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
Andreas Baumbach
born in: Hünfeld, Deutschland

Oral examination: 2020 November 11th

From microscopic dynamics to ensemble behavior in spiking neural networks

Referees:
Dr. Johannes Schemmel (Heidelberg University)
Prof. Dr. Thomas Gasenzer (Heidelberg University)

From microscopic dynamics to ensemble behavior in spiking neural networks

Dynamical oddities and their functional implications

As the end of Moore's law nears and the energy demand for computing increases the search for alternative means of computation becomes more and more relevant. One natural paragon is the animal brain as one of the only known naturally occurring general-purpose computing systems. While its computing model is largely unexplained, it has been shown that biologically inspired artificial neurons – subject to high-frequency noise – can approximately implement Boltzmann machines. In the first part of this thesis we explore such a spike-based Bayesian computing framework, compare its observed dynamics to simpler stochastic samplers and develop an improved Markovian model for single LIF neurons. The second part of the thesis then focuses on ensemble phenomena, where we show that a nearest-neighbor connected lattice shows a qualitatively different phase diagrams for different microscopic neuron models. Nevertheless, we can recover the correct critical exponent γ in all cases. Finally, we show two functional demonstrations, including a representation of quantum states, on the accelerated mixed-signal neuromorphic system BrainScaleS. This paves the way for a better understanding of the capabilities of this computational model.

Von mikroskopischer Dynamik zu Ensembleverhalten von spikenden neuronalen Netzwerken

Dynamische Besonderheiten und ihre funktionalen Implikationen

Durch das sich abzeichnende Ende der Gültigkeit des Moorschen Gesetzes und des kontinuierlich steigenden Energiebedarfs der weltweiten Computerinfrastruktur steigt die Relevanz der Suche nach alternativen Berechnungskonzepten. Als eines der wenigen in der Natur auftretenden Berechnungsobjekte stellt das biologische Gehirn ein naheliegendes Vorbild da. Es ist gezeigt worden, dass biologisch inspirierte LIF Neuronen, unter hochfrequentem Poisson Stimulus, näherungsweise klassische Boltzmannmaschinen implementieren können. In der ersten Hälfte dieser Dissertation untersuchen wir diesen spike-basierte Bayesschen Ansatz, vergleichen dessen zu beobachtende Dynamik mit einfacheren statistischen Samplern und entwickeln schlußendlich ein verbessertes Markov'sches Modell. Im zweiten Teil dieser Dissertation untersuchen wir dann Ensemblephänomene, wo wir zeigen, dass die Phasendiagramme von nächsten-Nachbarn verbundener Gittern je nach Neuronmodell qualitative Unterschiede aufzeigen. Nichtsdestotrotz finden wir in allen Fällen den korrekten kritischen Exponenenten γ. Schlußendlich präsentieren wir noch zwei funktionale Anwendungen, unter anderem die Repräsentation quantenmechanischer Zustände, auf dem beschleunigten mixed-signal System BrainScaleS.

Contents

1. Introduction

When we think about computers, we tend to think about these boxes under our desks or the laptops in our bags and recently the phones in our pockets. Computers have become truly ubiquitous to a point where modern human life could not exists without them. At least not without dramatic changes, from how supply chains and logistics in our society work to how we communicate in our personal lives. This ubiquity is largely rooted in the amount of additional compute power that became available. The number of transistors – the building blocks of processors – per unit area roughly doubled every one to two years [Moore et al., 1965]. This, so-called Moore's law, yielded similarly increased performance over time. In combination with our improved understanding of general computing concepts this allowed us to build compiler technology that permit very abstract problem description. For problems which are sufficiently precisely understood there exists an enormous amount of support infrastructure – reducing the required level of understanding for the end user – which admit the ubiquitous application of raw compute power.

This ability comes at price in terms of required energy consumption. While the transistor sizes shrank to a few nanometers [Yeap et al., 2019], the energy density increased. Due to the currently reached feature sizes, which are only two orders of magnitude larger than the size of a single atom $\approx 0.1\,\text{nm}$, the end of Moore's law is near or already reached[1] [Khan et al., 2018]. Due to the exponential growth of the *general*-purpose computing systems, *specialized* solutions tended to provide limited, at least time-wise, value (for example the GRAPE boards [Ebisuzaki et al., 1993] for astrophysical N-body simulations were superseded by standard graphics cards). Nowadays, as the natural improvements from Moore's law diminish, there is an emergence of a more diverse set of (co-)processors with different performance characteristics. These are the spiritual successors of dedicated on-chip compute circuits (e.g. floating point executors) and optimize highly specific workloads, be that BitCoin mining or artificial network evaluations [Jouppi et al., 2017]. These efforts are partially driven by the desire for accelerated computations, but also through the increased energy efficiency and thus economical cost[2]. By 2030, it is expected that between 8 % and 50 % of the global energy consumption will be expended for computing [Andrae and Edler, 2015]. It is this latter point that makes the search for new compute paradigms and platforms, if not desperately needed, then at least very valuable.

Besides the obvious elephant in the room of quantum computing[3], it is the human

[1] At least if we continue to build transistors from atoms.

[2] We save the discussion about the ecological impact for a different day.

[3] Which will always be well-financed as it promises to efficiently solve the problem of integer factorization [Shor, 1994]. As this underpins all of our modern cryptography, there is an obvious incentive there.

brain that is a "computer" which occurs in nature. There are two ways to motivate the research of the function of the human brain: On the one hand, we would like to understand it as part of the medical research, i.e., the alleviation of physical trauma and cognitive diseases. Here, we would ideally want to understand its mechanics in a way that we can "repair" it, i.e., we want to understand the specific functionality of the brain. However, if we are simply interested in the computational properties, we look at the general properties that make it the only truly intelligent[4] object in nature. While state-of-the-art implementations of a Go player, which only selects the moves and does not actually places the stones, or a StarCraft-II bot can outclass their human opponents, these implementations require multiple kilowatts of power [Silver et al., 2017, Vinyals et al., 2019]. The human brain can accomplish these tasks within its power budget of 20 W [Drubach, 2000]. Even though this represents a fifth of the power consumption of the human body [Rigden, 1996] – thereby generating a significant amount of evolutionary pressure – most of it is base load and not dependent on the specific higher cognitive task like the selection of the next Go move.

In a way it is the flexibility of the general-purpose von-Neumann architecture with its fundamental separation between memory and computation, that dictates its energy requirements. Data has to move from memory to the processing unit and the result is then written back to memory. It is this transfer that generates most of the energy cost. Unlike von-Neumann computers, the brain does not distinguish between compute components and memory components. Rather it is a collection of billions of neurons that form a directed and weighted compute graph through which the sensory input propagates in order to generate the action. It is in this connection structure that our memory has to be embedded.

Modern, neural-network-based machine learning uses this kind of "brain-inspired" computational graph in order to mimic the implementation of the human brain. To understand why and how the brain works – and why typical neural networks are fundamentally different – it is instructive to think about what traditional algorithmic computing does well and where it struggles. Taking the example of image recognition: Two, at least superficially, near-identical tasks of a) deciding whether two copies of an image are identical or b) whether two images show the same object, require vastly different algorithmic implementations. For the first task it is sufficient to compare the two images pixel-by-pixel. The problem here is solely that comparing all 8 million pixel of a 4K-image is tedious. The second task, on the other hand, requires an *understanding* of what is depicted in the image. It requires context and, up to a point, personal judgment of what we still consider to be the *same* object. This second-to-last sentence is a fascinating example of human language. Neither *requires* nor *understanding* nor *depicted* has a readily available – in the form of a precise mathematical statement – definition of their meaning. It is this kind of fuzzy task description – where we humans can all agree on the meaning, but where we are unable to generate a precise definition which can be written down as an algorithmic solution – for which we are looking for "intelligent" solutions.

Traditional neural networks "solve" this by using vast amounts of pre-labeled examples

[4] There is an argument to be made that we humans just delude ourselves, but we will ignore that option.

on which they train a regression model. These models can, depending on the training data, become exceptionally good classifiers of the training set and therefore, at least ideally, solve exactly the problem they were trained on (cf. AlphaGo, StarCraft-II). In contrast, we humans, at least typically, do not solve such tasks in isolation[5], but rather they naturally emerge throughout our life. We humans have a sense of what feels natural, that essentially represents consistency with prior experience[6]. The Bayesian brain hypothesis [Doya et al., 2007] describes how the brain forms a (stochastic) model as a prior for the world and updates it based on the incorporation of new evidence generated by the sensory organs. In particular, it explains how we deal so well with ambiguous input such as language. This also gives rise to an enormous amount of redundancy within human-to-human communication: You are able to raed this snetence, even thgouh mnay of the wdros are mieepsllsd and yuor biarn has to dcdiee which wrod was maent to be wtreitn by the author[7]. This biological implementation of error correction allows for a nuanced level of coherence.

In order to reproduce the brain's ability we need to answer the question: How does the brain implement its computation mechanistically? In the spirit of

> R. Feynman: What I cannot create, I do not understand.

we would ultimately like to be able to construct a (physical) model that reproduces the *essential* features of the brain's computation. This is both in order to prove that we understand it but also in order to have a model to test hypothesis on, without potentially harming humans and thereby alleviating some ethical concerns. As these models are an awkward fit for traditional von-Neumann hardware, this results in expensive simulations. The field of neuromorphic engineering – going back to Caver Mead [Mead, 1990] – tries to implement a physical model that mimics the brain. The hallmark of the neural communication is its spiking nature. While we assume at least some computational relevance, i.e., whether spikes are incidental or fundamental, this is still an openly debated question in neuroscience. On the other hand there seems to be a wide consensus of potential benefits from energetic and temporal reasons. While the term "neuromorphic" system expanded its meaning over time, we use it only to refer to spike-based neural network implementations. Without knowing the complete connectome we are in need for a compute model that incorporates the information of the spike times and the sparse connectivity and activity of the brain.

One spiking model of Bayesian computing is based on networks of Leaky-integrate and fire (LIF) neurons. Mimicking the diverse stimulus of cortical neurons by high-frequency Poisson noise sources, Petrovici et al. have shown that the dynamics of such networks can be linked to binary Boltzmann distributions [Petrovici et al., 2016]. It has

[5] There is an argument that academic tests are such isolated tasks.

[6] This goes so far that in case of inconsistent sensory input the natural reaction of the human body is to vomit as it assumes that we took something unbecoming. For example, this requires a bird flight simulator to blow air into your face rather than your neck [Pinotti, 2020].

[7] In fact this sentence is harder than the original psychological experiment. It simply shuffles the letters and as such has a notable effect on the understandability. If we had only exchange letters with the same height structure the effort would be significantly reduced.

been shown that this kind of Bayesian model can implement both generative as well as discriminative networks [Petrovici et al., 2016, Leng et al., 2018, Dold et al., 2019, Kungl et al., 2019]. However, these results have also shown that the description as Boltzmann machines is limited.

In this thesis we further the understanding of this link between a purely statistical description on the one hand and the complex LIF dynamics on the other hand. We start in Chapter 2 by introducing the necessary background information both regarding models for biological neurons as well as the theory of probabilistic computing. We will end this first chapter with the introduction of the LIF sampling framework by [Petrovici et al., 2016] before investigating the single neuron behavior in more detail in Chapter 3. Here, we will first take a look at the complications that the LIF dynamics introduce before developing the more detailed LMM. In Chapter 4 we then turn to the dynamics of ensembles of such neurons. We begin by introducing a dynamical notion of temperature. Thus equipped, we establish a link between the classical Ising model for ferromagnetism and Boltzmann machines, before investigating the differences between the phase diagrams for Ising-like networks of the different neuron models we introduced so far. Finally, in Chapter 5, we present two functional applications of this Bayesian computing framework on neuromorphic hardware, before ending the monograph in Chapter 6 with a discussion and outlook for future work.

2. Background: Biology & Probabilistic computing

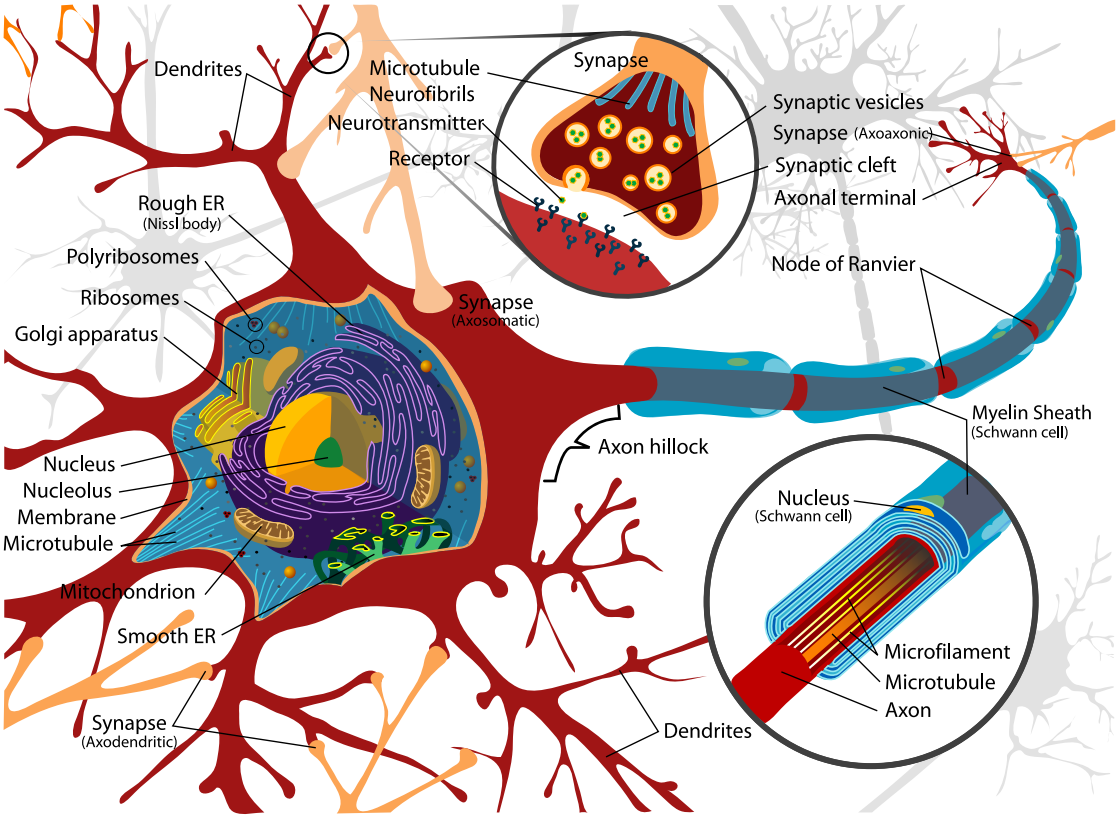

Figure 2.1.: **A biological neuron:** Stylistic representation of a biological neuron cell. Besides the common cell features like the nucleus, mitochondria, (poly)ribosomes, Golgi apparatus and the endoplasmatic reticulum (ER), neurons have characteristic features like the axon (myelin sheath surrounded "cable" going to the background) and the dendritic trees (tree like structures in red on the left). Together with the main cell body, also called soma, the last two form the computational resources of the neuron. The axon acts as the output and connects via synapses to the dendritic trees or directly the soma of other neurons. Figure taken from [Wikipedia, contributors, 2020].

2.1. Biological neurons in computational neuroscience

Before we discuss the interpretative models [Dayan and Abbott, 2001] used to describe biological neurons, let's briefly discuss biological neurons themselves. The human brain has approximately 50×10^9 to 100×10^9 neurons [von Bartheld et al., 2016] and in total around 15×10^{14} synaptic connections between them [Pakkenberg et al., 2003]. Later we will model these neural network as a series of point-neurons with instantaneous connections between them, however, we need to remember that this is a drastic simplification.

Biological neurons are extended cells, a schematic representation is shown in Fig. 2.1. We will ignore all of the inner cell structure, the cell organelles like the nucleus where

the DNA is located, the mitochondria which are the local power stations, the ribosomes and the endoplasmatic reticulum (ER) that synthesize new proteins and other molecules for both inside and outside the neuron and the Golgi apparatus which packs proteins for the outside into vesicles and collect received vesicles. The microtubules are the main skeleton of the cell and give it its characteristic shape [Welsch and Deller, 2016]. These are all common features of biological cells and, since they are not specific to neurons, we will not return to discuss them.

For neurons in particular, most of the information processing happens via the (chemical and electrical) potential difference across the cell membrane (red surface in Fig. 2.1). A neuron has three distinct parts: The dendritic trees, the main cell body (soma) and the axon.

The dendritic trees act as the collection site for input to the neuron. Here synaptic input is collected and injected into the soma. The typical neuron in the cortex is estimated to receive input from 1×10^4 other neurons [Pakkenberg et al., 2003]. This connectivity number or fan-in depends on the brain area and should be taken as an order of magnitude approximation, rather than a precise measurement.

The main cell body, the soma, acts as a capacitance and integrates the inputs that are injected from the dendritic trees. This input is implemented in form of different ion-channels spanning through the cell membrane. The main ion types are sodium Na^+, potassium K^+, calcium Ca^{2+} and chloride Cl^-. There are two large classes of pathways through which the cell may exchange ions with its surrounding: The passive ion-channels and the active ion-pumps. The passive channels are driven by the chemical and/or electrical imbalance and allow specific ion types to diffusively pass through the membrane. The pumps can move ions against the chemical/electrical gradient under usage of energy (ATP). The resting potential of neurons, i.e., the electric potential at which the neuron is in equilibrium with the intra-cellar medium, largely corresponds to the diffusion potential of the K^+-ions at about $-70\,\mathrm{mV}$ with respect to the intercellular medium.

The ion channels have a peculiar eigendynamik, namely once the membrane potential exceeds a threshold value[1], runaway dynamics of the voltage-gated ion channels are triggered. After stimulus-triggered injections of Na^+ or Ca^{2+} have increased the membrane voltage beyond the threshold ($\approx -55\,\mathrm{mV}$), the increased potential blocks the standard K^+ channels which prevents an outflux of K^+ ions and hence increases the potential. For a sufficiently high voltage voltage-gated Na^+ channels get triggered and enter a positive feedback loop which triggers the action potential (AP) or spike (cf. Fig. 2.2a). This spike, over the timescale of less than 1 ms, pulls the membrane voltage upwards above 0 mV and depolarizes the membrane. As the sodium channels close over time the slower potassium channels open. This allows the cell membrane to restore the electrical equilibrium again by releasing the excess K^+ ions (repolarization), which takes again less than 1 ms. Due to voltage-gated Ca^{2+}-channels there is now a Ca^{2+} excess in the neuron cell which triggers additional calcium-gated potassium channels which further reduce the membrane potential below its normal resting value (hyperpolarization). At

[1] In practice, the crossing of the threshold also needs to be sharp enough to trigger an action potential.

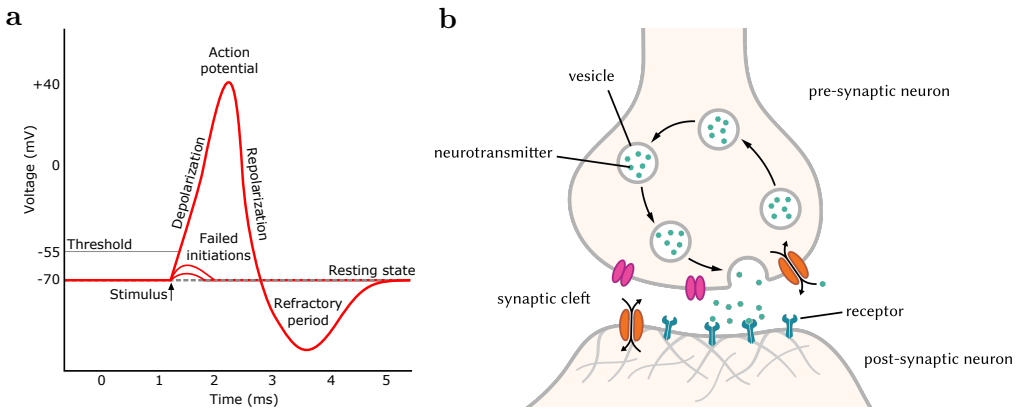

Figure 2.2.: **Spike-based communication between biological neurons: a** The action potential (AP), also called spike, is triggered for sufficient excitatory input, such that the membrane potential crosses a threshold. This triggers the characteristic eigendynamik, resulting in the prototypical voltage spike (depolarization), followed by an hyperpolarization period in which the neuron's activity is suppressed (refractory period). This voltage spike travels along the axon (see Fig. 2.1) and triggers synaptic connections towards the dendritic trees of the post-synaptic neurons. Figure adapted from [Chris73, 2007]. **b** Sketch of a chemical synapse. The voltage increase on the axon side triggers the release of neurotransmitters into the synaptic cleft. On the post-synaptic side these bind to receptor proteins and open ion-channels, thereby eliciting a post-synaptic potential (PSP). Figure adapted from [Splettstoesser, 2015].

the same time the Na^+ channels, responsible for the runaway dynamics from before, are suppressed and the neuron therefore unable to elicit a new spike. This phase is called the refractory period and is typically a few ms long. Hodgkin and Huxley first introduced a functional model that accurately represents this prototypical action potential generation [Hodgkin and Huxley, 1952]. In particular, the form of the voltage spike is independent of the input[2] and does not contain additional information beyond its spike time [Hodgkin and Huxley, 1952].

So far we have discussed the voltage evolution at the soma. Its membrane spike transfers to the axon at the axon hillock and propagates along it. For our purposes here the signal forms a characteristic action potential that does not (significantly) decay. The axon splits up into multiple axon terminal which themselves connect to the dendritic trees or directly the soma of other neurons [Welsch and Deller, 2016]. These connections are called synapses. The neuron that triggered the spike along the axon is called the pre-synaptic neuron. The neuron which receives the input is called the post-synaptic neuron. Whenever a voltage spike from the pre-synaptic neuron reaches the synapse it

[2]But not necessarily of the neuron type [Bean, 2007].

triggers a release of neurotransmitters into the synaptic cleft (cf. Fig. 2.2b).

There are two classes of neurons, excitatory and inhibitory ones. On the post-synaptic side of the synaptic cleft these bind to ion channels. This binding results in an effective post-synaptic current (PSC) to or from the cell, which in turn triggers a shift in the membrane potential of the post-synaptic neuron. For excitatory connections this shift is positive, i.e., towards the threshold value and the resulting potential change is an excitatory post-synaptic potential (EPSP). For inhibitory connections this shift is negative, i.e., away from the threshold and the resulting potential change is an inhibitory post-synaptic potential (IPSP). The soma has a finite capacitance and thereby acts as a low-pass filter on the synaptic input, be it mediated via the dendritic tree or directly attached to the soma.

A biological neuron has either exclusively inhibitory or exclusively excitatory effect on all its post-synaptic partners. This is called Dale's law [Dale, 1953]. Even though exceptions have been reported [Sulzer and Rayport, 2000], it is widely accepted that models should work with neurons that are either purely excitatory or purely inhibitory [Dayan and Abbott, 2001]. We will violate this "law" in most of our experiments. The assumption is that in larger networks we could find solutions of equivalent performance which do obey Dale's law. Our description of biological neurons is by no means complete and the interested reader is referred to e.g. [Kandel et al., 2000]. For us this high-level understanding of neurons as point objects which integrate synaptic input and emit spikes is sufficient.

2.1.1. Leaky-integrate and fire (LIF) neuron model

If we accept that neurons are essentially capacitors and their main method of communication are spikes then the arguably simplest model for them is the Leaky-integrate and fire (LIF) model [Brunel and Van Rossum, 2007]. It is used extensively in the computational neuroscience community, both in neuromorphic hardware platforms (see [Thakur et al., 2018] for a review) and simulations of spiking networks [Tavanaei et al., 2019]. In particular, the BrainScaleS neuromorphic platforms can implement LIF neurons (cf. Chapter 5). While it is the simplest model it is by far not the only one, the interested reader is referred to [Gerstner and Kistler, 2002b] for an extensive collection of spiking neuron models.

The LIF model is arguably what happens when you leave electrical engineers alone to build a neuron: The soma is represented by a capacitor C_m, which is connected to a leakage potential V_l via a leak conductance g_l. The dendritic trees are represented by synaptic input circuits (see Fig. 2.3 for a conductance-based model) implementing a time-dependent input current $I^\mathrm{syn}(t)$. The form of I^syn is discussed below (cf. Eqs. (2.8) and (2.12)). The differential equation governing the evolution of the membrane potential u (voltage over C_m) can then be constructed via Kirchhoff's rules

$$C_m \frac{du}{dt} = g_\mathrm{l} \left(V_\mathrm{l} - u\right) + I^\mathrm{syn}(t) \tag{2.1}$$

As the form of the action potential or spike does not contain any information, the spike mechanism is implemented via a simple threshold comparison. Whenever the neuron

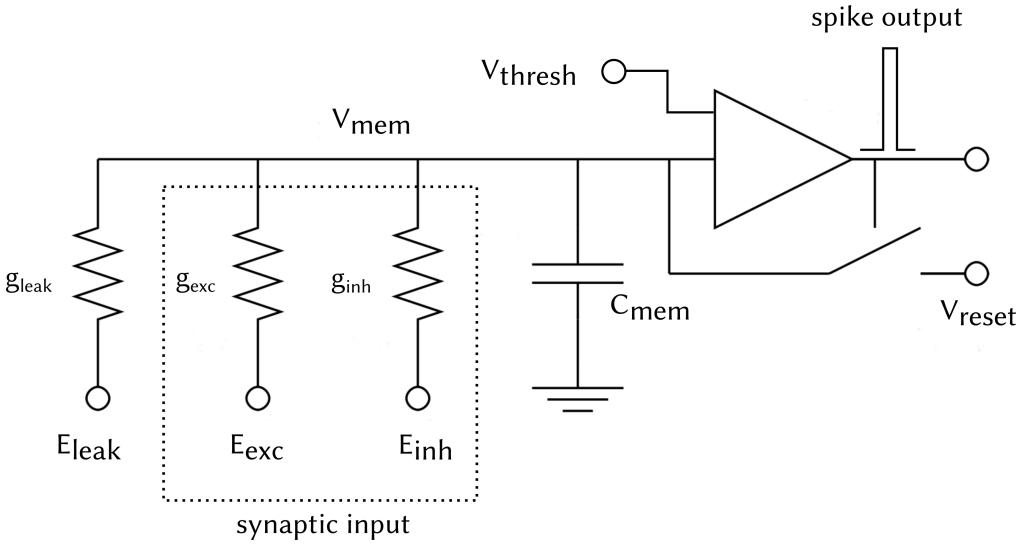

Figure 2.3.: **LIF equivalent circuit:** Modeling the neuron as a capacitance C_m the membrane potential u is generated by the conductances to the leak V_l, excitatory reversal $V_\mathrm{exc}^\mathrm{rev}$ and inhibitory reversal $V_\mathrm{inh}^\mathrm{rev}$ potentials. The spiking mechanism is modeled by the comparator with the threshold V_thresh voltage. If $u > V_\mathrm{thresh}$ a digital spike is generated and the membrane capacitance short circuited to the reset potential V_reset for the refractory period τ_ref. For a current-based LIF neuron the synaptic input is modeled by a time dependent current source, rather than the shown conductances (in reality they would be time dependent). Figure taken from [Stöckel, 2015].

crosses the threshold value V_thresh its membrane potential u is short-circuited to a reset value V_reset for a refractory period τ_ref:

$$u = V_\mathrm{reset} \quad \text{for} \quad t \in (t_\mathrm{spk}, t_\mathrm{spk} + \tau_\mathrm{ref}) \quad \text{if} \quad u(t_\mathrm{spk}) = V_\mathrm{thresh} \tag{2.2}$$

Together with Eq. (2.1) this generates a list of output spike times for each neuron, its so-called spike-train:

$$S(t) = \sum_{t_\mathrm{spk} \text{ for neuron}} \delta\left(t - t_\mathrm{spk}\right) \tag{2.3}$$

This spike train contains all the information post-synaptic neurons can use and therefore this is all we need to communicate within the model.

Current-based synaptic connections

Spike-based input is modeled via kernels $\kappa(t)$. In the current-based (CUBA) model this kernel directly represents the post-synaptic current (PSC), which gives the input current

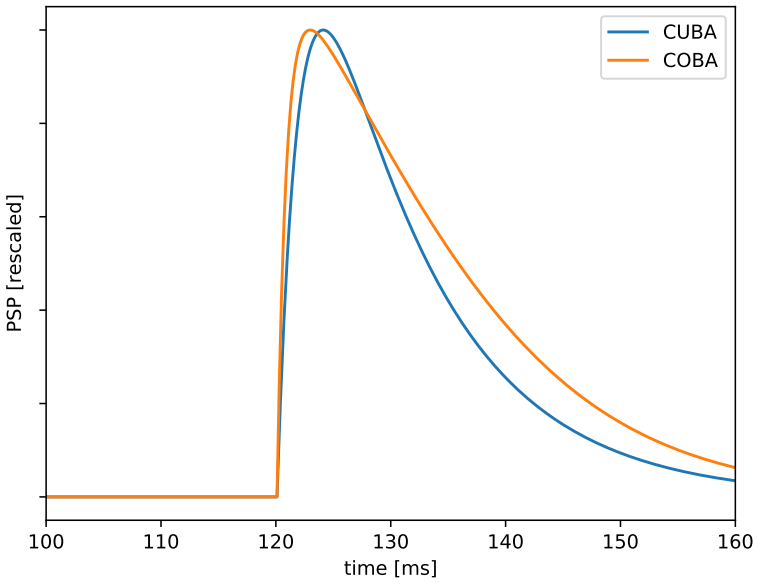

Figure 2.4.: **Isolated PSP:** Normalized time course of the PSP for a single spike input to a neuron with current (blue) or conductance (orange) synaptic input with $\tau_\mathrm{m} = 2\,\mathrm{ms}$ and $\tau_\mathrm{syn} = 10\,\mathrm{ms}$. The conductance-based PSP rises faster due to the input-induced conductance reducing the effective membrane time constant τ_m, which results in the decaying slope of the COBA PSP. Simulation parameters can be found in Appendix B.1.1.

from one particular pre-synaptic partner i as

$$I_i^\mathrm{syn}(t) = \sum_{t_\mathrm{spk}^i} W_i \kappa(t - t_\mathrm{spk}^i) \tag{2.4}$$

with the strength of the synaptic connection W_i and the kernel $\kappa(t)$. Due to causality we require $\kappa(t < 0) = 0$. The sum goes over all spike times of the pre-synaptic neuron i. In the simplest model, which is used throughout this thesis, the currents of multiple synapses add up linearly, such that the total synaptic input becomes:

$$I^\mathrm{syn}(t) = \sum_{\text{syn i}} I_i^\mathrm{syn}(t) = \sum_{\text{syn i}} \sum_{t_\mathrm{spk}^i} W_i \kappa(t - t_\mathrm{spk}). \tag{2.5}$$

We mainly use an exponentially decaying synaptic kernel

$$\kappa(t) = \Theta(t) \exp\left(-\frac{t}{\tau_\mathrm{syn}}\right), \tag{2.6}$$

which is one of the more popular choices, with $\Theta(t)$ being the Heaviside step function. Other popular choices include the delta peak kernel $\kappa(t) = \delta(t)$ and alpha-shaped kernel

$\kappa(t) = \Theta(t)\frac{t}{\tau_{\text{syn}}}\exp\left(-\frac{t}{\tau_{\text{syn}}}\right)$. We choose the exponential decay as it simulates an immediate release of available neurotransmitters (δ-function) on passing of the spike, with exponential decay of the efficacy of the ion channels afterwards on the time scale of the synaptic time constant τ_{syn}:

$$\frac{dI^{\text{syn}}}{dt} = -\frac{I^{\text{syn}}}{\tau_{\text{syn}}} + W\delta\left(t - t_{\text{spk}}\right). \tag{2.7}$$

It lends itself to a rather simple implementation in electronic circuits which we will be using in Chapter 5. We can find the instantaneous input current $I_i^{\text{syn}}(t)$ from the synapse i by integrating Eq. (2.7) for a given spike train S_i, resulting in:

$$I_i^{\text{syn}} = \sum_{t_{\text{spk}}^i \in S_i} W_i \Theta(t - t_{\text{spk}}^i) \exp\left(\frac{t - t_{\text{spk}}^i}{\tau_{\text{syn}}}\right) \tag{2.8}$$

Current-based synapses are used to represent synapses that are located far away from the soma. The synaptic interaction triggers additional pathways, i.e., conductances, far away from the soma and therefore does not contribute to the leak conductance of the main body. As such a current source is an appropriate model.

The resulting membrane time course $u(t)$ can be calculated analytically [Petrovici, 2015] to be:

$$u(t) = V_{\text{l}} + \frac{I_{\text{ext}}}{g_{\text{l}}} + \sum_{\text{syn}}\sum_{k\,\text{spk}}\sum_{s} \frac{\tau_{\text{syn}}^k W_k}{g_{\text{l}}\left(\tau_{\text{syn}}^k - \tau_{\text{m}}\right)} \Theta\left(t - t_{\text{spk}}\right)\left[\exp\left(-\frac{t - t_{\text{spk}}}{\tau_{\text{syn}}^k}\right) - \exp\left(-\frac{t - t_{\text{spk}}}{\tau_{\text{m}}}\right)\right] \tag{2.9}$$

with the membrane time constant

$$\tau_{\text{m}} = \frac{C_m}{g_{\text{l}}}. \tag{2.10}$$

The difference of Eq. (2.9) to the baseline level $V_{\text{l}} + \frac{I_{\text{ext}}}{g_{\text{l}}}$ is the sum of all PSP each of which follows a difference of exponentials (cf. blue curve in Fig. 2.4).

Conductance-based synaptic connections

The other large class of synapse models is conductance-based. Here the effect of incoming synaptic activity mediates the (ion-specific) permeability between the cell and its surrounding. A spike elicits a PSC (this time representing the post-synaptic *conductance*) that, depending on the ion type, increases the neuron's coupling to an excitatory $V_{\text{exc}}^{\text{rev}}$ or an inhibitory reversal potential $V_{\text{inh}}^{\text{rev}}$. Conductance-based neurons are used to model synaptic connections at or close to the soma, where they also contribute to the effective leak conductance g_{l} and hence influence the membrane time constant τ_{m}.

The PSC obeys a similar ODE than the synaptic current Eq. (2.7):

$$\frac{dg}{dt} = -\frac{g}{\tau_{\text{syn}}} + W\delta\left(t - t_{\text{spk}}\right) \tag{2.11}$$

Note that W now has the unit of a conductance (siemens) rather than a current (ampere). Integrating this over an input spike train S gives the time course of the conductance of a particular synaptic connection i:

$$g_i = \sum_{t^i_{\text{spk}} \in S_i} W_i \Theta(t - t^i_{\text{spk}}) \exp\left(\frac{t - t^i_{\text{spk}}}{\tau_{\text{syn}}}\right). \tag{2.12}$$

The total synaptic input is again the sum over the current generated by all incoming synapses, with all excitatory connections contributing to the coupling to $V^{\text{rev}}_{\text{exc}}$ and all inhibitory ones to $V^{\text{rev}}_{\text{inh}}$:

$$I^{\text{syn}}(t) = \sum_{\text{exc syn } i} g_i(t)\left(V^{\text{rev}}_{\text{exc}} - u(t)\right) + \sum_{\text{inh syn } j} g_j(t)\left(V^{\text{rev}}_{\text{inh}} - u(t)\right) \tag{2.13}$$

The resulting membrane time course $u(t)$, at least in general, cannot be calculated analytically as the effective membrane time constant is now also time dependent. We will later use a high-conductance state (HCS) [Kumar et al., 2008] generated by high-frequency Poisson input such that we can make the assumption:

$$g_{\text{syn,tot}}(t) = \sum_{\text{exc syn } i} g_i(t) + \sum_{\text{inh syn } j} g_j(t) \approx \langle g_{\text{tot}} \rangle \tag{2.14}$$

yielding a quasi-constant effective membrane time constant

$$\tau^{\text{eff}} = \frac{C_m}{g_{\text{l}} + g_{\text{syn,tot}}} \approx \left\langle \tau^{\text{eff}} \right\rangle \tag{2.15}$$

and hence we can calculate the PSP of a single *additional* synaptic input to be [Petrovici, 2015]

$$u(t) - \langle u \rangle = \frac{\tau_{\text{syn}} \left\langle \tau^{\text{eff}} \right\rangle W \left(V^{\text{rev}} - \langle u \rangle\right)}{C_m \left(\tau_{\text{syn}} - \langle \tau^{\text{eff}} \rangle\right)} \Theta\left(t - t_{\text{spk}}\right) \left[\exp\left(-\frac{t - t_{\text{spk}}}{\tau_{\text{syn}}}\right) - \exp\left(-\frac{t - t_{\text{spk}}}{\langle \tau^{\text{eff}} \rangle}\right)\right] \tag{2.16}$$

where the reversal potential V^{rev} depends on the type of the synapse. The PSP is now the difference to the average membrane potential $\bar{u} = \langle u \rangle$ evaluated in an ensemble sense. I.e., the mean of the membrane potential of a collection of neurons with identical parameters and subject to the same noise configuration. The value of $\bar{u}$ depends on both the neuron parameters, and the noise configuration (cf. Section 4.1).

Without the simplifying assumptions the PSP shape is more complicated, see orange curve in Fig. 2.4 for an isolated PSP. It rises faster due to the decreased effective membrane time constant[3] τ_{m}. The slower decay is due to the voltage-dependent current strength, which increases for a fixed $g(t)$ for a decreased membrane potential u.

[3] The rising flank of the PSP corresponds to the lower of the two relevant time constants τ_{m} and τ_{syn}, the falling flank to the higher. We will only use configurations in which $\tau_{\text{m}} \ll \tau_{\text{syn}}$.

Effective formulation

We can make the dynamical difference between COBA and CUBA neurons more obvious when we rewrite Eq. (2.1) as a low pass filter with a time constant τ and effective target potential $u^{\text{eff}}(t)$:

$$\frac{du}{dt}\tau = u^{\text{eff}} - u \tag{2.17}$$

For current-based neurons (cf. Eq. (2.9)) we can read off the parameters directly as

$$\tau = \tau_{\text{m}} = \frac{C_m}{g_{\text{l}}} \tag{2.18}$$

and

$$u^{\text{eff}}(t) = \frac{I^{\text{syn}}(t)}{g_{\text{l}}} + V_{\text{l}}. \tag{2.19}$$

Hence, a CUBA neuron acts simply as a low-pass filter on its synaptic input with an attached threshold mechanism for spike generation.

The COBA dynamics are slightly more complicated. The synaptic input also adds conductance and therefore the membrane time constant τ_{m} becomes time-dependent

$$\tau_{\text{m}}(t) = \frac{C_m}{g_{\text{l}} + g^{\text{syn}}_{\text{exc}}(t) + g^{\text{syn}}_{\text{inh}}(t)} \tag{2.20}$$

where $g^{\text{syn}}_{\text{exc}}(t)$ and $g^{\text{syn}}_{\text{inh}}(t)$ are the sum of all excitatory and all inhibitory synaptic conductances respectively (cf. Eq. (2.15)). The target potential can then be written as

$$u^{\text{eff}}(t) = \frac{g_{\text{l}}V_{\text{l}} + g^{\text{syn}}_{\text{exc}}(t)V^{\text{rev}}_{\text{exc}} + g^{\text{syn}}_{\text{inh}}(t)V^{\text{rev}}_{\text{inh}}}{g_{\text{l}} + g^{\text{syn}}_{\text{exc}}(t) + g^{\text{syn}}_{\text{inh}}(t)}. \tag{2.21}$$

As such we can see that the membrane potential u of the COBA model is bounded by

$$\min(V_{\text{l}}, V^{\text{rev}}_{\text{inh}}) < u < \max(V_{\text{l}}, V^{\text{rev}}_{\text{exc}}). \tag{2.22}$$

Typically we choose $V^{\text{rev}}_{\text{inh}} < V_{\text{l}} < V^{\text{rev}}_{\text{exc}}$. These bounds exist completely independent of the input, unlike in the CUBA model which can, at least in principle, reach arbitrarily high (low) membrane voltage values. From a perspective of biological plausibility the conductance-based model has one less potential pitfall.

However, this boundedness also affects the amplitude of single PSPs. These are now – rather strongly – history dependent, as it scales with the difference between the reversal potential and the current membrane voltage $V^{\text{rev}} - u$. At first glance this implies a strong variance between the information transmitted by different PSPs through their dynamical effect. However, the information of an input is actually encoded at the level of the synaptic input. Here, both the current-based as well as the conductance-based case simply act with a δ jump in the respective state variables $g(t)$ and $I^{\text{syn}}(t)$. It is only the effect on the membrane potential that differs. In fact, we do not even care about the precise membrane potential evolution as only the times of the threshold crossings (spike times) are relevant for the dynamics of post-synaptic neurons.

This means – at least for the particular parameter choices we will later make – that all information about the input history is stored in the state of the synaptic input, which holds for both the current-based as well as the conductance-based models. The reset after a spike does not erase information. It is the dynamics of u that are short-circuited and thereby artificially suppressed. At the level of u^{eff}, which is completely defined by the state of the synaptic input, the dynamics continue uninhibited. The neuron is therefore still able to integrate additional input, even though it can only act on said input after the end of the refractory period with some additional delay due to the finite distance between $V_{\text{thresh}} - V_{\text{reset}} > 0$. This is a consequence of our choice of $\tau_{\text{m}} \ll \tau_{\text{syn}} \approx \tau_{\text{ref}}$, which we will motivate in Section 2.2.4.

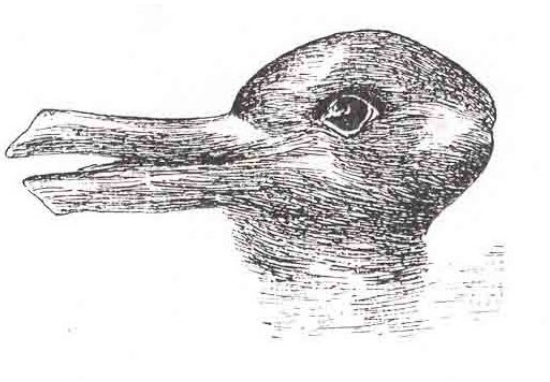

Figure 2.5: **Ambiguous input:** The figure is the prototypical example of an ambiguous image. It could both be a duck looking to the left or a rabbit looking to the right. The 2D still image is not sufficient to allow for a unique identification. Image taken from [Wikimedia, 2016a].

2.2. Probabilistic computing

Before we can introduce the particular implementation using spiking neurons that we are interested in, we need to introduce a general problem that we, as humans, are faced with every day: Probabilistic computing

While this statement may sound ambitious or contentious at first glance it really is not once we understand how we interact with the world. The world is an enormously complex thing that we *cannot* have complete information about. Here we do not mean to make some fundamental statement like the Heisenberg uncertainty relations, but rather a practical one. The input we receive through our senses (vision, hearing, touch, smell and taste) is extremely limited. One well-known example is an illusionary image (e.g. Fig. 2.5), where a two dimensional photograph of an object simply does not contain enough information about the object to allow a unique identification.

At this point we can either give up, or we can find a way to live with and incorporate this uncertainty. While we may not be able to know whether the image was taken of a rabbit or a duck (or whether the artist wanted to draw a rabbit or a duck, if one prefers the fictional discussion), we do know that it is definitely not a car or an airplane. As such we do have information on which we can act.

The Bayesian brain hypothesis [Doya et al., 2007] promotes the following idea: The human brain constantly tries to build a *model* of the world that explains all the input it receives. As a consequence we need to infer things about the world that we do not explicitly know, e.g. it is likely that someone tapped on the light switch if all I know is that the light went out. In most situations in the real world there is *one* exceedingly good explanation. Nevertheless, it is straightforward to construct edge cases where two options are similarly likely. In Fig. 2.5 one such example is shown. The image is compatible with both a rabbit looking to the right and slightly upwards or a duck looking to the left.

Most or all humans switch between all or some compatible interpretations. I.e. I will see, at any given point in time, either a duck or a rabbit in Fig. 2.5. But I will always know it must be either the one or the other and not both at once. In a way our brain seems to *sample* from the evidence-compatible interpretations and only ever offer a single explanation. This is a strong evidence that the brain implements probabilistic

computation in a sample-based fashion [Sundareswara and Schrater, 2008, Gershman et al., 2009].

In the following section we will describe, in a mechanistic way, how we can model probability distributions over distinct event outcomes and where – if not fundamental, then at least practical – problems of this kind of descriptions lie. After the introduction to the traditional handling of Boltzmann distributions over binary random variables in Section 2.2.1, we describe the Gibbs sampling method and how to evaluate the quality of a generated sample in Section 2.2.2. We then move to a more biologically inspired sampling implementation by recapping the neuron model from [Buesing et al., 2011] in Section 2.2.3 and finally introducing the Leaky-integrate and fire (LIF) sampling framework from [Petrovici et al., 2016] in Section 2.2.4. The latter is the main model we will be investigating throughout this thesis.

2.2.1. Boltzmann distributions over binary random variables

Definition 2.2.1. *Probability distribution*
A probability distribution p is a mapping from the discrete set of events X to real numbers $p(x) \in [0, 1]$ such that the $p(x)$ corresponds to the probability of finding the element $x \in X$ in a random draw.

In the introductory example the possible events of the probability distribution for the example shown in Fig. 2.5 would be

- "The image represents a rabbit."
- "The image represents a duck."

Without any further information we would probably use a flat prior and assign to both states equal probability:

$$p(\text{duck}) = 0.5\ , \quad p(\text{rabbit}) = 0.5. \tag{2.23}$$

Here we assumed already a number of things: We are absolutely sure that it has to be either a duck or rabbit and not anything else (completeness of the state space) and that there is no bias between the two outcomes (flat prior). The former is inherent to the definition of probability distributions, i.e., we assume that we will *always* have to get an answer. The mathematical formulation of this is that the sum of all probabilities adds up to unity:

Corollary 1. *Property*
The sum over all probabilities of all events $x \in X$ must add up to 1,

$$1 = \sum_{x \in X} p(x). \tag{2.24}$$

We will not discuss the problem of choosing priors further, as we deal with the *implementation* and not the *interpretation* or *generation* of probability distributions. For us

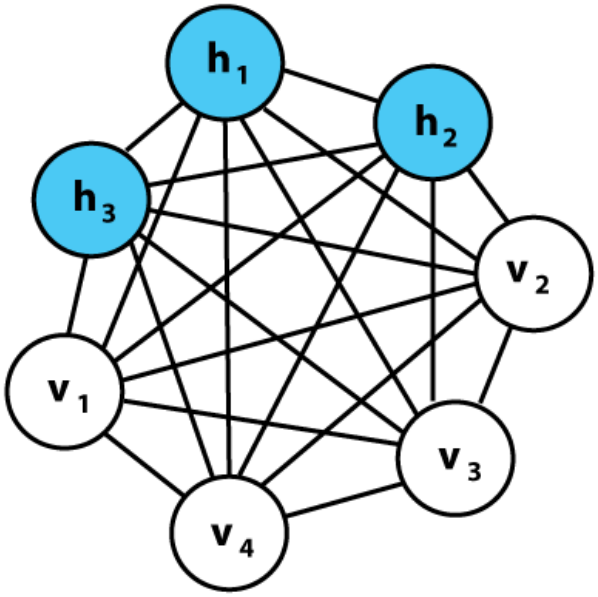

Figure 2.6.: **Neural Network:** Stylistic representation of a neural network. Here neurons are objects with a binary state $z \in \{0, 1\}$. They form a fully-connected bi-directional graph with connection weights w_{ij} between neuron i and neuron j without self-connections. Later we will distinguish between visible v and hidden h neurons and restrict the connectivity matrix to only $v - h$ connections to form a restricted Boltzmann machine (RBM). Figure taken form [Wikimedia, 2020].

it suffices to give the warning, that none of this is easy, even though choosing priors is a non-avoidable complication of a Bayesian world view [Bailer-Jones, 2017].

In practice, we will be dealing with probability distributions over n binary random variables z. In other words, each state x will be represented by the state of a n-unit network where each unit is either in state $z = 0$ or state $z = 1$. The state space is therefore $X = \{0, 1\}^n$ and each event x corresponds to a particular vector $\vec{z} \in \{0, 1\}^n$. The arguably simplest probability distribution over this kind of system is a so-called binary Boltzmann distribution which allows only for non-trivial correlations at the one- and two-point level (cf. Eq. (2.26)).

Definition 2.2.2. *Binary Boltzmann distribution*

A binary Boltzmann distribution is a probability distribution over a set $\vec{z}$ of n binary random variables $z \in \{0, 1\}$ defined by

$$p(\vec{z}) = \frac{1}{Z} \exp\left(-E(\vec{z})\right), \tag{2.25}$$

with the partition sum $Z = \sum_{\{\vec{y} \in \Omega\}} \exp\left(-E(\vec{y})\right)$. The partition sum over the complete state space $\Omega = \{0, 1\}^n$ ensures normalization.

We will only use Boltzmann distributions, whose energy function

$$E(\vec{z}) = \frac{1}{2}\vec{z}^T w \vec{z} + \vec{b}^T \vec{z} \tag{2.26}$$

only depends on the state of single neurons (via the bias $\vec{b} \in \mathbb{R}^n$) and the combination of two neurons (via the weight $w \in \mathbb{R}^{n\times n}$). We use weight matrices w that are symmetric and have a zero diagonal. While this does not change the value $E(\vec{z})$ it does influence the sampling implementation (cf. Eq. (2.35) below).

In general, we are interested in expectation values over some variable or combinations thereof:

$$\langle f(\vec{z})\rangle_p = \sum_{\vec{z}\in\Omega} f(\vec{z})p(\vec{z}). \tag{2.27}$$

If we were for example to ask whether the object in Fig. 2.5 likes carrots we would end up with:

$$\langle \text{love for carrots}\rangle = \text{ a lot } \times p(\text{rabbit}) + \text{ a lot less } \times p(\text{duck}). \tag{2.28}$$

Going back to the definition of $p(\vec{z})$ we see that the normalization constant Z is calculated as a sum over the complete state space Ω. For even moderately large number of binary neurons, this becomes unfeasible. The available compute power nowadays allows for evaluations of up to $n = 40$, i.e., a billion terms, without too much of a problem. Beyond that the curse of dimensionality supersedes technological advantages[4].

It is the pure number of possible states that is the problem here, not the calculation of any single term of the sum. On the other hand most of these states will not have a significant chance of appearing at all. Taking these two things in combination the sampling approach is an intuitive approach. Monte-Carlo sampling [Metropolis and Ulam, 1949] relies on the fact that we can easily calculate the relative probabilities of two states $\vec{z_1}$ and $\vec{z_2}$,

$$\frac{p(\vec{z_1})}{p(\vec{z_2})} = \exp\left(-E(\vec{z_1}) + E(\vec{z_2})\right), \tag{2.29}$$

as in this calculation the normalization constant Z drops out completely.

We can therefore side step the one-time calculation of Z with its 2^n terms by proposing new samples and accepting them in such a way that the original probability distribution $p(\vec{z})$ is the fixed point of this update scheme.

2.2.2. Gibbs sampling and Kullback-Leibler divergence

The arguably simplest sampling scheme was introduced by [Geman and Geman, 1984] and is named after Josiah Willard Gibbs (†1903). It is a special case of the standard

[4] The increase in terms of the sum is the main reason why we consider $n \approx 50$ q-bits to be the minimum number above which quantum supremacy comes to effect, with recent work claiming to demonstrate said supremacy [Arute et al., 2019] and others counterclaiming that the case of $n = 53$ q-bits can still be simulated on a classical computer [Pednault et al., 2019]. Independent of the specific claim of demonstrated quantum supremacy, distributions of $n > 60$ neurons are already far beyond the brute force tractability of the largest supercomputers.

Metropolis-Hastings sampling algorithm [Metropolis et al., 1953]. It suggests to update each of the n binary units in turn to the state $z = 1$ while keeping the rest of the network fixed. This suggestion is then accepted with the conditional probability:

$$p(z_{t+1}^k = 1|z_{\setminus k}) = \frac{p(z_{t+1}^k = 1)}{p(z_{t+1}^k = 1) + p(z_{t+1}^k = 0)} \tag{2.30}$$

$$= \frac{1}{1 + \frac{p(z_{t+1}^k=0)}{p(z_{t+1}^k=1)}} \tag{2.31}$$

which only depends on the relative probabilities for the configurations $(z_1, \ldots, z_{k-1}, z_k = 1, z_{k+1}, \ldots, z_n)$ and $(z_1, \ldots, z_{k-1}, z_k = 0, z_{k+1}, \ldots, z_n)$. For these we use the shorthand notation of $z_{t+1}^k = 0$ and $z_{t+1}^k = 1$ respectively. This fraction can then be calculated from Eq. (2.25) with energy function Eq. (2.26) as the problematic normalization factor Z drops out:

$$\frac{p(z_{t+1}^k = 0)}{p(z_{t+1}^k = 1)} = \frac{\frac{1}{Z}\exp\left(-E(z_{t+1}^k = 0)\right)}{\frac{1}{Z}\exp\left(-E(z_{t+1}^k = 1)\right)} \tag{2.32}$$

$$= \frac{\exp\left[-\sum\limits_{i\neq k} z_i(b_i + \frac{1}{2}\sum\limits_{j\neq i} w_{ij}z_j)\right]}{\exp\left[-\sum\limits_{i\neq k} z_i(b_i + \frac{1}{2}\sum\limits_{j\neq i} w_{ij}z_j) - b_k - \frac{1}{2}\sum\limits_{j\neq k}(z_j w_{jk} + w_{kj}z_j)\right]} \tag{2.33}$$

$$= \exp\left[-b_k - \frac{1}{2}\sum_{j\neq k}(z_j w_{jk} + w_{kj}z_j)\right] \tag{2.34}$$

$$= \exp\left[-b_k - \sum_{j=0}^{n} w_{kj}z_j\right], \tag{2.35}$$

where we used the assumptions $w_{ij} = w_{ji}$ and $w_{ii} = 0$ in the last step. These assumptions do not matter at the level of the Boltzmann distribution itself, but they do matter at the level of the sampling dynamics[5]. Plugging Eq. (2.35) back into Eq. (2.31) gives the gain function of the Gibbs sampler:

$$p(z_{t+1}^k = 1) = \frac{1}{1 + \exp\left(-b_k - \sum_{i\neq k} w_{ik}z_i\right)} = \frac{1}{1 + \exp\left(-u_k\right)} = \sigma(u_k). \tag{2.36}$$

where we introduced the short hand

$$u_k = b_k + \sum_{i\neq k} w_{ki}z_i \tag{2.37}$$

at the second equality sign. [Buesing et al., 2011] call this the *neural computability condition* that the post-synaptic unit has to fulfill in order to work as a sampling unit.

[5]The assumed symmetry reflects the third Newtonian law at a statistical level. Mathematically it is always possible to write the sum as $\sum_{i<j}$ rather than $\frac{1}{2}\sum_{i,j}$.

Note 1. *This is a sufficient rather than a necessary condition:*

It corresponds to balanced influx and outflux of probability mass at the level of each state pair $\vec{y}, \vec{z}$ for the steady-state distribution p. However, in order for p to be a fixed point of the sampling mechanism it would suffice if the net influx to every state was 0. *In other words, it is only necessary that the sum of the flux from one state $\vec{z}$ to* all *other states $\{\vec{y}\}$ equals the flux from all other states $\{\vec{y} \neq \vec{z}\}$ to the state $\vec{z}$. This is a significantly weaker constraint than the balance at the level of each individual state pair $\vec{z}, \vec{y}$.*

For reasons that will become obvious in Section 2.2.4 we call $\sigma(u)$ the activation or gain function of the sampler. However, we can already now see the similarity between Eq. (2.37) and the instantaneous target potential $u^{\text{eff}}(t)$ from Section 2.1.1.

Performing this update scheme in sequence for all neurons results in a new sample from the target probability distribution $p(z)$. The pseudo-code implementation of Gibbs sampling looks like:

Algorithm 2.1: Gibbs sampling: A simple realization of Gibbs sampling. The selection of the next variable for updates can vary for different implementations.

Data: Vector $\vec{z}$ of length N, probability distribution $p(\vec{z})$ specified by w, b
Result: Chain of samples $\vec{z}$ distributed according to $p(\vec{z})$
start from given $\vec{z^{(0)}}$;
while *required number of samples not reached* **do**
  choose a $z_k^{(n)}$ from $\vec{z}^{(n)}$;
  calculate neural computability condition: $u_k = b_k + \sum_{i \neq k} w_{ki} z_i$;
  calculate conditional probability $p(z_k^{(n)} = 1 | \vec{z}_{\setminus i}) = \sigma(u_k)$;
  accept sample with $\sigma(u_k) > r \sim U[0, 1]$;
  obtain $\vec{z}^{(n+1)} = (z_1^{(n)}, \ldots, z_{k-1}^{(n)}, z_k^{(n+1)}, z_{k+1}^{(n)}, \ldots, z_N^{(n)})$
end

At this point we have reduced the problem of calculating expectation values with respect to $p(\vec{z})$ from a one-time calculation of 2^n terms (each with a sum over n^2 terms) for the partition sum Z to the calculation of n^2 terms per new sample[6]. One can prove that the sampled set converges towards the target distribution. However, it is not known in advance how many samples will be required before the approximation is *good enough*. We will discuss this further when we talk about how to implement a tempering scheme in LIF networks in Section 4.1. For even moderately large numbers of units the sampling-based alternative will be beneficial as it will be faster to calculate near arbitrary numbers of samples than to calculate Z once.

[6] n neurons have to be updated with the potential each being a sum over $n - 1$ terms.

Performance evaluation

Having now a methodology of generating samples and having already remarked on the lack of a priori knowledge of the necessary number of samples, the next question becomes: How does one evaluate the quality of the generated set? Finding a good measure is always also dependent on the particular problem, but one standard measure for the distance between two probability distributions p and q is the Kullback-Leibler divergence (DKL) [Kullback and Leibler, 1951]:

Definition 2.2.3. *Kullback-Leibler divergence (DKL)*
The DKL is defined as

$$\mathrm{DKL}\left(p||q\right) = \sum_x p(x) \log \frac{p(x)}{q(x)} \tag{2.38}$$

It measures the relative entropy difference from the distribution p to the distribution q. Where the entropy of a distribution is defined as:

Definition 2.2.4. *Entropy (information theory)*
The entropy of a probability distribution is defined as the expectation of the log probability

$$S(p) = -\sum_x p(x) \log p(x) \tag{2.39}$$

Each alphabet has a space-optimal encoding that uses on average $S(p)$ bits per character (with p being the average frequency of the characters in texts). Intuitively speaking this means: If one encodes an alphabet distributed according to p with the (space-)optimal encoding according to q this encoding wastes on average $\mathrm{DKL}\left(p||q\right)$ many bits per character.

Property 1. *Positivity and equality*
The DKL *is positive for all p and q over the same support. Proof:*

$$\begin{aligned}
\mathrm{DKL}\left(p||q\right) &= \sum_x p(x) \log \frac{p(x)}{q(x)} && (2.40)\\
&= -\sum_x p(x) \log \frac{q(x)}{p(x)} && (2.41)\\
&\overset{(*)}{\leq} -\sum_x p(x) \left(\frac{q(x)}{p(x)} - 1\right) && (2.42)\\
&= -\sum_x q(x) + \sum_x p(x) = 0 && (2.43)
\end{aligned}$$

where we use the well known inequality $\log a \leq a - 1$ *in* $(*)$*. Equality holds if and only if* $p(x) = q(x)\ \forall x$ *such that*

$$\log \frac{q(x)}{p(x)} = \frac{q(x)}{p(x)} - 1 \tag{2.44}$$

holds for all x.

There are two special cases we need to be aware of: In principle both p and q can assign events the probability 0, which is a problem for the logarithm in Eq. (2.38). As

$$\lim_{x \to 0} x \log x = 0 \tag{2.45}$$

only $q(x) = 0$ results in a divergence. This means that we want to measure the distance in the opposite direction as typically defined in the literature. For us p is the arbitrary target distribution which is externally supplied and we want to approximate it with the model (Boltzmann) distribution q. Since p is externally supplied we may not impose any restrictions onto it, in particular, it is okay to have degenerate states x such that $p(x) = 0$.

While there is no state with $q(x) = 0$ within a Boltzmann machine (BM), in practice it may happen, that we *also* do not sample every single state x. In this case we need to be careful of how to implement the DKL calculation. In general there are two ways:

1. add a small offset number of samples to all states
2. discard all states x that were not sampled at all

The latter option requires a renormalization of *both* probability distributions, otherwise the DKL could become negative. The former introduces a (hopefully small) bias towards a uniform distribution. In practice, these ad hoc workarounds serve as an illustration on the limit of the DKL as a useful measure: If the state space is sufficiently large – such that we cannot sample all states at least once – then a distance measure that is based on the correctness on the frequency of all states is most likely not going to be useful. For small scale demonstrations we will be using sufficient sample numbers to ensure that all states are sampled. For larger systems a more effective measure, like a classification error, will be chosen.

It can be shown that an ideal sampling framework implements an estimator for the true DKL that converges with $\frac{1}{N}$ with N being the number of independent samples [Cai et al., 2006, Paninski, 2003]. In Gibbs sampling the notion of an independent sample is reached after an update for each neuron, for the sampling methods that follow below this notion is no longer as simple (see Fig. 2.7b and Fig. 3.4b).

restricted Boltzmann machine (RBM) and wake-sleep training

So far we have made no topological restrictions to the network of the BM, i.e., we allowed for arbitrary connection matrices w_{ij} as long as they are symmetric (cf. Eq. (2.35)). The price for this is that the update of every neuron depends on the state of all other neurons. If we restrict the allowed connectivity to block off-diagonal matrices, we restrict us to the so-called RBM [Smolensky, 1986]. It is a layered network structure where each neuron *only* connects to neurons of *other* layers. This makes the update probabilities of neurons within the same layer independent of each other. It is this independence that allows us to a) parallelize the updates and b) eases the trainability of the system. The latter is a non-obvious statement, as a fully connected Boltzmann machine can always *also*

implement a restricted one. However, the reduced number of available parameters eases the formation of the distribution by reducing the number of local minima within the parameter space [Hinton, 2012].

The RBM formulation also introduces a clear separation of concerns: One layer is designated to be a representation of the received input from the world, this is typically called the visible layer (consisting of the visible neurons v), and the rest of the network (i.e., the other layers) are used to form and interpret this model[7], these are the so-called hidden neurons h. These layers differ on a very fundamental level. The visible layer is a representation of the world, that means it is externally defined and we may not impose restrictions on it. In biology, this would correspond to the external stimulus that our senses transfer to the brain and conditions our world model. This also imposes the coding scheme to be compatible with the sensory output. In practice, our visible layer will correspond to some image data set or a target probability distribution that we want to reconstruct. The other layers we are in principle free to choose, as they "only" represent the inner workings of our model. It is always possible to describe an RBM as a fully-connected BM where the state z is the concatenation of the states of the single layers v and h and the connection matrix only has block-off-diagonal entries[8].

When trying to produce a good generative network the task is to make a model distribution p match an arbitrary target distribution q. The model distribution p is formed by the network parameters w_{ij} and b_i (cf. Eq. (2.26)). We already introduced the DKL as a difference measure between two probability distributions p and q. While it is not a distance measure (it lacks the symmetry property) minimizing it still improves the match of p and q. Hence we can derive a learning scheme for arbitrary distributions by calculating the gradient of the DKL with respect to the network parameters w_{ij} and b_i.

After some tedious calculations (see Appendix A.4) we find that the learning rule for the weights takes the form:

$$\Delta w_{ij} \propto \langle v_i h_j \rangle_{q(v)p(h|v)} - \langle v_i h_j \rangle_{p(v,h)} \tag{2.46}$$

$$= \langle v_i h_j \rangle_{\text{data}} - \langle v_i h_j \rangle_{\text{model}} \tag{2.47}$$

$$\Delta b_i = \propto \langle z_i \rangle_{\text{data}} - \langle z_i \rangle_{\text{model}} \tag{2.48}$$

This is a standard Hebbian learning rule and known as the wake-sleep algorithm [Hinton et al., 1995, Gerstner and Kistler, 2002a]. Intuitively we subtract the correlations under the free model distribution p from the correlations under the desired target distribution q. In the latter case we acquire the hidden-visible correlations through the conditional probability $p(h|v)$ imposed by the model distribution p as the data distribution q, by construction, does not provide us with information about the hidden layer. Note that Eq. (2.48) requires accurate samples of the complete distribution $p(v, h)$ as

[7] This can also be implemented in a fully-connected network graph, cf. Fig. 2.6. In this case we designate a subset of the neurons to correspond to the external distribution.

[8] We silently dropped the vector sign when referring to the state z and mark the individual states z_i by the neuron index i from now on.

well as $q(v)p(h|v)$ and as such they are computationally intensive, at least for traditional Monte-Carlo methods.

Traditional performance optimizations such as Contrastive Divergence replace the complete distribution sample with a single point estimate and generates the appropriate averages over multiple training steps via smaller learning rates [Carreira-Perpinan, Miguel A and Hinton, 2005, Taylor and Hinton, 2009, Sutskever and Hinton, 2007]. Later (cf. Chapter 5) we will deal with neuromorphic systems, where the actual number of samples drawn is of limited importance as the run time is dominated by communication overhead and thereby number of configurations (weight updates). Therefore we would profit from generating both data and model terms from a single sampling run, but not so much from reducing the number of samples required. We can, in fact, infer the data term from the model sample set, and thereby reduce the required number of sampling runs by two:

$$\begin{aligned}
\Delta W_{ij} &= \langle z_i z_j \rangle_{\text{data}} - \langle z_i z_j \rangle_{\text{model}} && (2.49)\\
&= \langle v_i h_j \rangle_{q(v)p(h|v)} - \langle v_i h_j \rangle_{p(v)p(h|v)} && (2.50)\\
&= \sum_{\{v,h\}} v_i h_j q(v) p(h|v) - v_i h_j p(v) p(h|v) && (2.51)\\
&= \sum_{\{v,h\}} v_i h_j p(h|v) \left(q(v) - p(v)\right) && (2.52)\\
&= \sum_{\{v,h\}} v_i h_j p(h|v) p(v) \left(\frac{q(v)}{p(v)} - 1\right) && (2.53)\\
&= \left\langle v_i h_j \left(\frac{q(v)}{p(v)} - 1\right) \right\rangle_{p(h|v)p(v)} && (2.54)\\
&= \left\langle v_i h_j \left(\frac{q(v)}{p(v)} - 1\right) \right\rangle_{\text{model}}. && (2.55)
\end{aligned}$$

Essentially, we measure the correlations from the model and reconstruct the data correlations by reweighting the samples according to their target frequency[9].

2.2.3. Neuronal sampling following Buesing et al. [2011]

The first major obstacle between using Gibbs sampling as a model for Bayesian computation with biological neurons is the lack of an explicit notion of time. Buesing et al. added the notion of a refractory time τ_{ref} after a flip to $z = 1$ [Buesing et al., 2011]. In other words the neuron is not allowed to leave the state $z = 1$ for τ_{ref} updates. Here, we sketch the sampling method and refer the interested reader to the original publication for the proof of correctness and further information.

[9] This is how demographic adjustment in polling works. There, the tricky part is to define, or select, the correct target frequencies, but luckily for us this is trivial as we know the exact target distribution $q(\vec{v})$.

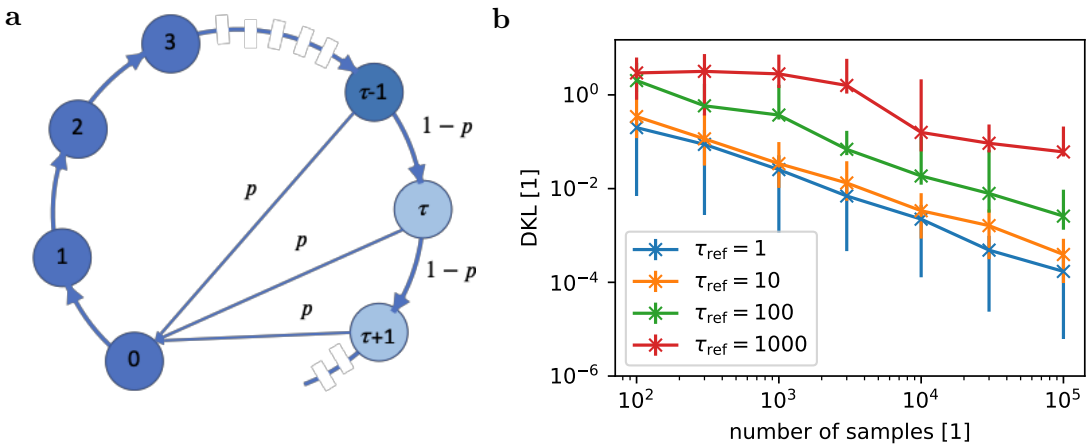

Figure 2.7.: **Buesing neuron model: a** State machine: The internal state of the neuron ζ (number in the circles) in our implementation. In contrast to the original implementation [Buesing et al., 2011], it counts the time since the last emitted spike. In each time step $\zeta \to \zeta+1$ is updated. Dark-blue states are interpreted as $z = 1$ and light-blue ones as $z = 0$ according to the binary decision $z = \mathbb{1}_{\zeta<\tau_{\text{ref}}}$. The spiking probability is given by $p = \sigma(u, \tau_{\text{ref}})\Theta(\zeta - \tau_{\text{ref}})$. For $\tau_{\text{ref}} = 1$ this implements Gibbs sampling exactly. **b** Sampling convergence for different refractory times as a function of the number of update steps. Longer refractory times τ_{ref} correspond to higher number of required updates. The points show the median DKL and the error bars show the minimal and maximal observed value over 11 independent sampling runs.

In order to implement a finite refractory time τ the sampling unit's state needs to be augmented with information about the time since the last spike. For reasons that will become obvious we deviate from the original formulation and define the internal state variable ζ to count the number of update steps (or time steps) since the last spike[10]. It's update procedure is hence defined as:

$$\zeta(t+1) = \begin{cases} = 0 & \text{if spike} \\ = \zeta(t) + 1 & \text{else} \end{cases} \tag{2.56}$$

The binary state z can be recovered from this via:

$$z = \mathbb{1}_{\zeta<\tau_{\text{ref}}} \tag{2.57}$$

where $\mathbb{1}$ is the indicator function. As long as $\zeta < \tau_{\text{ref}} - 1$ the neuron is not allowed to spike again. Fig. 2.7a presents this evolution schematically. Dark-blue ζ's are considered

[10]The original formulation counted down from $\zeta = \tau_{\text{ref}}$ immediately after a spike to 0, staying there till the next spike.

as $z = 1$ states and light-blue ζ's as $z = 0$ states. Outside of this refractory period the probability of the reset (or spike) is given by [Buesing et al., 2011]:

$$\sigma\left(u, \tau_{\text{ref}}\right) = \sigma\left(u - \log \tau_{\text{ref}}\right) = \frac{1}{1 + \exp(-u + \log \tau_{\text{ref}})} = \frac{1}{1 + \tau_{\text{ref}} \exp(-u)}. \tag{2.58}$$

We call this function the activation function of the neuron, as it represents the firing probability of the neuron. Eq. (2.58) corresponds to the gain function of the Gibbs sampler $\sigma(u)$ Eq. (2.36) with the argument shifted by $-\log \tau_{\text{ref}}$. This lowers the spiking probability in order to compensate for the stickiness of the $z = 1$ state. The intuition is that the new normalization factor is given by $1 = p(z_k = 0) + \tau p(z_k = 1)$[11].

This stochastic neuron model is completely equivalent to the Gibbs sampling model in the sense that it can be proven that the stationary distribution of this sampling method is the same Boltzmann distribution [Buesing et al., 2011]. Their convergence behavior (cf. Fig. 2.7b) differs slightly due to the introduced autocorrelation by the stickiness of the $z = 1$ state, which explicitly breaks the symmetry between the $z = 0$ and the $z = 1$ state.

Comparing this model to what we learned in Section 2.1.1 about biological neurons and their interactions we note that the Buesing neuron gets us only a part of the way. It allows us to reduce the communication over the state of the neuron to the spike times (if the refractory time τ_{ref} is known or some global constant), but it still calculates the membrane potential based on

$$u_k = b_k + \sum_i w_{ki} z_i \tag{2.59}$$

using the binary state z_i of all other neurons.

We already made the remark that this looks eerily similar to the form of the synaptic input when we named this as the potential of the neuron in Eq. (2.37). Here we need to take a bit of a closer look in order to find the differences that remain with the direct interaction. Effectively the formulation from [Buesing et al., 2011] uses rectangular PSPs that onset at the flip from $z = 0$ to $z = 1$ and end τ time steps later. In Section 2.1 we calculated the form of the PSP between for COBA and CUBA neurons, Eq. (2.16) and Eq. (2.9) respectively. For $\tau_{\text{m}} \to 0$ these reduce to exponential PSPs on the membrane potential. Since the Buesing neurons do not implement the leaky integration this should be the relevant limit. As such the proposition would be that the shape of the PSP is the most obvious deviations that [Buesing et al., 2011] does not yet account for[12].

If we are willing to give up on the mathematical proofs, changing the interaction shape becomes trivial. We can rewrite the membrane calculation in terms of interaction kernels as

$$u_k = b_k + \sum_i w_{ki} \kappa\left(\zeta_i, \tau_{\text{syn}}\right), \tag{2.60}$$

[11] The rest, as they say, is just math. We refer the interested reader to the original publication for the explicit proof [Buesing et al., 2011].

[12] We will see in Section 3.3 that this is not necessarily correct.

where we allow the interaction kernel to be parametrized by a single time constant τ_{syn}, which we will later link to the synaptic time constant from Section 2.1.1.

In the original formulation from [Buesing et al., 2011] the interaction only depended on the binary state $z = \mathbb{1}_{\zeta<\tau_{\text{ref}}}$. Restating this as an interaction kernel this results in a rectangular interaction (cf. Fig. 2.8a):

$$\kappa_{\text{rect}}(\zeta, \tau_{\text{syn}}) = \Theta(\zeta)\,\Theta(\tau_{\text{syn}} - t) = \mathbb{1}_{\zeta<\tau_{\text{syn}}} = z \tag{2.61}$$

Where the length of the rectangular kernel τ_{syn} is chosen to correspond to the refractory time constant of the sampler τ_{ref}. This description might seem a bit clumsy, but it will fall into place once we get to finally discuss the sampling with LIF neurons in the next section (Section 2.2.4).

For now let us introduce the additional interaction kernels we want to use: With the PSPs being approximated by an exponential function (cf. Fig. 2.8d), this is the next interaction kernel that we are interested in:

$$\kappa_{\text{exp}}(t, \tau_{\text{syn}}) = \frac{\Theta(t)}{\sum_{\hat{t}\leq\tau_{\text{ref}}} \exp\left(-\frac{\hat{t}}{\tau_{\text{syn}}}\right)} \exp\left(-\frac{t}{\tau_{\text{syn}}}\right). \tag{2.62}$$

The normalization ensures that the integral of the elicited PSP is a fixed number independent of the time scale of the exponential kernel τ_{syn}. At this point it becomes obvious why we had to switch our definition of ζ in contrast to the original model [Buesing et al., 2011]. They used ζ to count down until the neuron can fire again, at which point the internal state is stuck at $\zeta = 0$, effectively erasing the information of the spike time. However, for the exponential interaction we do require this knowledge, even after the neuron is no longer considered to be in state $z = 1$, in order to calculate the non-zero PSP contribution at times $t - t_{\text{spk}} \geq \tau_{\text{ref}}$. The original publication could ignore those for a slightly simpler notation in their proofs [Buesing et al., 2011].

At this point we took into account both the asymmetry between the *has just fired* $z = 1$ and the *can fire* $z = 0$ state and also added an option to take the PSP form of a single input spike into account. Since we only track the time since the last spike we also drop all but the newest term from Eq. (2.8) and prevent stacked PSPs from the same source. We will discuss the reasoning, beyond mathematical and modeling simplicity, behind this later in Section 3.1.2 after we introduce how to do sampling with LIF neurons in the next section (cf. Fig. 3.1).

For reasons that we will discuss in Section 4.3 we define two additional kernels: An exponential kernel without the tail contribution at $t \geq \tau_{\text{ref}}$ (cf. Fig. 2.8b):

$$\kappa_{\text{cuto}}(t, \tau_{\text{syn}}) = \begin{cases} = \kappa_{\text{exp}}(t, \tau_{\text{syn}}) & \text{if } t \leq \tau_{\text{ref}} \\ = 0 & \text{else} \end{cases} \tag{2.63}$$

and a rectangular kernel with an additional tail contribution for $t \geq \tau_{\text{ref}}$ (cf. Fig. 2.8c)

$$\kappa_{\text{tail}}(t, \tau_{\text{syn}}) = \begin{cases} = \kappa_{\text{rect}}(t, \tau_{\text{syn}}) & \text{if } t \leq \tau_{\text{ref}} \\ = \kappa_{\text{exp}}(t, \tau_{\text{syn}}) & \text{else} \end{cases} \tag{2.64}$$

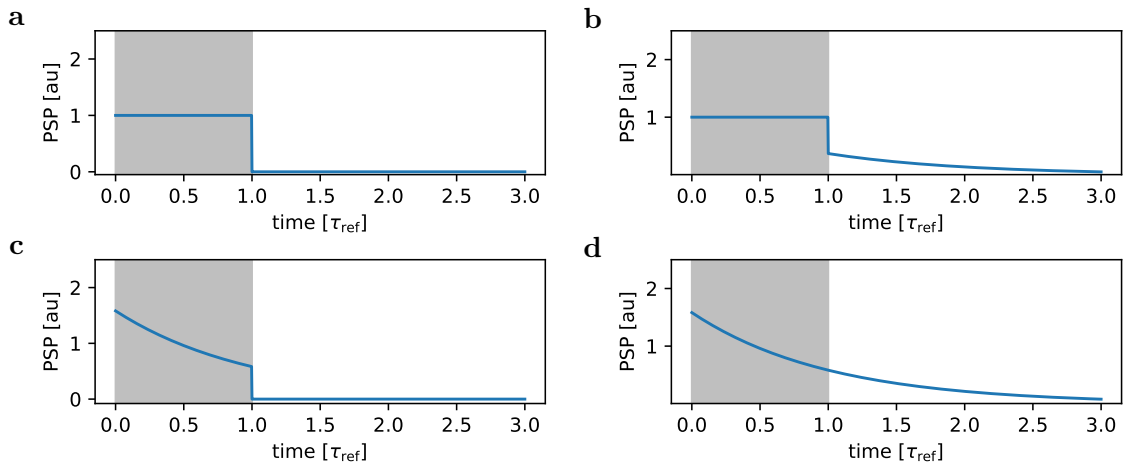

Figure 2.8.: **Different shapes of the PSP:** We discriminate the parts of the PSP within the refractory period $t - t_{\text{spk}} < \tau_{\text{ref}}$ and after the refractory period $t - t_{\text{spk}} > \tau_{\text{ref}}$ and choose in both parts between a rectangular and an exponentially decaying function. **a-d** show κ_{rect}, κ_{tail}, κ_{cuto} and κ_{exp} respectively for $\tau_{\text{syn}} = \tau_{\text{ref}}$. The exponential function is always scaled such that the integral (evaluated only at the simulation time steps) over the refractory period is equal to τ_{ref}.

These two allow us to separate out the contributions of the additional tail versus the contributions of the initial overshoot of the interaction on the dynamical evolution.

We, silently, introduced the normalization as the total interaction mass U within the refractory period by our choice of the denominator in Eq. (2.62). This is based on the translation scheme for LIF networks that we will discuss in Section 2.2.4 [Petrovici et al., 2016]. So far we *choose* $\tau_{\text{ref}} = \tau_{\text{syn}}$. In principle we could loosen this restriction, at which point the interaction kernels Eqs. (2.61) to (2.64) need to be normalized again. For consistency with the initial motivation of linking parameters between networks of LIF neurons and Boltzmann distributions we choose to keep the integral within the refractory period τ_{ref} constant:

$$\sum_{\zeta<\tau_{\text{ref}}} \kappa_{\text{rect}}(\zeta, \tau_{\text{syn}}) = \sum_{\zeta<\tau_{\text{ref}}} \kappa_{\text{exp}}(\zeta, \tau_{\text{syn}}) = \sum_{\zeta<\tau_{\text{ref}}} \kappa_{\text{cuto}}(\zeta, \tau_{\text{syn}}) = \sum_{\zeta<\tau_{\text{ref}}} \kappa_{\text{tail}}(\zeta, \tau_{\text{syn}}) = \tau_{\text{ref}} \tag{2.65}$$

and the weight w directly scales the, thus normalized, kernels.

2.2.4. Sampling with LIF neurons

Over the last few sections we presented stochastic neuron models and how they implement sampling from binary Boltzmann distributions. Now, we need to implement such a scheme with biologically-inspired neurons. In Section 2.1.1 we introduce a simple

mechanistic description of a spiking neuron, the LIF model.

The most glaring difference to the stochastic models we discussed before is the determinism of the LIF model. It is known since the works of Habenschuss et al. that recurrent networks of LIF neurons sample from a static probability distribution [Habenschuss et al., 2012]. But it is the work of Petrovici et al., that showed that LIF neurons of a particular parameterization and subject to high-frequency Poisson spike input approximately sample from Boltzmann distributions [Petrovici et al., 2016].

In this subsection we briefly recap the setup and motivate its relation to the Buesing model from the previous subsection (Section 2.2.3). Later in Chapter 3 we will perform a more detailed discussion of the dynamical aspects of these LIF networks, going further than what was already developed in [Petrovici et al., 2016, 2015], which will be the focus here. As the following line of argument for COBA and CUBA neurons nearly agree and CUBA neurons are slightly less involved, we will restrict our explanation here to the latter case and only remark on the equivalent results for the COBA case. For the exact derivation and more details the interested reader is referred to [Petrovici, 2015].

Each sampling LIF neuron is subject to stochastic input form two sources, one excitatory and one inhibitory one, of random spikes (cf. Fig. 5.3A). In practice, these will be Poisson-distributed with rates r_{exc} and r_{inh} and connected with synaptic strengths W_{exc} and W_{inh}, respectively. The neuron's membrane potential time course $u(t)$ can then be shown to implement an Ornstein-Uhlenbeck (OU) process.

Definition 2.2.5. *Ornstein-Uhlenbeck (OU) process*

The OU [Bibbona et al., 2008, Wikipedia, 2016] process x_t is defined by the following stochastic differential equation:

$$dx_t = \theta(\mu - x_t)\, dt + \sigma\, dW_t \tag{2.66}$$

where θ, σ and μ are parameters for the autocorrelation time, the mean of the drift and the scale of the noise, respectively. W_t denotes the Wiener process.

In terms of probability density functions $P(x,t)$ which describes the probability of the value of x_t over time, this process obeys the Fokker-Planck equation:

$$\frac{\partial P}{\partial t} = \theta \frac{\partial}{\partial x}((x-\mu)P) + \frac{\sigma^2}{2}\frac{\partial^2 P}{\partial x^2}. \tag{2.67}$$

For this process the transition probability between two values x and x' at times t and t' is given by:

$$P(x,t \mid x',t') = \sqrt{\frac{\theta}{2\pi\frac{\sigma^2}{2}(1-e^{-2\theta(t-t')})}} \exp\left[-\frac{\theta}{\sigma^2}\frac{(x-x'e^{-\theta(t-t')})^2}{1-e^{-2\theta(t-t')}}\right]. \tag{2.68}$$

Mathematically speaking this is the solution to the Fokker-Planck equation Eq. (2.67) for the initial condition $P(x,t') = \delta(x-x')$.

For the membrane potential evolution of our LIF neuron we make the following identifications:

1. x is the membrane potential value u
2. $\theta = \frac{1}{\tau_{\text{syn}}}$ is given by the synaptic time constant
3. σ is given by the a combination of the noise parameters $r_{\text{exc}}, r_{\text{inh}}, W_{\text{exc}}, W_{\text{inh}}$ and the neuron parameters, in particular, the leak conductance g_{l} and the membrane capacitance C_m.

The time constant of the OU process actually refers to the larger of the two timescales that influence the membrane evolution, namely the time constant of the synaptic input τ_{syn} and membrane time constant τ_{m}. Since it is the synaptic input that we are interested in, we want the membrane potential to follow the input quickly. This requires a small membrane time constant:

$$\tau_{\text{m}} = \frac{C_m}{g_{\text{l}}} \ll \tau_{\text{syn}}. \tag{2.69}$$

As the capacitance C_m is fixed by the neuron's physiological properties the natural way to achieve this small τ_{m} is via a large conductance. In the CUBA model case this means that we have to choose a large leak conductance g_{l}. For COBA neurons there are multiple conductances that all contribute the the effective membrane time constant (cf. Eq. (2.20)):

$$\tau_{\text{m}} = \frac{C_m}{g_{\text{l}} + g_{\text{exc}}^{\text{syn}}(t) + g_{\text{inh}}^{\text{syn}}(t)} \tag{2.70}$$

and therefore the membrane time constant of a COBA neuron becomes small automatically for a significant level of synaptic input. In other words, when we are requiring a small membrane time constant τ_{m} we only require the neuron to be in a high-conductance state (HCS) [Destexhe et al., 2003, Kumar et al., 2008].

Ignoring the threshold and spike-behavior we can calculate the mean $\bar{u}$ and variance $\text{Var}[u]$ of the membrane potential time course $u(t)$ implementing the OU process [Petrovici et al., 2016]:

$$\bar{u} = V_{\text{l}} + \frac{I_{\text{ext}}}{g_{\text{l}}} + \frac{\sum_k W_k \nu_k \tau_{\text{syn}}^k}{g_{\text{l}}} \tag{2.71}$$

$$\text{Var}[u] = \sum_k \left[\frac{\tau_{\text{m}} \tau_{\text{syn}}^k}{C_m \left(\tau_{\text{m}} - \tau_{\text{syn}}^k\right)} \right]^2 W_k^2 \nu_k \left(\frac{\tau_{\text{m}}}{2} + \frac{\tau_{\text{syn}}^k}{2} - 2 \frac{\tau_{\text{m}} \tau_{\text{syn}}^k}{\tau_{\text{m}} + \tau_{\text{syn}}} \right) \tag{2.72}$$

In Fig. 2.9 we show an exemplary time evolution of $u(t)$ in blue with a smaller-than-threshold mean $\bar{u} < V_{\text{thresh}}$. The neuron stochastically fires and enters the refractory period (gray shaded periods in Fig. 2.9), see [Petrovici, 2015] for the derivation. Within this refractory time frame the neuron is considered to be in state $z_k = 1$. Otherwise we identify it as $z_k = 0$. This association is chosen due to the fact that the spike time corresponds to the onset of the interaction, which gives a non-zero contribution (which in turn corresponds to the multiplication with $z_k = 1$ rather than $z_k = 0$). The task now becomes to calculate the relative fraction of time the neuron spends in the $z = 1$ state. This is related to the *neural computability condition* for Gibbs and Buesing neurons in Eq. (2.35).

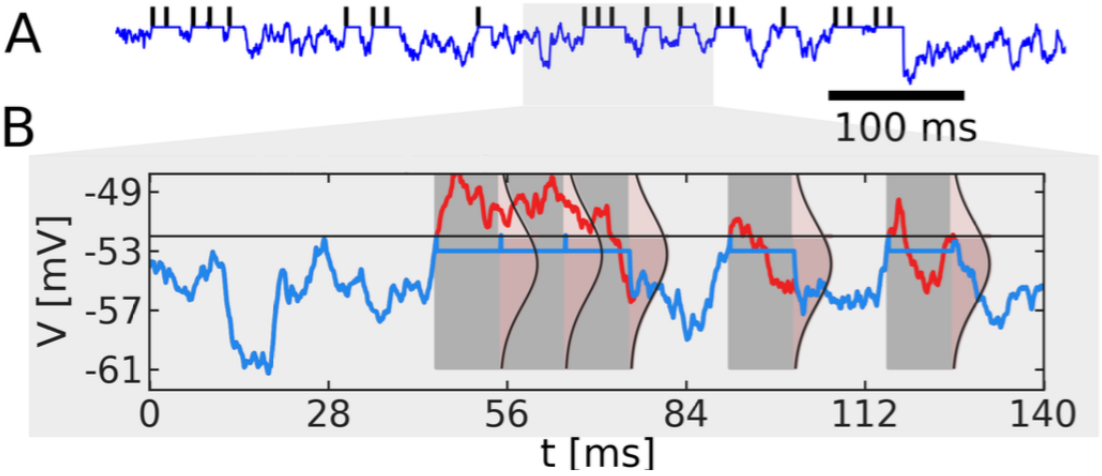

Figure 2.9.: **LIF sampling and state association: A** Membrane potential time course of a single sampling neuron under high frequency Poisson noise input. **B** Actual (blue) and effective (red) membrane potential. Whenever the membrane potential u crosses the threshold V_{thresh} the neuron enters its refractory period by being clamped to the reset potential $V_{\text{reset}} < V_{\text{thresh}}$ (gray shade). Within this time frame the neuron is considered to be in state $z = 1$, otherwise it is assigned the state $z = 0$. At the end of the refractory period the membrane potential decays towards the effective membrane potential value (distribution shown in pink). At the end of each burst period the neuron remains in $z = 1$ longer than the spiking condition $u > V_{\text{thresh}}$ is fulfilled. Figure adapted from [Petrovici et al., 2016].

Activation function

Let us now look at a single LIF neuron under Poisson input and let us assume that we can only observe the output spikes. In Eq. (2.19) we introduced the concept of the instantaneous target membrane potential u^{eff}. For a sufficiently low mean potential $\bar{u} = \langle u^{\text{eff}} \rangle$ we will not observe any output spikes as a supra-threshold potential $u > V_{\text{thresh}}$ will never occur. At this point we know the membrane potential distribution $\rho(u)$ is the steady-state distribution of the OU process. It is a Gaussian with mean $\bar{u}$ (cf. Eq. (2.71)) and variance $\text{Var}[u]$ (cf. Eq. (2.72)).

Shifting now $\bar{u}$, typically implemented by changing V_{l}[13], the frequency with which u crosses the threshold steadily increases. The mean time between the end of a sequence of consecutive spikes, also called a burst, and a next first spike T_1 can be calculated from the moments of the first passage time distribution of an OU process (see [Petrovici, 2015]). However, this covers only the spikes that happen after a long time. I.e., where we do not have any additional information and we therefore know the membrane potential distribution. The longness of this time is given by the timescale parameter of the OU

[13]Changing I_{ext} would be equivalent up to the factor g_{l}.

process Θ which corresponds to the inverse of the synaptic time constant τ_{syn}. This gives the autocorrelation time scale of the membrane potential. Therefore, a couple of τ_{syn} are required to pass before the neuron *forgot* its old state completely (cf. Section 3.1.3 and Fig. 3.3).

We chose

$$\tau_{\text{ref}} = \tau_{\text{syn}}. \tag{2.73}$$

in order to relate the time scale on which the post-synaptic neuron experiences the interaction triggered by the spike (i.e., the $z = 0 \to z = 1$ transition) of the pre-synaptic neuron τ_{syn} to the time scale on which the pre-synaptic neuron experiences the same transition, i.e. where its dynamics are suppressed, τ_{ref}. We equate the two in order to force some consistency between the experiences of the two neurons. However, this choice also means that the distribution of u^{eff} at the end of the refractory time of a single spike is significantly different from the free distribution (cf. Section 3.3 for more details).

We call second and following spikes, which happen closely after the end of the refractory period following the initial spike, part of a burst. The probability of having a spike immediately after the end of the refractory period is significantly larger than at a random later point. This is because we do have the knowledge about the membrane potential $u(t_{\text{spk}}) = V_{\text{thresh}}$ at the first spike time t_{spk} to take into account. At this point of a first spike we know (at least for $\tau_{\text{m}} \to 0$) that the membrane potential is crossing the threshold, i.e., $u^{\text{eff}} = V_{\text{thresh}}$. From this [Petrovici et al., 2016] calculated the supra-threshold probability mass

$$P_2 = \int_{V_{\text{thresh}}}^{\infty} du^{\text{eff}} \rho(u^{\text{eff}}(t_{\text{spk}} + \tau_{\text{ref}})) \tag{2.74}$$

and thereby estimated the probability of a burst with at least 2 spikes. At this point we do not know the exact target voltage u^{eff} at the time of the second spike anymore which makes the calculation of P_3 more involved. We will elaborate on this in more detail in Section 3.3, the interested reader should also read the original publication [Petrovici et al., 2016].

In analogy to the neural computability condition, we are interested in the probability to find the neuron in state $z = 1$ as a function of constant input, i.e., for fixed $\bar{u}$. Petrovici et al. [2016] has calculated this to be

$$p(z = 1)(\bar{u}) = \frac{\sum_{n=1}^{\infty} P_n(\bar{u}) n \tau_{\text{ref}}}{\sum_{n=1}^{\infty} P_n(\bar{u})(n\tau_{\text{ref}} + \sum_{k=1}^{n-1} \tau_k^b + T_n(\bar{u}))} \tag{2.75}$$

where P_n is the probability of a burst consisting of n spikes, T_n is the mean first passage time of the OU process starting after the end of a burst of length n and τ_k^{b} is the sum of the drift times between the spikes within the burst. The latter tends to 0 as $\tau_{\text{m}} \to 0$, P_n is close to exponentially decreasing and T_n can be calculated from the moments of the first passage time distribution of the OU process [Petrovici, 2015]. The resulting activation function is shown in Fig. 2.10a and b for current- and conductance-based neurons respectively. We will discuss this function in more detail when we discuss the peculiarities of LIF sampling in Section 3.3. For now, we remark on the functional

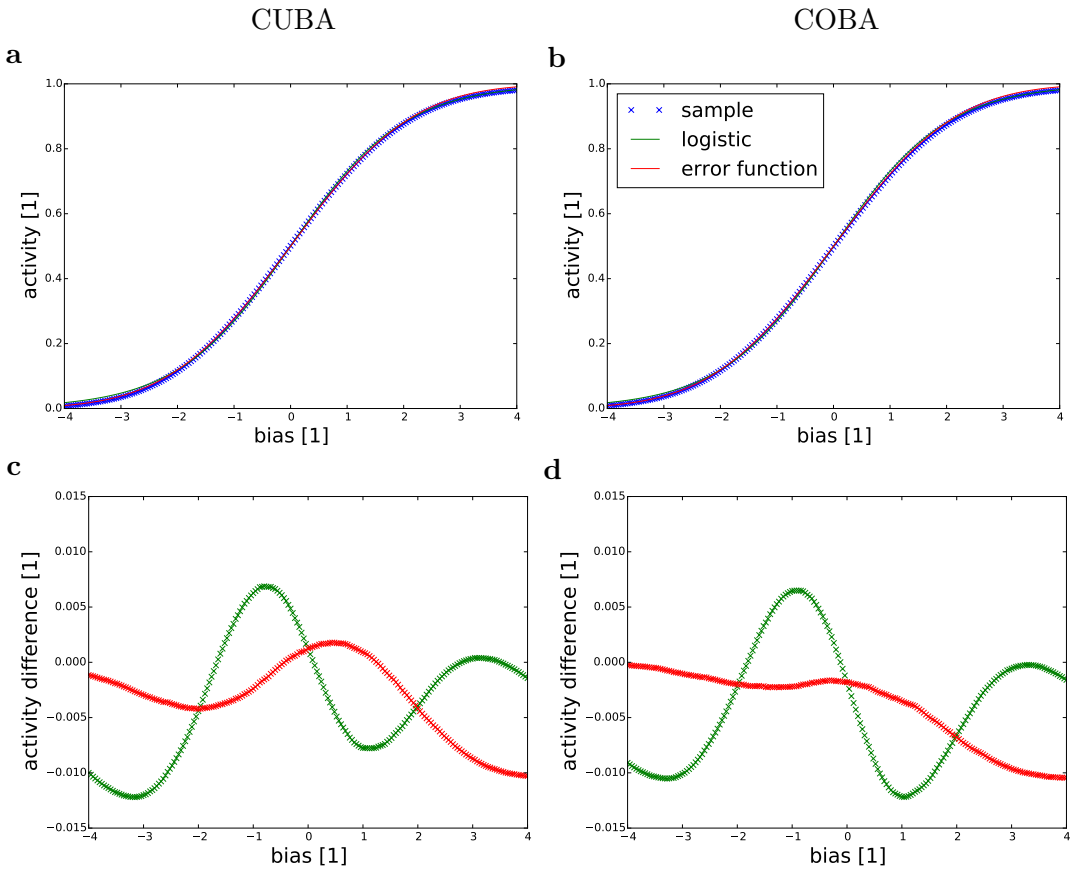

Figure 2.10.: **Activation function: a(b)** Mean activity, $p(z = 1)$, of a current-based (conductance-based) neuron for different bias configurations, see main text for the implementation via V_l. **c(d)** Residuals to a logistic $\sigma(x)$ and error function $\mathrm{erf}(x)$ respectively. Both functions describe the form of the activation function well, with the error function exhibiting slightly smaller residuals. Figure taken from [Baumbach, 2016].

similarity to the activation function from the Buesing and Gibbs neurons Eq. (2.58) and Eq. (2.36), which leads us to fit a sigmoid

$$p(z=1)(\bar{u}) = \sigma\left(\frac{\bar{u} - u_{p05}}{\alpha}\right) \tag{2.76}$$

where the two free parameters u_{p05} and α are (complicated) functions of the noise and neuron parameters. We will discuss their meaning and functional implications in Section 4.1.

We also fit the error function $\mathrm{erf}\left(\frac{x-x_0}{\alpha}\right)$ as it is the cumulative function of the underlying Gaussian distribution. The resulting residuals are slightly smaller (cf. Fig. 2.10c and d) but still significant variations are observable.

Parameter translation

In order to use such a network of LIF neurons to implement sampling from Boltzmann distributions we need to establish a relationship between the parameters of said distribution w, b and their LIF counterparts W, V_l. We already alluded to our choice of implementing a bias for a stochastic LIF neuron via its leak potential V_l via which we used to manipulate the mean of its membrane potential distribution Eq. (2.71). In other words for an arbitrary bias Boltzmann parameter b, we can look up the target activation A from the Gibbs gain function Eq. (2.36) as

$$A = \langle z \rangle = \sigma(b) \tag{2.77}$$

from this, together with the parameters of the activation function of the LIF neuron Eq. (2.76) we can find the leak potential V_l that is required to achieve the same mean activation (cf. Fig. 2.10):

$$V_\mathrm{l} = \alpha b + u_{p05} \tag{2.78}$$

where both parameters α and u_{p05} depend on the neuron parameters as well as the noise configuration which gave rise to this particular activation function. This gives us the first part of our parameter translation. Here, we made the *choice* to implement the bias of the LIF neuron V_l via the leak potential V_l, in Section 4.1 we will use an external current and in Section 5.1 we will also see an implementation via a synaptic connection.

For the connection weights w the argument is a bit more involved: We already identified the strength of the synaptic connection to implement the two-point Boltzmann parameter w and we fixed the synaptic time constant τ_syn to correspond to the length of the refractory period τ_ref. Therefore, the remaining question is how we should scale the amplitude W such that the resulting stochastic behavior approximates the effect of the weight parameter in a Boltzmann distribution w.

For the Buesing neuron model we did scale the rectangular interaction kernel $\kappa_\mathrm{rect}(t)$ with the Boltzmann weight parameter w_{ij}. For LIF neurons Petrovici et al. [2016] fixes the integral of the PSP within the refractory period of the pre-synaptic neuron to the integral of the Buesing interaction (cf. Fig. 3.1a):

$$\tau_\mathrm{ref} w_{ij} = \int_0^{\tau_\mathrm{ref}} \frac{Wij}{\alpha} PSP(t) dt \tag{2.79}$$

with the $PSP(t)$ from Eq. (2.9) or Eq. (2.16). In order to make these two comparable (think dimension of the variable) we needed to normalize the PSP height by the width parameter of the activation function α. In other words the COBA-LIF weight parameter is given by:

$$W_{ij} = w_{ij} \frac{1}{\alpha C_m} \frac{V^{\text{rev}} - \bar{u}}{1 - \frac{\tau_{\text{syn}}}{\tau_{\text{m}}}} \left[\tau_{\text{syn}}(e^{-1} - 1) - \tau_{\text{m}} \left(e^{-\frac{\tau_{\text{syn}}}{\tau_{\text{m}}}} - 1 \right) \right] \tag{2.80}$$

the formula for CUBA can be derived analogously.

Both the translation of the biases as well as the weights depend on the parameters of the activation function. In particular, it is the width α that *defines* which neuron parameters W and V_l correspond to the meaning of the Boltzmann parameters $w = 1$ and $b = 1$ respectively.

Our choice of the leak potential V_l as the implementation basis for the bias is not biologically plausible. We learned in Section 2.1 that the leak potential is essentially given by the diffusion potential of the K^+-ions. However, as we are only interested in neurons under significant Poisson stimulus this resting potential does not correspond to the mean of the membrane potential $\bar{u}$. The latter is rather a function of the leak potential V_l and the excitatory and inhibitory noise input. We will discuss how the noise input influences the activation function in Section 4.1 and how we would implement a bias in a biology-compatible way. In practice we will still just adjust the value of the leak potential V_l in our simulations.

We stress that this translation scheme is a first-order approximation which requires the correspondence of $\tau_{\text{syn}} = \tau_{\text{ref}}$ at least up to a factor of 2 before the sampling performance degrades [Probst et al., 2015]. But even so this relation completely ignores the different dynamics between LIF and Buesing neurons which we will discuss in Section 3.3 and drops the effect of the lingering PSP after the end of the refractory period. This leads to interesting behavior in Ising-like networks which we will discuss in Section 4.3.

Even without the direct training, we can implement sampling with LIF neurons from only observing their activation function (cf. Fig. 2.10). The typical toy-examples of small Boltzmann distributions with $n \approx 5$ neurons result in DKLs on the order of 1×10^{-2} to 1×10^{-3}, depending a bit on the neuron and noise configuration as well as the distribution [Petrovici et al., 2016, Bytschok et al., 2017]. Larger networks, with post-training, have been demonstrated to implement generative models for binary images [Petrovici et al., 2017b, Leng et al., 2018, Kungl et al., 2019, Dold et al., 2019] and recently also a representation of quantum states has been shown to be possible Section 5.2.

3. Dynamical aspects of LIF sampling

At the end of the last chapter we introduced a model for spike-based probabilistic computing, in this chapter we will now discuss the dynamics of single such neurons. We start this discussion by having a look at the effect of the temporally extended interactions in Section 3.1. In Section 3.2 we present the biological and modeling motivation of the background noise sources, discuss different implementations and their resulting correlation structure. Finally, we present a more involved Markovian model in Section 3.3 and show that it significantly outperforms the stochastic neuron model developed in [Buesing et al., 2011] in predicting the spike response of stochastic LIF neurons. In the last section, Section 3.4 we give a brief overview of the different models presented. In particular, we will compare which components they attempt to cover and discuss where the fundamental limits of the applicability of the models lie.

3.1. Issues originating in the interaction shapes

In Section 2.2.4 we introduced the general idea of LIF sampling originally developed by [Petrovici et al., 2016]. There, we discussed the relationship to other, more abstract, sampling schemes and motivated the parameter translation formulas Eq. (2.78) and Eq. (2.80). At that point we already mentioned that, in particular, the translation of the weights Eq. (2.80) is based on a number of assumptions. The biggest of which is that all spikes are going to have identical dynamical impact – independent of the history of the synapse. In this section we will discuss in how far this assumption is true and which biological mechanisms exist to improve its validity.

3.1.1. Bursting neurons and short-term plasticity

In the following these boxes will preface sections of the manuscript that contain work done in close collaboration with other people. If applicable they refer to the relevant publications or if non is available reference the relevant master thesis. This Section 3.1.1 repeats results already contained in [Petrovici, 2015] but which forms necessary background information.

At the end of Section 2.2.4 we remarked that the parameter choice of $\tau_{\text{syn}} = \tau_{\text{ref}}$ is made in order to have the interaction act on the same time scale on which the state $z = 1$ is fixed. Due to the exponential decay of the synaptic interaction[1] the effect on the membrane potential is not restricted to this time frame. There is still a significant contribution due to the exponential tail (cf. Fig. 3.1a).

In the bare (static) synapse model from Section 2.1.1 an additional spike always adds a fixed amount of additional current[2] proportional to the strength of the synapse W. This means that the assumptions underlying the weight translation (cf. Eq. (2.80)) are fundamentally flawed as we postulated that only the current spike influences the effect on the post-synaptic neuron. However, with the potential remains of the exponential tail, it is also the history of the pre-synaptic activity that determines the height of the current PSP.

To make it explicit why this is a problem: Suppose that we have a model distribution $p(\vec{z})$ and we want to calculate the distribution conditional to $z_0 = 1$. In order to implement this we would *clamp* the neuron 0 to the state $z_0 = 1$ by e.g. setting its leak potential V_l to be very large. At this point neuron 0 would be forced to be continuously

[1] Remember: This is also true for the effect on the membrane potential, the PSP, as we operate in the limit of $\tau_\text{m} \to 0$, cf. Section 2.2.4.

[2] We will restrict the discussion to the current-based case.

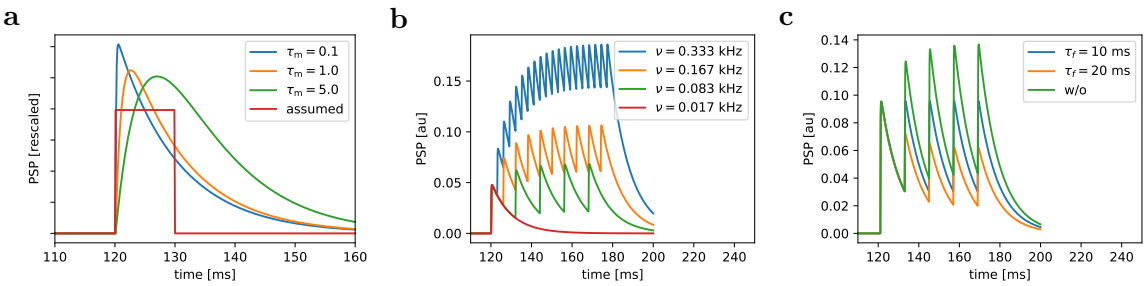

Figure 3.1.: **Stacking PSPs: a** Shape of a single current-based post-synaptic potential (PSP) on a resting neuron for different membrane time constants τ_m. The weight translation from Eq. (2.80) matches the integral within τ_ref with the red rectangular PSP. Shorter membrane time constants τ_m leads to earlier and higher peaks with a reduced tail amplitudes. **b** The height of PSP for consecutive spikes also depends on the remaining amplitude of the previous PSP. Higher input frequencies lead to higher steady-state PSP heights and at the end of a high stimulus phase (at 180 ms) the effect remains for multiple τ_ref. **c** Effect of the depressing configuration of the Tsodyks-Markram short-term plasticity mechanism. With a recovering time constant $\tau_\mathrm{rec} = \tau_\mathrm{syn}$ (blue) the maximum of the elicited PSPs is constant. Simulation parameters for these figures can be found in Appendix B.1.2.

active (burst) and produce a regular spike train with frequency[3]:

$$\nu = \frac{1}{\tau_\mathrm{ref}}. \tag{3.1}$$

The first PSP would make the membrane potential u rise to a significantly lower level than the second PSP and so on (see Fig. 3.1b). A higher membrane potential u leads to an increased leak current:

$$I_\mathrm{l} = g_\mathrm{l}\,(u - V_\mathrm{l}) \tag{3.2}$$

as it is proportional to the distance to the leak potential $u - V_\mathrm{l}$. At some point the additional leak compensates the added post-synaptic current (PSC) and therefore the membrane potential u approaches a steady-state level. This happens within a few τ_ref depending on the frequency of the regular spike stimulus (see Fig. 3.1b). From a modeling perspective we would like *all* spikes from the spike source to have the same effect on the neuron, as the spike source represents a pre-synaptic neuron which we interpret as a binary unit.

Luckily for our probabilistic interpretation this is also a problem for the biological interpretation of the LIF model. As is, static synapses allow for arbitrarily strong synaptic

[3] We again neglect the drift after the end of the refractory time for this argument. In practice, the frequency would have to be slightly lower.

currents generated by ion pumps. In practice, the amount of current available to change the membrane potential u is limited by the availability of neurotransmitters. The standard model for these effects – essentially modeling resource availability – is the so called Tsodyks-Markram model [Tsodyks and Markram, 1997] for short-term plasticity (STP).

This model separates the neurotransmitters into three partitions: The recovered partition R, the active partition A and the inactive partition I. The recovered neurotransmitters are available to be activated by a spike traveling along the axon, the active partition corresponds to the active synaptic input current and the inactive partition is in a hold-off state, where they no longer contribute to the synaptic input I^{syn} but are also not yet available again.

For the moment we assume that the total amount of neurotransmitters (roughly: the strength W) at a synapse is constant and hence we know:

$$1 = R + A + I. \tag{3.3}$$

Whenever the synapse registers an action potential from its pre-synaptic neuron it moves a fraction U (utilization factor) from the recovered partition R into the active partition A. The utilization factor describes the efficiency of the synapse, where a higher U results in the first spike being more influential, but also depletes more of the resources that require recovery afterwards. In the original version the active partition A is what gives rise to the synaptic input current

$$I^{\text{syn}}(t) = WA(t) \tag{3.4}$$

These active neurotransmitters then exponentially decay into the inactive state I with some time constant τ_{rec}. The inactive neuron transmitters I recover into the available partition R with some (potentially) different facilitating time constant τ_{facil}. Putting this description into formulas results in the system of coupled ODEs that form the Tsodyks-Markram (TSO) model:

$$\frac{dA}{dt} = -\frac{A(t)}{\tau_{\text{rec}}} + UR(t)\delta\left(t - t_{\text{spk}}\right) \tag{3.5}$$

$$\frac{dI}{dt} = -\frac{I(t)}{\tau_{\text{facil}}} + \frac{A(t)}{\tau_{\text{rec}}} \tag{3.6}$$

$$\frac{dR}{dt} = \frac{I(t)}{\tau_{\text{facil}}} - UR(t)\delta\left(t - t_{\text{spk}}\right). \tag{3.7}$$

The assumption of static synapse strength (in terms of total number of neurotransmitters) is justified as retained changes of synapse strength occur on longer timescales than the ones involved here (minutes to hours vs milliseconds to seconds [Markram et al., 2011a]).

Different settings for the facilitation τ_{facil} and recovery time constant τ_{rec} lead to different behavior of the membrane potential of a single neuron. With the choice of

$$U = 1\,, \quad \tau_{\text{facil}} = 0\,\text{ms} \quad \text{and } \tau_{\text{rec}} = \tau_{\text{syn}} = 10\,\text{ms} \tag{3.8}$$

the height of the PSP envelop is constant as the recovered partition R contains exactly enough neurotransmitters to compensate for the already lost PSP height (see Fig. 3.1c for examples). Petrovici et al. [2016] introduced the term *renewing* synapses for this configuration of synapses and has shown that it is beneficial for the sampling accuracy. Higher values of τ_{rec} (as compared to the synaptic time constant τ_{syn}) lead to a depressed reaction while lower values lead to an increased PSP height, with $\tau_{\text{rec}} = 0\,\text{ms}$ corresponding to the original static case (cf. Fig. 3.1c).

There is also a way to implement facilitating synapses, where the first few inputs act as a primer for the synapse to get started. For these the synaptic current is proportional to the third of three partitions I, rather than the second A as in our case. We will use renewing synapses and refer the interested reader to [Tsodyks and Markram, 1997] for a general discussion of the TSO mechanism and to [Leng et al., 2018] for a functional application of TSO as a mixing facilitator within the LIF sampling framework. We should also note that the hardware implementations within the BrainScaleS system only allows for one of the two parameters to be non-zero, albeit we will not be using TSO in Chapter 5.

3.1.2. Long-term influences of synaptic input

In the previous section we have seen how we can avoid a variation in the PSP height between different spikes within a burst and thereby prevent the most egregious violation of the assumption that all $z = 1$ states and thereby all spike effects are equal. However, even with TSO being perfectly trimmed for our use case, it only alleviates with the variation of the membrane potential within the refractory period of the pre-synaptic neuron. The membrane potential after the pre-synaptic neuron becomes available again can still vary depending of it continuing to burst or the previous spike being the last spike of the burst. The latter always produces excess PSP mass U in the tail that we did not take into account in the translation Eq. (2.80). In contrast this excess PSP mass U is absorbed into the next PSP if the pre-synaptic neuron spikes again.

In order to quantify this we need to define the PSP mass that we used colloquially so far:

Definition 3.1.1. *PSP mass*

For an isolated neuron exposed to only a single spike source we define the mass of the PSP of an input spike at time t_0 as the integral over the displacement of the membrane potential time course $u(t) - V_\text{l}$ of the neuron up to the next input spike at time t_1

$$U = \int_{t_0}^{t_1} dt\; u(t) - V_\text{l} = \int_{t_0}^{t_1} dt\; PSP(t). \tag{3.9}$$

We separate this combined effect into the *tail* contribution for times $t > t_0 + \tau_{\text{ref}}$ (white area in Fig. 3.2a and b) and the *refractory* contribution for times $t_0 < t < t_0 + \tau_{\text{ref}}$ (orange shaded area in Fig. 3.2a and b).

The amount of the PSP tail that is visible depends on the activity level of the pre-synaptic neuron: For low activity neurons most of the tail is visible (Fig. 3.2a for $\langle z \rangle =$

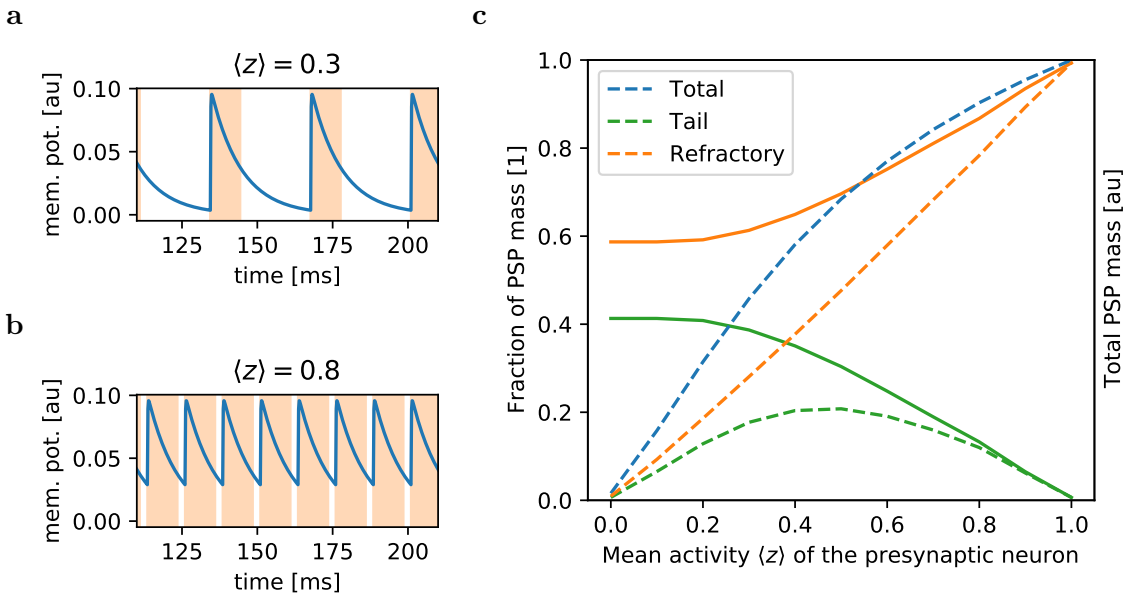

Figure 3.2.: **Relative contributions to the membrane potential: a (b)** Membrane potential time course due to a single regularly firing neuron with low (high) activity and excitatory synaptic weight. Inhibitory connections work analogously. The synaptic connection is configured renewingly and the time frame in which the pre-synaptic neuron is considered to be in state $z = 1$ is marked in orange. **c** Relative fraction of tail (orange) and refractory (green) PSP mass U (left y-axis, solid lines) and absolute PSP mass U (right y-axis, dashed lines) for different activities $\langle z \rangle$. Total (blue) and refractory PSP mass increase monotonously, with the latter a, by construction, linear growth. The relative fraction of the tail decreases roughly linearly above a mean activity of $\langle z \rangle = 0.4$. Simulation parameters can be found Appendix B.1.3.

0.3) and thereby the (unaccounted) *fraction* of the tail mass is large. In contrast, for high activity neurons, most of the tail mass is absorbed into the next PSP (see Fig. 3.2b for $\langle z \rangle = 0.8$).

In Fig. 3.2c we see the dependence of both the fractions (solid lines) as well as the integrals (dashed lines) as a function of the activity of a regularly firing pre-synaptic neuron for a fixed simulation time $T = 1000\,\mathrm{ms}$. The *total* PSP mass U (green) increases monotonously with increased pre-synaptic activity, but only the *refractory* mass (orange) does so linearly as we would expect if the PSP mass U were a good proxy for the expected contribution $\langle wz \rangle$. The *tail* mass (green), at least for our parametrization, is always smaller than the *refractory* mass, albeit for very low activity levels it contributes more than 40 % of the total. It is only for activities $\langle z \rangle > 0.4$ that this fraction starts to decrease notably, with it reaching 0 at $\langle z \rangle = 1$. This behavior does not significantly change when we use a stochastically firing pre-synaptic neuron (data not shown).

While we will see in Section 3.3 that the dynamical effect on the firing rate of the post-synaptic neuron is much more involved, here, we can already see that different pre-synaptic activity levels change the meaning of a connection parameter W. We will come back to this as a part of the explanation of what we observe for the phase space of Ising like networks in Chapter 4.

3.1.3. Autocorrelation or when is a new state a new state?

So far we have looked at the effect of a single neuron's output on an otherwise isolated neuron. This is a rather unusual and therefore uninteresting situation. Typically we are interested in the effect of a single neuron's output on a neuron under heavy noise input. As discussed in Section 2.2.4 this noise exposure makes the membrane potential u implement an Ornstein-Uhlenbeck (OU) process [Uhlenbeck and Ornstein, 1930, Bibbona et al., 2008] with time constant

$$\frac{1}{\Theta} = \tau = \tau_{\mathrm{syn}}, \tag{3.10}$$

target mean (cf. Eq. (2.71)):

$$\mu = \bar{u} = V_{\mathrm{l}} + \frac{I_{\mathrm{ext}}}{g_{\mathrm{l}}} + \frac{\sum_k W_k \nu_k \tau_{\mathrm{syn}}^k}{g_{\mathrm{l}}} \tag{3.11}$$

and variance (cf. Eq. (2.72))

$$\sigma^2 = \mathrm{Var}[u] = \sum_k \left[\frac{\tau_{\mathrm{m}} \tau_{\mathrm{syn}}^k}{C_m \left(\tau_{\mathrm{m}} - \tau_{\mathrm{syn}}^k \right)} \right]^2 W_k^2 \nu_k \left(\frac{\tau_{\mathrm{m}}}{2} + \frac{\tau_{\mathrm{syn}}^k}{2} - 2 \frac{\tau_{\mathrm{m}} \tau_{\mathrm{syn}}^k}{\tau_{\mathrm{m}} + \tau_{\mathrm{syn}}} \right) \tag{3.12}$$

. We expect the autocorrelation of the time course of the membrane potential

$$\mathrm{ACF}\,(u; \Delta t) = \left\langle \frac{(u(t) - \bar{u})\,(u(t + \Delta t) - \bar{u})}{\mathrm{Var}[u]} \right\rangle \tag{3.13}$$

to be exponentially decaying with the OU time constant $\tau_{\mathrm{syn}} = 10\,\mathrm{ms}$ as we have chosen the membrane time constant τ_{m} to be small (mimicking the high-conductance state

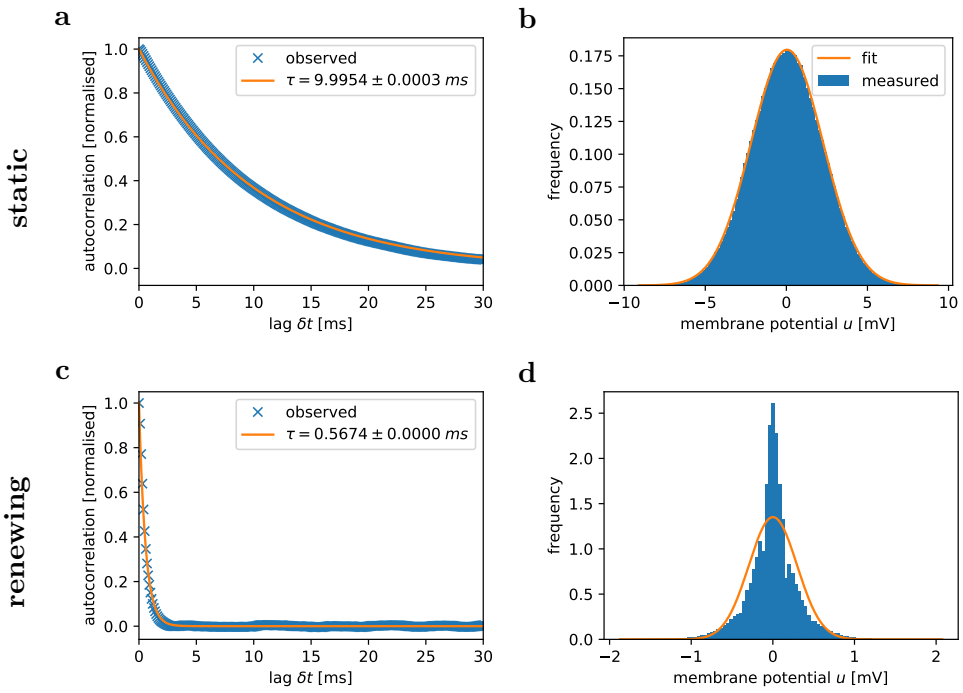

Figure 3.3.: **Autocorrelation of the membrane potential: a** Autocorrelation function of the (free) membrane potential evolution under Poisson noise with static synaptic connections. Observed values (blue crosses) and the fitted exponential, resulting in the expected decay time constant $\tau = 10\,\text{ms} = \tau_{\text{syn}}$ (orange). **b** Observed (free) membrane potential distribution from a. **c** Same as a but with renewing synapses. **d** Same as b but with renewing synapses.

(HCS) found in cortical neurons). In the general case the autocorrelation time scale would be formed by the larger of the two time constants $\tau = \max(\tau_{\text{m}}, \tau_{\text{syn}})$.

Without the short-term plasticity mechanism introduced in Section 3.1.1 we can see that the simulation results agree nearly perfectly with these predictions (Fig. 3.3a and b). If we employ TSO both the autocorrelation function as well as the membrane potential distribution change significantly (Fig. 3.3c and d). It is now significantly more peaked and decidedly non-Gaussian. In order to reach the tail values, i.e., larger deviations from $\bar{u}$, multiple PSPs need to add up. The TSO mechanism makes multiple input spikes *from the same synapse* sub-additive and therefore more extreme values of $u - \bar{u}$ are less likely to be reached. This has the effect of significantly reducing the autocorrelation time constant to 0.56 ms, which remains significantly higher than the membrane time constant $\tau_{\text{m}} = 0.1\,\text{ms}$.

We use static synapses for the noise input as, from a modeling perspective, we want the Gaussian shape of the distribution. This also makes sense from a biological perspective,

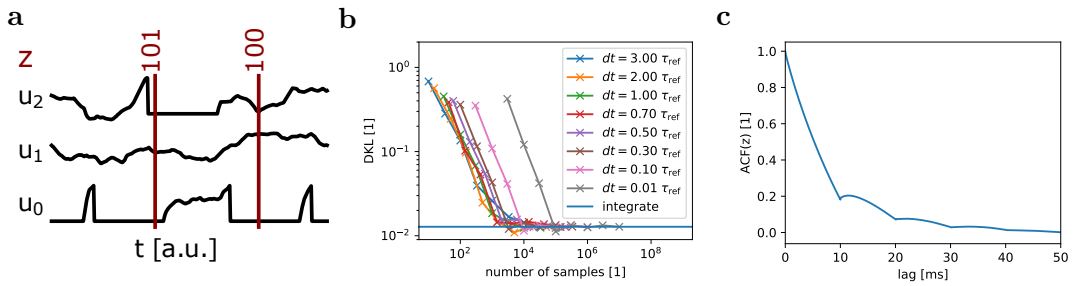

Figure 3.4.: **Different state generation dts: a** Exemplary membrane traces and state assignments. The choice of state assignment time deltas dt is bounded from below by the simulation time step. In a continuous system this corresponds to an integral over indicator functions for the different states over the experiment time. Figure adapted from [Dold et al., 2019]. **b** Sampling performance, as measured by the DKL, for different state assignment time deltas. For small dt the same performance requires significantly more samples due to their correlated nature. Above 2 samples per refractory time this degradation is observable, this limit corresponds to the Nyquist condition [Nyquist, 1928], with the correlation length being given by $\tau_{\text{ref}} = \tau_{\text{syn}}$. **c** Autocorrelation function of the state variable $z(t)$. The kinks at multiples of $\tau_{\text{ref}} = 10\,\text{ms}$ are due to a connected neuron firing regularly ($\langle z_2 \rangle \approx 1$). Aside from these the decay corresponds to the autocorrelation of the membrane time course $u(t)$ (cf. Fig. 3.3a).

as we use the high-frequency Poisson sources as a stand-in for a plethora of pre-synaptic partner neurons with low activity and low connectivity strength. As these all arrive at *different* biological synapses each input spike finds a "fresh" synapse. It is only the *sum* of all the pre-synaptic input that leads to the high-frequency Poisson stimulus. As such the implementation via a static synapse is appropriate. We will discuss the characteristics of the noise sources in more detail in Section 3.2.

State assignment

So far we have continued to poke holes into the assumption of the state definition introduced in Section 2.2.4. From experimental evidence we know that the assignment is valid, in the sense that we can use this model to actually train a sampling network towards a given target distribution [Petrovici et al., 2015, 2016, 2017a, Leng et al., 2018, Kungl et al., 2019, Dold et al., 2019]. Now we turn towards the question: How does our *translation* of the *observed* dynamical behavior of the neurons, i.e., their spike trains, into our *interpretation* of their states $z \in \{0, 1\}$ affect the resulting distribution.

In Fig. 3.4a we see exemplary membrane traces with the associated state assignments at two points in time. It is the collection of these states that forms our sampled distribution. Formally we choose a discrete set of times t_i (red vertical lines) within the

simulation time T_{sim} at which we want to assign a state $z \in \{0, 1\}$ to all neurons according to their refractory state. How we choose this temporal lattice is a free parameter. It only affects our *interpretation* but not the underlying *dynamics* of the system. In the extreme case we do this assignment at each possible spike time t_{spk}, which in numerical simulations corresponds to the simulator time step. Without this numerical complication it would also be possible to take the integral limit where we assign a new state the moment a new spike occurs and integrate the time the system stays within each possible state in order to determine the distribution.

The question that we should now ask is: Does our *choice* of the time spacing dt matter? Fig. 3.4b gives an affirmative answer. There, we show the DKL of the sampled distribution after a given number of *samples* for different time spacings dt. The left-most cross on each line corresponds to the same simulation time $T = 30\,\text{ms}$, the second cross to $T = 100\,\text{ms}$ and so forth. We see that the number of samples required to achieve a given DKL value increases for short $dt < \frac{\tau_{\text{ref}}}{2}$. Conversely the required simulation time increases (number of points from the left) for large dt.

Neither effect is surprising once we think about it for a moment: On the large-dt side it is obvious: If we wait too long we waste simulation time as we already have a new sample which we refuse to count. Once the autocorrelation between the states becomes negligible further waiting does not increase the independence. The autocorrelation function of the state variable z is shown in Fig. 3.4c. The kinks around multiples of the refractory period $\tau_{\text{ref}} = 10\,\text{ms}$ are due to a high-bias neuron in the network which is bursting for most of the simulation and thereby induces a significant autocorrelation. Aside from this oddity the autocorrelation is similar to the one on the level of the membrane potential time course $u(t)$ (cf. Fig. 3.3).

The more interesting question is the low-dt side, which (at least in the physical system, cf. Chapter 5 or in biology) does not increase *runtime*. Why do we need more samples here? The answer lies in the Nyquist (or Shannon-Nyquist) theorem [Nyquist, 1928], which states that in order to not loose any information in an optical system one is required to perform two measurements per wavelength. The wavelength essentially gives the correlation scale of the system, below which one cannot distinguish point sources due to their overlapping effects. In our case this correlation scale is given by the time constants τ_{syn}, τ_{ref} and τ_{m}. The former two are chosen to be equal and much larger than the membrane time constant. Therefore a state assignment for time spacing $dt = \frac{\tau_{\text{ref}}}{2}$ is advantageous[4].

One should note here that, at least for simulations of LIF neurons, the dealing with the binary states is not a significant cost factor in terms of computational requirements. In other words, for software simulations the choice of dt is rather unimportant, as the simulation time scales with the length of the simulation and not the number of samples. However, with the accelerated hardware used in Chapter 5 the handling of the states becomes a significant cost (and if the hardware access is optimized, the dominant one). At this point choosing a factor of 2 larger dt actually saves this factor in wall clock time.

[4]This parameter is taken for all simulations done sufficiently late in the course of this thesis.

3.2. Where does the noise come from?

In Section 2.2.4 we discussed how the exposure to noise, in the form of Poisson-distributed spike trains, transforms a deterministic LIF neuron into a probabilistic sampler. At that point we did not discuss the origin and thereby the characteristics (correlation structure, frequency, etc.) of these spike trains. We already briefly hinted at the nature of the Poisson sources in the previous section, where we discussed that the noise synapses are static rather than renewing (cf. Section 3.1.3). Here, we will discuss the biological origin of such stochastic spike sources in our models. Afterwards, we present several ways of realizing sources that are compatible with the neuromorphic hardware we will later use in Chapter 5. We will also, briefly, go into the dynamical consequences of choosing different implementations.

The aim of the LIF sampling framework is to model and explain the brain's ability of performing Bayesian computations on a neuronal level. As we will see, the activity of most of these neurons can be modeled as noise. A typical neuron in the cortex has on the order of 10 000 pre-synaptic partners [Pakkenberg et al., 2003], i.e., a spike from each of these 10 000 neurons affects the post-synaptic neuron's membrane potential [Sporns et al., 2005]. From a modeling perspective this large fan-in represents a significant challenge: Not only would it require the simulation of several thousand neurons, we also need to choose a parameterization of all the neurons and all connections according to their biological values. Even though it is possible to simulate networks with connection densities approaching biologically observed values, these simulations take enormous resources and do not lend themselves to systematic study (see e.g. [Markram et al., 2015, van Albada et al., 2018, Einevoll et al., 2019]). While biological data is not precise enough to adequately constrain more than the mean and variance of neuron parameters, recently, such simulations have been shown to exhibit strong scaling, i.e., the simulation time does not scale with network size, as long as one scales the number of utilized compute nodes appropriately [Jordan et al., 2018]. As such, one is, at least in principle, not limited by compute time[5].

On the other hand, the distribution of observed synaptic connection strength is dominated by many, very weak synapses [Barbour et al., 2007]. It is unlikely that the individual contribution of those meaningfully impacts the spiking behavior of the neuron. Even if there is no clear separation between a meaningful and a meaningless synaptic input, when ordering the synaptic connections by strength, there should be at least some cut-off beyond which the individual contribution becomes irrelevant to model. In order to not loose the aggregate contribution of these small-weight connections we can replace the individual neurons with their individual firing history by a random spike source with a compatible weight and the correct total firing rate.

If we accept the argument that most *meaningful* information is concentrated in few (comparatively strong or weaker, but synchronized) inputs, then the vast majority of the remaining inputs act essentially as a "heat bath" that introduces noise on the mem-

[5] However, the number of supercomputers available to simulate networks of several million spiking neurons is limited.

brane voltage (see also discussion in Section 2.2.4 and Section 4.1). At this point we can describe these two parts of the network separately: With one part we model the information processing network in detail, using as few neurons and synapses as possible, thereby limiting the required computational resources. In the other part we represent all the other, weakly connected, neurons as stochastic background spike sources. In this section we discuss (efficient) implementations both in software as well as under the constraints of the BrainScaleS hardware platforms (cf. Chapter 5) and the resulting characteristics of the noise in terms of the autocorrelation function of the membrane potential of the receiving neuron.

3.2.1. Poisson sources

The canonical way to represent *abstracted-away neurons* is via Poisson sources [Gerstner and Kistler, 2002b]. In our specific sampling network we utilize two per sampling neuron: One for all excitatory and one for all inhibitory pre-synaptic partners.

Definition 3.2.1. *Poisson source*
A Poisson source of rate r is a pre-synaptic partner in a network of spiking neurons that generates a spike train where the spike times are distributed such that the probability of having n spikes in an arbitrary time interval Δt follows a Poisson distribution

$$P(n \text{ spikes in } \Delta t) = e^{-r\Delta t}\frac{(r\Delta t)^n}{n!} \tag{3.14}$$

with the mean firing rate r.

This distribution has the property that the variance of the spike count equals the mean of the spike count

$$\langle n \rangle = \mathrm{Var}[n] = r\Delta t \tag{3.15}$$

leading to a Fano factor[6]

$$F = \frac{\mathrm{Var}[n]}{\langle n \rangle} = 1. \tag{3.16}$$

This kind of distribution is typically implemented via its waiting time between two successive spikes which follows an exponential distribution

$$P(\text{next spike after } \Delta t) = \exp(-r\Delta t) \tag{3.17}$$

leading to the cumulative probability of having a spike within Δt of

$$P(\text{next spike before } \Delta t) = 1 - \exp(-r\Delta t) \tag{3.18}$$

which is zero for $\Delta t = 0$ and converges to one for $\Delta t \to \infty$. This is essentially a Hazard process [Miller Jr, 2011] where the elicitation of a spike is the end of the process.

[6] Side remark: Comparing variances and means is in general problematic as they have different units. It works out here as we are comparing mean and variances of numbers, which are themselves unit-less.

We can calculate the probability density of a second spike occurring at precisely $t_{\text{spk}} + \Delta t$ as:

$$p(t_{\text{spk}} + \Delta t) = \frac{d}{dt}\left(1 - \exp(-r\Delta t)\right) = re^{-r\Delta t} \tag{3.19}$$

from which we can calculate the mean of the inter-spike interval (ISI), the time between two consecutive spikes, as:

$$\langle ISI \rangle = \int_0^\infty \Delta t \; p(\Delta t) dt = \frac{1}{r} \tag{3.20}$$

and the variance of the ISI distribution as

$$\text{Var}[ISI] = \int_0^\infty (\Delta t)^2 p(\Delta t) dt = \frac{1}{r}, \tag{3.21}$$

resulting in a coefficient of variance[7] of

$$C_V = \frac{\sqrt{\text{Var}\, ISI}}{\langle ISI \rangle} = 1. \tag{3.22}$$

These characteristic numbers, the Fano factor $F = 1$ and the coefficient of variance $C_V = 1$, are not unique to Poisson distributed spike trains – although they are sometimes used as a quick check for irregular spiking activity [Grübl and Baumbach, 2017].

Technical implementation and pitfalls

We can generate a Poisson spike train by deciding for each sufficiently small δt (e.g., the simulation time step) whether the source should generate a spike according to

$$P(\text{spike in } \delta t) \approx r\delta t. \tag{3.23}$$

In practice, a Poisson spike source is an object already provided by the simulation backend (NEST [Peyser et al., 2017]). While this prevents us from introducing bugs ourselves we still need to be aware of its limitations:

1. Spikes can only be produced at multiples of the simulation time step dt.

   For rates $r \ll \frac{1}{dt}$ this does not introduce any limitations[8], as there will almost never be a spike and all properties discussed above hold, although the underlying time is discretized. However, for higher rates, definitely for $r > \frac{1}{dt}$ but also approaching this value, the stochastic process changes. There will almost always be a spike in each time bin and, depending on the implementation, we either get a regular spike train – if the implementation allows at most one spike per simulation step – or the noise process changes from "*Is there a single spike in the time bin* Δt" to "*How many spikes are there in the time bin* Δt*?*".

[7] This is a much more natural comparison (as compared to the Fano factor) as standard deviations and means always have the same units.

[8] Or, more precisely, no additional limitations besides standard numerics.

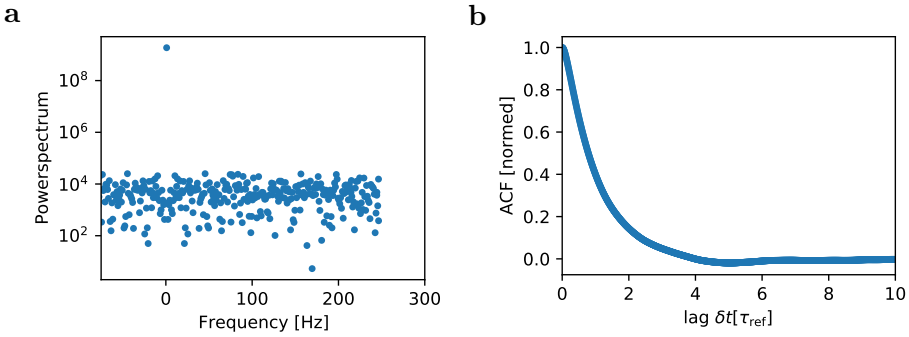

Figure 3.5.: **Poisson noise: a** Power spectrum of a Poisson spike train with a binning of $2\,\text{ms} = 0.2\tau_{\text{ref}}$ for the time series. For visibility reasons we plot the max filtered signal with a width of 200. **b** Same as Fig. 3.3a. The autocorrelation function of the membrane potential is a delta peak convolved with the PSP kernel. All variations beyond $4\tau_{\text{ref}}$ are due to finite time statistics. Simulation description can be found in Appendix B.2.1.

2. Ornstein-Uhlenbeck process assumes adiabatic changes.

   The OU process assumes an infinitely fast sampling of the Wiener process W (cf. Eq. (2.66)) with infinitely small contributions of each sample. In practice we are unable to reach this as we use finite time steps in the numerical simulation. This limits the self-similarity of the Wiener process on short time scales. For the most part, as the membrane time constant acts as a filter on the input and thereby remove the high-frequency components of W, this does not impact the observable behavior. However, for very small noise frequencies $r \ll \frac{1}{\tau_{\text{syn}}}$, the membrane potential u of the neuron "forgets" its input faster than new spikes arrive. In our modeling we expect a stochastic walk of u such that it implements an OU process. For this to happen, superpositions of single inputs must be observable on the membrane potential. In practice, this requires a Poisson rate of $r > \frac{3}{\tau_{\text{syn}}}$. Below this noise level the membrane potential evolution shows deterministic behavior after a single input spike. Only the times of the deflection generate a non-zero variance.

For our discussion it is only important to be aware that these limitations exist and that we will stay far away from those limits. We are not interested in the properties of the Poisson process itself but rather in the effect of the generated spikes on the target neuron. In order to gain an understanding of this effect we take a look at the autocorrelation function of the membrane potential of the receiving neuron (cf. Eq. (3.13)). We already hinted at this in Section 2.2.4 and in Section 3.1.3, where we noted that the exponential decay of the synaptic input triggered by the noise spikes implements an OU process on the membrane potential.

The autocorrelation function of the membrane potential is generated by the auto-

correlation of the stochastic process that generates the noise spikes convolved with the PSP kernel of the neuron [Petrovici, 2015]. In Fig. 3.5a we see the power spectrum of a Poisson spike train with $r = 1\,\mathrm{kHz}$ with a binning of $t_{\mathrm{bin}} = 2\,\mathrm{ms} = 0.2\tau_{\mathrm{ref}}$. The binning is necessary as spike trains are a series of delta functions for which an autocorrelation function is not well-defined. The choice of the binning influences the absolute level of the power spectrum but not the frequency distribution (aside from cutting off high-frequency parts due to their finite size). As expected, the power spectrum essentially consists of a δ-function, showing that the spike times are completely independent from each other. The noise level of the surrounding parts is only due to the finite simulation time and the contrast increases for longer simulations.

The autocorrelation function of a membrane potential under two (one excitatory, one inhibitory) such Poisson sources is shown in Fig. 3.5b (showing the same data as Fig. 3.3a). We see that the autocorrelation time scale is $\tau_{\mathrm{syn}} = \tau_{\mathrm{ref}}$ and all long-term variations with a lag of $\delta t > 4\tau_{\mathrm{ref}}$ are, again, only due to the finite simulation time. For such a configuration the theoretical treatment from Section 2.2.4 is possible. However, the simplifications we made there do have significant effects which we will discuss in Section 3.3.

This implementation requires us to feed each neuron with its (two) *private* spike train(s) of rate r. Remember: The noise sources represent the majority of the fan-in and therefore r is typically chosen between 1 kHz to 10 kHz. In software simulations this is a manageable challenge. We can generate the noise spikes on the fly and with relatively little compute power[9]. At most, we run into the problem of providing *consistent* random numbers to multiple compute nodes at which point seeding the processes becomes a non-trivial task. For the hardware platforms that we will discuss in Chapter 5 providing sufficient *external* stimulus is a tougher problem. In the naive implementation we would need to generate the spike trains on the host computer (either before the experiment or on the fly) and we then have to transfer these spikes *in time* to the emulating system. In practice this quickly overwhelms the available bandwidth to the accelerated systems we are using, forcing us to look for other options. Here we will only discuss alternative solutions and defer the concrete hardware limitations to the respective later sections in Chapter 5.

We should, however, make a fundamental point explicit: If the single neurons were to depend on external noise sources only, every compute substrate implementing this kind of computation would be limited in scope. The amount of external input that can be provided to the system necessarily is constrained by the surface it shares with its surrounding[10]. As such the noise generation has to happen locally (i.e. within the system) and most neuromorphic platforms provide some form of on-chip noise generation in order to not be constrained by external bandwidth. For us here the practical workarounds are more interesting:

[9] Random number generation takes only about one clock cycle. The majority of the cost comes from the unpredictability and as such potential branch mis-predictions in the CPU.

[10] This holds for any kind of external dependency, i.e., also the power consumption. As long as it has to be provided from the *outside*.

3.2.2. On-chip sources

For the HICANNv4 chips of BrainScaleS-1 (cf. Section 5.1) – with its speed-up of 1×10^4 over biological real-time – the limit is roughly enough noise for 2 neurons per single chip (cf. Section 5.1 and [Kungl, 2016, Kungl et al., 2019]). For the HICANN-Xv1 chips of the BrainScaleS-2 system the effective external bandwidth is higher due to its reduced selected speed-up of 1×10^3. Both systems feature on-chip spike generators that can be configured to generate Poisson-like spike trains. As we will only use the ones on BrainScaleS-2 in Section 5.2, we will restrict the discussion to their implementation.

The dynamical time scales on BrainScaleS-2 are such that they provide a speedup of 1×10^3 resulting in:

$$\tau_{\text{ref}} \approx \tau_{\text{syn}} \approx 10\,\mu\text{s}. \tag{3.24}$$

The results in Fig. 3.6 are therefore rescaled by this factor such that the resulting frequency scale of the power spectrum (cf. Fig. 3.6b) is given in kHz. For the autocorrelation of the membrane potential in Fig. 3.6c this change of time scale is absorbed into the definition of τ_{ref}.

The on-chip source is a 32-bit pseudo random number generator (PRNG) (cf. Figure 3.6a, [Schemmel et al., 2020]). It consists of a series of coupled linear-feedback shift register (LSFR) which produce a total of 16-bit entropy per cycle [Schemmel, 2020]. Starting from some initial configuration (also called the seed) an LSFR *deterministically* updates its internal state by shifting all bits by one and generating the now unassigned bit via some function of the other bits. A typical implementation is a sequence of XOR operations on the output bit (the one that got shifted out of the LSFR state) and some arbitrary, but fixed, bits. These are called the *taps* of the LSFR and they define the properties of the resulting pseudo random number (PRN).

Any PRNG with n bits of internal state can at most generate $2^n - 1$ independent samples before the internal state recurs and the sequence of output bits repeats. This gives us a sequence length of a bit more than a million "random" bits[11].

In practice, we only require a random number to decide whether a spike should occur in a time bin or not and we can choose these time bins to be more coarse grained than the clock frequency. We will choose a noise period of about 50 clock cycles[12]. This means the PRNG generates an 16-bit random number with a frequency of 5 MHz or once every $0.2\,\mu\text{s} \approx 0.02\tau_{\text{ref}}$, which is comparable to the chosen resolution of the software simulators of $dt = 0.1\,\text{ms}$. It generates a spike by comparing an 8-bit random number with an 8-bit user-defined rate and only creates a spike packet if the PRN is larger than the rate value [Schemmel et al., 2020].

The other 8-bit of pseudo-randomness provide an additional source of randomness: The spike packets on the BrainScaleS platforms contain a tuple of source ID and time.

[11] This is a simplification, as an LSFR is not cryptographically secure and one can reconstruct the internal state from parts of the output sequence. The coupling between multiple LSFR implements a cryptographically secure PRNG [Schemmel, 2020]. On our level of interest this distinction does not matter.

[12] We choose 53 in order to avoid potentially hitting a bug when using non-prime numbers that shortens the cycle length.

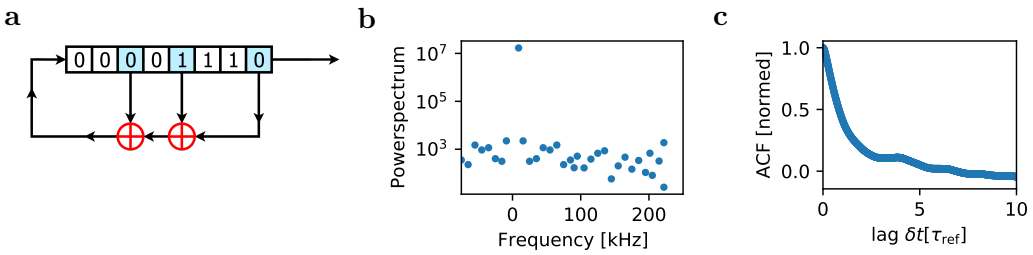

Figure 3.6.: **LSFR noise: a** Schematic of an 8-bit LSFR with taps at the 3rd and 5th position. For each update the new bit (left) is generated from the final bit (right) sequentially xor'd with the two taps and all bits are shifted by one. The PRNG sources on BrainScaleS-2 use multiple coupled LSFRs to provide 16-bit entropy per update cycle. Figure taken from [Wikimedia, 2016b]. **b** Power spectrum of the ISI distribution of the spike train resulting from $k = 5$ IDs of the PRNG source of BrainScaleS-2. Each ID provides a spike frequency of $r = 80\,\text{kHz}$. For visibility reasons we show the maximum filter over 200 points of the power spectrum with a binning of $2\,\mu\text{s}$. **c** Auto-correlation function of the membrane of a neuron under LSFR input from **b**.

While the time is purely for bookkeeping purposes (and not used on-chip, cf. Section 5.2) the source ID is important for the routing. We set 5 bits of this ID randomly and use this feature to generate 32 sources from a single PRNG circuit. These sources are not independent, i.e., if we choose the rate to be close to the maximum and the PRNG generates a spike every time it gets updated, then each spike train will still look random. However, taking the sum of all 32 ones would result in a periodic spike train[13] with frequency $\nu = 5\,\text{MHz}$. In practice, each neuron will be connected to a random subset of 5 of the 32 generated spike trains from the used two noise generators. Each ID generates 80 kHz of noise, resulting in a total of $400\,\text{kHz} \approx \frac{4}{\tau_{\text{ref}}}$ input per neuron. As this noise is now correlated between different neurons, we implicitly use that shared noise correlations can be trained away [Bytschok et al., 2017, Bytschok, 2017]. We use one of the noise generators exclusively for excitatory sources and one exclusively for inhibitory sources.

We acquired the noise spike trains by a single execution of the sampling framework for the new BrainScaleS-2 platform (cf. Appendix C.2) where we additionally read out the on-chip generated spike times. The thus-acquired spike trains we then injected in a simulated neuron in NEST [Peyser et al., 2017]. This separates the effect of the on-chip spike sources from effects originating in the, potentially imprecise, parametrization of the analog neuron circuit. The power spectrum (Fig. 3.6b) looks as expected, i.e. we observe a delta distribution, where the surrounding noise level is consistent with being generated by the limited experiment duration. For visibility reasons we again show the

[13] This is not a unique feature of the on-chip generators, but rather inherent to the finite time resolution. See also the discussion for pitfalls in the high-frequency limit in Section 3.2.1.

maximum filter over 200 bins. The autocorrelation function of the membrane potential (Fig. 3.6c) shows qualitatively the same behavior as expected but does saturate at about 0.1 between $3 - 4\tau_{\text{ref}}$. This excess correlations diminishes after $\approx 7\tau_{\text{ref}}$. The most likely reason for this is the finite experiment duration.

3.2.3. Random network

In case of BrainScaleS-1, the on-chip sources were not available and feeding in external noise not feasible for larger networks due to bandwidth constraints (cf. Section 5.1). As discussed previously, the noise is actually an abstraction for a sparsely connected network. The size of BrainScaleS-1 allows us to implement such a network and use their output spikes in lieu of a noise source. The question becomes: What is a network that is a) simple to implement and b) produces sufficiently Poisson-like spike trains?

Generating self-sustaining network dynamics in a recurrent neural network is, at least in general, a non-trivial task [Destexhe, 2009]. Especially networks with both excitatory and inhibitory connections between leak-lower-than-threshold $V_{\text{l}} < V_{\text{thresh}}$ neurons tend to either an epileptic state, where the activity diverges or to a quiescent state where there is no activity at all [Brunel, 2000]. Neither of those states produces the stochastic behavior we require for a noise reservoir. If we allow for $V_{\text{l}} > V_{\text{thresh}}$ neurons, we can find that a simple network of inhibitorily connected neurons produces adequate behavior. While the leak-over-threshold parametrization ensures non-zero network activity the inhibitory connections prevent the neurons from firing completely regularly. In combination with the inhomogeneous parametrization, due to the circuit-to-circuit variations on the various BrainScaleS chips (cf. Chapter 5), this leads to variations in the ISI distributions.

There are two degrees of freedom in a network of n such LIF neurons (cf. Fig. 3.7a): We can choose the number of pre-synaptically connected neurons k and we can choose the strength of the connection W_{inh}. In the power spectrum of the noise spike trains (Fig. 3.7b, again with binning of $dt = 2\,\text{ms} = 0.2\tau_{\text{ref}}$) we can still see the peaks introduced by the eigenfrequency of the leak-over-threshold neurons, which is only slightly above $\frac{1}{\tau_{\text{ref}}}$. The side peaks of the power spectrum show up in the ACF of the membrane potential (cf. Fig. 3.7c) which also corresponds to the autocorrelation function of the state of the neuron. Different values of k and W influence the frequency and amplitudes of the side peaks (different colors in Fig. 3.7c). Higher k and stronger inhibition are associated with lower peak amplitudes. The frequency of the side peaks stays at intervals of τ_{ref}, which is due to the potential burst of single neurons (cf. Section 3.3).

This idea of using inhibitory random networks as noise sources was first demonstrated on the Spikey-chip, however, without functionally using the generated noise[14] [Pfeil et al., 2016]. In [Kungl et al., 2019] such a setup is used on the BrainScaleS-1 system in order to implement LIF sampling (cf. Section 5.1). Jordan et al. [2019] showed that also more biologically plausible random networks can be used as sources of stochasticity [Jordan et al., 2019]. As we will see in Chapter 5 it turns out that the noise provisioning is not a

[14]This is now part of the current FP lab course F09/10 [Grübl and Baumbach, 2017].

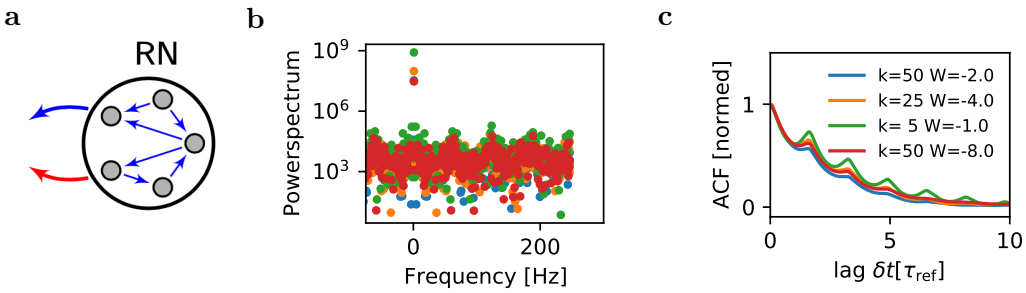

Figure 3.7.: **RN noise:** **a** Schematic representation of a random network (RN) noise source. Figure taken from [Kungl et al., 2019]. **b** Power spectrum of the spike train from 20 neurons of a simulated random network with a total of $n = 400$ neurons, each of which has $k = 50$ pre-synaptic partners and an inhibitory synaptic weight $W_{\text{inh}} = 2.0\,\text{nA}$. We see a clear structure in the power spectrum with peaks around the firing rate of a free neuron. These would further wash out when we would use some variation in the neuron parameters instead of simply using inhomogeneous initial conditions. **c** Autocorrelation function of a neuron with an excitatory and an inhibitory source from **b**. The peaks in the power spectrum translate to the side peaks of the autocorrelation function and we again see the PSP shape with the exponential decay of $\tau_{\text{syn}} = 10\,\text{ms}$. Simulation descriptions can be found in Appendix B.2.2.

significant limitation of the sampling setup as we reach similar performance to software simulations[15].

Other BMs and shared sources

Ultimately there is no evidence that the human body sustains parts of the brain that do not themselves also have some functional implications. As such it is unlikely that there exist neurons whose sole purpose it is to provide uncorrelated input to the functional neurons. Therefore, the objective is to have a model which uses the output of other functional networks as stochastic input to sustain its stochastic computations. This idea was developed and reported in [Dold et al., 2019]. It forms a network of spiking sampling networks, where each single network samples from one particular Boltzmann distribution and a sparse connectivity between these networks sustains the stochastic computation within these. At this point we necessarily use the equivalence between correlated noise sources and synaptic connections [Bytschok et al., 2017] in order to compensate for the, now necessarily, correlated spike trains from the "noise" neurons.

In a way this closes the circle which we opened when we replaced all the neurons we could not simulate explicitly into Poisson sources. There we removed neurons in favor

[15] Which says more about the other limitations of our understanding of LIF sampling, than about the applicability of colored noise in general.

for a) a better performance and b) a simpler mathematical description. Dold et al. [2019] allows us to utilize a network of functional networks. As we will not be returning to these noise implementations we leave it by this very brief introduction and refer the interested reader to the original publications [Dold et al., 2019, Bytschok et al., 2017] and associated PhD theses [Dold, 2020, Bytschok, 2017].

3.3. A Markovian description of LIF sampling

This section presents work done in close collaboration with Nico Gürtler as part of his master thesis [Gürtler, 2018] which I had the privilege to supervise.

In this section we will sketch the LIF Markov model (LMM), a stochastic model for the spike response of LIF neurons under Poisson stimulus, that Nico Gürtler developed as part of his master thesis [Gürtler, 2018]. For most of the mathematical derivation and proofs we refer the interested reader to the original manuscript as we will focus on giving a high-level description of the dynamics of single neurons within the LIF sampling framework. In Section 2.2.4 we showed that the activation function of an LIF neuron, under noise input, resembles a sigmoid and we took that as a motivation to the LIF dynamics to sampling from a Boltzmann distribution. Here, we will first, in Section 3.3.1, go into more detail on the behavior of LIF neurons in the absence of network input and how this depends on the mean of the membrane potential distribution. We then describe the spike response to a single synaptic input, both excitatory and inhibitory, in Section 3.3.2 before showing that this framework is also able to deal with multiple inputs (Section 3.3.3).

3.3.1. LIF activation function - revisited

Before we come to the description of the actual network interactions we need to derive a stochastic formulation of the LIF neuron under noise stimulus. The transition from an, in principle deterministic, system to a stochastic one is always based on information that is not taken into account. In our case we generate the LMM by restricting us to use only a priori available information (e.g. neuron and noise parameters) and the output spike times of the neurons. In particular, we hide the precise spike times of the noise sources[16] or the resulting membrane potential u. The two latter statements are equivalent as one can reconstruct the membrane potential time course from the input spike train and vice versa. Nevertheless, it is illustrative to explicitly enumerate the non-accessible information.

Over the rest of this section we will first derive a formulation of the membrane potential density $f(u;t)$[17] that is informed only by the network spike times $\{t_{\text{spk}}\}$ (and the neuron parameters). This corresponds to the distribution of an ensemble of neurons which share the same network input but are subject to different incarnations of the noise spike trains. We will then see how network input shifts this distribution $f(u)$ around and, in turn, elicits changes in the spike response rate ρ. In the end we will find that this density-

[16] Taking the precise spike times out of consideration is appropriate, as we *postulated* that these are not relevant and serve as an abstraction for functionally unimportant inputs.

[17] This is the ρ from Section 2.2.4 which we will later use for the spike response function in order to be consistent with [Gürtler, 2018].

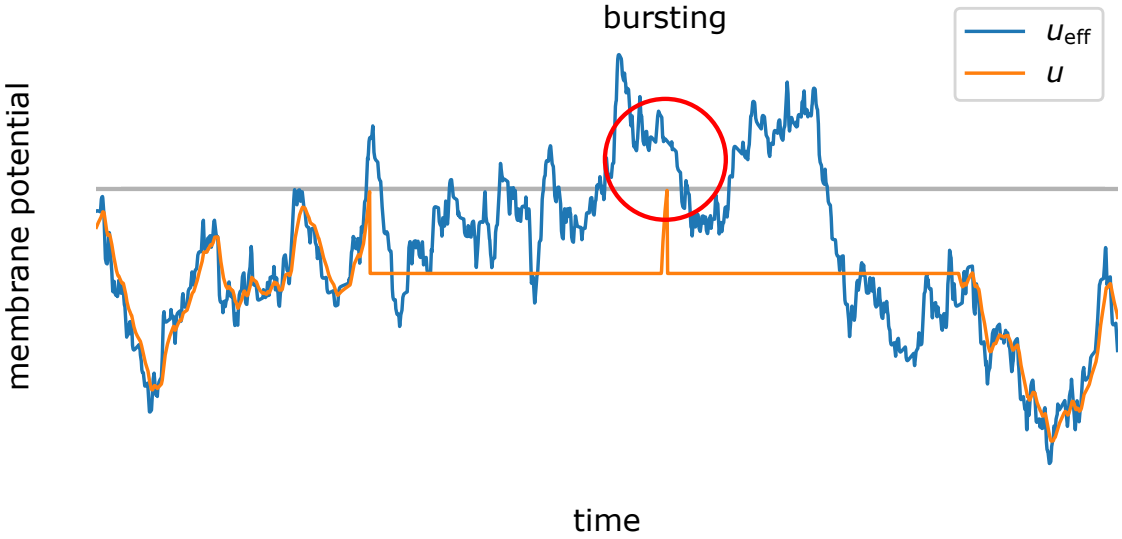

Figure 3.8.: **Evolution of membrane potential:** The orange line shows the realized membrane potential u of an LIF neuron under Poisson noise. u is a low-pass filtered version of the effective potential u^{eff} (blue), the instantaneous target value imprinted by the synaptic input. Image taken from [Gürtler, 2018].

based description models the dynamics of LIF neurons much more faithfully than the description by [Buesing et al., 2011].

Fig. 3.8 depicts a situation were there is no network input to the neuron. Similar to Fig. 2.9, we again see the actual realized membrane potential u in orange and the instantaneous target voltage u^{eff} in blue. This is again for a single current-based LIF neuron under Poissonian noise input. In this section the input frequency is increased to 30 kHz and the synaptic time constant of the *noise* input reduced to $\tau_{\text{syn}}^{\text{noise}} = 3\,\text{ms}$. By choosing a shorter synaptic time constant τ_{syn}, we effectively reduce the autocorrelation of the instantaneous target for the membrane u^{eff} (cf. Sections 3.1.3 and 3.2). In particular, the reduced correlation between the initial spike time t_{spk} and the end of the first refractory period $t_{\text{spk}} + \tau_{\text{ref}}$ will be helpful for the stochastic modeling.

The membrane potential u (orange) is a low-pass filtered (with time constant $\tau_{\text{m}} \ll \tau_{\text{syn}}$) version of the effective membrane potential u^{eff} as long as $u^{\text{eff}} \ll V_{\text{thresh}}$. Note, due to this filtering it can happen that the effective membrane potential rises above the threshold $u^{\text{eff}} > V_{\text{thresh}}$ without the actual membrane potential u following to supra-threshold values. This happens only if u^{eff} quickly returns to sub-threshold levels. In this case no spike will be elicited (cf. slightly left of the first spike in Fig. 3.8). All following arguments assume a quasi-instantaneous following of u to u^{eff}, this would imply $\frac{\tau_{\text{m}}}{\tau_{\text{syn}}} \to 0$ in our current-based LIF neuron.

We reduced our available information to the scope that also the post-synaptic partner neurons have access to: Namely the set of output spike times t_{spk} of the original neuron. The question now becomes: What do we still know about the effective membrane potential u^{eff}? As we are only allowed to describe its evolution stochastically, we will

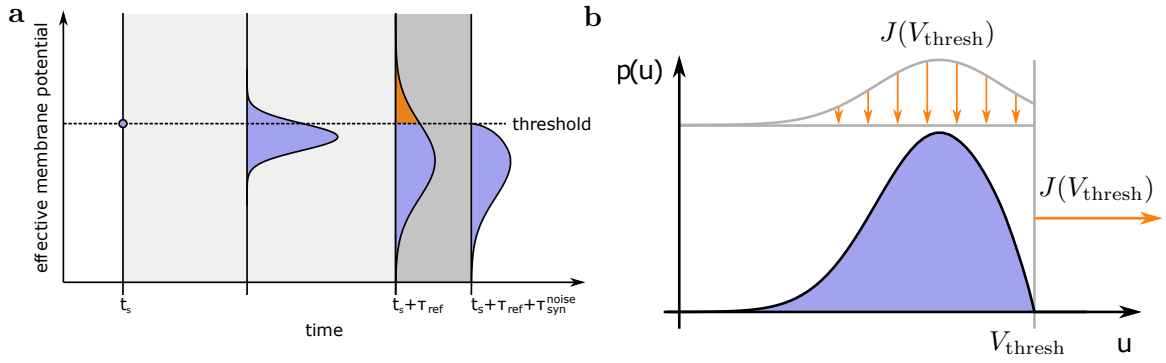

Figure 3.9.: **Evolution of membrane potential distribution: a** Schematic representation of the time evolution of the membrane potential distribution after a spike. At spike time t_{spk} the membrane potential u is known to be V_{thresh}. During the refractory time the u-dynamic is overridden by the reset mechanism. Due to the continued noise input the state of the synaptic input continues to evolve the effective membrane potential u^{eff}. Its evolution follows an Ornstein-Uhlenbeck (OU) process with the initial condition of $f(u) = \delta(u - V_{\text{thresh}})$. At the end of the refractory period only the OU-generated distribution is known. The part above the threshold value V_{thresh} (orange) leads to an immediate spike and generates the bursting. Beyond the end of the refractory time, only the sub-threshold distribution remains. **b** Illustration of the probability flow in the stationary distribution of non-refractory neurons. The flux $J(V_{\text{thresh}})$ over the threshold corresponds to the spiking neurons. The source term (distribution above) models neurons at the end of their last refractory period. As the mass in the stationary distribution is conserved, the integral of the source term has to be $J(V_{\text{thresh}})$. The source distribution is not exactly the final below-threshold distribution from **a** as it also contains the final u^{eff} distribution of neurons with more than one consecutive spike (see main text). Images taken from [Gürtler, 2018].

only talk about it in terms of its membrane potential distribution function $f(u^{\text{eff}})$. The moment in which we can still have certainty of the membrane potential is when there is a (first) spike elicited by the neuron. For every spike time t_{spk} we do know

$$u = V_{\text{thresh}}. \tag{3.25}$$

The special thing about a *first* spike is that we know:

$$u \approx u^{\text{eff}} \tag{3.26}$$

since the real membrane potential u is the τ_{m}-filtered version of u^{eff} whenever the neuron is not refractory and $\tau_{\text{m}} \to 0$ in our parameterization. In the following we will drop the distinction between u and u^{eff} for notational simplicity, in the understanding that we are almost always talking about the effective one.

Formulating the last statement again in terms of membrane potential distributions gives the following for a first spike time t_{spk}:

$$f(u; t = t_{\text{spk}}) = \delta(u - V_{\text{thresh}}) \tag{3.27}$$

Together with the Fokker-Planck equation of the Ornstein-Uhlenbeck (OU) process Eq. (2.67) and the resulting the transition probability (restating Eq. (2.68)):

$$P(u, t \mid u', t') = \sqrt{\frac{\theta}{\pi\sigma^2\left(1 - e^{-2\theta\tau_{\text{ref}}}\right)}} \exp\left[-\frac{\theta}{\sigma^2}\frac{(u - u'e^{-\theta(t-t')})^2}{1 - e^{-2\theta(t-t')}}\right], \tag{3.28}$$

this allows us to calculate the time evolution of $f(u; t)$ for an arbitrary initial condition $f(u; t_0)$ as:

$$f(u; t) = \int_{-\infty}^{\infty} du' P(u, t \mid u', t_0) f(u'; t_0). \tag{3.29}$$

In Fig. 3.9a this evolution is shown stylistically.

An accurate prediction becomes more challenging for high-activity LIF neurons (cf. Section 2.2.4). This indicates that the membrane potential distribution $f(u; t_{\text{spk}} + \tau_{\text{ref}})$ around the end of the refractory period is of particular interest. Plugging $t = t_{\text{spk}} + \tau_{\text{ref}}$ into Eq. (3.29) together with Eq. (3.27) gives:

$$f(u; t_{\text{spk}} + \tau_{\text{ref}}) = \int_{-\infty}^{\infty} du' P(u, t_{\text{spk}} + \tau_{\text{ref}} \mid u', t_{\text{spk}}) \delta(u' - V_{\text{thresh}}) \tag{3.30}$$

$$= \sqrt{\frac{\theta}{\pi\sigma^2\left(1 - e^{-2\theta\tau_{\text{ref}}}\right)}} \exp\left[-\frac{\theta}{\sigma^2}\frac{\left(u - u'e^{-\theta\tau_{\text{ref}}}\right)^2}{1 - e^{-2\theta\tau_{\text{ref}}}}\right]. \tag{3.31}$$

At this point we do *not* know the precise value of u^{eff} anymore.

If and only if $u^{\text{eff}} > V_{\text{thresh}}$ the neuron will immediately fire again and enter a burst. The probability of such an immediate spike occurring is given by the supra-threshold part of Eq. (3.31) (cf. orange part of Fig. 3.9a at $t_{\text{spk}} + \tau_{\text{ref}}$):

$$P(\text{refire}) = P_1 = f(u > V_{\text{thresh}}; t_{\text{spk}} + \tau_{\text{ref}}) = \int_{V_{\text{thresh}}}^{\infty} f(u; t_{\text{spk}} + \tau_{\text{ref}}) du \tag{3.32}$$

There is no closed-form solution for $f(u > V_{\text{thresh}}; t)$ such that we are limited to this integral of integrals[18] formulation without further approximations. As we can see in Eq. (3.32), we do not know the exact value of the effective membrane potential u^{eff}. Therefore the initial distribution in Eq. (3.29) is no longer a simple δ-function and the equivalent of Eq. (3.32) for higher burst lengths becomes more involved.

The sub-threshold part of said distribution belongs to the neurons that do not fire immediately again. Before we continue with the sub-threshold neurons, let us trace the probability of having a third (or more spikes) in a burst further. We can calculate it in analogy to Eq. (3.32) by plugging the supra-threshold part of the probability distribution at the end of the first refractory period $f(u > V_{\text{thresh}}; t_{\text{spk}} + \tau_{\text{ref}})$ into Eq. (3.29):

$$f(u; t_{\text{spk}} + 2\tau_{\text{ref}}) = \int_{V_{\text{thresh}}}^{\infty} du' P(u, t_{\text{spk}} + 2\tau_{\text{ref}} \mid u', t_{\text{spk}} + \tau_{\text{ref}}) f(u'; t_{\text{spk}} + \tau_{\text{ref}}), \quad (3.33)$$

and again take the supra-threshold part of the resulting membrane potential distribution to result at the probability of a burst of length 3 or more:

$$P_3 = f(t_{\text{spk}} + 2\tau_{\text{ref}}; u > V_{\text{thresh}}) = \int_{V_{\text{thresh}}}^{\infty} f(u; t_{\text{spk}} + 2\tau_{\text{ref}}) du. \quad (3.34)$$

In the same manner we can iteratively calculate the probability of all higher burst lengths[19].

While we lack a closed-form formulation of these later $f(u; t)$ we can still gain some intuition from the picture in Fig. 3.9a: The choice of $\tau_{\text{syn}} \ll \tau_{\text{ref}}$ allows us to make the simplifying assumption that the *shape* of the membrane potential distribution at the end of the refractory time is essentially the Gaussian steady-state distribution (with mean and variances given by Eq. (2.71) and Eq. (2.72) respectively). This distribution $f(u; t_{\text{spk}} + 2\tau_{\text{ref}})$ is slightly different from the one at the end of the first refractory time $f(u; t_{\text{spk}} + \tau_{\text{ref}})$ as the former is generated from a delta source, which is farther away from the final Gaussian. This additional simplification means that the relative fraction of supra-threshold mass and thereby the probability to spike again:

$$p_k = f(u > V_{\text{thresh}}; t_{\text{spk}} + k\tau_{\text{ref}}) \quad (3.35)$$

is constant for all $k > 2$. As such the burst length distribution will be well approximated by an exponential decay with

$$P_n = P_1 p^{n-1} \quad (3.36)$$

with P_1 being the fraction of the supra-threshold membrane distribution at the end of the first refractory time $f(u; t_{\text{spk}} + \tau_{\text{ref}})$ and p the supra-threshold fraction at the end of the following distributions, with each $f(u; t_{\text{spk}} + n\tau_{\text{ref}})$ being normalized individually. Both of these constants are strongly dependent on the mean Eq. (2.71) and the variance Eq. (2.72) of the membrane potential distribution function. For larger τ_{syn}, and the

[18] $f(u)$ is, in general, also only known in integral form.

[19] We leave the exercise of writing down the equations to the interested reader and refer to Gürtler [2018] for inspiration.

ensuring larger autocorrelation time scales, we would have to treat more than just the membrane potential distribution at the end of the refractory period after the first spike explicitly.

So far we have only talked about the supra-threshold part of the distributions, which lead to immediate refiring and therefore bursts. Let us now turn to the opposite case of a non-firing neuron. In case of a very high threshold V_{thresh} this is the normal case and therefore corresponds to the steady-state distribution of the underlying OU process:

$$f(u) = \frac{1}{\sqrt{2\pi \operatorname{Var}[u]}} \exp\left\{ -\frac{1}{2} \left(\frac{u - \bar{u}}{\sqrt{\operatorname{Var}[u]}} \right)^2 \right\}, \tag{3.37}$$

where the variance $\operatorname{Var}[u]$ and mean $\bar{u}$ are given by Eq. (2.72) and Eq. (2.71), respectively. This distribution always describes the probability density for neurons that have not spiked for a long time (more than a few OU time constants τ_{syn}). We largely eliminated the transition period closely after a spike by having the refractory time be long with respect to the synaptic time constant.

Let's recap what we did so far:

1. We calculated the probability of a neuron to keep a burst active as the integral over the supra-threshold distribution.

2. We calculated the steady-state distribution of the underlying OU process.

What we actually want to know is the steady-state distribution function $f(u)$ integrated over the neuron's temporal evolution. Alternatively, we can perform this analysis in an ensemble formulation (Ergodic theorem): A schematic representation is shown as the blue distribution in Fig. 3.9b, where the flux of probability mass over the threshold V_{thresh} is given by $J(V_{\text{thresh}})$. As the neuron number is conserved, we know that the flux leaving the sub-threshold part of the distribution is also the flux returning to the population at the end of the refractory periods. This generates a source distribution $J(u)$ that is a weighted sum of

1. the sub-threshold part of the end-of-refractory time distribution of the first spikes $f(u; t_{\text{spk}} + \tau_{\text{ref}})$,

2. the sub-threshold part of the later membrane potential distributions $f(u; t_{\text{spk}} + n\tau_{\text{ref}})$,

the relative fraction of which we can estimate from the burst probabilities P_n, cf. the discussion above.

Gürtler [2018] empirically uses a fit to the following prototypical probability density function:

$$f(u) = C e^{-\frac{1}{2}\frac{(u-\mu)}{\sigma^2}} \left(1 - a e^{\frac{\alpha(1-a)(u - V_{\text{thresh}})}{\sigma}} \right) \left(1 - c e^{\frac{u - V_{\text{thresh}}}{A}} \right), \tag{3.38}$$

where A corresponds to the amplitude of a noise PSP, α is an appropriately chosen constant for the exponential cut-off at the threshold value V_{thresh}, C is a normalization

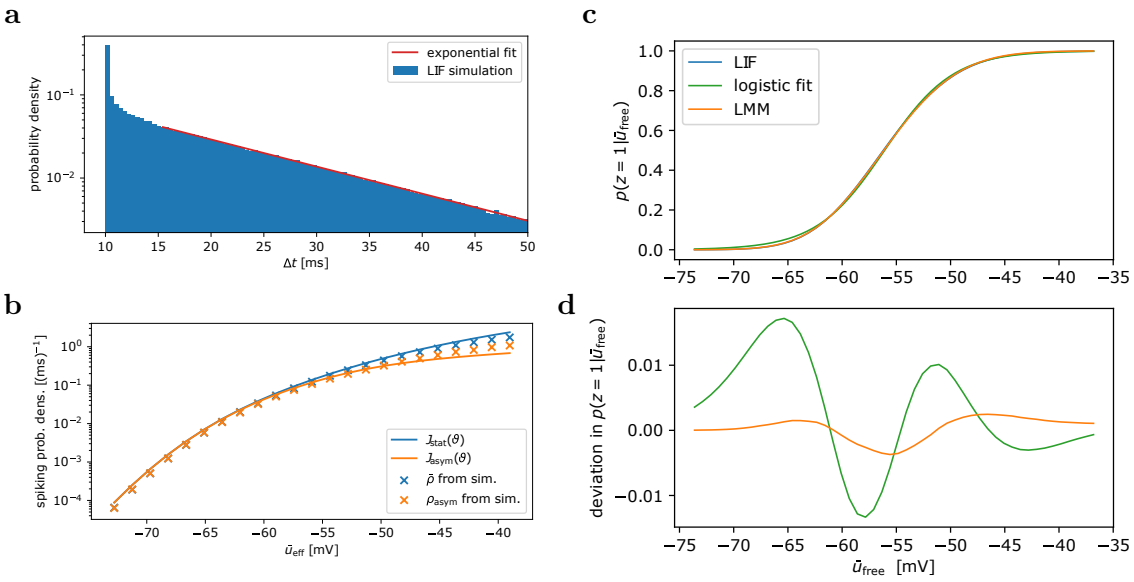

Figure 3.10.: **Single neuron statistics:** **a** The ISI distribution of a single neuron for constant $\bar{u}$ for large Δt's follows an exponential distribution, as expected by the assumed Poisson process. For the times shortly after the end of the refractory period excess activity is observed due to burst firing. The asymptotic spiking probability density ρ_{asym} is given by the exponential decay constant. **b** Total (blue) and asymptotic (orange) spiking probability density of a single neuron as a function of $\bar{u}$. For smaller $\bar{u}$ hardly any bursting occurs. For larger $\bar{u}$ the burst activity starts to dominate and deviations between model (lines) and simulation (crosses) behavior increases. **c** Resulting activation function for the LIF Markov model (LMM) and LIF simulations. The remaining deviations (**d**) are due to neuron parameters being at the edge of the diffusion approximation. Images taken from [Gürtler, 2018].

constant and a and c are the actual fit parameters. These have to be measured once in order to be able to evaluate the values for $P_n(u)$ and thereby inform the stochastic model.

However, we are not interested in the probability density of the membrane potential distribution, but rather the resulting instantaneous firing rate $\rho(u)$. It is again necessary to split the two sources up: On the one hand we have neurons that have not spiked for a long time. These fire due to the diffusion above the threshold potential V_{thresh} from the asymptotic distribution $f(u)$. This we can calculate via the Fokker-Planck equation Eq. (2.67) at the threshold value V_{thresh}. Gürtler [2018] calls this the asymptotic instantaneous firing rate ρ_{asym}. On the other hand, neurons spike immediately after the end of their refractory period τ_{ref} if their effective membrane potential is above the

threshold $u^{\text{eff}} > V_{\text{thresh}}$. This we calculated as P_n in Eq. (3.36) from the stationary distribution and we call this ρ_{stat}.

When we related the LIF sampling framework to sampling from Boltzmann distributions in Section 2.2.4 we essentially postulated that the ISI distribution of an LIF neuron under constant $\bar{u}$ follows an exponential distribution. The actual observed ISI distribution in Fig. 3.10a shows deviations for small inter spike intervals $\Delta t \approx \tau_{\text{ref}}$. These are spikes from bursting neurons and the excess contribution shortly after $\tau_{\text{ref}} = 10\,\text{ms}$ corresponds to ρ_{stat}. For large Δt the frequency distribution follows the expected power law, giving us a Poisson process with fixed mean firing rate $\nu = \rho_{\text{asym}}$.

From earlier discussions we would expect the burst spikes to form a δ-peak at $t = \tau_{\text{ref}} = 10\,\text{ms}$. The exponential decay that we actually observe is rooted in the finite membrane time constant τ_{m} that gives rise to the drift times τ^b's in Eq. (2.75), which we ignored here[20].

In Fig. 3.10b the total spiking probability density $\hat{\rho}$ (blue) and the fraction due to non-bursting spikes ρ_{asym} (orange) is shown as a function of mean membrane potential $\bar{u}$. We show both simulations (crosses) and pure model predictions (solid lines). The simulation numbers are derived from the ISI distribution (i.e., the histogram in Fig. 3.10a), where the non-bursting spiking probability density ρ_{asym} is integrated over the long-tail fitted exponential decay and the total spiking probability density is integrated over all spikes. The deviations between the simulated and the analytical numbers are due to a combination of

1. finite membrane time constant τ_{m}, limiting the instantaneousness of the bursts,

2. finite noise frequencies r_{exc} and r_{inh} and thereby finite W_{exc} and W_{inh}, limiting the diffusion approximation underlying the stochastic model,

3. finite $\frac{\tau_{\text{syn}}}{\tau_{\text{ref}}} > 0$, limiting the validity of the approximation via $f(u; t_{\text{spk}} + k\tau_{\text{ref}}) = c^k f(u; t_{\text{spk}} + 2\tau_{\text{ref}})$.

Nevertheless, the model qualitatively predicts the correct behavior and we also note the logarithmic scale of Fig. 3.10b, meaning that for higher mean membrane potentials $\bar{u}$ almost all spikes are from bursts. Whereas for lower mean membrane potentials $\bar{u}$ almost all spikes are first spikes, i.e. there is an ISI $\Delta t \gg \tau_{\text{ref}}$ between the previous and the current spike.

Even though the spiking probability density ρ differs between model and simulation the resulting activation function (cf. Fig. 3.10c) of the LIF Markov model (LMM) developed here fits significantly better the observed activity in simulations, when compared to to the logistic function Eq. (2.58) of the Buesing model. The deviations are reduced by a factor of about 3 (Fig. 3.10d). We still did not take into account the finite membrane time constant τ_{m} (which is a step backwards compared to the original treatment in [Petrovici et al., 2016]) and reduced the synaptic time constant τ_{syn} as compared to the

[20] An exact treatment of these is of course possible, but would only serve to complicate the explanation for little gain. See e.g. [Petrovici et al., 2016, Gürtler, 2018] for detailed derivations.

refractory time τ_{ref}. In exchange, we gained a better intuition about the short-term dynamics of our stochastic LIF neurons.

While our LMM-derived activation function is a marked improvement, we should note here that predicting the *steady-state* activation function correctly is a necessary, but not sufficient condition for a good dynamical model of LIF sampling networks. We also point out that nothing in this section is strictly limited to LIF neurons as the argument is rooted in the effective membrane potential u^{eff} which is a function of the synaptic input only. As such it is much more bound by the choice of synapse model than neuron model. The only characteristics used are the hard refractoriness and the existence of spikes. The principles of the LMM should therefore be transferable to any kind of fixed threshold or quasi-fixed threshold spiking neuron.

3.3.2. Response to a single synaptic input spike

So far we have discussed the case of a single neuron without any *meaningful* – non-noise – input. While we already had to make assumptions[21] to get this far, being *only* able to describe such a neuron would not be particularly useful. What we are after is a description of the neuron's response to synaptic input from other network neurons. In the discussion above, spikes were either generated via diffusion of the membrane potential above the threshold value V_{thresh} or via the supra-threshold part of a neuron that was becoming available-to-fire again at the end of a refractory period. In this picture the effect of a synaptic input is a shift of the membrane potential distribution of the receiving neuron.

This is in stark contrast to the Buesing model we discussed in Section 2.2.3. There a synaptic input simply shifts the firing probability of the neuron according to the connecting weight w_{ij}. Here we will shift parts of the membrane potential distribution *above* or *away* from the threshold, which will lead to a significantly faster reaction of the system. We will start by looking at a single excitatory synaptic input:

Excitatory input

Fig. 3.11a shows a schematic representation of the reaction of a single excitatory PSP (green) on the membrane potential distribution (blue). By construction the initial distribution corresponds to the steady-state distribution $f(u)$. For now we assume that to be the steady-state distribution at an activity level of $\langle z \rangle = 0.5$, in practice it will be a complicated distribution depending on the previous time course of the input. At the onset of the PSP this distribution $f(u)$ gets shifted upwards by the amplitude of the PSP. This shift moves a significant fraction of the available – non-refractory – neurons above the threshold V_{thresh} (orange part in Fig. 3.11a) and triggers an immediate spike response. Within the rising part of the PSP the spiking probability density is massively increased. This has two reasons: First, the rising mean of the distribution which shifts mass above the threshold, and secondly, the higher mean $\bar{u}$ itself, which is associated

[21] Which do not completely hold in our simulations.

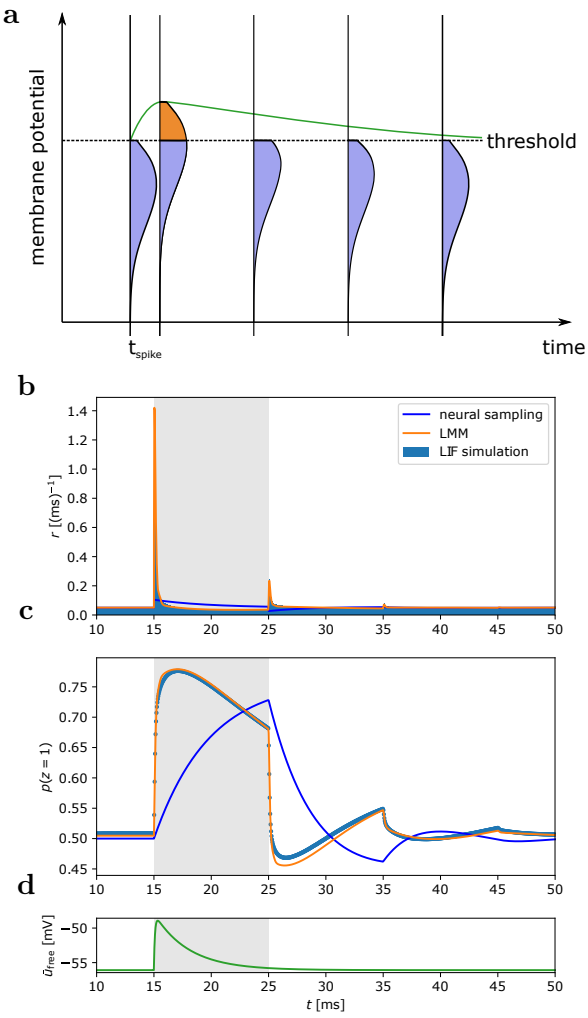

Figure 3.11: **Excitatory input:** **a** Illustration of the time evolution of the membrane potential distribution $f(u)$ of non-refractory LIF neurons. An excitatory PSP shifts it upwards, pushing a fraction above the threshold leading to immediate spikes (orange). Spike (**b**) and activity (**c**) responses of LIF (blue bars), Buesing (blue line) and LMM (orange line) neurons after an excitatory PSP. **d** $\bar{u}$ evolution due to the excitatory PSP. The Buesing neuron model is unable to cover the immediacy of the LIF reaction. Images taken from [Gürtler, 2018].

with stronger fluxes over the threshold $J(V_{\text{thresh}})$ (cf. Fig. 3.10b). While the former effect is by far the dominant one, the second one leads to the counterintuitive case that at the beginning of the decay of the PSP, the spiking probability density is still increasing because of the higher $\bar{u}$ even though the decay of $\bar{u}$ is already reducing f.

In Fig. 3.11b the firing probability densities of the available part of an ensemble of neurons for the LMM (orange line), LIF neurons (blue bars) and Buesing neurons (blue line) in response to the PSP in Fig. 3.11d is shown. For the Buesing model we use the rectangular PSPs with equivalent integral within the refractory period (i.e., under the same assumption that we used to motivate the translation in Section 2.2.4). We see a difference in the reaction of the two neuron models, both in scale and in form. For a Buesing neuron the spiking probability density within the refractory period is always increased as long as there is a positive PSP contribution. At the end of the refractory period the spike activity drops below the steady-state level the pool of available neurons

was depleted by the excess activity.

In contrast, the LMM model correctly captures the extremely bursty nature of the LIF neuron, where most of the response is generated within the rising flank of the PSP. While this spike response is more than an order of magnitude stronger than the response of the Buesing neuron, it is also short-lived. The increasing $\bar{u}$ sustains a larger than average spike response for only about 2 ms, before the response drops below the steady-state value again. Here the PSP still retains nearly half its maximum height (cf. Fig. 3.11d around 17 ms), but the thus induced excess flux gets counteracted by the downward drift of the membrane potential distribution, due to the receding PSP. This diminishes the flux above V_{thresh} until the end of the refractory period. The spike response is also subdued as the pool of non-refractory neurons depletes. At the end of the refractory period we see a smaller echo of the initial response, which is generated by the supra-threshold part of the effective membrane potential distribution of the neurons from the initial spike response. Effectively this is the burst excess that caused the complication in the activation function (cf. Fig. 3.10).

These response differences lead to deviations in the mean state of the ensemble (cf. Fig. 3.11c). For the LIF/LMM neuron, $p(z = 1)$ increases in the beginning. Here, even the LMM struggles to capture the immediate rise correctly. This is largely because of the non-adiabatic nature of the shift due to the PSP. The maximum ensemble reaction is reached at the point where the spike response falls below the steady-state value (2 ms see above), after which more neurons exit their refractory state than enter it by spiking. For the latter two thirds of the refractory period the mean activity of the post-synaptic neuron is decaying. At the end of the refractory time the mean activity of the ensemble drops sharply as most of the neurons from the first response become inactive again.

In contrast, the Buesing neuron shows an increased spike response over the whole refractory period. This is expected as the neuron's response function only depends on the input. The shift by w increases the instantaneous firing probability by a factor of $\exp(w)$ for all of the pre-synaptic neuron's refractory period. The mean response rate (cf. Fig. 3.11c) decays immediately as the pool of available neurons starts to deplete, but for our choice of w stays elevated while the PSP contribution is non-zero.

On the level of the mean state of the ensemble (Fig. 3.11c) it exhibits a slowing increase over the complete refractory period. Effectively the mean activity decays with some time constant towards its new steady-state activation level of

$$p(z = 1) = \sigma(w). \tag{3.39}$$

In particular, $p(z = 1)$ reaches its maximum at the end of the refractory period. At this point the LIF ensemble already lost 0.12 activity as compared to its maximum.

For both Buesing and LMM model, the mean activity oscillates visibly for $(2-3)\tau_{\text{ref}} \approx (6 - 10)\tau_{\text{syn}}$. These oscillations originate in the uneven distribution of the refractory states of the neurons. In the steady-state case, the distribution of "ζs" for refractory neurons is flat as the number of spiking neurons and the number of neurons leaving the refractory period is constant. This uniformity is broken by the synaptic input and only gets restored via a diffusive process after the end of the refractory period. The anti-cyclic

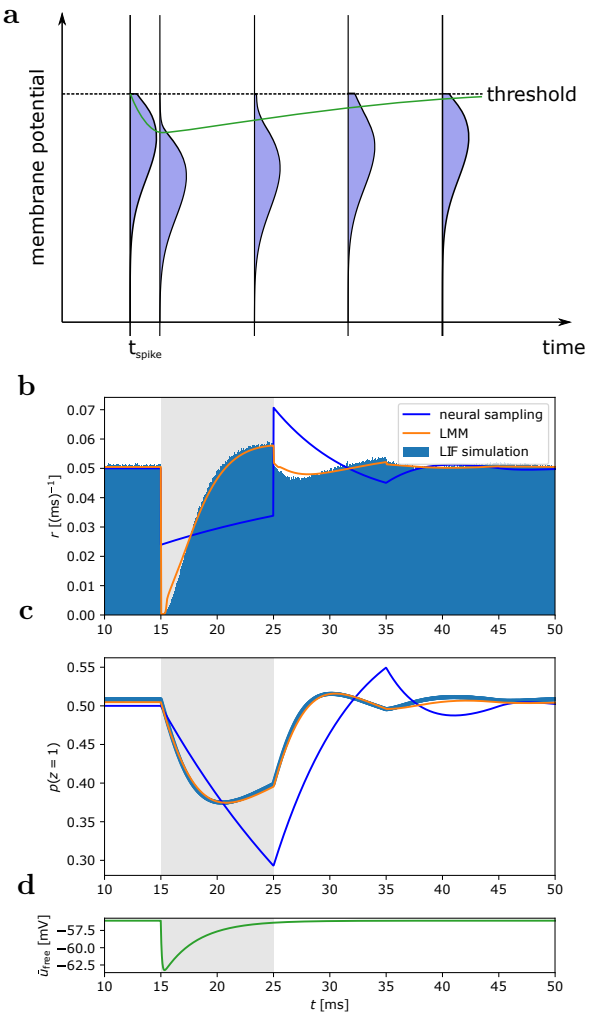

Figure 3.12: **Inhibitory input:** Figure similar to Fig. 3.11 only with an inhibitory PSP. An inhibitory PSP shifts the distribution downward, resulting in a reduced firing immediately after the onset of the PSP. Within the PSP decay the firing rate is increased due to the diffused distribution being shifted upwards again. Images taken from [Gürtler, 2018].

nature of the LIF reaction (early response) and the Buesing reaction (late response) is echoed throughout the oscillations.

Inhibitory input

Let us now turn to take a look at a single inhibitory PSP. For the Buesing neuron (blue line) the situation is still symmetric to the excitatory input (Fig. 3.12a). Rather than an upwards shift of $f(u)$ by W it is now a downwards shift by $-W$. Due to the reduced number of spiking neurons, the number of non-refractory – and hence available – neurons increases. This leads to a moderate recovering of the ensemble spike response rate $r(t)$ already during the refractory period. At the end of the refractory period the spiking probability returns to its original value and the ensemble spike response rate $r(t)$ is thereby increased beyond its steady-state value.

In contrast to this, during the falling flank of the PSP the spike response of the LIF

neurons drops to zero (cf. Fig. 3.12b around 15 ms). Similar to the excitatory input case, the LMM model struggles to correctly model the scale of the initial response exactly, but the principal shape is captured correctly. Here, we can observe the inverse effect to the excitatory case as the spike response starts to recover already before the crest of the PSP is reached. This happens since the diffusion can compensate the slowing downwards pull of the distribution due to the PSP. Within the rising slope of the recovering mean $\bar{u}$ of the membrane potential distribution the spike rate recovers due to the two effects already discussed in the excitatory case:

1. The reversion of $\bar{u}$ to its steady-state value, i.e., the parts of the distribution $f(u)$ just below the threshold value V_{thresh} are shifted upwards eliciting spikes.

2. The higher $\bar{u}$ (even without the active upwards shift) also leads to an increased flux over the threshold $J\left(V_{\text{thresh}}\right)$.

Unlike the excitatory case these two effects work in unison and *both* increase the spike response. It surpasses the steady-state response level at roughly half of the refractory period[22]. The ensemble response $r(t)$ drops sharply at the end of the refractory period (25 ms in Fig. 3.12b). This is due to the lack of burst spikes from neurons that were supposed to spike at 15 ms, but were prevented from doing so by the inhibitory PSP. In this particular situation the effect of the increased pool of available neurons, which would increase $r(t)$, and the decrease due to the lack of burst spikes roughly cancel each other out.

Compared to the excitatory case there are some more prominent variations between the LMM prediction and the LIF simulations. The LMM model underestimates the suppression in the immediate aftermath of the synaptic input and compensates for this by suppressing the recovery a bit. In particular, it is the transition from the downwards shift (with $r(t) \approx 0$) to the diffusive recovery that is not perfectly described by the LMM. This is not too surprising as we lack a from-first-principles description in the non-steady-state situations. Here this is apparent in the "jump" in the LMM prediction for $r(t)$ immediately after the onset of the PSP (orange line in Fig. 3.12b).

In addition to this special limitation, we also note that the assumption of LIF simulations being in the diffusion limit is constrained by the finite simulator time step of 0.1 ms, the finite weight of the noise input and the finite rate of the noise input. While these limitations are of a technical nature, and hence, not fundamental to the model but rather rooted in our limited computing capabilities, there is another fundamental limitation of which we cannot judge the importance due to these technical restrictions: The ad hoc measure of modeling the exponential cutoff of the sub-threshold part of the steady-state distribution of the OU process is unclear (cf. Eq. (3.38)).

Regardless of these limitations, the LMM reproduces the LIF dynamics significantly better than the Buesing model. The spike response of the latter is largely symmetric to the excitatory case (note that Fig. 3.11b and Fig. 3.12b have different scales on the y-axis). This is not surprising as the Buesing neuron model only explicitly breaks the

[22] Remember that we use $\frac{\tau_{\text{syn}}}{\tau_{\text{ref}}} = 0.3$ in this section and it is not completely obvious how this would extend to the case of $\tau_{\text{syn}} \approx \tau_{\text{ref}}$.

symmetry between the $z = 0$ and $z = 1$ state, whereas the LMM and LIF neurons also have an asymmetric spike response function due to the forced threshold crossing via shifts of $f(u)$.

On the level of mean activity $p(z = 1) = \langle z \rangle$ (cf. Fig. 3.12c) the inhibitory case leads to less dramatic effects. This holds also for the LMM and LIF models, as the spike response is bounded from below by $r = 0$. This leads to a limit of how quickly the mean activity can decay[23], unlike in the excitatory case where the initial response can, in principle, force all available neurons to spike immediately. Nevertheless, there is a more pronounced decrease for the LIF/LMM case when compared to the Buesing model. This is due to the immediate drop of the spike response to $r = 0$ of the LIF neurons, while the Buesing spike response only drops by about 50 %.

A side effect of this bound of $r = 0$ is an, at least initially, reduced variation between Buesing and LIF response on the mean activity level. While the spike response of the LIF/LMM model is still below its steady-state value (roughly for the first half of the refractory period τ_{ref}), the mean activities differ by less than 0.04. It is only in the later parts of the refractory period, where the LIF activity already starts recovering, that the Buesing network still predicts a reduced activity. After this the activity level of the LIF neuron is approximately constant as the firing rate recovers close to its steady-state value. The final activity difference at the end of the refractory period is significantly larger than in the excitatory case (0.1 vs 0.05).

For the following oscillating behavior similar arguments as in the excitatory case hold, and again, these are amplified as we are looking at the ensemble response rate rather than the individual neuron's firing probability.

Asymmetric reaction

We found that there is an inherent asymmetry in the dynamical response of LIF neurons with respect to the sign of their synaptic input. Unlike their Boltzmann-proven Buesing counterparts, where the activity is shifted by about ± 0.22 at the end of the refractory period, the corresponding activity level is $+0.18$ in the excitatory and -0.1 in the inhibitory case for LIF/LMM. These differ by nearly a factor of 2 with respect to the values for the Buesing model. This does not even account for the fundamentally different shape of the response *dynamics*, where not even the point at which the maximum response is observed is the same. The peak position of the response is different, while for LIF neurons it (roughly) corresponds to the peak of the PSP, for the Buesing neuron it reaches its peak response always at the end of the refractory period.

Again we stress that the inputs were supposed to be equivalent under our translation assumptions (see Section 2.2.4). While we only showed this for a single mean activity level of $\langle z \rangle = 0.5$, we note that this is also the most symmetric case. Even though extrapolating this onto arbitrary situations and input spike trains would be ill-advised,

[23] The speed of the decay depends on the relative distribution of the neurons throughout their refractory periods. Assuming this is the only input, this distribution is uniform and the decay in mean activity would be linear over the refractory time (for sustained suppression of spiking activity).

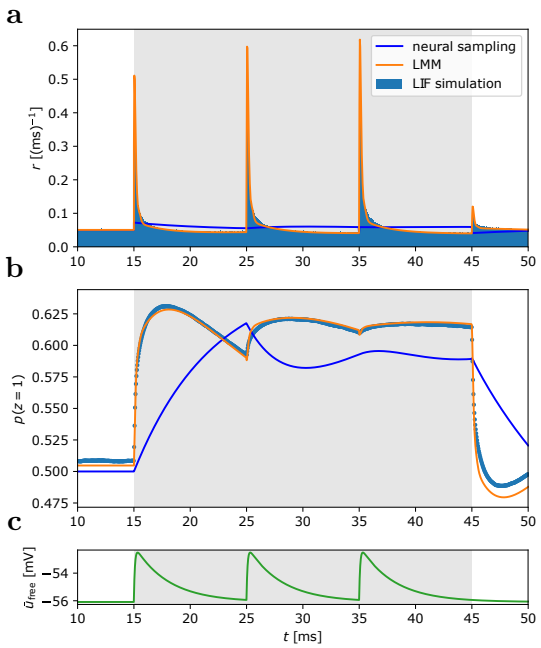

Figure 3.13: **Response to regular synaptic input: a-c** Similar to Fig. 3.11b-d spike, activity response and $\bar{u}$ evolution for multiple consecutive excitatory PSPs for Buesing, LMM and LIF neurons. The general form of the spike response is captured well by the LMM model. There is a significant deviation between the final steady-state activity levels induced by a clamped (bursting) neuron of nearly 0.04. A significant part of this would be compensated for a learned system. Image taken from [Gürtler, 2018].

it is at least indicative that the scales of excitatory and inhibitory synaptic plasticity (weight updates) are different for LIF neurons.

3.3.3. Response to multiple input spikes

After having discussed the dynamical response to single inputs in Section 3.3.2 we now turn to the slightly more involved setting with three successive input spikes of an excitatory pre-synaptic neuron. This roughly emulates the calculation of the probability conditioned on the state $z = 1$ of the pre-synaptic neuron.

The neuron is subject to three spikes with synaptic time constant $\tau_{\text{syn}} = 3\,\text{ms}$ but with a spacing (ISI) of $\tau_{\text{ref}} = 10\,\text{ms}$. For the discussion of the dynamics within the first refractory time we refer to the previous section. The second and third input trigger spike response peaks that are about 20 % and 25 % stronger than the initial one. This is not a plotting artifact, but rather a reflection of the echoes that we saw in Fig. 3.11b. In combination with the upwards shift of the membrane potential distribution elicited by the onset of the second (third) PSP the total ensemble response is increased.

Again the Buesing model has a significantly subdued response pattern, as its firing probability is only modulated by the $\exp(w)$ over the whole three refractory periods $3\tau_{\text{ref}}$. The observable variation in the ensemble spike response rate $r(t)$ is caused by the variations of the size of the pool of non-refractory neurons. Only after being roughly half-way through the second refractory period (30 ms in Fig. 3.13a) this stabilizes enough that variations of $r(t)$ become non-obvious.

For LIF neurons we do not observe the usual sharp drop in the activity level $p(z = 1)$ as the first wave of neurons becomes inactive again (cf. Fig. 3.11b). Due to the

upwards shift of the membrane potential distribution induced by the second PSP most of these neurons will also be forced to spike again and thereby sustain the excess activity. The form of the dynamics within the second refractory period (cf. 25 ms to 35 ms in Fig. 3.13b) resembles the form within the first refractory time, only on a higher absolute activity level with a less pronounced rising flank. Interestingly enough, already in the third refractory period the mean activity of the LIF ensemble is nearly constant, even though the input is sharply peaked (cf. Fig. 3.13c). A comparison with the equilibrium state should be treated with caution though. In the equilibrium state the distribution of internal states with in the refractory time $\zeta < \tau_{\text{ref}}$ is necessarily flat. This is *not* the case here as the spikes which perform the reset to $\zeta = 0$ are mostly synchronous to the external input.

Buesing dynamics again show a vastly different response function. The most important point here is that the steady-state activity level differs significantly at $\langle z \rangle = 0.58$ and $\langle z \rangle = 0.61$ when compared to the LIF/LMM response, even though we are using supposedly *equivalent* synaptic weights. Even in the "steady-state" case, where the short-term plasticity mechanism hides all of the tail contributions of the PSP (cf. Section 3.1.1) we do not see the convergence to the same values[24]. This is less of a problem in case of a learned system as these variations only correspond to a different scaling of the two weight regimes w (Boltzmann) and W (LIF). Nevertheless, the two ensemble responses only start to become comparable after more than $2\tau_{\text{ref}} = 20\,\text{ms}$ (cf. Fig. 3.13b at 35 ms). This corresponds to approximately $6\tau_{\text{syn}}$ in our parametrization here. Before this point the dynamics differ in a principled manner. In other words: Expecting those two models to be allowed to be used interchangeably is a significant gamble.

Random input

Finally let us take a look at a series of random input spikes with varying amplitudes. We choose a series of different weights w_i, translate these using Eq. (2.80) and send spikes in at different spike times t^i_{spk}. This situation resembles the typical state of a sampling neuron. We still limit the number of concurrent synaptic inputs in order to distinguish effects of single inputs.

We see that both the general form of the LIF spiking probability density (Fig. 3.14a) as well as the resulting mean activity $\langle z \rangle = p(z)$ (Fig. 3.14b) are well tracked by the LMM. It is mainly the crossover points between the initial spike reaction and the mid-term diffusive changes where we observe errors. These integrate up and lead to observable deviations on the order of a few percentage points of the mean activity $\langle z \rangle$ (around 28 ms in Fig. 3.14b). While these deviations are clearly observable and systematic, they are tiny when compared to the deviations to an ensemble of Buesing neurons. As already found in Section 3.3.2, the Buesing dynamics for excitatory spikes tend to be more subdued and on longer time scales than the LIF neurons actually elicit. In other words the LMM behaves qualitatively like LIF neurons, whereas Buesing neurons show fundamentally different behavior.

[24] We did not investigate whether a better TSO parameter set exists.

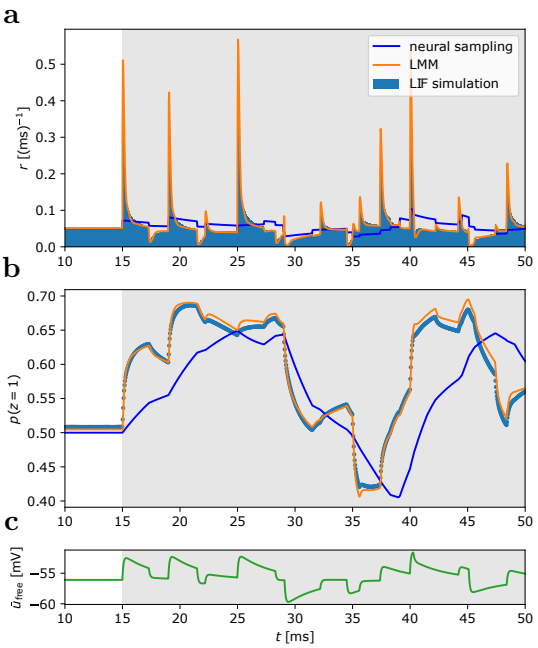

Figure 3.14: **Response to diverse synaptic input: a-c** Similar to Fig. 3.11b-d Spike and activity response as well as $\bar{u}$ evolution for multiple and overlapping PSPs for Buesing, LMM and LIF neurons. The LMM model is able to closely track the behavior of the LIF neuron, whereas the Buesing neuron shows significant deviations. Images taken from [Gürtler, 2018].

One particular example is the reaction to the second spike at 17 ms in Fig. 3.14b, which is an inhibitory spike with less synaptic weight than the first excitatory spike at 15 ms. The weight was chosen such that the second PSP eliminates the remaining effect of the first PSP exactly. That is, the $\bar{u}$ ends up back at its resting value. At the onset of the inhibitory PSP the activity of the LIF ensemble already started to decay, as the generating excitatory PSP is already on its falling slope. This dramatically reduces the firing probability of the LIF neurons and actively decreases the mean activity $\langle z \rangle$. In contrast, Buesing dynamics still show an increasing $\langle z \rangle$ as the firing probability is still elevated due to the net input at this point still being positive. The Buesing neuron, at this point in time, sees the sum of the first excitatory input w_1 and the second smaller inhibitory weight w_2:

$$\Delta u_k = w_1 - w_2 > 0. \tag{3.40}$$

Due to the less bursty nature, the Buesing network shows significantly reduced activity up to the second input (0.55 vs 0.64 at 17 ms). Therefore the continued increase over the inhibitory input actually ends up decreasing the variation (0.6 vs 0.56), while still showing an inverted behavior as compared to LIF neurons.

A similar discrepancy appears at 25 ms, the end of the refractory period τ_{ref} of spikes triggered by the first input spike at 15 ms. At this point the Buesing model's activity decreases due to the accumulated excess activity. Even the increased membrane potential (by the fifth synaptic input at 25 ms, which is excitatory again) does not increase the firing probability enough in order to sustain this activity level with $\langle z \rangle = 0.65$. On the other hand the upwards shift of the membrane potential distribution induces a further increase the activity both for the LIF ensemble as well as for the LMM model. Here, we

also see the limitations of the LMM as the amplitude and even the form of that particular increase is not accurately predicted. It is only with the large inhibitory PSP at 29 ms that the mean activity comes back down to around $\langle z \rangle \approx 0.5$. At this point we see again the better (or less bad) performance of the Buesing model for inhibitory spikes. While it still is not able to follow the drop (from $\langle z \rangle = 0.65$ down to $\langle z \rangle \approx 0.5$) completely the variations stay below 0.05. In contrast, for the excitatory phases the deviations are consistently above 0.1. The neuron also does not have the time to show the constant activity in the second half of the refractory period that we observed in Fig. 3.12c. This is an indication that having multiple input PSPs superimposed may actually be better for the dynamical "correctness" of the Buesing model.

Summary

Given all these dynamical variations that Gürtler [2018] uncovered, it is somewhat surprising that the translation scheme from [Petrovici et al., 2016] (cf. Section 2.2.4) works as well as we observe it to do. While we do not have a conclusive proof that these variations are the origin of the finite sampling accuracy observed in past experiments [Petrovici et al., 2016, 2017b, Leng et al., 2018, Kungl et al., 2019, Dold et al., 2019], it is very likely at least part of the reason.

The LMM model clearly describes LIF dynamics significantly better then the assumption of a Buesing neuron does. However, its performance is still somewhat limited. In particular, the initial response is tricky to be captured correctly. This is rooted in our failure to accurately calculate the burst probability and steady-state spike distribution as well as the resulting need for empirically measured and fitted values ρ_{asym} (cf. Fig. 3.10). Especially for the dynamic situation in the beginning, an adiabatic approximation of the shift over the threshold is bound to fail to be precise. More interesting is the failure in the "middle" where we transition from the shifting phase in the beginning to the diffusion phase on the (comparatively) slowly decaying flank. Here we are still in a non-equilibrium situation where we have little in terms of formal derivations to use as a guide. The resulting LMM is a surprisingly well fit, especially when compared to the Buesing dynamics, however, it is also not a universal solution when we want to actually calculate expectation values from first principles. The main problem is that we do not know whether there exists a closed-form solution of the steady state energy function. Whereas the dynamics of Gibbs or Buesing neurons are derived from the full Boltzmann distribution, here we search for an integrated description of specific microscopic dynamics.

The final experiment we discussed used a diverse set of weights and associated spike times. As such, it more closely resembles the situation of a typical LIF neuron in a sampling network. We reduced the number and thereby frequency of the stand-ins for pre-synaptic partner neurons to make Fig. 3.14 more amenable to discussion, but we would not expect qualitative differences for larger input sizes. There is however the limitation that we did not include any kind of recurrent connections, i.e., the input spike train $S(t)$ is independent of the neurons' response. This is not true for the typical sampling network, where we require even direct feedback connections. As such we are

not able to make a strong statement regarding the quality of the LMM approximation under such highly non-linear interactions. However, there is still reason to expect it to outperform the Buesing model.

In particular, we have seen a dynamical asymmetry between excitatory and inhibitory connections which is known in biology, where this asymmetry is observed at the neuron level (cf. the discussion of Dale's law in Section 2.1). One would assume that this also generates functional implications, which we do not honor by assuming a Boltzmann-like distribution. However, maybe the most important improvement is that the LMM can describe the dynamics of arbitrary connectivity graphs and is not restricted to symmetric pre-post and post-pre connections ($w^T = w$). The latter comes with the price of not having access to a closed-form global energy function $E(\vec{z})$. Unfortunately, we do not know whether we can derive different (and improved) learning schemes from the LMM, which is one of the most interesting further research questions. Gürtler [2018] made progress towards this as he found an implicit description of the target distribution as a set of constraint equations, which can be solved numerically. This improved the DKL of a translated system with parameters W and V_l by an order of magnitude.

3.4. Comparison of the models

After having discussed different neuron models it is instructive to briefly summarize their function and also the level of description at which the models are located. In this section we will take a quick overview over the relationships between Gibbs (Section 2.2.2), Buesing (Section 2.2.3), LMM (Section 3.3), LIF (Section 2.2.4) and biological neuron models. Table 3.1 gives a quick one-page overview.

Biological neurons

We started in Section 2.1 with a brief and rather superficial description of biological neurons and their peculiar eigendynamik which gives rise to the characteristic action potential or spike. Our comparison starts with cell-level models, where a significant fraction of the neuron's inner cell is already abstracted away. There are, of course, models that take this level of detail into account (i.e. they explicitly model the 3D nature of biological neurons e.g. [Stuart and Spruston, 2015, Stuart et al., 2016, Poirazi and Papoutsi, 2020]. For our purposes we summarize everything from the most detailed model of parts of the cell membrane up to the Hodgkin-Huxley model with its three coupled ODEs [Hodgkin and Huxley, 1952] as biological models. These can form near-arbitrarily complex systems of (ordinary or partial) differential equations. They share the excitability feature, i.e., they produce an action potential in response to excitatory stimuli crossing a quasi-hard threshold. These mechanistic models tend to be deterministic and if they introduce stochasticity it is typically to model their environment, be it the connected network activity or the intercellular medium. Simulating these models is typically very compute intensive and therefore prohibits large-scale simulations.

LIF model

The LIF model (cf. Section 2.1.1) is the arguably simplest of the biologically-plausible point neuron models. It still mechanistically describes the behavior of a single cell, but already simplifies the internal state of the neuron to its membrane and the synaptic inputs. In our sampling framework (cf. Section 2.2.4) we choose the membrane time constant to be low $\tau_\mathrm{m} \to 0$. Therefore its effective internal state u^eff is directly dictated by the state of the synaptic input and the reset mechanism does not additionally erase information. We explicitly introduce stochasticity via the noise input. This is not a necessary course of action as we could also use the output of other networks, even ones performing a computational task, rather than dedicated noise sources (cf. Section 3.2.3 [Dold et al., 2019]). This "random" input effectively implements an Ornstein-Uhlenbeck (OU) process on the effective membrane potential u^eff (and thereby also on u as long as the neuron is not in the reset state). Within a network of LIF neurons all communication is encoded in the precise spike times. Due to the exponential decay of the synaptic activity there is also no additional memory required. We choose to *interpret* the LIF neuron as "on" or $z = 1$ whenever it is refractory and "off" or $z = 0$ otherwise. This interpretation is required for our mapping to distributions over binary random variables, but does not affect the dynamics of the network.

The LIF Markov model (LMM)

In Section 3.3 we introduced the LMM where we absorbed the explicit injection of noise spikes into the transfer function. The idea behind it is to model the spike response of an ensemble of LIF neurons under (differently seeded) Poisson stimulus for a given network input. As such the ensemble follows the OU description and the network stimuli shift the resulting distribution $f(u)$ around. From this idea we derived the spike response ρ (the fraction of neurons whose u^{eff} gets shifted above the threshold). Due to the bursting behavior of LIF neurons it has not only a dependence on the membrane voltage u but rather also depends on the time since the last spike ζ in its transfer function $\rho(u, \zeta)$. The communicated information is still completely covered by the spike times and for the choice of exponentially decaying synapses again no additional memory is required. For more general interaction shapes $\kappa(t - t_{\text{spk}})$ either the current state of the interaction $\kappa(\zeta)$ needs to be communicated in each time step or the post-synaptic neuron needs to remember the spike times of all its partners and their interaction form. This is the same for all neuron models. Our binary interpretation of the neuron's state is again dependent on whether it is refractory or not. However, the LMM aims to reproduce the spiking probability distribution exactly, thereby a change of the interpretation would apply in the same manner as for the LIF model.

The (extended) Buesing model

The Buesing model (cf. Section 2.2.3) was originally derived by breaking the symmetry between the $z = 1$ and $z = 0$ state of the Gibbs neuron by introducing extended refractory times, i.e., allowing the $z = 1 \to z = 0$ transition only after a number of time steps τ. It calculates the local membrane potential as a simple weighted sum over the states of the other neurons $u_k = \sum_{ki} W_{ki} z_i$. In order to be able to investigate the influence of different PSP shapes we extended this model with an interaction kernel $\kappa(\zeta)$ which depends on the time ζ since the last spike. For this the membrane potential calculation changes to $u_k = \sum_{ki} W_{ki} \kappa(\zeta_i)$, where a rectangular κ_{rect} corresponds to the original Buesing model. We inverted the meaning of ζ which for us counts the time since the last spike and in the original formulation counted down the time till the next possible spike [Buesing et al., 2011]. This allows us to represent post-refractory interactions which was not possible in the original formulation. The transfer function is the same sigmoid derived for Gibbs sampling, but needs to be corrected for the extended refractory time τ, i.e., $\sigma_\tau(x) = \sigma\left(x - \log(\tau)\right)$. This transfer function gives the spiking probability, which is then evaluated for a spike by comparison with a uniform pseudo random number. Here, the association between the refractory state and the binary interpretation is more fundamental: It was used to derive the activation function.

The Gibbs model

In the original Gibbs sampling there is no notion of a division between the internal and external state as the unit does not require any memory of its past or the past of its partners. The probability of its own state being flipped to $z = 1$ depends only on the

current state of all other neurons. This corresponds to $\tau = 1$ in the Buesing model discussed below.

Mechanistic implementations

There are essentially two components to separate: On the one hand there is the dependence of the input on the output of other neurons and on the other hand there is the translation of said input into the output of the neuron itself. On the abstract side (Gibbs sampling) these two points are identical, i.e., each neuron directly depends on the state z of the other neurons and forms its own state based on a weighted sum. The more we go towards biological neurons the more complicated this sum becomes. For the original Buesing neuron this is only a simple check for its refractory state, while with our general kernel $\kappa(\zeta)$ we require the precise time since the last spike. On the other hand the more biological models implement the synaptic input as dedicated differential equations. Here, the input spikes define the points in time where additional interaction strength is deposited in the synapse. In principle these all can be reduced to the transfer of the information at the $z = 0 \to z = 1$ transition times which we associate with spikes.

Stochasticity and relevance

The spike generation on the other hand shows stronger deviations between the different models. Starting again with the abstract Gibbs model, we can analytically calculate the conditional probabilities $p(z = 1)$ and $p(z = 0)$ (cf. Eq. (2.31)) which directly gives us the activation function $\sigma(x)$ (cf. Eq. (2.36)). This probability is transfered to the actual state assignment via the comparison with uniform pseudo random numbers. The introduction of finite refractory times τ requires an appropriate correction of the transition probability $\sigma_\tau(x)$, but otherwise the output generation stays the same. It is only for the LMM where we start to take the more involved LIF dynamics into account. Here, we used the ensemble formulation to derive an approximation of the probability density of the membrane potential $f(u;\zeta)$. This allows us to derive its spike response function $\rho(u)$ as the shift of $f(u)$ that is induced by the synaptic input. The spikes are still generated by comparing $\rho(u)$ with a uniform random number. In contrast the original LIF model implements the noise directly on the membrane potential u rather than in the transfer function. This allows the LIF model to fire stochastically without ever having to use an explicit random number at the neuron level.

The other large question is the one of relevance: Here the answer is more complicated as it depends on the question what the relevance is supposed to be for? From an information processing point of view it is Gibbs sampling that is the most effective way (at least from the ones discussed here) to sample Boltzmann distributions. All other models we use here are complications, starting with the Buesing model, which is essentially a slowed-down Gibbs sampler, to the LIF sampling framework where the sampling is only approximately correct. On the other hand there is evidence that the human brain is performing sampling-based Bayesian inference [Doya et al., 2007]. LIF neurons under Poisson stimulus are a very good, mechanistic, model of how spiking neurons can im-

	Bio. neurons	LIF	LMM	ext. Buesing	Buesing	Gibbs
Input	PDEs/ODEs spike-trig. ion channel activity	ODE spike-trig. PSCs	sum spike-trig. PSPs; no explicit noise	sum $\sum w_{ij}\kappa(\zeta_j)$	sum $\sum w_{ij}z_j$	sum $\sum w_{ij}z_j$
Output	AP/spike "1-bit" + timing	spike time t_{spk} 1-bit + timing	spike time t_{spk} 1-bit + timing	$\kappa(\zeta)$ or t_{spk} 1-bit + timing	$z \in \{0,1\}$ 1-bit	$z \in \{0,1\}$ 1-bit
Transfer function	quasi-hard threshold	hard threshold	$\rho(u,\zeta)$	$\sigma_{\tau_{\text{ref}}}(u)$	$\sigma_{\tau_{\text{ref}}}(u)$	$\sigma(u)$
Stochastic process	emergent from complex network dynamics	dedicated noise spikes, effectively OU	integration of an OU process	PRN comp	PRN comp	PRN comp
Internal state	state of all ion channels	state of synaptic input	time since last spike ζ	time since last spike ζ	time since last spike ζ	z
External state	excitability/refractoriness	(non-)refractoriness	(non-)refractoriness	(non-)refractoriness	(non-)refractoriness	$\{0,1\}$
Communication	t_{spk}	t_{spk}	t_{spk}	$t_{\text{spk}}/\kappa(\zeta)$	t_{spk}/z	z
Interpretation/ Coding	unclear	(non-)refractoriness $(z=0)z=1$	$z = \mathbb{1}_{\zeta<\tau_{\text{ref}}}$	$z = \mathbb{1}_{\zeta<\tau_{\text{ref}}}$	$z = \mathbb{1}_{\zeta<\tau_{\text{ref}}}$	z

Table 3.1.: **Model comparison:** Bullet-point comparison between the different models ranging from biological neurons to binary Gibbs neurons along different axes.

plement this. The specific requirements are either given by default in the hippocampus ($\tau_\mathrm{m} \to 0$ due to the high-conductance state (HCS) [Kumar et al., 2008] or a simplification to come closer to abstract Boltzmann machines ($\tau_\mathrm{syn} = \tau_\mathrm{ref}$, symmetric weight matrix W_{ij} breaking Dale's law). It is by no means clear that asymmetric connectivity matrices do not generate a stationary distribution. It is simply that we do not have access to a suitably simple high-level description for these. Here, the LIF sampling framework offers a way to show that such an implementation works and, in particular, allows us to use spiking neuromorphic hardware platforms (cf. Chapter 5).

4. Ensemble phenomena in Ising-like networks of spiking neurons

After having discussed the intricate dynamics of single Leaky-integrate and fire (LIF) neurons under high-frequency Poisson stimulus and, in particular, the differences to the Buesing neuron model in Chapter 3, we now turn to the network-level dynamics of such spiking neurons. As a significant part of this work was done prior to the development of the LMM (cf. Section 3.3), most of later parts of this chapter still focus on the extended Buesing model. The overarching theme will be the notion of temperature and its meaning within a stochastic system.

Temperature – in the sense of statistical mechanics – is always associated with unordered motion. Within the LIF sampling framework this motion is generated by the background Poisson sources. In Section 4.1 we will formalize the relationship between the noise parameters and the resulting activation function. It is the width of said function that we will then associate with the dynamical temperature of the system. Equipped with this notion of temperature we turn to the arguably simplest and yet widely utilized network structure, the nearest neighbor connected Ising model. We describe this model, its notion of the critical temperature, its temperature and external field dependence and relationship to Boltzmann distributions in Section 4.2. Finally, we will take a look at and explain the phase diagram of Ising-like networks consisting of Buesing neurons, with both rectangular and exponential interaction shapes, (cf. Section 4.3) and LIF neurons (cf. Section 4.4).

4.1. Temperature in LIF networks - spike based tempering

This section presents work done in collaboration with Agnes Korcsak-Gorzo for which a publication is in preparation [Korcsak-Gorzo et al., in prep.].

We, off-handedly, used the concept of a heat bath to describe the dynamical effects of the noise input both when we introduced sampling as a concept in Section 2.2 and when we discussed the nature of the noise sources and their correlation structure in Section 3.2. Before we establish that connection more concretely let us first remember what the dynamic effect of the noise is supposed to be: A way to have the *deterministic* LIF neuron model perform *stochastic* computation.

The noise input serves to implement an OU-like random walk on the membrane potential u. This is a valid description only if the noise is of sufficiently high frequency, such that the membrane is always under the influence of noise (cf. Section 3.2). Imagine if there would only be a "noise" input spike every minute. Even with our choice of the synaptic time constant $\tau_{\text{syn}} = 10\,\text{ms}$ being already on the upper end of biological values (cf. Section 2.1), for the vast majority of the time we would see $u \approx V_{\text{l}}$. It would only be for the few synaptic time constants after the random noise spike time where the membrane potential is excited. We therefore require noise frequencies of at least a couple of noise spikes per synaptic time constant. In our parameterization ($\tau_{\text{syn}} = \tau_{\text{ref}} = 10\,\text{ms}$) this means that we require at least 300 Hz of Poisson noise per synapse type (excitatory and inhibitory). Below this our stochastic description is simply not applicable.

4.1.1. Temperature and sampling

Before we go to the LIF-specific incarnation of temperature it is illustrative to think back to the textbook introduction to temperature as one of the state variables for a system of classical point particles in a box. Here, temperature represents information about the variance of the movement of the single (gas) particles. Together with the other state variables pressure P and volume V, it is used to compress the description of the particles from $6N$ numbers representing all the position and all the velocity vectors (the so-called micro state) to just three numbers (the so-called macro state). The macro state retains an effective description of what a test particle would experience if it was to be inserted into the box. Or what we would experience if we were to change e.g. the volume of the box. It turns out that almost all of the information in the micro state of the system is unnecessary to answer this kind of question.

Of course we do loose information: It is not possible to reconstruct the exact micro state with only the macro state information. In other words, we would need to generate the missing $6N - 3$ numbers which we did not retain explicitly but rather only in a statistical sense. I.e., we only have a probability distribution $p_T(\vec{x}, \vec{v})$ that is parameterized by the state variables.

At this level the similarity to our description of our noise sources their treatment in Section 3.2 becomes obvious. There is one fundamental issue though: In a box of gas particles the number of particles is enormous with $N_a = 1 \times 10^{26}$ being a typical number. The number of neurons in the brain (50×10^9 [von Bartheld et al., 2016]) on the other hand – while still being far too many for us to handle modeling- or simulation-wise – is significantly smaller. In addition, the variation between different neuron types, and even more in their local connectivity structure, is qualitatively different from the homogeneous nature of gaseous particles. As such, large-scale averages are not necessarily as meaningful for each test particle. Which of course does not stop people from trying mean-field approaches, often because it is the only accessible avenue.

We ignored all these fundamental questions of applicability when we abstracted most of the connected neurons away and replaced them by Poisson sources. This input, similarly to the gas particles in a box, randomly interact with our neuron (which would correspond to a test particle in the box) and makes the membrane voltage u perform its random walk. We can *interpret* this random walk as sampling from some probability distribution.

In the "test particle in a box" example, without any kind of externally provided scale, all we would be able to see is that it randomly walks around. When we increase the temperature, and by that the force that single collisions exert on the particle, we would see that it walks around farther and faster. But it is only in comparison to the behavior before that we can say "farther and faster". If we now introduce an external scale, e.g. by introducing an additional collision of known force[1], only then can we measure the scale in an absolute (rather than a relative) manner and thereby define an absolute temperature. In a way we need to define what the temperature $T = 1$ *means*. Here, we assume that no motion at all corresponds to $T = 0$, which for practical reasons we cannot reach (see discussion above).

In general, the probability distribution of a micro state $p_T(\vec{x}, \vec{v})$ from above it takes the form

$$p_T(\vec{x}, \vec{v}) = \frac{1}{Z} \exp\left(-\frac{E(\vec{x}, \vec{v})}{T}\right) \tag{4.1}$$

where the partition sum Z again ensures the correct normalization and integrates over all possible values for $\vec{x}$ and $\vec{v}$. The energy function E is given by the physical properties of the components of the system (e.g. charge, size of the particles, mass of the particles, etc.) and does not depend on the temperature or other effective state variables. It also has the physical unit of an energy and as such needs to be divided by another number with the unit energy. This is what the temperature in a physical system is: Its typical energy scale[2].

How does this now relate to our network of LIF neurons? The Poisson sources correspond to the unordered movement of gas particles inducing the random walk. In our incarnation, it is the variation of the membrane potential that spans the activation function, which in turn provides the relevant scale against which the network input is

[1] Or using a calibrated meter.

[2] We happily ignore k_B, which is only a unit conversion factor.

compared to and therefore defines the dynamical temperature. However, the width of the activation function α also goes into our parameter translation scheme, Eqs. (2.78) and (2.80). This translation is essentially how we *define* the external measurement scale, or in other words, we generate the $T = 1$ association via the translation scheme. In some way this assumes that the translation works perfectly and the deviations we saw in Section 3.3 do not exist. Nevertheless the principal argument still holds in the general case.

Changing the temperature can be implemented via a *change* of the width α of the activation function:

$$T = \frac{\alpha}{\alpha_0} \tag{4.2}$$

where α_0 is the width of the activation function we use in the translation scheme and α is the width of the activation function of the configuration we actually use to generate the sample with.

In traditional sampling methods the change of the sampling temperature is used to improve mixing for distribution with very deep modes[3]. We can construct a pathological example by implementing the probability distribution $p = \{\frac{a}{2}, \frac{2-a}{2}, \frac{2-a}{2}, \frac{a}{2}\}$. For $a \to 2$ this leads to a preference of the 11 and 00 states, such that we do not see the two cross states 01 and 10. This requires a parameterization of $b_1 = b_2 = b \to -\infty$ and $W_{12} = W_{21} = -2b \to 2\infty$ of our network[4]. If we now start in the 00 state, the network will be stuck there as the large negative bias make a spontaneous activation very unlikely. If we were to start in the 11 state, on the other hand, we would be stuck there as both neurons now are under an effectively large positive bias $b + W = \infty$ and cannot move to the $z = 0$ state. In such a configuration we would only ever see a part of the distribution.

This problem is general to all sampling systems. On the one hand, at least in general, we do not know the target distribution and therefore cannot use distance measures to it (like the DKL, cf. Section 2.2.2) as a means to judge the sampling quality. On the other hand, even if we do see a diverse set of modes, we still do not know whether there do not exist additional modes the sampler did not yet reach. One of the standard workaround methods is tempering, i.e., reducing the impact of the connectivity structure, by either reducing the weights or increasing the width of the activation function [Nadler and Hansmann, 2007]. An activation function does not exist in general, as this is an artifact of our probability distributions being defined on a hypercube and updating the neurons sequentially. Effectively, this means the relevant part of our probability distribution only ever has 2 states. In the general case one generates a new proposed sample, typically selected a Gaussian around the current state and accepts or rejects the new sample according to their relative probabilities. Within this framework the mixing can also be improved by increasing the radius from which new samples are proposed [Han and Carlin, 2001]. Increasing the temperature squeezes the energy landscape and by that also reduces the rejection chance due to the flatter probability distribution. This makes

[3] A mode refers to a cluster of related micro states $\{z\}$, e.g. all hand written depictions of the digit 1. We call it deep if the switch into a different mode is hard and does not happen spontaneously.

[4] The precise values for a and b are irrelevant.

the flip from $11 \to 00$ or vice versa more likely[5]. Lowering the temperature reduces the flip probability and thereby deepens the mode.

Here we will argue that the well-known phenomena of cortical oscillations [Berger, 1929, Varela et al., 2001, Engel and Singer, 2001, Izhikevich, 2007] could function as a modulation of the dynamical temperature of the Bayesian brain. The term cortical oscillation typically refers to oscillations in the electroencephalogram (EEG) which have been known since the beginning of the 20th century [Berger, 1929]. They show distinct peaks in the power spectrum [Jung and Berger, 1979], whose amplitude and position also depend on the current state of the brain, for example whether the human is awake or sleeping [Buzsaki, 2006]. The EEG measures changes in the local electric field potential generated by the spikes that neurons elicit. On a microscopic level these oscillations therefore correspond to changes in the activity level of large areas of the brain. There is a significant body of literature that ascribes some functional use to these observed behavior [Diekelmann and Born, 2010, D'Angelo et al., 2011, Obien et al., 2015, Clayton et al., 2015], even though relatively simple networks (in particular, also ones without function of their own) can be shown to exhibit similar oscillations [Brunel, 2000, Chialvo, 2010]. Here we postulate a functional use case for periodic oscillations of the background activity.

4.1.2. Influence of background variations

In Section 2.2.4 we calculated the mean and the variance of the free membrane potential distribution function as

$$\bar{u} = V_\mathrm{l} + \frac{I_\mathrm{ext}}{g_\mathrm{l}} + \frac{\sum_k W_k \nu_k \tau_\mathrm{syn}^k}{g_\mathrm{l}} \tag{4.3}$$

and

$$\mathrm{Var}[u] = \sum_k \left[\frac{\tau_\mathrm{m} \tau_\mathrm{syn}^k}{C_m \left(\tau_\mathrm{m} - \tau_\mathrm{syn}^k\right)} \right]^2 W_k^2 \nu_k \left(\frac{\tau_\mathrm{m}}{2} + \frac{\tau_\mathrm{syn}^k}{2} - 2 \frac{\tau_\mathrm{m} \tau_\mathrm{syn}^k}{\tau_\mathrm{m} + \tau_\mathrm{syn}} \right) \tag{4.4}$$

respectively. Since we model the temperature change as a consequence of the oscillating activity course of the brain, we use static noise weights $W_\mathrm{exc}, W_\mathrm{inh}$ and only change the rates $r_\mathrm{exc}, r_\mathrm{inh}$ of the sources dynamically. As such the variance of the membrane potential scales with

$$\mathrm{Var}[u] \propto (r_\mathrm{exc} + r_\mathrm{inh}) \tag{4.5}$$

and hence the standard deviation scales with

$$\sigma_u = \sqrt{\mathrm{Var}[u]} \propto \sqrt{r_\mathrm{exc} + r_\mathrm{inh}}. \tag{4.6}$$

On the other hand the mean scales with

$$\bar{u} \propto (r_\mathrm{exc} - r_\mathrm{inh}) \,. \tag{4.7}$$

In case of an asymmetric choice of $|W_\mathrm{exc}| \neq |W_\mathrm{inh}|$ there is an additional factor $\frac{W_\mathrm{exc}}{W_\mathrm{inh}}$ between the r_exc and r_inh scaling. For the ease of argument in the following discussion

[5] It also makes 01 and 10 more likely, thereby distorting the distribution.

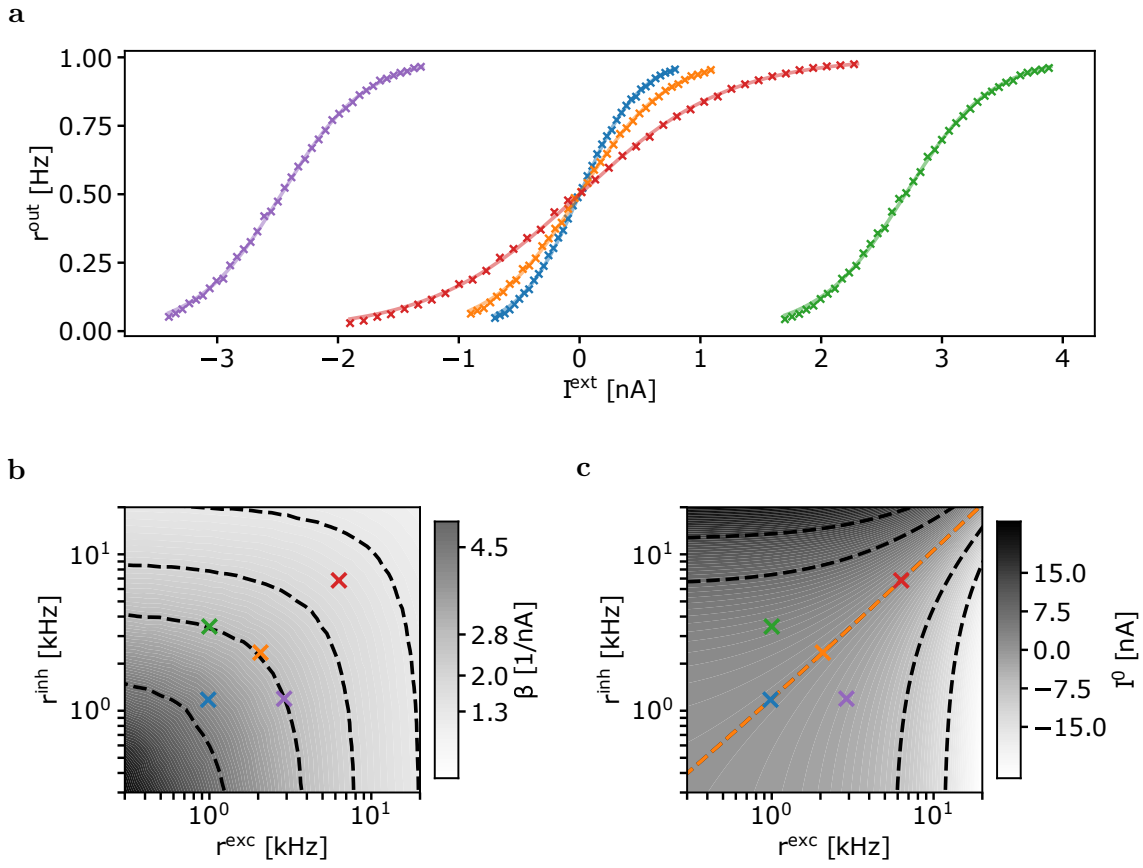

Figure 4.1.: **Membrane potential distributions and activation functions: a** Activation functions for the noise configurations from the marked parameter sets in **b/c**. The crosses show the measured data and the lines the best fit which gives the slope $\beta = \frac{1}{\alpha}$ and offset current I^0 parameters. **b/c** Slope β and central position I^0 of the activation function of a CUBA neuron. Dashed lines represent lines of constant slope and shift. Lines of constant slope β in **b** correspond to approximately constant $r_{\text{exc}} + r_{\text{inh}}$, which would be directly visible with linear axis. The orange cross and line in **c** correspond to the parameter set used as reference configuration (defining $T = 1$) and a bias-free temperature change, respectively. Image adapted from the upcoming publication [Korcsak-Gorzo et al., in prep.].

we assume this scaling factor to be unity, i.e. $W_{\text{exc}} = -W_{\text{inh}}$. This does not restrict the generality of the argument but simplifies the mathematical formulation.

While we do not have a simple relation between the distribution of the membrane potential $f(u)$ and the generated activation function $\sigma(u)$ (cf. Sections 2.2.4 and 3.3.1), we can see from empirical evidence that there is a close correspondence. The width of the activation function α is proportional to the standard deviation of the distribution σ_u. The relationship between the mean of $f(u)$ and the inflection point I^0 is slightly more complicated. It scales linearly with the difference between the excitatory and inhibitory noise rates but there is some offset, which is a complicated function of the reset and burst behavior.

Here, we use external currents I rather than shifting the leak potential V_l to implement the bias to be consistent with [Korcsak-Gorzo et al., in prep.]. Both formulations are mathematically equivalent. Fig. 4.1b shows the slope β rather than the width α ($\beta = \frac{1}{\alpha}$) of the activation function for the same reason. As α corresponds (up to the $T = 1$ definition) to the temperature scaling, we find that the temperature of the system scales with

$$T \propto \sigma_u \propto \sqrt{r_{\text{exc}} + r_{\text{inh}}} \tag{4.8}$$

In other words changing the input frequency of both noise sources by a factor of 4 should change the temperature of the sampling system by a factor of 2. Unfortunately the argument becomes a bit more involved due to the dependence of the central point of the activation function $\bar{u}$ on the noise parameters $r_{\text{exc}}, r_{\text{inh}}$ mentioned above. In Fig. 4.1c we can see that a naive rescaling of both noise rates r_{exc} and r_{inh} (parallels to the diagonal) would, in general, shift the inflection point of the activation function I^0.

We can use any arbitrary noise configuration (and the thus implied activation function) as the basis of the definition of our $T = 1$ point. From this we are fixed to a particular relation between the two noise rates $r_{\text{inh}} = r(r_{\text{exc}})$, such that the offset parameter I^0 is constant by interpolating the measured data from Fig. 4.1c. Once we have this relationship r we can change the temperature of our system near arbitrarily[6].

In Fig. 4.1a we see the activation function parameters as a function of some values $r_{\text{exc}}, r_{\text{inh}}$. We choose the arbitrary point $r_{\text{exc}} = r_{\text{inh}} = 2\,\text{kHz}$ (orange cross in Fig. 4.1b and c) as our definition of unit temperature $T = 1$. This is the activation function which informs the translation from the abstract Boltzmann parameters w, b to the LIF network parameters W, I^0 (cf. Eqs. (2.78) and (2.80)). In order to change the temperature we need to adjust $r_{\text{exc}}, r_{\text{inh}}$ according to $r_{\text{inh}} = r(r_{\text{exc}})$ (orange line in Fig. 4.1c) which leaves the I^0 constant. For example $T = 2$ and $T = 0.5$ are marked with red and blue crosses and the resulting activation functions are unshifted (cf. Fig. 4.1a). If we were to increase the rates arbitrarily we would end up at different width and mean levels (purple and green lines in Fig. 4.1a) which would introduce a bias to all neurons in the system simultaneously. Avoiding this bias creates a fine-tuning problem as the adaptation of the excitatory and inhibitory rates are now tightly coupled via $r_{\text{inh}} = r(r_{\text{exc}})$. For our

[6] We need stay within the applicable parameter range of the model, i.e., the noise input is sufficiently high to have the membrane implement an OU process.

particular choice of $T = 1$ with

$$W_{\text{exc}} = -W_{\text{inh}} = 0.01\,\text{nA} \tag{4.9}$$

and

$$r_{\text{exc}} = r_{\text{inh}} = 2\,\text{kHz} \tag{4.10}$$

together with the rest of the neuron parameters (cf. Appendix B.3), this line ends up very close to $r_{\text{exc}} = r_{\text{inh}}$. This near-linear relation is a pure artifact of our *choice* of parameters. We optimized for ease of reasoning rather than biological plausibility. While this may seem pedantic, we feel the need to stress that $\frac{r_{\text{exc}}}{r_{\text{inh}}} \approx 1$ is not a requirement for the discussed effects to be observable. In particular, the fine-tuning problem always exists and a bias-free modulation of the dynamical temperature of the neurons will require some form of active stabilization mechanism. Simply rescaling the two frequencies in lock-step is not necessarily more plausible than having any other relation.

However, there is another biologically-challenged point in LIF sampling that we can clean up here: Back in Fig. 2.9 we implemented the bias parameter of the Boltzmann distribution b via the leak potential V_{l} and here we use an artificial external current I to achieve the same goal. These are, from a modeling perspective, convenient choices, but from a biological standpoint they are also implausible, as we (at least in principle) require arbitrary values for V_{l}. Remember that V_{l} was actually thought to implement the ion concentration in the intracellular medium[7]. The state of the intracellular medium will have to be a) rather static, in order of the cells to survive, and b) shared between all neurons, which would make the bias shared, or at least correlated between all neurons. Eq. (4.7) allows us to sidestep this, as we can now implement the bias, i.e., the shift of the activation function, via either, a change of r_{exc} and r_{inh} that moves us away from the iso-u_{p05} configuration, or an adaptation of the noise weights $W_{\text{exc}}, W_{\text{inh}}$. The latter feels more natural, as the connection strength is where plasticity – and therefore learning – typically operates. It also fits into the biological interpretation, that there is *no* fundamental difference between noise and network connections. *That* distinction must be only a modeling artifact as there is no large class divide between synaptic input evident in biological data.

Conductance-based neurons

For conductance-based neurons the arguments become a bit more involved as the synaptic effect happens on the level of I^{syn} rather than directly on the membrane potential u. For current-based neurons the relation between the synaptic current I^{syn} and the resulting effect on the membrane u is a simple scaling factor depending on the capacitance C_m and leak conductance g_{l}. Therefore, we can equivalently discuss effects on the I^{syn} or on the u level. For conductance-based neurons on the other hand, the membrane time constant τ_{m} becomes variable and as such also the heights of the elicited PSP changes.

[7] Again we are skipping over the fact that different ions have different permeabilities through the membrane, and therefore it is a bit more complicated than having *one* leak potential, but the analogy holds well enough.

For a fixed synaptic weight W the PSP is reduced by the larger total conductance g_{tot} achieved at larger noise levels (cf. Section 2.1.1). As such the width of the membrane potential distribution becomes nearly independent of the noise frequency for a sufficiently high-conductance state (HCS) [Petrovici et al., 2016]. However, this reduction in PSP height also affects the network PSPs. For the spike behavior it is the relative height of a network PSP and a noise PSP that is relevant. This fraction $\frac{PSP}{\sigma_u}$ behaves the same for conductance-based neurons as we have discussed here for the current-based neurons.

Here also our implementation of the bias b via the leak potential V_l becomes implausible. For higher noise rates, the relative contribution of the fixed leak conductance g_l shrinks. In order to generate enough input current I to move the mean activity of the neuron significantly we need to set the leak potential V_l to extreme values ($\pm$V. This is a purely technical issue and can be absorbed into the configuration of the two noise rates r_{exc} and r_{inh}.

4.1.3. Effects on the imprinted distribution

For our experiments we choose a sinusoidal time course of the excitatory noise rate

$$r_{\text{exc}}(t) = A \sin\left(\frac{t}{2\pi P} + \phi\right) + b \quad r_{\text{inh}}(t) = r(r_{\text{exc}}(t)) \tag{4.11}$$

with some irrelevant initial phase ϕ and a fixed period length $P = 1\,\text{s}$ to $100\tau_{\text{ref}}$. We choose the amplitude A and offset b such that the lowest excitatory frequency is $r_{\text{exc}}^{\text{min}} = 500\,\text{Hz}$ and the maximum is $r_{\text{exc}}^{\text{max}} = 10\,\text{kHz}$:

$$A = 4.75\,\text{kHz}\,; \quad b = 5.25\,\text{kHz} \tag{4.12}$$

We select the inhibitory noise frequency time course for excitatory and inhibitory sources, such that the inflection point of the activation functions u_{p05} coincides with $u_{p05}(r_{\text{exc}} = 2\,\text{kHz}, r_{\text{inh}} = 2\,\text{kHz})$. This choice corresponds to the orange line in Fig. 4.1.

Fig. 4.2A shows the two time courses $r_{\text{exc}}(t), r_{\text{inh}}(t)$ (black and gray lines) as well as the corresponding temperature course (blue line, right axis). The points of lowest (blue), hottest (red) and reference rates at the cold-to-hot transition (yellow) are marked by the vertical lines. The variance of the membrane potential Eq. (2.72) varies in a sinusoidal fashion as it is proportional to $r_{\text{exc}} + r_{\text{inh}} \approx 2r_{\text{exc}}$. The temperature, therefore, shows a $\sqrt{\sin(t)}$ time course, as the width of the activation function is proportional to the standard deviation of the membrane potential distribution (cf. Eq. (4.8)).

We implement a network of $n = 4$ neurons with all-to-all connections and draw Boltzmann parameters w, b from a standard normal distribution with width $\sigma = 1$ and mean $\mu = 0$. We then translate these parameters according to Eq. (2.80) and Eq. (2.78) into LIF parameters W and offset currents I. The choice of $\sigma = 1$ leads to strong weights which in turn lead to significant contrast between the probabilities of the different states z (cf. orange bars inFig. 4.2B for the reference distribution). In total we run the experiment for 10 000 periods P and for technical reasons we adapt the implemented rate every 5 ms. For each of the resulting 200 configurations over a period we collect the 10 000 sampled states individually.

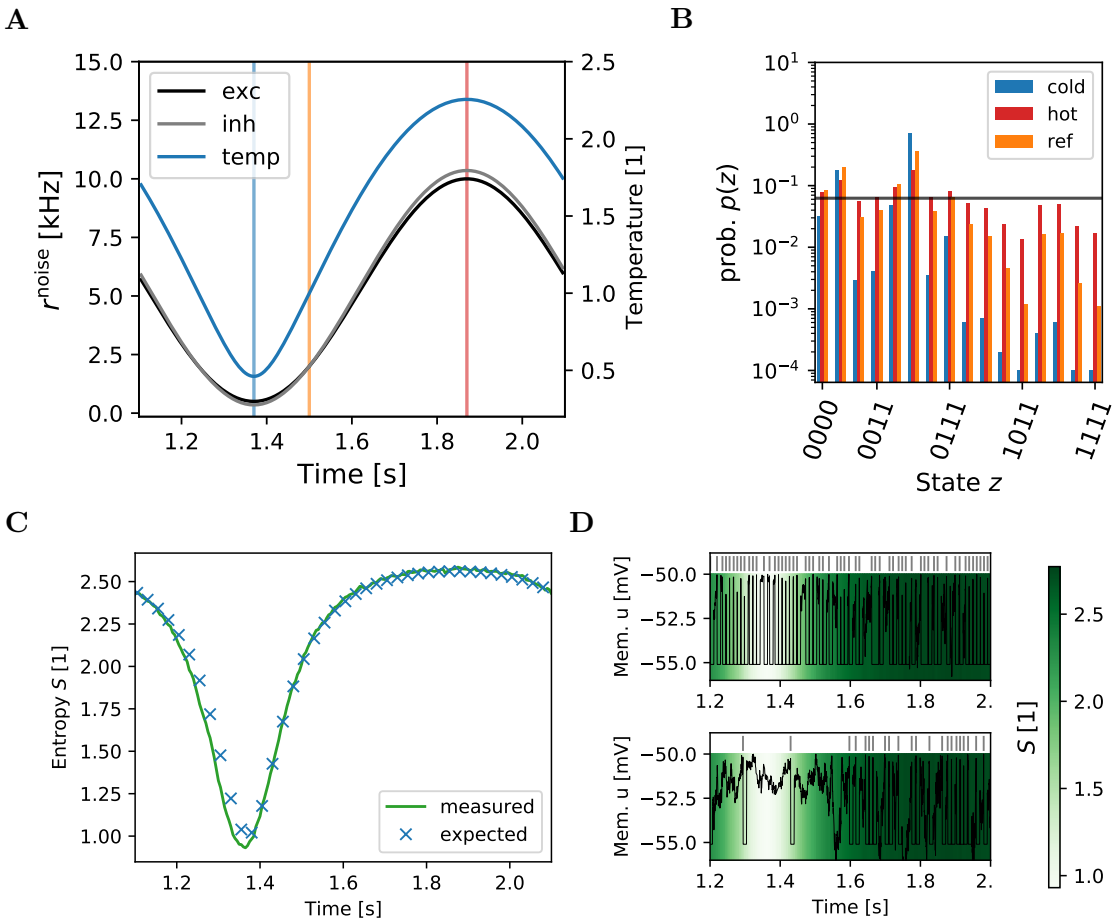

Figure 4.2.: **Imprinted distribution under tempering:** **A** Frequency time course of excitatory (black) and inhibitory (gray) noise sources within the experiment. The relevant low, high and cold-to-hot reference points are marked in blue, red and yellow respectively. **B** Observed probability distribution at the three points from **A**. The cold (hot) distribution is closer (farther away) from the uniform distribution (dashed line) than the reference. **C** Entropy S of the observed distributions. Measured data (green line) and expected entropy via rescaling (blue crosses, see main text). Distributions are sampled for 10 000 periods. **D** Membrane potential evolution with entropy time course in the background. More extreme membrane potential oscillations for higher temperature levels (around 1.8 s) are due to the higher noise level. The spike activity (|) does not only depend on the temperature but also the bias. High bias neurons (upper panel) fire more often for the low temperatures around 1.4 s, low bias neurons (lower panel) less often. Figure taken from [Korcsak-Gorzo et al., in prep.].

Within the distributions at the three distinguished configuration (lowest temperature blue, highest temperature red and reference temperature orange) the behavior is as expected: In general higher temperatures lead to a closer-to-uniform distribution (horizontal line in Fig. 4.2B), whereas lower temperatures are associated with more extreme probabilities. These extremes can either be lower (e.g. 1011) or higher (e.g. 0101 with nearly 80 %) depending on whether the original probability $p_{\text{ref}}(z)$ is above or below the uniform level $\frac{1}{2^n} = \frac{1}{16}$. This latter statement does not completely hold as the normalization leads to additional non-linear effects.

We can now measure the entropy S of these resulting probability distributions (cf. Eq. (2.39)):

$$S[p] = \sum_x -p(x) \log\left(p(x)\right) \tag{4.13}$$

The entropy measures the disorder of the distribution. It reaches its maximum $\log(|\Omega|) = \log(2^n) = \log\big(2^4\big) \approx 2.77$ for the uniform distribution and is zero for a degenerate distribution with $p\left(\hat{x}\right) = 1$ with $\hat{x}$ being the certain outcome.

Using now the 200 measured distributions we can take a look at the time course of the entropy (cf. Fig. 4.2C). We see that higher entropy states are associated with higher temperatures and vice versa. We can calculate the expected entropy change from the *observed* reference distribution by defining the energy for all states as

$$E(z) = -\log\left(p(z)\right). \tag{4.14}$$

These energies can then be rescaled by the temperature T and we calculate the associated probability distribution via Eq. (2.25):

$$p_T(z) = \frac{1}{Z_T} \exp\left(-\frac{E(z)}{T}\right) \tag{4.15}$$

with the normalization constant $Z_T = \sum_y \exp\left(-\frac{E(y)}{T}\right)$ being different for each temperatures T and unity for the reference configuration at $T = 1$.

The observed entropy change tracks the expected variations well. In the cooling-down period the measured entropy drops slightly earlier than the calculation predicts. This likely originates in the fact that the distribution which we observe at the reference rate $r_{\text{exc}} = 2\,\text{kHz}$ is formed by the state of the synaptic inputs at this point in time. However, these are the result of the previous $\approx 3\tau_{\text{syn}} = 30\,\text{ms}$. As such it is slightly "colder" than the actual target distribution, which also explains the different point of lowest entropy $S \approx 1$. Longer periods P would suppress this effect.

On the level of the membrane voltages we see the effect of the different noise levels in the width of the membrane potential. We show the actual membrane potentials (rather than the effective ones without the spike mechanism) in Fig. 4.2D. The reset behavior of the spike mechanism makes the observation a bit more involved. For lower-bias neurons (lower pane) the effect is more easily identified. Here, for cold temperatures (around 1.4 s) the neuron's membrane potential is rather stable and significantly below the threshold $V_{\text{thresh}} = -50\,\text{mV}$ and the neuron does not fire at all. For higher temperatures (around

1.8 s) the variance increases, which leads to sporadic activity depending on the state of the random walk. For high-bias neurons (upper pane in Fig. 4.2D) the low-temperature phase results in higher activity, as the noise levels are not sufficient to suppress the leak-over-threshold bias and the neuron is near-constantly active. Here, higher temperatures are associated with *less* spike activity.

Discussion

We should point out that none of the arguments presented here, except for the calculations, depends on the noise being Poisson distributed or the time course being a sine. Both of these are choices made to make the model and simulations more accessible. However, any kind of stochastic input and associated change in its intensity will have a qualitatively similar effect. In practice, the assumption of Poissonian distributed "background" is justified as the sources are a stand-in for a diverse population of pre-synaptic neurons. Even though there will be some correlation the law of large numbers is going to apply.

The assumption of a sine-like modulation of the network activity directly contradicts the available data. Brain waves can of course be decomposed into sine modes, but their superposition forms a much different time course [Buzsaki, 2006]. However, this does not invalidate our model here as all it really requires is that the *reference* condition is sufficiently long observable, such that the network can adequately adapt to it. In our model this requires time periods on the order of the synaptic time constant τ_{syn} in which the network is exposed to adequate noise input. As we do not use the shape of the time course (except to determine the temperature) the same principles also hold for arbitrary time courses and the sine choice only eases implementation.

One potential application is the mixing problem that we briefly introduced in the beginning of this section. Here lower activity levels of the background would correspond to phases where we more clearly "see" a particular realization and higher levels help us switch between the available compatible interpretations of the world. E.g. whether the example in Section 2.2 (Fig. 2.5) did indeed show a rabbit or a duck. In the statistical literature there has been a lot of work done to precisely tune the tempering schedule for optimal sampling and how to ensure that the sampled distribution is an unbiased estimate of the underlying one [Nadler and Hansmann, 2007]. We did not investigate effects of such optimizations so far, and it is unlikely that we will achieve the necessary precision to benefit from those optimizations. However, it has been shown that tempering can help mixing in spike-based generative models of both handwritten digits and more complex data sets in a setup similar to what we will use in Section 5.1 [Korcsak-Gorzo, 2017].

4.2. A simple network: The Ising model

In Chapter 3 and, in particular, in Section 3.3 we discussed the dynamical response of a single neuron with respect to synaptic inputs extensively. What we, intentionally, delayed was a discussion about network level effects and discrepancies between the different models. For functionally relevant networks – of which we will discuss two incarnations on neuromorphic hardware in the next chapter (Chapter 5) – we are immediately faced with a huge variability both between networks of similar performance, but also between different neurons within said networks. This makes the identification and isolation of effects of different components a very subtle task. Rather than facing this challenge directly, we use the arguably simplest possible network topology, the nearest neighbor connected 2D-Ising network. The hope behind this was that it will be easy to isolate effects and explain their variations[8].

We will first introduce the classical Ising model in Section 4.2.1. In Section 4.2.2 we then present the concept of the critical temperature and in Section 4.2.3 we discuss the Curie law and hysteresis effect that we will look at throughout the rest of this chapter. Finally in Section 4.2.4 we relate the Ising model to the BM formulation and give the translation of the phenomena from Section 4.2.3 in this formulation. Much of this work was done before [Gürtler, 2018], therefore most of the remaining chapter will be focused on investigations of the Buesing neuron model (Section 2.2.3 albeit with exponential interaction shapes), rather than the LIF model. There is also a computational reason for this, as LIF simulations are significantly more expensive and, in particular, setting up different initial states is not easily possible within the simulation framework SBS [Breitwieser et al., 2020].

4.2.1. Setup

The d-dimensional classical (as opposed to the quantum) Ising model consists of a d-dimensional lattice of mini-magnets or spins[9]. Each lattice site i consists of one spin σ_i that can either point upwards or downwards. We model these as

$$\sigma_i \in \{-1, 1\}\,. \tag{4.16}$$

Nearest neighbor sites are connected via a coupling parameter J, which represents the interaction between the mini-magnets. For $J > 0$ it is energetically preferable for neighboring spins to be aligned and the model represent ferromagnetic material, for $J < 0$ it is vice versa and it represents anti-ferromagnetic material [Nolting, 2013]. Additionally each spin also couples to an, at least in principle, site-local external field h_i. This can be used to represent a lattice-external magnetic field, e.g. the earth magnetic field.

[8] This is how the author started his investigation nearly five years ago. With the expectation of "just" finding obvious results and be done within half a year, before moving on to implementations on hardware. As it turns out: "It's never just" - Thomas Robitaille, ca. 2013.

[9] In contrast to Section 5.2 here we only discuss classical spins.

A d-dimensional lattice of length l has $n = l^d$ spins in total. Each micro state $\sigma \in \{-1, 1\}^n$ of the lattice is assigned an energy

$$E(\sigma) = -J \sum_{\langle i,j \rangle} \sigma_i \sigma_j - \sum_i h_i \sigma_i, \tag{4.17}$$

where the first sum only runs over the set of nearest neighbors i and j. In a thermal bath each σ occurs with a temperature-dependent probability

$$p_T(\sigma) = \frac{1}{Z} \exp\left(-\frac{E(\sigma)}{T}\right), \tag{4.18}$$

with $Z = \sum_{\{\sigma\}} \exp\left(-\frac{E(\sigma)}{T}\right)$ being the temperature-dependent normalization constant, or partition sum. All the arguments from Section 2.2.1, why this sum quickly becomes intractable and (efficient) sampling methods are required, apply.

The global magnetization of our model system is then given by

$$M = \frac{1}{N} \sum_i \sigma_i = \langle \sigma \rangle, \tag{4.19}$$

which in an ensemble average sense is given by

$$\langle M \rangle = \langle \sigma_i \rangle_{p_T}. \tag{4.20}$$

We will stick largely to the 2-dimensional ferromagnetic variant as this can be solved analytically [Onsager, 1944] and forms a robust comparison point. The choice of dimensionality is solely encoded in the set of index pairs i, j contained in $\langle i, j \rangle$ and, as such, easy to change. In the language of connectivity matrices from before the nearest neighbor connections correspond to a very sparse matrix J_{ij} with non-zero entries only for i, j that are nearest neighbors. Finally, we choose periodic boundary conditions, i.e. we treat the right-most and the left-most as well as the top-most and the bottom-most lattice sites as neighbors throughout the rest of this chapter. These choices of periodic boundary conditions and the 2D network are taken in order to minimize the finite-size effects.

4.2.2. Critical temperature T_{crit}

The Ising model gained fame through its ability to reproduce the phenomena of spontaneous magnetization. A piece of ferromagnetic material acts as a magnet only if all (or at least most) of its constituent mini-magnets align into one direction. At this point the whole block can interact with other magnetic material. Taking such a magnet and heating it up will lead the single spins to misalign and the magnet to no longer being magnetic at a macroscopic level (see Fig. 4.3a for a sketch).

Interestingly enough the opposite is also working: Instead of realigning all spins by some external field one can also cool down the magnet below its Curie or critical temperature T_{crit} and the spins realign spontaneously. The Ising model was the first to explain

this already observed phenomena [Ising, 1925]. Onsager then calculated the critical temperature in terms of the coupling strength J for the 2D model [Onsager, 1944]:

$$\frac{kT_{\text{crit}}}{J} = \frac{2}{\ln\left(1+\sqrt{2}\right)} \approx 2.269. \tag{4.21}$$

The critical point itself is for the external field $h_i = 0\ \forall i$. We see already here that this is a scale-free system, as there is one free parameter between J, h and T, with only the ratio of $\frac{T_{\text{crit}}}{J}$ being special.

The point $(J = 1, T = T_{\text{crit}}, h = 0)$ of the 2D-Ising model is probably the single most well understood parameter point in any model in statistical physics and we will not be able to do a satisfactory treatment of it and refer to the textbook literature (cf. [Nolting, 2013]). Of particular interest are the divergences of the state variables around this critical point, which are characterized by the critical exponents. We will only treat the critical exponent γ, which describes the divergence of the susceptibility

$$\chi = \frac{\partial M}{\partial h}. \tag{4.22}$$

χ diverges at $(J = 1, T = T_{\text{crit}}, h = 0)$ when approached from higher temperatures $T > T_{\text{crit}}$ as

$$\chi \propto \left(\frac{T - T_{\text{crit}}}{T_{\text{crit}}}\right)^{-\gamma} \tag{4.23}$$

with $\gamma = \frac{7}{4}$ for $d = 2$, which can be derived analytically [Onsager, 1944].

While we can calculate these expectation values for the traditional Ising model as derivatives from the partition sum Z, it would require us to use the knowledge that Eq. (4.18) is exact. Since we want to extrapolate to Buesing and LIF neurons we do not have this knowledge and, in fact, do have to assume that there will be significant deviations. Therefore, we need to do these calculations in a more manual way and approximate the gradient by a finite difference:

$$\chi(T) = \lim_{\Delta h \to 0} \frac{\langle M(T, h = \Delta h)\rangle - \langle M(T, h = -\Delta h)\rangle}{2\Delta h} \tag{4.24}$$

where we need to choose the finite difference $2\Delta h$ to be a small external field change.

In Fig. 4.3b we see the temporal evolution of the magnetization of such an Ising system, initialized at high temperature $T = 4$ and slowly (adiabatically) cooling down. Fig. 4.3c-e presents the iconic 2D images of typical states along this evolution, with the down-state being shown in yellow and the up-state in purple. For a hot system ($T \gg T_{\text{crit}}$, Fig. 4.3e) the spins effectively do not see each other and the resulting activity pattern resembles white noise with a flat power spectrum. At the critical temperature (Fig. 4.3d) we see a completely scale-free system. This means that the size distribution of iso-magnetization patches follows a power law, which prevents us from making a statement about the system size, as the figure would look the same for all possible "zoom" levels. Finally, for a cold system (Fig. 4.3c) all spins are aligned and the total absolute magnetization is close to its maximum (Fig. 4.3b).

4.2.3. Curie law and hysteresis

There are essentially two degrees of freedom in the Ising model: One is the external field h and the other is a combination of the temperature T and the coupling parameter J. Of the latter two we can set one arbitrarily to unity, typically one chooses the coupling parameter $J = 1$ as this is a material dependent quantity and should not be subject to changes in either external fields h or temperature T, which are experimentally accessible. The dependence of the magnetization on both of these parameters, in a linear approximation, is described in the original Curie law (named after Pierre Curie (1859-1906))

$$M \propto \frac{h}{T} \tag{4.25}$$

Essentially this expansion is valid for small values of $|M| \ll 1$, which are generated for small magnetic fields and not too low temperatures. In its general form it reads

$$M = M_0 \tanh\left(C\frac{h}{T}\right) \tag{4.26}$$

with M_0 being the maximally achievable magnetization of the system (essentially scaling with system size) and C being some constant depending on the coupling strength J. We will only be looking at magnetization per site and therefore $M_0 = 1$ for us.

There are now two dynamical experiments that one can look at:

1. Cooling-down

   We can fix the external field $h_i = h$ for the system, initialize it at a high temperature $T \gg T_{\text{crit}}$ and cool it down adiabatically. In Fig. 4.3b this is shown for $h = 0$ and we expect the system to fluctuate around $M = 0$ for $T > T_{\text{crit}}$, with fluctuations increasing when the temperature approaches T_{crit}. When crossing into the sub-critical temperature regime $T \leq T_{\text{crit}}$ the system transitions into either $M \approx 1$ or $M \approx -1$ with a characteristic decay. For different values of the external field h we expect to see a continuous decay towards $M = \text{sign}(h)$ with reaching the extreme value $M = \pm 1$ for $T = T_{\text{crit}}$.

2. Oscillating external field

   On the other hand we can fix T to some value and change the external field h adiabatically in an oscillating manner. For $T > T_{\text{crit}}$ this simply traces the Curie law Eq. (4.26) and reproduces the activation function of a single unit up to some scaling factor. For $T < T_{\text{crit}}$ only $M = \pm 1$ are states that can be sustained. If the system happens to be in an $M = 1$ state, then we need to tune the external field to a significantly negative value $h < 0$ in order to overcome the internal bias generated by all the aligned states before the whole system switches into the $M = -1$ state. In order to switch back again a significantly positive external field $h > 0$ is required. This effect is called hysteresis and does not depend on the oscillation frequency which, if chosen too high also induces a dynamical lag. We need to be especially careful due to the finite time constants used in the Buesing and LIF sampling frameworks ($\tau_{\text{ref}}, \tau_{\text{syn}}$).

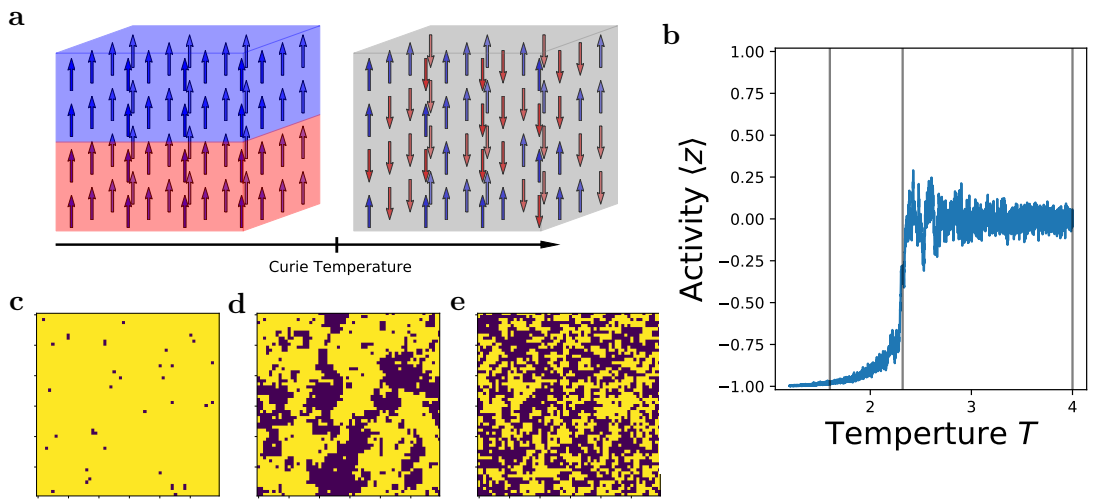

Figure 4.3.: **Ising model:** **a** Schematic representation of a physical magnet. Spins (small indivisible magnets, represented by arrows) are arranged on a regular lattice (here 3D, later 2D). Above the Curie temperature T_{crit} the single spins are randomly oriented resulting in a zero net magnetization. Below the Curie temperature the coupling between the spins dominates and all spins orient in the same direction, resulting in a macroscopic net-magnetization. **b** Mean activity $\langle z \rangle$ of the 2D system cooling down from $T = 4 > T_{\text{crit}}$ to $T = 1.2 < T_{\text{crit}}$. The increased fluctuations shortly above $T_{\text{crit}} \approx 2.27$ as well as the divergence shortly below it are characteristic for the system. **c-e** Exemplary states of the single spins (up in blue, down in yellow) of a 80x80 2D-system for different temperatures $T < T_{\text{crit}}$, $T \approx T_{\text{crit}}$ and $T > T_{\text{crit}}$. Far above the Curie temperature the orientation of the spins is random, far below the spin coupling dominates and essentially all spins are aligned. Around the Curie temperature (**d**) scale-free behavior can be observed and iso-magnetization areas of all sizes appear. For simulation parameters see Appendix B.4.

4.2.4. Connection to Boltzmann machines

So far the formulas for the Ising model are exactly identical to the ones we used to describe Boltzmann machines (cf. Section 2.2.1), except that we are now using $\sigma \in \{-1, 1\}$ instead of $z \in \{0, 1\}$ to describe the state of the single unit. This changes the achievable energy band of the states in Eq. (2.26) and Eq. (4.17) as well as their achievable mean activities. Since both these systems are classical two-state systems, which can all be related via a linear transformation, we can find an equivalent BM parameter set (w, b) for each set of Ising parameters (J, h) via:

$$w_{ij} = 4J_{ij} \tag{4.27}$$

$$b_i = 2h_i - 2\sum_{j=0}^{N} J_{ij} \tag{4.28}$$

and vice versa

$$J_{ij} = \frac{1}{4}w_{ij} \tag{4.29}$$

$$h_i = \frac{1}{2}b_i + \frac{1}{4}\sum_{j=0}^{N} W_{ij} \tag{4.30}$$

For states related by

$$z_i = \frac{1}{2}\left(1 + \sigma_i\right) \tag{4.31}$$

this results in energies differing only by a global constant $C = \frac{1}{2}\sum_{i,j}^{N} J_{ij} - \sum_{i=0}^{N} h_i$, which does not change the state-probabilities due to the normalization constant Z. For the proof of correctness of these calculation see Appendix A.

Eq. (4.28) and Eq. (4.30) give us that the $h = 0$ configuration requires, on the Boltzmann side, a fine-tuned configuration[10] between the coupling weight w and the bias b:

$$b_i = 2h_i - 2\sum_{j=0}^{N} J_{ij} = 0 - 2 \times 4J = -2 \times 4 \times \frac{1}{4}w = -2w \tag{4.32}$$

to compensate for the offset of the center of the activity scale with $\langle A \rangle = 0.5$. This compensation is trivial for standard Buesing neurons, but becomes tricky for different interaction kernels and LIF neurons. If it is only the scale of the interaction shape that we misjudged we would expect a constant offset from the predicted bias parameter b. Throughout the rest of this chapter we will use the bias offset $\Delta b = b + 2w$ as the stand in for the external field h. The scaling relation between these two is:

$$\Delta b = 2h \tag{4.33}$$

[10] Technically speaking, $h = 0$ is also a fine-tuning problem, but we can just *not* model h. Here, the additional practical problem is, that we do not know the "correct" value.

The movement to $w = 1$ also rescales the critical temperature by the weight-rescale factor of 4 from Eq. (4.28) with the resulting critical temperature being:

$$T_{\text{crit}}^{BM} \approx 0.567 \tag{4.34}$$

Lastly we need to find equivalents of the magnetization:

$$M = \langle \sigma \rangle_{p_T} \in [-1, 1] \tag{4.35}$$

with the exceptional point of vanishing external field $h = 0$ corresponding to $M = 0$. On the Boltzmann side this is the mean activity

$$A = \langle z \rangle_p \in [0, 1] \tag{4.36}$$

where the bias-free configuration implies $A = 0.5$. In order to more closely align with the magnetization from the original magnet formulation, we introduce the offset activity

$$\Delta A = A - 0.5. \tag{4.37}$$

There is one additional problem we side-stepped so far: Starting with the Buesing neuron all our models have some form of explicit temporal dynamics that the original Ising model and Gibbs sampling[11] do not have. These increase the effect of the finite system size. In the original formulation, where the spins where allowed to flip in each update step, the most a single flip could affect was the neighboring (directly connected) site. With the introduction of refractory periods τ_{ref} a flip into the $z = 1$ state keeps influencing its surrounding spins for the next τ_{ref} update steps. If the neighbor also spikes, than this original spike can propagate throughout the complete system if the interaction strength w is sufficiently strong before the original unit is released. At least for systems that are smaller than τ_{ref} along one dimension we use $l = 80$ and $\tau = 100$ for most of our simulations here. In practice, at least for the functional networks in Chapter 5, this is not our biggest concern as we do not need to go to very strong interactions or equivalently very low temperatures. The two statements are equivalent as it is only the ratio $\frac{w}{T}$ that defines the dynamical effect.

[11] Which is the equivalent to Glauber dynamics for spins.

4.3. Phase diagram of Ising-like networks with Buesing neurons

In the translation from the parameter set of the Ising model into our Boltzmann parameters (cf. Section 4.2.4) we established that we need to introduce a coupling between w and b in order to compensate for the asymmetry of the Boltzmann definition $z \in \{0, 1\}$ (cf. Eq. (4.32)). And in Section 2.2.4 and Section 3.3 we showed that the translation from the Boltzmann regime into the LIF regime is not overly reliable. In other words, we do not know the *correct* scaling between biases and weights, even at a single input level. While learned in Section 4.1 that temperature scaling seems to work as expected (in the high-temperature limit), here we will find unexpected behavior for this particular network configuration. We will just check the complete 2D parameter space of $\Delta b, T$ (cf. Eqs. (4.18) and (4.33)) in order to compensate for our lack of a priori knowledge about the parameterization.

In this section, we will first introduce the phase diagram by reproducing it for the original Buesing model. At this point we will learn that while Buesing is able to reproduce the high-temperature part of the phase diagram as expected, for the low-temperature part the initialization – and therefore the internal state ζ – becomes important. We demonstrate how to recover the critical exponent γ from the 2D phase diagram and how to relate said diagram to the dynamical experiments demonstrating the Curie law and hysteresis effect. From this point we transition to exponentially shaped interactions and learn that the phase diagram looks significantly different, yet still allows to recover the correct critical exponent (Section 4.3.2). Finally we distinguish the effects added by the overshoot of the interaction versus the additional tail, where we find that the most significant contribution comes from the tail part (Section 4.3.3).

4.3.1. Rectangular interaction

As Buesing et al. [2011] have proven that rectangular PSPs sample Boltzmann distributions faithfully, we can use it to explore the "correct" behavior with respect to the 2D scan of the external field proxy Δb and the temperature T. For each data point in Fig. 4.4d we initialize the network in a single micro state, randomly chosen with each single site having 0.5 probability to be $z = 1$ independent of the other sites. In a way we quench from an artificial state to the target parameter set. After quenching to the target we let the network run for a sufficient amount of time to measure the mean of the activity level (see Appendix B.5 for parameters).

For positive biases $b > 0$ the mean activity of the network is increased. The amplitude of this increase decreases with higher temperatures T. For negative biases the same holds with inverted sign. The two dashed lines in Fig. 4.4d) mark the activity levels of $A = 0.49$ and $A = 0.51$ which we use to calculate the susceptibility χ Eq. (4.24).

Moving along an iso-Δb line (horizontally in Fig. 4.4d) corresponds to the cooling-down of the experiment we used for measuring the Curie law. However, the data in Fig. 4.4b is not simply the data from a horizontal line but rather from additional experiments where the network is initialized again at high temperature and then cooled down adiabatically. For zero bias offset Δb, which corresponds to $h = 0$, we recover the expected behavior

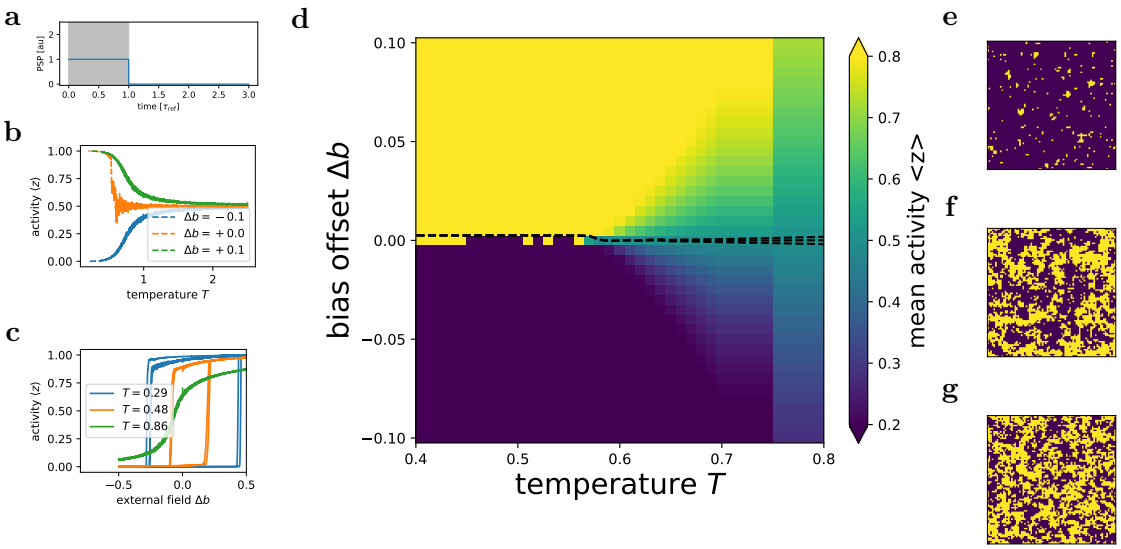

Figure 4.4.: **Rectangular interactions:** **a** Shape of the interaction of a standard Buesing neuron with refractory period $\tau = 100$. Curie law **b** and hysteresis **c** experiments with an 80x80 2D-Ising network. **d** Full phase diagram, comprising the temperature T and the external field h, here represented by the bias offset Δb (for the relation see the main text Eq. (4.33)) **e-g** Typical states along the $h = 0$ line for the network. These are in good agreement with Fig. 4.3. Simulation details can be found in Appendix B.5.1.

from Fig. 4.3b, with the spontaneous symmetry breaking at $T = T_{\text{crit}}$. This corresponds to the cooling of a ferromagnetic metal to generate a magnet experiment.

For higher and lower bias offsets $\Delta b = \pm 0.1$ we find the expected divergence according to the Curie law Eq. (4.26). Note also that the plotted ranges in the Curie plot are significantly larger than the range shown in the phase diagram. For visibility reasons we clip the color coding at 0.2 and 0.8 for the phase diagram.

Finally, we show three typical states from the $A = 0.5$-line which we will use to define $h = 0$ in the later configurations. For sub-critical temperatures $T < T_{\text{crit}}$ we define the $A = 0.5$-line as the line at which the $A = 1$ and $A = 0$ phases meet. For high temperatures $T > T_{\text{crit}}$ we see the quasi white-noise behavior (Fig. 4.3e). Since we do not use as high temperatures T as in Fig. 4.3 some correlations are still observable. At the critical point (Fig. 4.3f) we again see the quasi scale-free behavior and at low temperatures (Fig. 4.3g) we again see the total magnetization of the system.

The origin of the quench matters

Fig. 4.5a-c show the same phase diagram as Fig. 4.4d, except for the initial condition. We can think about this as a quenching experiment. The initial state of the system is

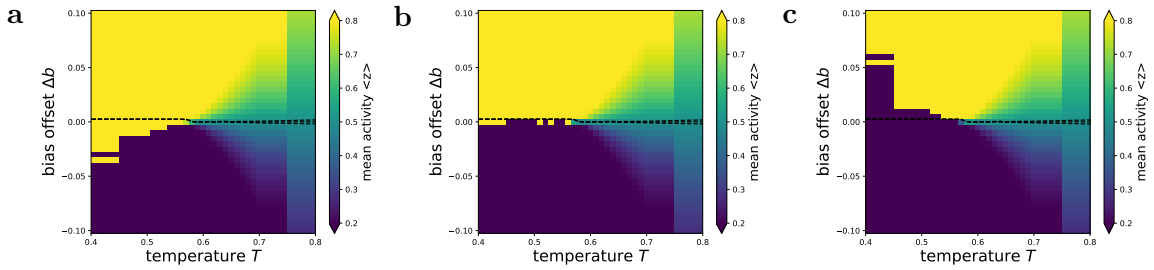

Figure 4.5.: **Initial conditions - Quench origins:** Phase diagrams for two different initializations as compared to Fig. 4.4d: **a** Initial conditions chosen as $\zeta \propto U(0, 2\tau)$. **b** Initial conditions from $T = T_{\text{crit}}; h = 0$ from Fig. 4.4d. **c** Initial conditions from $T = 0.5; h = -0.1$ from Fig. 4.4d. We see some dependence on the initial conditions for the sub-critical regime $T < T_{\text{crit}}$ where initial conditions cannot be never forgotten. Simulation parameters can be found in Appendix B.5.2.

the result of some preparation and at the beginning of the simulation the parameters are changed instantaneously, or quenched, to the target values for T and Δb. Depending on the initial condition this introduces a different set of biases. For Fig. 4.5a all $z = 1$ neurons are initialized with $\zeta \sim U[0, \tau)$ and all $z = 0$ neurons with $\zeta \sim U[\tau, 2\tau)$. The initial condition for Fig. 4.5b is a micro state at the critical point (cf. Fig. 4.4f) with the correct ζ-distribution. To show that for small temperatures $T < T_{\text{crit}}$ the initial bias can be sustained we show the initial condition for a low temperature and negative external field $\langle A \rangle \approx 0$ in Fig. 4.5c.

The observed variations only happen for sub-critical temperatures $T < T_{\text{crit}}$. This is a consequence of the diverged autocorrelation time scale. Intuitively speaking this means, that the system is unable to forget its history, which in this case is completely encoded in its initial condition of the simulation or in experimental terms: The quench origin. Effectively a non-natural state introduces a local bias. For high temperatures this bias is short lived and gets forgotten on the autocorrelation time scale of the system, which in this case corresponds to the autocorrelation time scale of the single units. For lower temperatures (approaching T_{crit}) the autocorrelation time on the system level starts to increase and diverges at $T = T_{\text{crit}}$. We choose our simulation times long enough to not be bothered by these initial offsets for $T > T_{\text{crit}}$. In other words, we do not resolve the critical point clearly enough to see significantly increased, but not yet diverged autocorrelation times.

For $T < T_{\text{crit}}$ this time scale has already diverged and hence the system cannot forget the bias of its initial condition which now competes with the external field. If the external field is strong enough to overcome the initial bias, the system will fall into the same final configuration (class) than the original Glauber dynamics [Glauber, 1963] would produce. However, for (comparatively) small external fields h (or bias offsets Δb) we find the

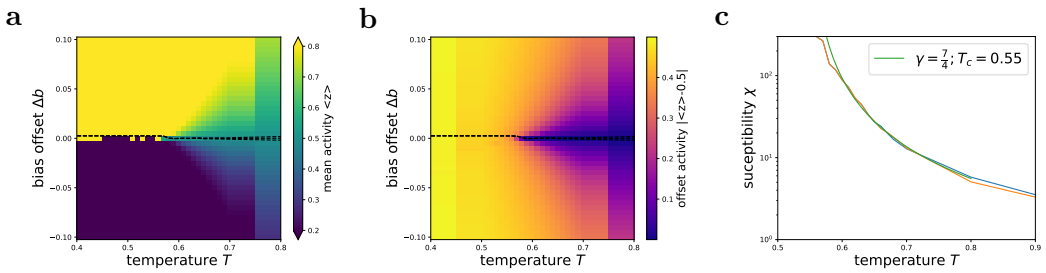

Figure 4.6.: **Recovering the critical exponent γ: a** phase diagram from Fig. 4.4d. **b** Activity difference to $\Delta A = A - 0.5$ (corresponding to $h = 0$) for the simulations from **a**. Dashed lines mark the $\Delta A = \pm 0.01$. **c** Upper (blue) and lower (orange) estimate for the susceptibility $\frac{\Delta z}{\Delta b}$ as defined by the finite difference measurements from **b**. The divergence at $T_{\text{crit}} \approx 0.55$ happens with the expected critical exponent $\gamma = \frac{7}{4}$.

situation that the local bias of the initial condition is stronger than the influence of the external field. In these cases we find $A = 0$ for positive external fields $h > 0$ (Fig. 4.5d) and vice versa $A = 1$ for negative external fields $h < 0$ (Fig. 4.5b) depending on the bias in the initial conditions. The expected behavior would be a random drop into either $A = 1$ or $A = 0$ for the exact case of $h = 0$. This we only recover when we use a state at the critical point as the initial condition (Fig. 4.5b). In a way this seems to be the most "natural" initial condition that obeys $A = 0.5$ for sub-critical temperatures $T < T_{\text{crit}}$.

Recovering the exponent

In order to determine the critical exponent γ first we need to find the $A = 0.5$ (corresponding to $h = 0$) line. For the rectangular interactions this seems like a pointless exercise (and it is), as we know the correspondence of bias offset Δb to h exactly. But we will treat this as a test run for the other interaction shapes later on and find the $A = 0.5$ line empirically in Fig. 4.6a. For ease of visualization we also plot the activity offset ΔA which acts as the direct proxy to the absolute magnetization $|M|$ in Fig. 4.6b. The purple $\Delta A = 0$ valley here lies at constant bias offset $b = 0$ as expected.

In order to now calculate the susceptibility, we need to find the bias offsets $b(T)$ such that the mean activity difference is a fixed small number $|\Delta A| = \delta A$. When choosing this small offset we need to trade between two different goals: On the one hand we require a strong enough signal to actually measure the resulting bias difference, on the other hand we need to stay small enough that the linear approximation in Eq. (4.24) still holds.

Having done this we can plot the susceptibility estimators from above (blue) and below(orange) for $\delta A = \pm 0.01$ in Fig. 4.6c. The measured results are in good agreement

with the expected divergence with the critical exponent γ:

$$\chi \propto \left(\frac{T - T_{\text{crit}}}{T_{\text{crit}}} \right)^{-\gamma} \tag{4.38}$$

with a critical temperature $T_{\text{crit}} \approx 0.55$, which is in good correspondence with the value calculated in Section 4.2.2 after we translate between the spin and the neuron domain (cf. Section 4.2.4; $T_{\text{crit}} = 0.567$). The remaining variations are most likely due to finite-size effects, which are somewhat exasperated by the long refractory period of $\tau = 100$.

So, to summarize what we learned so far, except for the slight hick-up with the additional complexity of the differently value $z = 1$ states, the proof from [Buesing et al., 2011] (unsurprisingly) turned out to hold also in practice. But we also learned in Section 3.3 that, even for single units, standard Buesing neurons with rectangular PSPs show markedly different behavior from LIF neurons. Let us now see what happens if we change the shape of the interaction between the neurons:

4.3.2. Exponential interactions

The closest we can get[12] should be by augmenting the Buesing neuron with exponential-shape PSPs (cf. Eq. (2.62), Fig. 4.7a).

This leads to massive changes in the phase diagram as shown in Fig. 4.7d. Not only do both the temperature scale (x-axis) as well as the bias offset (y-axis) change significantly, it is also the general form that is completely different from the $|\Delta A| = 0$ for non-constant bias offsets Δb.

The easiest thing to explain is the rescaled x-axis or temperature. This effect goes back to our translation scheme between LIF parameters W and Boltzmann parameters w (cf. Eq. (2.80)). We did essentially postulate that all of the interaction happens within the refractory period τ_{ref} and ignored the contributions of the tail. Therefore it is not surprising that we underestimate the effectively implemented w for a given utilized parameter W. As the temperature T is only meaningful in reference to the scale of the weights w and biases b (cf. Eq. (4.21) and Eq. (4.32)) a shift in the absolute value of the temperature T is to be expected. In fact, we can even use this as an empirical measure of what the factor of mismatch is, as the critical temperature is characteristic for the system on a fundamental level.

We also find that the critical point is shifted towards more negative bias offsets ($b \approx -0.4$). This shift has the same origin, i.e., the non-refractory contribution of the interaction. As the Ising network consists of solely excitatory connections, all the extra PSP mass (cf. Section 3.1.2) is excitatory, which we need to compensate for (if we want to keep the mean activity constant) via an additional negative bias offset $\Delta b < 0$. Therefore the tail of the PSP – which we did not account for in the translation (cf. Section 2.2.4) – offers a complete explanation of the shift of the critical point.

However, we see more complicated changes, as also the $h = 0$ line for higher temperatures does not correspond to a horizontal line (constant bias offset Δb). This is surprising

[12] Without doing the whole LMM treatment of adding a shadow membrane potential distribution that continues evolving within the refractory period, cf. u^{eff} Section 2.2.4.

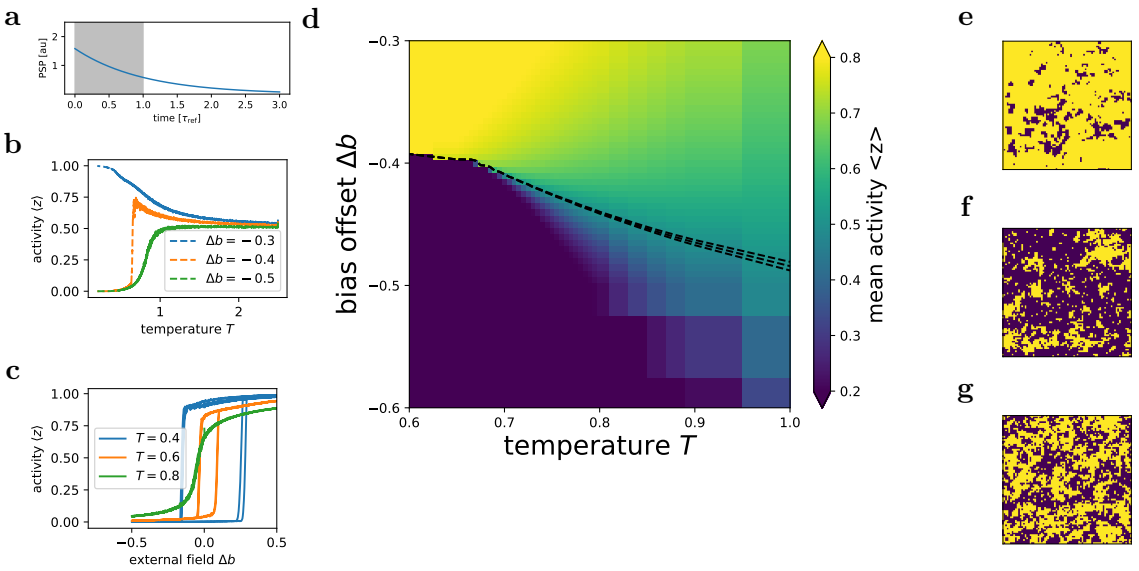

Figure 4.7.: **Exponential interactions:** **a** Shape of the exponential interaction used for a Buesing neuron with refractory period $\tau = 100$. Curie law **b** and hysteresis **c** experiments with an 80x80 2D-Ising network. The hysteresis simulations have an external field offset of $\Delta b = -0.4$. **d** Full phase diagram, comprising the temperature T and the external field h, here represented by the bias offset Δb (for the relation see the main text Eq. (4.33)) **e-g** Typical states along the $h = 0$ line for the network. These are in good agreement with Fig. 4.3. Simulation details can be found in Appendix B.5.3.

as we *only* change the width of the activation function, and not the scaling between the weight parameter w and the bias parameter b. As such we would not expect a shift in activity (beyond the Curie law as discussed in Section 4.2.3). There is, however, the point of the different fractions of PSP mass U that is contained in the tail and within the refractory period that we discussed around Fig. 3.2. As we shift the activity level[13] A we *also* change the meaning of the weight parameter w (cf. Section 3.1.2). This latter effect allows us observe more interesting phenomena when changing the "temperature" of the system.

In the cooling-down experiment (cf. Fig. 4.7b) we find asymmetric behavior for positive and negative bias offsets (with respect to the critical point). For more negative

[13] In fact the activity level alone is not enough, as for hysteresis experiments we would have *phases* of $A = 1$ and $A = 0$ respectively. In neither case there is a significant fraction of tails visible. It is only the stochastic firing that generates significant tail contributions. However, for the configurations with a single initial quench and then static parameters, the mean activity level over time is representative for the complete simulation. There is a slight deviation in the immediate vicinity of the critical point where the activation levels do also fluctuate over time.

offsets we find a sharper transition at higher values T than predicted by the Curie law Eq. (4.26). For higher values we fine a smoother transition, with the rise in A at detectable levels for significantly higher temperature values $T > 1.5T_{\text{crit}}$. Taking the offset value Δb of the critical point we see a completely unexpected cooling-down behavior. Namely we see that the activity A is *always* increasing for values of $T > T_{\text{crit}}$, which for the original Ising model would indicate the presence of a positive external field $h > 0$. Therefore, one would expect the system to end up in the $A = 1$ state. However, the system deterministically ends up in the $A = 0$ state. This behavior is rooted in the increased tail contributions, which increases the input that is required to keep the network active. This is similar to the pathological case of failure to mix that we discussed at the end of Section 4.1.1. As such momentary activity interruptions are a much more volatile process when compared to the original Ising formulation.

For sub-critical temperatures $T < T_{\text{crit}}$ we can again introduce different phase diagrams depending on the initial conditions. Following the discussion around Fig. 4.5 we initialized the system with a state close to the critical point (cf. Fig. 4.7f). Finding the exact critical point becomes somewhat trickier here as the system again tends to be a bit more bursty (cf. Section 3.3), making the behavior in the immediate vicinity more unstable and requiring increased simulation times. For typical states for higher and lower temperatures the question now becomes non-obvious what it means to "only change the temperature".

Note 2. *This is* not *the same biasing that we discussed Section 4.1.2. At that point we discussed a bias that was generated by an offset of the activation function, which was caused by changes in the background activity. Here the temperature T directly manipulates the width – and only the width – of the activation function.*

There are two ways to "define" temperature change:

1. Neuron level: Change the width parameter of the activation function

   This results in the cooling-down experiments from Fig. 4.7b, where we get the modified behavior due to the differently prominent tail contributions.

2. Network level: Follow the $\Delta A = 0$ line

   This results in nearly the same results as the original Ising formulation (cf. Fig. 4.3b). There are variations due to our finite parameter resolution but these are technical in nature and with sufficient compute power and interpolation to select an appropriate $b - T$ line one would be able to reproduce the intended behavior.

If we choose the latter definition the meaning of "cooling" down becomes more complicated: We now need to scale T and b co-dependently. But we can recover the original behavior.

Before we recover the critical exponents let us quickly mention the hysteresis experiment as well: Here we do *not* see fundamental differences in the form. We can already expect this from the phase diagram, along a vertical line (constant T) the resulting activities look just like in Fig. 4.4. It is only the relative position of neighboring "columns"

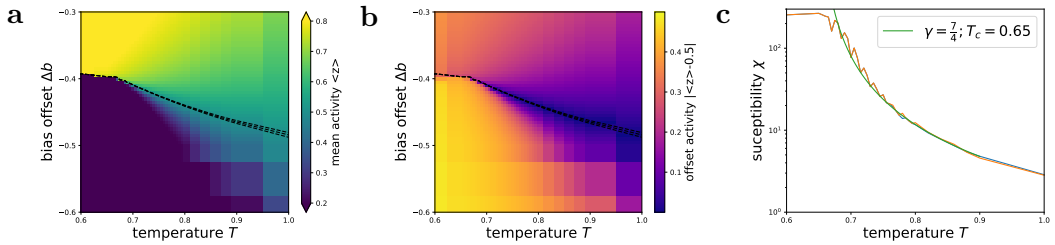

Figure 4.8.: **Recovering the critical exponent γ: a** Phase diagram from Fig. 4.7d. **b** Activity difference to $\langle z \rangle = 0.5$ (corresponding to $h = 0$) for the simulations from **a**. Lines mark the $\langle z \rangle = 0.49$, $\langle z \rangle = 0.5$ and $\langle z \rangle = 0.51$ levels. The $\langle z \rangle = 0.5$ line corresponds to an involved relation between the bias V_l and temperature T. **c** Susceptibility $\frac{\Delta z}{\Delta b}$ as defined by the measurements from **b**.

that is shifted. As such the precise values of offset biases required to flip from $A = 1$ to $A = 0$ and vice versa are changed and for different temperatures a different offset is needed, but the principal form stays the same (cf. Fig. 4.7c) .

Critical exponent

In Fig. 4.8b we again show the absolute activity difference to the balanced state $\Delta A = |A - 0.5|$. While the $\Delta A = 0$ line is now a more complex function of $\Delta b(T)$ rather than simply a horizontal line, we can still find it empirically. For sub-critical temperatures, this does not work, as we do not *actually* know the bias we are inducing by choosing a particular initial state (cf. Fig. 4.5). While one could in principle overcome this by doing an average over many initializations, we need to point out that the space of possible states is now $\{0, 1, 2, 3, \dots\}^N$ rather than $\{0, 1\}^N$ as our neurons retain information about their past, which has actual meaning beyond the refractory period and therefore their state variable z. This is the main reason why we limit ourselves to the discussion of the critical exponent γ. Following the argument from Eq. (4.24) and Fig. 4.6 we again mark the $\Delta A = \pm 0.01$ activity levels in Fig. 4.8b and estimate the susceptibility χ from the bias difference between these levels as a function of temperature. The resulting data shows the same kind of divergence as in the rectangular case (cf. Fig. 4.8c), with the exponent $\gamma = -\frac{7}{4}$ and a critical temperature $T_\mathrm{crit} \approx 0.65$. The saturation is an artifact due to the finite resolution of both the bias b and the activity difference δA.

Changing the effective temperature of this Ising-like connected network requires a different coordination between the width of the activation function, which is the actual adjusted parameter T and the offset bias Δb. Remembering the discussion for an LIF based implementation in Section 4.1.2 this would not introduce additional complexity as the excitatory and inhibitory noise inputs need to be fine tuned with respect to each other anyways. Nevertheless, we need to stress again, that these kind of Ising-like networks are not really our primary concern, as we typically are interested in the information processing capability of the brain. However, in order to be able to make

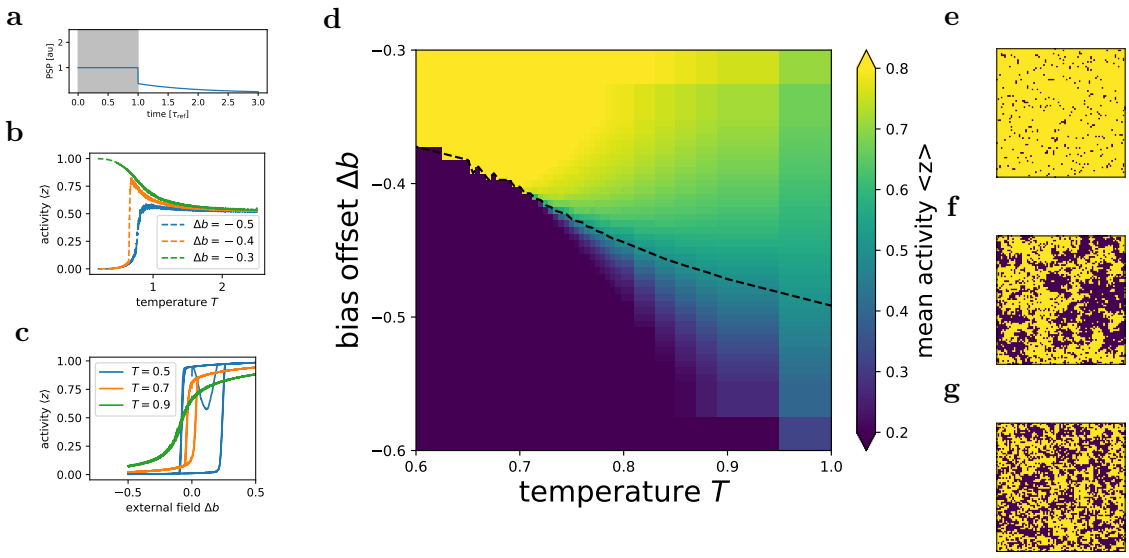

Figure 4.9.: **Tail interactions:** **a** Shape of the exponential interaction used for a Buesing neuron with refractory period $\tau = 100$. Curie law **b** and hysteresis **c** experiments with an 80x80 2D-Ising network. The hysteresis experiments have an external field offset of $\Delta b = -0.4$. **d** Full phase diagram, comprising the temperature T and the external field h, here represented by the bias offset Δb (for the relation see the main text Eq. (4.33)) **e-g** Typical states along the $h = 0$ line for the network. These are in good agreement with Fig. 4.3. Simulation details can be found in Appendix B.5.4.

confident statements about more complex systems we should be able to deal with these, at least conceptually, simple systems first.

4.3.3. Origin of the differences

There are two different possible possible origins for the observed deviations. On the one hand there is the different *form* of the interaction shape within the refractory period, on the other hand there is the *additional* tail. It turns out that the tail contribution is the more significant source of variations and we will therefore start with it.

Tail contributions

Simply adding the exponential tail of the interaction to the standard rectangular interaction results in the κ_{tail} from Eq. (2.64) (cf. Fig. 4.9a). The resulting phase diagram (Fig. 4.9d), at least qualitatively, shows the same features observed for the complete exponential interactions (cf. Fig. 4.7). The critical point lies at roughly ($T_{\text{crit}} \approx 0.73, b_{\text{crit}} \approx -0.52$). This is a slightly higher temperature indicating, that

the interaction strength transferred via κ_{tail} is slightly stronger than the one transferred by κ_{exp}. The $\Delta A = 0$ line shows the same qualitative behavior as for the full exponential kernel, with again only minor quantitative deviations due to the different normalization of the interaction kernels.

For the cooling-down experiments (Fig. 4.9b) for constant bias offset b we see the same, and even slightly more pronounced, asymmetry between offsets above and below the critical point that we already observed for κ_{exp}. The transitions below are again much sharper than the transitions for positive external fields $\Delta b > \Delta b_{\text{crit}}$. The transition through the critical point shows similar behavior as the one shown in orange, with a slight up tick above T_{crit} and a sharp drop at the $T = T_{\text{crit}}$ point to $A = 0$.

Letting the external field oscillate for a fixed temperature again show the expected hysteresis time course. There is one special case observable in Fig. 4.9c that we have not seen so far and that is the transient drop in activity A on one of the downward slopes of the oscillation for the cold temperature $T = 0.5$: In this particular case a large proto-Weiss domain[14] of $z = 0$ neurons forms and only breaks down after some time to rejoin the $z = 1$ region. This is a transient effect that we would be less likely to observe for slower cooling schedules and more likely for significantly larger systems. In the latter case the effects would, however, not be that prominent as it would be only a smaller *fraction* of neurons that form the "other" iso-magnetization domain.

For parameters along the $\Delta A = 0$ line we can again find typical states that show the same behavior as expected. For low temperatures the system forms a single iso-magnetization domain with only single neurons spontaneously switching into the wrong state but quickly returning to join the their neighbors (cf. Fig. 4.9e). For higher temperatures we again see a close-to-white-noise state, which is only limited by a) the finite size and b) the relatively low temperature $T > T_{\text{crit}}$ that we use here (cf. Fig. 4.9g). Around the critical point the typical quasi scale-free behavior is observable, where again size and parameter precision limit the model (cf. Fig. 4.9f).

Recovering the critical exponent would work similarly to the one shown in Fig. 4.8, where we again find the divergence in good agreement with $\gamma = \frac{7}{4}$ (data not shown).

Restricted exponential interaction

The other component of the interaction that changes is the form within the refractory period. Eliminating the tail of the exponential interaction results in κ_{cuto} from Eq. (2.63) (cf. Fig. 4.10a). Since we identified the difference in tail mass as the reason for the additional temperature dependence of the bias and therefore the non-horizontal $A = 0.5$ line, we expect this not to occur for such a restricted interaction. The zero-external field line (dashed line in Fig. 4.10d) however is still not at constant bias b. This is due to the non-linearity of the transfer function $\sigma(u)$ (cf. Eq. (2.58)) which we implicitly assumed when we made the original translation in Section 2.2.4. This here is not LIF but the same argument holds. Nevertheless the variations are significantly smaller and, in particular, non-monotonous such that the claim that the lion share of the variation originates in the

[14] It is not really a Weiss domain, as these are *stable* iso-magnetization regions and our simulated systems are far too small to sustain this kind of stability for multiple regions.

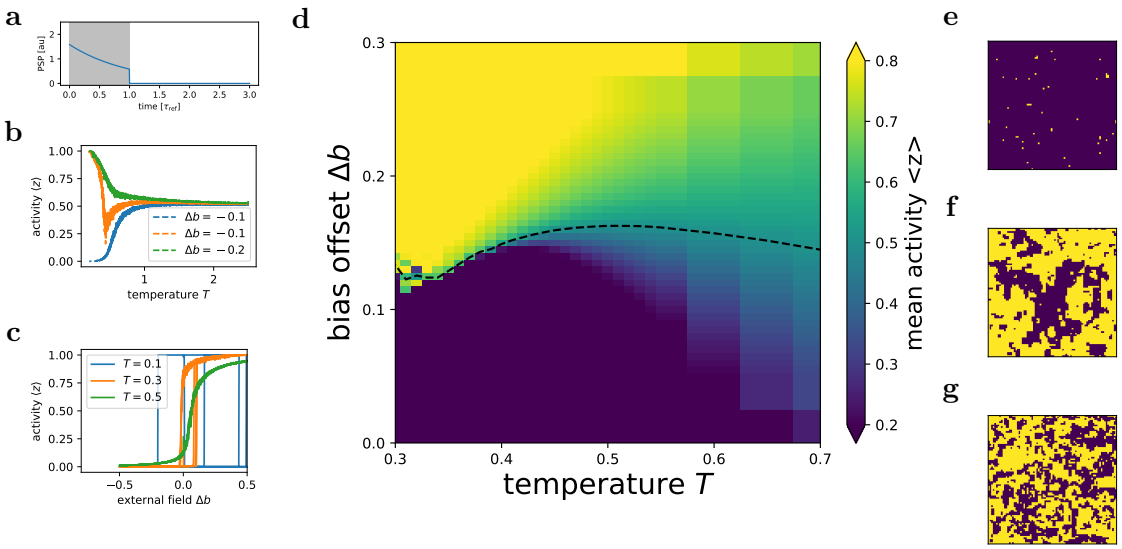

Figure 4.10.: **Restricted interactions:** **a** Shape of the exponential interaction used for a Buesing neuron with refractory period $\tau = 100$. Curie law **b** and hysteresis **c** experiments with an 80x80 2D-Ising network. The hysteresis experiments have an external field offset of $\Delta b = 0.1$. **d** Full phase diagram, comprising the temperature T and the external field h, here represented by the bias offset Δb (for the relation see the main text Eq. (4.33)) **e-g** Typical states along the $h = 0$ line for the network. These are in good agreement with Fig. 4.3. Simulation details can be found in Appendix B.5.5.

tail is still valid. There are other variations such as the general vicinity of the critical point being shifted upwards to positive bias offsets $b > 0$ and lower temperatures T. This is again expected as neurons now exhibit less interaction as compared to the complete exponential trace κ_{exp} (cf. Fig. 4.7a).

There is a problem with locating the critical point for this interaction. Before we *always* had the situation that, for low enough temperatures T the bias delta required to shift from the final state $A = 0$ to the final state $A = 1$ would converge to zero. This is not the case here. The reason for this can be seen in the hysteresis experiment (cf. Fig. 4.10c), where, even for the coldest experiment ($T = 0.1$) there are early transitions from $A = 1$ to $A = 0$. This is due to the active state being much more fragile than the inactive state as the system cannot generate a "buffer" of excitatory interaction (via the tails) and is at its weakest point at the end of the refractory period. As the overshoot in the beginning induces some clustering of the spike times, it is more likely for neighbors to spike at similar times. For rectangular PSPs κ_{rect} this is not a problem as the interaction strength does not depend on the time since the last spike ζ but only on the state z. Here, similar original spike times, mean weaker excitatory interaction when the first neuron

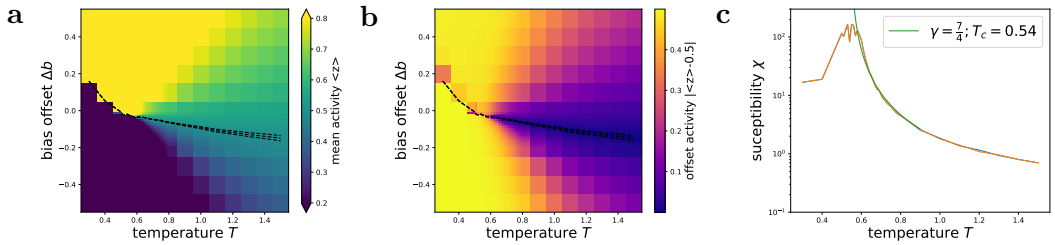

Figure 4.11.: **Critical exponent γ for LIF networks:** **a** Full phase diagram of an Ising-like network with CUBA LIF neurons. Temperature change is implemented similar to Fig. 4.1 for slightly different parameters (cf. Appendix B.5.6). Parameter translations according to Eqs. (2.78) and (2.80) and Eq. (4.28). **b** Activity difference to $A = \langle z \rangle = 0.5$ (corresponding to $h = 0$) for the simulations from **a**. Lines mark the $\langle z \rangle = 0.49$, $\langle z \rangle = 0.5$ and $\langle z \rangle = 0.51$. **c** Susceptibility $\chi = \frac{\Delta A}{\Delta b}$ as defined by the measurements from **b**. The divergence is in good agreement with the expectations.

becomes available again. Therefore, it is more likely to *not* fire again and switch to the $A = 0$ case. Since the initial overshoot will tend to *synchronize* the network, the weakness of the interaction becomes stronger over the course of the simulation time and as such the likelihood of falling out of $A = 1$ into $A = 0$ increases. As such we will always find some values of the bias offset b such that this happened somewhere in the middle of the simulation and therefore have a mean activity of $A \approx 0.5$. This is not the same as being at the critical point! But unfortunately it costs us the easy identification T_{crit}.

For a cooling-down experiment we now are able to find bias levels b for which the activity first decreases even though the network ends up in the $A = 1$ state (orange in Fig. 4.7b). The asymmetry for the transition though is still somewhat sharper on the negative bias offset side. However, recovering the critical exponent here is not straightforward.

4.4. Phase diagram of Ising-like LIF networks

Finally we discuss the phase diagram of networks of LIF neurons. We use the same parametrization as originally proposed by [Petrovici et al., 2016] and discussed in Section 2.2.4, with $\tau_{\text{syn}} = \tau_{\text{ref}} = 10\,\text{ms}$. In order to reduce the drift times τ^b at the end of a refractory period, and thereby improve the symmetry of the activation function, we reduce the difference between the reset V_{reset} and the threshold potentials V_{thresh}. However, the exact choice of the neuron parameters is of little importance as long as an effective high-conductance state (HCS), as measured by the (effective) membrane time constant τ_{m}, is achieved. The complete set of parameters can be found in Appendix B.5.6.

In order to implement the temperature change, we need to choose the rates of the Poisson spike sources such that the mean of the activation function stays fixed (cf.

Section 4.1). We measure activation functions for excitatory and inhibitory rates in a range from 100 Hz up to 25 kHz. On the lower end of this range the membrane potential does not implement an Ornstein-Uhlenbeck process anymore. We choose our translation point, which fixes $T = 1$ at $r_{\text{exc}} = 5.13\,\text{kHz}$ and $r_{\text{inh}} = 5.50\,\text{kHz}$. As the width of the activation function α scales as:

$$\alpha \propto \sqrt{r_{\text{exc}} + r_{\text{inh}}} \tag{4.39}$$

this gives us an effective temperature range of $0.25 < T < 2.1$, which is about an order of magnitude and enough for the comparison to the phase diagrams of the Buesing neurons with exponential interactions (cf. Section 4.3.2). We translate the parameters from the Boltzmann regime (w, b) into the LIF neuron parameters (V_l, W) according to Eqs. (2.78) and (2.80) and the activation function for our $T = 1$ configuration. In order to adjust for offsets we again use the bias offset Δb to implement an effective external field h.

If we choose this bias offset such that the mean activity is $\langle A \rangle = 0.5$ and thereby effectively implementing $h = 0$ we can again recover the correct critical exponent $\gamma = \frac{7}{4}$ around the critical point at $T_{\text{crit}} \approx 0.54$ and bias offset $\Delta b \approx -0.17$ (cf. Fig. 4.11). Unlike the Buesing neurons with exponential PSPs (cf. Fig. 4.7d) there is a continuous line of $\langle A \rangle = 0.5$ simulations throughout the sub-critical temperature region (cf. Fig. 4.11b). This is due to the network switching from the initial $\langle A \rangle \approx 1$-state into the $\langle A \rangle \approx 0$-state throughout the simulation (similar to Buesing with κ_{cuto}). As such the *shape* of the sub-critical phase boundary changes with the length of the simulation and is therefore not a model specific result. Our network retains the ability to forget its initial conditions for a positive initialization at $\langle A \rangle > 0.5$.

The unevenness of the lines of constant activities for supra-critical temperatures in Fig. 4.11a are an artifact of the reduced simulation time and network scale. We only use a 40-by-40 grid of neurons and fewer parameter configurations in order to constrain the required computation time. The temperature course of the bias offset $\Delta b(T)$ such that $\langle A \rangle = 0$ shows less dramatic variations when compared to Buesing neurons with exponential interaction kernels $\kappa(\zeta)$. It is close to a constant offset starting at about $\Delta b \approx -0.175$ at the critical temperature $T_{\text{crit}} \approx 0.54$ and reducing to $\Delta b \approx -0.1$ for $T = 1.5 > T_{\text{crit}}$.

5. Applications of LIF sampling on Accelerated Analog Hardware

For the parameter studies shown in Section 4.3 alone we ran nearly 40 000 simulations and while their runtime varies depending on the model and backend, the average stands at several minutes[1]. Especially the simulations of LIF networks take a long time. These simulations are only possible due to our access to the bwHPC system NEMO [von Suchodoletz et al., 2016] for which we gratefully acknowledge the support. However, all our simulations are still limited to about 60 s of equivalent biological time. Since, our networks are structurally simple and, at least for the most part, we only take interest in steady-state simulations, such short simulations are sufficient for us.

On the other hand, there are a lot of areas of research that do not parallelize as efficiently as our investigation of the phase diagrams in Section 4.3 or the activation function in Section 4.1. Anytime we require knowledge of the past iterations before we can perform the next step of the simulation only serially. This is, in particular, true for any kind of learning task. Any kind of learning can be formulated as an optimization of some cost function $\mathcal{L}$. The update (or learning) is then some function of an approximation of some function of this cost function $f(\mathcal{L})$. Typically this is some kind of gradient estimator [Hinton et al., 1995, LeCun et al., 1989, Carreira-Perpinan, Miguel A and Hinton, 2005] or at least some kind of performance-gated sampled search [Whiteson and Stone, 2006, Roelfsema and Ooyen, 2005]. We can, of course, parallelize within the gradient estimation, but still have to apply the updates sequentially. In all these approaches one can speed up the computation by parallelization for the price of additional communication of the results of the parallel processes.

Simulations are the scientific short cut to lab experiments. This might seem like a controversial statement as it can be construed to both denigrate the computational sciences and the experimental sciences, but it is a simple trade off: On the one hand, we have the ultimate arbiter, the physical experiment. There is no point in arguing with measured data, this is what nature's answer was to the question we asked with the experiment[2]. But experiments tend to be expensive, both in man power and in lab equipment. Doing simulations can be very clean in the sense that we can understand the model completely that is executed. This is an obvious disadvantage if the model is wrong, but extremely helpful to separate data from noise if the model is at least useful[3].

[1] The number of simulations run in total is lost in the ether and the author is very glad not to have payed the electrical bill himself.

[2] It may very well be that we do not understand the question, i.e., the experimental setup well enough to correctly interpret the results. But as long as data is not fabricated it is the ultimate true answer to some question.

[3] All models are wrong, but some are useful - [Box, 1976]

In the introduction we argued for neuromorphic hardware as a pathway to understand and harness the brain's computational power. In Chapter 4 we would have liked to have it just to accelerate our parameter studies. But in order to simulate lifelong learning, which takes years in biology and would take decades in simulations, the accelerated platforms are absolutely necessary [Schemmel et al., 2010, Furber et al., 2014, Akopyan et al., 2015, Jouppi et al., 2017, Davies et al., 2018]. In this chapter we will discuss two implementations of LIF-based probabilistic computations that where done on the BrainScaleS systems. The first one in Section 5.1 was done in close collaboration with Akos F. Kungl and implements a generative model for hand-written digits (MNIST) as well as the fashion MNIST dataset on the wafer-scale BrainScaleS-1 system. The second one in Section 5.2 was implemented in close collaboration with Stefanie Czischek and uses BrainScaleS-2 to represent entangled quantum many body states via a spike-based implementation of a positive operator-valued measure (POVM).

5.1. Discriminative and generative tasks on BrainScaleS-1

This section presents work done in collaboration with Akos F. Kungl and reported in [Kungl et al., 2019].

In this section we present the implementation of a Bayesian inference model on the BrainScaleS-1 platform [Schemmel et al., 2010]. We begin this section with a very brief description of the BrainScaleS-1 system in Section 5.1.1. In Section 5.1.2 we then discuss the experimental setup, both the noise generation and the network layout. Finally, we report the results on the MNIST and fMNIST datasets in Section 5.1.3.

5.1.1. The BrainScaleS-1 system

BrainScaleS-1 is the first generation of large-scale accelerated emulation platforms for spiking neural networks [Schemmel et al., 2008, 2010]. It is the successor of the Spikey chip [Pfeil et al., 2013], was initially developed in the BrainScaleS Project [BrainScaleS, 2011] and is now continued in the Human Brain Project (HBP) [Markram et al., 2011b]. The system aims to provide a platform enabling accelerated large-scale emulations of spiking neuronal networks. Accelerated here means that the required emulation time is lower than the biological real time that is emulated. This constant runtime is achieved by designing and manufacturing electronic circuits which implement dynamics equivalent to the ODE description of the LIF model (cf. Section 2.1.1). As the typical time constants of these circuits are smaller than their biological counterparts one ends up with an acceleration factor. Since the neurons are physical circuits their temporal evolution is inherently parallel and, in particular, the network size does not influence the emulation speed. For BrainScaleS-1 with its HICANNv4 chips the acceleration factor can be configured within the range of 1000 to 100 000 [Schemmel et al., 2008, 2010]. Without this speedup simulations of life-long learning or even evolutionary changes will hardly be possible[4].

The complete BrainScaleS-1 system consists of 20 modules, each housing one wafer of High Input Count Analog Neural Network (HICANN) chips (cf. Fig. 5.1cA). These HICANN chips (cf. Fig. 5.1b) form the heart of the BrainScaleS systems. Each chip features 512 adaptive exponential leaky-integrate-and-fire (AdEx, [Brette and Gerstner, 2005]) neuron circuits and up to 112 640 conductance-based synaptic connections. Fig. 5.1a shows the schematic of the analog circuits that implement the different components of the emulated ODE (cf. Eq. (2.1)). LIF is a subset of the AdEx model. As we only use the LIF part we do not discuss AdEx further. The 512 circuits are also call dendritic membranes (denmems) and their capacitances can be connected (short-circuited) to form larger neurons thereby trading more input per neuron for fewer individual neurons. The denmems are located in the middle of the chip in an upper and a lower row

[4] Emulating a biological year would take about a day, making lifelong learning a viable research subject, even on a mechanistic model.

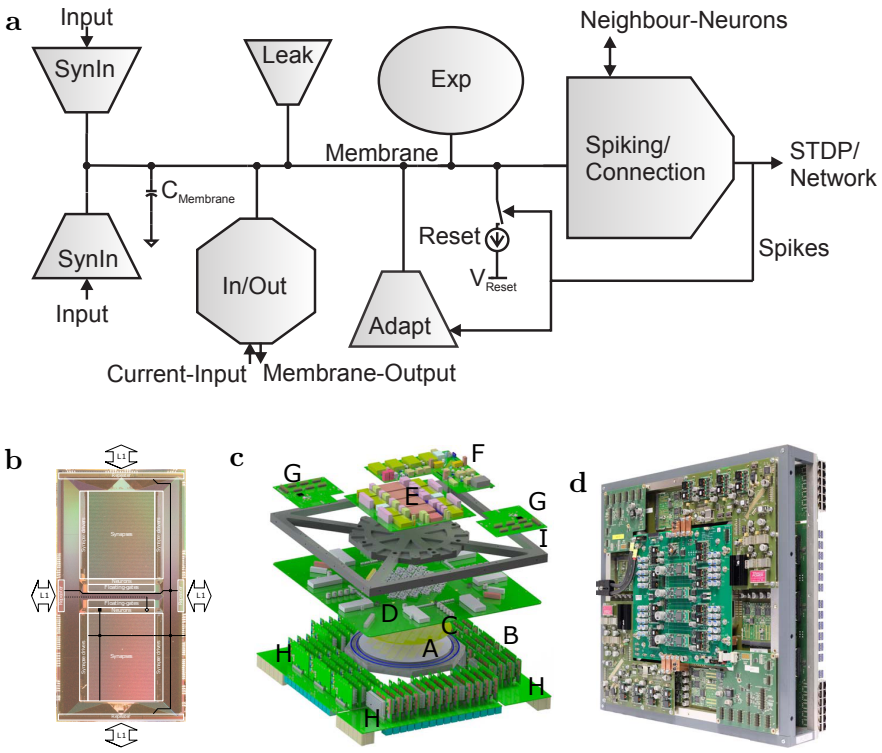

Figure 5.1.: **The BrainScaleS-1 system: a**, Schematic of the implemented AdEx circuit. Due to the modular design single components of the chip can be configured (and turned-off) independently. We only use the LIF part and ignore the Adapt and Exp terms/circuits. Image taken from [Millner, 2012] **b**, one of 384 High Input-Count Analog Neural Network (HICANNv4) chips of a BrainScaleS-1 wafer scale system. It features 512 AdEx neurons (middle row) with a total of 112 640 synapses in two blocks (upper and lower block). The synapse array takes most of the available chip space. Inter-chip communication is done via a packet-based nearest neighbor connected scheme (L1). Image taken from [Millner, 2012] **c**, Schematics of a complete BrainScaleS 1 module. (A) The wafer module with its 384 HICANN chips. (B) The positioning mask and (C) the elastomer connectors to the (D) main PCB board. (E-G) Further PCB boards for the power supply and analog readout capabilities. (H) Inter-wafer and host connectivity is provided via a custom-made USB form factor and Gigabit-Ethernet slots, respectively. (I) An aluminum frame for mechanical stability. Image taken from [Schmitt et al., 2017]. **d**, Photograph of an assembled wafer module, taken from [Schmitt et al., 2017].

(cf. Fig. 5.1b). They are flanked on the inner side by their parameter storage, which are implemented by so-called floating-gates [Lande et al., 1996], and the synapse arrays on the outside. The latter implement a total of 112 640 possible connections and take the largest part of the chip area.

Whenever the membrane voltage u at the capacitance exceeds the threshold value V_{thresh} the comparator circuit triggers the spiking behavior. This consists of a) short-circuiting the membrane to the reset potential V_{reset} and b) generating a digital spike package. The short circuiting is released after the refractory period τ_{ref} and the timing is controlled via one of the floating-gate values and is subject to significant variations for the large values of τ_{ref} that we desire[5] for a (relatively speaking) shorter effective membrane time constant. The spike event consist of a tuple of an ID and a time stamp. The latter is only for bookkeeping purposes (i.e., only evaluated off-chip and does not influence the actual on-chip dynamics). The package is generated at the neuron (white $\diamond$ in Fig. 5.1b) and routed along a predefined route (black lines). This route is near-arbitrarily configurable at chip initialization, but requires compute-intensive mapping. As such we map our desired network once at the beginning of the experiment and therefore treat the routes as predefined and static.

The spike package is injected onto a horizontal line which, due to the wafer-scale integration, is also connected to neighboring HICANNs (L1 in Fig. 5.1b). The horizontal lines can be connected to the vertical lines through a configurable, but sparse switch matrix. At an enabled switch (black dot), the signal is transfered to a vertical line in order to either reach HICANNs above or below (L1 on the top or bottom) the current one or to be injected into the synapse array. This injection happens via a synapse driver which *drives* two lines of synaptic input. Each of the 440 synapse lines has 256 single synapses. These connect to vertical lines above the synaptic input of each denmem circuit. The synapses check the source ID of the spike package and apply a 4-bit weight modifying the strength of the interaction. There exist 2 synaptic input circuits per denmem, one is configured to be excitatory, the other is configured to be inhibitory. Each synapse line can only exclusively connect to either of those (Dale's law [Dale, 1953], cf. Section 2.1).

For communication with the outside, each HICANN has so-called L2 connections (not shown). Per reticle, which consists of 8 HICANNs, there is one field-programmable gate array (FPGA) (Fig. 5.1cH). This grouping is an artifact of the production process where only groups of 8 can be printed at once. In total, there are 48 Kintex7-FPGAs, each having a 1-Gigabit connection to its associated reticle. In total 384 there are HICANNv4 chips per wafer. To put this bandwidth into perspective: With a single spike package requiring 27 bit data [Brüderle et al., 2011], this means we can transport at most:

$$f_{\text{event}} = \frac{48\,\text{Gbit}}{27\,\text{bit}} \approx 1 \times 10^9\,\text{s}^{-1} \tag{5.1}$$

events per second to and from the complete wafer. In other words on the biological time scale and with using all 512 possible neurons HICANN, this would correspond to a peak

[5] Effectively, the reset timer is implemented as a charging capacitance, where one needs to control a tiny current in order to generate long refractory periods.

lossless activity of

$$\nu_{\text{bio}}^{\text{lossless}} = \frac{f_{\text{event}}}{10000 \times 512 \times 384} \approx 0.5\,\text{Hz} \tag{5.2}$$

per neuron [Brüderle et al., 2011][6].

The rest of the assembled wafer module (cf. Fig. 5.1d) consists of auxiliary components, that provide for example power and analog readout capabilities. The connection to the controlling host system is implemented via Gigabit-Ethernet ports. In practice, this is all (relatively) nicely encapsulated, such that the end-user only thinks about the logical network [Jeltsch, 2014, HBP-SP9, 2020]. For the large scale system the mapping still requires significant hand-holding. This encapsulation is the result of significant software work[7] which handles all of the low-level parameter setting and also the translation from the biological domain onto the hardware domain [Brüderle et al., 2011].

While AdEx behavior has been shown to work and reproduce diverse firing patterns observed in biology [Tran, 2013] we will only use the LIF part and as such we refer the interested reader to [Kleider, 2017, Koke, 2017, Schmitt et al., 2017].

Even though the acceleration factor is a blessing for the execution time, it can become a liability when a lot of spikes need to be communicated between the host and the chip. Remembering the discussion of the origin and the implementation of the noise spikes from Section 3.2 and our requirement of at least 300 Hz of excitatory and inhibitory noise per sampling neuron[8], we find that the available input bandwidth per HICANNv4 reticle (consisting of 8 HICANNs) is

$$f_{\text{reticle}} = \nu_{\text{bio}}^{\text{lossless}} \times 512 \times 8 \approx 2\,\text{kHz}. \tag{5.3}$$

This is sufficient to sustain about 2 sampling neurons [Kungl, 2020]. We can circumvent this limitation by using additional reticles to feed in the input.

For chip-to-chip communication the direct L1 connections offer near-infinite, at least when compared to the off-system L2 connections, bandwidth (one spike per chip clock tick on each of the eight lanes in all directions). As such we implement the noise sources through output spikes from an inhibitorily-connected random network (cf. Section 3.2.3). Note: even though we can refrain from having to send spikes in, it is still possible that we loose spikes on the way out. In other words, there is the possibility of the network on the hardware generating more spikes than are observable from the host. Distributing the network sparsely, and thereby spreading the bandwidth requirement over multiple connections, helps. However, we still need to take care distributing the neurons adequately, such that all routes can be instantiated without loss. In practice, this mapping is what limits our implementable network sizes and force us to only solve restricted problems [Schmitt et al., 2017, Kungl et al., 2019]. We aid the mapping by

[6] Since the communication stages all have small buffers a very short spike above this rate does not necessarily lead to loss. The typical case is that one is either significantly above, or significantly below this threshold.

[7] To quote Steve Furber: A hardware project always also ends up being a software project.

[8] We follow [Kungl, 2020] and denote times and frequencies in the biological ranges throughout this chapter. The corresponding hardware values differ by the speed-up factor of 1×10^4.

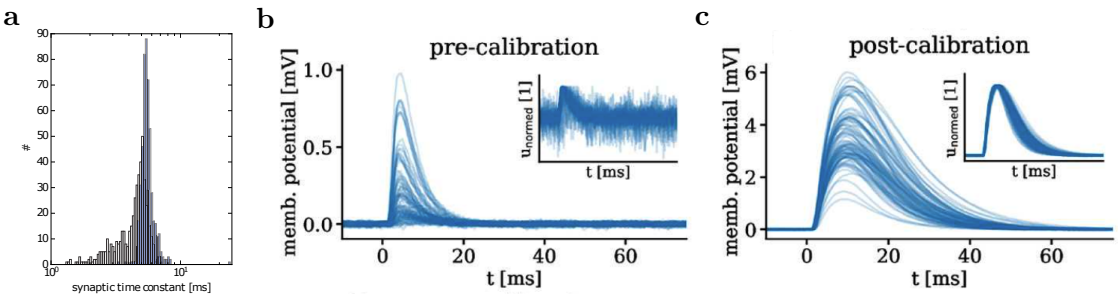

Figure 5.2.: **Parameter variations: a** Exemplary: synaptic time constants of all neurons on a HICANN chip. For homogeneous configuration variations arise due to circuit-to-circuit mismatch of the hardware circuits (white). After using the configurability of the circuits to compensate ("calibrating the circuits") the variation is reduced (blue). Figure take from [Schmitt et al., 2017]. Resulting PSP on chip before (**b**) and after (**c**) calibration. Both the amplitude and the time constants of the membrane shape vary significantly. For ease of comparison of the time constants the insets show the PSPs normalized to the maximum. Figure taken from [Kungl et al., 2019].

using neurons consisting of 4 denmems, which was also the supported standard at the time.

The use of 4-denmem neurons has the added benefit of, at least somewhat, averaging out circuit-to-circuit variations. There are multiple sources of these: First, all these analog circuits are subject to manufacturing variances. As such, setting the same configuration on two different circuits leads to different, but correlated, realizations of the dynamical parameters. This can be compensated for via a calibration, i.e., we measure the circuit and construct a map between the digital parameters we choose on the host computer and the actual realized configuration in hardware [Kleider, 2017, Koke, 2017]. In practice, this approach reaches its limit due to, on the one hand, the non-orthogonality of the parameters (e.g. the settings for the leak conductance g_l also influences the leak potential V_l, or multiple hardware parameters influencing the same dynamical parameter) and, on the one hand, the massive scale of the system. A single configuration of a complete wafer system has 44 MB of configuration data [Brüderle et al., 2011], storing each cross combination in a look-up table for arbitrary calibration is not feasible. In practice, we only calibrate the neuron parameters, as the synapse array exhibits comparatively low variation.

Second, while the floating-gates are energy efficient [Lande et al., 1996], at least their wafer-scale implementation comes with drawbacks. Their writing process is noisy [Kononov, 2011, Koke, 2017, Kleider, 2017, Müller, 2017, Schmitt et al., 2017, Kungl et al., 2019] and results in trial-to-trial variations that cannot be compensated for with calibration, but rather represent a fundamental uncertainty of the realized configuration. We work around this by only writing the analog configuration of the whole system (in-

cluding the routing configuration) once at the beginning of the experiment. During the learning experiment we only rewrite the 4-bit synaptic weights which are implemented digitally and do not exhibit these variations[9]. This improves the consistency within the training process. This does not alleviate the variations in the configuration for model storage and re-usage. The floating-gates also exhibit some drift over time [Koke, 2017], but this is not yet a problem for experiments on our timescales.

In Fig. 5.2a we see the distribution of the synaptic time constants τ_{syn} for different neurons before (white) and after (blue) the calibration. The spread before the calibration represents both the circuit-to-circuit variations as well as the trial-to-trial variations from rewriting the floating-gates. The latter forms the bulk of the spread of the post-calibration distribution, where the remaining offset due to the limited configurability (parameters are only 10-bit digital values) is negligible. Fig. 5.2b shows the PSPs for all neurons before the calibration. Each line is an average over many single PSPs (inset) in order to reduce noise of the readout. After the calibration (cf. Fig. 5.2c) both the spread of the time constants (better observable in the inset, where the maximum is normed) is significantly reduced and the average amplitude significantly increased. The amplitude varies still by nearly an order of magnitude.

5.1.2. Experimental setup and training

Due to the floating-gate variations, we want to avoid reconfiguration of the analog parameters of the neurons. Unfortunately, this includes the leak potential V_{l}. Therefore we introduce a periodically firing leak-over-threshold $V_{\text{l}} > V_{\text{thresh}}$ neuron ⓑ as a constant spike source. It allows us to implement the bias as another synaptic connection originating from ⓑ (cf. Fig. 5.3A and B). As BrainScaleS-1 was designed with Dale's law in mind each synapse driver can only connect to one type of synaptic input. Since we require both excitatory and inhibitory inputs, each neuron's output needs to be routed to two synapse drivers in order to have both connection types available. This also allows for a (fast) digital-only reconfiguration in case of a sign-flip of a logical connection throughout the training for the price of doubling the required number of synapses needed for the network connections.

Therefore each logical sampling neuron, consists of:

1. The actual sampling neuron ⓢ

   These are 4 denmems which are short circuited and calibrated to roughly match the requirements for sampling. I.e., the membrane time constant τ_{m} is small and the synaptic τ_{syn} and refractory τ_{ref} time constants are long and equal.

2. Two bias connections

   Two connections from a leak-over-threshold bias neuron ⓑ , with at least one of the two synapses being zero. The sign of the bias weight W_b determines which is the active (i.e., non-zero) synapse.

[9] This does not mean that the resulting PSPs form a 4-bit system as there also the configuration of the synaptic input is relevant. There are also variations in the linearity of the weight scaling [Koke, 2017], but again, this does not concern us on our level of discussion here.

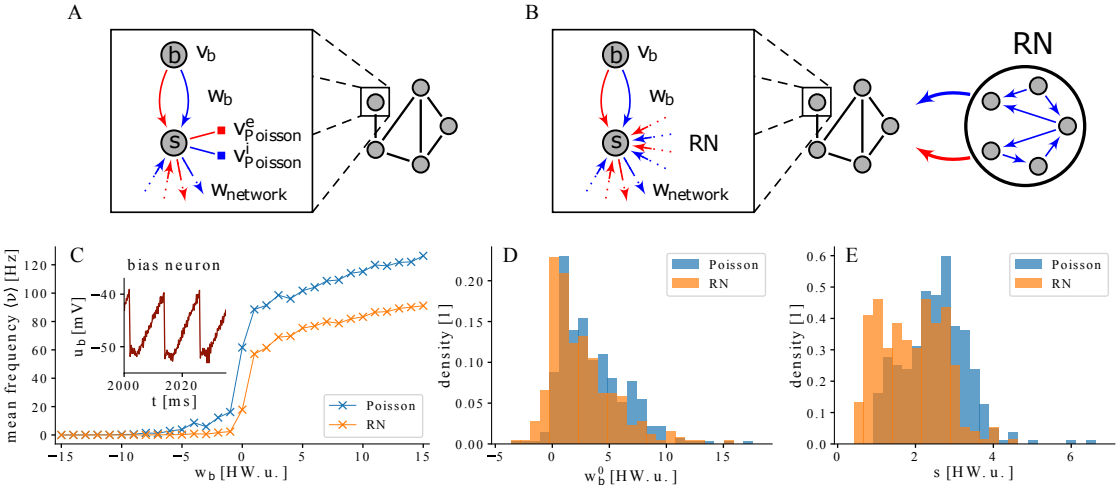

Figure 5.3.: **Experimental setup:** Each sampling unit is instantiated by a pair of neurons on the hardware. The bias neuron is configured with a supra-threshold leak potential such that it fires regularly, thereby acting as a bias to the sampling neuron. The stochastic noise can be provided via per-neuron-private Poisson-distributed spike trains from the host **A** or via random connections from a recurrently connected network of leak-over-threshold neurons **B**. **C** Activation function of single sampling neuron for Poisson (blue) and recurrent network (orange). The inset shows the membrane time course of the bias neuron with its regular spiking. **D** (**E**) Distribution of fitted offset (width) parameters of the activation function for Poisson (blue) and recurrent network (orange) provided noise. Figure adapted from [Kungl et al., 2019].

3. Noise input

   Either one excitatory and one inhibitory connection from externally fed in spikes (small demo example, Fig. 5.4, cf. Fig. 5.3A) or n excitatory and n inhibitory connections to $2n$ randomly selected neurons from the random network that forms the noise pool (cf. Fig. 5.3B). In both cases the weight of the noise connections is fixed.

4. Network connections

   Each logical connection w_{ij} is implemented via 2 synaptic connections, with the sign determining which of the two synapses that implement W_{ij} is active.

At the software level this is implemented on top of the automatically implemented routing. In other words, we need to manually take care of selecting the correct synaptic connection according to the sign and explicitly requesting the double connections. The bias neurons are implemented as a single leak-over-threshold neuron (b) which projects onto all sampling neurons (s) via double synapses.

This network is then mapped once in the beginning and the input-output relation (activation function σ, cf. Section 2.2.4) for all sampling neurons Ⓢ are recorded. For this we sweep the bias weight w_b over all available settings from -15 to $+15$. The resulting output firing rate ν is shown for one neuron in Fig. 5.3C. Whether we provide noise externally (blue) or from the random network (orange) does not change the behavior significantly. However, we observe that the activation functions are much rougher than the ones we encountered in Section 4.1 and they do not saturate on the high-bias side. The former is mostly due to the rough weight resolution of 4-bit and the non-monotonic behavior for some weight configurations, both of which limit the tuneability of the noise strength. In principle, we could change the scale of the x-axis by decreasing the noise strength and make the dynamic behavior more visible (cf. Section 4.1).

The slow upwards drift on the high-bias-weight side is due to the significant time spend in the drift after the actual reset is already released. In principle, we could reduce this by reducing the distance between the reset V_{reset} and threshold V_{thresh} potentials. In practice, this again is limited by the floating-gate variations as a $V_{\text{thresh}} < V_{\text{reset}}$ configuration leads to misbehaving circuits. The resulting asymmetry is one of the factors limiting the overall sampling quality.

In contrast, the difference in scale is only due to different neuron parameters due to the floating-gate reconfiguration. The resulting parameters of the activation function vary significantly between the different neurons (cf. Fig. 5.3D for the offset and Fig. 5.3E for the width parameter). The data shown in Fig. 5.3D and E is obtained from a single configuration for Poisson and a single configuration for the random network case. The offset is correlated as the neurons share the configuration for some parameters.

In principle it is possible that we find logical neurons, i.e., groups of 4 denmems, that are not well behaved. Either their excitability is too low, resulting in too low output rates ν or one of the synaptic input circuits has no effect. In this case we blacklist the resulting neuron circuits, rerun the mapping and repeat until we have a network of sufficiently well-behaved neurons. In practice, we found enough wafer real estate that a manual post-blacklisting beyond the one provided from the calibration database was not necessary [Kungl et al., 2019].

In order to later assign a state $z = 1$ to a neuron we need to know its refractory period τ_{ref}, while we can set this in principle using that information has two problems: The parameter variations introduces significant discrepancies between the requested and the implemented value. More importantly we do *not* really care for the time the neuron is in its actual reset state, but rather for the time it requires before it is able to spike again. As we can see from the inset in Fig. 5.3C the drift time after the end of the refractory period can be significant[10]. If we assign the state $z = 1$ for $t < \min[ISI]$ we introduce a minimal $p(z = 0)$ which limits the probability distributions that we *could* represent. Therefore we empirically measure the average ISI and use this value for the convolution of the spike train that results in the state vector $\vec{z}(t)$.

[10]The drift in Fig. 5.3C appears worse than it is as for the bias neuron the refractory time was not chosen to be particularly large. Nevertheless the drift of 7 ms is significant.

Training

In Eq. (2.48) we derived the update rule for general BM as

$$\Delta b_i \propto \langle v_i \rangle_{\text{data}} - \langle v_i \rangle_{\text{model}} \tag{5.4}$$

$$\Delta w_{ij} \propto \langle v_i h_j \rangle_{\text{data}} - \langle v_i h_j \rangle_{\text{model}} \tag{5.5}$$

where the data term corresponds to the target distribution and the model term corresponds to the currently implemented model. For small distributions we can calculate the data-correlations directly from the full distribution and thereby only require samples for the model terms. For larger distributions, or distributions that we only know partially, we need to sample both terms. In practice, we use this for RBM-like spiking neural networks. There the visible and label layer represent an image and a class label respectively with the hidden layer being responsible for forming the probability distribution over visible and hidden layer appropriately. Here we are required to enforce the *correct* distribution on the visible and label layer, while sampling the hidden layer. This enforcement in the language of statistics means that we need to condition the model distribution onto some sample from the training set.

For our binarized images this conditioning means forcing the respective neuron to continuously spike (for $z = 1$) or to never spike (for $z = 0$). In software simulations we implement this by setting the leak potential V_l to arbitrarily high (low) values. On hardware this is not an option, as there are fundamental limits on the dynamical range of the system, and for BrainScaleS-1, in particular, the floating-gate variations are again a problem. Furthermore, changing V_l would require a reconfiguration between each sample, which on BrainScaleS-1 requires a new experiment run[11]. Therefore we opted to feed in clamping spike-trains in form of additional excitatory and inhibitory connections to the neurons of the visible layer. These spike trains can be constructed on the host and – as long as we stay within the available communication bandwidth – allow us to clamp an arbitrary sequence of images in the visible layer. With the thus-conditioned system we can sample the data phase analogous to the model phase.

A single weight update thus consists of

1. Updating the (digital) synaptic configuration of the network

2. The sampling run

   In order to ensure proper clamping we needed to deactivate the connections originating in the hidden layer during the date phase. This forced us to use two experiments in order to collect the correlations in the two phases. During the data phase external spikes clamp the corresponding neurons, during the model phase no additional external spikes are provided. The achieved experiment repeat rate on BrainScaleS-1 was roughly one per 2 s.

3. Translation to states

[11] At least from the python interface we utilized.

For both phases the spikes to states translation is made separately based on the measured τ_{ref}.

4. Parameter update calculation

While the realized synaptic configuration is a set of 4-bit weights, we perform the calculations on full 64-bit floating point numbers (so called shadow weights [Courbariaux et al., 2015]). This does not provide additional overhead as we need to perform these calculations on the host anyways, but allows us to sidestep the discrete nature of the synaptic weights[12]. Without this quasi-continuous formulation multiple updates in different learning iterations could not add up to a bit-flip of a hardware weight. Only the latter has an observable impact on the dynamics. At this point we would need some other method to allow a series of small updates to influence the parameterization in order to avoid making each parameterization to a local minimum for sufficiently small learning rates [Maxfield, 2006].

In all cases involving larger networks we start by training in software simulations completely on the host, before transferring the pre-trained network to the hardware system via the measured activation functions (cf. Fig. 5.3C-E). We have to expect severe limitations of the performance as we are only capable of using 4-bit weights, variations between synaptic time constants τ_{syn} alone are on the order of a factor of 2 and the general effective parameter variations due to the floating-gates. In the post-training, where we iteratively reconfigure the hardware, we correct for these mismatches.

5.1.3. Results

In order to characterize the behavior of a sampling system it is always advantageous to start with a small network such that we can treat it a) analytically and b) can provide sufficient Poisson distributed external spike trains for all neurons. Using multiple HICANNs for the input routing it is possible to provide ample noise for an $n = 5$ neuron spiking sampling network. Fig. 5.4A shows the median final DKL for a single sampling run as a function of the training iteration. The shaded region corresponds to the interquartile range over 150 different initializations for Poisson (blue) and RN (orange) noise. The Poisson noise case converges slightly faster and shows larger variations between different initializations. The dashed lines give the minimal DKL observed for the respective noise configurations.

The evolution within the last sampling run at the end of the training is shown in Fig. 5.4B. The DKL averages down as expected as $\frac{1}{N}$ [Cai et al., 2006, Paninski, 2003] up to the point where the implemented precision is reached. At this point the DKL saturates. Fig. 5.4C and D show the joint and marginal distribution of the target (green), Poisson (blue) and RN (orange). We see that most probabilities vary within the error bars over the different initializations. Note the different scale of the y-axis between c and d. The error bars again show the interquartile range around the median.

[12] Technically 64-bit floats are still discrete values. Compared to the 4-bit discretization this is a largely philosophical point.

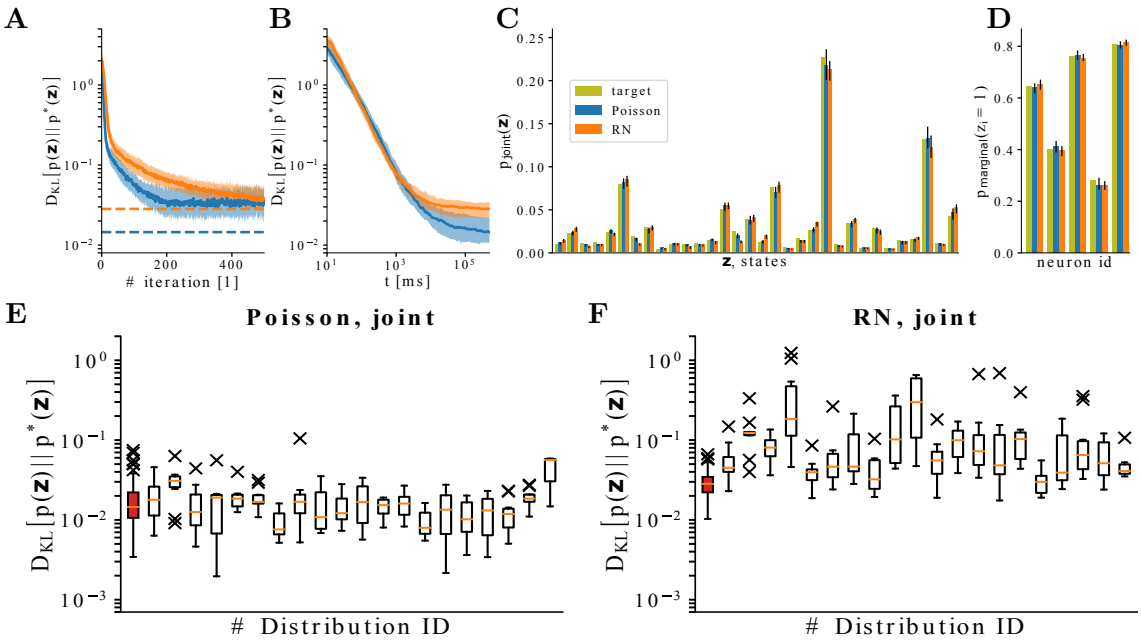

Figure 5.4.: **Sampling from arbitrary distributions:** **A** Sampling quality on BrainScaleS-1 for Poisson noise (blue) and recurrent networks (orange) as a function of training iteration. Poisson noise converges slightly faster and overall performance is largely limited by the available weight resolution. Here and in the other sub panels the mean and interquartile ranges for 150 different initializations of the network are shown. Dashed lines are the overall minimal achieved DKL values. **B** Convergence of the sampled distribution during a single run at the end of learning. **C** Final joint distribution. **D** Final marginal distribution. Final achieved DKL for 20 different target distributions with 10 repetitions per sample for Poisson noise (**E**) and recurrent network noise (**F**). The orange line represents the mean value, the box the interquartile range, the whiskers represent the full data range and the x represent the far outliers. The left most distribution corresponds to the distribution in **A**. Figure adapted from [Kungl et al., 2019].

In order to demonstrate that this kind of learning works for near-arbitrary distributions, we repeat the experiment for 20 different distributions. The different distributions are generated by sampling the parameters from a beta distribution $w_{ij}, b_i \propto 2\,[\mathrm{Beta}(0.5, 0.5) - 0.5]$. The Beta distribution is motivated by previous studies [Petrovici et al., 2016, Jordan et al., 2019]. In Fig. 5.4E and F the resulting DKLs at the end of the training for joint and marginal distributions are shown in a box-and-whisker scheme, where the extra symbols give outliers, the boxes represent to the interquartile range and the whiskers the full distribution minus the outliers. The number of repetitions per distribution was reduced to 10, except for the first distribution which is the one from Fig. 5.4A-D. In all these experiments we only reconfigured the digital part of BrainScaleS-1 and only did an analog configuration once per distribution. Poisson noise (Fig. 5.4E) shows significantly better results for all distributions and is less sensitive to different distribution parameters. When using the random network as the noise provider (Fig. 5.4F) the DKL for the typical distribution is higher, but also there is more variation between different distributions. Some distributions achieve near on-par quality and repeatability (e.g. number 6, directly above the #-sign), others show significant shifts or at least huge variations within the repetition for different initial network parameters b_i, w_{ij}. The different sensitivity can be explained by the induced correlations due to the shared noise pool between the neurons, which would favor distributions with a similar correlation structure.

External dataset (f)MNIST

We demonstrate that BrainScaleS-1 is able to implement a general model with the MNIST and fMNIST datasets [LeCun et al., 1998, Xiao et al., 2017]. The (f)MNIST data set consists of 28×28 pixel gray-scale images. We were limited to network of 208 (207 for fMNIST) network neurons and 400 neurons in the random network [Kungl et al., 2019]. As such we resized the dataset to 12×12 pixel and binarized the images around the median gray value of each image (see Fig. 5.5A and e for an impression of the resulting pixelation). Due to the harsh size reduction the discriminability of the data set is greatly reduced. Therefore and, in particular, based on the rough activation functions (see Fig. 5.4A) we expect the performance to suffer. In order to compensate we limit the model to 4 classes (3 for fMNIST).

We first train an abstract RBM[13] with the same number of neurons with standard wake-sleep learning in software (final performance as dashed lines in Fig. 5.5B and F). We then map this network in the spirit of Section 2.2.4 to the hardware parameters by using an heuristic scaling factor $c \approx \frac{\alpha_{hw}}{\alpha_{sw}}$:

$$w_{hw} = c(w_{sw} - w^0_{sw}) + w^0_{hw} \tag{5.6}$$

The scaling factor is chosen heuristically as the measurement of the complete activation function is time intensive (due to the required reconfigurations). It turns out, that the precise choice has comparatively little impact. This translation leads to a significant

[13] not a spiking network

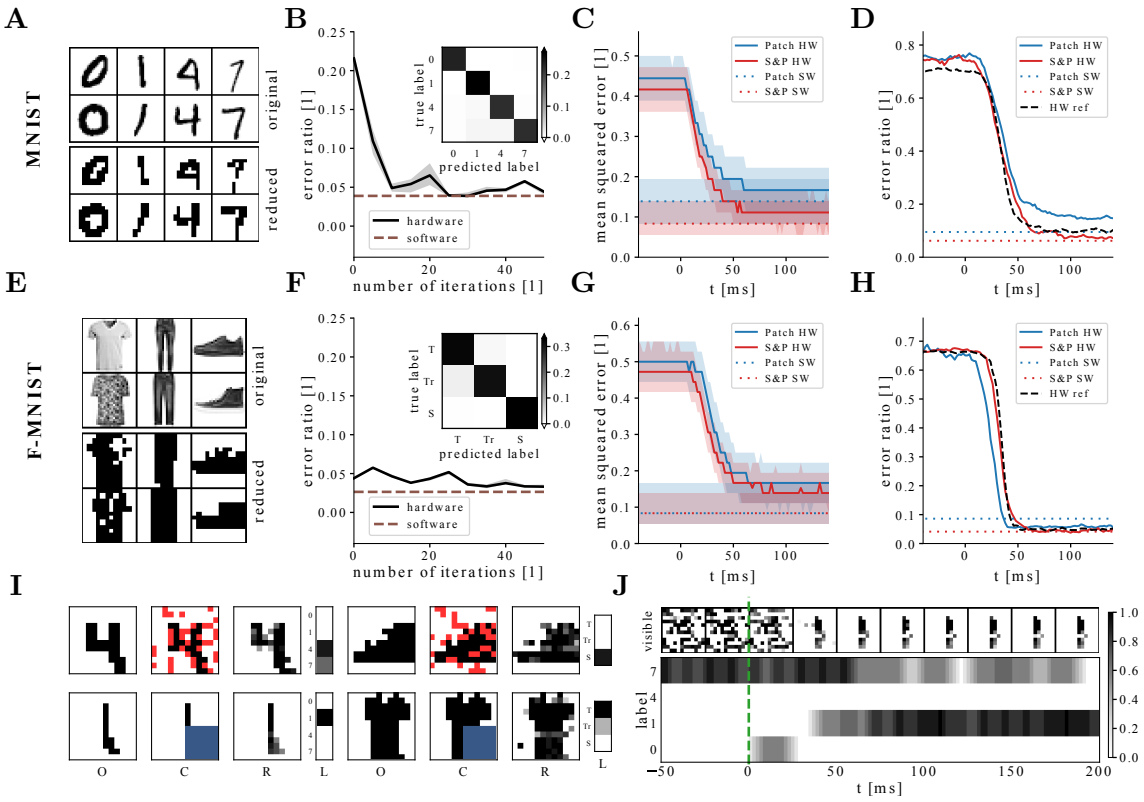

Figure 5.5.: **Generating (fashion) MNIST: A/E** Exemplary reduced (top row) and original (bottom row) images from the MNIST and fMNIST datasets used for training. **B** (**F**) Error ratio for the in-the-loop post-training after translation of pre-trained parameters (black) with software performance (red dashed line). Insets show the confusion matrix between the 4 (3) trained classes at the end of the training. **C** (**G**) Mean error of a partially obscured and reconstructed image of MNIST (fMNIST) as a function of time after a new image is clamped on hardware (solid lines) and achieved results in software (dashed lines). The performance is similar for patch (blue) and salt and pepper (red) masking. **D** (**H**) Likelihood of the network wrongly identifying the obscured image as a function of time since image change. The black dashed line shows the behavior for an completely shown image. **I** Snapshots of the pattern completion experiments: O - original image, C - clamped image (red and blue pixels are occluded), R - reconstructed image, L - response of the label layer. **J** Exemplary time evolution of the visible layer with patch occlusion. For better visualization of the activity in the visible layer, its discrete response is smoothed out by convolving its state vector with a box filter of 10 ms width. Image taken from [Kungl et al., 2019].

degradation of the classification error (0.04 to 0.21 for MNIST). This regression is expected, as we make both the step from abstract RBM to LIF sampling with its associated inaccuracies (cf. Section 3.3) and the translation to the hardware with its variations (cf. Fig. 5.2) at once.

However, most of the performance of the network on hardware is restored in only a few training iterations (solid lines in Fig. 5.5B and F). For the post-training phase the target images are clamped via externally provided spike-trains in order to measure the data terms (cf. Section 5.1.2). The confusion matrices (insets in Fig. 5.5B and F) show that the classification of the network is limited but not degenerately so, i.e., there is no single class where we fail. It is rather the total network quality that limits our performance than a particular use case. With the network size strongly limited by the mapping software (and the at that time patchy availability of the wafer area) compensating for the 4-bit weight resolution with larger hidden layer sizes was not possible.

In order to demonstrate the generative behavior we demonstrate the network's ability to perform pattern completion. We show only a part $\left(\frac{3}{4}\right)$ of the image by clamping that part of the visible layer. Unlike in the post-training phase we do not cut the connections from the hidden layer. This leaves the rest of the visible layer free to evolve under the (incomplete) clamping. In other words we ask the network to sample from the model distribution conditioned on the clamped (non-occluded) part of the image.

Starting from a random state the network quickly settles into a compatible mode to the non-occulted parts (cf. Fig. 5.5J). We filter the state of the visible layer neurons with a box-kernel of length 10 ms for visualization. The resulting mean squared distance of the visible layer to its original takes about 50 ms to converge to the steady-state value of 0.15 (Fig. 5.5C and G). The time constants of the system are chosen to be $\tau_{\text{ref}} = \tau_{\text{syn}} = 10\,\text{ms}$. It is surprising that the network takes on the order of 5 dynamical time scales to completely adjust to the new input.

The probability of misclassifying the image also drops over the same time frame (Fig. 5.5D and H). The latter reach similar classification performance as the non-occluded image, which suggests that there is still enough information to recover from a three-quarter digit/fashion piece. It turns out that the fMNIST data requires a smaller mean-squared distance to achieve good classification rates when compared to the original MNIST dataset. The completion and classification performance does not change under different occluding schemes. This is not surprising, as the *network* does not rely on topological information (such as which pixel are neighbors, as it only operates on a cloud of pixels) and we left a significant part of the image available.

5.2. Representing quantum states with BrainScaleS-2

This section presents work done in collaboration with Stefanie Czischek and reported in [Czischek et al., 2020] which is currently under review. The model was developed in consultation with Thomas Gasenzer, Martin Gärtner and Stefanie Czischek, the routing scheme and basic hardware configuration was provided by Sebastian Billaudelle and Benjamin Cramer.

In this section we learn to represent quantum states with the BrainScaleS-2 spiking neuromorphic hardware system. We use a similar setup to the pattern recognition and generation from the previous section on BrainScaleS-1. Here we will use the visible layer to implement a probability distribution that can be translated into the density matrix of a quantum state. The hidden layer is used to appropriately shape this part of the distribution. Unlike the pattern classification tasks in Section 5.1 we now a) know the exact target distribution and b) require a much higher precision in its reproduction in order to reproduce all expectation values and not only some marginalization precisely. This precision is achievable on the HICANN0Xv1 chip of the BrainScaleS-2 platform.

We will first (briefly) describe BrainScaleS-2 and discuss the improvements over its predecessor from Section 5.2.1, before sketching the positive operator-valued measure (POVM) that allow us to map density matrices ρ to probability distributions p in Section 5.2.2. Finally, we discuss the results of the experiments in Section 5.2.3. Most of this section is similar to the content of the publication [Czischek et al., 2020], though we give slightly more details regarding the hardware implementation and are a bit more explicit in the POVM derivation here.

5.2.1. The BrainScaleS-2 system

BrainScaleS-2 is a single-chip system featuring 512 current-based adaptive exponential LIF neuron circuits with 256 synapses per circuit [Friedmann et al., 2017, Aamir et al., 2018, Schemmel et al., 2020, Billaudelle, in preparation]. The HICANN-X application-specific integrate circuit (ASIC) is supposed to replace the HICANNv4 chip of the BrainScaleS-1 system, with a wafer-scale integration planned for the future. There are two major improvements between BrainScaleS-1 and BrainScaleS-2: First, the neuron parameter storage was redesigned replacing the noisy floating-gates (see Section 5.1.1) with a digital storage resulting in massively reduced trial-to-trial variations (data not shown, for further information see e.g. [Hock et al., 2013, Friedmann et al., 2017, Billaudelle et al., 2019b, Schemmel et al., 2020, Billaudelle, in preparation]). Second, there is an on-board general-purpose compute unit, the so-called plasticity processing unit (PPU). It is a PowerPC-based processor with vector extensions that can read and set all parameter storages on the chip, has access to most hardware observables (correlation sensors, membrane voltages or spike counters) and is intended to be used to implement

near-arbitrary learning rules. Implementing the learning rule directly on-chip removes the expensive outer loop between host and chip [Kungl et al., 2019] and significantly speeds up the training [Wunderlich et al., 2019, Billaudelle et al., 2019b,a, Cramer et al., 2020b,a].

Such an on-chip implementation of training sampling-based networks is currently under development (cf. Chapter 6), but requires the use of the correlation sensors and the data phase (cf. Eq. (2.48)) to be implemented via clamped neurons. While we will mention the former in Chapter 6, the achieved precision of the latter so far has proven to be insufficient for the task at hand here. Instead of investing time to improve this precision it was decided to use our improved learning scheme from Eq. (2.55). It only depends on the model term and therefore never requires us to actually enforce the target distribution on the visible layer during the training. The price for this is that we need to stick with the host-based training. As such, our implementation profits mainly from the improved reproducibility of HICANN-Xv1 as compared to HICANNv2 and not from its inherent learning capability. Even though we find that we only loose about a factor of 7 with respect to the theoretical speed up.

Routing

Since a general mapping framework was not yet available we were provided[14] with a blackbox-implementation of a network of 128 spike sources. Within this constraint we can connect the neurons or the on-chip spike generators – which are our two types of spike sources – arbitrarily with signed synapses. We use 2 of the 8 on-chip PRNGs to create 64 logical random spike sources, leaving us with at most 64 sampling neurons.

We divide the noise sources into exclusively excitatory and exclusively inhibitory sources, as this will, in the future, allow us to reduce the required number of synapse lines. In order to understand why, we need to look a bit closer at the synaptic array of BrainScaleS-2: Each block has 128 synapse drivers for 256 synaptic lines, with each driver being able to drive two lines (cf. Fig. 5.6b). Each line has to have an exclusive sign, i.e., if on any line there is to be one excitatory connection, then all other connections also have to be excitatory. This constraint is rooted in the circuit implementation and cannot be circumvented. Each neuron sits below a column of the synapse array and has two input circuits that each sum over the different synapse lines. One of those input circuits is excitatory and the other one is inhibitory. Each horizontal synapse line can only be wired up to either the vertical line that connects to the excitatory or to the inhibitory input circuits of the associated neurons[15].

As such we currently use two hardware synapse lines and one driver per logical synapse, i.e., spike source (cf. Fig. 5.6b), to implement the sign of the connection. This means for each of the signed synapses at least one is set to zero (with both synapses being

[14] For now Sebastian Billaudelle and Benjamin Cramer are the relevant high-priests of routing on BrainScaleS-2.

[15] This is less flexible than the implementation in early versions of BrainScaleS-1, where the two input circuits were able to switch their sign. This, in principle, allowed for a more flexible network setup, but came at the price of higher variations.

zero meaning that the synapse is not used at all). This allows for a flexible change of the connection matrix (no rerouting necessary) for the price of doubling the usage of hardware resources. For the noise synapses, however, we know the sign in advance and hence we know which of the two lines to use. The synapse drivers can receive spikes from multiple sources and the synapses themselves filter the received spikes.

In order to understand how this filtering works, we again need to take a closer look at what a *spike* actually is on the system. Whenever the membrane of the circuit crosses the threshold value V_{thresh}[16] it is supposed to spike. In the actual implementation (modulo some inertia of the circuit[17]) this means that the threshold comparator sets a bit "spiked" and triggers the reset circuitry. The digital part of the HICANN polls these "spiked" bits of the neurons and, if one is set, generates an appropriate spike package. This is a tuple of a 6-bit ID and an 8-bit time stamp [Schemmel, 2020]. The on-chip routing is *only* affected by the ID and completely ignores the time stamp, which gets augmented to a 43-bit time in the FPGA to be analyzed on the host. The routing up to the synapse driver is done independent of the ID and only depends on the a priori set routing configuration. The synapse driver now sends the package to either or both of its synaptic lines and each synapse checks if the source is relevant for it via a part of the spike ID [Schemmel, 2020]. In our routing implementation all synapses of the two lines are configured to listen to the same ID. Each synapse also holds a 6-bit unsigned weight w which scales the influence the synapse exerts on the input circuitry of the neuron. It is this 6-bit weight, in combination with the 10-bit leak potential, which we use to implement the bias, that form our network parameter that we want to optimize.

For the noise sources we enforce weights of the same sign (in the language of Fig. 5.6b either all synapses are zero on the upper or the lower row). Therefore, it is possible in some future implementation to route the output of one excitatory and one inhibitory source each to one driver. This would increase the possible number of network neurons to 96 without fundamentally changing the routing scheme. Even this is not the achievable limit, as we choose to have a fixed number k of noise source we listen to, since the synapses themselves can filter the sources and 6-bit is sufficient to distinguish between the 64 IDs we would be able to work with just k synapse driver lines. With this we can go up to $128 - k$ sampling neurons. In principle, another factor of two would be possible by using double circuit neurons (as were used for BrainScaleS-1, cf. Section 5.1), however, this would require significant effort for the routing. More neurons than these $256 - k$ are then only possible if we restrict the architecture of the network, both in its fan-in size and in its topology.

[16] Note that while circuit-to-circuit variations are significantly reduced for BrainScaleS-2, they still remain significant. Therefore setting the same digital configuration value to two different circuits still results in different realized threshold values. This does not matter in our particular setup, as it is just a movement of the $b = 0$ configuration. As such we did not bother to calibrate these circuits.

[17] This caveat is actually important as the circuitry does not play well with supra-threshold to sub-threshold to supra-threshold transitions.

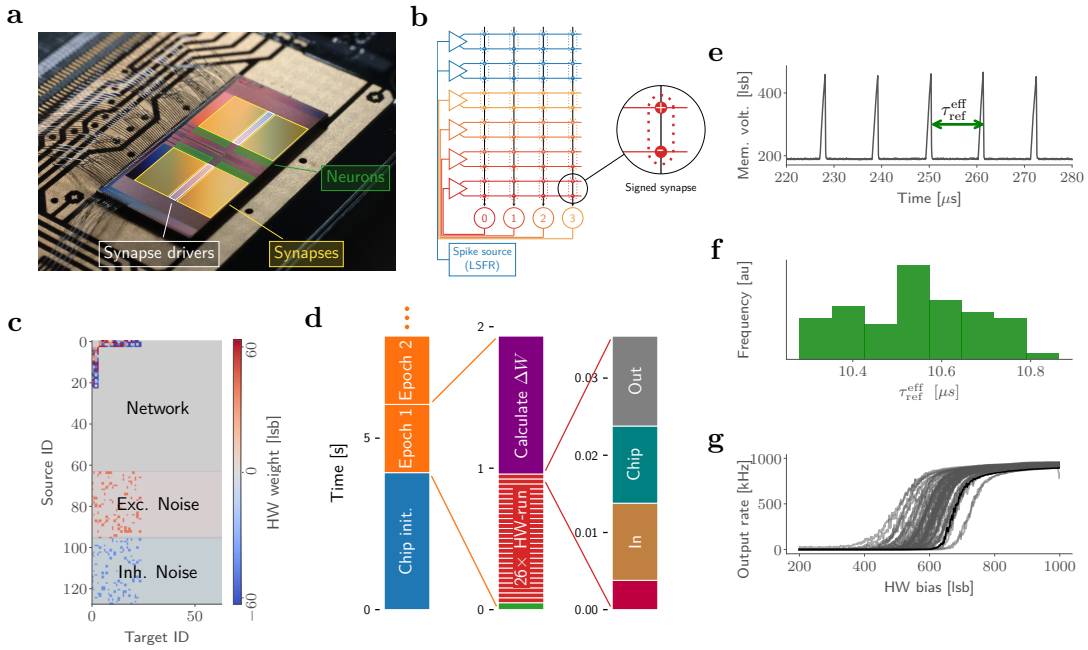

Figure 5.6.: **BrainScaleS-2 network implementation:** **a** Photograph of the BrainScaleS-2 chip with 4x128 AdEx-LIF neuron circuits (green), 2x2x128 synapse drivers (white) and 4 synapse arrays with 256x128 synapses (yellow). **b** Routing scheme used to implement the sampling spiking network. Each synapse driver projects onto two synapse rows in order to allow signed synapses. Besides the network sources, on-chip spike generators generate random spikes which are used as noise. **c** Utilized logical connectivity matrix projecting onto the 64 neurons used. Network (neuron-to-neuron) connections are truncated at index 24 (4 visible + 20 hidden) and hidden-hidden connections are not used. Each neuron receives noise input from 5 excitatory (64-95) and 5 inhibitory (96-127) sources, generated by one on-chip PRNG each. The appropriate synapse row is selected depending on the sign of the connection (cf. **b**). **d** Time usage across a training experiment: The initial configuration (blue) of the chip is comparable to a single weight update, here called epoch (orange). Each epoch consists of a parameter configuration (green), 26 sampling runs (red) and the update calculation (purple). Each hardware run consist of the construction of the playback program (dark red), the input buffering (light brown), the actual chip runtime (tile) and the readout to the host (gray). **e** Membrane trace of an exemplary neuron for a high-bias configuration. $\tau_{\mathtt{ref}}^{\mathtt{eff}}$ is the inter spike interval. **f** Histogram of measured $\tau_{\mathtt{ref}}^{\mathtt{eff}}$. Variations are due to the analog nature of the system. **g** Activation functions as a function of the leak potential under noise input of the 64 neurons used. Figure adapted from [Czischek et al., 2020] Supplemental material.

Noise sources and neuron calibration

Each logical spike source is configured to provide about 80 kHz of stochastic input (for noise characteristics see Section 3.2.2) and each neuron is randomly connected to 5 excitatory and 5 inhibitory sources (cf. lower part of Fig. 5.6c). The neuron circuits are calibrated in a limited fashion. In particular, we calibrate the synaptic time constants τ_{syn} to correspond to the refractory period $\tau_{\text{ref}} = 10\,\mu\text{s}$. The membrane time constant, on the other hand, is just chosen to be small. The calibration we used was provided by Sebastian Billaudelle and Benjamin Cramer[18].

The other somewhat critical parameter is the refractory time τ_{ref}. Its implementation has been changed since BrainScaleS-1 to a digital one, i.e., the actual reset time is a fixed number of chip-clock cycles and as such not subject to circuit variations. However, the drift time in between (see Fig. 5.6e) still is (see Fig. 5.6f). As these are fundamental to the state association we measure these at the far right side of the activation function measurements, i.e., for $V_{\text{l}} \gg V_{\text{thresh}}$. We then use the thus-obtained values $\tau_{\text{ref}}^{\text{eff}}$ (cf. Figure 5.6f for the resulting variations, which are on the order of some percent[19]). In principle shorter ISIs are possible as, in this setup, we only saturate the input current that can be provided by the leak potential circuit and ignore additional synaptic input. In practice, this works out fine and we do not observe spikes with a time difference smaller than τ_{ref} too often[20]. If we do, we consider the neuron to be continuously in state $z = 1$.

The resulting activation functions can be seen in Fig. 5.6g. There is a significant variation in the offset between different activation functions. This is expected as we do not calibrate all neuron parameters and, in particular, do not exactly calibrate the threshold value V_{thresh}. For technical reasons only a limited number of parameters may be set to the same digital configuration value. In order to prevent this for the non-calibrated parameters we add a small amount of noise to these configuration values. Since we are training the parameters anyways, rather than trying to reproduce a specified network directly, this should not affect performance. However, it does make reproduction more complicated (essentially it requires care when seeding the random offsets).

For a more detailed discussion of available functionality and a technical documentation of the HXSampling class see Appendix C.2.

Timing and speedup

Each learning experiment starts with a single chip initialization (cf. Fig. 5.6d blue) which has to be performed once in the beginning. We subsume the actual chip initialization (setting of technical parameters, power supply, communications, etc.), our specific routing configuration and the measurement of the activation functions and effective refractory times under this point. These are all things that are required to setup

[18] A general calibration framework for end users is under development, but not yet available for end users.

[19] As compared to factors of 2 in BrainScaleS-1.

[20] While not in principle a problem, the initial implementation of the state assignment assumed that all ISI are smaller than the measured $\tau_{\text{ref}}^{\text{eff}}$ resulting in a buggy spikes-to-states translation.

a sampling network on the chip and do not scale with either the emulation time or the number of training epochs. This part took about 4 s. There is a lot of optimization potential here, but as it is a one-time cost, of at most a couple of seconds, we did not bother.

From that on we only ever update the weight-part of the synapse array and the leak potential values. The rest of the configuration is static and not touched. Each iteration (or epoch) of the training consists of three phases:

1. Network parameter update (green)

   We first rewrite the synapse array and leak potential configurations (cf. Fig. 5.6d green), which takes about 49 ms. Further improvements here are possible but somewhat involved, i.e., one could only rewrite changed parameters and ignore the unchanged configuration or combine the weight and bias updates, which are currently two different hardware executions.

2. The network emulation (red)

   For (not completely understood) technical reasons the single emulation time was limited to about 10 ms. Therefore in order to achieve adequate sampling sizes we perform 26 HW-runs and combine the samples. Each hardware run consist of

   a) the construction of the playback program on the host (about 3.8 ms Fig. 5.6d dark red),

   b) an initial buffering of 10 ms to ensure that the execution does not stall due to network latency (light brown Fig. 5.6d),

   c) the actual run of 10 ms (tile Fig. 5.6d) and

   d) the data collection until the generated spike trains are available in the host's memory 11.6 ms (gray Fig. 5.6d).

3. The parameter update calculation (purple)

   Which contains the translation of spikes to states (cf. Fig. 3.4a) as well as the correlation calculation and the reconstruction of the target distribution (cf. Eq. (2.55)). With 975 ms this takes nearly half of the 1.991 s per training epoch.

There are a number of things that beg for a more detailed explanation: First, the initial buffering is used to guard against network stalls. The communication between host and FPGA runs over a 10-Gigabit switch which is shared between all BrainScaleS-2 setups. Each setup can, in principle, generate a total of 8 Gbit of bandwidth. Depending on the state of the other systems one might get a network stall. This is guarded against, as the host-FPGA communication can resend lost packages [Karasenko, 2020]. However, since we have a real-time system the chip execution will no longer fit in with the FPGA state. The initial buffering ensures that the complete program is at the FPGA in the beginning. We could drop this if the network bandwidth were guaranteed to be sufficient (and no operating system level complication arises) or reduce the impact by a larger single emulation duration. Second, the data collection time should, in principle, be the

constant offset between the end of the experiment on the chip and the last generated data arriving at the host, i.e., for longer emulation times its significance should diminish. This is currently not the case as we only start to analyze the results after the hardware run ends and all data is collected on the host. Third, the parameter calculation is somewhat optimized Python code (cf. Appendix C.2). There is room to speed up this calculation.

In total we simulate $T = 26 \times 10\,\mathrm{s} = 260\,\mathrm{s}$ of biological real time[21] per epoch in 2 s wall clock time. This gives us an effective speedup over biological real time of about 130. The remaining factor of 7 versus the nominal hardware acceleration is roughly half in the communication with the system and half in the weight update calculation. Both can still be optimized, but were deemed sufficient for the time being.

A comparison with software simulations is non-trivial, as the speed of most simulators depends on both the network size and activity. Using a comparable network setup as in Section 5.2.3 with 24, 44, 64 and 128 neurons a simulation of $T = 260\,\mathrm{s_{bio}}$ takes about 13 s, 20 s, 25 s and 51 s respectively in SBS with the NEST 2.14 backend [Breitwieser et al., 2020, Peyser et al., 2017]. In order to be conservative in our comparison, we only took the pure simulation time and ignored the setup and tear down of the NEST simulation as well as the calculation of the parameter updates (network construction by SBS). Complicating the comparison further is that our version of NEST [Peyser et al., 2017] is nearly three years old and SBS is still Python3 incompatible. If we were to dedicate effort to optimize the software comparison we would most likely end up with significantly smaller wall clock times. Nevertheless, even if we are being as unfair as possible to our hardware setup, i.e., we compare the smallest network and ignore the actual learning on the software side: We still find a speed up of at least a factor of 5.

Note 3. *Performance characteristics*

While this effective speedup, at such small network sizes and with the factor of 7 *that we loose due to inefficiencies, is surprising (at least for the author) we need to stress that the performance characteristics are qualitatively different. BrainScaleS-2 will always operate with its fixed speedup of* 1×10^3 *completely independent of the network size. I.e., for larger scale networks, while the communication bandwidth challenges increase (cf. Section 3.2), the execution time does not. With a wafer-scale setup like BrainScaleS-1 (cf. Section 5.1) a larger effective speed up factor over software simulations should be achievable.*

However, the real comparison, at least in the scope of this section should be with respect to an abstract sampling machine. Here generating a new sample takes on the order of one floating point operation (FLOP) per connection. We have 4 visible neurons times 60 hidden neurons times 125000 samples, which results in about

$$n_{\mathrm{FLOP}} = 2 \times 4 \times 60 \times 125000 = 60 \times 10^6 \tag{5.7}$$

FLOPs required for a complete new epoch[22]. Ignoring special circumstances[23] and with

[21] 10 ms actual hardware runtime with the speedup due to the different time scales of 1×10^3.

[22] We neglected the random number generation and the bias calculation, which only scale with the number of neurons and not the number of connections.

[23] Vector instructions, cache sizes, etc.

a clock frequency of $r = 2\,\mathrm{GHz}$ this requires about

$$t = \frac{n_{\mathrm{FLOP}}}{r} = 30\,\mathrm{ms}. \tag{5.8}$$

While these estimates should be taken in the spirit of a Fermi estimate and not as a specific performance prediction[24], the speed-up of nearly 5 orders of magnitude is far beyond errors on the rule of thumb. Again this comparison would change to the advantage of the neuromorphic system for larger and especially more densely connected networks.

5.2.2. A spiking implementation of POVMs

While we have already discussed the performance characteristics of the emulated network, we did not yet introduce the distribution we require to sample. The ultimate aim is to describe a quantum state with the BrainScaleS-2 platform. In this section we will introduce the means by which we are translating the general description of a quantum state, the density matrix ρ, into a probability distribution P. We will follow the writeup in [Czischek, 2020] which itself rehashes [Carrasquilla et al., 2019, Tabia, 2012, Nielsen and Chuang, 2002].

For any quantum state $|\Psi\rangle$, any operator $\mathcal{O}$ and a possible outcome of the measurement a we can write the probability $P(a)$ of the measurement $\mathcal{O}_a$ giving result a as:

$$P(a) = \left\langle \Psi \middle| \mathcal{O}_a^\dagger \mathcal{O}_a \middle| \Psi \right\rangle = \mathrm{Tr}\left[\rho \mathcal{O}_a^\dagger \mathcal{O}_a\right], \tag{5.9}$$

where the latter expression uses the trace formalism over the density matrix

$$\rho = |\Psi\rangle \langle\Psi|, \tag{5.10}$$

which is a generalization of the state formalism.

$$M^{(a)} = \mathcal{O}_a^\dagger \mathcal{O}_a \tag{5.11}$$

defines a positive semi-definite operator $M^{(a)}$ whose expectation value gives the probability of getting the measurement outcome a:

$$P(a) = \left\langle M^{(a)} \right\rangle \tag{5.12}$$

We can find a set of measurements $\{a_i\}$ that form a set of $\left\{M^{(a_i)}\right\}$, which add up to the identity when summing over all possible measurements

$$\sum_{\{a\}} M^{(a)} = \mathbb{1} \tag{5.13}$$

[24] I.e., these should be accurate to within an order of magnitude, assuming the program is efficiently programmed.

so that they form a general quantum measurement. It can be shown that the probability distribution $P(a)$ for such a system of $\{M^{(a)}\}$ fully characterizes the density matrix ρ. Furthermore the $M^{(a)}$ can be chosen such that they yield an invertible overlap matrix and hence Eq. (5.9) becomes invertible [Carrasquilla et al., 2019]. Note: This does not mean that every probability distribution maps to a *physical* density matrix ρ. In practice a general probability distribution P will result in a ρ that is not semi-positive definite.

The density matrix ρ and thereby a general quantum state of dimension d is characterized by $d^2 - 1$ real numbers. While ρ is a complex $d^2 \times d^2$ matrix, it is also hermitian and has trace $\mathrm{Tr}[\rho] = 1$, thus resulting in $d^2 - 1$ independent real degrees of freedom. As such, an informationally complete measurement [Tabia, 2012] description requires d^2 linearly independent measurements. The last one is to ensure proper normalization. A set of d^2 such operators is then called a positive operator-valued measure (POVM), with each $M^{(a)}$ being a POVM element. Here we follow [Czischek, 2020] and only consider projections as individual operators, which correspond to rank-1 POVMs [Czischek, 2020, Tabia, 2012, Nielsen and Chuang, 2002].

The Hilbert-Space dimension of a spin-$\frac{1}{2}$ particle is $d = 2$ as it can be either in the up $|\uparrow\rangle$ or the down $|\downarrow\rangle$ state. We therefore need $d^2 = 4$ linearly independent measurement outcomes $\left\{M^{(a_i)}\right\}$. Since the single measurements of a system of N spin-$\frac{1}{2}$ particles factorize, the general measurement on the complete system is given by the tensor product

$$M^{(\vec{a})} = M^{(a_1)} \otimes M^{(a_2)} \otimes \ldots \otimes M^{(a_N)} \tag{5.14}$$

and due to the informational completeness we can expand any operator in the spin basis as

$$\mathcal{O} = \sum_{\{\vec{a}\}} \mathcal{Q}_{\mathcal{O}}(\vec{a}) M^{(\vec{a})}. \tag{5.15}$$

The $\mathcal{Q}_{\mathcal{O}}(\vec{a})$ are yet-unknown coefficients which sum over all possible index combinations $\vec{a} \in \{0, 1, 2, 3\}^N$.

As the density matrix ρ is also an operator we can expand it as:

$$\rho = \sum_{\{\vec{a}\}} \mathcal{Q}_{\rho}(\vec{a}) M^{(\vec{a})} \tag{5.16}$$

and with $T_{\vec{a},\vec{a}'} = \mathrm{Tr}\left[M^{(\vec{a})} M^{(\vec{a}')}\right]$ write the probability of the measurement outcome $\vec{a}$ as:

$$P(\vec{a}) = \sum_{\{\vec{a}'\}} \mathcal{Q}_{\rho} T_{\vec{a},\vec{a}'} \tag{5.17}$$

The POVM can be chosen such that $T_{\vec{a},\vec{a}'}$ is invertible. As such we can write the coefficients $\mathcal{Q}_{\rho}(\vec{a}')$ as:

$$\mathcal{Q}_{\rho}(\vec{a}') = \sum_{\{\vec{a}\}} P(\vec{a}) T^{-1}_{\vec{a},\vec{a}'} \tag{5.18}$$

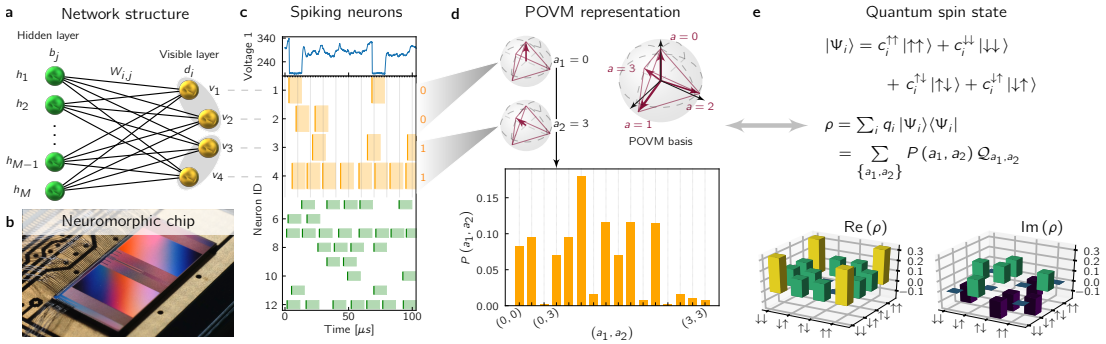

Figure 5.7.: **Neuromorphic representation of quantum states:** **a** Hierarchical spiking network with connections between visible (orange) and hidden (green) neurons. **b** Photograph of the HICANN-Xv1 ASIC which forms the core in the BrainScaleS-2 system. **c** Dynamical evolution of the network. Upper panel membrane voltage of the first LIF neuron integrating the synaptic input. Whenever the membrane crosses the threshold a reset is performed and the neuron is held there for a refractory period. Lower panels: Spikes (solid lines) for the visible (orange) and hidden (green) neurons as well as $z = 1$ time frames (shaded region). The network state is read out periodically (gray lines) and the resulting binary vectors form the sample set. **d** The 4-state POVM representation of a quantum spin-1/2 state (Bloch sphere) can be represented by two combined visible neurons. A combination of N 2-neuron pairs can thus represent a quantum N-body system. The observed frequencies of the states of the spiking neurons form the sampled approximation of the distribution (histogram). **e** Any quantum state can be represented as a density matrix ρ, which can be a statistical mixture of states $|\psi\rangle$. The complex-valued entries of the density matrix ρ can be reconstructed linearly from the sampled probabilities $P(a_1, a_2)$. Figure taken from [Czischek et al., 2020].

This allows us to write expectation values for arbitrary operators $\mathcal{O}$ as:

$$\langle\Psi|\,\mathcal{O}\,|\Psi\rangle = \mathrm{Tr}[\mathcal{O}\rho] = \sum_{\vec{a},\vec{a}'} P(a) T^{-1}_{\vec{a},\vec{a}'}\,\mathrm{Tr}\left[\mathcal{O}M^{(\vec{a}')}\right] \tag{5.19}$$

$$= \sum_{\vec{a},\vec{a}',\vec{a}''} P(a) T^{-1}_{\vec{a},\vec{a}'} \mathcal{Q}_{\mathcal{O}}(\vec{a}) T_{\vec{a}',\vec{a}''} \tag{5.20}$$

$$= \sum_{\vec{a}} P(\vec{a}) \mathcal{Q}_{\mathcal{O}}(\vec{a}) \tag{5.21}$$

where we used $\sum_{\vec{a}'} T^{-1}_{\vec{a},\vec{a}'} T_{\vec{a}',\vec{a}''} = \delta_{\vec{a},\vec{a}''}$ in the last step and otherwise the definitions from before. This expression allows for the efficient evaluation of arbitrary operators without having to reconstruct the complete density matrix ρ in between [Czischek, 2020].

The last piece missing is the calculation of the coefficients, which needs to happen for each operator $\mathcal{O}$ individually as:

$$\mathcal{Q}_{\mathcal{O}}(\vec{a}) = \sum_{\vec{a}'} \mathrm{Tr}\left[\mathcal{O} M^{(\vec{a}')}\right] T^{-1}_{\vec{a}',\vec{a}} \tag{5.22}$$

Specific choice of POVM

In practice we choose the tetrahedral representation (cf. Fig. 5.7d), where each measurement projects a single qubit onto one corner of a tetrahedron in the Bloch sphere [Carrasquilla et al., 2019]. The POVM elements M_{a_i} for each qubit i are expressed in the form

$$M_{a_i} = \frac{\mathbb{1} + \vec{s}_{a_i}\vec{\sigma}}{4} \tag{5.23}$$

with the Pauli operators $\vec{\sigma} = (\sigma^x, \sigma^y, \sigma^z)$ and

$$s_{a_0} = (0, 0, 1) \tag{5.24}$$

$$s_{a_1} = \frac{1}{3}(2\sqrt{2}, 0, -1) \tag{5.25}$$

$$s_{a_2} = \frac{1}{3}(-\sqrt{2}, \sqrt{6}, -1) \tag{5.26}$$

$$s_{a_3} = \frac{1}{3}(-\sqrt{2}, -\sqrt{6}, -1) \tag{5.27}$$

The POVM elements then take the form:

$$M_{a_i=0} = \frac{1}{2}\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \tag{5.28}$$

$$M_{a_i=1} = \frac{1}{6}\begin{bmatrix} 1 & \sqrt{2} \\ \sqrt{2} & 2 \end{bmatrix}, \tag{5.29}$$

$$M_{a_i=2} = \frac{1}{12}\begin{bmatrix} 2 & -\sqrt{2} - \mathrm{i}\sqrt{6} \\ -\sqrt{2} + \mathrm{i}\sqrt{6} & 4 \end{bmatrix}, \tag{5.30}$$

$$M_{a_i=3} = \frac{1}{12}\begin{bmatrix} 2 & -\sqrt{2} + \mathrm{i}\sqrt{6} \\ -\sqrt{2} - \mathrm{i}\sqrt{6} & 4 \end{bmatrix}. \tag{5.31}$$

This is a *choice*, in particular, we could have rotated the same tetrahedron arbitrarily within the Bloch sphere.

Quantum states and resulting probability distributions

We will first demonstrate the implementability for the prototypical quantum state, the Bell state [Nielsen and Chuang, 2002]:

$$\left|\Psi^{\mathrm{BP}}\right\rangle = \frac{1}{\sqrt{2}}\left(|\uparrow\uparrow\rangle + |\downarrow\downarrow\rangle\right) \tag{5.32}$$

and then, for slightly larger systems, by its higher-dimensional extension the GHZ state [Greenberger et al., 1989]

$$\left|\Psi^{\mathrm{GHZ}}\right\rangle = \frac{1}{\sqrt{2}}\left(|\uparrow\uparrow\ldots\uparrow\rangle + |\downarrow\downarrow\ldots\downarrow\rangle\right) \tag{5.33}$$

In order to show that this approach also works for non-pure states we additionally represent Werner states [Cabello et al., 2005]

$$\rho_{\mathrm{W}} = r\rho_{\mathrm{B}} + (1-r)\frac{\mathbb{1}}{4} = r\left|\Psi_{\mathrm{B}}\right\rangle\left\langle\Psi_{\mathrm{B}}\right| + (1-r)\frac{\mathbb{1}}{4} \tag{5.34}$$

where r is the ratio of the pure state as compared to the purely mixed state, which is represented by the term $\mathbb{1}$ which adds white noise.

Starting from the quantum state formulation in Eq. (5.32) we first need to derive the corresponding density matrix:

$$\rho_{\mathrm{BP}} = \left|\Psi^{\mathrm{BP}}\right\rangle\left\langle\Psi^{\mathrm{BP}}\right| \tag{5.35}$$

$$= \frac{1}{2}\begin{bmatrix}1&0&0&1\\0&0&0&0\\0&0&0&0\\1&0&0&1\end{bmatrix} \tag{5.36}$$

where we choose the indexing from left to right (or top to bottom) $\uparrow\uparrow$, $\uparrow\downarrow$, $\downarrow\uparrow$, $\downarrow\downarrow$. From this we can now find the target probability distribution:

$$P(a_1, a_2) = \mathrm{Tr}[\rho_{\mathrm{BP}} M_{\vec{a}_1} \otimes \vec{a}_2] \tag{5.37}$$

$$= \frac{1}{8}\begin{bmatrix}1&1/3&1/3&1/3\\1/3&1&1/3&1/3\\1/3&1/3&1/3&1\\1/3&1/3&1&1/3\end{bmatrix} \tag{5.38}$$

where the columns correspond to the index a_1 and rows to the index a_2.

Transfer to spiking neurons

We specified the probability distribution in Eq. (5.38) as a 4×4 real matrix to make the association with the outcomes $a_1^{\{0,1,2,3\}}$ and $a_2^{\{0,1,2,3\}}$ more obvious. But of course it is only the 16 numbers in that matrix that form the probability distribution. We can represent such a 16-state random variable as a distribution over $n = 4$ binary random variables. From Section 2.2 we know that a RBM with $n = 4$ visible units v_i and a sufficient number of hidden units h_j forms a probability distribution

$$p(\vec{v}, \vec{h}) = \exp\left(\sum_{i,j} b_i v_i + d_j h_j + W_{ij} v_i h_j\right) \tag{5.39}$$

Using the improved wake-sleep algorithm from Section 2.2.2 we can train the parameters W_{ij}, b_i and d_j[25] such that the marginal distribution of the visible layer corresponds to the desired one from Eq. (5.38).

For ease of association we use two visible neurons to represent the POVM of a single spin (gray shade in Fig. 5.7a) according to:

$$a_1 = 2v_1 + v_2 \tag{5.40}$$

This is a practical choice as our probability distribution in Eq. (5.38) is already non-degenerate ($P(\vec{a}) > 0$ for all $\vec{a}$) and as such representable within the Boltzmann framework. In principle it might be worthwhile to analyze the desired probability distribution to reduce the difference to a factorizing distribution by reordering the elements when reinterpreting them as a binary distribution.

On chip the network is then realized with a single LIF circuit per logical binary neuron. We read out the spike times from the chip (cf. Fig. 5.7c). From this we generate the state assignments by convolving with a rectangular kernel of the measured refractory time $\tau_{\text{ref}}^{\text{eff}}$ (cf. Section 5.2.1 and Fig. 5.6e and f). These are then collected at regular time intervals (gray vertical lines in Fig. 5.7c) and the frequency of the states for the estimator of the probability distribution (cf. Fig. 5.7d).

5.2.3. Results

In Fig. 5.8a the typical measurement setup for a Bell experiment is shown. We produce a pair of entangled spins from some source and distribute them to two parties Alice and Bob. The two spins constitute the Bell pair. Prior to the experiment Alice and Bob each select two measurements a_1, a_2, b_1, b_2 that they can perform. For each spin pair they *independently* decide which of their two measurements they will actually perform. We choose a single angle Θ to parameterize all 4 measurements[26] according to

$$a_1 = 0 \quad a_2 = 2\Theta \quad b_1 = \Theta \quad b_2 = -\Theta. \tag{5.41}$$

We choose exemplarily the Bell witness:

$$\mathcal{B}(\Theta) = E_{a_1,b_1} + E_{a_1,b_2} + E_{a_2,b_1} - E_{a_2,b_2} \tag{5.42}$$

to demonstrate the evolution of the observability for the entanglement in Fig. 5.8b. The expectation values $E_{x,y}$ are the mean adjusted covariances:

$$E_{x,y} = \langle S^x S^y \rangle - \langle S^x \rangle \langle S^y \rangle \tag{5.43}$$

As all the absolute values of the expectation values are bounded by unity $|E_{x,y}| \leq 1$, the Bell witnesses are constrained to $|\mathcal{B}| \leq 2$ for classical systems [Bell, 2004]. Therefore the observation of $|\mathcal{B}| \geq 2$ is a proof of quantum entanglement. Conversely the adherence

[25] Note: This W_{ij} is only the intra-layer connection matrix, i.e., the block off-diagonals of the full matrix we discussed before.

[26] This does not describe all possible measurements but allows us to produce Fig. 5.8b.

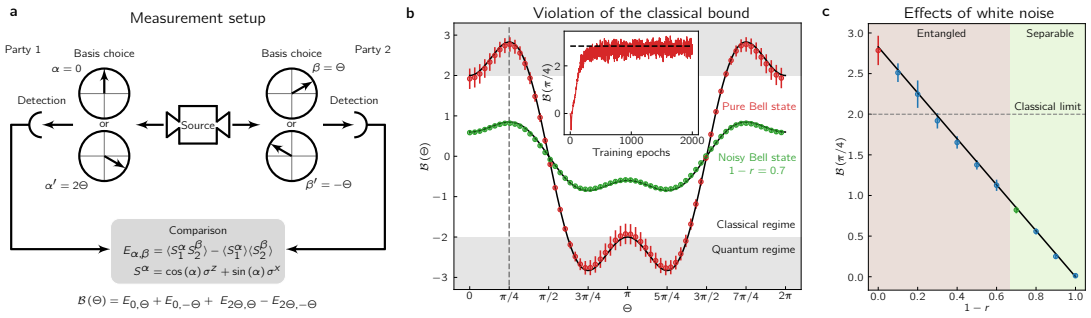

Figure 5.8.: **Measuring a Bell witness: a** Illustration of a typical Bell test scenario. One entangled spin pair is distributed to two parties, which then perform independent measurements. The comparison between the outcomes reveals correlations that cannot be captured by classical local hidden variable theories if the Bell witness $|B(\Theta)|$ exceeds the classical limit of 2. **b** Bell witness evaluated on the final results of the density matrix of the Bell state $\rho_B = |\Psi^+\rangle\langle\Psi^+|$ on BrainScaleS-2 with $M = 20$ hidden neurons. The inset shows the evolution of the Bell witness at $\Theta = \frac{\pi}{4}$ as a function of the training epoch. In the main frame the marker and bars depict the mean and standard deviation over the last 200 training epochs for all angles Θ. Note that all evaluations are performed on the same network state. Adding white noise to the pure Bell state results in the Werner state $\rho_W = r\rho_B + \frac{1-r}{4}\mathbb{1}$. The green points correspond to $r = 0.3$. **c** Bell witness at $\Theta = \frac{\pi}{4}$ for a Werner state as a function of noise strength. The exact solution (black) is well captured for both entangled and separable states, with fluctuations (error bars) decreasing with increasing noise. Figure taken from [Czischek et al., 2020].

to the classical bound does not provide information as entanglement is not observed when considering the spin in a single basis. For our choice of target Bell state $\left|\Psi^{\mathrm{BP}}\right\rangle$ (Eq. (5.32)) and our choice of Bell witness $\mathcal{B}$ (Eq. (5.42)) the violation is at its maximum for $\Theta = \frac{\pi}{4}$.

Fig. 5.8b shows the observed $\mathcal{B}(\Theta)$ for the pure Bell state (red) and a Werner state with a noise component of $r = 0.3$ (green) for all considered angles Θ. The shown values are the mean and variances of the last 200 training iterations. Each line corresponds to one particular state corresponding to a single network configuration. In other words all red (green) dots originate from the same sample[27] and only the evaluation changes. We find good agreement for all choices of Θ.

The inset of Fig. 5.8b shows the evolution of $\mathcal{B}\left(\frac{\pi}{4}\right)$ over the course of the training.

[27] We could repeat the sampling run once per Θ choice, but this would not yield significantly different results.

Most of the learning happens within the first 400 epochs, with only a slight drift happening beyond 1000 epochs. This depends slightly on the choice of hidden layer size and quantum state (cf. Fig. 5.9a and c). We find that we are able to approximate all possible states of a 2-spin system within the variations. In order to show this we mix in thermal noise resulting in a Werner state with density matrix

$$\rho_{\mathrm{W}} = r\rho_{\mathrm{B}} + \frac{1-r}{4}\mathbb{1} \tag{5.44}$$

In Fig. 5.8c we show different Werner states and thereby different training runs. We find that the variations and deviations decrease for higher noise contributions r.

For higher values of r the system becomes more unordered and the resulting probability distribution becomes more uniform. For a purely thermal state, the resulting probability distribution P is the uniform distribution. It is not surprising that we can approximate this kind of probability distribution P better. While we found in Section 3.3 that even the single-neuron activation function is not entirely trivial to predict, we can choose (at least in theory) the mean target activation with arbitrary precision. Given this we are still able to produce arbitrary *factorizing* distributions up to the resolution limit of our calibration. The uniform distribution also factorizes and as such it requires little in terms of synaptic connections, which we learned in Section 3.3.2 are the dominating sources of deviations.

Performance and scaling

For the pure Bell state – arguably the most challenging of the states we looked at – we investigate the performance scaling in Fig. 5.9a. We measure the performance both on the quantum side via the quantum fidelity (main plot of Fig. 5.9a):

$$\mathcal{F}(\rho_{\mathrm{B}}, \rho_{\mathrm{N}}) = \mathrm{Tr}\left[\sqrt{\sqrt{\rho_{\mathrm{B}}}\rho_{\mathrm{N}}\sqrt{\rho_{\mathrm{B}}}}\right] \tag{5.45}$$

as well as on the probability distribution side via the Kullback-Leibler divergence DKL (inset of Fig. 5.9a, cf. Eq. (2.38)). The quantum fidelity measures the overlap of the achieved state with the target state.

For increasing hidden layer size (number of neurons M) the performance increases. It saturates for $M = 30$ near a fidelity $F \approx 98\,\%$ with a corresponding DKL of $\mathrm{DKL} \lesssim 1 \times 10^{-2}$. Comparative implementations with traditional RBMs require significantly less hidden neurons (order 5 [Carrasquilla et al., 2019]) to form the distribution appropriately. This performance regression is most likely due to synaptic connections on BrainScaleS-2 being implemented as 6-bit weights[28]. From the discussion of Section 4.1.2 and Section 2.2.4 we know that the actual influence of the synaptic weight W is dependent on both the neuron as well as the noise configuration. As such we can shift the meaning of the synaptic weights from a lower maximum and finer resolution

[28] One could argue for 7-bit, as it really is two unsigned 6-bit circuits. But when compared to 64-bit floating point numbers that distinction is academic. In addition, we could further optimize the absolute scale of the weights further by choosing different noise parameters, cf. Section 4.1.

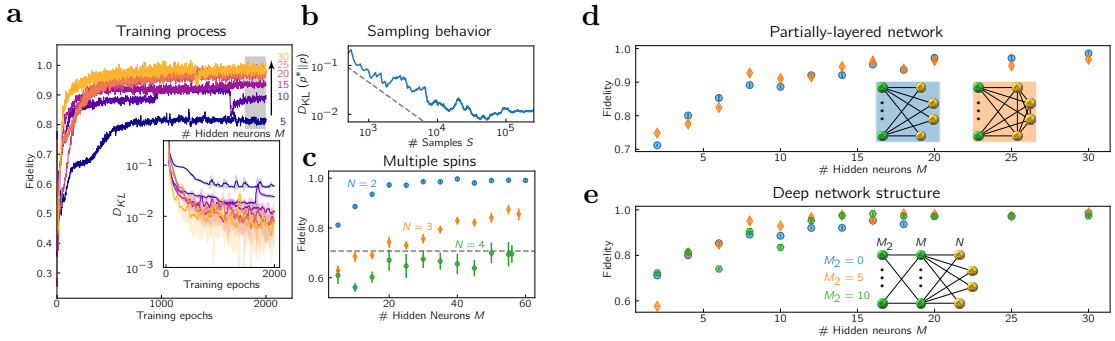

Figure 5.9.: **Performance of the neuromorphic implementation: a** Training performance for the pure Bell state ρ_B in a 2-layer network. Quality is measured by the quantum fidelity (main frame) and DKL (inset) for different numbers of hidden neurons (color coded). **b** DKL evolution as a function of samples for the last training epoch. The final flattening is the limit achieved learning quality at the end. **c** Final quantum fidelity for the considered Bell and GHZ states $|\psi\rangle = |\uparrow\rangle^{\otimes N} + |\downarrow\rangle^{\otimes N}$ for $N = 2, 3, 4$ and for different hidden layer sizes. The numbers reported here are averages over the last 200 training steps (gray shade in **a**). **d** Quantum fidelity as a function of hidden neurons for the strictly layered network (blue) and with additional visible-visible connections (orange). **e** Quantum fidelity for a deeper network architecture where the second hidden layer has $M_2 = 5$ (orange) and $M_2 = 10$ (green) neurons. Figure taken from [Czischek et al., 2020].

to a higher maximum interaction with a more coarse-grained resolution. We did some optimization here, but stopped at global changes to the number and weights of the noise connections of the neurons. In principle changing the number of connections per neuron could allow for more strongly excitable and less strongly excitable neurons and thereby somewhat alleviate this restriction[29].

We (somewhat) compensate for the lack of resolution in each synaptic weight by increased number of hidden units. However, this workaround comes at the price of an enlarged state space, which in turn increases the number of samples required to accurately recover the correlations. At least for $M < 40$ an increase in the number of hidden neurons both increases overall performance as well as training speed (cf. Fig. 5.9d). Again, we see most of the performance is gained in the first 500 epochs. There are some variations observable (e.g. around epoch 1000 for $M = 10$), that are due to underlying (and largely not understood) changes of BrainScaleS-2 system. We are aware of some temperature dependency, but this should not affect the performance on these time scales. At the time of writing this thesis the most likely explanation is network interference of

[29]This is not entirely straightforward, as we up to a point rely on the symmetry of the interaction ($w_{ij} = w_{ji}$ in the full-matrix formulation), which would then be explicitly broken. See also Chapter 6.

other experiments.

In general, the sampling behavior within a run is as expected in Fig. 5.9b, with the DKL decreasing roughly as $\frac{1}{N}$ until it reaches saturation. A single run of 10 ms corresponds to roughly 2000 independent samples. As such we would be limited to about DKL $\approx 5 \times 10^{-2}$ if we could not combine multiple runs. This accuracy would not be sufficient for reproduction of the desired quantum states.

In Fig. 5.9c we see the scaling behavior for both different hidden layer sizes (x-axis) as well as different quantum system sizes (colors). For the 2-spin system we see that the performance does not degrade for the available system size ($M < 60$) and reaches maximum fidelity around $M = 30$. For 3- and 4-spin states we take pure GHZ states [Greenberger et al., 1989] which are the large system analogs for the Bell state. For the 3-spin state we see an increased performance within the available system size, indicating that $F = 1$ could be achieved with a larger system. For the 4-spin state we see some marginal improvement up to $M = 30$ before the performance stagnates. There are multiple possible explanations for this lack of convergence: First, the probability distribution corresponding to the 4-spin state already consists of 64 states, it may be that our sampling time is simply insufficient to accurately represent both the original distribution as well as all correlations which form the relevant marginal over the visible layer we care about[30]. Second, the limitation from the 6-bit weights do not allow the target probability distribution to be accurately formed. Or third, the fundamental deviations from the LIF sampling networks (cf. Section 3.3) do not allow the formation of these distributions. It is likely that the reason is a combination of the former two, which are mostly technical in nature and subject to improvement, either by better control over the system or improved routing allowing for larger networks and multiple synaptic connections between two neurons. We should also point out that it is the pure state that offers the biggest challenge. Adding even some thermal noise to the state ρ improves the achievable quantum fidelity F significantly (data not shown).

Different architectures

Finally we investigated the effects of adding additional visible-visible connections (cf. Fig. 5.9d) as well as a second layer (making the network deep, cf. Fig. 5.9e):

Additional visible-visible connections, at least for the 2-spin Bell state, do not offer a significant advantage. At least not with a good activation function, i.e., the activation function is largely symmetric and covers a significant dynamical range (cf. Fig. 5.6g) and the drift at the end of the refractory period is short (cf. Fig. 5.6e). In a prior setup we, mistakenly, used a setup with a significantly larger membrane capacitance C_m, resulting in a significantly increased τ_m. Within this setup – being far from a good sampling setup – the activation function becomes asymmetric, as there is a slow[31] decrease of the ISI for higher biases. Under these conditions additional visible-visible connections improved

[30] The pure sampling error of a 64 state probability distribution even with the increased 225 000 samples is on the order of DKL $= 1 \times 10^{-4}$. It is unclear how this translates to limiting the achievable accuracy through training.

[31] Proportional to $\log \frac{V_\mathrm{thresh} - V_\mathrm{reset}}{\tau_\mathrm{m}}$ plus some constants, [Petrovici, 2015].

performance significantly, while still staying below the here presented quality (data not shown).

Deeper architectures did not provide additional capabilities and under certain circumstances (e.g. $M < M_2$) actively degraded performance. In case of a smaller first hidden layer an additional hidden layer is not expected to be helpful, as all the effect on the visible layer is constrained by the first hidden layer. In all cases we degraded performance when comparing with a single hidden layer with the same number of neurons as the deep network had in all hidden layers. This is not particularly surprising, as a 16 state probability distribution has only limited complexity that the hidden layer needs to mediate. Hence, the richer distributions of $p(h_1|h_2)p(h_2)$, as compared to the single factorizing distribution $p(h)$[32], does not provide noticeable benefit. Our lack of observed improvement here, however, does not mean that it might not be useful for more complicated setups. Here, we focused on demonstrating the flexibility of the network configuration rather than doing a systematic search of all possible hidden-layer configurations.

[32]This is somewhat simplistic, as the hidden layer distribution is also formed by the visible layer and hence does not factorize completely. However, we are *not* free to choose $p(v)$ arbitrarily and hence the freedom in $p(h)$ is the same as in a factorizing distribution.

6. Discussion and Outlook

If we had to summarize what we learned throughout this thesis in only one sentence then it would have to be: The dynamics of single Leaky-integrate and fire (LIF) neurons, and even more so the dynamics of ensembles of those, are complicated and a description as a Boltzmann distribution is incomplete. Even the extended Buesing model does not adequately capture the nature of the LIF dynamics. In particular, we have seen that the dynamics of Buesing neurons are far less reactive to a change in input (less "bursty"). For large-scale ensembles we identified the influence of the exponential PSP shape as the main origin of changes in the phase diagram of the Buesing neurons.

We started our discussion with the introduction of the necessary background in Chapter 2. Here, we introduced the notion of LIF neurons as an abstraction of biological neurons and established the framework of probabilistic computing with binary neurons that we uses throughout this thesis. After sketching the LIF sampling framework [Petrovici et al., 2016] at the end of Chapter 2, we turned to take a look at the complications arising through the autocorrelation introduced by the exponentially shaped synaptic interactions (Sections 3.1.2 and 3.2). In Section 3.3 we then explained in detail the spike response function $\rho(u)$ of LIF neurons under high-frequency noise stimulus. We generated the LIF Markov model (LMM) by looking at an ensemble of LIF neurons under the same noise stimulus. This allowed us to formulate the membrane potential distribution function $f(u;t)$ which allows the deduction of the spike response $\rho(u)$. To simplify the mathematical treatment we reduced the synaptic time constant of, and thus the autocorrelation generated by, the noise input. This allowed us to keep the internal state of the LMM to one internal variable ζ and having its spike response function $\rho(u)$ only additionally depend on the membrane potential u. While we did show that the LMM describes the dynamics of a network of LIF sampling neurons significantly better than a Boltzmann distribution, it came at the price of the closed-form description. At the same time, the LMM only relies on the synaptic input integration and the hard refractory time. The precise form of both the neuron and the synapse should not make a significant difference and as such the LIF neuron model should be seen as an example rather than an essential precondition.

In Chapter 4 we then shifted our focus to the dynamics of ensembles of spiking neurons. We started by introducing the notion of temperature of binary Boltzmann distributions in Section 4.1 as the width of the activation function. This width is generated by the stochastic noise input and depends on both its frequency and weight as well as the neuron parameters. We found that a bias-free – meaning an unshifted activation function – modulation of the temperature can be implemented by a change of the noise rates r_{exc} and r_{inh}. This bias-free modulation requires a tight and precise coupling between the two noise rates $r_{\text{exc}}, r_{\text{inh}}$ as otherwise a shift fulfills the role of a bias change

in the Boltzmann distribution. This offers a biologically plausible replacement of the bias implementation via the leak potential V_l as it moves its training process also to long-term synaptic plasticity, similar to the training of the weight. Finally, we showed that the functional implication of large-scale activity fluctuations, like the cortical oscillations, can implement tempering by showing the entropy evolution of the resulting distributions.

With the ability to control both temperature T and bias b of an LIF sampling neuron, we turned to the highly regular network structure of the classical Ising model in Sections 4.3 and 4.4. By our description as Boltzmann machines – which is mathematically equivalent to that of spin glasses – we expect all features from the classical Ising model to transfer to our spiking networks. However, we found that the phase diagram of such 2D nearest-neighbor-connected networks, strongly depends on the microscopic neuron model. For the extended Buesing model, both the shape within the refractory period and the lingering interaction after the end of the refractory period qualitatively modify the phase diagram. In particular, we can find cooling schedules (i.e., only changes in the width of the activation function) that exhibit neutral-on-off or neutral-off-on transitions of the activity when cooling from hot-to-cold temperatures at static external fields h. This is even after we correct for a possible bias due to the incorrect assumption on the scaling between weight and bias parameters. Nevertheless, we could recover the critical exponent $\gamma = \frac{7}{4}$ that describes the divergence of the susceptibility $\chi = \frac{\partial A}{\partial b}$ around the critical point $(T = T_{\text{crit}}, h = 0)$ [Onsager, 1944]. In order to do this, we needed to – a posteriorily – define the external field-free $(h = 0)$ configurations, as the neutral activity configuration $\Delta A = 0.5$. These form a non-trivial curve $\Delta b(T)$ in the phase diagram. This recovery procedure works for every neuron model we tried. For networks of LIF neurons the phase diagram looks again completely different. This further underlines the result from Section 3.3 and [Gürtler, 2018] that even Buesing neurons with exponential interactions are an incomplete model for networks of leaky neurons. While not explicitly shown in this thesis, the choice of the two-, rather than a higher dimensional, Ising network was *only* to reduce simulation time, and we can recover the expected behavior in higher dimensions equivalently.

After this check of our theoretical understanding we turned to functional implementations of this Bayesian compute model on neuromorphic hardware in Chapter 5. As expected from Section 3.2 the precise nature of the spike sources was shown to not be a major impairment for the sampling process. The precision of the imprinted distribution on both BrainScaleS-1 and BrainScaleS-2[1] was mainly constrained by the achievable strength and resolution of the single synaptic inputs. Nevertheless, we were able to implement a high-level classification task ((f-)MNIST, cf. Section 5.1) on BrainScaleS-1, with all its analog trial-to-trial variations and 4-bit synaptic connections. This represents the (semi-)large-scale end of the spectrum of Bayesian networks, where the true underlying distribution $p(\vec{v})$ is unknown and only a set of samples from it is available for training. On the other end of this spectrum lies the representation of quantum states in Section 5.2. Here we know, and are required to reproduce, the complete target distribution $p(\vec{v})$ in order to be able to reconstruct the associated density matrix ρ and thereby

[1] The used chips where HICANNv4 and HICANN-Xv1, respectively.

calculate all desired expectation values. While the larger weight resolution (6-bit instead of 4-bit) of BrainScaleS-2 was certainly helpful we traded this for stricter size constrains of the single chip system.

In summary we found that the understanding of LIF sampling as Boltzmann machines is sufficient for the implementation of high-level tasks and the description of *parts* of the distribution. For a full description on the other hand a more detailed model is required.

Outlook

Throughout this thesis, we learned a lot about the intricate dynamics of networks of LIF sampling networks and all the fine complications that arise. We learned about deep connections between various micro and macro effects. This furthers us along the path of connecting brain science to statistical physics, unlocking the ability to use many of the tools developed by the latter. With our current state of understanding there are a number of enticing research angles:

1. Can we improve the training of these spiking sampling networks by using our improved understanding?

   Here we mean, can we improve the training in principle? The LMM offers a better description of the underlying *dynamics* of the LIF neurons, can we use this description to derive improved learning rules than the assumption of Boltzmann distributions allows?

2. Is the binary coding enough?

   Closely related to the derivation of a better learning rule is the question whether a binary state $z \in \{0, 1\}$ has sufficient representative power to describe an LIF neuron adequately[2]. The LMM suggests that there is a significant difference in the dynamical state between times shortly after the end of the refractory period and long after the end of the refractory period and it may be that we require more than 1-bit of information to encode this dynamical state.

3. Can we improve the training on neuromorphic hardware?

   For both hardware implementations in Chapter 5 only a fraction of the total time was actually spent on emulating the network. BrainScaleS-2 features the plasticity processing unit (PPU), which can update the connectivity matrix based on on-chip observables. In particular, we could use spike-timing-dependent plasticity (STDP) to implement the idea of event-driven contrastive divergence [Neftci et al., 2014][3] if we use the on-chip correlation sensors.

[2] As the author's theoretical physics I tutor put it: "You are free to choose any coordinate system you want, but some make your problem easier."

[3] It really is an event-driven wake-sleep algorithm but the original author used the terminology "contrastive divergence" in the paper title, so we honor his decision. This is currently work in progress by a master student of mine, Timo Gierlich [Gierlich, 2020], and would remove the expensive host-chip-loop from the training procedure.

There is always the aim for larger systems, both on hardware and in software. On hardware, we would like to implement large image datasets on the one hand, and large (enough) scale Ising networks on the other. Both would allow us to make stronger statements regarding both the applicability of our parameter relation schemes on imprecise substrates and actual performance comparisons with respect to other platforms.

Regarding the quantum state representation there are also more "out there" questions: Each quantum state ρ is associated with a probability distribution p and continuous changes in ρ lead to continuous changes in p. The latter we associate with the changes of the network parameters w, b, i.e., plasticity or learning. This begs the question: Can we find a learning rule such that it implements changes on p that correspond to the (time)evolution of the quantum state ρ?

The other large and inconcrete question is whether we can assign a meaning to the intrinsic dynamics of the sampling course. So far, we assume that it is *only* the steady-state distribution of the network that is relevant. However, the reaction time of humans are on the order of 50 ms to 500 ms which corresponds to only a few dynamical time scales of the single neurons. As such, it is unlikely that the steady-state distribution is meaningfully captured in reactions to external inputs. This implies that either, the conditional distributions that are created due to sensory input collapse to very small modes or the brain has two fundamentally different modes of operation. This links up with other ongoing projects on BrainScaleS-2, where for example image classification with feed-forward networks shows results in less than two dynamical time scales [Göltz et al., 2019, Cramer et al., 2020a] for the price of not implementing generative or Bayesian properties. These systems tend to use a different coding scheme, which assign meaning directly based on single spike times, rather than the complex correlation between multiple spike trains we use in the sampling framework.

As such, the adequate summary may be: Even though we started with less knowledge than we have now, we also generated more questions than we have answered. This seems to be the nature of science. Our results here cast doubt upon the recently repeated claim that *Hopfield networks are all you need* [Ramsauer et al., 2020] – which essentially corresponds to the restriction to two-point correlations in Boltzmann distributions – as the dynamics of the neurons are not adequately captured by binary states. As we are standing on the shoulders of giants, we hope that the knowledge developed throughout this thesis is some small contribution to the understanding of spiking networks, and ultimately the brain.

7. Acknowledgments

> Bones mend. Regrets stay with you forever.
>
> *Kvothe, Kingkiller*
> *Name of the Wind by P. Rothfuss*

This part is the, woefully inadequate, attempt to form a list of all the people that supported me throughout the journey of my PhD, shared the burden we all experienced and made this work possible. I'm eternally indebted:

To Doro, my girlfriend, for always being there for me and all her love. Without you my life would be a significantly sadder affair. I know I'm not an easy person and there are enough points where I annoy you. While I may never understand why I'm good enough for you, I love you and the last 4 years would not have been the same without you.

To the late Prof. Karlheinz Meier, for forming a unique research environment with the Electronic Vision(s) group and tirelessly championing the BrainScaleS neuromorphic platforms.

To Dr. Mihai Petrovici, for never accepting half thought through statements and tirelessly continuing all our discussions. While the reason I joined the Electronic Vision(s) group was Prof. Meier, Mihai is the reason I stayed.

To Dr. Johannes Schemmel, for stepping into the shoes of Prof. Meier. Both as a group leader for the Vision(s) and as the referee of my thesis.

To Prof. Dr. Thomas Gasenzer for agreeing to be my second referee and the patience in all our discussion regarding the relationship between quantum states and neuromorphic systems over the last years.

To Prof. Dr. Manfred Salmhofer and Prof. Dr. Selim Jochim, for agreeing to take part in my defense.

To Dominik Dold and Akos Kungl, for being the best office mates around, for all the – at times exhausting, but always memorable – discussions. It was a pleasure working with you in the past and I can only hope that the future holds opportunities for collaboration. I wish you all the best.

To Nico Gürtler, Julian Göltz, Madison Cotteret and Timo Gierlich, for being exceptional master students which had the misfortune to have been supervised by me.

To Agnes Korcsak-Gorzo, for the collaboration regarding the implementation of tempering in spiking neural networks.

To all the former members of the TMA group, for all the interesting discussions and critical thinking I got taught here. You definitely left your mark.

To Dr. Eric Müller and his softies, for providing the operational support for the Electron Vision(s). It is in larger part to his credit that there was nearly no effect of the Covid-19 restrictions onto our work. While this is simple for the software simulation, it takes significant effort for the hardware parts. And also for introducing me to spack. The work for it taught me more about software development than I ever wanted to know, but it was also very rewarding.

To Sebastian Billaudelle and Benjamin Cramer, for being the high-priests of the BrainScaleS-2 system. Without you and your willingness to answer all our questions the work in [Czischek et al., 2020] would not have been possible.

To the rest of the Electronic Vision(s) group, for providing a, while at times opinionated and certainly unique, research environment with singular opportunities. And even after 5 years, I still cannot keep up with you at lunch.

To Dr. Martin Gärtner and Dr. Stefanie Czischek, for, together with Prof. Dr. Thomas Gasenzer, investing the time to have the countless discussions that lead to [Czischek et al., 2020].

To the research lab of Prof. Dr. Walter Senn in Bern, for hosting me for a very nice semester and introducing me to a clean river, even though it is quite cold. While you haven't broken my love for Heidelberg, I thoroughly enjoyed my time with you.

To my family, for all the support. While, at times, it is stressful that you have such unwavering faith in me, it does make my own doubts easier to bear.

To my friends back home, here in Heidelberg, Karlsruhe, Jena, Greifswald, Oxford, Tübingen and where you all move to, for being there for me, for sharing many nice barbecues and enduring my endless need for discussions.

Finally, this work would not have been possible without the financial support from the Manfred Stärk foundation. He acted as the incredibly generous mecenas of the TMA group enabling not only our research but also the visit to several workshops and conferences by me and my colleagues. The yearly symposiums we conduct with him as an interested visitor and also speaker are always very rewarding.

It would have also not been possible without access to the supercomputing facilities of the state of Baden-Württemberg. I acknowledge the 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).

A. Calculations

A.1. Conditional Probability

Calculating the conditional probability from the normalization

$$p(z_k = 1|p_{z_{\setminus k}}) + p(z_k = 0|z_{\setminus k}) = 1 \tag{A.1}$$

and the definition of the membrane potential

$$u_k = \log \frac{p(z_k = 1|p_{z_{\setminus k}})}{p(z_k = 0|z_{\setminus k})} \tag{A.2}$$

$$\Rightarrow e^{u_k} = \frac{p(z_k = 1|p_{z_{\setminus k}})}{p(z_k = 0|z_{\setminus k})} \tag{A.3}$$

yields

$$\begin{aligned} p(z_k = 1|p_{z_{\setminus k}}) &= e^{u_k} p(z_k = 0|z_{\setminus k}) && \text{(A.4)} \\ &= e^{u_k} \left(1 - p(z_k = 1|z_{\setminus k})\right) && \text{(A.5)} \\ &= \frac{e^{u_k}}{1 + e^{u_k}} && \text{(A.6)} \\ &= \frac{1}{1 + e^{-u_k}} = \sigma\left(u_k\right) && \text{(A.7)} \end{aligned}$$

We also attach a general calculation of the membrane potential of a sampler from the BM:

$$\begin{aligned} u_k &= \log \frac{e^{-E(z_k=1)}}{e^{-E(z_k=0)}} && \text{(A.8)} \\ &= \log \frac{e^{\frac{1}{2}\sum_{i,j\neq k} W_{ij} z_i z_j + \sum_{i\neq k} b_i z_i + \frac{1}{2}\sum_i W_{ki} z_i + \frac{1}{2}\sum_i W_{ik} z_i + b_k}}{e^{\frac{1}{2}\sum_{i,j\neq k} W_{ij} z_i z_j + \sum_{i\neq k} b_i z_i + 0}} && \text{(A.9)} \\ &= \log e^{\frac{1}{2}\sum_i (W_{ki} + W_{ik}) z_i + b_k} && \text{(A.10)} \\ &= \sum_i W_{ki} z_i + b_k && \text{(A.11)} \end{aligned}$$

For the last equality we use the fact that connectivity matrix of the BM are symmetric, i.e., $W_{ki} = W_{ik}\ \forall i, k$. This is a reminder of the symmetric nature of interactions in physics ("actio = reactio").

A.2. Spin to Neural relations

A.2.1. Logistic function and hyperbolic tangent

Explicit calculation of the relationship between the hyperbolic tangent function to the activation function of the neural network description:

$$\frac{1}{2}(1+\tanh(x)) = \frac{1+\frac{\sinh(x)}{\cosh(x)}}{2} \tag{A.12}$$

$$= \frac{1+\frac{\frac{e^x-e^{-x}}{2}}{\frac{e^x+e^{-x}}{2}}}{2} \tag{A.13}$$

$$= \frac{e^x+e^{-x}+e^x-e^{-x}}{2\left(e^x-e^{-x}\right)} \tag{A.14}$$

$$= \frac{e^x}{e^x+e^{-x}} \tag{A.15}$$

$$= \frac{1}{1+e^{-2x}} = \sigma(2x) \tag{A.16}$$

A.3. Energy of Two State Systems

A.3.1. Total Energy

We need to check that the update rules from [Petrovici et al., 2016] (Eq. (4.28)) are actually conserving the probability landscape. In order to do this we will explicitly calculate the translated total energy and show that its value (under this translation)

does only change by a constant value due to the description we choose.

$$E = \sum_{\mathrm{i,j}=0}^{n} \frac{1}{2} W_{\mathrm{ij}} z_{\mathrm{i}} z_{\mathrm{j}} + \sum_{i=0}^{n} b_{\mathrm{i}} z_{\mathrm{i}} + C \tag{A.17}$$

$$= \sum_{\mathrm{z_i}=1,\mathrm{z_j}=1} \frac{1}{2} W_{\mathrm{ij}} + \sum_{z_i=1} b_{\mathrm{i}} + C \tag{A.18}$$

$$= \sum_{z_i=1,z_j=1} 2\hat{W}_{\mathrm{ij}} + \sum_{z_i=1} \left(2\hat{b}_{\mathrm{i}} - 2 \sum_{j=0}^{n} \hat{W}_{\mathrm{ij}} \right) + \frac{1}{2} \sum_{i,j=0}^{n} \hat{W}_{ij} - \sum_{i=0}^{n} \hat{b}_i \tag{A.19}$$

$$= \sum_{z_i=1,z_j=1} \left(2\hat{W}_{\mathrm{ij}} - 2\hat{W}_{\mathrm{ij}} + \frac{1}{2}\hat{W}_{\mathrm{ij}} \right) + \sum_{z_i=-1,z_j=-1} \frac{1}{2}\hat{W}_{\mathrm{ij}} \tag{A.20}$$

$$+ \sum_{z_i=1,z_j=-1} \left(-2\hat{W}_{\mathrm{ij}} + \frac{1}{2}\hat{W}_{\mathrm{ij}} \right) + \sum_{z_i=-1,z_j=1} \frac{1}{2}\hat{W}_{\mathrm{ij}} \tag{A.21}$$

$$+ \sum_{z_i=1} \left(2\hat{b}_{\mathrm{i}} - \hat{b}_{\mathrm{i}} \right) - \sum_{z_i=-1} \hat{b}_{\mathrm{i}} \tag{A.22}$$

$$= \sum_{z_i=1,z_j=1} \frac{1}{2}\hat{W}_{\mathrm{ij}} \hat{z}_{\mathrm{i}} \hat{z}_{\mathrm{j}} + \sum_{z_i=-1,z_j=-1} \frac{1}{2}\hat{W}_{\mathrm{ij}} \hat{z}_{\mathrm{i}} \hat{z}_{\mathrm{j}} + \sum_{z_i=1,z_j=-1} \frac{3}{2}\hat{W}_{\mathrm{ij}} \hat{z}_{\mathrm{i}} \hat{z}_{\mathrm{j}} \tag{A.23}$$

$$- \sum_{z_i=-1,z_j=1} \frac{1}{2}\hat{W}_{\mathrm{ij}} \hat{z}_{\mathrm{i}} \hat{z}_{\mathrm{j}} + \sum_{z_i=1} \hat{b}_{\mathrm{i}} \hat{z}_{\mathrm{i}} + \sum_{z_i=-1} \hat{b}_{\mathrm{i}} \hat{z}_{\mathrm{i}} \tag{A.24}$$

$$= \sum_{i,j} \frac{1}{2}\hat{W}_{\mathrm{ij}} \hat{z}_{\mathrm{i}} \hat{z}_{\mathrm{j}} + \sum_{i} \hat{b}_{\mathrm{i}} \hat{z}_{\mathrm{i}} \tag{A.25}$$

$$\tag{A.26}$$

Where we have used the abbreviations $\sum_{i,j} = \sum_{i=1}^{n} \sum_{j=1}^{n}$ and $\sum_{z_i=1} = \sum_{i \in \{k | z_k = 1\}}$.

A.3.2. Transition Probability

The transition probability (spike probability) for a neuron is given by its activation function:

$$p(z_{\mathrm{k}} = 1) = \frac{e^{E(z_k=1)}}{e^{E(z_k=1)} + e^{E(z_k=0)}} = \frac{1}{1 + e^{-u_{\mathrm{k}}}} =: \sigma(-u_{\mathrm{k}}) \tag{A.27}$$

With its "membrane potential" given by

$$u_{\mathrm{k}} = \sum_{i=0} W_{\mathrm{ki}} z_{\mathrm{i}} + b_{\mathrm{i}} \tag{A.28}$$

Translating this into the Ising domain we get:

$$u_{\mathrm{k}} = \sum_{i=0} W_{\mathrm{ki}} z_{\mathrm{i}} + b_{\mathrm{k}} \tag{A.29}$$

$$= \sum_{z_i=1} W_{\mathrm{ki}} + b_{\mathrm{k}} \tag{A.30}$$

$$= \sum_{z_i=1} 4\hat{W}_{\mathrm{ki}} + 2\hat{b}_{\mathrm{k}} - 2\sum_{i=0}^{n} \hat{W}_{\mathrm{ki}} \tag{A.31}$$

$$= \sum_{i=0}^{n} 2\hat{W}_{\mathrm{ki}} \hat{z}_{\mathrm{i}} + 2\hat{b}_{\mathrm{k}} \tag{A.32}$$

$$= 2\hat{u}_{\mathrm{k}} \tag{A.33}$$

The factor of 2 is the reminder of the change in description and reflects the fact that the spacing between the energy levels is stretched by a factor of 2 when going from the neural description to the Ising description.

In other words

$$p(\hat{z}_i = 1) = \frac{e^{E(\hat{z}_i=1)}}{e^{E(\hat{z}_i=1)} + e^{E(\hat{z}_i=-1)}} = \sigma(-2\hat{u}_k) \tag{A.34}$$

as opposed to Eq. (A.27).

Which guarantees us identical behavior of both descriptions, as long as we pay attention whether we have to use the membrane potential (as calculated in Eq. (A.28)) or twice its value for the neurons/spins activation function.

However, it is worth again to point out, that this kind of treatment is not limited to our specific two descriptions, but rather a natural effect of the freedom in choosing our own descriptions of systems. Nature does (should) not care about how we label a two state system! And in this sense the whole calculation here is only necessary to translate the numerical predictions of Ising to our systems and not for their existence.

A.4. Wake-Sleep derivation

We start by rewriting the DKL as:

$$\mathrm{DKL}\left(q||p;\mathcal{W}\right) = \sum_{z} q(\vec{z}) \log \frac{p(\vec{z},\mathcal{W})}{q(\vec{z})} = \sum_{\vec{z}} \left(q(\vec{z}) \log p(\vec{z},\mathcal{W}) - q(\vec{z}) \log q(\vec{z})\right) \tag{A.35}$$

and note that the latter term is independent of the network parameters $\mathcal{W} = (W_{ij}, b_i)$ and therefore drops out when taking the gradient with respect to the parameters:

$$\frac{\partial\, \mathrm{DKL}\left(q||p;\mathcal{W}\right)}{\partial w_{ij}} = \sum_{\{\vec{z}\}} q(\vec{z}) \frac{\partial \log p(\vec{z};\mathcal{W})}{w_{ij}} \tag{A.36}$$

For ease of notation we first calculate:

$$\frac{\partial \log p(\vec{z};\mathcal{W})}{w_{ij}} = \frac{1}{p(\vec{z};\mathcal{W})}\frac{\partial p(\vec{z};\mathcal{W})}{\partial w_{ij}} \tag{A.37}$$

$$= \frac{1}{p(\vec{z};\mathcal{W})}\frac{\partial}{\partial w_{ij}}\left[\frac{\exp\left(-E(\vec{z};\mathcal{W})\right)}{\sum_{\vec{y}}\exp\left(-E(\vec{y};\mathcal{W})\right)}\right] \tag{A.38}$$

$$= \frac{1}{p(\vec{z})}\left\{\frac{\partial_{w_{ij}}\exp\left(-E(\vec{z};\mathcal{W})\right)}{\sum_{\vec{y}}\exp\left(-E(\vec{y};\mathcal{W})\right)} - \frac{\exp\left(-E(\vec{z};\mathcal{W})\right)\partial_{w_{ij}}\sum_{\vec{y}}\exp\left(-E(\vec{y};\mathcal{W})\right)}{\left[\sum_{\vec{y}}\exp\left(-E(\vec{y};\mathcal{W})\right)\right]^2}\right\} \tag{A.39}$$

Using the energy definition Eq. (2.26) and using the restricted connectivity with $w_{ij} \neq 0$ only for intralayer connections between hidden neuron j with state h_j and visible neuron i with state v_i, we find:

$$\frac{\partial p(\vec{v},\vec{h};\mathcal{W})}{\partial w_{ij}} = \left\{\frac{v_i h_j \exp\left(-E(\vec{v},\vec{h};\mathcal{W})\right)}{\sum_{\vec{y}}\exp\left(-E(\vec{y};\mathcal{W})\right)} - \frac{\exp\left(-E(\vec{v},\vec{h};\mathcal{W})\right)\sum_{\vec{y}}\hat{v}_i\hat{h}_j}{\sum_{\vec{y}}\exp\left(-E(\vec{y})\right)}\right\} \tag{A.40}$$

$$\tag{A.41}$$

$$\frac{\partial \mathrm{DKL}\left(p||q\right)}{\partial w_{ij}} = \sum_{\{\vec{v}\}} q\left(\vec{v}\right)\left[-\frac{1}{\tilde{p}\left(\vec{v};\mathcal{W}\right)}\frac{\partial \tilde{p}\left(\vec{v};\mathcal{W}\right)}{\partial w_{ij}} + \frac{1}{Z\left(\mathcal{W}\right)}\frac{\partial Z\left(\mathcal{W}\right)}{\partial w_{ij}}\right] \tag{A.42}$$

$$= \sum_{\{\vec{v}\}} q\left(\vec{v}\right)\left[-\frac{1}{p\left(\vec{v};\mathcal{W}\right)}\left(\sum_{\left\{\vec{h}\right\}} v_i h_j e^{-E\left(\vec{v},\vec{h};\mathcal{W}\right)}\right)\right. \tag{A.43}$$

$$\left. + \frac{1}{Z\left(\mathcal{W}\right)}\left(\sum_{\left\{\vec{v}',\vec{h}\right\}} v_i' h_j e^{-E\left(\vec{v}',\vec{h};\mathcal{W}\right)}\right)\right]$$

$$= \sum_{\{\vec{v}\}} = \langle v_i h_j\rangle_{\text{data}} - \langle v_i h_j\rangle_{\text{model}} \tag{A.44}$$

B. Simulation Parameters

All LIF sampling simulations were performed using the SBS library [Breitwieser et al., 2020] which internally uses PyNN [Davison et al., 2009]. The actual compute task were done with an older version of the NEST simulator [Peyser et al., 2017]. Simulations using the neuralsampler framework (cf. Appendix C.3) used the commit 6210746c2db61eb0804566d772a937781aa23b76.

Here, we will give the parameters used for each simulation, hopefully in a complete enough fashion to allow easy reproduction. Whenever not explicitly noted otherwise the simulator time step is set to 0.1 ms.

B.1. Single PSP

B.1.1. Isolated PSP

For this simulation we directly use the PyNN interface and we use `pyNN.IF_cond_exp` and `pyNN.IF_curr_exp` neurons with parameters:

C_m	0.2 nF
$V_{\text{exc}}^{\text{rev}}$	50 mV
$V_{\text{inh}}^{\text{rev}}$	−50 mV
V_{l}	0 mV
V_{reset}	−0.01 mV
V_{thresh}	0 mV
τ_{m}	2.0 ms
$\tau_{\text{syn}}^{\text{inh}}$	10 ms
$\tau_{\text{syn}}^{\text{exc}}$	10 ms
τ_{ref}	10 ms

Table B.1.: **Single PSP simulation:**For the CUBA case the $V_{\text{exc}}^{\text{rev}}$ and $V_{\text{inh}}^{\text{rev}}$ parameters need to be omitted.

A single input spike was generated at 120 ms and the weight was set to 0.1 using `sim.SpikeSourceArray` connect via `sim.connect`.

B.1.2. TSO Simulations

For this simulation we directly use the PyNN interface and we use `pyNN.IF_curr_exp` neurons with parameters:

C_m	0.2 nF
V_l	0 mV
V_reset	−0.01 mV
V_thresh	0 mV
τ_m	0.1 ms
$\tau_\mathrm{syn}^\mathrm{inh}$	10 ms
$\tau_\mathrm{syn}^\mathrm{exc}$	10 ms
τ_ref	10 ms

Table B.2.: **TSO simulations:**Same neuron parameters as above.

For the PSP traces in Fig. 3.1a we used the same setup as Appendix B.1.1 only exchanging the membrane time constant τ_m to the respective values.

For Fig. 3.1b we set $\tau_\mathrm{m} = 0.1\,\mathrm{ms}$ and varied the ISI of the stimulating spike source using $\Delta t \in \{3, 6, 12, 60\}$ms in the time frame 120 msto180 ms.

For Fig. 3.1c we exchanged the synapse type to `tsodyks2_synapse` of the NEST simulator and used $\tau_\mathrm{facil} \in \{10, 20\}$ms and the ISI setting of 12 ms.

B.1.3. Tail Contribution Simulations

For this simulation we directly use the PyNN interface and we use `pyNN.IF_curr_exp` neurons with parameters:

C_m	0.2 nF
V_l	0 mV
V_reset	−0.01 mV
V_thresh	0 mV
τ_m	0.1 ms
$\tau_\mathrm{syn}^\mathrm{inh}$	10 ms
$\tau_\mathrm{syn}^\mathrm{exc}$	10 ms
τ_ref	10 ms

Table B.3.: **Tail contributions:**Same neuron parameters as above.

We explicitly measure the membrane traces for activities $\langle A \rangle \in \{0.0, 0.1, \ldots 0.9, 1.0\}$ and interpolate in between. The stimulating spike train is generated with ISI $\Delta t = \frac{\tau_\mathrm{ref}}{\langle A \rangle}$ and the same script as for Appendices B.1.1 and B.1.2.

The script for everything up to here can be found in the thesis repository under `scripts/stackedpsps/stackedpsps.py`.

B.2. Noise simulations

The neuron parameters are shared between the different noise simulations and the script can be found in the thesis repository under `scripts/noise/randomnetwork.py`.

C_m	0.2 nF
V_l	0.01 mV
V_reset	−5 mV
V_thresh	0 mV
τ_m	1.0 ms
$\tau_\mathrm{syn}^\mathrm{inh}$	10 ms
$\tau_\mathrm{syn}^\mathrm{exc}$	10 ms
τ_ref	10 ms

Table B.4.: **Noise simulations:**Increased $V_\mathrm{thresh} - V_\mathrm{reset}$.

B.2.1. Poison Simulations

Neuron parameters from Table B.4 and the synapse is either `sim.StaticSynapse` with a weight of 1 or a `tsodyks2_synapse` with parameters:

U	1.0
τ_rec	10 ms
τ_facil	0 ms
weight	1

Table B.5.: **TSO parameters for Poisson noise:**Corresponds to renewing synapses.

The rate of the two Poisson source (`sim.SpikeSourcePoisson`) is set to $r = 1000\,\mathrm{Hz}$ and the simulation duration is $T = 10\,\mathrm{s}$. The binning of spiketrain for the power spectrum happens in 2 ms bins.

B.2.2. Random Network Simulations

We randomly initialize the neuron's membrane potentials according to a normal distribution with mean $\bar{u} = -1\,\mathrm{mV}$ and standard deviation $\sqrt{\mathrm{Var}[u]} = 0.3\,\mathrm{mV}$. The inhibitory connections are made via a `sim.FixedNumberPreConnector(n=k)` and the connections the initialized with `sim.StaticSynapse(weight=wi)`. The used combinations were $(k, w) \in \{(50, -2.), (25, -4.), (5, -1.), (50, -8.)\}$. We feed the superposition of 20 of the thus generated spiketrains to a single neuron with $V_\mathrm{thresh} = 100\,\mathrm{mV}$ which deactivates its spike mechanism. The binning for the power spectrum is again in 2 ms bins.

B.2.3. On Chip

Same as Appendix B.2.2 only that the spiketrains come from a single run of the `hxsampler` (cf. Appendix C.2) with the target rate being set to 80 kHz and a `noisemultiplier=4` was used.

B.3. Tempering Simulations

The simulations were performed by Agnes Korcsak-Gorzo.

C_m	0.2 nF
V_l	−50.0 mV
V_reset	−55.1 mV
V_thresh	−50.0 mV
τ_m	0.1 ms
$\tau_\mathrm{syn}^\mathrm{inh}$	10 ms
$\tau_\mathrm{syn}^\mathrm{exc}$	10 ms
τ_ref	10 ms

Table B.6.: **Noise simulations:**Increased $V_\mathrm{thresh} - V_\mathrm{reset}$.

B.4. Ising Simulations

The Ising simulations Fig. 4.3 were the only ones performed on a local laptop without using the `experiment` setup Appendix C.1 and the `neuralsampling` backend Appendix C.3. It shares most of the parameters with the experiments for Appendix B.5.

The parameters in addition to Table B.7 were augmented by `skripts/Ising/run.py`. Here and in Appendix B.5 we use a temperature setting of $T = 1$ and a default weight of the nearest neighbor connections to $w = 1$. The cooling-down experiment was initialized at $T = 1$ and cooled down to $T = 0.3$ (which in the Ising domain corresponds to $T = 4$ and $T = 1.2$).

delay	1
networkUpdateScheme	InOrder
neuronType	log
nupdates	1000
outputEnv	true
outputScheme	MeanActivity
randomSeed	42 424 243
randomSkip	1 000 000
subsampling	1
synapseType	rect
tauref	1
tausyn	1

Table B.7.: **Ising simulations:**Translation to the $\{-1, 1\}$ domain is done artificially, all calculations happen artificially according to Eq. (4.30).

B.5. Phase diagrams

The phase space diagrams measurements are orchestrated via the `experiment` library Appendix C.1, we give the shared parameters here once and then the overrides in each subsection below. For completeness the generation files are checked into the git repository that contains this thesis and are available e.g. at commit `51518d6ff9bbebfc77d4bbbd8383cf6e07dca93f` in the `parameterfiles` directory.

NeuronType	log
delay	1
networkUpdateScheme	InOrder
nupdates	2200000
subsampling	10
outputEnv	True
outputScheme	MeanActivity
randomSeed	42424243
randomSkip	1000000
synapseType	rect
tauref	100
tausyn	100

Table B.8.: **Base parameters phase space:**Shared parameters for all phase space simulations with the (augmented) Buesing neuron model.

B.5.1. Rect

Here we scan the offset bias Δb and the temperature T (`parameterfiles/ThesisRect`).

Δb	`np.linspace(-0.1, 0.1, 41)`
T	`np.linspace(0.2, 2.0, 10)` + `np.linspace(0.5, 0.7, 21)`

Table B.9.: **Phase space rect:**

B.5.2. Quench origins

The simulations differ by their initialization only. We use the final (internal) states from $(\Delta b, T) = (0.0, 0.59)$ as a stand-in for the critical point (`parameterfiles/ThesisRectCrit`) and from $(\Delta b, T) = (-0.1, 0.5)$ for the off-state initialization (`parameterfiles/ThesisRectOff`). Similar effects are observable for different choices of the ζ-distribution for identical z-distributions (`parameterfiles/ThesisRectFlat`).

B.5.3. Exp

Here we scan the offset bias Δb and the temperature T (`parameterfiles/ThesisExp`).

Δb	`np.linspace(-0.2, -0.7, 11) + np.linspace(-0.3, -0.5, 41)`
T	`np.linspace(0.1, 1.0, 10) + np.linspace(0.65, 0.8, 31 + np.linspace(0.8, 0.9,`

Table B.10.: **Phase space exp:**

B.5.4. Tail

Here we scan the offset bias Δb and the temperature T (`parameterfiles/ThesisTail`).

Δb	`np.linspace(-0.2, -0.7, 11) + np.linspace(-0.35, -0.55, 41)`
T	`np.linspace(0.1, 1.0, 10) + np.linspace(0.65, 0.8, 31 + np.linspace(0.8, 0.9,`

Table B.11.: **Phase space tail:**

B.5.5. Cuto

Here we scan the offset bias Δb and the temperature T (`parameterfiles/ThesisTail`).

Δb	`np.linspace(-0.1, 0.4, 6) + np.linspace(0.05, 0.25, 41)`
T	`np.linspace(0.1, 1.0, 19) + np.linspace(0.25, 0.55, 31`

Table B.12.: **Phase space cuto:**

B.5.6. LIF

We use the LIF parameters:

C_m	$0.002\,\mathrm{nF}$
V_l	$0.0\,\mathrm{mV}$
V_reset	$-0.001\,\mathrm{mV}$
V_thresh	$0.0\,\mathrm{mV}$
τ_m	$0.1\,\mathrm{ms}$
$\tau_\mathrm{syn}^\mathrm{inh}$	$10\,\mathrm{ms}$
$\tau_\mathrm{syn}^\mathrm{exc}$	$10\,\mathrm{ms}$
τ_ref	$10\,\mathrm{ms}$

Table B.13.: **Phase space LIF parameters:**Increased $V_\mathrm{thresh} - V_\mathrm{reset}$.

The simulation for LIF neurons are more involved as we first need to select the the appropriate noise parameters for a bias-free temperature modulation (cf. Section 4.1). We did this by measuring the whole 2D space for excitatory and inhibitory rates (similar to Fig. 4.1) and selecting a line of constant activation function position. The resulting noise rates are:

r_{exc}	r_{inh}	T
183.3 Hz	147.9 Hz	0.178
248.2 Hz	229.1 Hz	0.214
336.0 Hz	336.1 Hz	0.254
454.9 Hz	477.7 Hz	0.299
615.8 Hz	664.3 Hz	0.350
833.8 Hz	911.9 Hz	0.409
1128.8 Hz	1240.7 Hz	0.476
1528.3 Hz	1679.6 Hz	0.554
2069.1 Hz	2264.6 Hz	0.644
2801.4 Hz	3047.1 Hz	0.748
3792.7 Hz	4096.2 Hz	0.869
5134.8 Hz	5503.5 Hz	1.009
6951.9 Hz	7397.1 Hz	1.172
9412.0 Hz	9947.3 Hz	1.361
12 742.7 Hz	13 382.5 Hz	1.582
17 252.1 Hz	18 009.4 Hz	1.837
23 357.2 Hz	24 253.3 Hz	2.135

Table B.14.: **Phase space LIF temperature:**Increased $V_{\text{thresh}} - V_{\text{reset}}$.

Imprinting a correct initial state (i.e. setting all internal parameters) turned out to be surprisingly difficult (the state of the synaptic input is not easily exposed through the `PyNN` and `SBS` interaces). As such no attempt was madde to write the past here.

We scan the offset bias Δb and the temperature T (`parameterfiles/ThesisTail`).

Δb	`np.linspace(-0.5, 0.5, 11)` + `np.linspace(-0.2, 0.0, 21)`
T	`np.linspace(0.3, 1.5, 13)` + `np.linspace(0.5, 0.8, 31`

Table B.15.: **Phase space LIF scan:**

C. Software and Tooling

C.1. Experiment Control on bwNEMO

Most of the results presented in this thesis (e.g. Section 4.1, Section 4.3, parts of Section 5.2) come from large scale parameter studies on the bwHPC supercomputer NEMO. A typical phase diagram in Section 4.3 contains the results of O(1000) simulations with additional ones required for selecting these. Each simulation requires, depending on network size and duration, anywhere between few minutes and multiple hours of CPU time. Due to the parameterstudylike nature of the simulations no attempt to parallize a single simulation was taken, rather single thread performance was somewhat optimized and a tool put in place to rapidly start, modify and collect results from massive numbers of single simulations.

NEMO is a fairly typical, if well administered, HPC system, i.e., it provides a SLURM-based scheduler with the user-facing interface MOAB. The user submits a single bash script that controls the job and annotates it with the resources (mainly number of CPUs, nodes, amount of memory and walltime) requested. SLURM than takes care to allocate a suitable node, submit the job there, monitor the execution (in particular, that memory and walltime requirements are not exceeded) and collects the job exit codes. Since our simulations can be quite short-lived the delay between job submission and earliest start-up (which is O(min)) can become significant.

In this appendix we describe the implementation of the experiment controller, that

- Generates the single parameter finds from a higher level description of the study
- Groups these simulations such that each job fills a complete node (20 cores) and runs reasonably long (hours) in order not to tax the scheduling system
- Checks the experiment results
- Collects the simulation results and provides a single report file to analyse further

C.1.1. High level interface and experimentfile layout

The python tool experiment.py is always called on a YAML file that contains the high level description of the experiment. This file contains at least the following parts:

- `executionParams`: high level information for the scheduler.

  Which cluster does this run on (only NEMO is actually implemented) and with which parameters to schedule the jobs (number of cpus, nodes, walltime etc) as

well as an estimate for the duration of a single job. The latter is used to group multiple simulations in order to keep the runtimes reasonably long.

- `experimentName`: the name of the experiment. Mostly for the user to identify the experiment, used to backup the experiment YAML and as a parent directory name for the simulation folders.
- `executable`: The executable that consumes the simulation file and performs the actual simulation, including the analysis. This differs between the different parts of the thesis, but are generally python scripts that manage the simulation setup (e.g. the SBS network construction), execution (e.g. the generation of the spike trains and state series) and the analysis (e.g. the mean activity of the network) which results in a JSON-file called ‘analysis‘.
- `envscript`: A sourceable bash script that sets required environment variables. Mostly needed before the container setup was there, but allows for a fully modifiable and reproducible environment.
- `stub`: The simulation file that will be consumed by ‘executable‘. It can still contain replacement keys, from which the parameter study will be generated. See replacements below.
- `replacements`: a list of key-value pairs to generate the parameter study. The key is a value of the stub dictionary above and its value is a list of all values that should be in the parameter study. A simulation will be generated for each element in the Cartesian product of all replacement values. For convenience access to numpy's linspace and logspace functions is available via setting the value to a list with the first value being ‘func‘ and the second one being ‘linspace‘ or ‘logspace‘ with the following three arguments being the numbers to be passed to the respective numpy function. A combination of multiple of these ‘[func, linspace, value, value, value]‘ is possible via the ‘[func, multiple, linspacelist, linspacelist]‘ syntax.

The first step in the experiment execution is the setup of the single simulation parameter files which happens via the ‘generate-sims‘ flag. It produces a new simulation directory with subdirectories for each element in the Cartesian product of the lists in ‘replacements‘. The name of the subdirectories is derived from the parameters that are being varied and a ‘folder_template‘ with the order is placed in each resulting ‘sim.json‘.

Additionally a bash script named ‘job‘ is placed in each of the subdirectories, it contains the correct path to the subdirectory, the requested environment modifications, the (modified) content of the simulation stub as well as checkpoints for simulation start, end and analysis end. The latter are marked by creating empty files for each checkpoint, which allows for fast checking of the completeness and error state of the experiment (independent from the submitted jobs). In order for ‘executable‘ to be correctly work with the collection and distribution scheme it needs to adhere to the naming conventions required by ‘experiment.py‘. De facto that means that we wrote a short python script handling the interfacing which in turn generates a ‘run.yaml‘ that is handed to the actual

simulator (neuralsampling based simulations in Section 4.3) or the NEST frontend (SBS based simulations). This script is called by ‘job‘ and does the analysis and run in two separate steps consuming different parts of the parameter dictionary in ‘stub‘.

The generation step also submits the jobs (together with their estimated time requirement) to the PENDING queue of experiment.py. This allows for multiple experiments to be setup and collected into a common job. The actual job submission happens via the ‘execute-sims‘ flag, which consumes the ‘walltime‘ and ‘cpus‘ part of the ‘executionParams‘ part of the last experiment YAML file. According to these (which in practice are always 20 cores and 10-24 hours) it collects a group of pending simulations and collects a list of their directories in a ‘taskfile‘. For each ‘taskfile‘ a bash script is generated that uses a python multiprocessing-Pool with the requested number of CPUs as the number of workers and calls job in each of these directories. This script is then submitted to the scheduler and also takes care of redirecting the stdout and stderr streams into files in the respective simulation's directory.

The execution itself is of relatively little interest here as this is completely handled by SLURM and the multiprocessing pool takes care of using all allocated cores. In practice we achieve more than 99 % efficiency with this, so the convoluted way of running the simulations does not actually const runtime. The job behaves exactly as it would when manually executing ‘bash path/to/simulation/job‘, which is nice for debugging purposes.

Result collection happens via the ‘summarize-sims‘ flag. This collects ‘analysis‘ JSON files that are generated by the single simulations in a single JSON for the whole experiment. In order for these not to become prohibitively large care needs to be taken to limit the amount of raw data put into the ‘analysis‘ file. The collected JSON additionally contains a copy of the ‘sim.json‘ file in order to ease analysis.

C.1.2. Possible extensions

This is largely a collection of this would be nice to have, but I never got around to implement it and actually also didn't really need it.

- Monte-Carlo parameter selection. In most cases we are not really interested in a Cartesian grid of simulation parameters, but rather want to test a certain part of the parameter space. The multiple linspaces things is a very poor mans version of this, it allows for a denser testing of a certain hypercube over all others.

- Learned ETAs. In principle the ETA of a job is a relatively simple function of the input parameters and, in particular, need not be the same for all simulations. As such it would be nice to have the tool assign a likely time requirement on a single parameter set level.

- Only add missing simulations. In case of some simulations timing out the simplest way to redo the simulations is to requeue all and then returning early on checking that a successful simulation was already done. This is an unnecessary burden on both the scheduler as well as our resource requirement. In principle the information of whether a requeue is necessary is available at experiment execution.

- Timout notification emails. By default the user is notified via email when a job is killed due to a timeout. Currently this email does not contain any information on which simulations were part of this job running into the timeout.

C.2. HXSampling

This section describes the HXSampler object which encapsulates the sampling framework implement for the BrainScaleS-2 system. It uses a black-boxed implementation of a recurrently connected network of up to 128 spike sources which is provided by Sebastian Billaudelle and Benjamin Cramer and is based directly on the `haldls` library. When a pyNN-based frontend to the BrainScaleS-2 system becomes available, the internal implementation will be switched, but the externally visible API should be retained. This section will serve as a high-level description of the aims and thoughts behind the implementation. While the particular implementation should be checked in the source code as it will most likely have changed, the idea behind the object will stay the same.

C.2.1. Idea

On a very high-level a sampler needs to expose a method to reconfigure the target distribution – in our case specified by the weights and biases – and a method to generate a new sample set.

The `HXSampler` class holds two members for the `logical_weight` and `logical_bias` `numpy` arrays. These do not automatically apply to the hardware configuration but an update has to be manually triggered by manually calling the `update_parameters` method. The arrays can contain normal 64-bit floating point numbers which will be casted to by rounding down to the next smaller integer and clipped to the available range on hardware (i.e. ± 64 for the weights and $[0, 1022]$ for the biases). This covers the reconfiguration interface.

The sampling interface is `get_samples` which optionally includes the parameter write and returns a list of sampled states. For this to work this method needs to be provided with the effective refractory times – which are used in the state assignment from the spike trains. These can either be explicitly provided or the `HXSampler` implementation can measure them directly.

There is an underlying `run_network` function that does the actual spikes-in-spikes-out experiment run, which is internally used. But the user-visible interface is mostly covered by `get_samples` and `logical_weight/logical_bias`. In Appendix C.2.3 we present a list of the currently exposed methods of the `HXSampler` class.

C.2.2. Utilized methods from blackbox

These are the methods that we rely on to have exposed by the library that exposes the hardware handling to us. We give their current names here, but stress that these may change as it is pre-release software state.

1. `set_readout` Takes the neuron ID as an integer and does not return anything.

   It configures the routing such that the correct neuron is connected to the MADC, in the next call from stimulate this neuron's membrane potential will be read out.

2. `set_weights` Takes the weight matrix and does not return anything.

   It takes the logical weight matrix, i.e. the matrix of 128x128 signed 6-bit integers representing the connectivity matrix and configures the synapse array appropriately for the next call of stimulate.

3. `set_bias` Takes the bias vector and does not return anything.

   It takes the logical bias vector, i.e. the vector of 128 unsigned 10-bit integers that represent the leak potential configurations of the network and configures the leak potentials for the next call of stimulate.

4. `stimulate` Takes a duration and optionally input spike trains, a boolean record and a pre and post hook function, returns tuple (spikes, samples).

   The duration is a float representing the total experiment duration in seconds of hardware time.

   Optional, input spikes can be a list of tuples of (time, ID) of spikes to be injected by the FPGA, must be sorted.

   Optional, boolean record. Defaults to false. Whether to read out the MADC samples, see also *set_readout*

   Optional, post and pre hook functions. Used to set up the on-chip noise generators.

C.2.3. Methods of HXSampler

1. `setup_onchip_noise` Takes optionally an integer seed, an integer noisemultiplier and a matrix noise_matrix.

   Configures the on-chip noise generators by setting up appropriate *pre_hook* and *post_hook* functions. We generate 64 logical sources in total and use two PRNG.

   Optional, seed is an 32-bit integer that determines the internal state of the 32-bit on-chip PRNG. It is unclear how the remaining 12-bit affect the system.

   Optional, noisemultiplier. If noise_matrix is not explicitly provided populate the connectivity matrix with noisemultiplier-many ones per logical neuron. I.e., every network neuron receives input from noisemultiplier many sources. Be aware that a higher noisemultiplier leads to stronger correlations, both auto- and cross-.

   Optional, noise_matrix. If provided supersedes the random matrix generated from noisemultiplier. Must be of shape (64, number_of_neurons).

2. `random_noise_matrix` Takes optionally noisemultiplier.

   Used in *setup_onchip_noise*. Randomly generates noisemultiplier many ids to be set to one in a (32, number_of_neurons) zero-matrix.

3. `update_parameters` Writes the parameters held in self.weights and self.logical_-weight and self.logical_bias to the chip.

4. `get_noise_spikes` Takes a duration, returns a list of tuples (time, ID).

   Generates the noise spike trains, if noisetype "Poisson" is used.

5. `measure_activation_function` Takes optionally int stepsize, float duration and tuple bias_range, returns nothing.

   Measures the activation function by sweeping the leak potential configuration.

   Optional, stepsize. Default 200, sets the increment of the leak potential value between two measurement points.

   Optional, duration. Default 1×10^{-3} s, sets the duration per measurement point.

   Optional, bias_range. Default $[200, 1000]$, sets the range of the parameters that are being tested.

   Writes the measured activations and fit into the self.activation dictionary. Keys "bias", "spikes", "fit", values are all lists. Writes minimal observed ISI to self.measured_taurefs.

6. `measure_weight_activation` Takes sourceneuron, targetneuron and optionally int stepsize, float duration and float dt, returns nothing.

   Similar to *measure_activation_function*, but measures activation as a function of connection strength to a permanently firing neuron.

   sourceneuron is the ID of the leak-over-threshold neuron that is used as the source of the connection.

   targetneuron is the ID or list of IDs of the neurons whose activation function is being measured.

   dt, default 1×10^{-7} s is the time step on which the states are being evaluated.

   Writes measured data to self.weight_activation dictionary. Keys weight, prob_-source, prob_target, fit.

7. `get_samples` Main user facing function. Takes float duration and optionally float dt, tauref, clamped_states, clamp_type, statedt, number_of_spikes, readout_-neuron, set_parameters and returns a list of binary states.

   duration is the time in hardware seconds that the experiment is running.

   Optional dt, default 5×10^{-6} s. Time step where the network state is evaluated and added to the states list.

   Optional tauref, list of the refractory times to be used for the state evaluation. If not explicitly provided it tries to use the measured refractory times, see *measure_-activation_function*.

   Optional clamped_states, clamp_type, statedt, number_of_spikes. Abandoned interface for the implementation of clamping for the data phase of wake sleep

learning. The intention was to provide a series of states and a time per state *statedt* and the rest is done by the software. Currently not usable.

Optional readout_neuron. If provided, read out this neuron's membrane trace, will be written to self.voltages by run_network.

8. `run_network` Takes float duration and optionally list input_spikes, bool return_inputs, integer readout_neuron, bool plot_raster, bool save_parameters, bool set_parameters and returns the list of observed spike tuples (time, ID).

   Intended to be called by *get_samples* and *measure_activation_function*.

   duration is the time in hardware seconds that the experiment is running.

   Optional input_spikes. List of (time, ID) of externally provided spikes. Must be sorted

   Optional return_inputs, default false. Whether to return the provided input spikes or not.

   Optional integer readout_neuron. If provided, call *set_readout* and write the returned data to self.voltages

   Optional plot_raster, default false. Whether to dump a raster plot of all observed spikes to plots/raster_<time>.pdf

   Optional save_parameters, default false. Whether to dump the current parameters to plots/biases_<time>.npy and plots/weights_<time>.npy

   Optional set_parameters, default false. Whether to update the parameter configuration on hardware. Useful for speeding up repeated experiments.

C.2.4. Auxiliary functions in hxsampling.py

1. `sigmoid` Takes x, x0, alpha, rmax. Returns $r_{\max}\sigma\left(\frac{x-x_0}{\alpha}\right)$.

2. `poissonspiketimes` Takes float frequency, float duration and returns a list of Poisson-distributed spike times.

3. `get_noise_spikes` Takes float duration, int number_of_neurons, float noiserate, optionally int minoutid and returns a list of spike time tuples (time, ID)

   duration is the time in hardware seconds that the experiment is running.

   number_of_neurons is the number of spike trains to be generated.

   noiserate is the rate of the Poisson process of the generated spike train, see also *poissonspiketimes*.

   Optional minoutid is the offset ID to be added to the generated spike trains. Must be large enough not to interfere with the network neurons and small enough that the largest generated ID is below 128.

   Largely deprecated as on-chip sources are typically used.

4. `get_clamping_spikes` Never really implemented see also *get_samples*

5. `get_states_from_spikes` Takes int number_of_neurons, spikes, taurefs, float dt, optional float duration and returns list of states.

   number_of_neurons number of binary units to be assigned.

   spikes list of spike tuples (time, ID) of the experiment.

   taurefs list of the refractory times of the neurons. Gives the time frame for which the unit is to be considered in state $z = 1$ after a spike.

   dt time in hardware seconds after which a new state is produced. Must be smaller than the refractory time due to a performance improvement. For arbitrary large dt the function must be rewritten.

   Optional duration, default 0 s. If non-zero used as the explicit end of the experiment. Otherwise the end is inferred from the last spike time.

6. `get_isis` Takes list of spikes, int number_of_neurons and returns list of lists of ISIs.

   spikes list of spike tuples of the experiment

   number_of_neurons total number of neurons for which the lists of ISIs should be generated.

C.3. Neuralsampling

This section aims to give a high-level description of the neuralsampler simulator used throughout Section 4.3. It implements the augmented version of the Buesing neuron model described in Section 2.2.3 and was developed by the author.

Neuron

The center of it is the neuron object. There are two levels of parameterization. On the one hand there is the neuron type, which means the form of its interaction as well as its transfer function, and on the other hand there are the dynamical parameters, in particular, the time constants. On the other hand the object implements functions to query the internal and external state, as well as generating spikes (updating the state) and returning the current state of the interaction. The latter is implemented on the neuron level, rather than the network level, in order to limit the amount of storage required.

Implemented as interaction types are:

1. Rect

   Rectangular interaction of length τ_{syn} after a spike

2. Exp

   Exponentially decaying interaction with time constant τ_{syn}, reset after a spike and normalized such that the sum of the interaction equals the rectangular case

3. Cuto

   Like Exp only that the interaction is set to 0 for time delays larger than τ_{syn}.

4. Tail

   Within the refractory period the interaction shape is given by Rect, afterwards by Exp.

Implemented as transfer functions are:

1. Log

   $z = 1$ happens with probability $\sigma(b_i \sum_i W_{ki} \kappa(\zeta_i) - \log \tau)$, corresponding to the original activation function derived in [Buesing et al., 2011].

2. Erf

   $z = 1$ happens with probability $\text{erf}(c\, u)$, where $c = 0.416\,311\,18$ is chosen to minimize the L2 difference between $\sigma(u)$ and $\text{erf}(c\, u)$.

3. Step

   $z = 1$ is chosen by comparison with a threshold value. Experimental feature, with explicit provisioning of time-correlated noise to implement an Ornstein-Uhlenbeck (OU) noise process.

The dynamical parameters of the neuron object are

1. tauref

   $\tau_{\text{ref}} - 1$ gives the number of time steps after a spike (or equivalent the minimal internal state ζ) in which the spike mechanism is artificially suppressed to implement the refractoriness. The offset by 1 is required to make a continuous $z = 1$ configuration possible.

2. tausyn

   τ_{syn} gives the timescale of the interaction shape as discussed above.

3. delay

   The delay d gives the size of a ring buffer that caches the interaction strength. Updating the neuron writes the new interaction shape into the slot $n_t + 1\%d$. Whenever the neuron is interogated regarding its interaction strength the entry at slot position n_t is returned. This implements a delay of $d - 1$ time steps until the update is visible to other neurons in the network. Note: This currently implements a global delay per neuron rather than a connection specific one.

Additional state variables of the neuron object are

1. nspikes

   counts the total number of spikes the neuron has had over the simulation. Allows for efficient evaluation of mean activities.

2. state

   internal state ζ of the neuron. It represents the time since the last spike and informs the interaction strength calculated within update_neuron.

3. membrane_potential

   the current value of the membrane potential of the neuron. Required for the experimental autocorrelated input.

In addition there are cached values for $\frac{\tau_{\text{ref}}}{\tau_{\text{syn}}}$ and $\exp \frac{\tau_{\text{ref}}}{\tau_{\text{syn}}}$. These values are repeatedly calculated throughout the determination of the current interaction strength, but are constant. It makes a noticeable difference to manually cache these results as the parametrization is constant throughout a simulation. This is, however, only a runtime constant and not a compile-time one, which prevents the compiler from making this optimization automatically.

Network

The publicly visible object is the network. It takes a bias vector as a std::vector<double>, a connectivity matrix std::vector<std::vector<double> > and an initial state vector std::vector<int64> together with a configuration object holding the information for the neuron parameters. For a sufficiently sparse connection matrix the network automatically stores a list of connected ids with associated weight rather than the full matrix (sparse matrix representation). The config object also contains information regarding the update and the output schemes

As update schemes are implemented

1. InOrder

   update the neurons in order according to their id

2. Random

   randomly select one of the neurons to update in the next step

3. BatchRandom

   randomly select the next neuron to update, but ensure that in n update steps all neurons are updated exactly once.

It turns out that the choice of update scheme does not influence the behavior significantly as long as there are n neuron updates in one network update.

For the choice of output there are the following implementations available

1. SummarySpikes

   Only output the number of spikes per neuron at the end of the simulation

2. SummaryStates

   Only output the number of times each network state was hit. This will break for larger number of neurons n as the summary_states std::vector<int> will have 2^n many entries.

3. MeanActivityOutput

   Prints the sum of the (external) state over all neurons $\langle z \rangle = \sum_i z_i$ for each network update

4. MeanActivityEnergyOutput

   Prints both the mean activity $\langle z \rangle = \sum_i z_i$ and the energy of the associated state $E = \sum_i z_i b_i + \sum_{ij} z_i z_j W_{ij}$ for each network update

5. BinaryState

   Prints the binary state for each network update

6. InternalStateOutput

   Prints a space-separated list of the internal states ζ_i for each network update

7. SpikesOutput

   Prints the ids of all neurons that emitted a spike (and then have $\zeta_i = 0$) for each network update

Internally the network object iterates over all neurons (in a potentially randomly drawn order) and calculates the membrane potential u (get_potential_for_neuronid) in each update step. It exhibits the following methods

1. produce_header

   Prints a header for the output containing the type of neuron and the update and output scheme

2. produce_output

   Prints the current state of the network according to the selected output scheme (see above)

3. get_{binary_,internal}state

   Sets the std::vector<int> states member to the external z (internal ζ) state for readout

4. update_state

   Updates the network according to the update scheme. Takes optionally a temperature T and external field I_{ext} which rescale and shift the activation function respectively.

D. Publications and contributions

Peer-reviewed publications

Akos F. Kungl, Sebastian Schmitt, Johann Klähn, Paul Müller, **Andreas Baumbach**, Dominik Dold, Alexander Kugele, Eric Müller, Christoph Koke, Mitja Kleider, Christian Mauch, Oliver Breitwieser, Luziwei Leng, Nico Gürtler, Maurice Güttler, Dan Husmann, Kai Husmann, Andreas Hartel, Vitali Karasenko, Andreas Grübl, Johannes Schemmel, Karlheinz Meier and Mihai A. Petrovici, Accelerated Physical Emulation of Bayesian Inference in Spiking Neural Networks, *Frontiers in Neuroscience — Neuromorphic Engineering*, doi: 10.3389/fnins.2019.01201; 14 November 2019 Volume 13 pages 1201

Contribution: discussions on the study design and the evaluation

This study is discussed in Section 5.1.

Dominik Dold, Ilja Bytschok, Akos F. Kungl, **Andreas Baumbach**, Oliver Breitwieser, Walter Senn, Johannes Schemmel, Karlheinz Meier and Mihai A. Petrovici, Stochasticity from function Why the Bayesian brain may need no noise, *Neural Networks*, doi: 10.1016/j.neunet.2019.08.002; November 2019, Volume 119, Pages 200-213

Contribution: discussions on study design

This study is mentioned in Section 3.2.

Preprints

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, Versatile emulation of spiking neural networks on an accelerated neuromorphic substrate, *arXiv preprint*, 2019, arXiv:1912.12980

Contribution: implementing spike-based Bayesian inference on the BrainScaleS-2 neuromorphic system, contribution to the manuscript

Submitted and accepted to the proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS), but at the time of writing not yet published. The work here is a precursor of the work in Czischek et al. see below.

Julian Göltz, **Andreas Baumbach**, Sebastian Billaudelle, Oliver Breitwieser, Dominik Dold, Laura Kriener, Akos F. Kungl, Walter Senn, Johannes Schemmel, Karlheinz Meier, Mihai A. Petrovici, Fast and deep neuromorphic learning with time-to-first-spike coding, *arXiv preprint*, 2019, arXiv:1912.11443

Contribution: discussion on study design, contribution to the manuscript

This study is mentioned in Chapter 6 but uses a different compute paradigm – it encodes information in the time of the first spike of each neuron – and would not have fit well into the narrative of this dissertation. My involvement was similar to the work of Nico Gürtler in that I supervised Julian Göltz throughout his master thesis.

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, Spiking neuromophic chip learns entangled quantum states, *arXiv preprint*, 2020, arXiv:2008.01039

Contribution: study design, implementation of the neuromorphic framework, manuscript

This study is discussed in Section 5.2. My contributions focused on the utilization of BrainScaleS-2 providing Stefanie with the sampling framework (cf. Appendix C.2) on top of which we could then implement the POVM formulation of the quantum states.

Sections with contributions from other people

Section 3.1.1 The simulations were performed by me, but all ideas were already established by M. Petrovici and L. Leng [Petrovici, 2015, Leng et al., 2018]. While I was involved in some discussions during my early attendances of the TMA meeting (the meeting of the theory part of the Electronic Vision(s)), my contributions were mostly in critical questioning.

Section 3.3 The LMM was developed in the master thesis of N. Gürtler. He performed all the simulations and software development. Besides the contributions through the normal TMA meetings I was involved in the day-to-day discussions with Nico and in particular was his direct supervisor throughout his thesis.

Section 4.1 I took over the supervision of Agnes from Luziwei Leng when he graduated shortly after his handing in of his PhD thesis. From thereon Agnes and me collaborated in performing the simulations for [Korcsak-Gorzo et al., in prep.].

Section 5.1 See the discussion of the corresponding paper.

Section 5.2 See the discussion of the corresponding paper.

Acronyms

BM Boltzmann machine 30, 63, 104, 129, 155

COBA conductance-based 21, 22, 34, 36, 38

CUBA current-based 21, 22, 34, 36, 38, 92, 117

EPSP excitatory post-synaptic potential 17

FPGA field-programmable gate array 124

HCS high-conductance state 21, 38, 52, 84, 95, 117

HICANN High Input Count Analog Neural Network 122

IPSP inhibitory post-synaptic potential 17

LIF Leaky-integrate and fire 17, 18, 24, 28, 34–37, 41–43, 45–50, 52, 54–56, 58–84, 86–89, 91, 93–95, 97–99, 102, 104, 106, 110, 113, 116–120, 122–126, 128, 130, 132–134, 136, 138, 140, 142, 144, 146, 148, 150, 152, 153, 185

LMM LIF Markov model 64, 71–76, 78–84, 87, 110

LSFR linear-feedback shift register 60

ODE ordinary differential equation 20, 122

PRN pseudo random number 60

PSC post-synaptic current 16, 85

PSP post-synaptic potential 16, 19–22, 34–36, 42, 46–50, 52, 58, 62, 69, 72–80, 83, 85, 94, 95, 106, 110, 111, 117, 118, 126, 185

RBM restricted Boltzmann machine 25, 30, 129, 133, 147, 151

Bibliography

Syed Ahmed Aamir, Yannik Stradmann, Paul Müller, Christian Pehle, Andreas Hartel, Andreas Grübl, Johannes Schemmel, and Karlheinz Meier. 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, 2018.

Filipp Akopyan, Jun Sawada, Andrew Cassidy, Rodrigo Alvarez-Icaza, John Arthur, Paul Merolla, Nabil Imam, Yutaka Nakamura, Pallab Datta, Gi-Joon Nam, et al. 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, 2015.

Anders SG Andrae and Tomas Edler. On global electricity usage of communication technology: trends to 2030. *Challenges*, 6(1):117–157, 2015.

Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando GSL Brandao, David A Buell, et al. Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779): 505–510, 2019.

Coryn AL Bailer-Jones. *Practical Bayesian Inference*. Cambridge University Press, 2017.

Boris Barbour, Nicolas Brunel, Vincent Hakim, and Jean-Pierre Nadal. What can we learn from synaptic weight distributions? *TRENDS in Neurosciences*, 30(12):622–629, 2007.

Andreas Baumbach. Magnetic phenomena in spiking neural networks. *MSc thesis, Heidelberg University — Kirchhoff Institute for Physics*, 2016.

Bruce P Bean. The action potential in mammalian central neurons. *Nature Reviews Neuroscience*, 8(6):451–465, 2007.

John S Bell. *Speakable and unspeakable in quantum mechanics: Collected papers on quantum philosophy*. Cambridge University Press, 2004.

Hans Berger. Über das elektroenkephalogramm des menschen. *Archiv für Psychiatrie und Nervenkrankheiten*, 87(1):527–570, 1929.

Enrico Bibbona, Gianna Panfilo, and Patrizia Tavella. The ornstein–uhlenbeck process as a model of a low pass filtered white noise. *Metrologia*, 45(6):S117, 2008.

Sebastian Billaudelle. *No title yet.* PhD thesis, Ruprecht-Karls-Universität Heidelberg, in preparation.

Sebastian Billaudelle, Benjamin Cramer, Petrovici Mihai A, Korbinian Schreiber, David Kappel, Johannes Schemmel, and Karlheinz Meier. Structural plasticity on an accelerated analog neuromorphic hardware system. *arXiv preprint; arXiv:1912.12047*, 2019a.

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, and Karlheinz Meier. Versatile emulation of spiking neural networks on an accelerated neuromorphic substrate. *arXiv preprint; arXiv:1912.12980*, 2019b.

George EP Box. Science and statistics. *Journal of the American Statistical Association*, 71(356):791–799, 1976.

BrainScaleS. Brainscales - brain-inspired multiscale computation in neuromorphic hybrid systems, 2011. URL `http://brainscales.kip.uni-heidelberg.de/`. Accessed: 2019-07-30.

Oliver Breitwieser, Andreas Baumbach, Agnes Korcsak-Gorzo, Johann Klähn, Max Brixner, and Mihai Petrovici. sbs: Spike-based sampling (v1.8.2), February 2020. URL `https://doi.org/10.5281/zenodo.3686015`. This open source software code was developed in part in the Human Brain Project, funded from the European Union's Horizon 2020 Framework Programme for Research and Innovation under the Specific Grant Agreement No. 720270 (HBP SGA1) and 785907 (HBP SGA2).

Romain Brette and Wulfram Gerstner. Adaptive exponential integrate-and-fire model as an effective description of neuronal activity. *Journal of Neurophysiology*, 94(5): 3637–3642, 2005.

Daniel Brüderle, Mihai A Petrovici, Bernhard Vogginger, Matthias Ehrlich, Thomas Pfeil, Sebastian Millner, Andreas Grübl, Karsten Wendt, Eric Müller, Marc-Olivier Schwartz, et al. A comprehensive workflow for general-purpose neural modeling with highly configurable neuromorphic hardware systems. *Biological cybernetics*, 104(4-5): 263–296, 2011.

Nicolas Brunel. Dynamics of sparsely connected networks of excitatory and inhibitory spiking neurons. *Journal of Computational Neuroscience*, 8(3):183–208, 2000.

Nicolas Brunel and Mark CW Van Rossum. Lapicque's 1907 paper: from frogs to integrate-and-fire. *Biological cybernetics*, 97(5-6):337–339, 2007.

Lars Buesing, Johannes Bill, Bernhard Nessler, and Wolfgang Maass. Neural dynamics as sampling: A model for stochastic computation in recurrent networks of spiking neurons. *PLoS Comput Biol*, 7(11):1–22, 11 2011. doi: 10.1371/journal.pcbi.1002211. URL `http://dx.doi.org/10.1371%2Fjournal.pcbi.1002211`.

Gyorgy Buzsaki. *Rhythms of the Brain*. Oxford University Press, 2006.

Ilja Bytschok. *Computing with noise in spiking neural networks*. PhD thesis, Heidelberg University — Kirchhoff Institute for Physics, 2017.

Ilja Bytschok, Dominik Dold, Johannes Schemmel, Karlheinz Meier, and Mihai A Petrovici. Spike-based probabilistic inference with correlated noise. In *BMC Neuroscience 2017*, volume 18, page P200. Organization for Computational Neurosciences, 2017.

Adán Cabello, Álvaro Feito, and Antia Lamas-Linares. Bell's inequalities with realistic noise for polarization-entangled photons. *Physical Review A*, 72(5):052112, 2005.

Haixiao Cai, Sanjeev R Kulkarni, and Sergio Verdú. Universal divergence estimation for finite-alphabet sources. *IEEE Transactions on Information Theory*, 52(8):3456–3475, 2006.

Juan Carrasquilla, Giacomo Torlai, Roger G Melko, and Leandro Aolita. Reconstructing quantum states with generative models. *Nature Machine Intelligence*, 1(3):155–161, 2019.

Geoffrey E Carreira-Perpinan, Miguel A and Hinton. On contrastive divergence learning. *Aistats*, 10:33–40, 2005.

Dante R Chialvo. Emergent complex neural dynamics. *Nature physics*, 6(10):744–750, 2010.

Chris73. Sketch of an action potential, 2007. URL `FileURL:https://upload.wikimedia.org/wikipedia/commons/4/4a/Action_potential.svg`. accessed 2020-01-31, Licence: CC BY-SA at https://creativecommons.org/licenses/by-sa/3.0/, Original by en:User:Chris 73, updated by en:User:Diberri, converted to SVG by tiZom.

Michael S Clayton, Nick Yeung, and Roi Cohen Kadosh. The roles of cortical oscillations in sustained attention. *Trends in cognitive sciences*, 19(4):188–195, 2015.

Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Advances in neural information processing systems*, pages 3123–3131, 2015.

Benjamin Cramer, Sebastian Billaudelle, Simeon Kanya, Aron Leibfried, Andreas Grübl, Vitali Karasenko, Christian Pehle, Korbinian Schreiber, Yannik Stradmann, Johannes Weis, et al. Training spiking multi-layer networks with surrogate gradients on an analog neuromorphic substrate. *arXiv preprint arXiv:2006.07239*, 2020a.

Benjamin Cramer, David Stöckel, Markus Kreft, Michael Wibral, Johannes Schemmel, Karlheinz Meier, and Viola Priesemann. Control of criticality and computation in spiking neuromorphic networks with plasticity. *Nature Communications*, 11(1):1–11, 2020b.

Stefanie Czischek. *Neural-Network Simulation of Strongly Correlated Quantum Systems*. Springer Theses. Springer International Publishing, 2020. ISBN 9783030527143. URL `https://books.google.de/books?id=Gk6TzQEACAAJ`.

Stefanie Czischek, Andreas Baumbach, Sebastian Billaudelle, Benjamin Cramer, Lukas Kades, Jan M. Pawlowski, Markus K. Oberthaler, Johannes Schemmel, Mihai A. Petrovici, Thomas Gasenzer, and Martin Gärttner. Spiking neuromorphic chip learns entangled quantum states, 2020.

Henry Hallett Dale. *Adventures in Physiology: with excursions into autopharmacology*. Pergamon Press, 1953.

E D'Angelo, P Mazzarello, F Prestori, Jonathan Mapelli, S Solinas, P Lombardo, E Cesana, D Gandolfi, and L Congi. The cerebellar network: from structure to function and dynamics. *Brain research reviews*, 66(1-2):5–15, 2011.

Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, Yongqiang Cao, Sri Harsha Choday, Georgios Dimou, Prasad Joshi, Nabil Imam, Shweta Jain, et al. Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro*, 38 (1):82–99, 2018.

Andrew Davison, Daniel Brüderle, Jens Kremkow, Eilif Muller, Dejan Pecevski, Laurent Perrinet, and Pierre Yger. Pynn: a common interface for neuronal network simulators. 2009.

Peter Dayan and Laurence F Abbott. *Theoretical neuroscience: computational and mathematical modeling of neural systems*. MIT press, 2001.

Alain Destexhe. Self-sustained asynchronous irregular states and up–down states in thalamic, cortical and thalamocortical networks of nonlinear integrate-and-fire neurons. *Journal of computational neuroscience*, 27(3):493, 2009.

Alain Destexhe, Michael Rudolph, and Denis Paré. The high-conductance state of neocortical neurons in vivo. *Nature reviews neuroscience*, 4(9):739, 2003.

Susanne Diekelmann and Jan Born. The memory function of sleep. *Nature Reviews Neuroscience*, 11(2):114–126, 2010.

Dominik Dold. *Harnessing function from form: towards bio-inspired artificial intelligence in neuronal substrates*. PhD thesis, Kirchhoff Institute for Physics, Heidelberg University, 2020. at the time of writing, this thesis has been submitted but not yet published.

Dominik Dold, Ilja Bytschok, Akos F Kungl, Andreas Baumbach, Oliver Breitwieser, Walter Senn, Johannes Schemmel, Karlheinz Meier, and Mihai A Petrovici. Stochasticity from functionwhy the bayesian brain may need no noise. *Neural Networks*, 119: 200–213, 2019.

Kenji Doya, Shin Ishii, Alexandre Pouget, and Rajesh PN Rao. *Bayesian brain: Probabilistic approaches to neural coding*. MIT press, 2007.

Daniel Drubach. *The brain explained*. Prentice Hall, 2000.

Toshikazu Ebisuzaki, Junichiro Makino, Toshiyuki Fukushige, Makoto Taiji, Daiichiro Sugimoto, Tomoyoshi Ito, and Sachiko K Okumura. Grape project: an overview. *Publications of the Astronomical Society of Japan*, 45:269–278, 1993.

Gaute T Einevoll, Alain Destexhe, Markus Diesmann, Sonja Grün, Viktor Jirsa, Marc de Kamps, Michele Migliore, Torbjørn V Ness, Hans E Plesser, and Felix Schürmann. The scientific case for brain simulations. *Neuron*, 102(4):735–744, 2019.

Andreas K Engel and Wolf Singer. Temporal binding and the neural correlates of sensory awareness. *Trends in cognitive sciences*, 5(1):16–25, 2001.

Simon Friedmann, Johannes Schemmel, Andreas Grübl, Andreas Hartel, Matthias Hock, and Karlheinz Meier. Demonstrating hybrid learning in a flexible neuromorphic hardware system. *IEEE transactions on biomedical circuits and systems*, 11(1):128–142, 2017.

Steve B Furber, Francesco Galluppi, Steve Temple, and Luis A Plana. The spinnaker project. *Proceedings of the IEEE*, 102(5):652–665, 2014.

Stuart Geman and Donald Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on pattern analysis and machine intelligence*, 6:721–741, 1984.

Samuel Gershman, Ed Vul, and Joshua B Tenenbaum. Perceptual multistability as markov chain monte carlo inference. In *Advances in neural information processing systems*, pages 611–619, 2009.

Wulfram Gerstner and Werner M Kistler. Mathematical formulations of hebbian learning. *Biological Cybernetics*, 87(5-6):404–415, 2002a.

Wulfram Gerstner and Werner M Kistler. *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge university press, 2002b.

Timo Gierlich. Tbd. Master thesis, Universität Heidelberg, December 2020.

Roy J Glauber. Time-dependent statistics of the ising model. *Journal of mathematical physics*, 4(2):294–307, 1963.

Julian Göltz, Andreas Baumbach, Sebastian Billaudelle, Oliver Breitwieser, Dominik Dold, Laura Kriener, Akos F. Kungl, Walter Senn, Johannes Schemmel, Karlheinz Meier, and Mihai A Petrovici. Fast and deep neuromorphic learning with time-to-first-spike coding. *arXiv preprint; arXiv:1912.11443*, 2019.

Daniel M Greenberger, Michael A Horne, and Anton Zeilinger. Going beyond bell's theorem. In *Bell's theorem, quantum theory and conceptions of the universe*, pages 69–72. Springer, 1989.

Andreas Grübl and Andreas Baumbach. F09/f10 neuromorphic computing. *University of Heidelberg*, 2017.

Nico Gürtler. A markovian model of lif networks. Masterarbeit, Universität Heidelberg, 2018.

Stefan Habenschuss, Johannes Bill, and Bernhard Nessler. Homeostatic plasticity in bayesian spiking networks as expectation maximization with posterior constraints. In *Advances in Neural Information Processing Systems*, pages 773–781, 2012.

Cong Han and Bradley P Carlin. Markov chain monte carlo methods for computing bayes factors: A comparative review. *Journal of the American Statistical Association*, 96(455):1122–1132, 2001.

HBP-SP9. The hbp neuromorphic computing platform. https://electronicvisions.github.io/hbp-sp9-guidebook/, 2020. Accessed: 2020-08-21.

Geoffrey E Hinton. A practical guide to training restricted boltzmann machines. In *Neural networks: Tricks of the trade*, pages 599–619. Springer, 2012.

Geoffrey E Hinton, Peter Dayan, Brendan J Frey, and Radford M Neal. The" wake-sleep" algorithm for unsupervised neural networks. *Science*, 268(5214):1158–1161, 1995.

Matthias Hock, Andreas Hartel, Johannes Schemmel, and Karlheinz Meier. An analog dynamic memory array for neuromorphic hardware. In *2013 European Conference on Circuit Theory and Design (ECCTD)*, pages 1–4. IEEE, 2013.

Alan L Hodgkin and Andrew F Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of physiology*, 117(4):500–544, 1952.

E. Ising. Beitrag zur Theorie des Ferromagnetismus. *Zeitschrift fur Physik*, 31:253–258, February 1925. doi: 10.1007/BF02980577.

Eugene M Izhikevich. *Dynamical systems in neuroscience*. MIT press, 2007.

Sebastian Jeltsch. *A scalable workflow for a configurable neuromorphic platform*. PhD thesis, Ruprecht-Karls-Universität Heidelberg, 2014.

Jakob Jordan, Tammo Ippen, Moritz Helias, Itaru Kitayama, Mitsuhisa Sato, Jun Igarashi, Markus Diesmann, and Susanne Kunkel. Extremely Scalable Spiking Neuronal Network Simulation Code: From Laptops to Exascale Computers. *Frontiers in Neuroinformatics*, 12:2, 2 2018. ISSN 1662-5196. doi: 10.3389/fninf.2018.00002.

Jakob Jordan, Mihai A Petrovici, Oliver Breitwieser, Johannes Schemmel, Karlheinz Meier, Markus Diesmann, and Tom Tetzlaff. Deterministic networks for probabilistic computing. *Scientific reports*, 9(1):1–17, 2019.

Norman P Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, et al. In-datacenter performance analysis of a tensor processing unit. In *Proceedings of the 44th Annual International Symposium on Computer Architecture*, pages 1–12, 2017.

R Jung and W Berger. Fiftieth anniversary of hans berger's publication of the electroencephalogram. his first records in 1924–1931 (author's transl). *Archiv fur Psychiatrie und Nervenkrankheiten*, 227(4):279, 1979.

Eric R Kandel, James H Schwartz, Thomas M Jessell, Department of Biochemistry, Molecular Biophysics Thomas Jessell, Steven Siegelbaum, and AJ Hudspeth. *Principles of neural science*, volume 4. McGraw-hill New York, 2000.

Vitali Karasenko. *Von Neumann bottlenecks in non-von Neumann computing architectures*. PhD thesis, Ruprecht-Karls-Universität Heidelberg, 2020.

Hassan N Khan, David A Hounshell, and Erica RH Fuchs. Science and research policy at the end of moore's law. *Nature Electronics*, 1(1):14–21, 2018.

Mitja Kleider. *Neuron Circuit Characterization in a Neuromorphic System*. PhD thesis, Ruprecht-Karls-Universität Heidelberg, 2017.

Christoph Koke. *Device variability in synapses of neuromorphic circuits*. PhD thesis, Ruprecht-Karls-Universität Heidelberg, 2017.

Alexander Kononov. *Testing of an analog neuromorphic network chip*. PhD thesis, Diploma thesis, Ruprecht-Karls-Universität Heidelberg, 2011. HD-KIP-11-83, 2011.

A. Korcsak-Gorzo, L. Leng, A. Baumbach, J. Breitwieser, S. van Albada, W. Senn, K. Meier, and M. A. Petrovici. Spike-based tempering in neural networks. in prep.

Agnes Korcsak-Gorzo. Simulated tempering in spiking neural networks. *Masterarbeit. Universität Heidelberg*, 2017.

Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

Arvind Kumar, Sven Schrader, Ad Aertsen, and Stefan Rotter. The high-conductance state of cortical networks. *Neural computation*, 20(1):1–43, 2008.

Akos Kungl. Sampling with leaky integrate-and-fire neurons on the hicannv4 neuromorphic chip. *Masterarbeit, Universität Heidelberg*, 2016.

Ákos Ferenc Kungl. *Robust learning algorithms for spiking and rate-based neural networks*. PhD thesis, Ruprecht-Karls-Universität Heidelberg, 2020.

Akos Ferenc Kungl, Sebastian Schmitt, Johann Klähn, Paul Müller, Andreas Baumbach, Dominik Dold, Alexander Kugele, Eric Müller, Christoph Koke, Mitja Kleider, et al. Accelerated physical emulation of bayesian inference in spiking neural networks. *Frontiers in Neuroscience*, 13:1201, 2019.

Tor Sverre Lande, Hassan Ranjbar, Mohammed Ismail, and Yngvar Berg. An analog floating-gate memory in a standard digital technology. In *Proceedings of fifth international conference on microelectronics for neural networks*, pages 271–276. IEEE, 1996.

Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition, 1998.

Luziwei Leng, Roman Martel, Oliver Breitwieser, Ilja Bytschok, Walter Senn, Johannes Schemmel, Karlheinz Meier, and Mihai A Petrovici. Spiking neurons with short-term synaptic plasticity form superior generative networks. *Scientific reports*, 8(1):10651, 2018.

Henry Markram, Wulfram Gerstner, and Per Jesper Sjöström. A history of spike-timing-dependent plasticity. *Frontiers in synaptic neuroscience*, 3:4, 2011a.

Henry Markram, Karlheinz Meier, Thomas Lippert, Sten Grillner, Richard Frackowiak, Stanislas Dehaene, Alois Knoll, Haim Sompolinsky, Kris Verstreken, Javier DeFelipe, et al. Introducing the human brain project. *Procedia Computer Science*, 7:39–42, 2011b.

Henry Markram, Eilif Muller, Srikanth Ramaswamy, Michael W Reimann, Marwan Abdellah, Carlos Aguado Sanchez, Anastasia Ailamaki, Lidia Alonso-Nanclares, Nicolas Antille, Selim Arsever, et al. Reconstruction and simulation of neocortical microcircuitry. *Cell*, 163(2):456–492, 2015.

Clive Maxfield. An introduction to different rounding algorithms. *Programmable Logic Design Line*, pages 1–15, 2006.

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

Nicholas Metropolis and Stanislaw Ulam. The monte carlo method. *Journal of the American statistical association*, 44(247):335–341, 1949.

Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092, 1953.

Rupert G Miller Jr. *Survival analysis*, volume 66. John Wiley & Sons, 2011.

Sebastian Millner. *Development of a multi-compartment neuron model emulation*. PhD thesis, Heidelberg University — Kirchhoff Institute for Physics, 2012.

Gordon E Moore et al. Cramming more components onto integrated circuits, 1965.

Paul Müller. *Modeling and verification for a scalable neuromorphic substrate*. PhD thesis, Ruprecht-Karls-Universität Heidelberg, 2017.

Walter Nadler and Ulrich HE Hansmann. Generalized ensemble and tempering simulations: A unified view. *Physical Review E*, 75(2):026109, 2007.

Emre Neftci, Srinjoy Das, Bruno Pedroni, Kenneth Kreutz-Delgado, and Gert Cauwenberghs. Event-driven contrastive divergence for spiking neuromorphic systems. *Frontiers in neuroscience*, 7:272, 2014.

Michael A Nielsen and Isaac Chuang. Quantum computation and quantum information, 2002.

Wolfgang Nolting. *Quantentheorie des Magnetismus: Teil 2: Modelle*. Springer-Verlag, 2013.

Harry Nyquist. Certain topics in telegraph transmission theory. *Transactions of the American Institute of Electrical Engineers*, 47(2):617–644, 1928.

Marie Engelene J Obien, Kosmas Deligkaris, Torsten Bullmann, Douglas J Bakkum, and Urs Frey. Revealing neuronal function through microelectrode array recordings. *Frontiers in neuroscience*, 8:423, 2015.

Lars Onsager. Crystal statistics. i. a two-dimensional model with an order-disorder transition. *Phys. Rev.*, 65:117–149, Feb 1944. doi: 10.1103/PhysRev.65.117. URL `http://link.aps.org/doi/10.1103/PhysRev.65.117`.

Bente Pakkenberg, Dorte Pelvig, Lisbeth Marner, Mads J Bundgaard, Hans Jørgen G Gundersen, Jens R Nyengaard, and Lisbeth Regeur. Aging and the human neocortex. *Experimental gerontology*, 38(1-2):95–99, 2003.

Liam Paninski. Estimation of entropy and mutual information. *Neural computation*, 15 (6):1191–1253, 2003.

Edwin Pednault, John A Gunnels, Giacomo Nannicini, Lior Horesh, and Robert Wisnieff. Leveraging secondary storage to simulate deep 54-qubit sycamore circuits. *arXiv preprint arXiv:1910.09534*, 2019.

Mihai A. Petrovici. *Form vs. Function: Theory and Models for Neuronal Substrates*. PhD thesis, Heidelberg University — Kirchhoff Institute for Physics, 2015.

Mihai A. Petrovici, David Stöckel, Ilja Bytschok, Johannes Bill, Thomas Pfeil, Johannes Schemmel, and Karlheinz Meier. Fast sampling with neuromorphic hardware. In *Advances in Neural Information Processing Systems (NIPS)*, volume 28, 2015.

Mihai A Petrovici, Johannes Bill, Ilja Bytschok, Johannes Schemmel, and Karlheinz Meier. Stochastic inference with spiking neurons in the high-conductance state. *Physical Review E*, 94(4):042312, 2016.

Mihai A Petrovici, Sebastian Schmitt, Johann Klähn, David Stöckel, Anna Schroeder, Guillaume Bellec, Johannes Bill, Oliver Breitwieser, Ilja Bytschok, Andreas Grübl, et al. Pattern representation and recognition with accelerated analog neuromorphic systems. In *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–4. IEEE, 2017a.

Mihai A Petrovici, Anna Schroeder, Oliver Breitwieser, Andreas Grübl, Johannes Schemmel, and Karlheinz Meier. Robustness from structure: Inference with hierarchical spiking networks on analog neuromorphic hardware. In *Neural Networks (IJCNN), 2017 International Joint Conference on*, pages 2209–2216. IEEE, 2017b.

Alexander Peyser, Ankur Sinha, Stine Brekke Vennemo, Tammo Ippen, Jakob Jordan, Steffen Graber, Abigail Morrison, Guido Trensch, Tanguy Fardet, Håkon Mørk, Jan Hahne, Jannis Schuecker, Maximilian Schmidt, Susanne Kunkel, David Dahmen, Jochen Martin Eppler, Sandra Diaz, Dennis Terhorst, Rajalekshmi Deepu, Philipp Weidel, Itaru Kitayama, Sepehr Mahmoudian, David Kappel, Martin Schulze, Shailesh Appukuttan, Till Schumann, Hünkar Can Tunç, Jessica Mitchell, Michael Hoff, Eric Müller, Milena Menezes Carvalho, Barna Zajzon, and Hans Ekkehard Plesser. NEST 2.14.0. *Zenodo*, 10 2017. doi: 10.5281/ZENODO.882971.

Thomas Pfeil, Andreas Grübl, Sebastian Jeltsch, Eric Müller, Paul Müller, Mihai A Petrovici, Michael Schmuker, Daniel Brüderle, Johannes Schemmel, and Karlheinz Meier. Six networks on a universal neuromorphic computing substrate. *Frontiers in neuroscience*, 7:11, 2013.

Thomas Pfeil, Jakob Jordan, Tom Tetzlaff, Andreas Grübl, Johannes Schemmel, Markus Diesmann, and Karlheinz Meier. Effect of heterogeneity on decorrelation mechanisms in spiking neural networks: A neuromorphic-hardware study. *Physical Review X*, 6 (2):021023, 2016.

Andrea Pinotti. *Der Schnabel des Adlers und die Nase des Menschen. Können wir uns in einen Vogel einfühlen?*, volume 81. Walter de Gruyter GmbH & Co KG, 2020.

Panayiota Poirazi and Athanasia Papoutsi. Illuminating dendritic function with computational models. *Nature Reviews Neuroscience*, pages 1–19, 2020.

Dimitri Probst, Mihai A Petrovici, Ilja Bytschok, Johannes Bill, Dejan Pecevski, Johannes Schemmel, and Karlheinz Meier. Probabilistic inference in discrete spaces can be implemented into networks of lif neurons. *Frontiers in computational neuroscience*, 9:13, 2015.

Hubert Ramsauer, Bernhard Schäfl, Johannes Lehner, Philipp Seidl, Michael Widrich, Lukas Gruber, Markus Holzleitner, Milena Pavlović, Geir Kjetil Sandve, Victor Greiff, et al. Hopfield networks is all you need. *arXiv preprint arXiv:2008.02217*, 2020.

J Rigden. Body, physics of, 1996.

Pieter R Roelfsema and Arjen van Ooyen. Attention-gated reinforcement learning of internal representations for classification. *Neural computation*, 17(10):2176–2214, 2005.

Johannes Schemmel. Private communication, 2020-09-24, 2020.

Johannes Schemmel, Johannes Fieres, and Karlheinz Meier. Wafer-scale integration of analog neural networks. In *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, pages 431–438. IEEE, 2008.

Johannes Schemmel, Daniel Brüderle, Andreas Grübl, Matthias Hock, Karlheinz Meier, and Sebastian Millner. A wafer-scale neuromorphic hardware system for large-scale neural modeling. In *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, pages 1947–1950. IEEE, 2010.

Johannes Schemmel, Sebastian Billaudelle, Phillip Dauer, and Johannes Weis. Accelerated analog neuromorphic computing. *arXiv preprint; arXiv:2003.11996*, 2020.

Sebastian Schmitt, Johann Klähn, Guillaume Bellec, Andreas Grübl, Maurice Guettler, Andreas Hartel, Stephan Hartmann, Dan Husmann, Kai Husmann, Sebastian Jeltsch, et al. 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)*, pages 2227–2234. IEEE, 2017.

Peter W Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th annual symposium on foundations of computer science*, pages 124–134. Ieee, 1994.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.

Paul Smolensky. Information processing in dynamical systems: Foundations of harmony theory. Technical report, Colorado Univ at Boulder Dept of Computer Science, 1986.

Thomas Splettstoesser. Sketch of a chemical synapse, 2015. URL `FileURL: https://upload.wikimedia.org/wikipedia/commons/7/70/SynapseSchematic_`

`unlabeled.svg`. accessed 2020-01-31, Licence: CC BY-SA (https://creativecommons.org/licenses/by-sa/4.0).

Olaf Sporns, Giulio Tononi, and Rolf Kötter. The human connectome: a structural description of the human brain. *PLoS Comput Biol*, 1(4):e42, 2005.

David Stöckel. Boltzmann sampling with neuromorphic hardware. Bachelorarbeit, Universität Heidelberg, January 2015.

Greg Stuart, Nelson Spruston, and Michael Häusser. *Dendrites*. Oxford University Press, 2016.

Greg J Stuart and Nelson Spruston. Dendritic integration: 60 years of progress. *Nature neuroscience*, 18(12):1713–1721, 2015.

D Sulzer and S Rayport. Dale's principle and glutamate corelease from ventral midbrain dopamine neurons. *Amino acids*, 19(1):45–52, 2000.

Rashmi Sundareswara and Paul R Schrater. Perceptual multistability predicted by search model for bayesian decisions. *Journal of vision*, 8(5):12–12, 2008.

Ilya Sutskever and Geoffrey Hinton. Learning multilevel distributed representations for high-dimensional sequences. In *Artificial intelligence and statistics*, pages 548–555, 2007.

Gelo Noel M Tabia. Experimental scheme for qubit and qutrit symmetric informationally complete positive operator-valued measurements using multiport devices. *Physical Review A*, 86(6):062107, 2012.

Amirhossein Tavanaei, Masoud Ghodrati, Saeed Reza Kheradpisheh, Timothee Masquelier, and Anthony Maida. Deep learning in spiking neural networks. *Neural Networks*, 111:47–63, 2019.

Graham W Taylor and Geoffrey E Hinton. Factored conditional restricted boltzmann machines for modeling motion style. In *Proceedings of the 26th annual international conference on machine learning*, pages 1025–1032. ACM, 2009.

Chetan Singh Thakur Thakur, Jamal Molin, Gert Cauwenberghs, Giacomo Indiveri, Kundan Kumar, Ning Qiao, Johannes Schemmel, Runchun Mark Wang, Elisabetta Chicca, Jennifer Olson Hasler, et al. Large-scale neuromorphic spiking array processors: A quest to mimic the brain. *Frontiers in neuroscience*, 12:891, 2018.

Binh Tran. Demonstrationsexperimente auf neuromorpher Hardware. *Bachelor thesis (german), Universität Heidelberg*, 2013.

Misha V Tsodyks and Henry Markram. The neural code between neocortical pyramidal neurons depends on neurotransmitter release probability. *Proceedings of the National Academy of Sciences*, 94(2):719–723, 1997.

George E Uhlenbeck and Leonard S Ornstein. On the theory of the brownian motion. *Physical review*, 36(5):823, 1930.

Sacha J van Albada, Andrew G Rowley, Johanna Senk, Michael Hopkins, Maximilian Schmidt, Alan B Stokes, David R Lester, Markus Diesmann, and Steve B Furber. 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, 2018.

Francisco Varela, Jean-Philippe Lachaux, Eugenio Rodriguez, and Jacques Martinerie. The brainweb: phase synchronization and large-scale integration. *Nature reviews neuroscience*, 2(4):229–239, 2001.

Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, pages 1–5, 2019.

Christopher S von Bartheld, Jami Bahney, and Suzana Herculano-Houzel. The search for true numbers of neurons and glial cells in the human brain: A review of 150 years of cell counting. *Journal of Comparative Neurology*, 524(18):3865–3895, 2016.

D von Suchodoletz, B Wiebelt, K Meier, and M Janczyk. Flexible hpc: bwforcluster nemo. *Proceedings of the 3rd bwHPCSymposium: Heidelberg*, 2016.

Ulrich Welsch and Thomas Deller. *Lehrbuch Histologie: Unter Mitarbeit von Thomas Deller*. Elsevier Health Sciences, 2016.

Shimon Whiteson and Peter Stone. Evolutionary function approximation for reinforcement learning. *Journal of Machine Learning Research*, 7(May):877–917, 2006.

Commons Wikimedia. File:duck-rabbit illusion.jpg — wikimedia commons, the free media repository, 2016a. URL `https://commons.wikimedia.org/w/index.php?title=File:Duck-Rabbit_illusion.jpg&oldid=206363593`. [Online; accessed 22-July-2020].

Commons Wikimedia. File:lfsr fibonacci 8 bits.png — wikimedia commons, the free media repository, 2016b. URL `https://commons.wikimedia.org/w/index.php?title=File:LFSR_Fibonacci_8_bits.png&oldid=216904514`. [Online; accessed 5-August-2020].

Commons Wikimedia. File:boltzmannexamplev1.png — wikimedia commons, the free media repository, 2020. URL `https://commons.wikimedia.org/w/index.php?title=File:Boltzmannexamplev1.png`. [Online; accessed 4-August-2020].

Wikipedia. Ornstein–uhlenbeck process — wikipedia, the free encyclopedia, 2016. URL `https://en.wikipedia.org/w/index.php?title=Ornstein%E2%80%93Uhlenbeck_process&oldid=719546703`. [Online; accessed 3-July-2016].

Wikipedia, contributors. Neuron — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Neuron&oldid=960428815`, 2020. [Online; accessed 8-June-2020].

Timo Wunderlich, Akos Ferenc Kungl, Eric Müller, Andreas Hartel, Yannik Stradmann, Syed Ahmed Aamir, Andreas Grübl, Arthur Heimbrecht, Korbinian Schreiber, David Stöckel, et al. Demonstrating advantages of neuromorphic computation: a pilot study. *Frontiers in Neuroscience*, 13:260, 2019.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

Geoffrey Yeap, SS Lin, YM Chen, HL Shang, PW Wang, HC Lin, YC Peng, JY Sheu, M Wang, X Chen, et al. 5nm cmos production technology platform featuring full-fledged euv, and high mobility channel finfets with densest 0.021 μm 2 sram cells for mobile soc and high performance computing applications. In *2019 IEEE International Electron Devices Meeting (IEDM)*, pages 36–7. IEEE, 2019.

List of Figures

List of Tables

Statement of Originality (Erklärung):

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

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

Heidelberg, September 24, 2020

.....
(signature)