reports*, 2(1), 1–6.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., & Lerer, A. (2017). Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*.

Patterson, D. A. & Hennessy, J. L. (2016). *Computer Organization and Design ARM Edition: The Hardware Software Interface*. Morgan kaufmann.

Payeur, A., Guerguiev, J., Zenke, F., Richards, B., & Naud, R. (2020). Burst-dependent synaptic plasticity can coordinate learning in hierarchical circuits. *bioRxiv*.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.

Pereda, A. E. (2014). Electrical synapses and their functional interactions with chemical synapses. *Nature Reviews Neuroscience*, 15(4), 250–263.

Perez-Nieves, N. & Goodman, D. F. (2021). Sparse spiking gradient descent. *arXiv preprint arXiv:2105.08810*.

Perez-Nieves, N., Leung, V. C., Dragotti, P. L., & Goodman, D. F. (2021). Neural heterogeneity promotes robust learning. *bioRxiv*, (pp. 2020–12).

Perkel, D. H. & Bullock, T. H. (1968). Neural coding. *Neurosciences Research Program Bulletin*.

Pfeiffer, M. & Pfeil, T. (2018). Deep Learning With Spiking Neurons: Opportunities and Challenges. *Front. Neurosci.*, 12.

Pfeil, T., Grübl, A., Jeltsch, S., Müller, E., Müller, P., Petrovici, M. A., Schmuker, M., Brüderle, D., Schemmel, J., & Meier, K. (2013). Six networks on a universal neuromorphic computing substrate. *Frontiers in Neuroscience*, 7, 11.

Pfister, J.-P., Toyoizumi, T., Barber, D., & Gerstner, W. (2006). Optimal Spike-Timing-Dependent Plasticity for Precise Action Potential Firing in Supervised Learning. *Neural Computation*, 18(6), 1318–1348.

Poil, S.-S., Hardstone, R., Mansvelder, H. D., & Linkenkaer-Hansen, K. (2012). Critical-state dynamics of avalanches and oscillations jointly emerge from balanced excitation/inhibition in neuronal networks. *Journal of Neuroscience*, 32(29), 9817–9823.

Poirazi, P. & Mel, B. W. (2001). Impact of active dendrites and structural plasticity on the memory capacity of neural tissue. *Neuron*, 29(3), 779–796.

Ponulak, F. & Kasiński, A. (2009). Supervised Learning in Spiking Neural Networks with ReSuMe: Sequence Learning, Classification, and Spike Shifting. *Neural Computation*, 22(2), 467–510.

Potjans, T. C. & Diesmann, M. (2014). The cell-type specific cortical microcircuit: relating structure and activity in a full-scale spiking network model. *Cerebral cortex*, 24(3), 785–806.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. (2007). *Numerical recipes 3rd edition: The art of scientific computing*. Cambridge university press.

Priesemann, V., Munk, M. H., & Wibral, M. (2009). Subsampling effects in neuronal avalanche distributions recorded in vivo. *BMC neuroscience*, 10(1), 40.

Priesemann, V., Wibral, M., Valderrama, M., Pröpper, R., Le Van Quyen, M., Geisel, T., Triesch, J., Nikolić, D., & Munk, M. H. (2014). Spike avalanches in vivo suggest a driven, slightly subcritical brain state. *Frontiers in systems neuroscience*, 8.

Protachevicz, P. R., Borges, F. S., Lameu, E. L., Ji, P., Iarosz, K. C., Kihara, A. H., Caldas, I. L., Szezech Jr, J. D., Baptista, M. S., Macau, E. E., et al. (2019). Bistable firing pattern in a neural network model. *Frontiers in computational neuroscience*, 13, 19.

Purves, D., Augustine, G., Fitzpatrick, D., Katz, L., LaMantia, A., McNamara, J., & Williams, S. (2001). Neuroscience 2nd edition. sunderland (ma) sinauer associates. *Types of Eye Movements and Their Functions*.

Qiao, N., Mostafa, H., Corradi, F., Osswald, M., Stefanini, F., Sumislawska, D., & Indiveri, G. (2015). A reconfigurable on-line learning spiking neuromorphic processor comprising 256 neurons and 128k synapses. *Frontiers in Neuroscience*, 9, 141.

Reyserhove, H., Reynders, N., & Dehaene, W. (2014). Ultra-low voltage datapath blocks in 28nm utbb fd-soi. In *2014 IEEE Asian Solid-State Circuits Conference (A-SSCC)* (pp. 49–52).: IEEE.

Rezende, D. J. & Gerstner, W. (2014). Stochastic variational learning in recurrent spiking networks. *Front. Comput. Neurosci*, 8, 38.

Rongala, U. B., Mazzoni, A., & Oddo, C. M. (2015). Neuromorphic artificial touch for categorization of naturalistic textures. *IEEE transactions on neural networks and learning systems*, 28(4), 819–829.

Ros, P. M., Crepaldi, M., & Demarchi, D. (2015). A hybrid quasi-digital/neuromorphic architecture for tactile sensing in humanoid robots. In *2015 6th International Workshop on Advances in Sensors and Interfaces (IWASI)* (pp. 126–130).: IEEE.

Rossing, T. (2007). *Springer handbook of acoustics*. Springer Science & Business Media.

Rothman, J. S., Young, E. D., & Manis, P. B. (1993). Convergence of auditory nerve fibers onto bushy cells in the ventral cochlear nucleus: implications of a computational model. *Journal of Neurophysiology*, 70(6), 2562–2583.

Rousseau, A., Deléglise, P., & Esteve, Y. (2012). Ted-lium: an automatic speech recognition dedicated corpus. In *LREC* (pp. 125–129).

Roy, K., Jaiswal, A., & Panda, P. (2019). Towards spike-based machine intelligence with neuromorphic computing. *Nature*, 575(7784), 607–617.

Roy, K., Sharad, M., Fan, D., & Yogendra, K. (2014a). Brain-inspired computing with spin torque devices. In *2014 Design, Automation & Test in Europe Conference & Exhibition (DATE)* (pp. 1–6).: IEEE.

Roy, S., Banerjee, A., & Basu, A. (2014b). Liquid state machine with dendritically enhanced readout for low-power, neuromorphic vlsi implementations. *IEEE transactions on biomedical circuits and systems*, 8(5), 681–695.

Roy, S. & Basu, A. (2016). An online unsupervised structural plasticity algorithm for spiking neural networks. *IEEE transactions on neural networks and learning systems*, 28(4), 900–910.

Russell, S. J. & Norvig, P. (2016). *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,.

Rückauer, B., Känzig, N., Liu, S.-C., Delbruck, T., & Sandamirskaya, Y. (2019). Closing the Accuracy Gap in an Event-Based Visual Recognition Task. *arXiv:1906.08859 [cs]*. arXiv: 1906.08859.

Sansen, W. M. (2007). *Analog design essentials*, volume 859. Springer Science & Business Media.

Schemmel, J., Billaudelle, S., Dauer, P., & Weis, J. (2020). Accelerated analog neuromorphic computing. *arXiv preprint arXiv:2003.11996*.

Schemmel, J., Brüderle, D., Grübl, A., Hock, M., Meier, K., & Millner, S. (2010). A wafer-scale neuromorphic hardware system for large-scale neural modeling. *Proceedings of the 2010 IEEE International Symposium on Circuits and Systems (ISCAS"10)*, (pp. 1947–1950).

Schemmel, J., Brüderle, D., Meier, K., & Ostendorf, B. (2007). Modeling synaptic plasticity within networks of highly accelerated i amp;f neurons. In *2007 IEEE International Symposium on Circuits and Systems* (pp. 3367–3370).

Schemmel, J., Kriener, L., Müller, P., & Meier, K. (2017). An accelerated analog neuromorphic hardware system emulating nmda-and calcium-based non-linear dendrites. In *2017 International Joint Conference on Neural Networks (IJCNN)* (pp. 2217–2226).: IEEE.

Schmitt, S., Klähn, J., Bellec, G., Grübl, A., Güttler, M., Hartel, A., Hartmann, S., Husmann, D., Husmann, K., Jeltsch, S., et al. (2017). Neuromorphic hardware in the loop: Training a deep spiking network on the brainscales wafer-scale system. In *2017 International Joint Conference on Neural Networks (IJCNN)* (pp. 2227–2234).: IEEE.

Schmuker, M., Pfeil, T., & Nawrot, M. P. (2014). A neuromorphic network for generic multivariate data classification. *Proceedings of the National Academy of Sciences*, 111(6), 2081–2086.

Schreiber, K. (2021). *Accelerated neuromorphic cybernetics*. PhD thesis, Universität Heidelberg.

Schreiber, T. (2000). Measuring information transfer. *Physical review letters*, 85(2), 461.

Schuman, C. D., Potok, T. E., Patton, R. M., Birdwell, J. D., Dean, M. E., Rose, G. S., & Plank, J. S. (2017). A survey of neuromorphic computing and neural networks in hardware. *arXiv preprint arXiv:1705.06963*.

Schumann, D. (2019). dev-core. https://dev-core.org/. Accessed: 2019-08-23.

Schürmann, F., Meier, K., & Schemmel, J. (2005). Edge of chaos computation in mixed-mode vlsi-a hard liquid. In *Advances in neural information processing systems* (pp. 1201–1208).

Shannon, C. E. (1948). A mathematical theory of communication. *The Bell System Technical Journal*, 27(3), 379–423.

Shew, W. L. & Plenz, D. (2013). The functional benefits of criticality in the cortex. *The neuroscientist*, 19(1), 88–100.

Shew, W. L., Yang, H., Yu, S., Roy, R., & Plenz, D. (2011). Information capacity and transmission are maximized in balanced cortical networks with neuronal avalanches. *Journal of neuroscience*, 31(1), 55–63.

Shi, W. & Dustdar, S. (2016). The promise of edge computing. *Computer*, 49(5), 78–81.

Shimodaira, H. (1994). A weight value initialization method for improving learning performance of the backpropagation algorithm in neural networks. In *Proceedings Sixth International Conference on Tools with Artificial Intelligence. TAI 94* (pp. 672–675).: IEEE.

Shin, C.-W. & Kim, S. (2006). Self-organized criticality and scale-free properties in emergent functional neural networks. *Physical Review E*, 74(4), 045101.

Shouval, H. Z., Bear, M. F., & Cooper, L. N. (2001). A unified model of calcium dependent synaptic plasticity. *Proc. Nat. Acad. Sci*, 99(16), 10.

Shrestha, S. B. & Orchard, G. (2018). SLAYER: Spike Layer Error Reassignment in Time. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 31* (pp. 1419–1428). Curran Associates, Inc.

Shyu, J.-B., Temes, G. C., & Krummenacher, F. (1984). Random error effects in matched mos capacitors and current sources. *IEEE Journal of solid-state circuits*, 19(6), 948–956.

Sieroka, N., Dosch, H. G., & Rupp, A. (2006). Semirealistic models of the cochlea. *The Journal of the Acoustical Society of America*, 120(1), 297–304.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. (2016). Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587), 484–489.

Skowronski, M. D. & Harris, J. G. (2007). Automatic speech recognition using a predictive echo state network classifier. *Neural networks*, 20(3), 414–423.

Spiess, R., George, R., Cook, M., & Diehl, P. U. (2016). Structural plasticity denoises responses and improves learning speed. *Frontiers in Computational Neuroscience*, 10, 93.

Stepp, N., Plenz, D., & Srinivasa, N. (2015). Synaptic plasticity enables adaptive self-tuning critical networks. *PLoS computational biology*, 11(1), e1004043.

Sterling, P. & Laughlin, S. (2015). *Principles of neural design*. MIT Press.

Stern, E. A., Kincaid, A. E., & Wilson, C. J. (1997). Spontaneous subthreshold membrane potential fluctuations and action potential variability of rat corticostriatal and striatal neurons in vivo. *Journal of neurophysiology*, 77(4), 1697–1715.

Stiles, J. & Jernigan, T. L. (2010). The basics of brain development. *Neuropsychology review*, 20(4), 327–348.

Stöckl, C. & Maass, W. (2021). Optimized spiking neurons can classify images with high accuracy through temporal coding with two spikes. *Nature Machine Intelligence*, (pp. 1–9).

Stradmann, Y., Billaudelle, S., Breitwieser, O., Ebert, F. L., Emmel, A., Husmann, D., Ilmberger, J., Müller, E., Spilger, P., Weis, J., et al. (2021). Demonstrating analog inference on the brainscales-2 mobile system. *arXiv preprint arXiv:2103.15960*.

Stromatias, E., Neil, D., Pfeiffer, M., Galluppi, F., Furber, S. B., & Liu, S.-C. (2015). Robustness of spiking deep belief networks to noise and reduced bit precision of neuro-inspired hardware platforms. *Frontiers in Neuroscience*, 9, 222.

Strubell, E., Ganesh, A., & McCallum, A. (2019). Energy and policy considerations for deep learning in nlp. *arXiv preprint arXiv:1906.02243*.

Südhof, T. C. & Malenka, R. C. (2008). Understanding synapses: past, present, and future. *Neuron*, 60(3), 469–476.

Sutton, R. S. & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

Tavanaei, A., Ghodrati, M., Kheradpisheh, S. R., Masquelier, T., & Maida, A. (2018). Deep learning in spiking neural networks. *Neural Networks*.

Tax, T., Mediano, P. A., & Shanahan, M. (2017). The partial information decomposition of generative neural network models. *Entropy*, 19(9), 474.

Tetzlaff, C., Okujeni, S., Egert, U., Wörgötter, F., & Butz, M. (2010). Self-organized criticality in developing neuronal networks. *PLoS computational biology*, 6(12), e1001013.

Thalmeier, D., Uhlmann, M., Kappen, H. J., & Memmesheimer, R.-M. (2016). Learning Universal Computations with Spikes. *PLOS Comput Biol*, 12(6), e1004895.

Thimm, G. & Fiesler, E. (1995). Neural network initialization. In *International Workshop on Artificial Neural Networks* (pp. 535–542).: Springer.

Thrun, S. & Pratt, L. (2012). *Learning to learn*. Springer Science & Business Media.

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

Trachtenberg, J. T., Chen, B. E., Knott, G. W., Feng, G., Sanes, J. R., Welker, E., & Svoboda, K. (2002). Long-term in vivo imaging of experience-dependent synaptic plasticity in adult cortex. *Nature*, 420(6917), 788.

Tsodyks, M., Pawelzik, K., & Markram, H. (1998). Neural networks with dynamic synapses. *Neural computation*, 10(4), 821–835.

Turing, A. & Haugeland, J. (1950). *Computing machinery and intelligence*. MIT Press Cambridge, MA.

Turrigiano, G. G., Leslie, K. R., Desai, N. S., Rutherford, L. C., & Nelson, S. B. (1998). Activity-dependent scaling of quantal amplitude in neocortical neurons. *Nature*, 391(6670), 892–896.

Turrigiano, G. G. & Nelson, S. B. (2004). Homeostatic plasticity in the developing nervous system. *Nature reviews neuroscience*, 5(2), 97–107.

Urbanczik, R. & Senn, W. (2014). Learning by the dendritic prediction of somatic spiking. *Neuron*, 81(3), 521–528.

van Albada, S. J., Rowley, A. G., Senk, J., Hopkins, M., Schmidt, M., Stokes, A. B., Lester, D. R., Diesmann, M., & Furber, S. B. (2018). Performance comparison of the digital neuromorphic hardware spinnaker and the neural network simulation software nest for a full-scale cortical microcircuit model. *Frontiers in neuroscience*, 12, 291.

Verstraeten, D., Schrauwen, B., & Stroobandt, D. (2006). Reservoir-based techniques for speech recognition. In *The 2006 IEEE International Joint Conference on Neural Network Proceedings* (pp. 1050–1053).: IEEE.

Verstraeten, D., Schrauwen, B., Stroobandt, D., & Van Campenhout, J. (2005). Isolated word recognition with the liquid state machine: a case study. *Information Processing Letters*, 95(6), 521–528.

Viterbi, A. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE transactions on Information Theory*, 13(2), 260–269.

Von Neumann, J. (1993). First draft of a report on the edvac. *IEEE Annals of the History of Computing*, 15(4), 27–75.

Wallentowitz, S. (2021). Moore and more.

Wang, H. & Wagner, J. J. (1999). Priming-induced shift in synaptic plasticity in the rat hippocampus. *Journal of neurophysiology*, 82(4), 2024–2028.

Warden, P. (2018). Speech commands: A dataset for limited-vocabulary speech recognition. *arXiv preprint arXiv:1804.03209*.

Watson, H. W. & Galton, F. (1875). On the probability of the extinction of families. *The Journal of the Anthropological Institute of Great Britain and Ireland*, 4, 138–144.

Wei, C. & Winnicki, J. (1990). Estimation of the means in the branching process with immigration. *The Annals of Statistics*, (pp. 1757–1773).

Weis, J. (2020). Inference with artificial neural networks on neuromorphic hardware. Master's thesis, Universität Heidelberg.

Weis, J., Spilger, P., Billaudelle, S., Stradmann, Y., Emmel, A., Müller, E., Breitwieser, O., Grübl, A., Ilmberger, J., Karasenko, V., et al. (2020). Inference with artificial neural networks on analog neuromorphic hardware. In *IoT Streams for Data-Driven Predictive Maintenance and IoT, Edge, and Mobile for Embedded Machine Learning* (pp. 201–212). Springer.

Wen, W., Wu, C., Wang, Y., Chen, Y., & Li, H. (2016). Learning structured sparsity in deep neural networks. In *Advances in neural information processing systems* (pp. 2074–2082).

Werbos, P. J. (1990). Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10), 1550–1560.

Wessels, L. F. & Barnard, E. (1992). Avoiding false local minima by proper initialization of connections. *IEEE transactions on neural networks*, 3(6), 899–905.

Weymaere, N. & Martens, J.-P. (1994). On the initialization and optimization of multilayer perceptrons. *IEEE Transactions on Neural Networks*, 5(5), 738–751.

Whittington, M. A., Traub, R., Kopell, N., Ermentrout, B., & Buhl, E. (2000). Inhibition-based rhythms: experimental and mathematical observations on network dynamics. *International journal of psychophysiology*, 38(3), 315–336.

Wibral, M., Finn, C., Wollstadt, P., Lizier, J. T., & Priesemann, V. (2017a). Quantifying information modification in developing neural networks via partial information decomposition. *Entropy*, 19(9).

Wibral, M., Lizier, J., Vögler, S., Priesemann, V., & Galuske, R. (2014). Local active information storage as a tool to understand distributed neural information processing. *Frontiers in neuroinformatics*, 8, 1.

Wibral, M., Lizier, J. T., & Priesemann, V. (2015). Bits from brains for biologically inspired computing. *Frontiers in Robotics and AI*, 2, 5.

Wibral, M., Priesemann, V., Kay, J. W., Lizier, J. T., & Phillips, W. A. (2017b). Partial information decomposition as a unified approach to the specification of neural goal functions. *Brain and cognition*, 112, 25–38.

Williams, P. & Beer, R. (2010). Decomposing multivariate information. *arXiv preprint arXiv:1004.2515*.

Wilting, J., Dehning, J., Neto, J. P., Rudelt, L., Wibral, M., Zierenberg, J., & Priesemann, V. (2018). Operating in a reverberating regime enables rapid tuning of network states to task requirements. *Frontiers in systems neuroscience*, 12.

Wilting, J. & Priesemann, V. (2018). Inferring collective dynamical states from widely unobserved systems. *Nature Communications*, 9(1), 2325.

Wilting, J. & Priesemann, V. (2019). 25 years of criticality in neuroscience–established results, open controversies, novel concepts. *arXiv preprint arXiv:1903.05129*.

Wozniak, S., Pantazi, A., Bohnstingl, T., & Eleftheriou, E. (2020). Deep learning incorporating biologically inspired neural dynamics and in-memory computing. *Nature Machine Intelligence*, 2(6), 325–336. Number: 6 Publisher: Nature Publishing Group.

Wozniak, S., Pantazi, A., & Eleftheriou, E. (2018). Deep Networks Incorporating Spiking Neural Dynamics. *arXiv:1812.07040 [cs]*. arXiv: 1812.07040.

Wright, L. G., Onodera, T., Stein, M. M., Wang, T., Schachter, D. T., Hu, Z., & McMahon, P. L. (2021). Deep physical neural networks enabled by a backpropagation algorithm for arbitrary physical systems. *arXiv:2104.13386 [cond-mat, physics:physics]*.

Wunderlich, T., Kungl, A. F., Müller, E., Hartel, A., Stradmann, Y., Aamir, S. A., Grübl, A., Heimbrecht, A., Schreiber, K., Stöckel, D., et al. (2019). Demonstrating advantages of neuromorphic computation: a pilot study. *Frontiers in Neuroscience*, 13, 260.

Xu, T., Yu, X., Perlik, A. J., Tobin, W. F., Zweig, J. A., Tennant, K., Jones, T., & Zuo, Y. (2009). Rapid formation and selective stabilization of synapses for enduring motor memories. *Nature*, 462(7275), 915–919.

Yam, J. Y. & Chow, T. W. (2000). A weight initialization method for improving training speed in feedforward neural network. *Neurocomputing*, 30(1-4), 219–232.

Yamahachi, H., Marik, S. A., McManus, J. N., Denk, W., & Gilbert, C. D. (2009). Rapid axonal sprouting and pruning accompany functional reorganization in primary visual cortex. *Neuron*, 64(5), 719–729.

Yan, Y., Kappel, D., Neumärker, F., Partzsch, J., Vogginger, B., Höppner, S., Furber, S., Maass, W., Legenstein, R., & Mayr, C. (2019). Efficient reward-based structural plasticity on a spinnaker 2 prototype. *IEEE transactions on biomedical circuits and systems*, 13(3), 579–591.

Yasumatsu, N., Matsuzaki, M., Miyazaki, T., Noguchi, J., & Kasai, H. (2008). Principles of long-term dynamics of dendritic spines. *The Journal of Neuroscience*, 28(50), 13592–13608.

Yin, B., Corradi, F., & Bohté, S. M. (2020). Effective and efficient computation with multiple-timescale spiking recurrent neural networks. In *International Conference on Neuromorphic Systems 2020* (pp. 1–8).

Yuste, R. (2011). Dendritic spines and distributed circuits. *Neuron*, 71(5), 772–781.

Zambrano, D., Nusselder, R., Scholte, H. S., & Bohte, S. (2017). Efficient Computation in Adaptive Artificial Spiking Neural Networks. *arXiv:1710.04838 [cs]*. arXiv: 1710.04838.

Zapperi, S., Lauritsen, K. B., & Stanley, H. E. (1995). Self-organized branching processes: mean-field theory for avalanches. *Physical review letters*, 75(22), 4071.

Zenke, F. & Ganguli, S. (2018a). Superspike: Supervised learning in multilayer spiking neural networks. *Neural computation*, 30(6), 1514–1541.

Zenke, F. & Ganguli, S. (2018b). SuperSpike: Supervised Learning in Multilayer Spiking Neural Networks. *Neural Computation*, 30(6), 1514–1541.

Zenke, F. & Neftci, E. O. (2021). Brain-inspired learning on neuromorphic substrates. *Proceedings of the IEEE*, 109(5), 935–950.

Zenke, F. & Vogels, T. P. (2021). The remarkable robustness of surrogate gradient learning for instilling complex function in spiking neural networks. *Neural Computation*, 33(4), 899–925.

Zhang, T., Zeng, Y., Zhao, D., & Shi, M. (2018). A plasticity-centric approach to train the non-differential spiking neural networks. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Zierenberg, J., Wilting, J., & Priesemann, V. (2018). Homeostatic plasticity and external input shape neural network dynamics. *Phys. Rev. X*, 8, 031018.

Zierenberg, J., Wilting, J., Priesemann, V., & Levina, A. (2019). Tailored ensembles of neural networks optimize sensitivity to stimulus statistics. *arXiv preprint arXiv:1905.10401*.

Zohar, J., César, S., Jason, F., Yuxin, P., Hereman, N., & Adhish, T. (2018). Jakobovski/free-spoken-digit-dataset: v1.0.8.

Zucker, R. S. & Regehr, W. G. (2002). Short-term synaptic plasticity. *Annual review of physiology*, 64(1), 355–405.

Zuo, Y., Lin, A., Chang, P., & Gan, W.-B. (2005). Development of long-term dendritic spine stability in diverse regions of cerebral cortex. *Neuron*, 46(2), 181–189.

Zylberberg, J., Murphy, J. T., & DeWeese, M. R. (2011). A Sparse Coding Model with Synaptically Local Plasticity and Spiking Neurons Can Account for the Diverse Shapes of V1 Simple Cell Receptive Fields. *PLoS Comput Biol*, 7(10), e1002250.

Statement of Originality (Erklärung)

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

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

Heidelberg, October 15, 2021 _____