Dissertation

submitted to the
Combined Faculties for the Natural Sciences and for Mathematics
of the Ruperto-Carola University of Heidelberg, Germany
for the degree of
Doctor of Natural Sciences

Presented by
Dipl.-Phys. Ilja Bytschok
born in Tomsk, Russian Federation

Date of oral examination: January 25, 2017

Computing with noise in spiking neural networks

Referees: Prof. Dr. Karlheinz Meier
Prof. Dr. Daniel Durstewitz

Computing with noise in spiking neural networks

Trial-to-trial variability is an ubiquitous characteristic in neural firing patterns and is often regarded as a side-effect of intrinsic noise. Increasing evidence indicates that this variability is a signature of network computation. The computational role of noise is not yet clear and existing frameworks use abstract models for stochastic computation. In this work, we use networks of spiking neurons to perform stochastic inference by sampling. We provide a novel analytical description of the neural response function with an unprecedented range of validity. This description enables an implementation of spiking networks in simulations to sample from Boltzmann distributions. We show the robustness of these networks to parameter variations and highlight the substantial advantages of short-term plasticity in our framework. We demonstrate accelerated inference on neuromorphic hardware with a speed-up of 10^4 compared to biological networks, regardless of network size. We further explore the role of noise as a computational component in our sampling networks and identify the functional equivalence between synaptic connections and mutually shared noise. Based on this, we implement interconnected sampling ensembles which exploit their activity as noise resource to maintain a stochastic firing regime.

Rauschen als Prinzip der Informationsverarbeitung mit feuernden Neuronen

Neuronale Aktivität zeigt im Allgemeinen eine Trial-to-Trial Variabilität. Diese Variabilität wird vermutlich durch intrinsisches Rauschen verursacht und es deutet viel darauf hin, dass Trial-to-Trial Variabilität ein wichtiges Merkmal von Informationsverarbeitung in Netzwerken ist. Jedoch ist die Funktion des Rauschens bei der Informationsverabritung bislang nicht geklärt, weshalb in bereits existierenden Modellen abstrakte Beschreibungen stochastischer Informationsverarbeitung benutzt werden. In dieser Arbeit benutzen wir Netzwerke mit feuernden Neuronen zur stochastischen Informationsverarbeitung mittels Sampling. Dabei beschreiben wir die neuronale Aktivität für Parameterbereiche, für die es bislang keine Beschreibung gab, analytisch. Das erlaubt uns, mittels feuernden Neuronen in Simulationen aus Boltzmannverteilungen Stichproben zu ziehen. Die Robustheit solcher Netzwerke gegenüber Parametervariationen zeigen wir durch umfassende Simulationen. Weiterhin erläutern wir die substanziellen Vorteile unserer Netzwerke gegenüber konventionellen Implementierungen. Auf neuromorpher Hardware demonstrieren wir beschleunigtes Sampling, welches hier, unabhängig von der Netzwerkgröße, 10^4 mal schneller abläuft als in biologischen Systemen. Des Weiteren untersuchen wir die Rolle des Rauschens in der Informationsverarbeitung und finden eine Äquivalenzbeziehung zwischen synaptischen Gewichten und geteilten Rauschquellen. Auf dieser Beziehung basierend, verbinden wir Neuronenensembles, die ihre Aktivität als Rauschquelle nutzen und somit stochastisch feuern.

Contents

1 Introduction

Suppose you are on a game show, facing three doors. Behind one door there is the grand prize, a car. Behind each of the two remaining doors, there is a goat. The game show host assigns you to pick a door. You pick door number one. Then, the host proceeds to open one of the other doors, door number three. The door number three reveals a goat. Then, eventually, you are given a choice. "Do you want to pick door number two or stick with door number one?"
So, should you stick with door number one or switch to door number two?

Confronted with this thought experiment, the famous Monty Hall problem, many firmly refuse to believe the correct answer. This is an example of how counter-intuitive probabilistic reasoning can be. Although a mathematical description is easy to provide, it shows that the brain has problems with dealing with many situations related to probability and uncertainty. Almost ironically, recent neuroscientific research studies suggest that the human mind constructs and constantly reshapes an inherently probabilistic model of the external world, based on statistical inference of noisy and uncertain sensory data (*Fiser et al.*, 2010; *Brascamp et al.*, 2006). Studies on cognitive processes like sensorimotor learning indicate that the brain continuously integrates new knowledge into this model (*Körding and Wolpert*, 2004; *Kersten et al.*, 2004; *Pouget et al.*, 2013). It is proposed that this integration is carried out in stochastic processes, based on spontaneous, irregular cortical activity. This irregularity is reflected in the presence of stochasticity in in-vivo activity, where trial-to-trial variability can be clearly observed (*Rolls and Deco*, 2010). And yet, despite this irregularity in neural activity patterns, mammals are able to integrate sensory stimuli over a short time scale with very high precision. Survival in hostile environments hinges entirely on the ability to locate, interpret and act upon perceived stimuli on short time scales. Therefore, stochasticity may not only exist as a mere side effect of the physical substrate, but could be an indispensable part of neural information processing (*Yang and Shadlen*, 2007).

Although it is not clear in neuroscience how such probabilistic computations are carried out in neural tissue, the usefulness of stochasticity as a computational tool has already been recognized and is adopted in the field of machine learning. In this field, computational models are trained to fulfill a certain computational task, such as classification of data according to a learned data set. To be able to perform such inference, the model needs to inherit an abstract representation of the data characteristics by learning. In general, the more complex the data set, the more difficult it is to find a suitable representation. Amongst the most efficient network models are those whose functionality is inspired by biological neural networks. Networks such as Boltzmann machines (Sec-

tion 5.1) are apt at finding good representations of high-dimensional data sets (*LeCun et al.*, 2015). Upon defining a measure of inference quality, the applied training algorithms adjust the network topology to obtain satisfactory results. By readjusting the network topology, the network explores the network configuration space to find such a solution. Interestingly, searching for the optimal network configuration using stochastic network changes has yielded very good models. The resulting generative model is of probabilistic nature, as it reproduces the target data in a reliable, but stochastic manner.

However, when applying such neural networks to increasingly complex and diverse data sets, the network sizes need to increase as well (*Krizhevsky and Hinton*, 2009). This causes the operational power costs to grow, limiting the applicability of such networks. In a well-known recent event, the enormous computational costs have been showcased during a match of the game of Go between Google's *AlphaGo* computer and one of the world's best players, Lee Sedol. The running hardware infrastructure during the series consisted of 1920 CPUs and 240 GPUs to implement customized, pretrained convolutional nets (*Silver et al.*, 2016). The estimated energy consumption amounts to approximately 1 MW, compared to the consumption of 20 W for Lee Sedol's brain. Prior to the event, the computational model was trained for many months, improving its probabilistic model of Go by playing against itself with energy consumption on a similar order of magnitude as in competition.

This shows that the human brain is vastly more efficient than a machine which is customized to optimally perform upon the chosen task. There are two main causes for this difference of energy consumption levels. Firstly, the human brain utilizes different, yet unexplored computational methods. Secondly, the computations in the brain are performed on a substrate which allows for more efficient computation.
To increase the efficiency of neural networks, a promising approach consists in using a computing substrate that is more similar to the brain. Such an approach is the *emulation* of networks, which was popularized by Carver Mead decades ago (*Mead*, 1989, 1990). In an emulation of networks, important aspects of the network correlates are mimicked by the physical dynamics of the system. In case of so-called *neuromorphic hardware*, the electronic circuitry copies aspects of the brain to mimic the properties of the basic components of neural networks - neurons and synapses. This architecture makes it possible to access time scales that allow accelerated network runtimes compared to biological networks. Such neuromorphic implementations allow significantly faster inference (*Pfeil et al.*, 2013; *Schmuker et al.*, 2014; *Petrovici et al.*, 2015b).

In this thesis we will use a biologically realistic neuron model to build spiking networks that perform inference tasks. In Chapter 2, we will introduce the neuron model that will be used in simulations throughout this thesis. Inspired by biological mechanisms in the cortex, we will implement a columnar architecture and assess its performance in pattern completion tasks in Chapter 3. Due to the variety of dynamics in such biology-inspired architectures, the crucial high-activity states can be difficult to interpret. A statistical description of single-neuron dynamics would be helpful to understand network dynamics in inference tasks. Therefore, we will investigate single neuron dynamics in the biological

high-conductance state in Chapter 4. We will use this statistical characterization in Chapter 5 to build spike-based networks that perform stochastic inference by sampling from Boltzmann distributions. We will extensively evaluate the sampling performance of such networks for a wide range of neuron and synapse parameters. In Chapter 6, we will use these sampling networks to build spike-based, large-scale Boltzmann machines which perform inference on real-world data sets. Exploiting the inherent parallelism of neuromorphic architectures, we will demonstrate accelerated sampling from Boltzmann distributions on the neuromorphic Spikey chip (*Petrovici et al.*, 2015b; *Stöckel*, 2015). In Chapter 7, we will address the implementation of noise sources on a neuromorphic device. This leads to a more fundamental question of how to utilize stochasticity as a computational network component. We will show that common sources of background noise and synaptic connectivity in spiking networks can be used interchangeably for network computation. We will use this knowledge to interconnect sampling networks and remove external stochastic background. Finally, we will implement spiking networks which use their output as background noise, which is crucial for sampling. Thereby we show how spiking networks perform meaningful computation using network-inherent noise.

2 Key components of neural networks

In this chapter we will introduce the computational model which we will use throughout this work. We will use a biology-inspired model that is mathematically tractable, but still has functional components that are derived from biological mechanisms. We will characterize these components, namely spiking neurons and synapses of the so-called *leaky integrate-and-fire* model.

2.1 The leaky integrate-and-fire neuron

In biology, nerve cells can have many different shapes and therefore exhibit a variety of dynamics. The relationship between neuron characteristics and their computational function is still unclear to a large extent. Therefore, it is debatable how much of its morphology plays a computational role (*Lin et al.*, 2015; *Marder and Goaillard*, 2006). Additionally, a highly detailed neuron model is often intractable to simulate. In the following sections we will outline the *leaky integrate-and-fire* (LIF) dynamics and define the terminology to construct neural networks in the next chapter.

The LIF model incorporates a point-like (*single-compartment*) model, thereby disregarding the detailed morphology of biological neurons. This is a major simplification from biological neurons, as a lot of research is dedicated to functional aspects of neuronal spatial structures (*Ermentrout and Terman*, 2010; *Lindsay et al.*, 2005). The dynamics of a point-like LIF neuron is characterized by an ordinary differential equation (ODE). It describes the dynamics of the membrane potential u,

$$C_\mathrm{m} \cdot \frac{d}{dt} u(t) = -g_\mathrm{l} \cdot [u(t) - E_\mathrm{l}] + I_\mathrm{syn} + I_\mathrm{ext} \quad . \tag{2.1}$$

The characteristic constant E_l denotes the *leak potential* (also *resting potential*) and g_l constitutes the *leak conductance* g_l, which is the sum of conductances of all passive ion channels in the neuron membrane. This quantity thereby describes the total conductance of the membrane in absence of synaptic interactions. Interpreting the membrane as a capacitor with capacitance C_m and resistance $R = \frac{1}{g_\mathrm{l}} = R$, we can define the intrinsic membrane time constant as $\tau_\mathrm{m} = C_\mathrm{m} \cdot R = \frac{C_\mathrm{m}}{g_\mathrm{l}}$. This time constant determines the speed with which the membrane potential reacts to stimuli; the smaller τ_m, the faster the membrane potential u will respond to a stimulus. The parameter I_ext is a current resulting from external input. For now, we will return to a more detailed description of E_l, as it can be regarded as a starting point for the description of membrane dynamics. In the absence of stimulation, it represents the equilibrium membrane potential value.

Physiologically, the value of this quantity is the result of interactions between proteins in the cell membrane that regulate the flow of particular ion types. The membrane is mainly composed of a lipid bilayer which is impermeable for a majority of chemical compounds. For certain chemical compounds (e.g. K^+, Na^+, Cl^-, Ca^{2+}), ion-specific channels and pumps enable passive and active transport across the cell membrane. This results in differences in concentration of these ions in the interior and exterior of the cell. The resulting electric force at the membrane drives the ion flux and is opposed by a diffusion gradient. The opposing forces of both the diffusion and electric gradient establish an equilibrium at which certain ion concentrations and the electric membrane potential stabilize. For monovalent ions, the resulting equilibrium potential E_l can be calculated via the Goldmann-Hodgkin-Katz equation,

$$E_\mathrm{l} = \frac{R \cdot T}{F} \ln \left[\frac{\sum_i^n P_{\mathrm{X}_i^+} [\mathrm{X}_i^+]_\mathrm{out} + \sum_i^n P_{\mathrm{Y}_i^-} [\mathrm{Y}_i^-]_\mathrm{in}}{\sum_i^n P_{\mathrm{X}_i^+} [\mathrm{X}_i^+]_\mathrm{in} + \sum_i^n P_{\mathrm{Y}_i^-} [\mathrm{Y}_i^-]_\mathrm{out}} \right] \quad . \tag{2.2}$$

Here, P_ion denotes the permeability for the corresponding ion and $[\mathrm{ion}]_\mathrm{site}$ represents the ion concentration inside or outside of the cell. The constants R and F denote the ideal gas constant and Faraday's constant, respectively.

2.2 Synaptic activity in the leaky integrate-and-fire model

The synaptic current I_syn models the impact of synaptic interaction on the membrane potential and therefore plays a central role for communication in networks. We will now discuss briefly how it is generated in chemical synapses.

We can write down the synaptic current as

$$I_\mathrm{syn}(t) = \sum_i g_i^\mathrm{syn}(t) \cdot [E_\mathrm{i}^\mathrm{rev} - u(t)] \quad . \tag{2.3}$$

When an action potential arrives at a synapse, synapse-specific neurotransmitters are released into the synaptic cleft. These neurotransmitters can bind to receptors located in the postynaptic cell membrane, which are coupled to ion channels and thereby regulate the influx of specific ions into the postsynaptic cell. The conductances of each of these channels are summed up in $\sum_i g_i^\mathrm{syn}$ and mark the first term of the r.h.s. in Equation 2.3. As a result, the membrane potential is driven towards the *reversal potential* $E_\mathrm{i}^\mathrm{rev}$. This new equilibrium can be calculated similarly to the leak potential E_l in Equation 2.3 via the Nernst equation. For each of these channel-specific ion types, a reversal potential exists and gives a contribution to the synaptic current. As can be seen in Equation 2.3, this contribution depends on the conductance g_i^syn and the difference $(E_\mathrm{i}^\mathrm{rev} - u)$. This means that, the more neurotransmitters were released at site i, the more i-specific channels open. This increases the current I_syn, driving u towards $E_\mathrm{i}^\mathrm{rev}$. For one dominant

channel i, this would lead to saturation of the membrane potential at $E_{\mathrm{i}}^{\mathrm{rev}}$. In general, the synaptic current is controlled by many channels, shaped by a weighted mean of all $E_{\mathrm{i}}^{\mathrm{rev}}$, as can be seen in Equation 2.3.

Aside of u, the conductances g^{syn} are the dynamic variables which make up the LIF neuron dynamics by controlling the synaptic current. Hence, we need to describe the second differential equation of our LIF system - the temporal evolution of g^{syn},

$$\frac{d}{dt} g_i^{\mathrm{syn}}(t) = -\frac{g_i^{\mathrm{syn}}(t)}{\tau_{\mathrm{syn}}} + \sum_{i,s} w_i \cdot \delta(t - t_s^i) \quad . \tag{2.4}$$

In absence of stimuli, the differential equation above describes exponential decay of the synaptic conductance with the time constant τ_{syn}. An increase in conductance is initiated by neurotransmitter release (i.e. synaptic activity) to the synaptic cleft if an action potential is activated. In our model of LIF neurons we define the time of this action potential as t_s and a spike is modelled as a δ-function time event, $\delta(t - t_s^i)$. A sequence of action potentials is then defined as,

$$S = \sum_{i,s} \delta(t - t_s^i) \quad . \tag{2.5}$$

These points in time model the process of neurotransmitter release at site i. The spike time series in Equation 2.5 constitutes the series of time events at which synapses are activated with coupling strengths w_i. We will denote these strengths *synaptic weights*, or simply *weights*. In biology, the strength of synaptic connections can not be expressed as a number, but depends on many factors[1]. For our purposes, this abstract quantity characterizes the responsiveness of postsynaptic cells to incoming presynaptic signals. We call the conductance increase following a stimulation by neurotransmitters a *postsynaptic conductance* (PSC) (Figure 2.1). It triggers the ion flux at the cell membrane, inducing a synaptic current and changing the membrane potential, as illustrated in Figure 2.2. This change of membrane potential is called the *postsynaptic potential* (PSP). A positive change of the membrane potential is called *excitation*, a negative change is called *inhibition*.

Excitation and inhibition are typically caused by release of different neurotransmitters. For inhibition, one prominent neurotransmitter is GABA (γ-Aminobutyric acid), causing K^+ channels to open. Subsequently, the positive ions flow outside of the cell, hyperpolarizing the membrane potential towards $E_{\mathrm{K}^+}^{\mathrm{rev}}$. For excitation, a common transmitter is glutamate, regulating the gating of Na^+ ion channels. This influx of positively charged ions depolarizes the membrane potential, driving it towards $E_{\mathrm{Na}^+}^{\mathrm{rev}}$.

[1] The morphology of neurons and chemical interactions can be very complex and impact the signal transmission properties of nerve cells (*Qu et al.*, 2009).

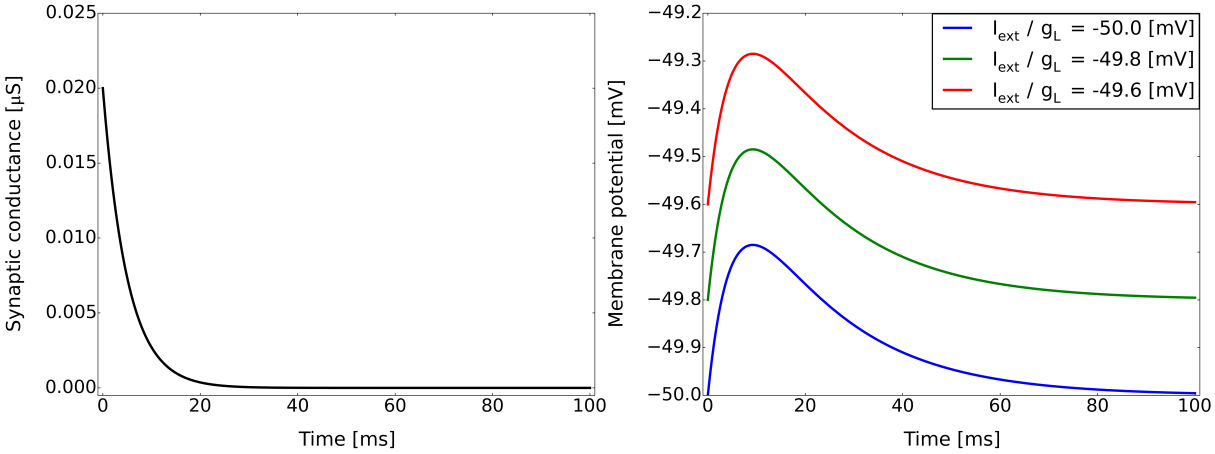

Figure 2.1: **(Left)** In LIF neurons, the synaptic conductance is a dynamic variable modelling the impact of presynaptic spikes on the postsynaptic neuron (Equation 2.4). Synaptic release is considered instantaneous at time $t = 0\,\text{ms}$ with the coupling parameter $w = 0.02\,\mu\text{S}$ as synaptic weight. The exponential decay time constant is set to $\tau_{\text{syn}} = 5\,\text{ms}$. **(Right)** For excitatory synapses, the generated PSP depolarizes the membrane potential (Equation 2.1). In this exmple, we demonstrate the offset of the PSP, given by a leak potential E_{l} and additional currents I_{ext}. The decay time scale of the PSP determined by the membrane time constant $\tau_{\text{m}} = 20\,\text{ms}$.

In our LIF model, an *action potential* is triggered if the membrane potential exceeds a certain *threshold potential* θ. The action potential is a characteristic property of neurons and is regarded as the primary means for signal transmission in neural circuitry. We will now describe the generation of these events and will refer to action potentials as *spikes*. The act of spike emission will be called *firing*.

An action potential is a sharp rise in membrane potential (i.e., depolarization), followed by hyperpolarization and a subsequent phase of inactivity. The reason for this chain of events is the voltage-dependence of specific K^+ and Na^+ ion channels, which will be explained in the following.

As the Na^+ influx increases the membrane potential, K^+-specific channels open at strongly depolarized potentials and lead to hyperpolarization by outflow of K^+ ions. Also, the Na^+ channels are closed and deactivated for a certain amount of time. Thus, even after no more K^+ ions flow, the deactivated Na^+-channels render further spiking impossible. The duration of this inactivity is called *refractory time* and is a common property of neurons. This characteristic inability of neurons to fire immediately after an action potential is called *refractoriness*.

Although the described biological background of these processes is fairly complex, we will model a simplified version of the action potential. In our LIF model, an action

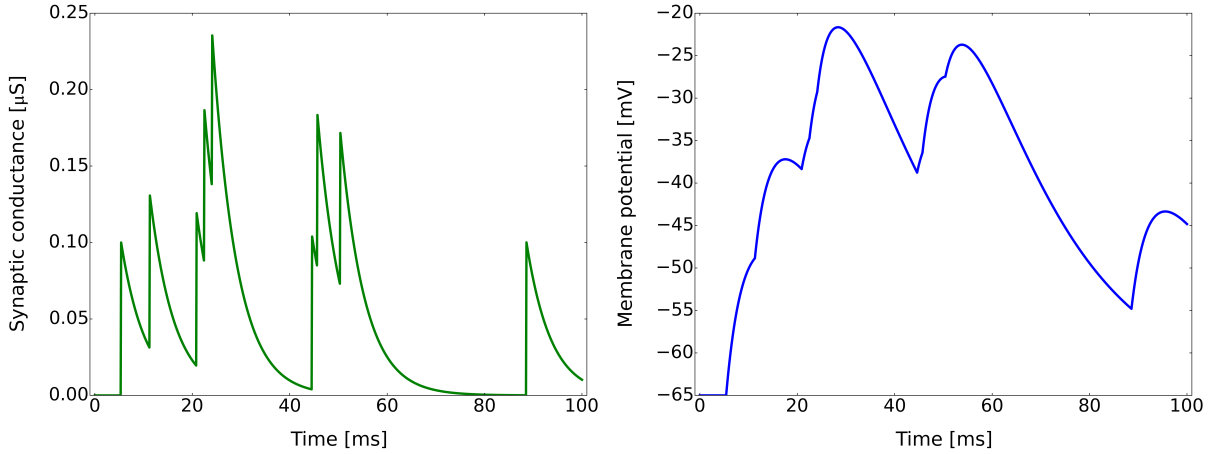

Figure 2.2: **(Left)** A leaky integrate-and-fire neuron is excited by multiple synaptic inputs, which are modeled as a superposition of exponential kernels with time constant τ_{syn}. **(Right)** The resulting ion flux generates multiple excitatory PSPs on the membrane potential.

potential is emitted as soon as excitation of the membrane exceeds the threshold θ. The model neuron will then instantly register a spike. Afterwards, u stays fixed at reset potential ρ for a refractory period τ_{ref}.

Although it is not entirely clear whether the shape of biological action potentials can be regarded as generic (*Villiere and McLachlan*, 1996; *Sasaki et al.*, 2011), in the LIF model every action potential has an identical time course.

At this point we conclude the description of the conductance-based[2] LIF model and introduce two important extensions to LIF dynamics.

2.3 Adaptive exponential leaky integrate-and-fire model

Our previously defined LIF neuron model has two dynamic variables, the membrane potential $u(t)$ and synaptic conductance $g^{syn}(t)$. Their evolution in time is described by two differential equations, Equation 2.1 and 2.4, respectively. The *Adaptive exponential leaky integrate-and-fire model* (AdEx) (*Brette and Gerstner*, 2005) is an extension of our basic LIF model. It adds another term to the differential equation of u. It also adds a new dynamic variable, representing adaptation. Thereby it is able to more closely reproduce recorded electrophysiological data. The two defining differential equations of the AdEx model are,

[2]The current-based LIF model is also widely-used and exhibits somewhat simpler dynamics, see *Bytschok* (2011) for a more detailed discussion.

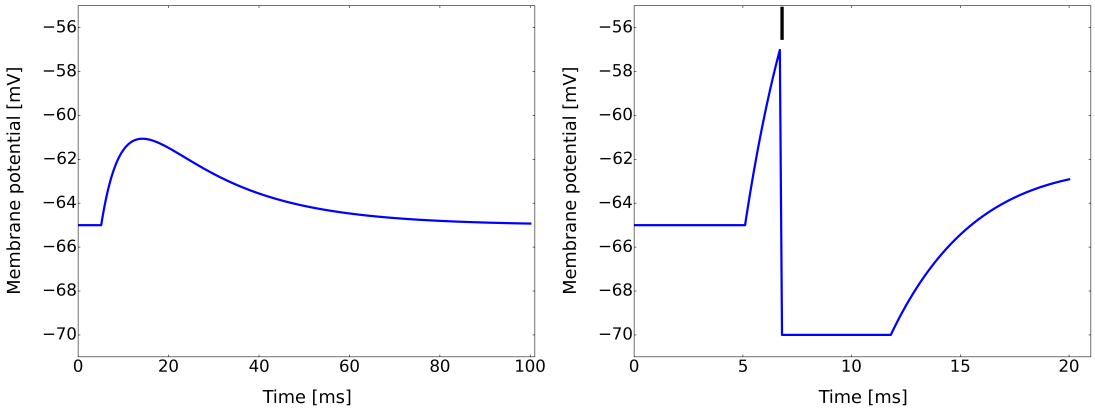

Figure 2.3: **(Left)** An excitatory PSP depolarizes the membrane potential towards the reversal potential, $E_{\text{exc}}^{\text{rev}}$. **(Right)** The PSP is large enough to lift the membrane potential above the threshold $\theta = -57\,\text{mV}$, triggering a spike (action potential). Immediately after the spike signal, a refractory period of 5 ms ensues, fixing the potential at $\rho = -70mV$. After $\tau_{\text{ref}} = 5\,\text{ms}$, the membrane potential recovers and overshoots the equilibrium potential at $E_{\text{l}} = -65\,\text{mV}$ due to excess excitation prior to the spike.

$$C_{\text{m}} \cdot \frac{d}{dt}u(t) = -g_{\text{l}} \cdot [u(t) - E_{\text{l}}] + g_{\text{l}}\Delta_T \exp\left(\frac{u - E_T}{\Delta_T}\right) - w + I_{\text{syn}} + I_{\text{ext}} \quad , \tag{2.6}$$

$$\tau_w \cdot \frac{d}{dt}w(t) = a \cdot [u(t) - E_{\text{l}}] - w + b\tau_w \sum_{t_{\text{spk}}} \delta(t - t_{\text{spk}}) \quad . \tag{2.7}$$

In Equation 2.6, there are two additional terms compared to Equation 2.1 of the LIF model.

The exponential term $g_{\text{l}}\Delta_T \exp\left(\frac{u-E_T}{\Delta_T}\right)$ implements the fast upsurge of membrane potential u when it approaches a predefined threshold value E_T. Note that E_T is not the hard spiking threshold θ we introduced in Chapter 2.1. Instead, E_T indicates the potential value at which the exponential term becomes dominant and pulls u further upwards until a spike is emitted at $u = \theta$. The slope of this exponential increase is governed by the constant Δ_T.

The second term in Equation 2.6 is the *adaptation current* w. This current changes the neuron's equilibrium potential and adapts the membrane potential to a given input. In particular, a strong excitatory input would trigger a high firing rate. The negative adaptation current $-w$ pulls down u, decreasing the firing rate by "adapting" to the strong stimulus (Figure 2.4). This effect is often observed in biology and is essential to reproduce a variety of firing sequences that would not be possible without adaptation (*Naud et al.*, 2008).

The second equation, 2.7, describes the temporal evolution of w. The *adaptation time constant* τ_w determines the time scale on which w decays towards zero. The parameter a determines the influence of leak potential E_l. The parameter b defines the increment added to the adaptation w after each spike in the spike train $\sum_{t_\mathrm{spk}} \delta(t - t_\mathrm{spk})$.

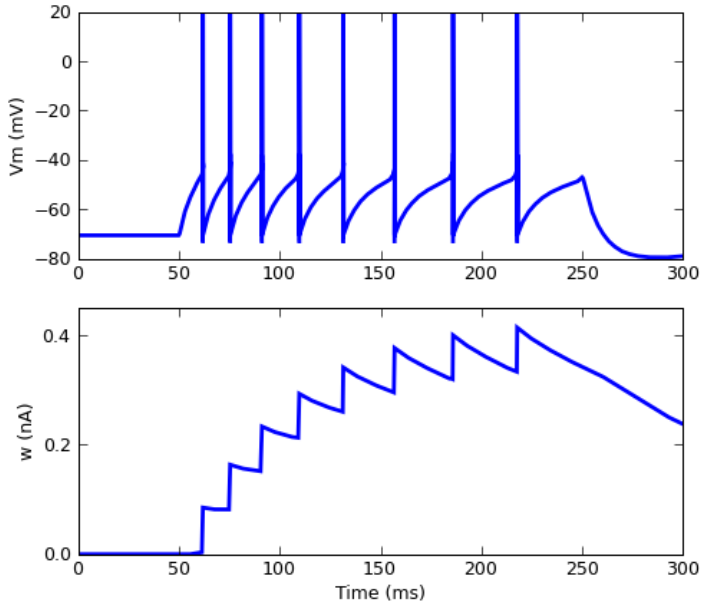

Figure 2.4: Membrane potential u **(top)** and adaptation current w **(bottom)** of an AdEx neuron, receiving a strong, constant current stimulus that causes the neuron to spike repeatedly. In contrast to the basic LIF model, the distance between consecutive action potentials descreases due to an increase of adaptation current w, as described in Equation 2.6 and 2.7. The adaptation current is displayed below, increasing stepwise for each emitted action potential. Eventually, the stimulus is turned off at about 250 ms and prevents the neuron from spiking further. Figure taken from *Gerstner et al.* (2014).

2.4 Synaptic short-term plasticity

Thus far, our LIF neuron model has been extended to an AdEx model. An exponential term was implemented to model subthreshold dynamics more realistically, and an adaptation current enables adaptation.

These extensions change the neuron's membrane characteristics but the synaptic properties of the LIF model remain unaffected. In fact, the modeled synapses so far

are completely static. This means that the value of the synaptic strength (i.e. synaptic weight w) remains constant. In contrast to this, biological synapses are *plastic*. There are many mechanisms which alter the synaptic properties on shorter time scales - in the range of milliseconds (*Stevens and Wang*, 1995; *Markram and Tsodyks*, 1996; *Abbott et al.*, 1997) and longer time scales - several hours and longer (*Bi and Poo*, 1998; *Song et al.*, 2000; *Song and Abbott*, 2001; *Abbott and Nelson*, 2000). These dynamics are essential for reshaping brain structure and therefore enabling learning processes.

Here, we do not discuss long-term learning, but only focus on short-term changes in our networks. The *Tsodyks-Markram mechanism* (TSO) describes the change of synaptic coupling depending on the number of utilized and available neurotransmitters in the synaptic cleft (*Tsodyks and Markram*, 1997; *Markram et al.*, 1998).

In this model, three fractions of neurotransmitter pools exist at the synapse: effective, recovered and inactive pools of neurotransmitters. The three fractions will be denoted E, R, I and constitute the entirety of neurotransmitter capacity in the biological system. Therefore, their sum is one:

$$E + R + I = 1 \tag{2.8}$$

Without any preceding synaptic interaction, there are no neurotransmitters in the cleft. Also, all neurotransmitters are actively available, therefore $R = 1$, $E = 0$, $I = 0$.

As soon as synaptic activation happens, a fraction of R is transferred to the effective partition E, from where it decays exponentially into the inactive partition I. Then it is recovered with a certain time constant, returning to R. These processes can be described mathematically with the following equations,

$$\frac{dR}{dt} = \frac{I}{\tau_{\text{rec}}} - \sum_{t_{\text{spk}}} U \cdot R \cdot \delta(t - t_{\text{spk}}) \quad , \tag{2.9}$$

$$\frac{dE}{dt} = -\frac{E}{\tau_{\text{inact}}} + \sum_{t_{\text{spk}}} U \cdot R \cdot \delta(t - t_{\text{spk}}) \quad . \tag{2.10}$$

Equation 2.9 models the evolution of the recovered pool R. For each spike time in $\sum_{t_{\text{spk}}} \delta(t - t_{\text{spk}})$, the fraction $U \cdot R$ is depleted for synaptic activation. The parameter U is also called *utilization* and determines utilized fraction of available resources at spike arrival times t_{spk}. For instance, for $U = 1$ all available neurotransmitters are released at every spike time t_{spk}. The term $\frac{I}{\tau_{\text{rec}}}$ models the gradually exponential recovery of R with time constant τ_{rec}. Equation 2.10 describes the time course of active neurotransmitter fraction, E. For every spike time t_{spk} it increases with $U \cdot R$ and declines exponentially with time constant τ_{inact} into the inactive pool I.

The above mechanism is based on in-vitro measurements and affects the synaptic conductance g^{syn}, since synaptic coupling strength depends on changes in available neurotransmitter concentration. The time constants of interaction strength, τ_{rec} and τ_{inact},

take on values $< 1\,\mathrm{s}$, indicating that this form of synaptic plasticity takes place on a relatively small time scale. These short-term changes in synaptic coupling are not permanent and persist only during the spike-triggered activity. During this period, the amplitude of the post-synaptic conductances g^{syn} is decreased, which is called *short-term depression* (STD). The TSO mechanism can also describe temporary facilitation (*Markram et al.*, 1998), which is called *short-term potentiation* (STP). To achieve this, the utilization U can also be modeled to increase by discrete amounts ΔU following each spike. The utilization is then a dynamic variable as well, modeled by

$$\frac{d}{dt}U(t) = \frac{U_0 - U}{\tau_{\mathrm{facil}}} + \sum_{t_{\mathrm{spk}}} \Delta U \delta(t - t_{\mathrm{spk}}) \quad , \tag{2.11}$$

$$\Delta U = U_0(1 - U) \quad . \tag{2.12}$$

Here we assume a resting value of utilization U_0, which changes at every spike by ΔU. The amount of utilization then recovers exponentially to U_0 with time constant τ_{facil}. Since U can become larger than U_0, it can have an amplifying effect on the post-synaptic conductances in g^{syn}. The time constant τ_{facil} determines the duration of the potentiation effect.

In the next chapters we will describe the impact of the TSO mechanism on the network functionality, as it will play an integral role in network dynamics.

2.5 Simulation framework: PyNN & NEST

All neural networks that we will discuss in this thesis are based on the differential equations that we have described in the previous sections. We will use a simulation framework to perform the numerical integration of these equations for large networks. For our simulations, we will use the *NEST* simulation kernel (*Diesmann and Gewaltig*, 2002). For network simulations in Chapters 5 and 6 we used the *spike-based sampling* (`sbs`) module developed by Oliver Breitwieser. This module is a modification of NEST and simplifies network setup and spike data evaluation for stochastic networks. We will use the *PyNN* package (*Davison et al.*, 2008) as an API (application programming interface) to generate and run our LIF-based neural networks. It provides the definition language for constructing neural networks.

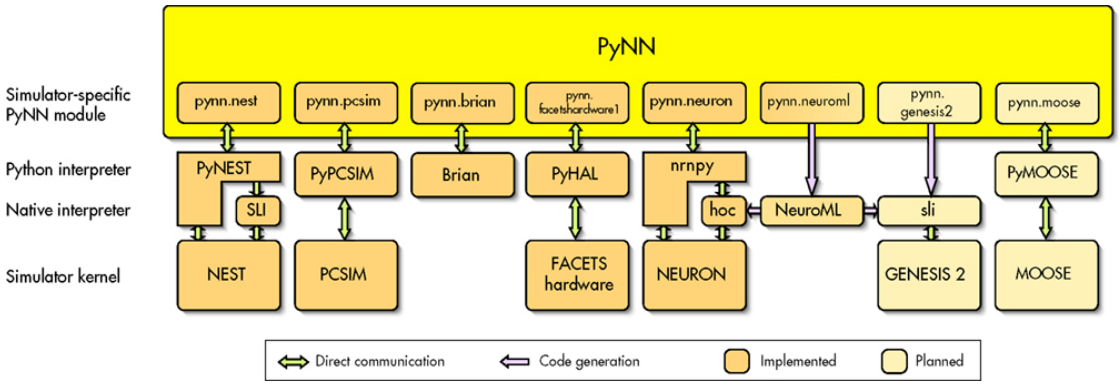

Figure 2.5: For simulations, we used the *PyNN* package, which embeds different simulation modules, based on different Python interpreters. The simulator kernel we used in our simulations was *NEST*. Figure is taken from *Davison et al.* (2008).

3 Cortical attractor networks

In the previous chapter, we have discussed the neuron model and simulation tools for our neural network simulations. In this chapter we will build an AdEx-based cortical network model and study its proficiency at performing inference tasks. The work presented in this chapter has been done in collaboration with Boris Rivkin, Mihai A. Petrovici and Oliver Breitwieser. Also, personal communication with Anders Lansner and Florian Fiebig from the Royal Institute of Technology was very helpful. All simulations in this chapter have been conducted by Boris Rivkin, who was supervised by the author of this thesis.

3.1 Cortical layer 2/3 networks with leaky integrate-and-fire neurons

The cerebral cortex is a key research object in neuroscience, as it constitutes one of the key architectures for information processing in the mammalian brain. The cortex is a $2\,\mathrm{mm} - 4\,\mathrm{mm}$ thick sheet of neural tissue, which is composed of six layers, each distinct in neuron morphology, topology and connectivity (*Kandel et al.*, 2000; *Shipp*, 2007) (Figure 3.1). It plays an important role in information processing tasks linked to perception, memory and attention mechanisms (*Kandel et al.*, 2000). We will focus on its role on working memory tasks that are processed by the prefrontal cortex, which is located in the frontal lobe. The working memory describes the capacity of mammals to store, retrieve and manipulate information in short-term. This capacity is an important attribute that has far-reaching effects on higher-order processes like reasoning and decision making upon available information (*Yang and Raine*, 2009). Well-known examples of working memory studies are experiments where human beings are presented sensory stimuli which they are assigned to recall in a certain time frame ("short-term memory"). In such experimental setups, sensory stimuli can be given by visual (e.g., colored shapes) or auditory (e.g., pitch) cues. The capacity can represent the number of different stimuli that the subject can recall during a certain amount of time. As a result of such studies, the activity in prefrontal and parietal neurons has been linked to working memory (*Fuster et al.*, 1971; *Gnadt and Andersen*, 1988). These studies suggest that working memory is an integral part of information processing in mammalian brains. Important cognitive processes rely on the mechanisms of information encoding and processing in the prefrontal lobe. Therefore, we will model the neural activity in the prefrontal cortex using the leaky integrate-and-firing model.

The neural activity in these regions has been researched in several studies. The investigated spike train patterns during working memory tasks suggest that populations

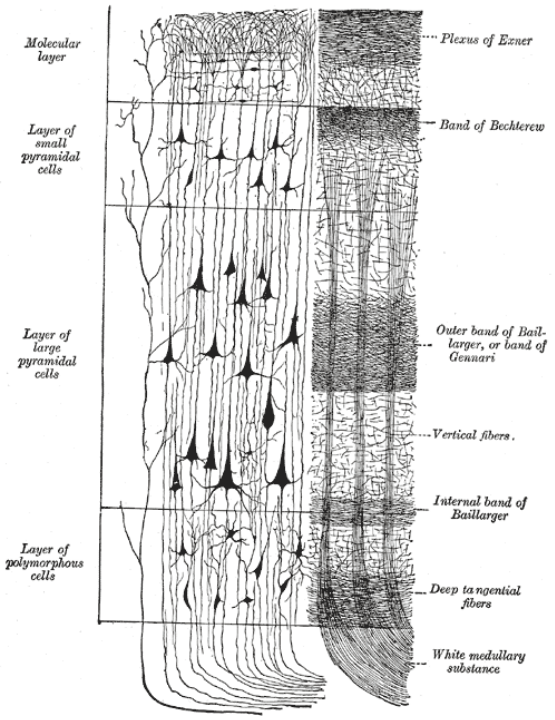

Figure 3.1: Illustration of neural tissue of the layered cortical sheet. The layers are characterized by different cell populations (left) and fibers (right). The morphology and density of neural tissue is layer-specific as well. Figure has been taken from *Gray* (1918).

in prefrontal neurons can exhibit both, irregular low-activity firing and regular high-frequency firing (*Pesaran et al.*, 2002; *Compte et al.*, 2003; *Shafi et al.*, 2007). During the tasks, there appears to be a coexistence of these two population firing modes. The low-frequency firing mode, which is referred to as background activity, is not selective to the presented stimuli, but is carried out by the majority of the populations. The persistent firing mode, on the other hand, is selective to stimuli and exhibits a considerably higher frequency (*Tsodyks and Sejnowski*, 1995; *van Vreeswijk and Sompolinsky*, 1996; *Roudi and Latham*, 2007; *Barbieri and Brunel*, 2007).

To investigate the computational properties of working memory in spiking networks, we will define a model that captures the functional aspects of the prefrontal cortex and reproduces the above described firing modes. Here, the first step lies in simplifying biological architecture to be able to model networks on scales that allow simulations in a reasonable time frame (i.e., several hours and up to two days).

We will use the layers 2 and 3 (L23) of the prefrontal cortex as the biological correlate for our neural network architecture. Additionally, we will model the signal transmission

from layer 4. To simplify the cortical architecture, we will assume a columnar structure of the cortex. This follows the so-called *minicolumn hypothesis*, stating that the cortical area can be subdivided into minicolumns, which can then be understood as functional modules (*Mountcastle*, 1979; *Hubel and Wiesel*, 1962, 1965). These modules can be regarded as clusters of neural tissue with high spatial density and intrinsic connectivity. Topologically, these clusters permeate all cortical layers and expand perpendicularly to the cortical sheet.

Although this structural interpretation of the cortex is debated (*Horton and Adams*, 2005; *Buxhoeveden and Casanova*, 2002a,b), it provides a framework that can be used for computational studies (*Markram et al.*, 2015). It offers the possibility to reduce the network complexity by subsuming neurons into minicolumns, which are then interpreted as the core computational components. This also applies to L23 networks, which have been implemented using Hodgkin-Huxley neurons (*Lundqvist et al.*, 2006, 2010) and AdEx neurons (*Petrovici et al.*, 2014).

Our task is to evaluate the L23 model as a columnar network model comprised of AdEx neurons and assess its usability for inference tasks. An overview of the network architecture can be seen in Figure 3.2.

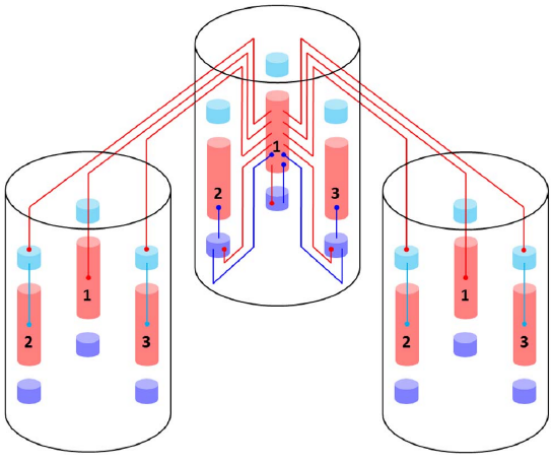

Figure 3.2: Three example hypercolumns, drawn as cylindrical shapes. In each of these hypercolumns three minicolumns are located. One minicolumn is considered as a basic computational node, consisting of multiple populations and encompassing the second and third cortical layers. Figure has been taken from *Breitwieser* (2011).

To study the potential of an AdEx-based working memory L23 network, we define a task which the network is set up to solve. As a benchmark task to evaluate this potential we will use a pattern completion setup where we store information in the network and provide cues to initiate recall activity. As stated before this inference task is related to human information processing and could potentially be used for classifying

data. Images that are incomplete or noisy, could be retrieved by the network. Similar approaches have already been investigated in *Breitwieser* (2011). The information will be stored as so-called *patterns*, which is a subset of minicolumns that share strong excitatory interconnections. Providing information that corresponds to a pattern means that a part of the minicolumn subset is activated by excitation. As soon as pattern-specific cues are provided, the network will tend to switch into the respective pattern state by activating the remaining pattern minicolumns. This firing state will exhibit regular high-frequency firing, which we will call an *attractor* state. Here, the provided cues are modelled as additional high-frequency excitation from the cortical layer 4 and applied to a predefined fraction of the pattern minicolumns. In case of absence of any cues, the full network will be set to exhibit irregular activity that corresponds to observed *background* states which are pattern-unspecific (*Amit and Brunel*, 1997a; *Durstewitz et al.*, 2000).

To reproduce this characteristic dynamics in the L23 network, we will implement an AdEx-based network topology that has been studied in *Breitwieser* (2011); *Petrovici et al.* (2014). Substantial parts of the following descriptions are based on the work of Boris Rivkin, which is documented in *Rivkin* (2014). The work published in *Rivkin* (2014) and was supervised by the author of this thesis.

3.2 Layer 2/3 network architecture

The L23 network consists of hypercolumns (HCs), each of which contains a predefined number of minicolumns (MCs, Figure 3.2). To describe the network topology more formally, we will use a matrix notation, where populations have indices i and j, denoting the HC and MC position, respectively. Each minicolumn includes three populations, which have different parameters and specific roles in the columnar architecture. The populations in a minicolumn are *pyramidal cells* (PYR), the *basket cells* (BAS) and the *regular spiking non-pyramidal* (RSNP) cells. The neuron parameters for the AdEx populations can be found in Appendix A.2. Before we explain how exactly a pattern is formed, we will describe the connectivity in the network. We denote unilateral connections as $\rightarrow$ and bilateral connections as $\leftrightarrow$. Additionally, we index inhibitory and excitatory connections with inh or exc, respectively. The connection probabilities between the populations will be denoted as P. We will now write down the connectivity and the respective connection probabilities for a hypercolumn i and minicolumn j,

$$\mathrm{PYR}_{ij} \overset{\mathrm{exc}}{\longleftrightarrow} \mathrm{PYR}_{ij} \text{ with } P_{\mathrm{PYR}\leftrightarrow\mathrm{PYR}} = 0.25 \tag{3.1}$$

$$\mathrm{RSNP}_{ij} \overset{\mathrm{inh}}{\longrightarrow} \mathrm{PYR}_{ij} \text{ with } P_{\mathrm{RSNP}\rightarrow\mathrm{PYR}} = 0.7 \tag{3.2}$$

$$\mathrm{BAS}_{ij} \overset{\mathrm{inh}}{\longrightarrow} \mathrm{PYR}_{j} \text{ with } P_{\mathrm{BAS}\rightarrow\mathrm{PYR}} = 0.7 \tag{3.3}$$

$$\mathrm{PYR}_{ij} \overset{\mathrm{exc}}{\longrightarrow} \mathrm{BAS}_{ij} \text{ with } P_{\mathrm{PYR}\rightarrow\mathrm{BAS}} = 0.7 \tag{3.4}$$

There are also synaptic connections that are not restricted to the same hypercolumn with index j,

$$\mathrm{PYR}_{ij} \overset{\mathrm{exc}}{\longrightarrow} \mathrm{BAS}_{|8i} \text{ with } P^{|8i}_{\mathrm{PYR}\rightarrow\mathrm{BAS}} = 0.7 \quad . \tag{3.5}$$

Here, the connectivity of the PYR_{ij} cells is restricted to the BAS cells of the eight minicolumns closest to minicolumn j.

For a minicolumn that is inside a pattern k, the pyramidal populations have additional excitatory connections with every other minicolumn from pattern k, located in different HCs,

$$\mathrm{PYR}_{kj} \overset{\mathrm{exc}}{\longleftrightarrow} \mathrm{PYR}_{k\setminus j} \text{ with } P^{\mathrm{pat}}_{\mathrm{MC}\leftrightarrow\mathrm{MC}} = 0.3 \quad . \tag{3.6}$$

The above relationship describes the excitatory connection that forms a pattern in the network. There are additional projections from PYR cells from a pattern k to RSNP cells from differing patterns ($\setminus k$) regardless of hypercolumn location j,

$$\mathrm{PYR}_{k} \overset{\mathrm{exc}}{\longrightarrow} \mathrm{RSNP}_{\setminus k} \text{ with } P^{\setminus k}_{\mathrm{PYR}\rightarrow\mathrm{RSNP}} = 0.17 \quad . \tag{3.7}$$

In our networks, each pattern consists of at least one MC and each MC that belongs to the same pattern is located in a different HC. On the other hand, an MC can be included in multiple patterns by having excitatory connections to the corresponding MCs. Patterns that do not share any MCs with other patterns are often referred to as *orthogonal*, whereas patterns that include MCs from other patterns, are described as *non-orthogonal* (Figure 3.3).

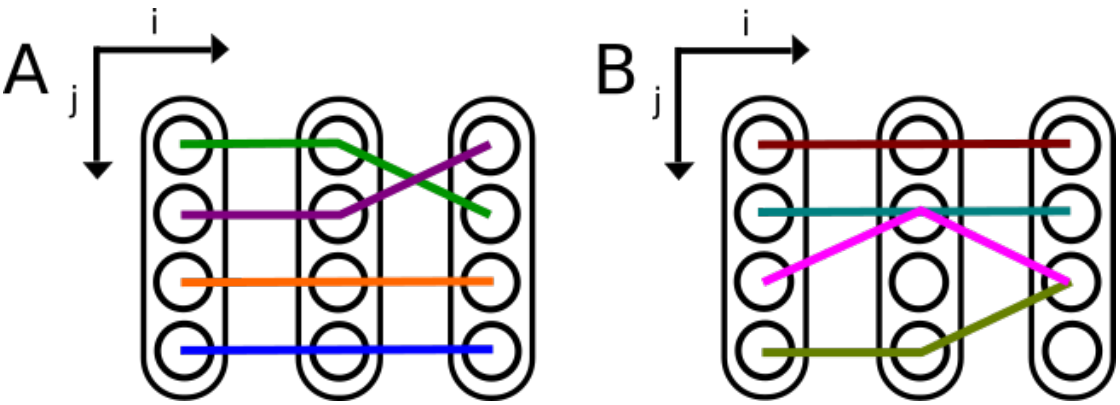

Figure 3.3: A cortical column network with example pattern topologies. **(A)** In a network consisting of three hypercolumns H and four minicolumns U per hypercolumn, there are four patterns which are marked by color. Since every MC encodes for at most one pattern, the network has only orthogonal patterns. **(B)** The layout is identical to the architecture in (A), illustrating non-orthogonal patterns where two minicolumns encode for multiple patterns. For both cases, each pattern includes at most one MC per HC.

The connections that form a pattern are given in Connection 3.1. For a pattern k, the cue in form of external excitatory input is presented to PYR cells in minicolumns that

are also part of the pattern k as excitatory input. Then, the excitation spreads to the remaining minicolumns of k and within the PYR populations via Connections 3.1. As a result of this excitation, attractor states emerge, resulting in high-frequency activity of the PYR populations in the pattern.

During the activation process of patterns, the activity of pyramidal cells can become very high. The BAS populations limit the PYR firing rates of its minicolumn by inhibiting PYR cells (Connection 3.3). This restrains the firing frequency of the PYR cells. Even more importantly, the BAS cells also inhibit all pyramidal cells that do not belong to their pattern (Connection 3.3). This mechanism enforces activity of the activated minicolumns, but also suppresses PYR activity from the remaining patterns. This mechanism is called *winner-take-all* (WTA) and is essential for the functionality of pattern activation in many network architectures (*Oster et al.*, 2009).

In the network states where cues are absent, PYR activity exhibits low-frequency irregular firing which we have referred to as background activity. This irregularity of the PYR cells results from additional presynaptic input that we have not discussed so far. This input is not generated within the L23 network, but constitutes external diffusive noise which induces the background activity that is also observed in biology (*Amit and Brunel*, 1997a). The noise is modeled as spike sequences whose time events are Poisson-distributed ("Poisson spike train"). Hence, their distribution in time is uniform and each spike is uncorrelated to any other spike in the sequence. This implementation of background activity is based on the L23 networks described in *Breitwieser* (2011); *Petrovici et al.* (2014). Despite weak synaptic connections of this presynaptic input, the Poisson spike trains ensure a certain PYR ground level activity. We will refer to this activity state as the *ground state*. Due to its irregularity, the resulting activity of pyramidal cells can lead to spontaneous pattern activation. The Poisson spike trains are the main source of temporal stochasticity in the network and are the reason why pattern activation is a stochastic process.

In the following, we will describe our setup for a working memory experiment to benchmark the capacity of our AdEx-based networks to store and retrieve patterns.

3.3 Pattern completion task as a benchmark

In the previous section we have introduced the layer 2/3 (L23) network and its architecture. In this section we propose a working memory task to assess the pattern storage and retrieval capacity of these networks. The resulting success rate of this task will be referred to as the *memory characteristic* of the network.

For our simulations we will use the adaptive exponential LIF model (AdEx) with adaptation and synaptic short-term plasticity, which were explained in detail in Sections 2.3, 2.4. Both components, the adaptation and the short-term plasticity mechanism play an important role for our L23 networks. The role of the AdEx parameters is to adapt

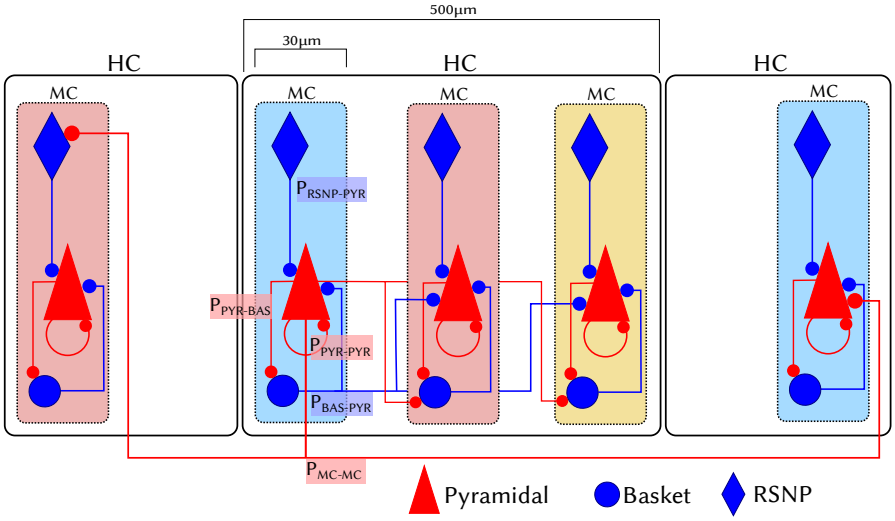

Figure 3.4: A schematic of the L23 attractor network topology. Each hypercolumn consists of multiple minicolumns. In each minicolumn, three different populations are located: pyramidal (PYR), basket (BAS) and regular spiking non-pyramidal (RSNP) cells. A pattern is represented by a group of minicolumns which are connected by excitatory PYR-PYR connections and encode a specific firing pattern with their PYR firing activity. To constrain network activity, inhibitory basket cells enforce inhibition via BAS-PYR connections. To increase the probability that only one pattern is active, PYR-RSNP connections between competing minicolumns activate RSNP cells, which inhibit their own PYR cells. This implements a so-called winner-take-all mechanism. Figure has been taken from *Breitwieser* (2011).

the membrane potential of pyramidal cells to strong excitation during attractor states. Excitation from other pyramidal cells not only excites PYR neurons, but also induces the adaptation current w. This counteracts the increasing PYR firing rate by lowering the membrane potential. As a result, the pyramidal cells exhibit biologically plausible firing rates during the attractor state. Also, the network can leave the attractor state due to declining firing activity when adaptation becomes strong enough. This weakens the attractor state, making a transition to other network states more probable. Similar to adaptation, the Tsodyks-Markram (TSO) mechanism in Section 2.4 also serves as a stabilization mechanism for the network. This synaptic depression mechanism weakens the synaptic connections between pyramidal cells from different minicolumns. During attractor states, the synapses exhaust their synaptic resources and excitation is reduced. This limits the duration of attractor states approximately to the intrinsic time scale of the synaptic depression, τ_{rec}, as illustrated in Figure 3.5.

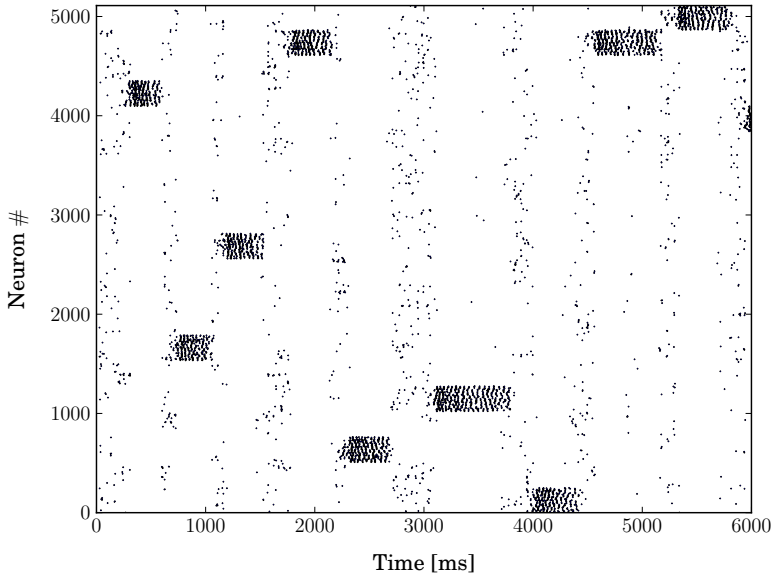

Figure 3.5: An example spike train of a full L23 network consisting of 8 hypercolumns and 20 patterns ($8H \cdot 20U$). The network is comprised of 5125 neurons in total and each pattern includes 8 minicolumns. We see two types of network states. The high activity states indicate an attractor state of minicolumns that comprise a pattern. During these high activity periods, a stored pattern is retrieved by the network, silencing minicolumns from the remaining patterns. The other type of network state is the ground state which occurs between attractor states and exhibits a sparse, homogenous firing activity of all PYR neurons. This ground state can be seen as a transient state between emerging attractor states. Its average duration depends on both number and size of competing patterns. During these ground states all PYR neurons exhibit sparse random activity modulated by Poisson background input. Figure is taken from *Rivkin* (2014).

3.3.1 Criteria for pattern completion

In order to assess the memory characteristics of our L23 attractor networks, a suitable simulation setup will be defined in this section. Since we are interested in the storage and retrieval of patterns, we will first explain both processes.

The patterns are stored in the network by creating excitatory PYR-PYR connections between minicolumns in different hypercolumns. To retrieve a target pattern, we randomly choose a fixed number of minicolumns of the targeted pattern and feed additional excitatory Poisson input into the PYR neurons. The biological correlate of this Poisson source originates from the cortical layer 4 (L4) and stimulates PYR neurons in layers 2 and 3 (*Miller*, 2003). In our simulations, this Poisson input is generated in the same way as the background input which induces the background activity in every PYR neuron. The L4 Poisson stimulus has a higher rate, but is connected with a connection probability of 70% to a PYR neuron (see Appendix A.3). The L4 input causes an upsurge of the PYR firing rate. This activity is spread via the excitatory connections between minicolumns (Connection 3.6). For a successful pattern retrieval, all target PYR populations exhibit strong activity. To evaluate whether the network can retrieve a pattern, we analyze the resulting network state in the following $T_{\text{sim}} = 400\,\text{ms}$ after the activation.

We consider the pattern retrieval successful if the following conditions are fulfilled:

- During T_{sim}, all PYR populations of the pattern minicolumns exhibit strong activity
- The PYR population activity from other patterns declines significantly during T_{sim}.

We say that a pattern is activated if in every hypercolumn, the PYR population of the stimulated pattern is the most active one. To compare activity, we average over the PYR population rates of the respective minicolumns (Figure 3.6).

The advantage of this criterion is that no additional parameters or membrane potential traces are required, but only the spike counts of PYR populations. Extensive discussions on advantages and downsides of this metric can be found in *Breitwieser* (2011); *Rivkin* (2014).

3.4 Measurement protocol of the memory characteristic

To measure the L23 pattern retrieval success rate, we will use the following protocol.

- **Network generation:** We create an AdEx network with H hypercolumns and U minicolumns per hypercolumn. We assign N_P random patterns by establishing excitatory PYR-PYR connections amongst the randomly chosen minicolumns. For H total hypercolumns in the network, each pattern has at most one minicolumn in a hypercolumn ($N_P \leq H$). A minicolumn can encode for multiple patterns, as exemplified in Figure 3.3. Patterns sharing these MCs are non-orthogonal. As

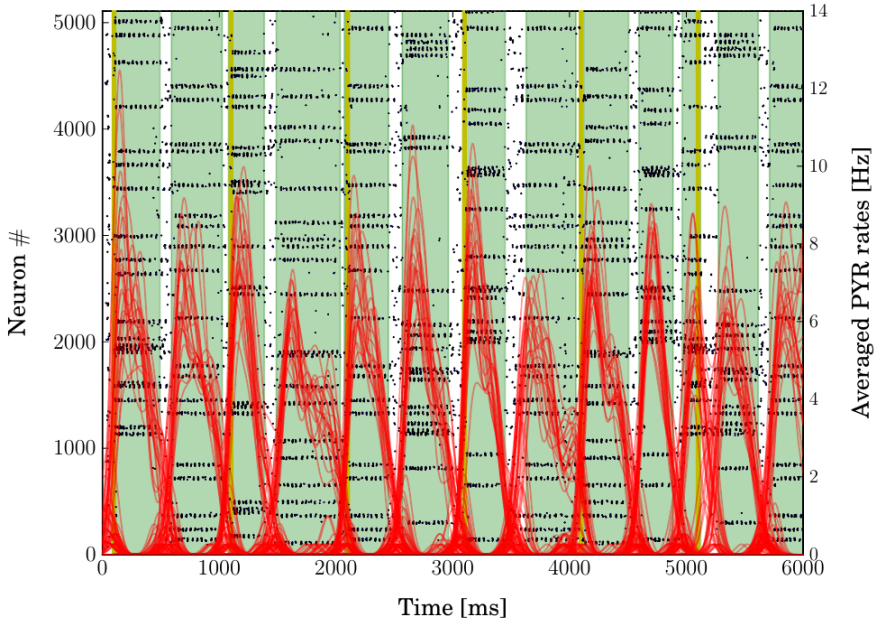

Figure 3.6: The spike train pattern shows the L23 network's response to L4 stimulus. The L4 stimulus is active in the yellow-colored time frames. As the L4 Poisson stimulus is added (yellow), the probability of switching into an attractor state (green background) increases. We can see that in most cases, the PYR populaton rates increase (red curves) immediately after the L4 stimulus, initiating attractor states. In some cases the L4 stimulus does not induce attractor states, as seen in the time frame 5100 ms − 5200 ms. Figure is taken from *Rivkin* (2014).

the number of patterns is increased for a fixed network size, non-orthogonality is inevitable. The PYR populations that encode for multiple patterns then have additional connections. In our simulations, we vary the number of patterns from 6 to 48 and the number of stimulated minicolumns from 8 to 20. All simulations have been conducted using networks of size $20H \cdot 8U$.

- **Stimulation of patterns:** For a pattern completion run, we first choose a target pattern that we want to activate. Then we randomly select minicolumns from the target pattern. The number of selected minicolumns is also called *occlusion*. We stimulate these patterns for a short duration by adding Poisson stimulus. The connection probability to this L4 stimulus is $P_{\mathrm{L4}\to\mathrm{PYR}} = 0.75$ with a total input rate per neuron of $\nu_{\mathrm{L4}\to\mathrm{PYR}} = 1500\,\mathrm{Hz}$.

- **Simulation trials:** We set up the network and run all pattern completion trials during one simulation. One completion trial lasts 1000 ms. We repeat this simulation run 10 times with different random seeds to collect statistics. In-between completion trials there is a buffer time of 500 ms to ensure that the network is not in the stimulated state anymore before the next trial.

- **Invalid trials in case of spontaneous attractor states:** We evaluate the PYR population rates after the L4 stimulus, as defined in Section 3.3.1. This evaluation also includes verifying whether the run was a valid trial, as there are conditions that can invalidate the trial. One important reason is the occurrence of spontaneous attractor states, i.e., persistent firing states which emerge by chance in PYR populations. If a pattern spontaneously starts firing before stimulation of the target pattern, it impacts the probability of pattern completion. Since a (spontaneously) emerging pattern silences all the others by the WTA mechanism, any stimulated pattern is less likely to establish an attractor state. Also, if the target pattern itself was in such a spontaneous attractor state before the L4 stimulation, side effects will still persist during the stimulation. The side effects are the resulting negative adaptation and short-term depression of PYR cells, which can last long enough to decrease the activation probability when the stimulus is introduced. As a consequence, if we detect spontaneous attractor states during the trial, we discard the trial.

The network sizes are set according to existing studies (*Lundqvist et al.*, 2006, 2010; *Johansson et al.*, 2006; *Johansson and Lansner*, 2007; *Petrovici et al.*, 2014). If not stated otherwise, we choose a network configuration of $20H \cdot 8U$. In this case, the network simulations can be performed within a day and analysis of spike data can be conducted in several hours. Each MC consists of 30 PYR cells, two RSNP cells and one BAS cell.

Results from previous work (*Petrovici et al.*, 2014) suggest that network sizes in this order of magnitude yield results that are valid even for larger networks. All remaining neuron and synapse parameters for the simulations can be found in Appendix A.2.

3.4.1 Measurement results

The results of this series of simulations can be seen in Figure 3.7, where the memory characteristic is represented as a hyperplane. Here we vary the total number of patterns in the network and the number of stimulated minicolumns during a trial (occlusion). The z-value of Figure 3.7 gives the percentage of successful pattern completions, which is also indicated by the color of the hyperplane.

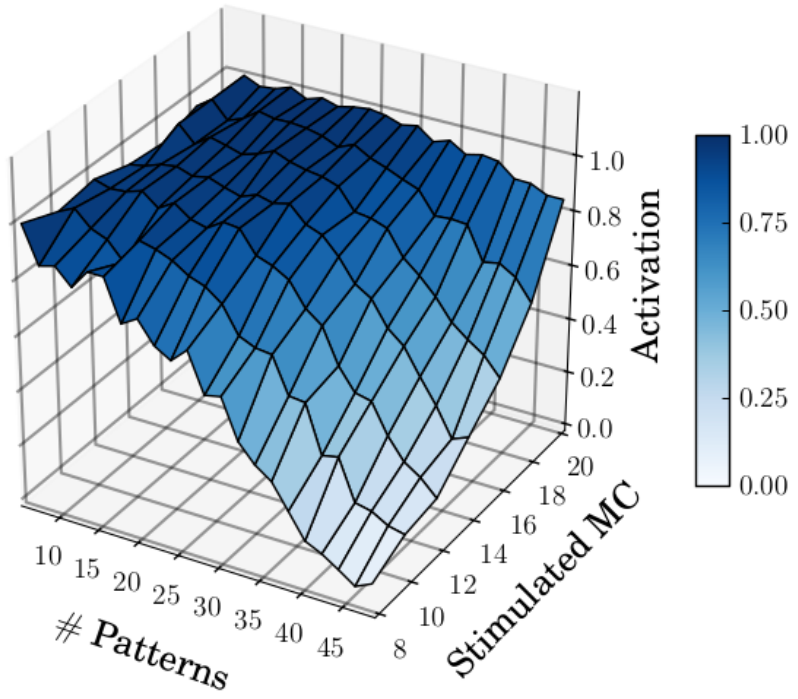

Figure 3.7: The memory characteristic of a $20H \cdot 8U$ network is the pattern completion success ratio, depending on the number of stored and stimulated minicolumns per pattern. We vary both paramters during the completion run on the `Pattern`− and `SimulatedMC`-axis. Each pattern configuration is initialized 10 times and the success ratio is an average. The pattern completion activation ratio critically depends on the number of patterns stored in the network. For high pattern densities, a satisfying success ratio ($\geq$ 80%) can only be achieved if a high percentage of pattern minicolumns is stimulated. If fewer minicolumns are stimulated, the stimulus can be too weak compared to the background noise. Also, if only few minicolumns are stimulated and shared MCs are among them, then the cue becomes ambiguous and activates multiple patterns at once. Figure is taken from *Rivkin* (2014).

For a low number of patterns, the network can store the patterns reliably, as even

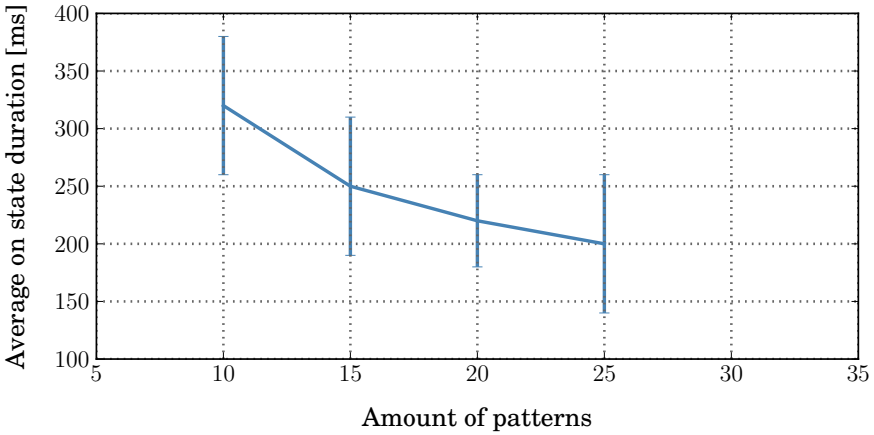

Figure 3.8: The relationship between the length of attractor states and the amount of patterns that are stored in the network. The network parameters are chosen as in Figure 3.9. Synaptic depression and spike adaptation limit the attractor length, as they counteract the increased synaptic activity. The error bars denote standard deviations from 10 trial runs for all pattern completion trials. Figure is taken from (*Rivkin*, 2014).

the minimum occlusion number leads to a very high completion rate. As the pattern count increases, larger numbers of minicolumns need to be stimulated to complete the pattern. This is expected, since a larger number of patterns increases the number of similar patterns. This leads to activation of patterns that are similar to the targeted one. A second important observation of our simulations are the durations of detected attractor states, (Figure 3.8). Here we see that the duration decreases with an increasing number of patterns in the network. To find the reason for this decrease, we look at the spike trains during the pattern completion trials in Figure 3.9. In addition to registered attractor states (green background), we see strong ground state activity. Although the ground states are not recognized as attractor states, in many PYR and RSNP populations the activity is very high. The reason for this activity increase is given by stimulated minicolumns that encode multiple patterns. The PYR populations in these minicolumns excite PYR populations in patterns where they also encode, but which are not activated by L4 stimuli. In this scenario, PYR populations from many competing minicolumns are active, suppressing the stimulated pattern. As a result, there is no dominating pattern and therefore no attractor state is detected. We can identify these elevated ground states due to strong excitation (PYR activity) and inhibition (RSNP activity) at the same time (Figure 3.4).

These ambiguous high-activity states are very important for our further discussion of the L23 network and we will call them *spurious attractors*. The elevated network activity during these spurious states causes the short duration of detected attractors (green), because elevated activity facilitates adaptation of the membrane and depletion of synaptic resources. If an unambiguous attractor state emerges soon after a certain period

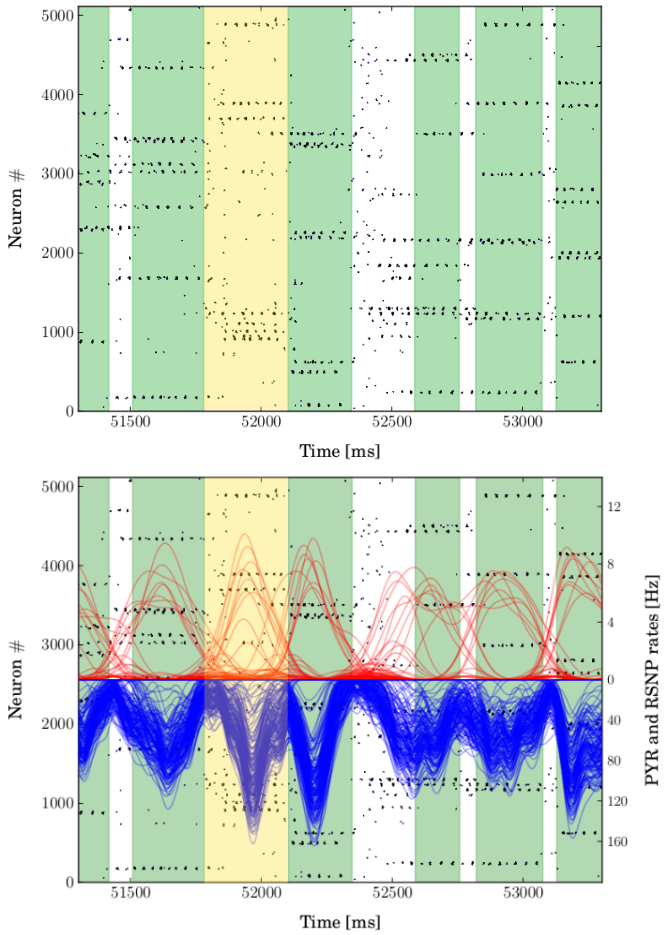

Figure 3.9: **(Top)** Spike trains of a pattern completion run with high pattern density. We see ground states (white background) and detected attractor states (green background). Even in ground states, the network exhibits elevated firing activity. Since many patterns are stored in the network, active minicolumns are likely to encode for multiple competing patterns. These minicolumns facilitate activity in multiple patterns, which initiates inhibition directed to the targeted pattern. This results in high-activity ground states where many competing minicolumns are active, resulting in *spurious attractors*. Between 51700 ms and 52100 ms we see strong activity from more than one pattern (yellow). **(Bottom)** The same spike train pattern is seen as in (A), overlayed with excitatory PYR population rates (red) and inhibitory RSNP activity (blue) from all $8 \cdot 20$ minicolumns. We see a pronounced spurious attractor state during the time period between 51700ms to 52100ms, where few red peaks of active PYR populations induce inhibition to large parts of the network, including the targeted pattern itself. This network-wide inhibition is illustrated by the high amplitudes of blue peaks. These peaks show inhibition by RSNP populations that belong to patterns from shared MCs. Activated MCs encoding multiple patterns activate RSNP populations of non-targeted patterns. The resulting inhibition is then directed to all PYR populations, including the targeted one. Figure is taken from *Rivkin* (2014).

of spurious activity, it is still affected by adaptation and lack of synaptic resources. Even during the attractor state, the dominating pattern still receives inhibition from MCs that are shared with competing patterns.

3.4.2 Detection of attractor states in presence of spurious attractors

The results presented in Section 3.4.1 fully rely on the detection and measurement of attractor states during pattern completion trials. For high pattern densities, the stimulated minicolumns excite competing patterns, since these MCs are likely to encode for multiple patterns. In our current setup, these trials are considered valid and lead to a lower success ratio of pattern completion. The lower success ratio results from inhibition of the stimulated patterns by other non-orthogonal patterns in the network. Thus, some minicolumns that belong to the target patterns are not necessarily the ones with the highest activity in their respective hypercolumn. This leads to a failed pattern completion result due to the conditions set in Section 3.3.1. A manual assessment of the network states reveals that many failed pattern completion results could still be considered successful because a majority of target minicolumns is still dominant in their respective hypercolumn. Although not all pattern MCs are the dominant ones in their respective HC, we can argue that a pattern generated by the network is active if it shows the strongest similarity to the targeted pattern. To consider such trial runs successful, we need to modify our current pattern completion conditions that were defined in Section 3.3.1.

3.5 Pattern detection with distance measure

In this section we will make a more formal approach to interpret attractor states and adjust the present methodology of state detection. As argued in the previous section, we will not require that every MC in the target pattern is the dominant one in its hypercolumn to register a successful trial. Instead, we introduce a metric which will quantify the distance between network activity after L4 stimulation and every stored pattern. A trial is defined as successful if the distance between network activity and the target pattern is the shortest one. We define the distance between different activity states as,

$$p_k = 1 - \frac{\vec{m_k} \cdot \vec{m_{\text{net}}}}{\vec{m_k} \cdot \vec{m_k}} \quad . \tag{3.8}$$

The distance is based on comparison of the PYR population rates in each HC. Here, $\vec{m_k} = \left(r_k^1, \cdots, r_k^N\right)$ is the population rate vector of a pattern k with length N. It includes entries of the N PYR population rates that encode for this pattern. The population rates are calculated by applying a gaussian kernel on spike trains of each PYR cell and averaging afterwards over the whole PYR population. The rates $r_k^{\{1,\dots N\}}$ result from a complete activation of pattern k and are measured before the pattern completion runs.

Therefore, the rate vector $\vec{m_k}$ is used as a reference for the population rates of each pattern during their active state. The population rates $\vec{m_{\text{net}}} = \left(r^1_{net}, \cdots, r^N_{net}\right)$ are network PYR rates measured after L4 stimulation. The dot product between both vectors, $\vec{m_k} \cdot \vec{m_{\text{net}}}$, computes the distance between network activity and activity of pattern k. The components in the vectors $\vec{m_k}$ can vary for different networks and therefore are normalized by $\vec{m_k} \cdot \vec{m_k}$ in the denominator. The resulting number $\frac{\vec{m_k} \cdot \vec{m_{\text{net}}}}{\vec{m_k} \cdot \vec{m_k}}$ is then subtracted from unity to define a distance measure between rate vectors $\vec{m_k}$ and $\vec{m_{\text{net}}}$. If the complete pattern k is perfectly stimulated during the trial, we would yield $\vec{m_{\text{net}}} \approx \vec{m_k}$, resulting in $p_k \approx 0$. In case of stimulating only minicolumns that do not encode pattern k, $\vec{m_{\text{net}}}$ and $\vec{m_k}$ are orthogonal and result in $p_k \approx 1$. Therefore, the distances lie in the range $p_k \in [0, 1]$. In Figure 3.10 we see the continuous distance measure for four patterns.

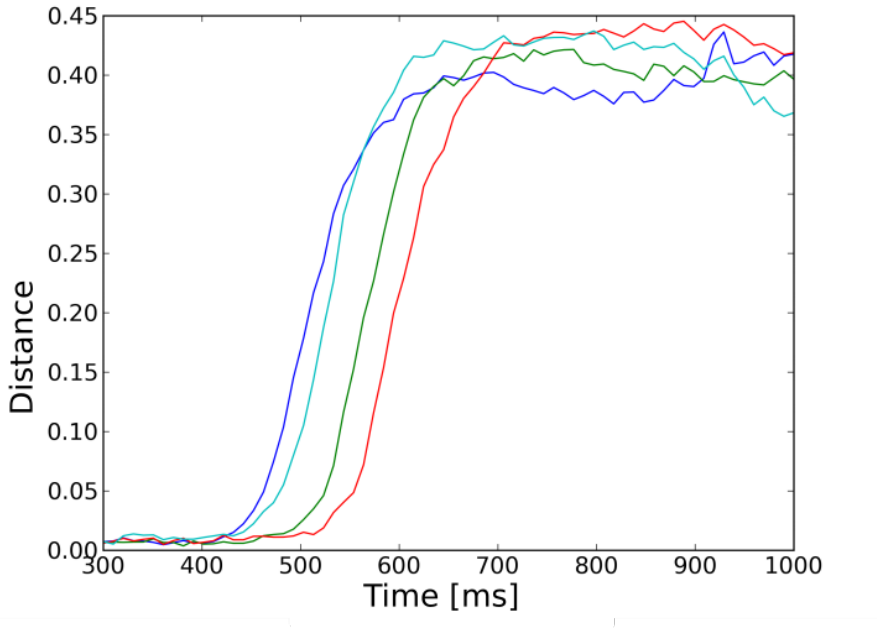

Figure 3.10: Distance measure p_k for four network patterns. At time $t = 0\,\text{ms}$, we stimulate all minicolumns that encode the pattern and start measuring the distance on the `Distance`-axis. In the interval $\sim 400\,\text{ms} - 500\,\text{ms}$ the attractors begin to fade and the network states become less similar to the pattern. This is indicated by the increase of p_k.

Examples of this measure can be seen in Figure 3.11. Since the distance p is a continuous metric, it does not provide information whether the distance is low enough to interpret the network state as an attractor state at all or whether the elevated activity corresponds to a spurious attractor. To register a pattern, we define a threshold value p^{thresh} for the distance,

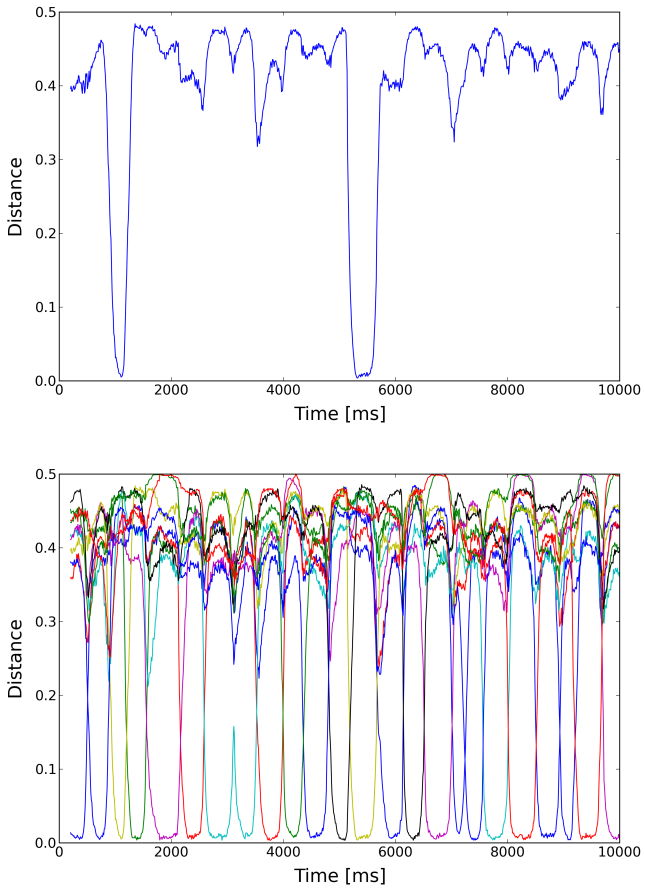

Figure 3.11: **(Top)** The y-axis shows the continuous distance p_k (Equation 3.8) that quantifies the difference between the network state and a specific pattern k. We see two time frames where the network switches into the target attractor state of pattern k, indicated by low distance value p_k. **(Bottom)** Different patterns are activated sequentially during a simulation run. On the y-axis, the different colors show the distance p_k between the network state and different patterns k. The L4 excitation of a pattern k causes a drop in the respective distance p_k for an amount of time that varies from 100ms to 500ms. Due to shared MCs in these patterns, the distance of similar attractors $p_{k \neq l}$ can also become low.

$$p_k \geq p^{\text{thresh}} \Rightarrow \text{not attractor state } k \quad , \tag{3.9}$$

$$p_k < p^{\text{thresh}} \Rightarrow \text{attractor state } k \quad . \tag{3.10}$$

This raises the problem of defining a threshold p^{thresh} that is applicable to all attractor states. If the threshold is set too low, we do not register all valid attractor states. If the threshold is set too high, we wrongfully register spurious states as attractors. Any threshold value would be an arbitrary parameter requiring adjustment for each specific case, since it depends on the specific network size and number of stored patterns. To avoid the introduction of such a free parameter, we will propose a mechanism which automatically sets an appropriate threshold p^{thresh}.

3.5.1 Automatic state detection for pattern completion

The continuous distance p_k expresses the difference between a network state and pattern k. In this section we will describe a method to set a threshold p^{thresh} for a certain network configuration that determines whether a network state can be interpreted as an attractor state of pattern k.

First, we measure a distance p_k for each pattern k in the network, immediately after an L4 stimulation of the full pattern. The resulting distribution of distances Σ_{net} can be seen in Figure 3.12, most patterns can be produced reliably by the network, i.e., most distances are low and accumulate on the left side. Still, some patterns are misclassified by the network, indicated by a high distance value on the right end of the distribution. In order to define a threshold p^{thresh}, we estimate the network's response to patterns that are not stored, so-called "test patterns". Therefore, we present the network unknown, random patterns and evaluate the distance. We stimulate random MCs from the network and again calculate the distance p in a time frame of 500ms after stimulation. We have averaged the network's response over 10 trials for each pattern, resulting in the distance distribution Σ_{test} (red color in Figure 3.12). To determine the threshold p^{thresh}, we sum both distributions, Σ_{net} and Σ_{test}, and define p^{thresh} at the minimum value of their sum,

$$p^{\text{thresh}} := \min\left(\Sigma_{\text{net}} + \Sigma_{\text{test}}\right) \tag{3.11}$$

The resulting threshold is low enough to reject a majority of test patterns and high enough to include a vast majority of patterns that are stored in the network.

The results of the memory characteristic achieved using the metric from Equation 3.8 can be seen in Figure 3.13. With the new detection method, we do not require all minicolumns of the stimulated pattern to have the highest PYR firing rate in the respective hypercolumns. Every network state that is similar enough to be below threshold value p^{thresh} is accepted as an attractor state. The results have improved significantly compared to the results shown in Figure 3.7.

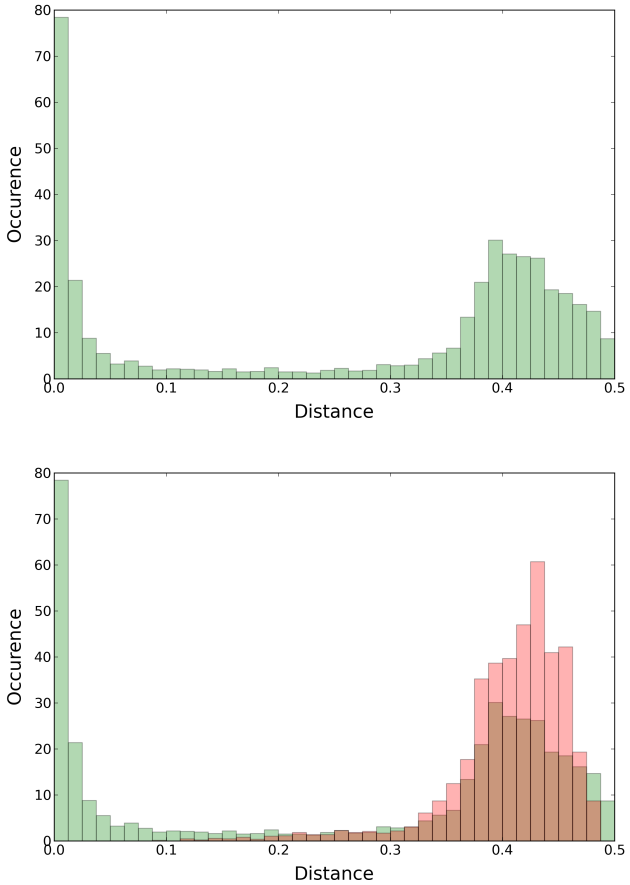

Figure 3.12: **(Top)** The stored attractor states are compared with network states after pattern stimulation by measuring the distance metric given in Equation 3.8. The resulting distribution shows low distance values for a majority of patterns. This indicates that the network states are indeed similar to target attractor states. We also see occurrence of higher distances, leading to a low peak at the right. The cause of the right peak is a spread in attractor state durations of the network. Attractor states that have ended before the measurement time ends contribute to the peak at the right. This shows the difficulty to capture all possible attractor states. The variation in attractor length can also be seen in Figure 3.10. **(Bottom)** The plot shows the same distribution as in (A), colored green, and measurements of test patterns, i.e., patterns not ingrained in the network (red). Since the patterns are not ingrained in the network topology, the distance is very high on average. The resulting threshold for registering attractor states is extracted by taking the minimum of the sum of both distributions, as described in Equation 3.11.

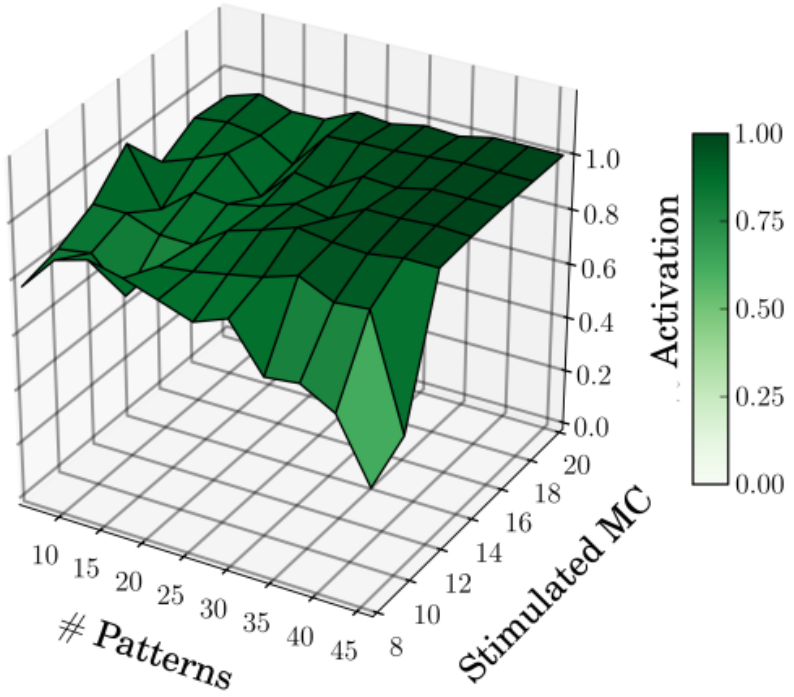

Figure 3.13: Using the distance in Equation 3.8 we see vast improvements of pattern completion for higher pattern densities compared to the previously used method (Figure 3.7) defined in Section 3.3.1. The success ratio stays approximately constant even for a low number of stimulated minicolumns. Therefore, this metric is the preferred method to detect attractor states in the network.

3.6 Conclusion and outlook: inference with cortical layer 2/3 networks

The goal of this chapter was to investigate the working memory characteristics (Figure 3.7) of a cortical layer 2/3 attractor network using AdEx neurons. We have benchmarked the performance of these networks during a pattern completion task. We found that this measure of network performance serves as a good indicator to assess the capability of the network to store patterns and react to stimulation. The measure works best with the pattern activation condition that we have introduced in Equation 3.8. This, however, requires us to collect statistics of the network reaction to the stimulation of all patterns before the pattern completion task, and to introduce test patterns to compare the network performance.

Since we implemented AdEx neurons with short-term plasticity and adaptation, the network dynamics is complex and exhibits variability. This variability is reflected in the

attractor dynamics of the system for high pattern densities. Attractor activation during the pattern completion task is often ambiguous due to the high number of competing, non-orthogonal patterns. In such cases, spurious attractors complicate the evaluation of the memory characteristic.

4 The behavior of leaky integrate-and-fire neurons under stochastic stimulus

In the previous chapter we have studied a cortex-inspired network architecture, which was adapted to function with AdEx neurons. The inferential performance of this down-scaled cortical architecture was evaluated on a benchmark inference task. Although the performance on this task was good, we encountered conceptual difficulties at interpreting the network dynamics in terms of computation. For the functionality of the L23 attractor dynamics, the adaptation was an important component. For an interpretation of network states, it would be helpful to know the output rates of neurons for given synaptic input. For the AdEx model, this relationship is difficult to compute. Therefore, we will derive such an input-output relationship in this chapter for LIF neurons, omitting both the adaptation and the exponential terms. This relationship will help us to develop a theoretical framework to describe firing statistics for LIF neurons. Instead of studying a fully-connected network like in the previous chapter, this framework will first include only single neurons which receive stimulus that mimics network activity. Instead of fully adapting a cortical architecture as an inference framework, we will look at a single neuron from such a network. Its characterization will serve as a building block to model networks performing stochastic inference in the next chapter.

4.1 The high-conductance state in biological neurons

When we introduced the leaky integrate-and-fire model in Chapter 2, we provided a mechanistic description of a single spiking neuron. However, the exceptional computational capabilities of neural networks unfold when vast amounts of nerve cells transmit electric signals in dense neural tissue. The dynamics in in-vivo spike trains exhibit high variability and are highly sensitive to their environment (*Fiser et al.*, 2004). Due to this variability in network dynamics, recent studies employed statistical approaches to understand coding schemes that appear to be of stochastic nature (*Rolls and Deco*, 2010; *Fiser et al.*, 2010; *Berkes et al.*, 2011). However, the neuron model that we use is deterministic. We will characterize the dynamics of a single LIF neuron embedded in a network receiving high-frequency Poisson stimulus. This type of stimulus will be used to model the temporal variability that is observed in the firing patterns of cortical neurons in attentive, responsive states (*Destexhe et al.*, 2003; *Kumar et al.*, 2008a). Observations from in-vivo experiments suggest that this so-called *high-conductance state* is a hallmark of cortical information processing, as it enhances alertness and decreases response time

for integration of sensory stimuli (*Hô and Destexhe*, 2000; *Shu et al.*, 2003; *Chance et al.*, 2002).

It is important to note that the high-frequency stimulus which every cortical neuron receives, is of stochastic nature. Each cortical neuron can have more than 10000 synaptic connections on average (*Binzegger et al.*, 2004). These connections exhibit a certain degree of irregularity in signal transmission and can be considered independent from each other (*Gerstner and Kistler*, 2002, Chapter 5). Here we refer to presynaptic signal transmission events as spikes which are propagated action potentials, as introduced in Section 2.2. The rate of spikes received by a cortical neuron can be estimated to 1000 Hz (*Pare et al.*, 1998). It can be argued that many of these presynaptic connections are random, so their presynaptic activity is uncorrelated and the presynaptic spikes have a uniform time distribution. Due to the Palm-Khintchine theorem (*Heyman and Sobel*, 2003), all the spike time events from this input can be characterized as Poisson point processes (*Brunel*, 2000; *Deco and Jirsa*, 2012). Since the sum of these processes is again a Poisson point process with a correspondingly increased rate, we can interpret the sum of these time series as one time series. This argument is very important, as we subsume all inputs from different synapses under one time series including the total number of events. In the following section we will use the Poisson properties to formalize the statistical moments of an LIF neuron resulting from this type of synaptic stimulus. A part of the results described in this chapter also appear in already published material, (*Petrovici et al.*, 2013, 2015a, 2016), which was co-authored by the author of this thesis.

4.2 The high-conductance state of leaky integrate-and-fire neurons

In the LIF model, the primary impact of presynaptic stimulus on the neuron happens through its synaptic conductance. This conductance has been defined in Equation 2.4 as

$$\frac{dg_i^{syn}}{dt} = -\frac{g_i^{\mathrm{syn}}}{\tau_{\mathrm{syn}}} + \sum_s w_i \cdot \delta(t - t_s^i) \quad . \tag{4.1}$$

We assume that the synaptic time constant τ_{syn} is equal for all synapses i (therefore, $\tau_{\mathrm{syn\,i}} \equiv \tau_{\mathrm{syn}}$). The solution of this differential equation is a linear superposition of exponential kernels,

$$g_i^{\mathrm{syn}}(t) = \sum_s \Theta(t - t_s^i) w_i \cdot \exp\left(\frac{t_s^i - t}{\tau_{\mathrm{syn}}}\right) \quad . \tag{4.2}$$

Now we can use the membrane potential differential Equation 2.1,

$$C_{\mathrm{m}} \cdot \frac{d}{dt}u(t) = -g_{\mathrm{l}} \cdot [u(t) - E_{\mathrm{l}}] + I_{\mathrm{syn}} + I_{\mathrm{ext}} \quad , \tag{4.3}$$

and plug Equation 4.2 into the synaptic current of Equation 2.3,

$$I_{\text{syn}}(t) = \sum_i \sum_s \Theta(t - t_s^i) w_i \cdot \exp\left(\frac{t_s^i - t}{\tau_{\text{syn}}}\right) \cdot [E_{\text{i}}^{\text{rev}} - u(t)] \quad . \tag{4.4}$$

Now we can write down our membrane potential with Poisson-induced synaptic current in Equation 4.4,

$$C_{\text{m}} \cdot \frac{d}{dt} u(t) = -g_{\text{l}} \cdot [u(t) - E_{\text{l}}] + \sum_i \sum_s \Theta(t - t_s^i) w_i \cdot \exp\left(\frac{t_s^i - t}{\tau_{\text{syn}}}\right) \cdot [E_{\text{i}}^{\text{rev}} - u(t)] + I_{\text{ext}} \quad . \tag{4.5}$$

This differential equation is a complete formulation of the LIF neuron stimulated by Poisson spike trains on each synapse i. The current I_{ext} is a parameter that shifts the membrane potential by a constant value. In the following, we will derive the statistical moments of the membrane potential.

First, we transform Equation 4.5 by introducing the total conductance of the membrane,

$$g_{\text{tot}}(t) = \sum_i g_i^{\text{syn}}(t) + g_{\text{l}} \quad . \tag{4.6}$$

We divide Equation 4.5 by the total conductance in Equation 4.6, yielding

$$\frac{C_{\text{m}}}{g_{\text{tot}}(t)} \cdot \frac{d}{dt} u(t) = -\frac{g_{\text{l}} \cdot [u(t) - E_{\text{l}}]}{g_{\text{tot}}(t)} + \frac{\sum_i \sum_{t_s} \Theta(t - t_s^i) w_i \cdot \exp\left(\frac{t_s^i - t}{\tau_{\text{syn}}}\right) \cdot [E_{\text{i}}^{\text{rev}} - u(t)]}{g_{\text{tot}}(t)} + \frac{I_{\text{ext}}}{g_{\text{tot}}(t)} \quad . \tag{4.7}$$

On the left hand side (LHS) we see the term $\frac{C_{\text{m}}}{g_{\text{tot}}(t)}$, which has the unit of a time constant and is similar to the membrane time constant $\tau_{\text{m}} = \frac{C_{\text{m}}}{g_{\text{l}}}$, as introduced in Section 2.1. In Section 2.1 we described τ_{m} as a time constant which indicates the response time of the membrane potential to synaptic stimuli. In Equation 4.7 we can extend our notion of the membrane time constant by introducing the time-dependent *effective membrane time constant* τ_{eff},

$$\tau_{\text{eff}}(t) \coloneqq \frac{C_{\text{m}}}{\sum_i g_i^{\text{syn}} + g_{\text{l}}} \quad . \tag{4.8}$$

The time constant τ_{eff} depends on the synaptic input $\sum_i g_i^{\text{syn}}$. This relationship is crucial because it shows that the membrane potential changes fast for high synaptic conductances. As synaptic input $\sum_i g_i^{\text{syn}}$ decreases, the membrane potential change slows down. Without any synaptic input, the effective time constant τ_{eff} becomes identical to τ_{m}. The membrane potential response time can be observed at the rising flank of the shape of the postsynaptic potential (PSP) in Figure 2.1 for $\tau_{\text{eff}} < \tau_{\text{syn}}$. From now on we will always assume the synaptic time constant larger than the effective time constant because we will presuppose a fast membrane and a long synaptic interaction for our probabilistic models in the next chapter. The smaller τ_{eff}, the sharper the PSP slope will be at the beginning. For $\tau_{\text{eff}} \to 0$, the PSP approaches an exponential with an instant rise.

We can now reformulate Equation 4.7 using Equation 4.8 to obtain

$$\tau_{\text{eff}}(t)\frac{d}{dt}u(t) = u_{\text{eff}}(t) - u(t) \quad , \tag{4.9}$$

where we define a new auxiliary variable, the effective potential u_{eff},

$$u_{\text{eff}} \coloneqq \frac{g_{\text{l}} \cdot E_{\text{l}} + \sum_i g_i^{\text{syn}}(t) E_{\text{i}}^{\text{rev}} + I_{\text{ext}}}{g_{\text{l}} + \sum_i g_i^{\text{syn}}(t)} \quad . \tag{4.10}$$

In the numerator of Equation 4.10, u_{eff} is essentially the synaptic conductance $\sum_i g_i^{\text{syn}}(t)$, weighted by $E_{\text{i}}^{\text{rev}}$ and shifted by $g_{\text{l}} \cdot E_{\text{l}} + I_{\text{ext}}$. In the next steps we will simplify Equation 4.10 by eliminating time dependencies in the total conductance $g_{\text{tot}}(t)$. Essentially we can substitute the total conductance by its mean value if we can assume that the mean remains approximately constant during the high-conductance state. To estimate the correctness of this assumption, we need to calculate the statistical moments of the total conductance.

The calculations of the mean $\langle g^{\text{syn}} \rangle$ and variance σ_g^2 of Equation 4.2 are straightforward and can both be found in *Bytschok* (2011). The mean $\langle g^{\text{syn}} \rangle$ and variance σ_g^2 are given by

$$\langle g^{\text{syn}}(t) \rangle = w_{\text{syn}} \cdot \nu_{\text{syn}} \cdot \tau_{\text{syn}} \tag{4.11}$$

$$\sigma_g^2 = \frac{1}{2} w_{\text{syn}}^2 \cdot \nu_{\text{syn}} \cdot \tau_{\text{syn}} \quad . \tag{4.12}$$

With these quantities we can estimate the ratio between the standard deviation and the mean of synaptic conductance using the *coefficient of variation* (c_v):

$$\begin{aligned} c_v = \frac{\sigma}{\mu} &= \frac{\sqrt{\frac{1}{2} w_{\text{syn}}^2 \nu_{\text{syn}} \tau_{\text{syn}}}}{w_{\text{syn}} \cdot \nu_{\text{syn}} \cdot \tau_{\text{syn}}} \\ &= \frac{1}{\sqrt{2 \nu_{\text{syn}} \tau_{\text{syn}}}} \end{aligned} \tag{4.13}$$

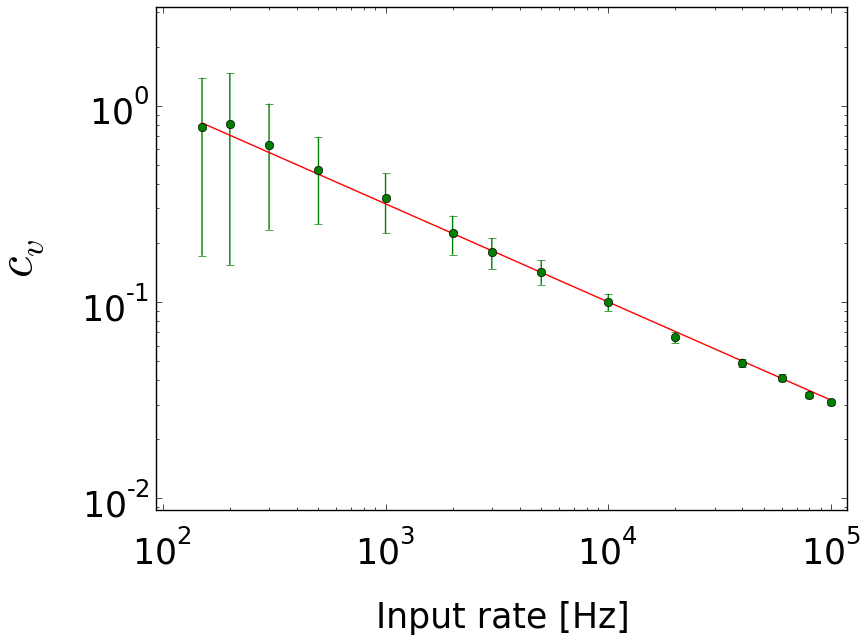

Figure 4.1: Coefficient of variation as a function of the synaptic conductance. The simulation data (green) confirms the theoretical prediction (Equation 4.13), with the square root dependence directly reflected by the $-\frac{1}{2}$ slope of the red curve. The error bars indicate the error of the mean, calculated from ten simulation runs. Figure is taken from *Bytschok* (2011).

Equation 4.13 shows that the coefficient of variation decreases with the square root of the input rate. Hence we can approximate the synaptic conductance as the mean for the high-conductance state,

$$g_i^{\mathrm{syn}} \coloneqq \langle g_i^{\mathrm{syn}}(t) \rangle \approx g_i^{\mathrm{syn}}(t) \quad . \tag{4.14}$$

Due to the relationship between synaptic conductance and the time constant in Equation 4.8, we now can assume $\tau_{\mathrm{eff}}(t)$ as constant. If we assume the synaptic conductance sufficiently high ($g_{\mathrm{tot}} \to \infty$), then $\tau_{\mathrm{eff}} \to 0$ according to Equation 4.8. In this case we can identify the membrane potential as the effective potential due to $\tau_{\mathrm{eff}} \to 0$ in Equation 4.9,

$$u(t) = \frac{g_{\mathrm{l}} \cdot E_{\mathrm{l}} + \sum_i g_i^{\mathrm{syn}}(t) E_{\mathrm{i}}^{\mathrm{rev}} + I_{\mathrm{ext}}}{g_{\mathrm{l}} + \sum_i g_i^{\mathrm{syn}}} \quad . \tag{4.15}$$

We note that the membrane reacts immediately to synaptic input, as it is a superposition of exponential synaptic kernels. From now on, we will presuppose the high-conductance state. Therefore, the membrane potential in Equation 4.15 will be assumed for all our following studies.

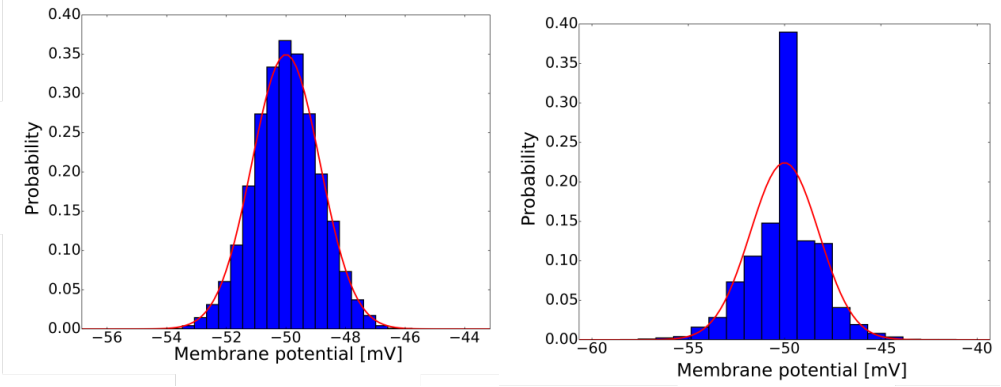

Figure 4.2: The *free* membrane potential of an LIF neuron is the membrane potential without a spiking mechanism. **(Left)** During high-frequency Poisson stimulus, the resulting histogram shows the membrane potential values (blue bars) from a $T_{\text{sim}} = 10^4$ ms simulation run. The resulting probability distribution can be described by a normal distribution (red curve) with the statistical moments from Equations 4.16, 4.17. **(Right)** In the low-frequency stimulus case, the membrane potential distribution (blue) takes a complicated form and does not allow for a straightforward theoretical description. The normal distribution curve (red) is not a good fit for the histogram.

One major consequence of the Poisson statistics of the synaptic input is the resulting shape of the membrane potential distribution. Due to the central limit theorem, the sum of synaptic kernels $g^{\text{syn}}(t)$ is a Gaussian (*Petrovici*, 2016, Section 4.3.2).

We see that the membrane potential in the high-conductance state (Equation 4.15) is a rescaled and shifted synaptic conductance. Therefore, the membrane potential itself also follows a normal distribution (Figure 4.2). Here, we introduce the so-called *free* membrane potential as the LIF membrane potential without a spiking mechanism. Therefore, the distribution is not distorted by the refractory mechanism. The mean and variance of the LIF neuron's free membrane potential have been calculated in *Petrovici* (2016) and are given as

$$\langle u(t) \rangle = \frac{E_{\text{l}} \cdot g_{\text{l}} + \sum_i w_i \cdot \nu_i \cdot \tau_{\text{syn}} \cdot E_{\text{i}}^{\text{rev}}}{\langle g_{\text{tot}} \rangle + I_{\text{ext}}} \quad , \tag{4.16}$$

$$\text{Var}\left[u(t)\right] = \sum_i \nu_i \cdot \left[\frac{w_i (E_{\text{i}}^{\text{rev}} - \langle u \rangle)\tau_{\text{syn}}}{\tau_{\text{syn}} - \tau_{\text{eff}}}\right]^2 \cdot \left(\frac{\tau_{\text{syn}}}{2} + \frac{\tau_{\text{eff}}}{2} - 2\frac{\tau_{\text{eff}}\tau_{\text{syn}}}{\tau_{\text{syn}} - \tau_{\text{eff}}}\right) \quad . \tag{4.17}$$

Note that above expressions also yield valid approximations for the membrane potential in low-conductance states, as $\tau_{\text{eff}} \neq 0$. Nevertheless, other side effects arise for low conductances, as discussed extensively in *Bytschok* (2011); *Petrovici* (2016). Since we have presupposed the high-conductance state for our framework, above equations can be simplified with $\tau_{\text{eff}} \to 0$. These statistical properties of the membrane potential are key to derive the firing statistics of an LIF neuron in the high-conductance state.

4.3 The high-conductance dynamics as an Ornstein-Uhlenbeck process

In the previous section we have derived the properties of the LIF membrane potential distribution during the high-conductance state, generated by high-frequency Poisson inputs.

In this section we introduce the so-called *Ornstein-Uhlenbeck process* (OU process). It can be understood as a continuous formulation of a physical random walk process with an exponential decay towards an equilibrium. An intuitive illustration of this process is is an overdamped oscillator (*Kungl*, 2016). The OU process has been studied extensively and has found many applications, e.g. in financial markets (*Chan et al.*, 1992), physical systems (*Cáceres and Budini*, 1997) and neuroscientific modeling (*Ricciardi and Sacerdote*, 1979). In the following, we will show that the LIF neuron in the high-conductance state exhibits the dynamics of an OU process. This equivalence plays a crucial role, as we will later apply the knowledge of OU statistics to derive the firing rate statistics of LIF neurons in the high-conductance state. We will start by giving a characterization of the OU process (a detailed explanation can be found in *Finch* (2004)).

The stochastic differential equation of the OU process can be written as

$$dx = \Theta\left[\mu - x(t)\right] dt + \sigma dW(t) \quad . \tag{4.18}$$

Here, $x(t)$ is the dynamic variable of the OU process at time t with time-average μ and Θ being the inverse time constant of the exponential decay towards μ. It is a positive constant with inverse time as unit. The two terms of the RHS of Equation 4.18 have an intuitive meaning. The first term $\Theta\left[\mu - x(t)\right] dt$ models the exponential drift towards the equilibrium value μ with time constant $\frac{1}{\Theta}$. The second part $\sigma dW(t)$ implements a so-called *Wiener-Lévy process*. It is the stochastic part of the OU differential equation. In every time step, the increment $\sigma dW(t)$ is a sample from a normal distribution, $\sigma dW(t) \in \mathcal{N}(0, \sigma^2)$. In short, it is the continuous version of a random walk of a particle with friction and a pullback towards the mean μ. Without the first term in Equation 4.18, the particle would drift away from the initial position, analogously to a random walk (*Durrett*, 1996). In the OU process the particle is drawn by the first term to an equilibrium μ with a time constant $\frac{1}{\theta}$ in an exponential decay.

In the following we will derive the equivalence of an LIF neuron in a high-conductance state and the OU process. Our starting point will be the *Fokker-Planck equation*, which is a partial differential equation describing the time evolution of a probability density under certain physical conditions. It will be the starting point to show the analogy between LIF neuron statistics in the high-conductance state and an OU process. This derivation is, in most parts, taken from *Petrovici et al.* (2016).

For the OU process, such a probability density evolution function is known (*Kolmogorov*, 1931) and can be defined as

$$f(x,t|x_0) = \sqrt{\frac{\theta}{\pi\sigma^2(1-e^{-2\theta t})}} \exp\left\{\frac{-\theta}{\sigma^2}\left[\frac{(x-\mu+(\mu-x_0)e^{-\theta t})^2}{1-e^{-2\theta t}}\right]\right\} \quad . \tag{4.19}$$

This probability density function is the unique solution to the Fokker-Planck equation,

$$\frac{1}{\theta}\frac{\partial f(x,t)}{\partial t} = \frac{\partial}{\partial x}\left[(x-\mu)f\right] + \frac{\sigma^2}{2\theta}\frac{\partial^2 f}{\partial x^2} \quad . \tag{4.20}$$

To describe the probability density evolution function of an LIF neuron in the high-conductance state, we define an auxiliary quantity, the *synaptic noise* $J^{\mathrm{syn}}(t)$, as the dynamic variable in this system,

$$J^{\mathrm{syn}}(t) \coloneqq \sum_i g_i^{\mathrm{syn}}(t) E_{\mathrm{i}}^{\mathrm{rev}} \quad . \tag{4.21}$$

It models synaptic conductance kernels weighted by their reversal potential $E_{\mathrm{i}}^{\mathrm{rev}}$. Therefore, we can write down a differential equation for $J^{\mathrm{syn}}(t)$ analogously to the synaptic kernels from Equation 2.4,

$$\frac{dJ^{\mathrm{syn}}}{dt} = -\frac{J^{\mathrm{syn}}}{\tau_{\mathrm{syn}}} + \sum_i \sum_{t_s} \Delta J^{\mathrm{syn}} \delta(t-t_s) \quad . \tag{4.22}$$

Here $\Delta J^{\mathrm{syn}} = w_i E_{\mathrm{i}}^{\mathrm{rev}}$ is the increment to the synaptic input due to a spike event. We use the so-called *Chapman-Kolmogorov equation* to describe the temporal evolution of J^{syn} after a duration Δt,

$$f(J^{\mathrm{syn}}, t+\Delta t) = \int_{-\infty}^{\infty} f(J^{\mathrm{syn}}, t+\Delta t|J', t) f(J', t) dJ' \quad . \tag{4.23}$$

This integral describes the evolution of the distribution $f(J^{\mathrm{syn}}, t)$ starting at time t with J' and evolving for a time interval Δt. Equation 4.23 integrates over all possible intermediate states J' with their respective probability leading to state $J^{\mathrm{syn}}(t+\Delta t)$. We can simplify Equation 4.23, since our synaptic input is a Poisson process. Therefore, we can choose Δt small enough, so that the probability of multiple Poisson events occurring in Δt vanishes. The probability of a single Poisson event occurring in Δt remains $\Delta t \sum_i \nu_i$. There are two possible ways J' can evolve during Δt. Either one spike occurs or no spike occurs,

$$\begin{aligned} f(J^{\mathrm{syn}}, t+\Delta t|J') = &\left[1 - \Delta t \sum_i \nu_i\right] \delta\left[J^{\mathrm{syn}} - J' \exp\left(-\frac{\Delta t}{\tau_{\mathrm{syn}}}\right)\right] \\ &+ \Delta t \sum_i \nu_i \delta\left[J^{\mathrm{syn}} - \left(J' + \Delta J_i^{\mathrm{syn}}\right) \exp\left(-\frac{\Delta t}{\tau_{\mathrm{syn}}}\right)\right] \quad . \end{aligned} \tag{4.24}$$

The first term describes the approximated Poisson probability $[1 - \Delta t \sum_i \nu_i]$ that no spike occurs and J' decays exponentially resulting in $J^{\text{syn}} = J' \exp(-\frac{\Delta t}{\tau_{\text{syn}}})$. The second term represents the occurrence of a spike during Δt, incrementing J' by ΔJ_i^{syn}. Their sum $J' + \Delta J_i^{\text{syn}}$ decays exponentially over time Δt.

We can insert 4.24 into the Chapman-Kolmogorov Equation 4.23. The integral eliminates the δ-functions, resulting in

$$f(J^{\text{syn}}, t + \Delta t) = \left(1 - \Delta t \sum_i \nu_i\right) \exp\left(\frac{\Delta t}{\tau_{\text{syn}}}\right) f\left[J^{\text{syn}} \exp\left(\frac{\Delta t}{\tau_{\text{syn}}}\right), t\right] + \Delta t \sum_i \nu_i \exp\left(\frac{\Delta t}{\tau_{\text{syn}}}\right) f\left[J^{\text{syn}} \exp\left(\frac{\Delta t}{\tau_{\text{syn}}}\right) - \Delta J_i^{\text{syn}}, t\right] \quad . \tag{4.25}$$

Since we assumed $\Delta t \to 0$, we can now perform a Taylor expansion of $f(J^{\text{syn}}, t + \Delta t)$ up to first order in Δt,

$$f(J^{\text{syn}}, t + \Delta t) \approx f(J^{\text{syn}}, t) + \left.\frac{\partial f(J^{\text{syn}}, t + \Delta t)}{\partial \Delta t}\right|_{\Delta t = 0} \Delta t \quad . \tag{4.26}$$

We can now reformulate the equation above and take the partial derivative on both sides of Equation 4.25 by $\partial \Delta t$,

$$\begin{aligned}
\frac{f(J^{\text{syn}}, t + \Delta t) - f(J^{\text{syn}}, t)}{\Delta t} &= \left.\frac{\partial f(J^{\text{syn}}, t + \Delta t)}{\partial \Delta t}\right|_{\Delta t = 0} \\
&= \Bigg\{ - \sum_i \nu_i \exp\left(\frac{\Delta t}{\tau_{\text{syn}}}\right) f\left[J^{\text{syn}} \exp\left(\frac{\Delta t}{\tau_{\text{syn}}}\right), t\right] \\
&\quad + \left(1 - \Delta t \sum_i \nu_i\right) \frac{1}{\tau_{\text{syn}}} \exp\left(\frac{\Delta t}{\tau_{\text{syn}}}\right) f\left[J^{\text{syn}} \exp\left(\frac{\Delta t}{\tau_{\text{syn}}}\right), t\right] \\
&\quad + \left(1 - \Delta t \sum_i \nu_i\right) \frac{1}{\tau_{\text{syn}}} \exp\left(2\frac{\Delta t}{\tau_{\text{syn}}}\right) J^{\text{syn}} \frac{\partial f\left[J^{\text{syn}} \exp\left(\frac{\Delta t}{\tau_{\text{syn}}}\right), t\right]}{\partial J^{\text{syn}} \exp\left(\frac{\Delta t}{\tau_{\text{syn}}}\right)} \\
&\quad + \sum_i \nu_i \exp\left(\frac{\Delta t}{\tau_{\text{syn}}}\right) f\left[J^{\text{syn}} \exp\left(\frac{\Delta t}{\tau_{\text{syn}}}\right) - \Delta J_i^{\text{syn}}, t\right] \\
&\quad + (\dots)\Delta t \Bigg\}_{\Delta t = 0} \quad .
\end{aligned} \tag{4.27}$$

Now we can finally set $\Delta t = 0$ in Equation 4.27. This simplifies the equation, resulting in

$$\frac{\partial f(J^{\text{syn}}, t)}{\partial t} = \frac{1}{\tau_{\text{syn}}} \frac{\partial}{\partial J^{\text{syn}}} \left[J^{\text{syn}} f(J^{\text{syn}}, t)\right] + \sum_i \nu_i \left[f(J^{\text{syn}} - \Delta J_i^{\text{syn}}, t) - f(J^{\text{syn}}, t)\right] \quad . \tag{4.28}$$

At this point we can discuss conditions which need to be valid to sustain a high-conductance state. In the biological high-conductance state, the impact of single spike events on the conductance is negligible. It is the sum of synaptic kernels that generates a noise environment of a biological neuron (*Burkitt*, 2006). As a consequence, the increment of the synaptic noise due to a single event, $\Delta J^{\mathrm{syn}} = w_i E_{\mathrm{i}}^{\mathrm{rev}}$, can be considered as negligible, i.e., $\Delta J^{\mathrm{syn}} \to 0$.

We can use this approximation and apply it to Equation 4.28, performing a Taylor expansion up to the second order at $\Delta J^{\mathrm{syn}} = 0$. This expansion is also known as the *Kramers-Moyal expansion* (*Paul and Baschnagel*, 1999) and results in

$$\frac{\partial f(J^{\mathrm{syn}},t)}{\partial t} = \frac{1}{\tau_{\mathrm{syn}}} \frac{\partial}{\partial J^{\mathrm{syn}}} \left[\left(J^{\mathrm{syn}} - \sum_i \nu_i \Delta J_i^{\mathrm{syn}} \tau_{\mathrm{syn}} \right) f(J^{\mathrm{syn}},t) \right] + \frac{\sum_i \nu_i \Delta J_i^{\mathrm{syn}2}}{2} \frac{\partial^2 f(J^{\mathrm{syn}},t)}{\partial {J^{\mathrm{syn}}}^2} \quad . \tag{4.29}$$

Above equation has the same form as the Fokker-Planck Equation 4.20 and we can identify the parameters μ, σ and Θ in the LIF domain,

$$\theta = \frac{1}{\tau_{\mathrm{syn}}} \quad , \tag{4.30}$$

$$\mu = \frac{I_{\mathrm{ext}} + g_{\mathrm{l}} E_{\mathrm{l}} + \sum_i \nu_i w_i E_{\mathrm{i}}^{\mathrm{rev}} \tau_{\mathrm{syn}}}{\langle g_{\mathrm{tot}} \rangle} \quad , \tag{4.31}$$

$$\frac{\sigma^2}{2} = \frac{\sum_i \nu_i \left[w_i \left(E_{\mathrm{i}}^{\mathrm{rev}} - \mu \right) \right]^2 \tau_{\mathrm{syn}}}{2 \langle g_{\mathrm{tot}} \rangle^2} \quad . \tag{4.32}$$

The equivalence between the Fokker-Planck Equation 4.20 and the high-conductance formulation in Equation 4.29 shows that a LIF neuron in the high-conductance state exhibits the dynamics of an OU process. Consequently, the statistical moments of the OU process, μ and σ, in Equations 4.31,4.32 are identical to the LIF-based statistical moments $\langle u(t) \rangle$ and $\mathrm{Var}\left[u(t)\right]$ from Equations 4.16, 4.17. We can also see that Θ is the inverse of the synaptic time constant. This result is intuitive, since τ_{syn} is the exponential decay time constant for synaptic noise. For the OU process, Θ determines how quickly the process is drawn towards the equilibrium value μ. This can be described as a correlation parameter. It should be noted that the membrane potential u (described in Equation 4.15) in the high-conductance state is merely a linear transformation of the noise J^{syn}.

With this result we have completed the derivation of the equivalence between the Ornstein-Uhlenbeck process and the membrane potential of an LIF neuron stimulated with high-frequency synaptic Poisson input. In the next section we will use characteristics of OU processes to derive the firing statistics of LIF neurons.

4.4 The activation function of a leaky integrate-and-fire neuron

Nerve cells can be characterized by many properties to understand their functional role in networks. A particularly useful attribute of a neuron is its response behavior to stimulus in a network. A so-called *activation function* (or response function) is a measure of the spike output rate of a neuron receiving synaptic input. This measure is also applied to LIF neurons to characterize their output spike rate as a function of input current. We have argued at the beginning of this chapter that an analytic approach for AdEx neurons is difficult to achieve (*Hertäg et al.*, 2014). Here we will focus on a single LIF neuron without the AdEx extension and show how its activation function can be derived analytically.

4.4.1 Leaky integrate-and-fire neurons as stochastic computing units

We have shown that the dynamics of an LIF neuron can be described by an Ornstein-Uhlenbeck process. Since the membrane potential of the LIF neuron is stimulated by Poisson input, the firing behaviour of the neuron can also be formulated as a stochastic process. As a prerequisite for the description of the firing behaviour, we introduce a statistical interpretation of a neuron's activity.

We define the neuron state as a binary random variable $z_i(t) \in \{0, 1\}$ for neuron i as,

$$z_i(t) = \begin{cases} 1, & \text{neuron has spiked during } (t - \tau_{\text{ref}}, t) \\ 0, & \text{else} \end{cases} \tag{4.33}$$

This means that spikes are the defining property of a neuron state. In particular, the exact timing of a spike generation is crucial, since it encodes the transition between two states. This means that neuron i stays at $z_i = 0$ until it spikes at a time t_{spk}. The spike sets the neuron to $z_i = 1$ instantaneously for the duration of the refractory period $t_{\text{spk}} + \tau_{\text{ref}}$. After the refractory time, the state is reset to $z_i = 0$. This definition appears like an oversimplification of an LIF neuron's dynamics, as we propose only two possible states that encode a neuron's full state, instead of membrane potential $u(t)$ and conductance $g^{\text{syn}}(t)$. However, the membrane potential still plays an essential role, as it needs to exceed the threshold θ to trigger a spike event.

Our goal is to predict the activation function in terms of neuron states $z(t)$ by predicting the rate of threshold-crossings. To do this, we will use the properties of the OU process and transfer it to the LIF domain.

4.4.2 The first-passage times of the Ornstein-Uhlenbeck process

A fundamental property of the Ornstein-Uhlenbeck process is the so-called *first-passage time* (FPT). Starting at an initial state, the FPT is the time a stochastic process needs to cross a certain threshold for the first time since starting at the initial state. Due to the stochasticity of the system, the passage times follow a distribution, as can be seen

in Figure 4.3.

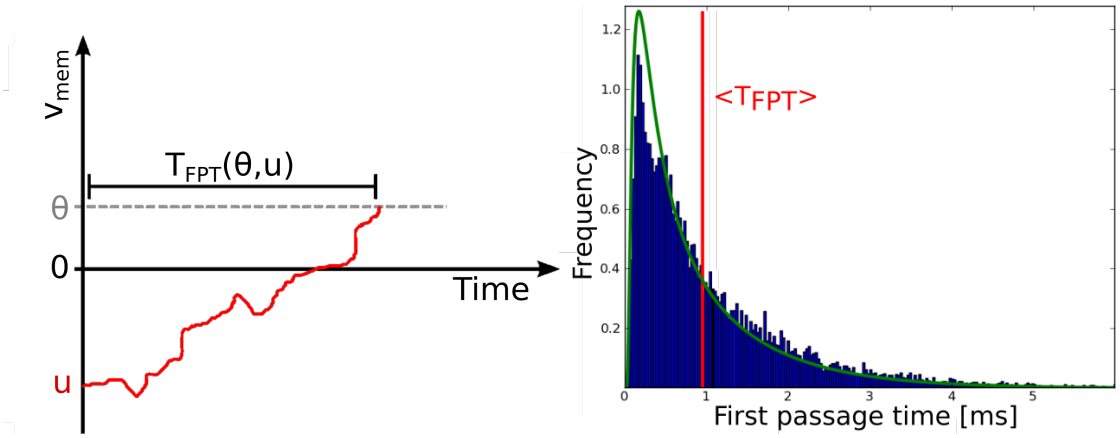

Figure 4.3: **(Left)** The first-passage time (FPT) measures the average duration of a process starting at u and crossing a threshold θ. **(Right)** The mean FPT can be calculated, but there is no closed-form description of its distribution. The average FPT is marked as a red line and the green curve is a numerical approximation (*Plesser and Tanaka*, 1997) of the distribution. The simulation (blue bars) has been performed for a quasi-ideal high-conductance state of an LIF neuron, with $\nu_{\text{exc}} = \nu_{\text{inh}} = 5 \cdot 10^4\,\text{Hz}$, $w_{\text{exc}} = w_{\text{inh}} = 0.003\,\mu\text{S}$, $\tau_{\text{eff}} = 10^{-2}\,\text{ms}$.

We will use the first-passage times to calculate the activation function of LIF neurons in the high-conductance state. Due to $\tau_{\text{eff}} \to 0$, we have identified membrane potential u (Equation 4.15) as the effective membrane potential u_{eff} (Equation 4.10). As soon as u_{eff} crosses the threshold θ, n spikes are triggered with following refractory periods $n \cdot \tau_{\text{ref}}$. The length of the spike sequence n depends on the excitation of the LIF neuron. Spikes are triggered as long as condition $u_{\text{eff}} > \theta$ is fulfilled. As soon as u_{eff} falls below the threshold, we calculate the FPT as the time it takes to get from the end of the spike sequence ($u_{\text{eff}} < \theta$) to the beginning of the next one, which again is triggered by ($u_{\text{eff}} > \theta$). For a finite time T_{sim}, this allows us to calculate the total number of output spikes N_{spk}. We can describe the output rate in terms of the probability to find the neuron in the active state $z = 1$, $p(z = 1)$,

$$p(z = 1) = \frac{N_{\text{spk}} \tau_{\text{ref}}}{T_{\text{sim}}} \quad . \tag{4.34}$$

This probability depends on the length of spike sequences. This equation needs to be adjusted to different spike sequence lengths n,

$$p(z = 1) = \frac{\sum_n P_n \cdot n \cdot \tau_{\text{ref}}}{\sum_n P_n \cdot (n\tau_{\text{ref}} + T_n)} \quad . \tag{4.35}$$

In the numerator, the $z = 1$ state duration during T_{sim} is considered. A sequence of n subsequent spikes causes a $z = 1$ duration of length $n \cdot \tau_{\text{ref}}$. Each sequence of length n has a probability to occur, P_n. Therefore, we sum over every possible sequence with n consecutive spikes and weight them with the probability of the occurrence of the sequence. In the denominator, both the duration of the $z = 1$ state and the subthreshold state, $z = 0$, is included. The time the membrane spends in the subthreshold regime ($u_{\text{eff}} < \theta$) is defined by the first-passage times between the spike sequence intervals, T_n. Just like the probabilities P_n, the first-passage times T_n depend on the length of the triggered spike sequence. Both these quantities are crucial for the calculation of $p(z = 1)$ and will be calculated in the following.

For the case $n = 1$, we need to calculate P_1 and T_1. As the beginning of the one-spike sequence, the membrane potential at time $t_0 = 0$, $u_0 := u(t_0)$, crosses the threshold ($u_0 \geq \theta$) and activates an action potential (see Figure 4.4). After the refractory time τ_{ref}, the membrane potential u_1 ($u_1 := u(t_0 + \tau_{\text{ref}})$) converges to the effective membrane potential u_{eff}. Since the LIF neuron is stimulated by Poisson input, the membrane potential follows a probability distribution. Then, the probability that only one spike occurs ($n = 1$), is given as

$$P_1 := \int_{-\infty}^{\theta} du_1 p(u_1|u_0 = \theta) \quad . \tag{4.36}$$

To consider a one-spike event, the membrane potential value is below the threshold, $u_1 \in (-\infty, \theta)$. The conditional probability density $p(u_1|u_0 = \theta)$ is a Gaussian which incorporates the condition that the neuron spikes at $t_0 = 0$ (*Burkitt*, 2006). The Gaussian can be given as

$$p(u(t)|u_0 = \theta) = \frac{1}{\sqrt{2\pi \cdot \sigma_t^2(t)}} \exp\left(\frac{(u - \mu_t(t))^2}{2\sigma_t(t)^2}\right) \quad . \tag{4.37}$$

The conditional distribution is governed by the time-dependent statistical quantities, μ_t and σ_t,

$$\mu_t(t) = \mu - (\mu - \theta) \exp\left(-\frac{t}{\tau_{\text{syn}}}\right) \quad , \tag{4.38}$$

$$\sigma_t^2(t) = \sigma^2(1 - \exp\left(-\frac{t}{\tau_{\text{syn}}}\right) \quad . \tag{4.39}$$

To calculate the first-passage time T_1, we need to calculate the integral

$$T_1 = \int_{-\infty}^{\theta} du_1 p(u_1|u_0 = \theta) \langle T_{\text{FPT}}(\theta, u_1)\rangle \quad . \tag{4.40}$$

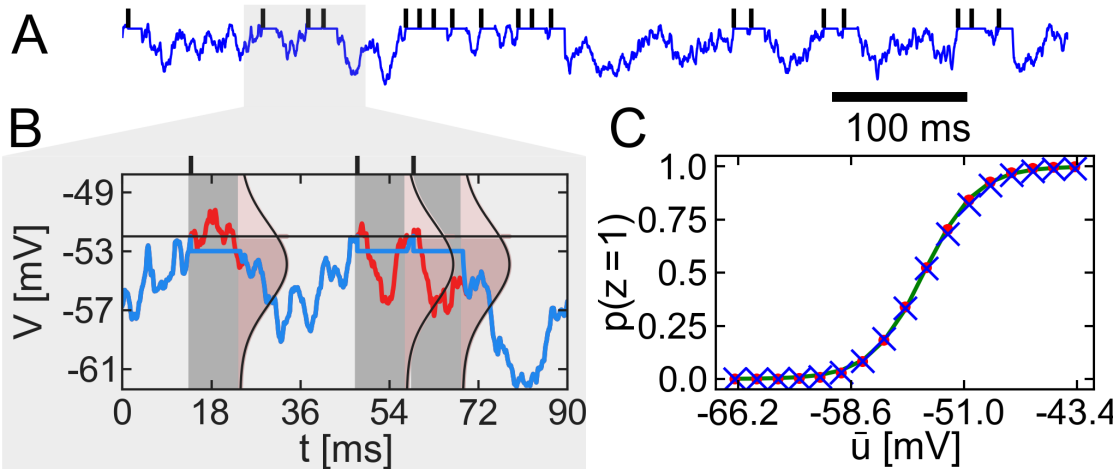

Figure 4.4: **(A)** The blue trace shows the membrane potential of an LIF neuron in the high-conductance state. The black bars show the output spike times of the LIF neuron. **(B)** In a cutout region of (A), we see the membrane potential u (blue), which activates three spikes. During the subthreshold activity, the membrane potential is identical to the effective membrane potential u_{eff} (red), which does not have a spike mechanism. As soon as the threshold θ is crossed, the neuron transitions into an activity state $z = 1$, while u_{eff} is suprathreshold. The refractory times are colored grey. At the end of each refractory time, the membrane potential converges to the effective membrane potential. The conditional probability distributions (purple) at the end of the refractory period indicate whether u is subthreshold or suprathreshold. In the subthreshold case ($\approx 25\,\text{ms}$), T_1 is calculated to estimate the next activation of a spike sequence. In this case, the next activation happens at $\approx 46\,\text{ms}$. In this case, a two-spike sequence is triggered, with the second spike at ($\approx 56\,\text{ms}$). After this spike sequence ($n = 2$), T_2 determines the time until the next activation. **(C)** The blue crosses show simulation results from a high-conductance neuron. The red dots represent the theoretical calculation of the activation function, as given in Equation 4.35. The green curve is a fitted logistic function σ. The simulation results are based on $T_{\text{sim}} = 10^4\,\text{ms}$ runtime and default parameters, as given in Appendix A.3. Figure is taken from *Petrovici et al.* (2013).

The quantity $T_{\mathrm{FPT}}(\theta, u_1)$ defines the first-passage time of the OU process (*Ricciardi and Sato*, 1988). For the membrane potential u_1 after the spike sequence, $T_{\mathrm{FPT}}(\theta, u_1)$ is the average time it takes to cross the threshold θ again and activate the next spike sequence (Figure 4.3). Since u_1 follows a distribution, we need to consider each membrane potential after the refractory period (Figure 4.4). Therefore, the integral in Equation 4.40 includes all subthreshold membrane potentials, $u_1 \in (-\infty, \theta)$.

We will not provide the calculations for spike sequences with $n > 1$. The corresponding first-passage times and probabilities, T_n and P_n, are different for each n and need to be calculated seperately. In general, an n-spike event occurs with a probability of P_n and is followed by a first-passage time of T_n until the next spike sequence (Figure 4.4). The calculations of the n-order spike-events is done recursively and will not be described here. A very general treatment and calculation of this is detailed in *Petrovici et al.* (2015a); *Petrovici* (2016); *Petrovici et al.* (2016). In general, spike-sequence length n can become arbitrarily high if the mean membrane potential μ is high enough. The membrane potential after an n-spike sequence is then $u_n = u(n \cdot \tau_{\mathrm{ref}})$. We then need to calculate all FTPs T_n to ensure a sufficiently accurate activation function. In general, the longer the spike sequence n, the smaller its occurrence probability P_n becomes. The first-passage times T_n become longer with increasing n as well.

We have now demonstrated the conceptual approach to calculate the activation function of an LIF neuron in the high-conductance state. The accuracy of Equation 4.35 can still be improved. For LIF neurons in a high-conductance state, Equation 4.35 provides a very good approximation of the activation function. For finite effective time constants τ_{eff}, the membrane potential u is slower than the effective membrane potential u_{eff}. After a spike sequence, this time difference becomes negligible compared to T_n. But in-between spikes in a spike sequence, additional down-states occur between the end of a refractory period and the next spike within the sequence. The calculation of the full activation function has been performed in collaboration with Mihai Petrovici. We will only show the final result here, as a detailed discussion can be found in *Petrovici et al.* (2013, 2015a); *Petrovici* (2016); *Petrovici et al.* (2016).

The full activation function can be given as

$$p(z=1) = \frac{\sum_n P_n \cdot n \cdot \tau_{\mathrm{ref}}}{\sum_n P_n \cdot \left(n\tau_{\mathrm{ref}} + \sum_{k=1}^{n-1} \overline{\tau_k^{\mathrm{b}}} + T_n \right)} \quad . \tag{4.41}$$

Compared to Equation 4.35, the above equation not only respects the first-passage times T_n, but also includes the inter-spike downtimes $\overline{\tau_k^{\mathrm{b}}}$. These downtimes become significant when the subthreshold membrane potential is not fast enough to converge to the effective membrane potential, which is still suprathreshold. As a result, the membrane potential u is too slow to trigger the spikes in an n-spike sequence. This introduces additional down-states in-between spikes. In an n-spike sequence, the down-state between

spikes k and $k+1$ is given as $\overline{\tau_k^{\mathrm{b}}}$. It is the time the membrane needs to cross the distance $\rho - \theta$ in-between spikes k and $k+1$. This is summed for all $n-1$ spike intervals. Since we include these additional down-states into the calculation of the activation function, Equation 4.41 can be applied for parameter sets that do not rely on the high-conductance state. Specifically, it improves the accuracy of the activation function for slow membrane potentials, i.e., high τ_{eff}. This becomes crucial when the membrane potential needs to rise from ρ to θ to switch states $z = 0 \to z = 1$ during a spike sequence. Then τ_k^{b} is the additional time the membrane needs to cross the distance $\rho - \theta$ in-between spikes k and $k+1$. This quantity becomes particularly important for the parameter configuration $\rho \ll \theta$, because for long distances $\rho - \theta$ the membrane needs longer to reach θ to initiate the next spike in the sequence.

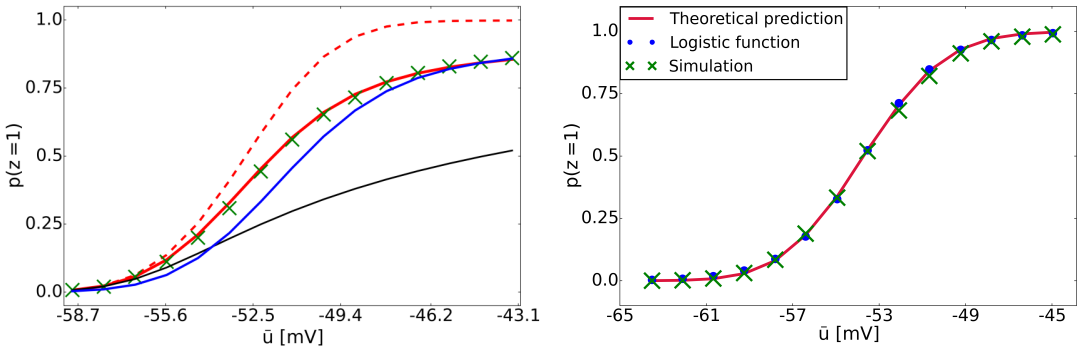

Figure 4.5: **(Left)** The activation function of an LIF neuron that is not in a high-conductance state. The green crosses show simulation results and are described well by Equation 4.41. The dashed red activation curve assumes high conductance ($\tau_k^{\mathrm{b}} \approx 0$) and therefore overestimates the activity. This highlights the importance of the τ_k^{b} rise times for parameter ranges outside of high-conductance dynamics. The black curve shows a calculation from *Brunel and Sergi* (1998) and assumes a very small synaptic time constant. The blue curve assumes only slow changes of the membrane potential (*Moreno-Bote and Parga*, 2004) and also has no concept of spike sequences. **(Right)** A very important attribute of an LIF neuron with high conductance is its symmetric activation function. The prediction (Equation 4.41) as well as simulation results show that it can be fitted to a logistic function σ (Equation 4.42).

We can see the comparison between our prediction in Equation 4.41 to known predictions from literature (*Brunel and Sergi*, 1998; *Moreno-Bote and Parga*, 2004) and simulation results. In both existing calculations of the LIF activation function, in *Brunel and Sergi* (1998) and *Moreno-Bote and Parga* (2004), there is no concept of spike sequences. In *Brunel and Sergi* (1998), the activation function can only be applied to parameter ranges where $\tau_{\mathrm{syn}} \ll \tau_{\mathrm{ref}}$ and $\tau_{\mathrm{syn}} \gg \tau_{\mathrm{eff}}$. The description does not describe firing statistics caused by strong autocorrelations in the membrane potential due to a long synaptic time constant τ_{syn}. As a result, the activation function shows deviations

for strong synaptic inputs. In *Moreno-Bote and Parga* (2004), the activation function calculation was provided for $\tau_{\text{syn}} \gg \tau_{\text{eff}}$. Due to this relationship between the time constants, the effective membrane potential is governed by the synaptic time constant only. Since τ_{syn} is large, the membrane potential reacts slowly to changes in the synaptic current. This causes an underestimation of firing activity in the activation function, as seen in Figure 4.5. The prediction of the activation function given in Equation 4.41 is valid for every parameter regime, including the ones described in *Brunel and Sergi* (1998); *Moreno-Bote and Parga* (2004).

4.5 Conclusion: significance of the activation function

The mathematical analysis of an LIF neuron in the Poisson-induced high-conductance state description enabled us to derive an essential characteristic of this neuron model, namely its activation function. Also, we introduced a binary random variable that represents the neuron's activity state, $z(t) \in \{0, 1\}$. We used the average first-passage times of the Ornstein-Uhlenbeck process, T_{FPT}, to accurately predict the activation function (Figure 4.5) as the probability of the neuron being active, $p(z = 1) \in (0, 1)$. We consider the neuron active ($z = 1$) during its refractory time in the sense that it transmits synaptic activity within this time frame. For our probabilistic networks in the next chapter, it is essential that the activation function is symmetric in the high-conductance state (see Figure 4.5). We can fit the LIF response curve to a logistic function $\sigma(x)$,

$$\sigma(x) = \frac{1}{1 + \exp(-x)} \quad . \tag{4.42}$$

We will see in the next chapter that this attribute is crucial for LIF networks to perform inference.

5 Probabilistic computing with leaky integrate-and-fire neurons

We have achieved a stochastic formulation of single LIF neuron states driven by stochastic synaptic inputs. We will see that this formulation provides us a rigid theory framework to construct stochastic inference networks. This approach will differ from our L23 network evaluation in Chapter 3 insofar as the statistical formulation provides a concept of predicting activity of single neuron states. Therefore, introducing free parameters is not necessary. Another crucial difference is that we will not use continuous population rates for encoding, as population rates are not timing-sensitive. Also, population rates are quantities that reduce stochasticity in the system, since they constitute averages over single neuron's activity states. In such a neural system, stochasticity is not considered a mean of computation, but is rather a side-effect that arises from thalamo-cortical connectivity patterns (*Bruno and Sakmann*, 2006).

Our more probabilistic interpretation of neuron states from the previous chapter is consistent with recent findings that show evidence for optimization of statistical computation in the brain (*Fiser et al.*, 2010; *Berkes et al.*, 2011). Such models are supported by measurements of in-vivo activity showing wide ranges of trial-to-trial variability on single neuron level. Physiological reasons behind such variability lie in the inherently stochastic manner of synaptic interaction. The neurotransmitter release and diffusion of these compounds within the synaptic cleft (*Faisal et al.*, 2008; *Yang and Xu-Friedman*, 2013), as well as the depletion on the postsynaptic membrane are stochastic processes. These findings show a certain disparity between neural processes where stochasticity is present and, on the other hand, neuron models with deterministic spiking mechanisms where noise is merely a side-effect or even an obstacle. Although it is not yet clear to what extent noise constitutes a computational component or hindrance in the cortical circuitry, it certainly is present during cortical information processing (*López*, 2001).

Our goal in this chapter is to build a network model that links both domains - using the mechanistic LIF model with Poisson-induced stochasticity to perform stochastic inference. Hence, we can define a network consisting of N LIF neurons with a random state vector $\boldsymbol{z}(t) = (z_1(t), \ldots, z_i(t), \ldots, z_N(t))$, encoding 2^N states. Under certain conditions, the dynamics of such a network represent a well-defined probability distribution $p(\boldsymbol{z})$. We can then interpret the network activity as *sampling* from $p(\boldsymbol{z})$. We will refer to the corresponding network dynamics as *neural sampling*.

From a purely computational perspective, such an approach is vastly more efficient than population-based coding. The reason for this are the required neural resources for encoding information. For instance, in the L23 network, encoding of attractor states was carried out by several minicolumn populations. Each minicolumn incorporated hundreds

of AdEx neurons to represent an attractor state. In the neural sampling framework, only N LIF units are necessary to sample from 2^N states. In the following sections we will motivate and describe this framework.

5.1 Neural computability condition in recurrent neural networks

To define a probabilistic representation of neural activity, we have introduced the single neuron random variables $z_i(t)$. We can also describe each neuron's average activity as an activation function $p(z = 1)$. These attributes so far characterized single units. For a network representation of random variables $z_i(t)$, we need a model to implement interaction between the neurons. Since LIF neurons mediate interaction via PSPs, this model must link membrane potential changes to the probability distribution $p(\boldsymbol{z})$ of the random variable vector $\boldsymbol{z}(t)$. Such a relationship between probabilistic states and membrane potentials of neurons has been established in *Buesing et al.* (2011) and will be referred to as the *neural computability condition* (NCC):

$$v_k = \log\left(\frac{p(z_k = 1|z_{\setminus k})}{p(z_k = 0|z_{\setminus k})}\right) \tag{5.1}$$

It states that the membrane potential v_k of neural unit k can be defined as a logarithmic ratio (log-odds) between the conditional probability of unit k being in an active ($z_k = 1$) and inactive ($z_k = 0$) state. This equation allows to derive the membrane potential v_k given the conditional probability distribution function $p(z_k|z_{\setminus k})$.

The authors of *Buesing et al.* (2011) have developed a neural sampling framework for an *abstract neuron model* (ANM) and have shown that, under the neural computability condition, neural network states $\boldsymbol{z}(t)$ can be interpreted as *samples* from the target distribution $p(\boldsymbol{z})$. More specifically, the underlying sampling mechanism of the ANM is a form of *Markov-Chain Monte Carlo* sampling. We will explain the basics of the ANM in the next section in more detail.

In principle, it is possible to insert any well-defined probability distribution $p(\boldsymbol{z})$ into Equation 5.1 and derive the corresponding membrane potential v_k. We will focus on a particular distribution, which is well-studied, the *Boltzmann distribution*,

$$p(\boldsymbol{z}) = \frac{1}{Z}\exp\left(-\frac{E(\boldsymbol{z}^\mathsf{T})}{\mathrm{kT}}\right) \overset{(kT)^{-1}=1}{=} \frac{1}{Z}\exp\left(\frac{1}{2}\boldsymbol{z}^\mathsf{T}\boldsymbol{W}\boldsymbol{z} + \boldsymbol{z}^\mathsf{T}\boldsymbol{b}\right) \quad ,$$

$$= \frac{1}{Z}\exp\left(\frac{1}{2}\sum_{i,j} W_{ij}z_iz_j + \sum_i b_iz_i\right) \quad , \tag{5.2}$$

$$Z = \sum_{\boldsymbol{z}}\exp\left(-\frac{E(\boldsymbol{z})}{\mathrm{kT}}\right) \overset{(kT)^{-1}=1}{=} \sum_{\boldsymbol{z}}\exp\left(\frac{1}{2}\boldsymbol{z}^\mathsf{T}\boldsymbol{W}\boldsymbol{z} + \boldsymbol{z}^\mathsf{T}\boldsymbol{b}\right)$$

$$= \sum_{\boldsymbol{z}}\exp\left(\frac{1}{2}\sum_{i,j} W_{ij}z_iz_j + \sum_i b_iz_i\right) \quad . \tag{5.3}$$

This distribution was originally formulated by Ludwig Boltzmann to describe the state distribution of gas particles in a thermal equilibrium. It was later reformulated by Josiah Willard Gibbs (*Gibbs*, 1961). The energy of the system E depends on the states $\boldsymbol{z}$, given a temperature T, where k is the Boltzmann constant.

Although this description was established in the context of statistical physics, it has been generalized and widely adopted to problems of information processing. For our purposes, we set the *inverse temperature* $\beta = \frac{1}{\mathrm{kT}} = 1$ in Equations 5.2 and 5.3 and will review this parameter in the next chapter. Most prominently, it describes recurrent neural networks consisting of computing nodes that interact with each other.

Such networks are widely known as *Boltzmann machines* (BMs) and are used for a vast range of optimization problems[1] (*Geman and Geman*, 1984; *Ackley et al.*, 1985; *Hinton and Salakhutdinov*, 2006).

Our goal will be to use the Boltzmann distribution (as defined in Equations 5.2, 5.3) to build Boltzmann machines with LIF neurons by using the neural computability condition (NCC, Equation 5.1). At first glance, this seems to be an odd choice for neural modeling, since BMs originate from physics and lack biological context. Their connection matrix is symmetric ($W_{ij} = W_{ji}$) and the NCC in Equation 5.1 has no biological foundation. But from a practical perspective, they are well-studied, efficient and sufficiently versatile to be applied to many different problems in pattern classification and generative tasks (*Sutskever et al.*, 2009; *Nair and Hinton*, 2010) on big data sets. These types of stochastic recurrent networks were fundamental for many state-of-the-art network models and significantly progressed the field of machine learning.

For this reason these functional networks can be used to study the significance of spiking neurons with biology-inspired noise sources on solving inference problems. We will develop LIF-based BMs and compare their performance to conventional BM implementations known from machine learning.

[1] Boltzmann machines are similar to *Hopfield nets*, which were described in *Hopfield* (1982) and popularized the idea of using recurrent network descriptions for optimization problems.

5.2 Leaky integrate-and-fire networks as Boltzmann machines

There are two principal attributes that LIF neurons need in order to function as sampling Boltzmann machines. Both attributes can be derived from the NCC in Equation 5.1. The first attribute concerns the interaction of LIF neurons, which has to encode joint states like in conventional Boltzmann machines. The second attribute is the single LIF neuron firing statistics, induced by Poisson noise. These statistical attributes must be implemented accordingly.

To obtain both attributes, we will first derive the membrane potential v_k from the NCC under the condition that $p(\boldsymbol{z})$ is a Boltzmann distribution. If we insert the Boltzmann distribution from 5.2 into the RHS of 5.1, the exponential function disappears and the normalization Z cancels out. We then obtain

$$\log\left(\frac{p(z_k=1|z_{\setminus k})}{p(z_k=0|z_{\setminus k})}\right) = \sum_i W_{ik}z_i + b_k \tag{5.4}$$

According to the NCC (Equation 5.1), we can identify the RHS of Equation 5.4 with the membrane potential for v_k,

$$v_k = \sum_i W_{ik}z_i + b_k \quad . \tag{5.5}$$

The membrane potential v_k, as well as all the other components of this equation are unitless. We will explain the membrane potential of BM units after deriving the activation function $p(z_k=1|z_{\setminus k})$, which describes the activity of a single BM unit. For the derivation, we use the property $1 = p(z_k=1|z_{\setminus k}) + p(z_k=0|z_{\setminus k})$ and calculate $p(z_k=1|z_{\setminus k})$ from Equation 5.4 explicitly,

$$\log\left(\frac{p(z_k=1|z_{\setminus k})}{1-p(z_k=1|z_{\setminus k})}\right) = \sum_i W_{ik}z_i + b_k \tag{5.6}$$

$$p(z_k=1|z_{\setminus k}) = \exp\left(\sum_i W_{ik}z_i + b_k\right)\left[1-p(z_k=1|z_{\setminus k})\right] \tag{5.7}$$

$$p(z_k=1|z_{\setminus k}) = \frac{\exp\left(\sum_i W_{ik}z_i + b_k\right)}{1+\exp\left(\sum_i W_{ik}z_i + b_k\right)} = \frac{1}{1+\exp\left(-\sum_i W_{ik}z_i - b_k\right)} \tag{5.8}$$

$$p(z_k=1|z_{\setminus k}) \overset{\text{Eq. 5.5}}{=} \frac{1}{1+\exp(-v_k)} \tag{5.9}$$

We can now elaborate on the results. First, we take a look at the membrane potential v_k. This "abstract" potential consists of two terms and is the membrane of the abstract neuron model (ANM) we referred to at the beginning of this section. The first term is a

weighted sum of network states $\sum_i W_{ik} z_i$. This part describes the interaction between network units. The abstract membrane of neuron k is incremented by W_{ik} if presynaptic unit i is active ($z_i = 1$) and connected to unit z_k ($W_{ik} \neq 0$). We will refer to W_{ik} as *Boltzmann weights*, which are elements of the symmetric coupling strength matrix $\boldsymbol{W}$ in a Boltzmann machine. The weights multiplied by the state can be regarded as a sum of rectangular kernels. This means that the mediated interaction by W_{ik} is a constant PSP that is added to the membrane potential v_k for a certain period of time. Additionally, the membrane potential v_k has a fixed value b_k, which is referred to as *bias*. This quantity adds a constant shift to the membrane potential. Although v_k has no evident similarities to the LIF neuron model (Equation 2.1), it models a linear sum of weights as a consequence of synaptic interaction. This is an important correspondence between both models, which we will use later to define LIF Boltzmann machines.

The second result of the derivation, $p(z_k = 1|z_{\setminus k})$, is a sigmoidal function $\sigma(v_k) = 1/\left[1 + \exp(-v_k)\right]$, with v_k being the abstract, unitless membrane potential. It represents the probability of a network unit z_k being active ($z_k = 1$), given all the other network unit states $v_{\setminus k}$. This is consistent with our result from Equation 4.42, where we have shown that the symmetric LIF activation function can be identified as a logistic function. This means that, in principle, a LIF neuron with membrane potential u_k can be configured to exhibit firing statistics according to the activation curve $\sigma(u_k)$.
Therefore, if we want to run LIF networks as Boltzmann machines, we need to construct a mapping scheme from LIF parameters to Boltzmann parameters,

$$(E_\mathrm{l}, E_\mathrm{i}^\mathrm{rev}, w_i, \nu_i, \ldots) \longleftrightarrow (\boldsymbol{W}, \boldsymbol{b}) \quad . \tag{5.10}$$

To develop such translation rules, we will take a closer look at the ANM from *Buesing et al.* (2011). This model will serve as a reference point on our way to define LIF-based Boltzmann machines.

The membrane potentials of an ANM network are as stated in Equation 5.5. In every point in time t, the membrane potential value $v_k = \sum_i W_{ik} z_i + b_k$ is used to calculate the spiking probability as the activation function $\sigma(v_k)$ (shown in Equation 5.9). The bias of neuron k equates to a constant offset probability towards $z_k = 1$ or $z_k = 0$, with $b_k > 0$ or $b_k < 0$, respectively. The interaction weights W_{ik} increase or decrease the spiking probability.

If neuron k is inactive at time t, the probability $\sigma(v_k)$ determines whether it will transition into state $z_k = 1$ or remain inactive. The higher the membrane potential v_k, the more likely it switches to $z_k = 1$. If the neuron is activated, it remains in state $z_k = 1$ for a fixed period of time, which we will call τ_on. This duration has two roles in the model. Firstly, it can be understood as a refractory period that lasts for a certain number of time steps after the neuron switched to $z_k = 1$. Secondly, in this time the unit k transmits a rectangular PSP to its postsynaptic neighbor j with height W_{ij}. After this refractory period τ_on, the PSP ends and the probability $\sigma(v_k)$ is evaluated again to determine a state change.

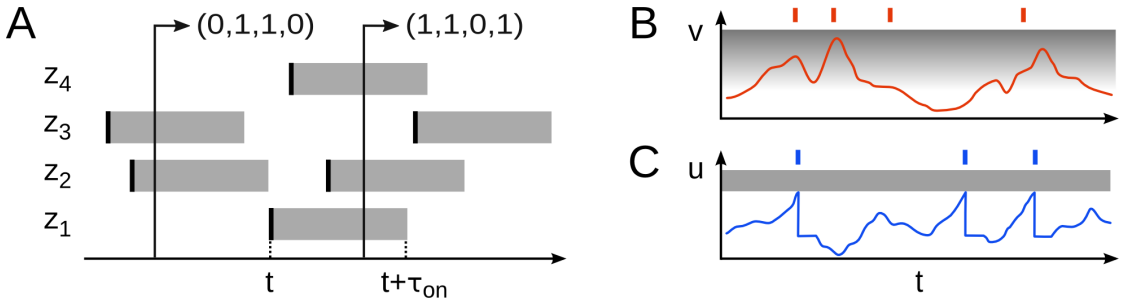

Figure 5.1: **(A)** We see the encoding of four states (z_1, z_2, z_3, z_4). The grey area marks periods of activity with $z_i = 1$, triggered by action potentials (black bars). The binary states can be read out at any time, providing an approximation of the underlying distribution with 2^4 network states. **(B)** The schematic shows the membrane potential from the abstract neuron model in *Buesing et al.* (2011). The spikes are triggered according to the logistic firing probability $\sigma(v)$. The higher the membrane potential v, the more likely an action potential becomes. Note that the membrane potential is unbounded and can rise to arbitrarily high levels. **(C)** The LIF spiking mechanism is triggered if potential u exceeds the threshold ρ, which is colored grey. In contrast to the abstract model shown in (B), the mechanism is deterministic. The grey bar illustrates the upper bound of the threshold. Figure is taken from *Petrovici et al.* (2013).

The authors in *Buesing et al.* (2011) have shown that this mechanism implements MCMC sampling from a target Boltzmann distribution with network parameters $(\boldsymbol{W}, \boldsymbol{b})$. Further, they have proven that the distributions sampled by the network converge to the target distribution on an exponential time scale. A more detailed and technical description of these network dynamics can be found in *Buesing et al.* (2011); *Petrovici* (2016).

5.2.1 From sampling with abstract neurons to leaky integrate-and-fire networks

A conceptual difference between the ANM and LIF neurons is the entirely different way of embedding stochasticity in the network. In the LIF model, we impose biology-inspired temporal Poisson input onto a deterministic spiking mechanism. In the ANM, the spiking probability $\sigma(v_k)$ is compared in each discrete time step with a random number, determining whether to switch states in the state vector $\boldsymbol{z}$. The contrast between both models is shown in Figure 5.1. Despite the conceptual discrepancy between discrete time MCMC sampling and an LIF neuron driven by Poisson input, we will show that it is possible to use LIF neurons with average membrane potentials $\bar{u}_k$ as Boltzmann units

with a spiking probability $\sigma(u_k)$. We demand this condition for each LIF neuron,

$$\sigma(\bar{u}_k) \stackrel{!}{=} \sigma(v_k) \quad . \tag{5.11}$$

Note that for the LIF domain we use the average membrane potential $\bar{u}_k$, since the stochasticity is present as temporal noise, which follows a distribution over time. There is no state transition probability for the LIF neuron since its membrane potential is deterministic in a given time step. The time-averaged membrane potential and the firing statistics of LIF neurons have been calculated in Chapter 4.

For every average membrane potential $\bar{u}$ there is a well-defined total average firing probability $\sigma(\bar{u}_k) = p(z_k = 1)$ in the LIF domain. We define a mean membrane potential $\bar{u}_k^0$, which is characterized by $\sigma(\bar{u}_k^0) = \frac{1}{2}$. For this specific average membrane potential the firing activity is exactly at the inflection point of $\sigma(\bar{u}_k)$. This membrane potential implies the bias $b_k = 0$ in the LIF domain. Since both the ANM domain and the LIF domain have different scales, we define a scaling parameter α to rescale the voltage-based LIF membrane potential into the dimensionless BM domain. We can now express the bias b_k in terms of $\bar{u}_k$:

$$b_k = \frac{\bar{u}_k - \bar{u}_k^0}{\alpha} \quad . \tag{5.12}$$

Using Equation 5.12, we can now equate the firing probabilities σ for both domains:

$$\sigma(b_k) = \sigma(\frac{\bar{u}_k - \bar{u}_k^0}{\alpha}) = \sigma(\bar{u}_k) \tag{5.13}$$

In the activation function the parameter α determines the slope and $\bar{u}_k^0$ is a constant shift of the logistic curve. Both parameters, α and $\bar{u}_k^0$, can be predicted by calculating the activation function for a LIF neuron in Equation 4.41.

The firing probability in Equation 5.13 is identical in both domains. The interaction between the Boltzmann units is not included in the translation of the sigmoid $\sigma(b_k)$. To implement the neuron-to-neuron interaction in the LIF domain, we aim to approximate the PSP shapes of the ANM as closely as possible, as their network evolution is guaranteed to converge to the target distribution.

The PSPs of the ANM are rectangular and, in contrast to physical systems, they are mediated without delay (Figure 5.2). When neuron i switches into state $z_i = 1$, the membrane of the postsynaptic neuron j is incremented instantaneously by W_{ij} for a duration of τ_{on}.

The crucial point here is that synaptic transmission is carried out for a fixed duration and is deactivated after τ_{on}, unlike LIF interaction. In the LIF domain, the synaptic interaction is mediated by exponentially-shaped PSPs. If the synaptic time constant τ_{syn} is not significantly smaller than the refractory time constant τ_{ref}, then the exponential tail of the PSP gives a contribution to the membrane potential even after the refractory

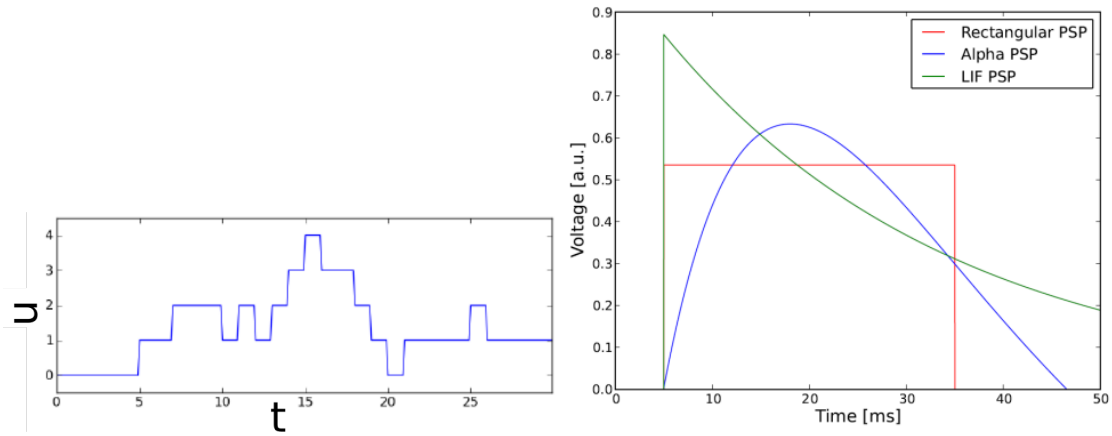

Figure 5.2: **(Left)** We see an exemplary membrane potential trace of an abstract neuron. The potential is essentially a superposition of synaptic weights, as described in Equation 5.5. **(Right)** The plot shows a comparison between the PSP shapes of a LIF neuron and an abstract neuron. The refractory time has been chosen $\tau_{\text{on}} = \tau_{\text{ref}} = 30\,\text{ms}$. The rectangular ANM PSP (red) is induced instantaneously after synaptic transmission has a fixed length τ_{ref}, while the LIF curve (green) declines exponentially with time constant τ_{syn}. In contrast to the ANM PSP, the LIF PSP continues after τ_{ref} and contributes even after the refractory period to the membrane potential. Figures is taken from *Petrovici* (2016).

period. This is an important difference to the rectangular PSPs of the ANM that bears important consequences.

When approximating the PSP shape of the ANM, we can adjust several free LIF parameters. The most important ones are refractory time τ_{ref} and synaptic transmission time constant τ_{syn}. We will propose a method to map Boltzmann interaction to LIF neurons motivated by following arguments:

1. To compare the membrane potential time scales between LIF and ANM neurons, we set a reference time scale. This time scale is the duration in which the network unit is active, which is the $z_k = 1$ state. Since the interaction happens during this refractory time for both models, we set both activity durations equal, $\tau_{\text{ref}} = \tau_{\text{on}}$.

2. In the ANM, the refractory phase τ_{on} is also the duration of synaptic signal transmission. We transfer this relationship of refractoriness and synaptic transmission into the LIF domain by setting $\tau_{\text{syn}} = \tau_{\text{ref}}$. This relationship is a heuristic approximation, as the synaptic interaction for LIF neurons is not cut off, but declines exponentially. By performing this approximation, we assume that the relevant interaction happens during the time frame of the refractory period, $(t_{\text{spk}}, t_{\text{spk}} + \tau_{\text{ref}})$.

3. Having fixed $\tau_{\text{syn}} = \tau_{\text{ref}}$, the final step to achieve a mapping between both domains is to equate the impact of synaptic interaction. The area under the PSP is often used as a measure of interaction impact.

We can formalize the last condition,

$$W_{ij} \cdot \alpha \cdot \tau_{\text{ref}} \stackrel{!}{=} \int_0^{\tau_{\text{ref}}} \text{PSP}_j^{\text{LIF}}(t)dt \quad . \tag{5.14}$$

The PSP time course of the LIF neuron, $PSP(t)$, can be approximated in the high-conductance state. A solution was given in *Bytschok* (2011) and reads

$$\text{PSP}_j^{\text{LIF}}(t) = \frac{w_{ij}(E_{\text{j}}^{\text{rev}} - \bar{u}_j)\tau_{\text{syn}}}{g_{\text{tot}}(\tau_{\text{syn}} - \tau_{\text{eff}})} \left[\exp\left(-\frac{t}{\tau_{\text{syn}}}\right) - \exp\left(-\frac{t}{\tau_{\text{eff}}}\right)\right] \quad . \tag{5.15}$$

Note that the difference-of-exponential shape of the LIF PSP becomes exponential as $\tau_{\text{eff}} \to 0$ in the high-conductance state. We can insert Equation 5.15 into Equation 5.14 and set $\tau_{\text{ref}} = \tau_{\text{syn}}$ as argued above. This yields

$$W_{ij} = \frac{1}{\alpha C_{\text{m}}} \frac{w_{ij}(E_{\text{j}}^{\text{rev}}) - \bar{u}_j}{\frac{1}{\tau_{\text{syn}}} - \frac{1}{\tau_{\text{eff}}}} \cdot \left[\frac{1-e}{e} - \frac{\tau_{\text{eff}}}{\tau_{\text{syn}}}\left(e^{-\frac{\tau_{\text{syn}}}{\tau_{\text{eff}}}} - 1\right)\right] \quad . \tag{5.16}$$

This represents the translation between Boltzmann weights W_{ij} and LIF weights w_{ij}. The weight translation itself, just like the parameter translation for the bias in Equation 5.12, can be done in both directions. Starting from a LIF neuron, the translation of

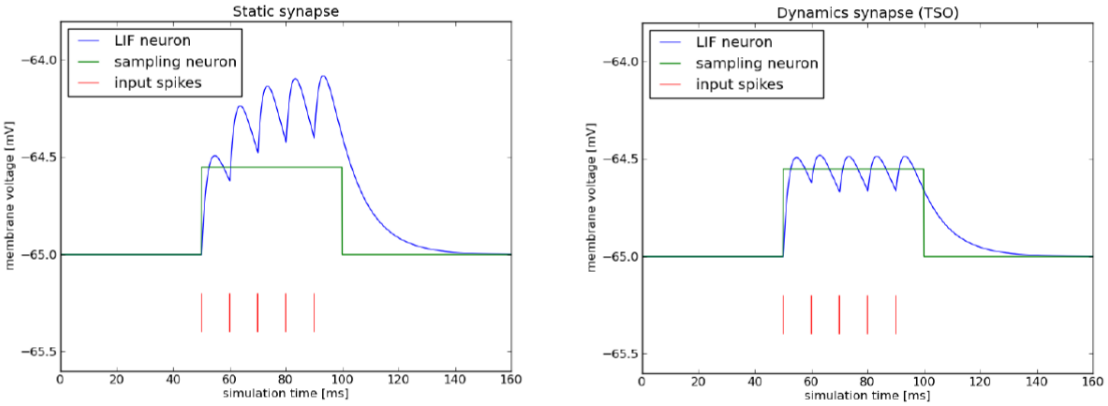

Figure 5.3: On the left plot we see the membrane potential during a succession of presynaptic spikes. The membrane potential is amplified because the tail-ends of the exponential residuals sum up after τ_{ref}. In this case, the approximation made in Equation 5.16 does not hold. In the right plot we see the same sequence of PSPs with the TSO mechanism (see Section 2.4) implemented. The recovery time constant τ_{rec} can be chosen such that the origin PSP amplitude is recovered after precisely $\tau_{\text{ref}} = \tau_{\text{rec}}$. This mechanism improves the quality of LIF sampling in general, and in particular in cases with high spiking probabilities. Figure is taken from *Petrovici* (2016).

the biases and weights towards the BM domain can be done directly by using Equations 5.12 and 5.16. Translating a BM to a LIF neuron, however, involves choosing LIF parameters deliberately, as the RHS of Equation 5.16 depends on numerous LIF parameters. We will show that, following certain principles, many LIF parameter configurations can be used to achieve accurate sampling from Boltzmann distributions.

5.2.2 Synaptic short-term plasticity in sampling

In the previous section we have shown a method to approximate a single LIF PSP to rectangular PSPs of the ANM (see Figure 5.2). One major systematic deviation results from the tail-end of the LIF PSP after τ_{ref}. This deviation becomes significant when persisting synaptic input occurs at high spiking probabilities. In this case, the exponential tails of LIF PSPs will sum up after the refractory period and add to subsequent PSPs. This amplifies the synaptic interaction, as can be seen in the left plot of Figure 5.3. We can alleviate this effect by introducing short-term depression into synapses between LIF Boltzmann units. The underlying Tsodyks-Markram mechanism was described in Section 2.4. The mechanism temporarily decreases the synaptic efficacy in network connections by modelling neurotransmitter consumption. The recovery time constant τ_{rec} is chosen such that the original PSP height is recovered after the refractory time, hence $\tau_{\text{rec}} = \tau_{\text{ref}}$. Implementing this biological mechanism into the LIF-based Boltzmann machine is a heuristic method that accommodates to strong interaction between two

LIF neurons in a BM.

To quantify the performance of LIF-based Boltzmann machines, we will introduce a measure to compare the performance of probabilistic LIF networks with conventional sampling implementations.

5.3 Sampling with leaky integrate-and-fire neurons

To assess the inference capabilities of LIF-based Boltzmann machines, we need a method to evaluate how well the LIF network samples from a Boltzmann distribution. We can evaluate the performance measuring how close the LIF-sampled distribution $p_{\mathrm{N}}(\boldsymbol{z})$ is to a target BM distribution, $p_{\mathrm{T}}(\boldsymbol{z})$.

A commonly used distance measure is the so-called *Kullback-Leibler divergence* (DKL). It quantifies the difference between two probability distributions, $p(\boldsymbol{z})$ and $q(\boldsymbol{z})$,

$$\mathrm{D}_{\mathrm{KL}}\left(p(\boldsymbol{z}) \,||\, q(\boldsymbol{z})\right) = \sum_{\boldsymbol{z}} p(\boldsymbol{z}) \log\left(\frac{p(\boldsymbol{z})}{q(\boldsymbol{z})}\right) \quad . \tag{5.17}$$

This measure has several important characteristics that we point out:

- The DKL is positive-semidefinite ($\mathrm{D}_{\mathrm{KL}}\left(p(\boldsymbol{z}) \,||\, q(\boldsymbol{z})\right) \geq 0$) and zero for identical distributions, $p(\boldsymbol{z}) \equiv q(\boldsymbol{z})$.
- It is not symmetric ($\mathrm{D}_{\mathrm{KL}}\left(p(\boldsymbol{z}) \,||\, q(\boldsymbol{z})\right) \neq \mathrm{D}_{\mathrm{KL}}\left(q(\boldsymbol{z}) \,||\, p(\boldsymbol{z})\right)$), therefore not a metric. The order of arguments can be chosen freely, but needs to be specified when comparing results.
- The DKL requires to sum over all states $\boldsymbol{z}$. This becomes intractable for a large number of neurons N, as the state size grows with 2^N. Because of this limitation, we will apply the DKL only for small networks.
- Due to finite sampling time, there can be valid network states $\boldsymbol{z}_j$ that the network never visits, e.g., $q(\boldsymbol{z}_j) = 0$. This results in a $\log\left[q(\boldsymbol{z}_j) = 0\right]$ calculation. We avoid such cases by interpreting the theoretical target probability distribution as the second distribution in Equation 5.17, $q(\boldsymbol{z})$. Due to the exponential function inside the Boltzmann distribution, the probability is never zero.

Due to the second point, we will use the DKL to study small networks, where we compare both, the network-sampled distribution $p_{\mathrm{N}}(\boldsymbol{z})$, and the theoretical target distribution $p_{\mathrm{T}}(\boldsymbol{z})$,

$$\mathrm{D}_{\mathrm{KL}}\left(p_{\mathrm{N}}(\boldsymbol{z}) \,||\, p_{\mathrm{T}}(\boldsymbol{z})\right) = \sum_{\boldsymbol{z}} p_{\mathrm{N}}(\boldsymbol{z}) \log\left(\frac{p_{\mathrm{N}}(\boldsymbol{z})}{p_{\mathrm{T}}(\boldsymbol{z})}\right) \quad . \tag{5.18}$$

To evaluate the quality of LIF-based BMs, we set up networks consisting of five neurons and draw their Boltzmann parameters $(\boldsymbol{W}, \boldsymbol{b})$ from a beta distribution $\mathcal{B}(a, b)$ with parameters a, b (Appendix A.4).

The results in Figure 5.4 show a randomly chosen distribution of $2^5 = 32$ network states, where the average of 10 simulation runs (blue) shows a good correspondence to the theoretical target bars (red). The distribution was taken after a simulation time of 10^4 ms. The sampling results indicate a good approximation of the target distribution after this time frame, the sampled network distribution still improves when sampling longer. In Figure 5.4C we see two DKL curves on a logarithmic x- and y-axis. The dashed line shows the sampling quality of the ANM, converging to the target distribution $p_\mathrm{T}(\boldsymbol{z})$ exponentially as $\mathrm{D_{KL}}\left(p_\mathrm{ANM}(\boldsymbol{z}) \,||\, p_\mathrm{T}(\boldsymbol{z})\right) \to 0$. The full colored lines show ten simulations of LIF-based sampling with different seeds. We see that the quality improves as $\mathrm{D_{KL}}\left(p_\mathrm{N}(\boldsymbol{z}) \,||\, p_\mathrm{T}(\boldsymbol{z})\right)$ decays exponentially, even after 10^4 ms. This illustrates the *anytime computing* aspect of the sampling method: the accuracy improves continuously for longer sampling runtimes and the results can be evaluated at any point in time. This provides a tradeoff between speed and accuracy of the sampling approximation. This is an advantage to inference algorithms which allow an evaluation only after a certain (algorithmic) runtime (e.g., belief propagation). We see that the LIF-DKL curve in Figure 5.4C saturates at a certain point. This means that systematic deviations between both models are present at any runtime.

The main reason for this deviation is the interaction via PSPs. After refractoriness in the LIF model, interaction still exists because of exponential PSP decline. This contributes to the LIF membrane potential even after synaptic interaction. Although the TSO mechanism mitigates this effect to a certain degree (see Figure 5.3), it still impacts the sampling dynamics. Another reason for the deviation is the fact that the ANM employs PSPs with constant amplitude for a duration of τ_on. The probability of spiking therefore stays constant. For the LIF network, on the other side, the PSP shape is exponential and reaches its peak at the beginning of the interaction. This affects the sampled probability distribution.

In general, sampling with LIF neurons is still very accurate, as can be seen in Figure 5.4D. The distribution of DKL results indicates very good agreement between LIF-sampled distributions and the randomly drawn target distributions.

To demonstrate the scalability of our LIF sampling framework, we conducted simulations with larger networks. We have randomly drawn Boltzmann parameters $(\boldsymbol{W}, \boldsymbol{b})$ to set up 500-neuron networks with a 10% connectivity (see Appendix A.4 for more details). To demonstrate the versatility of the sampling framework, we have chosen three network activity modes that reflect vastly different firing states. The first firing state is the so-called *Asynchronous irregular* state, which is known from neuroscientific studies (*Destexhe*, 2009; *Destexhe et al.*, 2001; *Brunel*, 2000). In this state the network exhibits sparse and stochastic firing (Figure 5.5A). The asynchrony signifies that the network neurons fire independently from each other and irregularity indicates that the spike train of each neuron is random. It is similar to ground states of L23 cortical

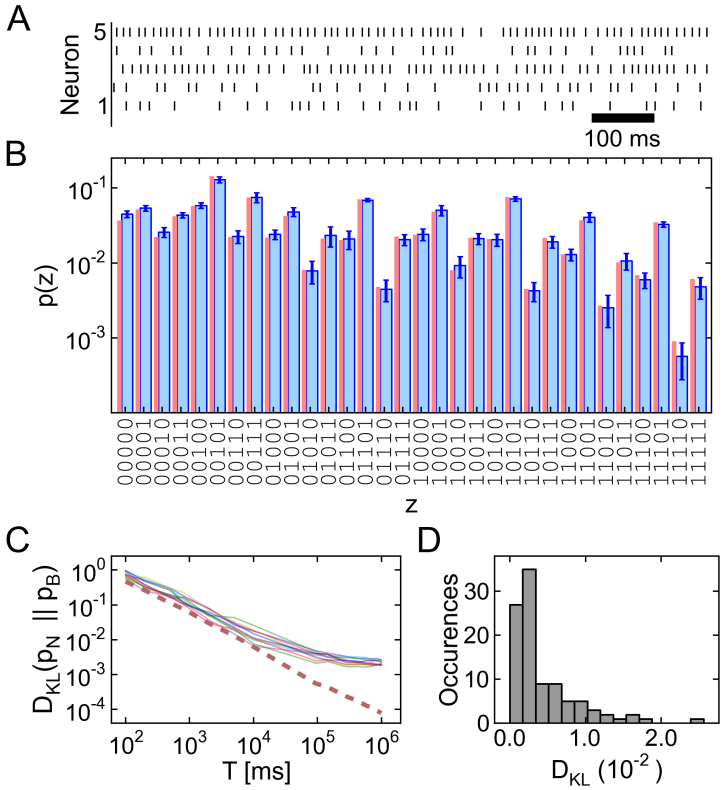

Figure 5.4: The randomly chosen Boltzmann parameters $(\boldsymbol{W}, \boldsymbol{b})$ define an LIF network of 5 neurons. **(A)** The firing pattern of the network encodes a Boltzmann probability distribution. **(B)** The resulting distribution is shown in blue bars after a simulation time of $T_{\text{sim}} = 10^4$ ms, averaged over 10 runs. The network distribution is compared with the target distribution (red bars). The error bars result from the standard deviation over the runs. We see that the agreement between both distributions is very good, especially in the frequently-occurring high probability states. **(C)** The plot shows the sampling performance of the ten LIF simulations with progressing simulation time on the logarithmic x-axis. The ten sampled distributions are compared with the target distribution with the distance $\mathrm{D}_{\mathrm{KL}}\left(p_B(\boldsymbol{z}) \,||\, p_{\mathrm{T}}(\boldsymbol{z})\right)(t)$ (colored curves) on a logarithmic y-scale. The exponential decrease of $\mathrm{D}_{\mathrm{KL}}\left(p_{\mathrm{LIF}}(\boldsymbol{z}) \,||\, p_{\mathrm{T}}(\boldsymbol{z})\right)(t)$ shows the improvement of the LIF sampling accuracy with increasing sampling duration. The dashed line represents the quality of the ANM model as $\mathrm{D}_{\mathrm{KL}}\left(p_{\mathrm{ANM}}(\boldsymbol{z}) \,||\, p_{\mathrm{T}}(\boldsymbol{z})\right)(t)$. We see the exponential convergence of p_{ANM} towards p_T as $\mathrm{D}_{\mathrm{KL}}\left(p_{\mathrm{ANM}}(\boldsymbol{z}) \,||\, p_{\mathrm{T}}(\boldsymbol{z})\right)(t) \to 0$. In contrast, the systematic deviations towards the LIF model are visible, as $\mathrm{D}_{\mathrm{KL}}\left(p_{\mathrm{LIF}}(\boldsymbol{z}) \,||\, p_{\mathrm{T}}(\boldsymbol{z})\right)(t)$ saturates and does not converge to zero for $T_{\text{sim}} \to \infty$. **(D)** The histogram consists of $\mathrm{D}_{\mathrm{KL}}\left(p_{\mathrm{LIF}}(\boldsymbol{z}) \,||\, p_{\mathrm{T}}(\boldsymbol{z})\right)(t)$ results from 100 randomly chosen LIF-based Boltzmann machines after 10^6 ms simulation time. The parameters were drawn from a beta distribution. Figure is taken from *Petrovici et al.* (2013).

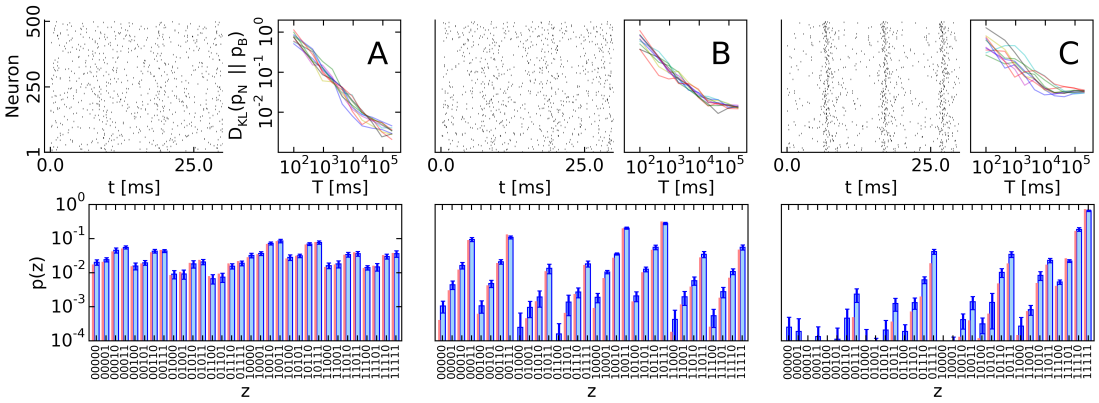

Figure 5.5: We test the scalability of the sampling framework by evaluating the LIF sampling performance on networks of 500 neurons. The Boltzmann parameters $(\boldsymbol{W}, \boldsymbol{b})$ for the networks were drawn from a beta distribution. We chose networks that produce three important firing modes in A, B, C respectively. **(A)** The network produces asynchronous and irregular spiking patterns. We randomly select five neurons and compare their marginal distribution to an MCMC-sampled estimate of the target distribution. We see that the network distribution approximates the target distribution very accurately. The $\mathrm{D_{KL}}\left(p_{\mathrm{LIF}}(\boldsymbol{z}) \,||\, p_{\mathrm{MCMC}}(\boldsymbol{z})\right)_{\mathrm{A}}(t)$ curves of 10 trials show the improving quality of the network distribution on the logarithmic x- and y-axis up to the saturation point. **(B)** We doubled the amplitude of randomly drawn Boltzmann weights, yielding more synchronized firing patterns. The target distribution is still approximated well. We see that $\mathrm{D_{KL}}\left(p_{\mathrm{LIF}}(\boldsymbol{z}) \,||\, p_{\mathrm{MCMC}}(\boldsymbol{z})\right)_{\mathrm{B}}(t)$ converges to a higher saturation point than $\mathrm{D_{KL}}\left(p_{\mathrm{LIF}}(\boldsymbol{z}) \,||\, p_{\mathrm{MCMC}}(\boldsymbol{z})\right)_{\mathrm{A}}(t)$ due to larger systematic deviations resulting from the weight translation (see text). **(C)** We quadrupled the weights from (A), resulting in strong interaction, causing synchronized network spiking activity. The high-probability modes can still be approximated well and the saturation point of $\mathrm{D_{KL}}\left(p_{\mathrm{LIF}}(\boldsymbol{z}) \,||\, p_{\mathrm{MCMC}}(\boldsymbol{z})\right)_{\mathrm{C}}(t)$ lies significantly higher. Figure is taken from *Petrovici et al.* (2016).

networks that we have discussed in Section 3.2. Although BMs do not exhibit biological neural dynamics, the class of Boltzmann distributions is powerful enough to reproduce biological firing patterns. Compared to spiking networks with biological architecture, like the previously discussed L23 network, we can measure the network performance for our sampled networks more intuitively using the DKL distance.

Aside from these biological firing domains we also evaluated 500-neuron BMs using larger weights, thereby regularizing network activity, as seen in Figure 5.5B. To test an even more extreme case, we further increased the average amplitude of Boltzmann weights $\boldsymbol{W}$ to yield even more regular and synchronized network firing patterns (Figure 5.5C). This firing mode is very similar to attractor states of the cortical L23 network, where a significant part of the network also fires synchronously. This shows that we can reproduce the firing behaviour of cortical spiking networks with LIF-based Boltzmann machines.

Even for large-scale networks, the performance of the LIF-based BMs can be estimated. Since we cannot calculate all 2^{500} states in the 500-neuron networks for the DKL, we evaluated the marginalized probability distribution of 5 randomly chosen neurons after 10 trials with the same network topology. We did this for all three firing modes in Figure 5.5. We see that the network-sampled distribution in Figure 5.5B is still a very good approximation despite significantly increased network weights. The network distribution in Figure 5.5C shows clear deviation from theoretical results, but still samples correctly from the most frequent states - the high-probability bars are well-matched by the network probabilities. We can quantify the improvement of the network distribution with ongoing sampling by evaluating the DKL runtime curves. In Figure 5.5A we can see the DKL converging with increased simulation time to a constant value, which is a saturation we already discussed in Figure 5.4. This saturation point indicates the order of magnitude of systematic errors compared to a conventional Markov-Chain Monte Carlo sampler. We see that the saturation point becomes higher with increasing weights for Figure 5.4A,B,C. With increasing LIF PSPs, the rectangular PSP approximation, described in Figure 5.2, becomes less accurate.

We still can conclude that even for larger networks we can sample reliably from target Boltzmann distributions. Due to the approximation of LIF PSPs in Equation 5.16, the extent of network interactions is an important factor which determines systematic errors.

5.4 Parameter robustness of sampling leaky integrate-and-fire networks

A Boltzmann distribution is fully defined by its network parameters $(\boldsymbol{W}, \boldsymbol{b})$. An LIF network, on the other hand, is specified by many neuron and synapse parameters. In this section we will evaluate LIF sampling performance for different parameter sets. These results will reveal the parameter configurations which yield the best performance and parameters that the sampling networks are sensitive to.

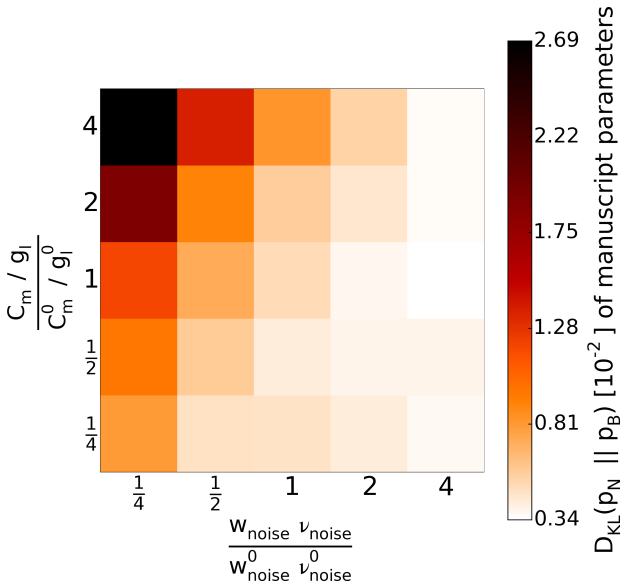

Figure 5.6: We investigate the sampling quality of LIF Boltzmann machines. We vary the membrane responsiveness on the y-axis, $\frac{C_\mathrm{m}}{g_\mathrm{l}}$, and the mean Poisson noise amplitude on the x-axis, $w_i\nu_i$. The results are evaluated in terms of $\mathrm{D_{KL}}\left(p_\mathrm{LIF}(\boldsymbol{z}) \,||\, p_\mathrm{T}(\boldsymbol{z})\right)$ for the 5-neuron network described in Fig. 5.4A,B,C. The $\mathrm{D_{KL}}\left(p_\mathrm{LIF}(\boldsymbol{z}) \,||\, p_\mathrm{T}(\boldsymbol{z})\right)$ for default parameters $\frac{C^0_\mathrm{m}}{g^0_\mathrm{l}}$ and $w_i^0\nu_i^0$ is shown in the center and is used as a reference.
The colormap shows two important results, which can be seen along the y-axis. A declining ratio of $\frac{C_\mathrm{m}/g_\mathrm{l}}{C^0_\mathrm{m}/g^0_\mathrm{l}}$ implies an increasing responsiveness of the membrane potential. Consequently, the DKL becomes lower, indicating better sampling. The second key impact factor is the change in noise amplitude on the x-axis. As we increase $w_i\nu_i$, the membrane does not only become more responsive, but also more stochastic and similar to an Ornstein-Uhlenbeck process, as explained in Section 4.3. Since increased stochasticity facilitates state changes, it improves sampling. The simulations were run for 10^6 ms and averaged over 10 trials. The error bars show the standard error of the mean. Figure is taken from *Petrovici et al.* (2013).

We have already pointed out that the responsiveness of LIF neurons in sampling networks is vital to encode fast transitions between $z_k = 0$ and $z_k = 1$. The parameter that controls the responsiveness of LIF neurons, is the effective time constant $\tau_{\text{eff}} = \frac{C_{\text{m}}}{g_{\text{l}} + g^{\text{syn}}}$. The smaller τ_{eff} becomes, the faster the membrane reacts to stimuli. This time constant can be changed by membrane properties or the noise amplitude. The former can be modified by changing the membrane capacitance C_{m} in the numerator or changing the leak conductance g_{l} in the denominator. The noise amplitude can be controlled by the average synaptic input $\langle g^{\text{syn}} \rangle \propto w_i \nu_i$.
We vary both the noise amplitude $w_i \nu_i$ and the membrane responsiveness $\frac{C_{\text{m}}}{g_{\text{l}}}$, investigating the effects on the sampling performance of a small 5-neuron network. The default parameters that were used in all our previous studies, will be defined as w_i^0, ν_i^0, C_{m}^0 and g_{l}^0. The parameter list of the default values can be found in Appendix A.7.

The results of these simulations are illustrated in Figure 5.6, where the color-coded DKL values indicate the sampling quality. The findings validate two important assumptions. The first finding is that a less responsive membrane due to a larger ratio $\frac{C_{\text{m}}}{g_{\text{l}}}$ is detrimental to sampling quality. The second result shows that increasing the noise $w_i \nu_i$ is beneficial, as stochasticity and responsiveness are increased at the same time. The increase of stochasticity in each neuron improves the approximations made in Section 4.3, where we identified the LIF membrane potential as an Ornstein-Uhlenbeck process.

Aside from membrane and noise parameters, the LIF neuron's sampling quality also depends on the distance between the spiking reset and threshold, $\Delta u_{\theta\rho} \coloneqq \theta - \rho$. Most importantly, when sampling from a Boltzmann distribution, we assume that switching of neuron states $z = 0 \to z = 1$ and $z = 1 \to z = 0$ happens instantaneously. This is true for the ANM implementation, where state changes can occur at any point in time. For an LIF neuron this is fundamentally different. Here the membrane potential propagates continuously in time and requires finite time to propagate from $u = \rho$ to $u = \theta$. This finiteness becomes crucial in case of a very high probability of spiking, $p(z = 1)$. In such a case synaptic input is so high that it pulls the membrane potential upwards immediately after the refractory period ends with $z = 1 \to z = 0$. According to the target distribution, an immediate state switch $z = 0 \to z = 1$ is very likely. In LIF simulation though, this state change is delayed because the neuron requires a certain time to elevate from ρ to θ. This duration becomes longer if the distance $\Delta u_{\theta\rho}$ increases. During this rise time, the LIF sampling network introduces a systematic error, as it necessarily samples from the $z = 0$ state while approaching the threshold. Since the effective time constant τ_{eff} governs the speed of this increase, a sufficiently low time constant will bridge an arbitrarily large distance $\Delta u_{\theta\rho}$ in a short time window. We can see in Figure 5.7A that the DKL is unaffected by an increase of the distance $\Delta u_{\theta\rho}$ as long as the effective time constant is small enough.

The last important parameter study is dedicated to synaptic interaction of LIF neurons. In Section 5.2.1 we have set $\tau_{\text{ref}} \equiv \tau_{\text{syn}}$ to equate the time window where the

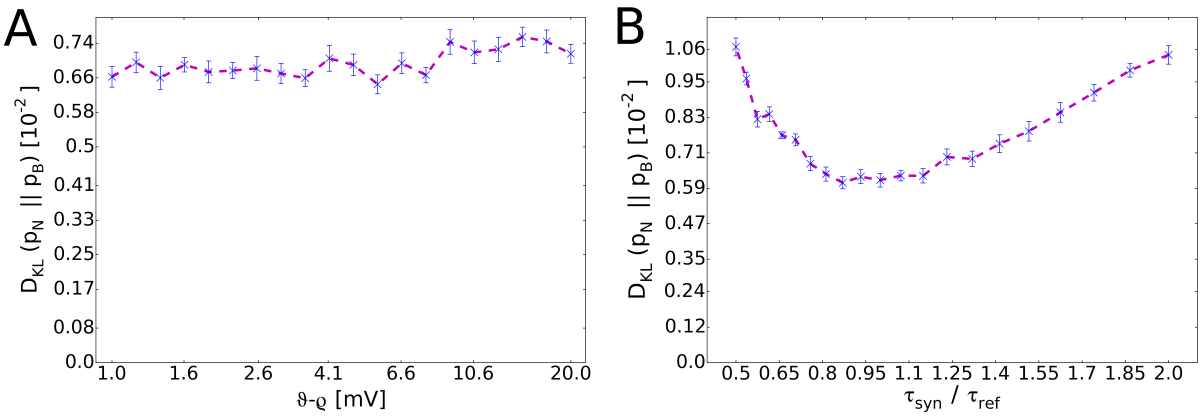

Figure 5.7: We study the impact of crucial parameter changes in the LIF sampling network described in Figure 5.4. We do not change the remaining parameters (Appendix A.7). **(A)** We vary the distance between threshold $\Delta u_{\theta\rho} = \theta - \rho$, with $\theta := -52\,\mathrm{mV}$ being fixed. The y-axis gives the sampling performance of the 5-neuron network as $\mathrm{D_{KL}}\left(p_{\mathrm{LIF}}(\boldsymbol{z}) \,||\, p_{\mathrm{T}}(\boldsymbol{z})\right)$. Due to the high noise input and an effective time constant of $\tau_{\mathrm{eff}} \approx 0.2\,\mathrm{ms}$, the membrane potential is fast enough to overcome the potential difference $\Delta u_{\theta\rho}$ and to perform state transitions $z = 0 \to z = 1$ quasi-instantaneous. **(B)** The variation of the ratio between refractory and synaptic transmission duration, $\frac{\tau_{\mathrm{syn}}}{\tau_{\mathrm{ref}}}$ at $\tau_{\mathrm{ref}} = 10\,\mathrm{ms}$ is shown. The sampling performance measure $\mathrm{D_{KL}}\left(p_{\mathrm{LIF}}(\boldsymbol{z}) \,||\, p_{\mathrm{T}}(\boldsymbol{z})\right)$ indicates the best performance at the default setting, $\frac{\tau_{\mathrm{syn}}}{\tau_{\mathrm{ref}}} \approx 1$.
All simulations have been run for $10^5\,\mathrm{ms}$ and have been averaged over 20 simulation runs. The error bars indicate the standard error of the mean. Figure is taken from *Petrovici et al.* (2013).

state is $z = 1$ and synaptic interaction occurs, τ_{syn}. To validate this heuristic approach, we swept over the parameter ratio of $\frac{\tau_{\text{syn}}}{\tau_{\text{ref}}}$ while keeping $\tau_{\text{ref}} = 10\,\text{ms}$. We see in Figure 5.7B that the $\mathrm{D}_{\mathrm{KL}}\left(p_{\text{LIF}}(\boldsymbol{z}) \,||\, p_{\text{T}}(\boldsymbol{z})\right)$ minimum (i.e. best sampling performance) is located at the ratio $\frac{\tau_{\text{syn}}}{\tau_{\text{ref}}} \approx 1$. This supports the rationale behind the heuristic approach.

Conclusively, we can summarize LIF sampling parameter studies and write down key points that are crucial for good sampling performance:

- **Stochasticity:** Providing a reasonable amount of stochastic input via Poisson noise weights w_i and input rates ν_i is necessary to facilitate state changes in the network. Lack of stochastic resources would result in more deterministic network dynamics, unable to transition to all states that make up the target distribution[2].
- **Membrane responsiveness:** The speed of neuron state transitions plays an important role in sampling. In the ANM a spike is triggered according to a probability and state changes are instantaneous. Therefore, multiple spikes can be triggered immediately after the elapsed activity period τ_{on}. For LIF networks this does not apply after the refractory period, as it requires finite time to cross the threshold θ. By increasing the membrane conductivity or the synaptic noise input, we decrease state transition times. With sufficiently high synaptic input we can achieve a sufficiently low effective time constant τ_{eff}. This enables state changes $z = 0 \to z = 1$ that require negligible time.

5.5 Conclusion: sampling with leaky integrate-and-fire neurons

In this chapter we extended the framework from single LIF neurons with high conductance to a network description that allows sampling from Boltzmann distributions. To assess its performance, we benchmarked the LIF sampling framework using the Kullback-Leibler divergence measure and performed LIF parameter studies to investigate valid parameter ranges for LIF Boltzmann machines. In our LIF sampling framework single spikes encode a probability distribution and are driven by stochastic membrane dynamics. Therefore, it is crucial that the membrane potential responds fast enough to stochastic inputs, ensuring the desired spike-based encoding of states.

Following this principle we can also reproduce biological firing patterns that we analyzed in cortical networks in Chapter 3. Although we can encode states more efficiently with the sampling framework than with the cortical network, the sampling framework has several non-biological attributes. One important constraint is the symmetry of Boltzmann weights, which restricts possible connectivity topologies of sampling networks. Also, neurons in a Boltzmann machine do not act exclusively inhibitorily *or* excitatorily

[2] In Chapter 7 we will see that Poisson input is not mandatory to achieve stochastic network dynamics. We will present different methods to enforce stochastic network behavior.

on other neurons. In physiology, the so-called *Dale's principle* is an empirical law stating that neurons are limited to one set of neurotransmitters, thereby enforcing one type of synaptic interaction (*Eccles et al.*, 1954). This is one of the reasons why Boltzmann machines are not directly mappable to cortical structures. On the other hand, LIF-based Boltzmann machines can also be seen as practical tools to explore real-world inference tasks, utilizing the benefits of spike-based encoding and biological mechanisms like short-term plasticity.

6 Applications of leaky integrate-and-fire Boltzmann machines

In the course of the previous two chapters we have developed a framework that enables probabilistic inference by sampling from binary probability distributions using spiking neurons in biological parameter regimes. The performance of these networks indicated potential for real-world inference tasks.

In the development of a sampling LIF network, the abstract model from *Buesing et al.* (2011) was an important reference point. However, there are many other sampling algorithms that are engineered to perform efficient inference on given data sets. In fact, the field of machine learning is dedicated to develop custom-tailored models to draw and process information on real-world data (*Bishop*, 2009). In this chapter we want to point out why using LIF neurons for predictive data analysis tasks is preferable to conventional sampling methods. One of the key points is the spike-based signal transmission, which is inspired from the transmission mechanisms in neural tissue. Biological neurons communicate primarily via emission of spikes. In contrast to this, in the abstract neuron model (ANM) each neuron communicates its state during every time step, increasing the necessary number of calculations per time step.

In this chapter we will see how spiking LIF networks employ mechanisms to use stochasticity as a computational tool, making them more efficient than conventional sampling techniques (e.g., Metropolis-Hastings sampling). To review these properties, we will construct *restricted Boltzmann machines* using our LIF sampling framework and apply them on inference tasks. These layered architectures are versatile networks and are commonly used as generative models and classifiers for big data sets (*Tieleman*, 2008; *Le Roux and Bengio*, 2008; *Sutskever et al.*, 2009).

6.1 Real-world inference tasks with spiking neurons

Before introducing multi-layered Boltzmann machines for stochastic inference, we will present a simple demonstration of LIF-based stochastic inference with only one layer. The network will be assigned to generate images from a benchmark data set, the *MNIST database* (Mixed National Institute of Standards and Technology) (*LeCun and Cortes*, 1998). It contains 60000 black-white hand-written digits with an image size of 28x28 pixels (Figure 6.1). The simulations described in Section 6.1 are based on the publications *Petrovici et al.* (2013, 2016), which were co-authored by the author of this thesis.

Figure 6.1: Subset of 100 images with 28x28 pixels from the MNIST database from a training set of 50000 digits (*LeCun and Cortes*, 1998).

We will illustrate a simple inference problem using a LIF network with 144 neurons. To reduce simulation overhead and limit the complexity of the images, we reduced the original image size of 28x28 to 12x12 pixels by averaging over pixel intensities. Accordingly, we arrange the network as a 12x12 grid where each neuron's firing rate represents a pixel value from white to black (0 to 255) with an all-to-all connectivity matrix. A white pixel indicates that the neuron is inactive on average with $p(z_{\text{white}}) = 0$, while a black pixel signifies constant firing with $p(z_{\text{black}}) = 1$. The connectivity matrix $\boldsymbol{W}$ has 144x144 entries and the bias $\boldsymbol{b}$ is represented as a 144-dimensional vector. In our demonstration we choose three MNIST images for this representation. Each image displays a single digit in $(0, 3, 4)$.

After initialization, the network is assigned to learn a Boltzmann distribution $p(\boldsymbol{z})$ with parameters $(\boldsymbol{W}, \boldsymbol{b})$ as a statistical model of the three digits (see Section 6.1.1). Note that we describe the network in the Boltzmann domain $(\boldsymbol{W}, \boldsymbol{b})$, since they define the distribution which we want to sample from. We transfer the distribution parameters to the LIF domain by applying the translation rules which we presented in Section 5.2.1. The distribution $p(\boldsymbol{z})$ corresponds to the prior distribution of the data. The network encodes pixel intensities as firing activity and is trained to reproduce the trained image data. This internal model of the three-digit data set is connected to an external input $\boldsymbol{y}$. The y_k nodes can be thought of as observed values for internal neuron states z_k, with $k \in \{1, \ldots, 144\}$.

Given the shape of a likelihood distribution $p(\boldsymbol{y}|\boldsymbol{z})$ as a Gaussian (Figure 6.2), the LIF-based BM with the distribution $p(\boldsymbol{z}, \boldsymbol{y})$ aims to infer the corresponding posterior distribution $p(\boldsymbol{z}|\boldsymbol{y})$. Since the network has learned the prior $p(\boldsymbol{z})$, it infers according to Bayes' law

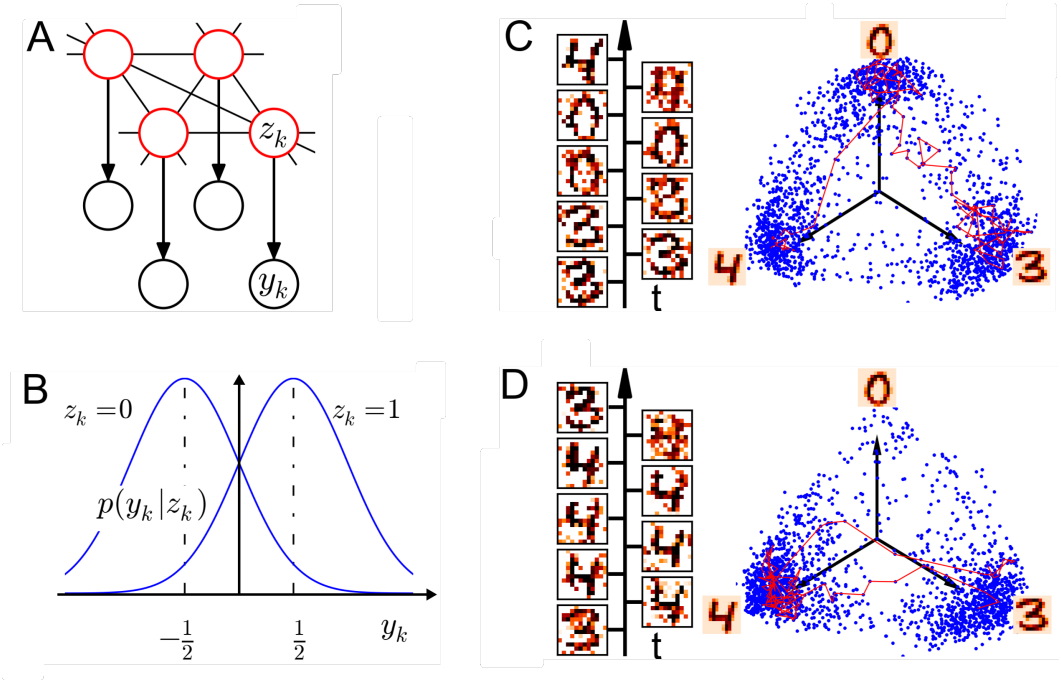

Figure 6.2: The figures show a proof-of-concept network setup to evaluate stochastic inference with LIF neurons. In **(A)** the graphical model is illustrated. The red circles show a fully-connected network of sampling units z_k that are trained on three digits, "0", "3" and "4". The black circles show the connected bias currents y_k. **(B)** The likelihood function $p(y_k|z_k)$ is modeled as a Gaussian. **(C)** After the network of 144 neurons has learned the three digits (axes), the network freely samples from all digits for a duration of 4000 ms with the same frequency. The network samples are shown as blue dots. The red trace shows a continuous propagation between all three digits on the axes. The timeline on the left displays sampled network states during the continuous propagation. We see that the network recognizes the images reliably. **(D)** The input y_k is inserted into sampling units z_k as an additional positive bias that increases the spiking rate of the four center neurons. This fixes four black pixels in the center of the images. The network infers correctly that the underlying image is "3" or "4". Consequently, the network samples (blue dots) from regions close to these two digits with equal probability, but neglects the "0" regions. The network infers from the given input y_k correctly that black center pixels are not compatible with the "0" digit.
Figure is taken from *Petrovici et al.* (2013).

$$p(\boldsymbol{z}|\boldsymbol{y}) \propto p(\boldsymbol{z}) \cdot p(\boldsymbol{y}|\boldsymbol{z}) \quad . \tag{6.1}$$

In Figure 6.2C we see a 2D projection of the network state $\boldsymbol{z}(t)$. This projection shows trajectory of the network state, traversing the state space. Most samples (blue dots) are accumulated at the axes, where the three MNIST digit states are located. The red curve portrays a trajectory of the network state, also displayed at the time axis to the left. We introduce an additional bias input, $\boldsymbol{y}$, to the trained network. Effectively, this corresponds to a positive current that shifts the mean membrane potential. The bias current is injected into the four center neurons, effectively clamping them to achieve maximum firing activity. The relationship between input $\boldsymbol{y}$ and states $\boldsymbol{z}$ is determined by a multivariate Gaussian $p(\boldsymbol{y}|\boldsymbol{z})$.

Since we activated the pixels in the center of the image, the underlying posterior $p(\boldsymbol{z}|\boldsymbol{y})$ is a distribution with a high frequency of states in the proximity of the "3" and "4" digits, but does not enclose states in the proximity of "0". Since all states $\boldsymbol{z}$ have a corresponding energy value $E(\boldsymbol{z})$, we will refer to this landscape as the energy landscape.

We see this network behaviour in Figure 6.2D, as the projected network states (blue points) are located close to "3" and "4". The regions in the proximity of "0" are visited significantly less frequent since we imposed black pixels in the center of the images with current $\boldsymbol{y}$.

This simple setup already shows the potential of the LIF networks, but one key attribute is the training algorithm for neural network performance. Since the used training algorithm follows a general principle, we will describe it in detail.

6.1.1 The contrastive divergence training algorithm

For a given data set, we define a Boltzmann machine with parameters $(\boldsymbol{W}, \boldsymbol{b})$ and a binary state vector $\boldsymbol{z}$. The objective is to adjust the parameters such that the Boltzmann machine is able to reproduce the data from the target data set. In such a case, averaging over the states in vector $\boldsymbol{z}$ after a sufficiently long of sampling duration, the states would reproduce the pixel intensities of the images as firing probabilities. To achieve such a generate model of the data set, we want to iteratively modify the network parameters until the given data can be reproduced by binary state vector $\boldsymbol{z}$.

A very common approach (not limited to Boltzmann machines) are *maximum likelihood* algorithms. This class of algorithms is based on finding the network parameters $\boldsymbol{W}, \boldsymbol{b}$ that maximize the likelihood $p(\boldsymbol{z}|\boldsymbol{W}, \boldsymbol{b})$ of observing the target data set by means of $\boldsymbol{z}$. To achieve this, so-called *gradient descent* methods are applied. To maximize $p(\boldsymbol{z}|\boldsymbol{W}, \boldsymbol{b})$, we look for the minima of the derivatives $\frac{\partial p(\boldsymbol{z}|\boldsymbol{W},\boldsymbol{b})}{\partial W_{ij}}$, $\frac{\partial p(\boldsymbol{z}|\boldsymbol{W},\boldsymbol{b})}{\partial b_i}$ reached. From now on we will omit the parameters $\boldsymbol{W}$ and $\boldsymbol{b}$ in the conditional distribution. We will formulate the maximization condition for W_{ij}, as the formulation is analogous for the biases b_i. We condition can be stated as,

$$\frac{\partial p(\boldsymbol{z})}{\partial W_{ij}} = \frac{\partial \left[\frac{e^{-E(\boldsymbol{z})}}{\sum_{\boldsymbol{z}'} e^{-E(\boldsymbol{z}')}}\right]}{\partial W_{ij}} \tag{6.2}$$

$$= e^{-E(\boldsymbol{z})} z_i z_j \left[\sum_{\boldsymbol{z}'} e^{-E(\boldsymbol{z}')}\right]^{-1} - e^{-E(\boldsymbol{z})} \left[\sum_{\boldsymbol{z}'} e^{-E(\boldsymbol{z}')}\right]^{-2} \sum_{\boldsymbol{z}'} \left[e^{-E(\boldsymbol{z}')} z_i' z_j'\right] \tag{6.3}$$

$$= p(\boldsymbol{z}) z_i z_j - p(\boldsymbol{z}) \left[\frac{\sum_{\boldsymbol{z}'} \left(e^{-E(\boldsymbol{z}')} z_i' z_j'\right)}{\sum_{\boldsymbol{z}'} e^{-E(\boldsymbol{z}')}}\right] \quad , \tag{6.4}$$

where we used the relationship $\frac{\partial E(\boldsymbol{z})}{\partial W_{ij}} = -z_i z_j$ to evaluate the derivative of the energy terms. Here $\boldsymbol{z}$ denotes the states produced by data and $\boldsymbol{z}'$ corresponds to states generated by the network model. We can reformulate the RHS of Equation 6.4 to eliminate $p(\boldsymbol{z})$ on both sides by taking the partial derivative of the logarithm, $\frac{\partial \log p(\boldsymbol{z})}{\partial W_{ij}}$. This reads

$$\frac{\partial \log p(\boldsymbol{z})}{\partial W_{ij}} = z_i z_j - \left[\frac{\sum_{\boldsymbol{z}'} \left(e^{-E(\boldsymbol{z}')} z_i' z_j'\right)}{\sum_{\boldsymbol{z}'} e^{-E(\boldsymbol{z}')}}\right] \quad , \tag{6.5}$$

with the second term on the RHS being the mean of all model states $z_i' z_j'$. We can therefore take the average of the states given by the data on both sides. This reads

$$\left\langle \frac{\partial \log p(\boldsymbol{z})}{\partial W_{ij}} \right\rangle_{\text{data}} = \langle z_i z_j \rangle_{\text{data}} - \langle z_i z_j \rangle_{\text{model}} \quad . \tag{6.6}$$

This equation describes the gradient of the log-likelihood. In this form, maximizing the log-likelihood is equivalent to minimizing the gradient $\left\langle \frac{\partial \log p(\boldsymbol{z})}{\partial W_{ij}} \right\rangle_{\text{data}}$. This can be described by the RHS intuitively: the gradient is minimized by decreasing the average distance between data states $\langle z_i z_j \rangle_{data}$ and model states $\langle z_i z_j \rangle_{model}$. Ideally, this difference corresponds to a model which produces states that are similar to the desired data states.

We can discretize the differentials in Equation 6.6 to adjust Boltzmann parameters $(\boldsymbol{W}, \boldsymbol{b})$. This results in

$$\Delta W_{ij} = \eta(\langle z_i z_j \rangle_{\text{data}} - \langle z_i z_j \rangle_{\text{model}}) \quad . \tag{6.7}$$

The complete derivation can also be done analogously for the biases b_i. Therefore, we only provide the result,

$$\Delta b_i = \eta(\langle z_i \rangle_{\text{data}} - \langle z_i \rangle_{\text{model}}) \quad . \tag{6.8}$$

These equations represent the change of parameters after one training step, where the model state average $\langle z_i z_j \rangle_{\text{model}}$ was subtracted from the data state average $\langle z_i z_j \rangle_{\text{data}}$.

This result is multiplied by η, which is referred to as the *learning rate*. This quantity modulates the amplitude of the increments ΔW_{ij} and Δb_i. It determines how large the change of Boltzmann parameters will be in each training step and often needs to be adjusted to the targeted data set. If the learning rate is set too low, the system only slowly converges towards a gradient minimum. If it is chosen too high, the system performs too large changes, causing large parameter fluctuations. The reason for the strong fluctuations is that the parameter increments described in Equations 6.7, 6.8 are linear approximations of the gradient and therefore only valid in the proximity of the current parameters.

Equations 6.7 and 6.8 conclude our derivation of the learning algorithm that we will use for Boltzmann machines. One important consideration here is that the training step in form of Equation 6.6 is intractable due to the evaluation of the term $\langle z_i z_j \rangle_{\text{model}}$. The calculation of this term requires to average over every state in the probability landscape given by $p(\boldsymbol{z})$. Due to the number of possible states, this approach is not feasible. As an example, our system consists of 2^{144} states and a calculation of the probabilities $\langle z_i z_j \rangle_{\text{model}}$ would demand a calculation of all the states to evaluate the normalizing partition function, $Z = \sum_{\boldsymbol{z}}$ and collecting a sufficient amount of samples from all of these model states. In conventional CPU architectures the calculation would take an insurmountable amount of time and overflow any existing memory.

To avoid these problems, a common approximation is made by averaging over a finite number n of model states before calculating the average. This method is a well-known approximation called *contrastive divergence* (CD), which is described in *Hinton* (2002). This heuristic approximation method speeds up the calculation of parameter updates and has proven to provide very good results, in many cases for only few samples ($n \approx 1$). There is no mathematical proof that CD minimizes the likelihood, but has shown to yield very good approximations for a variety of training tasks.

We have already shown the applicability of the CD training method for the proof-of-concept generative model we described in the previous Section 6.1, where we trained a Boltzmann machine consisting of ANM units and transferred the result to a LIF-based BM. The network parameters are described in Appendix A.5. In the following we will use our LIF sampling framework and approach bigger data sets with a layered network topology.

6.2 Deep learning architectures with spiking neurons

For a very simple 3-digit case we employed a Boltzmann machine with all-to-all connectivity, where the number of neurons (144) corresponds to the number of image pixels in a grid (12x12). Such an architecture where every network unit has a corresponding input unit, is called a *visible Boltzmann machine*. We trained this network with a simple CD algorithm on a network of ANM units. For larger data sets, this type of architecture declines in generative performance, as the number of parameters becomes insufficient to model all possible correlations between pixels. Also, not every data set is suitable to be

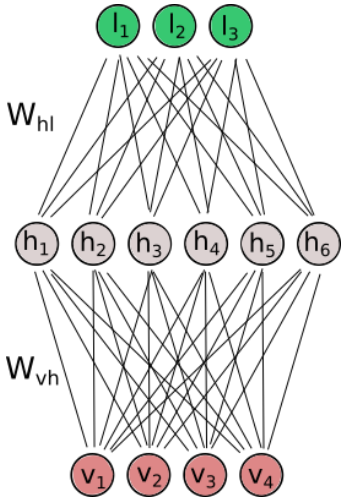

Figure 6.3: The layout of a restricted Boltzmann machine (RBM) consists of three layers. The lowest layer is the input layer, which is referred to as the visible layer. The states of the visible units are trained to represent the image pixel intensities (see Equation 6.9). The visible layer is connected to the hidden layer, which models the correlations between images from the data set. The label layer is an extension of the RBM architecture and serves as a readout for classification. Each label neuron is assigned to a class and fires as soon as the network recognizes the respective image.

represented by a Boltzmann machine. Every neuron has only one bias value and one possible trained weight to another neuron. For a sufficiently high variety of images, one weight per input unit cannot model all possible relationships between pixels from different images. In this section we want to test the generative and discriminative properties of the LIF-based framework on a large data set. To do this, we train the Boltzmann machines on the full MNIST training set, which includes 50000 hand-written digits of 10 digit classes (with 10000 test images). Afterwards we evaluate the classification and generative performance of the LIF model. The content in Section 6.2 is based on *Leng et al.* (2016), which was co-authored by the author of this thesis. All simulations in Section 6.2 have been conducted by Luziwei Leng.

Layered architectures are characterized by lateral and forward connections, which play different roles in information propagation. We will use so-called *restricted Boltzmann machines* (RBMs), which incorporate three different layers (Figure 6.3) (*Smolensky*, 1986). The lowest layer is called the *visible layer*, where the number of units corresponds to the number of inputs. In our case, all 28x28 pixels require 784 neurons. The visible layer connects to the next-higher layer, which is called the *hidden layer*. The name originates from the property that hidden states are not considered observable through pixel intensities. The purpose of the hidden layer is to provide additional degrees of freedom to a Boltzmann distribution for data set representation.

Adding even only one additional hidden layer for more degrees of freedom increases the computational overhead for simulation and training, as the number of connections is vastly increased. So far we also have not imposed any boundary conditions on how additional connections of the hidden units should be shaped. This introduces the problem that training of these connections leads to overfitting of data due to an increased size of non-observed, free parameters. These problems can be avoided using RBMs. For this type of network, all lateral (visible-to-visible, hidden-to-hidden) connections are removed and the network is restricted to inter-layer connections. This is a fixed constraint that simplifies the training of Boltzmann machines, since it removes lateral dependencies between random variables.

We can now adapt the CD algorithm to RBM connectivity by defining $\boldsymbol{v}$ as *visible states* and $\boldsymbol{h}$ as *hidden states*. We then maximize the log-probability of the network to generate visible states $\boldsymbol{v}$:

$$\left\langle \frac{\partial \log p(\boldsymbol{v})}{\partial W_{ij}} \right\rangle_{\text{data}} = \langle v_i h_j \rangle_{\text{data}} - \langle v_i h_j \rangle_{\text{model}} \tag{6.9}$$

Here we define $\boldsymbol{a}$ and $\boldsymbol{b}$ as the biases for visible and hidden units, respectively. Then we obtain weight and bias increments analogously to Equations 6.7, 6.8:

$$\Delta W_{ij} = \eta(\langle v_i h_j \rangle_{\text{data}} - \langle v_i h_j \rangle_{\text{model}}) \quad , \tag{6.10}$$

$$\Delta a_i = \eta(\langle v_i \rangle_{\text{data}} - \langle v_i \rangle_{\text{model}}) \quad , \tag{6.11}$$

$$\Delta b_i = \eta(\langle h_i \rangle_{\text{data}} - \langle h_i \rangle_{\text{model}}) \quad . \tag{6.12}$$

For the full MNIST data set, we use 784 (28x28) neurons and 1200 hidden units. Additionally, we introduce so-called *label units*. Each of these ten auxiliary units encodes a respective digit class by exhibiting activity if the network is shown an image from the corresponding class. Each label unit's firing activity indicates the digit class the network generates at that point in time. The label unit provides the readout for discriminating between different MNIST digits and the hidden units shape the generative network model for the data set. Hidden units encode similarities between digits and generalize common features to a certain degree. For instance, a similar curvature of the digits "8" and "9" causes correlated firing states between certain hidden units after training the network.

The RBM architecture is a widely-used building block for multi-layered architectures. In these so-called *deep Boltzmann machines*, the network is extended by additional hidden layers with restricted connectivity to map more complex data. Multiple network layers introduce a certain "depth", allowing to encode more features than pixel intensity or curvature. However, we will only discuss RBM architectures, since the same computational principles apply to networks with more layers. Also, we will see that LIF-based RBMs already achieve very good results for our applied problems. To identify unique properties of LIF-based networks, we will compare their performance to results from conventional methods.

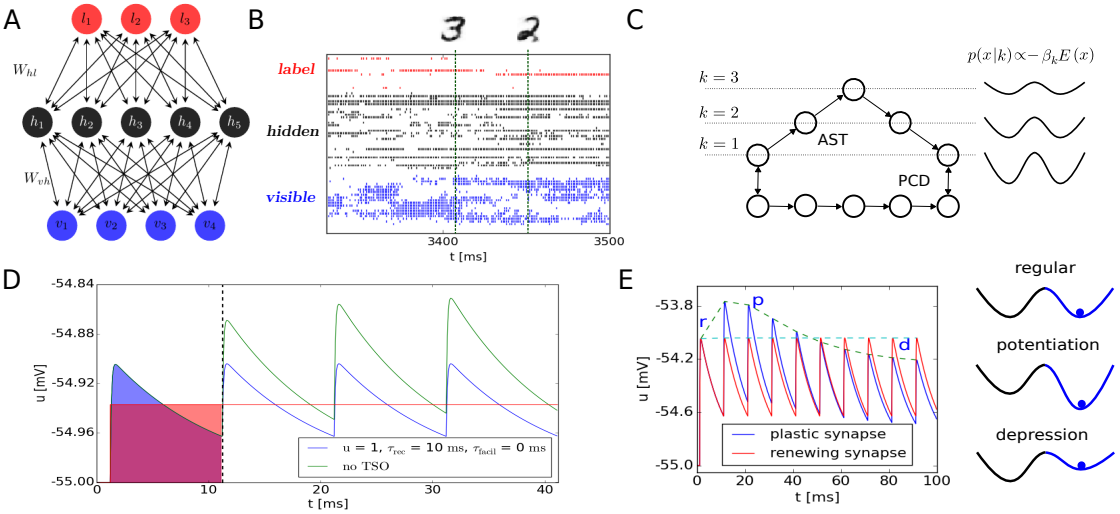

Figure 6.4: The setup and functionality of the LIF-based RBM. **(A)** A simplified illustration of the RBM network architecture. The visible layer consists of 784 neurons to encode input from 28x28 pixel-sized images. The hidden layer incorporates 1200 neurons and for each of the 10 digit classes, a label neuron is assigned. **(B)** An example spike train sequence shows the output of the network. **(C)** To ensure that the network learns digits from the entire probability space, an annealing algorithm is used (*Salakhutdinov*, 2010). It introduces a second Markov chain (AST), which samples on a reshaped energy landscape. In this chain, the energy is modulated by modifying the inverse temperature β with a control parameter k. The higher k, the flatter the energy landscape becomes, increasing the rate of state transitions to disjoint states. After a certain number of sampling steps, k is degraded to $k = 1$, returning to the original chain. **(D)** The LIF sampling framework implements alpha-shaped PSPs (blue, green) that approximate a rectangular shape of abstract neurons. The TSO mechanism (see Section 2.4) determines the PSP amplitude during subsequent spiking (blue). The orange area marks the constant PSP area that is set in the abstract neuron model. The orange line marks the targeted amplitude height for the translation from the ANM to the LIF model. In our approximation, the blue and orange area are identical (Section 5.2.1). **(E)** The TSO mechanism incorporates potentiation (p) and depression (d). In a system without TSO (r), the energy landscape is static and PSPs are kept constant, as shown in (D). Potentiation (p) increases the amplitude at the beginning of the spike sequence, consolidating the network in a mode in the network by deepening the potential valley. To transition from this state, synapses are weakened using depression (d), flattening the potential minimum. This leads to an uplift (p), then a decline (d) of PSP amplitudes. The neuron and synapse parameters of the network can be found in *Leng et al.* (2016). Figure is taken from *Leng et al.* (2016).

6.2.1 Deep learning on the full MNIST data set

For the visible Boltzmann machine we used the standard CD parameter update scheme that we described in Section 6.1.1. Since we will use the full MNIST data set (50000 training images), the network takes longer to generate a probabilistic model of the data set. Also, an increased heterogeneity of the data increases the difficulty of traversing the full energy landscape. We will look at the probability density $p(\boldsymbol{v})$ and the energy function $E(\boldsymbol{v}, \boldsymbol{h})$,

$$p(\boldsymbol{v}) = \frac{1}{Z} \sum_{vh} \exp\left(-E(\boldsymbol{v}, \boldsymbol{h})\right) \quad , \tag{6.13}$$

$$E(\boldsymbol{v}, \boldsymbol{h}) = -\sum_{i \in \text{visible}} a_i v_i - \sum_{j \in \text{hidden}} b_j h_j - \sum_{i,j} W_{ij} v_i h_j \quad . \tag{6.14}$$

States with a higher occurrence probability have a lower energy. The probability landscape is fully defined by the target data set. Big, inhomogeneous data sets cause many potential walls in the energy landscape, often hindering sampling from parts of the state space during training. Consequently, data subsets that show less similarity to the remaining data, are often separated by potential walls and often undersampled by the sampler during the training procedure. Since this problem is well-known, many methods exist to facilitate transitions to all high-probability modes in separated state regions after updating the parameters. The so-called *mixing time* describes the time it takes to traverse separated regions of different high-probability modes in the energy landscape. To decrease the mixing time (i.e., facilitate training), we will extend the CD training mechanism to facilitate improbable state transitions. We will train the ANM-based RBM defined by $(\boldsymbol{W}, \boldsymbol{a}, \boldsymbol{b})$, using a training algorithm called *coupled adapted simulated tempering* (CAST) (*Salakhutdinov*, 2010). We will briefly outline how the training mechanism works in principle. More detailed discussions can be found in *Leng* (2014); *Martel* (2015); *Leng et al.* (2016). The CAST mechanism combines CD training with a tempering procedure shown in Figure 6.4C. This combination includes two sampling Markov chains in order to collect the model state averages to implement the parameter updates in Equations 6.10, 6.11, 6.12. The first chain implements a procedure called *Persistent contrastive divergence* (PCD), which updates the network parameters after a predefined number of sampling steps. The second chain, however, samples in parallel to the first chain while reshaping the energy landscape. This part constitutes the *adapted simulated tempering* (AST) of the CAST algorithm. To provide a closer look at the tempering mechanism, we consider the inverse temperature $\beta = \frac{1}{kT}$ in the Boltzmann probability distribution,

$$p(\boldsymbol{v}) = \frac{1}{Z} \sum_{vh} \exp\left(-\beta E(\boldsymbol{v}, \boldsymbol{h})\right) \quad . \tag{6.15}$$

Here, the inverse temperature β is used to reshape the probability landscape. In our initial definition in Equation 5.3 we set $\beta \equiv 1$. Now we use β to change the probability

landscape. This is analogous to physical systems, where an increase of temperature results in an increase of energy and thereby increases the probability of state transitions. For our RBMs this has comparable consequences, since it alters the probability landscape. In our CAST training algorithm, the tempering Markov chain takes valid samples only at the default temperature ($\beta = 1$). The temperature is increased to enable unlikely state transitions. To make sure that the system transitions into all probability regions, the temperature is varied accordingly. If the network samples oversamples a certain probability region, the temperature is increased ($\beta < 1$) to facilitate a state transition into undersampled regions. The CAST algorithm modulates the probabilities of state transitions; the higher the temperature becomes, the more likely it is to transition into an undersampled, high-probability state. This happens automatically, as the probability of sampling in a certain region (Equation 6.15) depends on β.

As soon as the tempering Markov chain switches back to $\beta = 1$, both the PCD and the AST chain switch their states. The purpose of this state switch is to initiate the successive training step in a state subspace different from the previous one. This ensures that the training includes samples from as many different regions as possible from the state space that represents the full MNIST data set with Boltzmann parameters $(\boldsymbol{W}, \boldsymbol{v}, \boldsymbol{h})$. These trained parameters define the full RBM. In principle we could choose any sampling model to evaluate the generative properties and perform classification of hand-written digits. For comparison, we will use the widely-adopted Gibbs sampling routine to sample from the probability distributions. We will compare the performance with LIF-based results. We will then qualitatively discuss the benefits and downsides of their biological properties compared to conventional approaches.

To evaluate both the generative and discriminative properties, we need means to assess this performance. To evaluate the performance of LIF-based RBMs as a generative model, we choose a projection which illustrates the differences between conventional MCMC sampling and our LIF-based approach. This requires to project the $\{0, 1\}^N$ state space onto a two-dimensional plane, retaining the structure of the internalized data set. It is crucial that neighbouring states still stay close to each other in the projected state space.

Unlike for the simple generative model with three digits in Section 6.1, we cannot visualize the full data set of 50000 digits on a two-dimensional plane by means of simple coordinate transformation. Instead, we use a method called *t-distributed stochastic embedding* (t-SNE), introduced in *Maaten and Hinton* (2008). In this projection method, data pairs that are close to each other in the $\{0, 1\}^N$ space have a high probability of being placed close to each other in a 2D space. Conversely, data pairs with dissimilar features have a low probability to be mapped in a neighborhood. We perform this nonlinear projection on the 2D space, making it possible to compare sampling paths of Gibbs sampling and LIF-based inference. More details on t-SNE projection of a probability landscape can be found in *Martel* (2015).

The results can be seen in Figure 6.5. Here we look at the projected states which the network assumed during inference. In 6.5C, Gibbs sampling has been used to draw 4000 samples from the trained probability distribution. We see that the network is unable to transition into states other than "7" and "9" during the whole runtime. It is trapped in these probability modes. In Figure 6.5C we see the sampling trace of the LIF-based

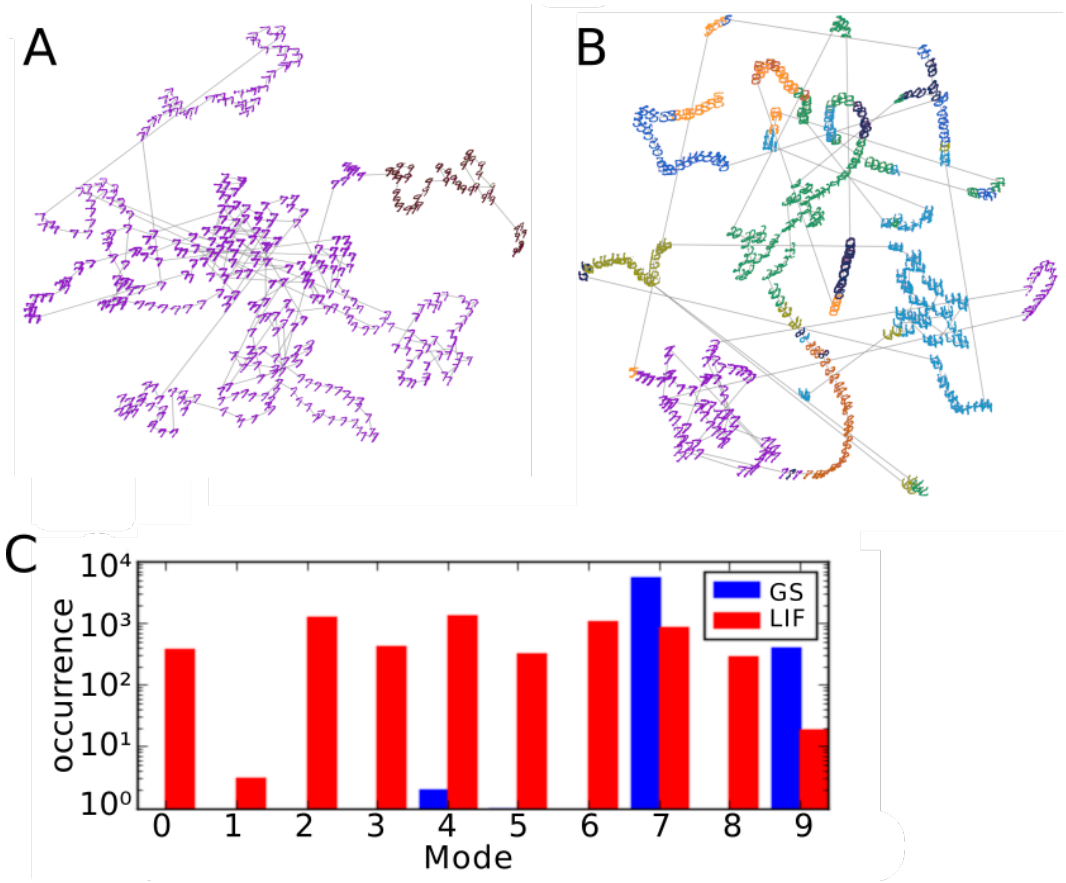

Figure 6.5: **(A)** The t-SNE projection of Gibbs-sampled states of the trained system shows that the network almost exclusively samples from the "7" states (purple), and occasionally transitions to "9" and "4" states. All in all, the network generates only a small fraction of the data set. The projections include 4000 sampling steps. In-between each plotted sampling step, 10 steps have been left out. **(B)** The LIF-sampled results show a uniform coverage of all digits, which are represented in different colors. The LIF simulation has been run for $T_{\text{sim}} = 4000 \cdot \tau_{\text{ref}}$, with $\tau_{\text{ref}} = 10\,\text{ms}$. Here, the refractory period in the LIF domain corresponds to one sampling step in the Gibbs domain.
(C) Both networks classify the digits. For Gibbs-sampled classification (blue bars), we see that only three digits are generated by the network. In the LIF case, we see that the network traverses through the modes of the entire data set and samples from all digit modes (red bars). The samples were classified during 4000 sampling steps with intervals of 10 sampling steps. Figure is taken from *Leng et al.* (2016).

network. To compare the LIF-based sampling sampling time to the Gibbs sampling in discrete time steps, we assumed one sampling step as τ_{ref}, since it constitutes the fixed time period a network spends in the $z = 1$ state. Therefore, Figure 6.5C shows states resulting from the runtime of $\tau_{\text{ref}} \cdot 4000 = 40000\,\text{ms}$. We see that the LIF network exhibits transitions between all ten digit classes and samples from a complete data set. For both networks we used the same trained parameters $(\boldsymbol{W}, \boldsymbol{a}, \boldsymbol{b})$. The difference between both models therefore lies in the dynamics of the network.

A key difference that explains the superior LIF sampling is the short-term plasticity of the LIF synapses. In the abstract model, the constant rectangular PSPs are a consequence of the NCC (Equation 5.1) and follow as a necessary condition from sampling from a Boltzmann distribution. In the LIF model, however, the TSO short-term plasticity mechanism is used to achieve equal PSP amplitudes for successive spiking (Figure 6.4). This mechanism temporarily changes the local energy landscape of the system during each sampling step.

An energy minimum in the data space represents a state region where the network state is similar to a digit of the data set. In this probability mode, the hidden units encoding for the digit are excited and units that do not encode for this digit, are inhibited by visible units. The stronger the output from the digit-encoding hidden units, the more distinct the network-sampled digit becomes. We can strengthen these connections to deepeen the energy minimum by introducing short-term potentiation (STP), as described in Section 2.4. This makes the generated image more recognizable. On the other hand, it also decreases the probability to transition to another mode since it deepens the potential well. We add short-term depression (STD) to enforce a state transition a to different digit when the network has settled in the local mode. As soon as the system produces a highly recognizable (i.e., discriminative) digit, the STD mechanism initiates a weakening of digit-encoding synaptic connections and facilitates transitions into other modes (Figure 6.4).

Therefore, we apply short-term plasticity to balance two effects:

- **Improving image recognition:** The specialized, feature-encoding hidden-to-visible synapses are strengthened to generate more distinctive digits. We apply the STP mechanism to deepen the corresponding energy minimum.
- **Improving transition rate to other digits:** As soon as the system samples for a sufficiently long time from the distinct digit, the STD mechanism weakens the hidden-to-visible connections to lift the energy minimum. This increases the probability to leave the mode.

Using these mechanisms, the LIF network continuously reshapes local regions of the energy landscape to sample from the data set more efficiently than a CAST tempering algorithm. This means that it changes the connectivity strength and thereby locally changes the energy landscape. This allows shorter mixing times. In contrast to this, the Gibbs algorithm exhibits significantly longer mixing times, although it samples according

to the trained probability distribution $p(\boldsymbol{v})$. Here, some digits are more probable to occur than others due to their distinct properties like total pixel intensity or uniqueness in curvature. In practical applications, we are often interested in modelling all different classes with the same probability despite differences in training data count. In a data set where a class is overrepresented by more available training images, the network will map the data set accordingly, sampling frequently from the overrepresented image class modes. To build a model that generates all image classes equally often, the skewed probabilities towards specific digits then need to be compensated. In LIF networks this compensation occurs via short-term plasticity during reshaping of the local energy landscape. These generative properties of the RBM are difficult to asses quantitatively, since the number of network states do not allow a straightforward calculation of measures like the DKL, which we have used in the previous chapter. An extensive discussion on the assessment of the generative properties can be found in *Leng* (2014). Aside of the generative properties, the LIF-based network also achieves very good discriminative results. The classification rate of LIF-based units amounts to 96.4%, while the Gibbs-sampled network achieves a rate of 96.7% (*Leng et al.*, 2016).

6.2.2 Sampling from an inhomogeneous MNIST data set

We have argued that the LIF network is capable to efficiently sample from data sets which have a heterogeneous energy landscape. To further validate these LIF properties, we train our networks on an unbalanced data subset which contains 1000 digits from the original MNIST data set and evaluate the samples obtained by the LIF network. From these digits, 820 images contain the "1" image and "0", "2", "3" and "8" are included in 45 different realizations each. We have specifically chosen this set of digits because the "1" digit is particularly dissimilar to all the other digits, therefore difficult to transition to the probability modes of the remaining digits. We trained an RBM with 28x28 visible neurons, 400 hidden neurons and 5 label neurons, with one label per digit. We have chosen less hidden units for this subset because we only have 5 different digit classes and therefore require less feature-encoding units.

As in the previous section, we trained an ANM network using the CAST algorithm, where temperature changes facilitate state transitions. On this trained RBM, we performed sampling via Gibbs sampling, tempered Gibbs sampling and LIF sampling. We can see the sampled results in Figure 6.6. Standard Gibbs sampling produces almost exclusively "1" digits and is basically trapped in this mode (Fig. 6.6A). Tempered sampling performs significantly better, but despite its temperature-modulated state transitions, it samples from the "1" state for a majority of time (Fig. 6.6B). From a statistical point of view, this represents the trained Boltzmann distribution accurately, since the "1" digit is overrepresented in the training data set.

In the LIF-based case, the short-term plasticity mechanisms reshape the trained energy landscape to sample equally often from all trained digits (Fig. 6.6C). This highlights the ability of short-term plasticity to locally change the underlying connectivity structure. For practical purposes this is an essential feature of functional networks, as it naturally

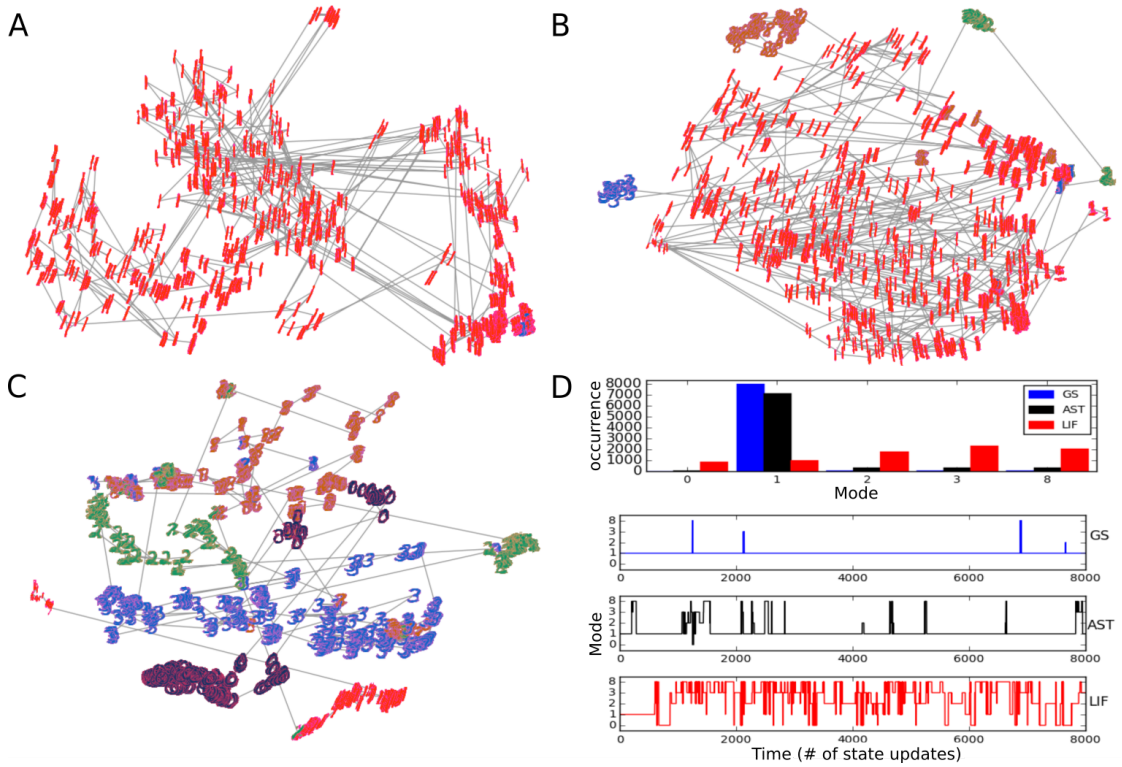

Figure 6.6: The RBM has been trained on a data subset $\{1, 2, 3, 8, 0\}$ of 1000 images with the "1" image being represented in 820 images. All remaining images are equally represented in 45 images. **(A)** The states are projected with t-SNE to two dimensions and show Gibbs-sampled states, where only the overrepresented digit "1" is generated. The grey lines indicate the traveled distance in energy landscape. **(B)** The states were sampled using simulated annealing to transition to remote states. The tempered system performs better than the Gibbs-based system. **(C)** LIF-based sampling shows a homogeneous generation of MNIST digits, independent from their occurrence in the data subset. **(D)** All digits are sampled equally often in the LIF simulations (red bars). The AST sampler generates all digits, but shows a strong preference of the "1" mode (black bars). The Gibbs sampler only samples from the dominant "1" mode. The timelines below the distribution show state changes during a duration of 8000 sampling steps for all three samplers. We rarely observe state transitions of the Gibbs sampler, whereas the AST sampler exhibits infrequent mode changes and samples predominantly from the "1" mode. The LIF network switches frequently between all modes, independent from the training data sizes. Figure is taken from *Leng et al.* (2016).

incorporates a way to sample from all possible states equally often.

To strengthen this property in LIF networks, one can think of more biologically plausible mechanisms that naturally extend the properties of stochastic networks. So far we have only studied short-term plasticity. Mechanisms such as adaptation of the membrane (AdEx) or spike-timing-dependent plasticity (STDP) can potentially improve the sampling performance of spike-based networks (*Pfister et al.*, 2006; *Neftci et al.*, 2015; *Nessler et al.*, 2013).

Thus far we have discussed the computational properties of LIF networks, but the simulation substrate remained the same as for conventional sampling techniques. On a fundamental level, network simulations can only be as fast as the simulation substrate allows. Although the LIF network performs significantly better than the MCMC sampling algorithms, it is still governed by a differential equation that needs to be solved in each time step for each neuron and synapse. This constraint appears unnatural in the sense that the power of neural networks in biology stems from computing in parallel. In the next section we will introduce the idea of a *physical model*, which is a computing architecture that incorporates model components as physical entities.

6.3 Neuromorphic hardware

Biological neural systems can be regarded as exceptional computational devices in terms of energy efficiency and robustness. To use such systems for real-world computational tasks, there are two key aspects. Firstly, an adequate representation of the neural correlate needs to be chosen in a computational model. For instance, in this work we have thus far used AdEx and LIF neurons, which employ the membrane potential and conductance as dynamic variables in an ODE. The second key component is the substrate that is used for computation. Up to now, we have built computational networks based on differential equations. We performed numerical integration of these equations using the simulation infrastructure presented in Section 2.5 on conventional computers. For large network sizes and dense connectivities, this conventional computing substrate shows severe limitations in terms of speed and energy consumption. These limitations make it necessary to simplify the neural networks to a substantial degree. With many computational constituents in neural nets still unknown, it is often not clear which segments of the neural architecture should be simplified. Some computational traces can often only be observed at large-scale simulations (*Markram et al.*, 2015; *van Albada et al.*, 2015). For such studies, often neuron counts up to the order of 10^8 with extensive connectivity patterns (10^8 synapses) need to be simulated (*Djurfeldt et al.*, 2008; *Helias et al.*, 2012). To conduct such large-scale simulations, computing clusters are used to cope with the vast simulation overhead (*Markram*, 2006; *Potjans and Diesmann*, 2012). For spiking neurons with adaptation properties, simulating large topologies even on a time scale of seconds takes many hours, depending on the connection density (*Kunkel et al.*, 2014). Although simulations on time scales of seconds can be sustainable, de-

velopmental processes that happen over several months or years in biology require an insurmountable amount of energy.

The reason for this energy consumption is the key difference between computations of nerve cells and von-Neumann simulation substrates: the prime attribute of neural computing is the parallel and asynchronous signal processing. Using von-Neumann simulation substrates diminishes the advantage of spike-based computing compared to other computational algorithms. Spike-based encoding becomes efficient in substrates that do not rely on iterating over all network components in a time step. In biology, instead, each nerve cell carries out computations in parallel. A resolution of this energy efficiency problem could lie in emulating the biological substrate. Consequently, the solution is a *physical model* of biological tissue. Such a physical model requires a physical representation of the components from neural networks. A device that embeds such representations and mimics the biological template of neural networks is called *neuromorphic hardware*. The idea to represent neural tissue on electronic devices was first proposed by C. Mead and has become increasingly popular since (*Mead*, 1990). In principle, many types of neuromorphic architectures can be thought of and customized to solve specific computational problems. We will restrict ourselves to very-large-scale integration (VLSI) of electrical circuits that are engineered to *emulate* electric properties of neurons and synapses (*Schemmel et al.*, 2010). In particular, we will focus on systems that incorporate LIF networks.

In the following we will describe the challenges of such implementations and present results of small stochastic networks.

6.3.1 The Spikey chip-based system

We will start with the Spikey system (Figure 6.7), which is based on a neuromorphic chip built in the FACETS project (*FACETS*, 2010). The mixed-signal chip is developed in a 180nm CMOS process and is a physical model of the conductance-based leaky integrate-and-fire neuron,

$$C_{\mathrm{m}}\frac{du}{dt} = -g_{\mathrm{l}}(u - E_{\mathrm{l}}) - \sum_i g_i(u - E_{\mathrm{i}}^{\mathrm{rev}}) \quad , \tag{6.16}$$

where the quantities are denoted according to our introduction of the LIF model in Chapter 2. In this description, u is the membrane potential and g_{l}, E_{l}, $E_{\mathrm{i}}^{\mathrm{rev}}$ are the leak conductance, leak potential and reversal potential, respectively. The synaptic conductance $g_i(t)$, however, is composed of multiple factors,

$$g_i(t) = p_i(t) \cdot g_{\mathrm{i}}^{\max} \cdot w_i \quad . \tag{6.17}$$

Here the synaptic weights are denoted as w_i and have a 4-bit resolution due to hardware constraints. The conductance kernels $p_i(t)$ are chosen to be exponential functions. The quantity $g_{\mathrm{i}}^{\max}$ is a multiplicative hardware parameter which increases amplitude of

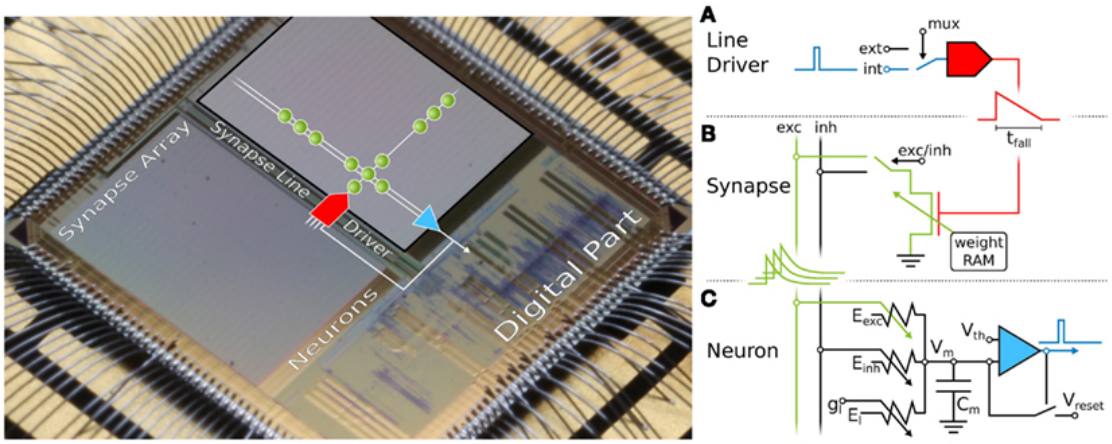

Figure 6.7: **(Right)** The core part of the Spikey system is the neuromorphic chip with a size of 5x5mm^2. The chip has been developed in a CMOS 180 nm fabrication process. The picture shows a Spikey chip with the digital part in the lower right and the neuromorphic part in the upper left. The digital part encompasses the infrastructure of configuration and spike data transmission to the host PC. **(Right, A)** The digital spike signals are transmitted as rectangular signals to one of the 256 synapse drivers (red), where they are converted to voltage kernels with a falling edge t_{fall}. **(Right, B)** In the synapse nodes the voltage is amplified by the excitatory or inhibitory weights w_i and converted to exponentially decaying currents (green curves). **(Right, C)** These currents are transmitted to the neuron circuits, where they are converted into voltages proportional to reversal potentials E_{exc} and E_{inh}. Note that the synaptic input is the same for all neurons that are connected to the same synapse column. This restricts the total connection topology of the system. If V_m exceeds threshold value V_{th}, the membrane is clamped to V_{reset} and a digital spike is emitted in the shape of a rectangular pulse (blue). The Figure is taken from *Pfeil et al.* (2013).

the synaptic conductance $g_i(t)$. More detailed descriptions can be found in *Schemmel et al.* (2006).

The Tsodyks-Markram short-term plasticity mechanism (TSO, Section 2.4) is also implemented. In the hardware implementation of TSO, additional circuitry modulates the conductance time course $p_i(t)$ to either induce depression (decrease of amplitude) or potentiation (increase of amplitude) of the conductance trace $g_i(t)$. In the previous section we have already highlighted its importance for stochastic inference with LIF neurons.

The physical implementation of the LIF components in the Spikey chip is outlined in Figure 6.7. The core functional blocks of the neuromorphic chip are the neuron and synapse circuitry. Each neuron can be connected with any neuron on the chip via so-called synapse arrays. In these arrays, synaptic signals are integrated in nodes into synaptic currents and relayed to the postsynaptic neurons. The postsynaptic neurons integrate arriving signals in the membrane potential according to Equation 6.16. The conductance, as well as the membrane potentials of all neurons, are governed by differential equations of the LIF model, devised as electronic circuits placed on the chip. Consequently, the dynamic variables of the LIF neuron model are analog signals resulting from physical processes that occur on the chip.

In this analog approach the temporal scales are reduced. An indicator of the time scale of the neuron is the membrane time constant $\tau_m = \frac{C_\mathrm{m}}{g_\mathrm{l}} = C_\mathrm{m} \cdot R$. In biological neurons the membrane capacitance integrated over the whole membrane surface is in the order of $C_\mathrm{m} \approx 0.1\,\mathrm{nF}$ (*Alberts et al.*, 1994). The nanoscale circuit integration on the Spikey chip employs capacitances C_m and resistances R such that the time scales are smaller by a factor of 10^{-4}. This implies that dynamics on the hardware propagates 10000 times faster than biological systems in real-time. We will therefore refer to the acceleration of network dynamics by 10^4 as the *speed-up factor*. Since the correlates of neuron parameters are physical components, they evolve in continuous time. This makes neuromorphic computing substantially more energy-efficient than solving differential equations in simulation environments (*Brüderle et al.*, 2011). In addition to the advantages of analog aspects of the system spike event transmission is digital. A transmission via currents would require would introduce additional resistive elements along the signal lines, causing energy loss during transmission. Instead, the digital FPGA-based communication in this mixed-signal architecture transmits time stamps of spike events and does not require large quantities of current flows in the system. This further increases the energy efficiency of the system. This type of communication is similar to biological systems where communication occurs primarily through spike events.

Variability of neuromorphic components

A neuromorphic system based on analog dynamics not only brings the advantage of a significant speed-up, but also entails challenges for users. As typical to any hardware production process, resulting mismatches of the components are a natural side effect

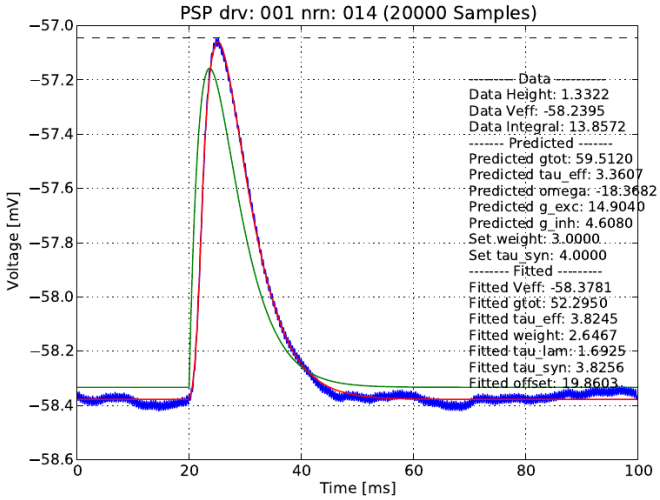

Figure 6.8: The figure shows an example calibration procedure of an excitatory PSP generated by neuron interaction on the chip. The objective here is to find sets of biological parameters for hardware settings. In this example case, we know the remaining biological parameters and look for the time constants τ_{syn} and τ_{m} for the given hardware settings. In theory, we know the dependencies of the PSP of all parameters, therefore we can use Spikey-produced PSPs to obtain fitted values of the time constants. Figure is taken from *Petkov* (2012a).

of the fabrication process. In digital computing devices, such mismatches are removed and do not affect the computation outcome. But in computation using analog signals, the mismatch characteristics of each neuron and synapse unit are reflected in network dynamics. This spatial variation between different parts of hardware circuitry is often referred to as *fixed-pattern noise*. Neuron and synapse circuits having the same specification on a device can exhibit significant deviations from ideal behavior. However, these systematic deviations can be characterized and dealt with to a certain degree, making emulation possible. Technical details to the order of magnitude of these variations can be found in *Pfeil* (2011); *Petkov* (2012b).

A second type of variation that needs to be considered is the so-called *trial-to-trial variation*. Here, the same hardware component exhibits different dynamics on each run due to statistical fluctuations, e.g. temperature variations. For instance, the leak membrane potential of a neuron can vary in every run. In contrast to simulations, emulations relying on analog signals are not deterministic. Since this trial-to-trial variation is caused by statistical properties, it is possible to decrease these variations by performing multiple emulation runs. This methodology is encouraged for analog computing devices because of the high speed-up, making it possible to measure and reset hardware parameters on a time scale of seconds before performing the emulation run.

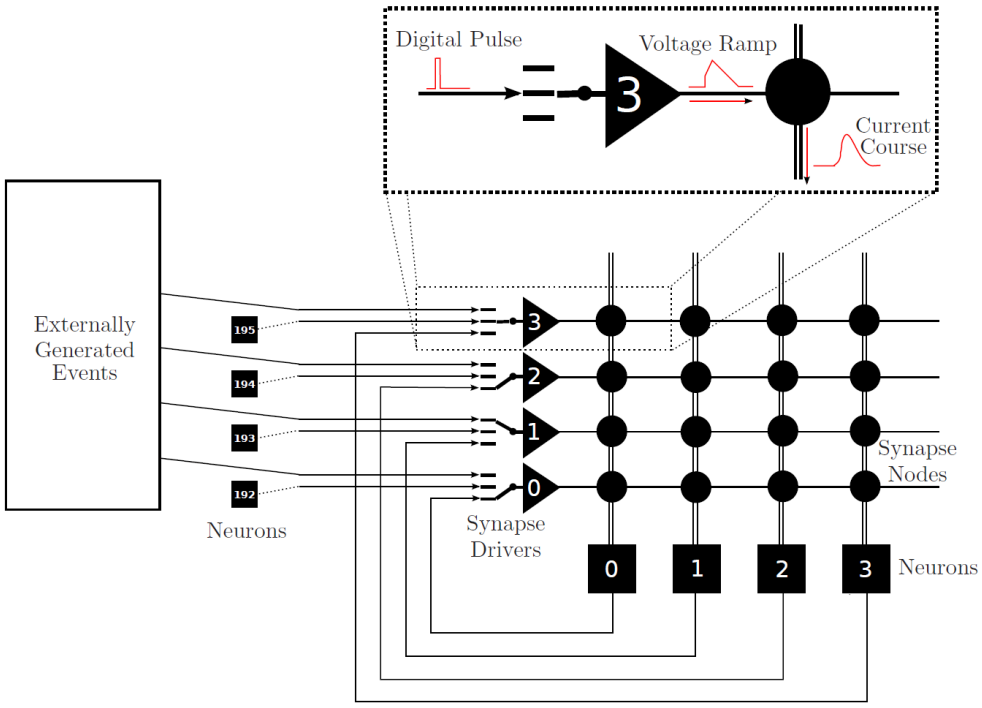

Figure 6.9: The schematic shows the communication infrastructure of the Spikey system. The user defines network topologies on the host computer via the PyNN-backend (see Section 2.5). The software interface transmits the configuration data. The FPGA module stores the network configuration onto the DAC/ADC modules to set up the emulation. During the emulation run, the results are stored to the 512MB memory (RAM) and retrieved at the end of the emulation. Figure is taken from *Pfeil et al.* (2013).

To make such measurements possible, it is necessary to provide a translation between parameters in the biological network specification and in terms of hardware settings. Ideally, the hardware settings are automatically read out and deployed according to the network specification. Finding such a mapping between hardware and biological networks will is referred to as *calibration*. The calibration procedure involves sweeping over an extensive range of parameters to find agreeable settings for network specifications. As a result, fixed-pattern noise can be compensated for in part by finding configurations for each neuron and synapse to reproduce the desired values. This is exemplified in Figure 6.8 for the calibration of a PSP shape. Calibration settings are embedded in the software framework of the Spikey chip and can be loaded automatically for each individual emulation run (*Brüderle et al.*, 2011). We will outline the infrastructure to run experiments on the Spikey chip in the next section.

Emulation interface of the Spikey chip

To run network emulations on the hardware, a communication infrastructure is in place to transmit the target network configuration (Figure 6.9). This infrastructure consists of field-programmable gate arrays (FPGA) that relay configuration data in digitized form onto the random access memory (RAM) on the chip. Before the emulation, the data is read out and set to analog signal values via so-called digital-to-analog converters (DAC). Here the values of the DACs are set according to the calibration to ensure the best possible settings. As soon as the network is set up, the network emulation is initiated. During the runtime the spike times of all neurons are routed as digitized events via FPGA and stored on the RAM of the chip. It is possible to record membrane potential and conductance traces of eight different neurons on a chip. After the network emulation, the stored data can be transmitted to a local host computer.

The described usage of the Spikey chip relies on a comprehensive software framework to access, control and store the network data in an easily manageable database. The application programming interface (API) of Spikey was adapted to the simulation environment of PyNN, which we also use to set up and run neural networks (see Section 2.5). This front-end allows to run emulations on the Spikey chip with the same interface as the PyNN-supported simulators (*Davison et al.*, 2010).

6.4 Boltzmann machines on the Spikey chip

The potential benefit of network experiments with a 10^4 speed-up compared to real-time is a substantial advantage compared to conventional simulations. Due to its size and low energy consumption, the chip has potential to serve as a portable device solving numerous tasks (*Pfeil et al.*, 2013). As we have noted before, to implement network models on the neuromorphic chip, the model has to be adapted to analog computing devices.

We will present the implementation of LIF-based Boltzmann machines on the Spikey chip and evaluate the potential of solving neuromorphic inference tasks. The following results were preceded by collaborations of several years in the VISIONS group. It took several years of hardware testing, software development, calibration and hardware-tailored development of network models to achieve stochastic computing on the platform. The subsequent implementation of Boltzmann machines has been done in collaboration with David Stöckel, Thomas Pfeil, Mihai A. Petrovici, Johannes Bill and Alexander Kononov. In particular, material has been taken from the Bachelor's thesis *Stöckel* (2015), which was co-supervised by the author of this thesis.

The starting point of Spikey-based Boltzmann machines is the implementation of stochastically firing LIF neurons and a fast membrane potential, as argued in Section 5.4. Therefore, as a prerequisite, we will aim to produce the correct firing statistics in the high-conductance state of LIF neurons on Spikey, as described in Chapter 4. Essentially, we want to use LIF neurons with high-frequency Poisson input and reproduce a sigmoidal activation function.

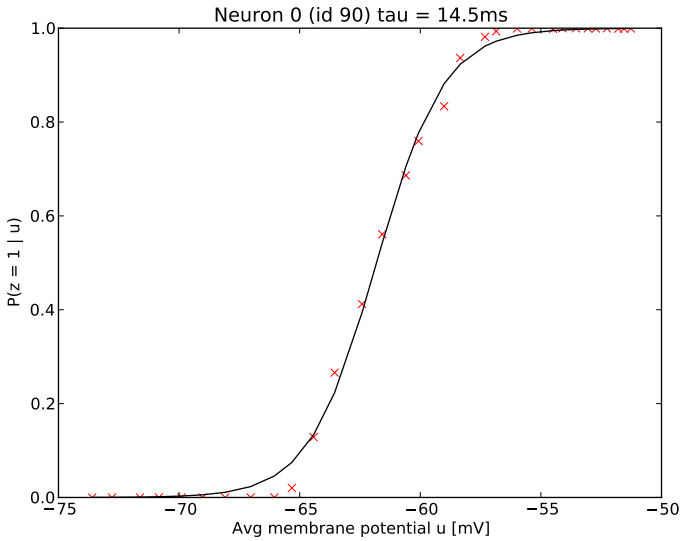

Figure 6.10: The single-neuron activation function of a LIF neuron on Spikey was recorded in a high-conductance state driven by Poisson noise stimuli. The red crosses show the firing probability of the LIF neuron. The black curve is a sigmoidal fit $\sigma(u) = \frac{1}{1+\exp(-u)}$. Although the onset and saturation is different, we see that the general shape of both results corresponds well. This indicates that the single-neuron firing statistics on the Spikey chip is compatible to the sampling framework. Figure is taken from *Bill and Petrovici* (2011).

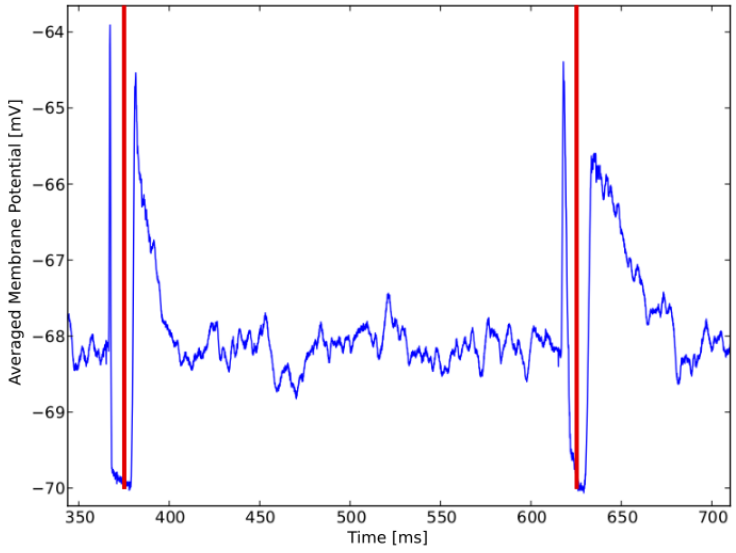

Figure 6.11: The membrane potential trace is recorded from a neuron S_1 in the high-conductance state and averaged over multiple runs. The neuron is connected to neuron S_2 via symmetric excitatory weights. As soon as S_1 spikes at $\propto 371\,\text{ms}$, it transmits a strong excitatory pulse to S_2. Neuron S_2 responds with an excitatory spike, arriving at the S_1 membrane potential at $\propto 374\,\text{ms}$ (red line). The time difference results from the signal travel distances $S_1 \longrightarrow S_2$ and back, $S_2 \longrightarrow S_1$. This implies a synaptic delay of $\approx 1.5\,\text{ms}$. The same phenomenon can be observed during the second spike at $\propto 625\,\text{ms}$. This delay is not accounted for in the software model, which assumes instantaneous synaptic transmission (see Figure 5.2). Figure is taken from *Bill and Petrovici* (2011).

In software simulations, the stochastic input has been implemented by applying presynaptic spike trains with Poisson-distributed spike times. To supply networks on hardware, we generated the Poisson spike trains on the host PC and applied them as presynaptic input via external connections to the Spikey LIF neurons. The Poisson input rate was chosen $\nu_{\text{exc}} = \nu_{\text{inh}} = 400\,\text{Hz}$, high enough to elevate the LIF neuron to a high-conductance state, ensuring a sufficiently fast effective time constant $\tau_{\text{eff}} = \frac{C_{\text{m}}}{g_{\text{l}}+g^{\text{syn}}}$. The resulting activation function is shown in Figure 6.10. The Spikey measurements (red crosses) can be fitted by a logistic function $\sigma(u) \approx \frac{1}{1+\exp(-u)}$ (black curve), which is a prerequisite to apply the LIF sampling framework described in Chapter 5. We identify a good correspondence between the experimental data and a logistic fit and conclude that single LIF neurons on the chip are capable to reproduce the necessary firing statistics to perform sampling. Technical details to these single-neuron simulations can be found in *Bill and Petrovici* (2011).

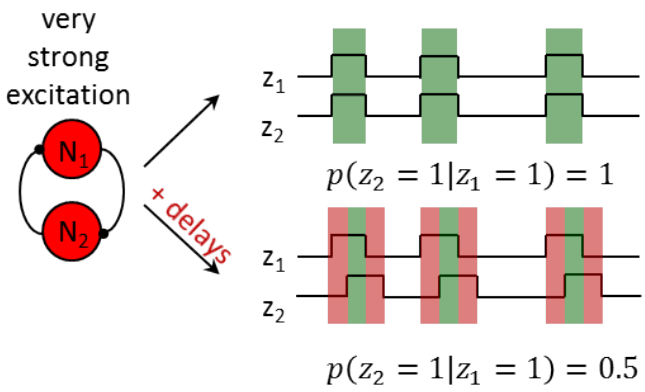

Figure 6.12: The schematic shows a two-neuron Boltzmann machine with a very strong excitatory weight. The upper case shows perfectly correlated states z_1 and z_2 due to the strong excitation. The lower case shows the states with an extreme synaptic delay, $\tau_{\text{del}} = \frac{1}{2}\tau_{\text{ref}}$, to illustrate the impact on the sampled distribution. We see that the delay introduces erroneous states (red bars), severely impairing the joint firing probability. Figure is taken from *Petrovici* (2016).

As a next step, we evaluate the dynamics of connected neurons with symmetric weights to validate whether neuron interaction on Spikey is compatible with Boltzmann interaction of sampling networks. We connect neurons S_1 and S_2 with a symmetric excitatory weight, building a two-neuron Boltzmann machine to study the simplest possible case. The resulting membrane potential trace of one of the two neurons can be seen in Figure 6.11. The membrane potential of S_1 has been averaged over multiple runs to decrease the effect of temporal noise. It shows two output spikes with ensuing refractory potentials at $\approx 70\,\text{mV}$. These two spikes trigger output spikes at the site of the other LIF neuron, S_2. The neuron S_2 immediately spikes and excites S_1, since the Boltzmann weight is symmetric.

During the refractory phase of $\tau_{\text{ref}} \approx 14\,\text{ms}$ we see another spike signal after $\approx 3\,\text{ms}$. This signal indicates the arrival of the spike transmitted from S_2, implying that there is a delay between spike signals of S_1 and S_2 of about 1.5 ms. These delays are a consequence of the physical model itself and are caused by signal integration times in the synapse driver nodes. This is a conceptual difference to simulations, where synaptic transmission can be lowered to the duration of a time step. Since time steps can be set arbitrarily short in simulations, it is possible to set synaptic transmission to a values negligible compared to neural dynamics. In a physical model, transmission delays are inevitable, since information is transmitted in finite time. We can illustrate the impact of hardware delays on the sampling performance by means of a simple thought experiment. In the above described emulation run, we observed a delay $\tau_{\text{del}} \approx 1.5\,\text{ms}$ in biological time. Effectively, this means that during each refractory state ($z = 1$), the network suffers a

relative systematic error. An extreme case of this effect is illustrated in Figure 6.12, where we see that the resulting systematic error distorts the probability distribution of the network.

To mitigate this systematic error, the ratio $\frac{\tau_{\text{del}}}{\tau_{\text{ref}}}$ needs to be as small as possible. Since $\tau_{\text{del}} \approx 1.5\,\text{ms}$ is a physical constraint bound to the hardware architecture, only the refractory period can be changed. On Spikey, refractory periods are set by a control current `icb`. The lower this current, the larger τ_{ref} becomes. Naturally, the lower a current becomes, the larger shot noise effects will be (*Schottky*, 1918). Consequently, a decrease of `icb` currents results in imprecise refractory times (*Bill and Petrovici*, 2011) in the range of $\propto 10\,\text{ms}$ (Figure 6.13). Since refractory times τ_{ref} in higher ranges (i.e., lower `icb` currents) become unreliable, LIF sampling on Spikey becomes unfeasible.

Conclusively, these experiments require a different approach to make LIF sampling possible on the Spikey chip. Since the problem was identified in the short refractory times on the chip, we will modify the networks to prolong the refractoriness of the neurons.

6.4.1 Sampling on hardware using Synfire chains

Since the single neuron LIF dynamics on Spikey yield the correct firing statistics (see Figure 6.10), the synaptic noise input does not need to be changed. However, during interaction between Boltzmann units, hardware delays introduce systematic errors into the sampled distribution. To eliminate this systematic deviation, we aim to increase the duration of the active state, $z = 1$. Since the duration of the active state is the refractory time, prolonging the $z = 1$ state duration decreases the ratio $\frac{\tau_{\text{del}}}{\tau_{\text{ref}}}$.

Essentially, the state $z_k = 1$ of a sampling neuron S_k is defined by two characteristics:

- Neuron S_k is not allowed to emit an action potential while being in state $z_k = 1$. It is fixed to the reset potential ρ.
- Neuron S_k emits a spike, which triggers a switch into state $z_k = 1$. At the same time, this spike causes synaptic interaction for the duration of the refractory period with neurons with synaptic weights $\boldsymbol{W}$.

Both properties are consequences of theoretical derivations explained in Section 5.1. Our goal is to set up a group of neurons to inhibit S_i for a time period T_{ref}, which is configured to be significantly longer than delay τ_{del}. This defines the new state $z_i = 1$ for the sampling neuron S_i,

$$z_i(t) = \begin{cases} 1, & \text{neuron has spiked during } (t - T_{\text{ref}}, t) \\ 0, & \text{else} \quad . \end{cases} \tag{6.18}$$

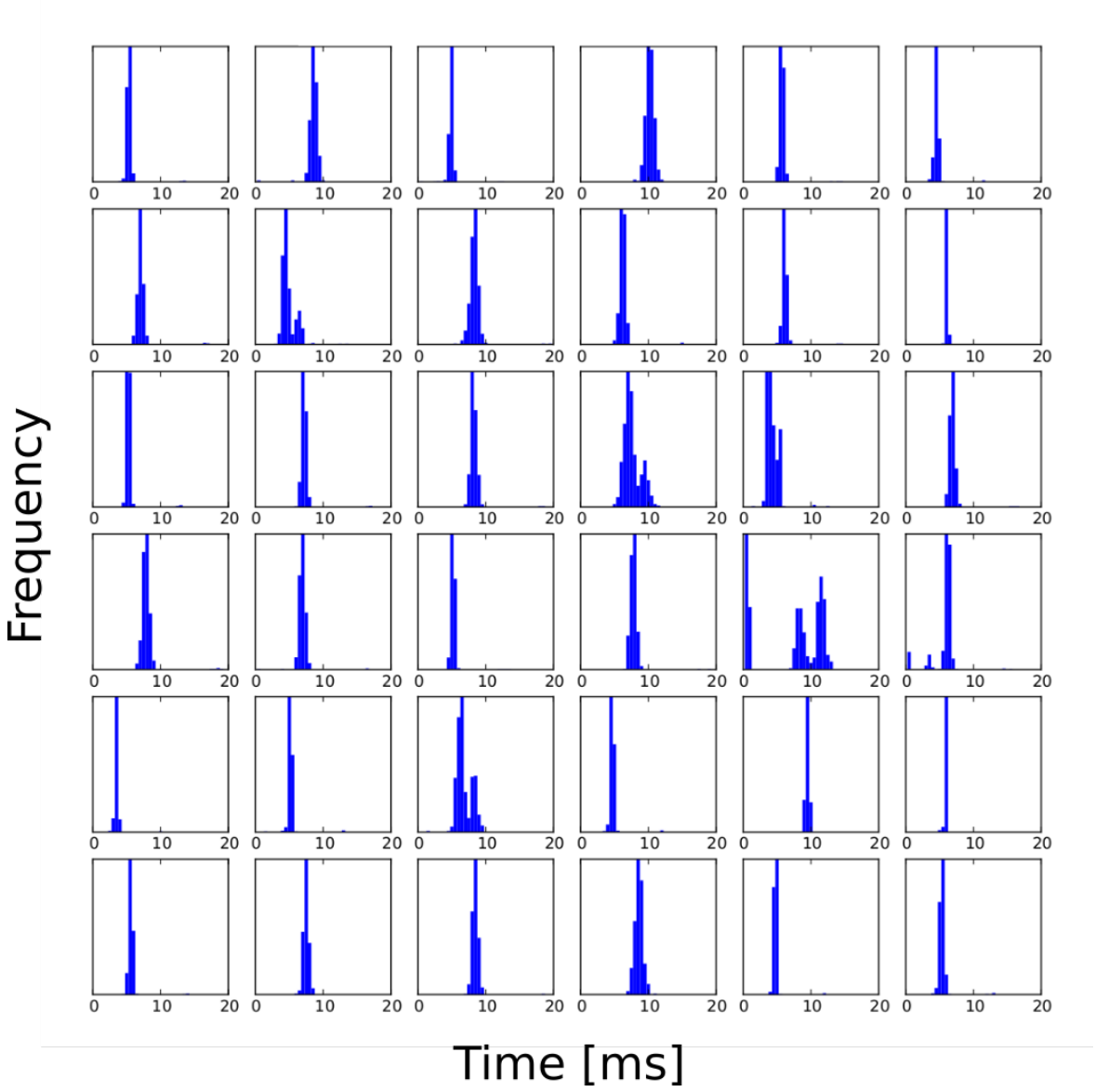

Figure 6.13: The histograms show refractory times of 36 randomly chosen neurons on the Spikey chip. The histograms show trial-to-trial variation of τ_{ref} from multiple runs. The τ_{ref} settings on Spikey are controlled by the so-called `icb` currents. We see a tendency for an incrase of trial-to-trial variability for larger τ_{ref} values in the histograms, as `icb`-related shot-noise becomes more significant. The delays around 10 ms show high variability. Figure is taken from *Bill and Petrovici* (2011).

This definition is analogous to the one provided in Section 4.4.1. The only difference is that T_{ref} is not the refractory period resulting from the LIF spiking mechanism, but an inactivity duration enforced through network architecture.

The sampling neuron S_i is in the high-conductnce state and spikes stochastically analogously to the simulation setup. We introduce an excitatory neuron population pool E_i^1, which is activated by S_i. Its purpose is to respond by activating inhibitory populations I_i^1. These activated inhibitory neurons, in turn, silence the sampling neuron S_i with inhibition. The neurons in the pools E_i^1 fire synchronously to activate I_i^1 and trigger a synchronous inhibitory burst to decrease the membrane potential of S_i. This ensures that S_i does not fire for a certain period of time. This inactivity time is our effective refractory period T_{ref}. To set the duration of T_{ref}, we connect E_i^1 excitatorily to another group, E_i^2. The neurons in this pool also synchronously fire a burst, activating I_i^2 to keep S_i inhibited, preventing it from spiking. We can prolong this feedforward chain of pools, E_i^c and I_i^c, to set T_{ref} to a desired length. For our experiments, the desired length of T_{ref} is around 15 ms. For this value, the ratio $\frac{\tau_{\text{del}}}{T_{\text{ref}}} \approx 0.1$ is small enough to achieve good sampling results.

This feedforward mechanism is known and defines a so-called *Synfire chain* (*Abeles*, 1991; *Diesmann et al.*, 1999; *Abeles et al.*, 2004) because it incorporates a chain of neuron pools with length n, where all neurons in pool E_i^c or I_i^c fire synchronuously to propagate a signal. This mechanism has been observed in biological studies (*Reyes*, 2003) and we will construct it on the chip with the purpose to obtain stable effective refractory times T_{ref}.

In our network, the chain propagates the signal that the sampling neuron S_i has fired and is to set $z_i = 1$. The length of the chain, n, determines the length of T_{ref}. At the end of the refractory period, S_i needs to be set to $z_i(t = t_{spk} + T_{\text{ref}}) = 0$ with a membrane potential that is elevated to its mean value. Therefore, a last excitatory neuron group E_i^{n+1} at the end of the chain is activated by E_i^n to excite S_i at the end of T_{ref}. This mechanism is described in Figure 6.14. Note that the binary random variables z_i are controlled by the activity of S_i and not by activity of pools E_i^c and I_i^c.

So far we have described how to prolong the refractory time of the neurons, but the setup also requires a new way to mediate synaptic transmission during the activity states $z_i = 1$. Since the duration of $z_i = 1$ is now controlled by pools E_i^c and I_i^c, we will use them for synaptic interaction during T_{ref}. Instead of direct interactions between S_i and S_j like in the software model, the Boltzmann weights W_{ij} are mediated by pool neurons $E_i^c \to S_j$ (W_{ij} excitatory) or $I_i^c \to S_j$ (W_{ij} inhibitory) (see Fig. 6.14). This ensures that synaptic interaction occurs only during T_{ref}, since E_i^c and I_i^c are only active during this time frame.

We have implemented this network architecture on the Spikey chip and evaluated the sampling performance. We set up a three-neuron Boltzmann machine with three sampling neurons S_1, S_2, S_3, defining their random variables z_1, z_2 and z_3. The refractory mechanism was implemented with pool populations consisting of 183 neurons in total, with 61 auxiliary neurons per random variable. The high number of auxiliary neurons ensured a high degree of robustness and accuracy of synchronous firing during

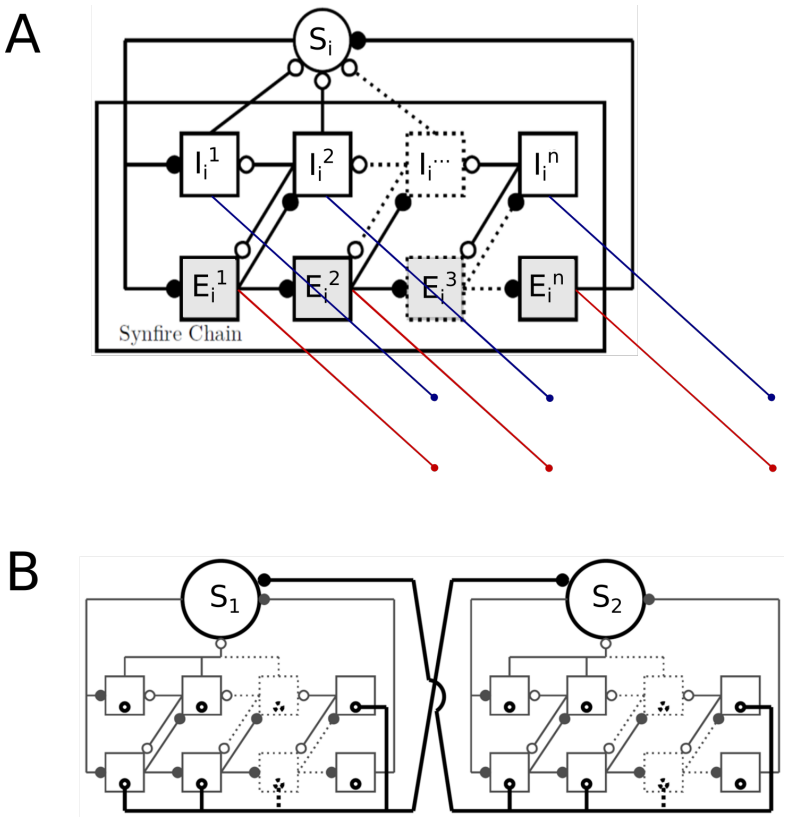

Figure 6.14: **(A)** The Synfire chain architecture is implemented to prolong the effective refractory duration T_{ref} of LIF neurons. The duration is controlled by the chain length n of auxiliary pools E_i^c and I_i^c. The chain is activated as soon as sampling neuron S_i spikes. It activates both E_i^1 and I_i^1. The inhibitory pool I_i^1 silences S_i with a spike burst. The excitatory pool E_i^1 triggers I_i^2 to deactivate I_i^1 and inhibit S_i again. Additionally, E_i^1 also triggers E_i^2 to activate the next pair E_i^3 and I_i^3 to prevent S_i from spiking. The feedforward propagation continues until the last excitatory pool E_i^{n+1} is activated. It eventually terminates the inactivity of S_i by exciting the sampling neuron to leave its inhibited state. **(B)** The Boltzmann connections between two sampling neurons S_1 and S_2 are established via unilateral pool connections from I_i^c (inhibitory weights) or E_i^c (excitatory weights). The pool neurons mediate interaction in synchronous bursts during T_{ref}. Figure is taken from *Stöckel* (2015).

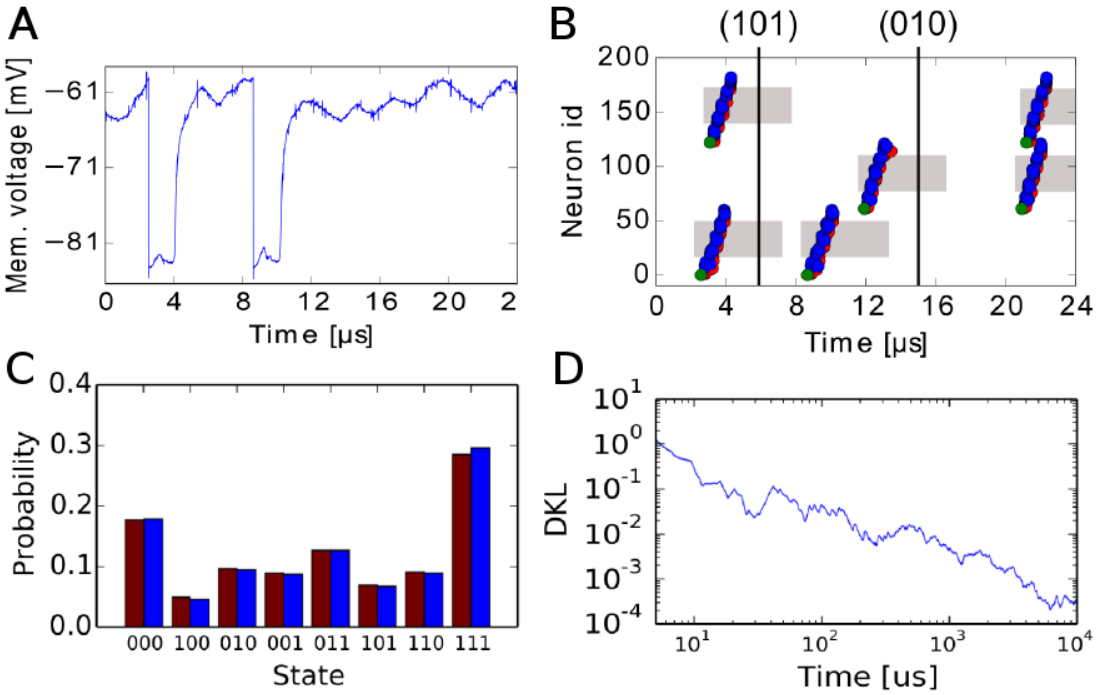

Figure 6.15: Sampling with LIF neurons on the Spikey chip. **(A)** The membrane potential of a sampling neuron which spikes on two occasions and is inhibited by pool neurons, keeping the membrane potential below −81 mV. Note that the time scale is in real-time, which translates into biological network time with the speedup factor of 10^4. **(B)** The spike times of this sampling neuron can be seen in the lowest row, marked as green dots. The state $z = 1$ is triggered at the spike time of the sampling neuron (green) and continues as long as excitatory pool neurons (red) activate the inhibitory pool neurons (blue) to inhibit the sampling neuron. The states of all three random variables are marked grey and exhibit a very robust refractory time of $T_{\text{ref}} \approx 1.5\,\mu\text{s} \cdot 10^4$. **(C)** The network-sampled probability distribution (red) is compared with the target Boltzmann distribution (red). The sampled probability distribution approximates the targeted one very accurately. **(D)** The $\mathrm{D}_{\mathrm{KL}}\left(p_{\mathrm{HW}}(\boldsymbol{z}) \,||\, p_T(\boldsymbol{z})\right)$ distance between both distributions is shown for an increasing sampling time in real-time on a log-log scale. In the context of the anytime computing paradigm, we can read out the sampled states anytime and accuracy increases with the runtime.

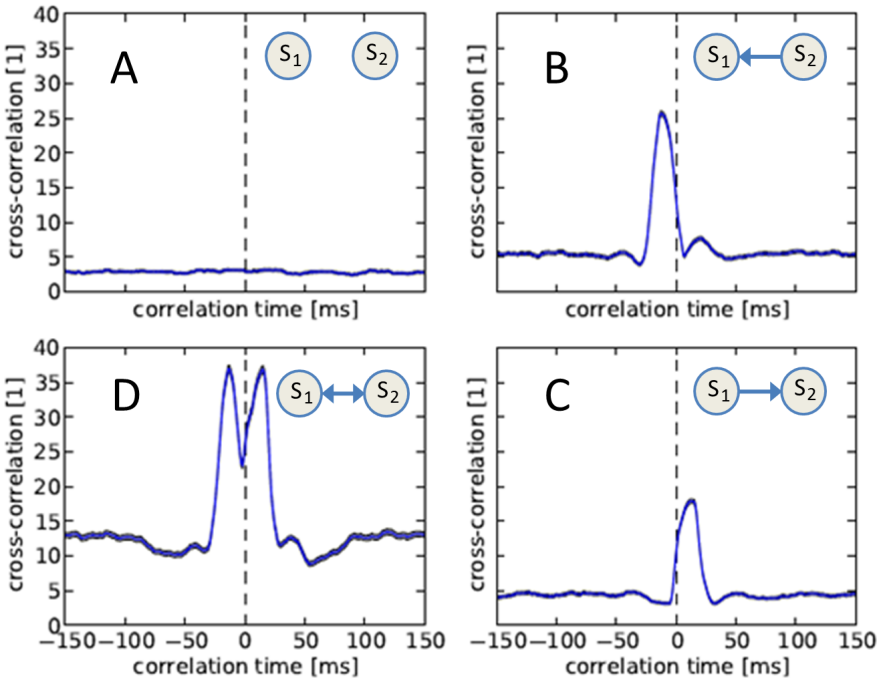

Figure 6.16: The correlation coefficient of two membrane potential traces U_1 and U_2. **(A)** The sampling neurons S_1 and S_2 are not connected, therefore there is no correlation pattern. **(B)** The increase of cross-correlation is due to an unilateral connection, W_{21}, from S_2 to S_1. **(C)** A connection from S_1 to S_2 by W_{12} also results in a visible cross-correlation pattern. **(D)** The connectivity between S_1 and S_2 is calibrated to a fixed lenght and amplitude. The symmetry in the correlogram indicates that the weights W_{12} and W_{21} are identical and the interaction is symmetric. Figure is taken from *Stöckel* (2015).

the refractory times. The described experiments were conducted on the Spikey v4 chip, where 192 neurons were usable. This pool size limits the sampling networks on Spikey to three random variables.

Calibration of the Spikey neurons

To set up the described network architecture, the neuron and synapse circuits on Spikey need to be configured. The main challenge consists in calibrating all 4-bit weights of the 183 E^c and I^c neurons. In contrast to many other neural networks, Boltzmann machines require a mathematically precise calibration to map the BM parameters $(\boldsymbol{W}, \boldsymbol{b})$. A particularly sensitive factor is the symmetry of the Boltzmann weight matrix, which needs to be incorporated into the connections of auxiliary pools. Due to the large number of auxiliary neurons and synapses, there are too many degrees of freedom to

calibrate each one of the pool neuron weights.

Due to this complexity, we did not configure hardware settings to achieve a mapping of BM parameters, but set the hardware parameters based on correlated membrane potentials during firing activity. We enabled synaptic interaction from pool neurons E^c (excitatory) and I^c (inhibitory) to sampling neurons S_i and S_j,

$$E_i^c \rightarrow S_j \text{ (excitatory from i to j)} \quad , \tag{6.19}$$
$$E_j^c \rightarrow S_i \text{ (excitatory from j to i)} \quad , \tag{6.20}$$
$$I_i^c \rightarrow S_j \text{ (inhibitory from i to j)} \quad , \tag{6.21}$$
$$I_j^c \rightarrow S_i \text{ (inhibitory from j to i)} \quad . \tag{6.22}$$

In Figure 6.16 we see the measured cross-correlations of two membrane potentials of S_i and S_j to calibrate an excitatory connection W_{ij}. These cross-correlation measurements have been done for all connections between all three random variables. The membrane cross-correlations have a relationship to the joint firing probability $p(z_i = 1|z_j = 1)$ of sampling neurons S_i and S_j. We adjusted the hardware synapse settings until the cross-correlations yielded correct firing statistics. The resulting PSPs induced by auxiliary pools in Figure 6.15 are similar to the rectangular shapes of the ANM, which served as the starting point of the LIF sampling framework (see Section 5.2.2). Detailed descriptions on the iterative weight calibration can be found in *Stöckel* (2015).

After the calibration is performed, we have run the three-RV network on the hardware and have evaluated its sampling performance. We used the DKL to measure the distance between the target Boltzmann distribution and the one sampled by the network on Spikey. In Figure 6.15C we see a predefined Boltzmann distribution (blue bars) which the network is set to sample from. The quality of sampling increases, since the DKL decreases on a logarithmic scale in Figure 6.15D. It shows sampling on neuromorphic hardware in real-time, measured in μs. Indeed, the real-time of $10^4\,\mu$s$(= 10\,\text{ms})$ is required to emulate network dynamics of 100 s biological time. The logarithmically decreasing DKL highlights the *anytime computing* aspect, which is a conceptual property of sampling as an inference method. The approximation of the target distribution becomes better with increasing sample size over time and can be evaluated at any point in time, in contrast to to other existing inference algorithms (e.g., belief propagation (*Pearl*, 1982)).

The successful demonstration of stochastic inference can be extended to host a larger number of random variables on the Spikey v5 chip, which has 384 available neurons. However, even with a more efficient allocation of neuron pool resources, the refractory mechanism impedes an implementation of large-scale networks that have been presented in Section 6.2.1. With only 384 available neurons, the experiments on Spikey would have a very limited practical applicability. Still, the demonstrated networks can be used as conceptual studies to identify important discrepancies between hardware design and

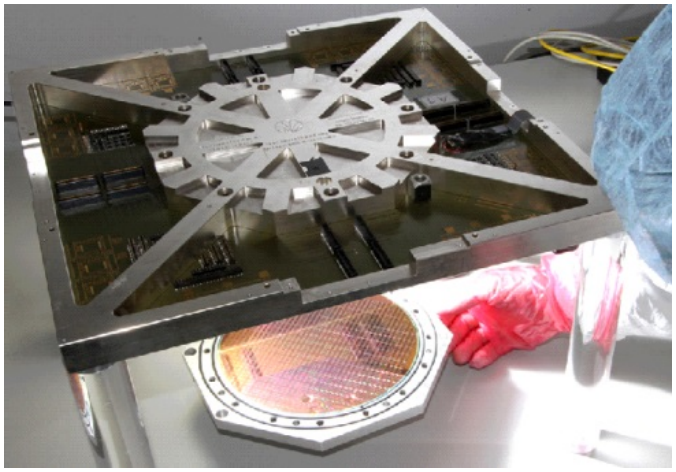

Figure 6.17: A photograph of the BrainScaleS wafer-scale system. It is a VLSI neuromorphic system which includes 384 HICANN chips, several FPGA communication boards and a power supply infrastructure. The hardware is operated with a 10^4 speed-up of biological processes due to the high integration density and implements AdEx networks. Image courtesy by Dan Husmann.

parameter requirements for stochastic networks. For instance, the problem of hardware delays was revealed in the course of many experiments and has influenced future hardware development to a great extent.
Consequently, the successor of the Spikey chip has been specifically designed to host large-scale networks and has been decisively influenced by countless experiments and research effort on the Spikey project.

6.4.2 Outlook: wafer-scale stochastic computing

The BrainScaleS neuromorphic system (BSS system) is a mixed-signal VLSI system developed during the BrainScaleS project (*BrainScaleS*, 2012). The system itself has undergone several revisions and its development is ongoing during the *Human Brain Project* (*HBP SP9 partners*, 2014). Although it is an analog computing system like the Spikey chip, it is devised for large-scale of networks. The system is operated as a full silicon wafer, which has a size of $\approx 21\,\mathrm{cm}$, governs 196608 neuron circuits and 44 million synapses (*Schemmel et al.*, 2010). After the fabrication process of the wafer, the single neuromorphic chips were not cut out but are kept on the original wafer. In an additional postprocessing step, wafer-wide connections are established. This approach simplifies the communication and energy supply infrastructure of the individual wafer components, which in turn lowers the energy consumption immensely.

The neuromorphic circuitry on the wafer is structured in blocks of neuromorphic chips, the so-called *High Input Count Analog Neural Networks* (HICANNs) (6.17). Each HICANN chip contains 512 so-called *dentritic membranes* (denmems), whose name is

derived from biological dendrites. Multiple dendrites are interconnected to construct a "neuron" circuit. Here, like on the Spikey chip, the inherent time constants of the electric components, $\tau_m = R \cdot C_\mathrm{m}$ are also in the order of microseconds and yield a 10^4 speed-up compared to biological dynamics.

Despite the architectual differences to the Spikey chip-based system, the BSS system also performs analog computing. Therefore, it is also sensitive to fixed-pattern noise and temporal noise. Correspondingly, calibration also needs to be conducted to be able to work with the system. Due to increased number of parameters and increased complexity of circuits, the calibration procedure on the wafer is still ongoing. A detailed and more technical treatment of these topics can be found in *Schwartz* (2012); *Schmidt* (2014); *Koke* (2017). In our work, we will focus on the potential of the BSS system in terms of stochastic inference. In particular, the implementation of large-scale networks presented in Section 6.2, in principle, appears to be viable on the BSS system, since the refractory mechanism allows longer stable τ_ref and has sufficient neuron and synapse resources available.

To implement deep learning architecture, we have already explored a simple learning method based on so-called contrastive divergence (CD). This learning method can be seen as a starting point for more elaborate training techniques. So far we have trained our Boltzmann machines on abstract neuron models and then transferred the Boltzmann parameters to LIF networks. To train LIF networks directly, stopping the network to update to the trained network parameters would be necessary. This describes so-called *offline learning*. On the BSS hardware, ongoing development is focused on embedding the hardware ciruits into such a training scheme. In the currently developed *in-the-loop* calibration framework in the VISIONS group, a network target functionality is defined and the network performance is measured during each run. According to a predefined functionality metric, the hardware parameters are adjusted iteratively to approximate the target functionality. Future platforms in the HBP are designed to train LIF networks without stopping the emulation after each learning step. With the recently developed plasticity processor "Nux" in place, the networks could be able to update parameters without interrupting the emulation run (*Friedmann*, 2013; *Hartel*, 2016). Such *online learning* mechanisms can further increase the efficiency of running neuromorphic networks. In particular, LIF networks could adapt the biological *spike-timing dependend plasticity* (STDP) mechanism to apply it on deep learning topologies in a noisy environment (*Bi and Poo*, 1998). Theoretical studies of such an approach exist and show promising results (*Neftci et al.*, 2015; *Weilbach*, 2015). Additional approaches may include the modulation of noise input to facilitate state transitions. This describes a tempering mechanism similar to AST, allowing LIF networks to learn data sets more efficiently.

In our work we will focus on a different and yet fundamental aspect of physical models, namely the embedding of stochasticity in the network environment. In software, this does not pose a problem, since it is always possible to seed a random number generator to provide pseudo-stochasticity and reproducibility of results. In the networks

we described, the source of randomness for LIF networks were temporal Poisson processes, which supplied each of the neurons to induce a stochastic high-conductance state. Large-scale models like the RBM described in Section 6.2 potentially require thousands of individual Poisson sources with rates of approximately 400 Hz to maintain the vital stochasticity in the network. For neuromorphic applications, this stochastic input has to either be input into the device or generated on the chips itself. Both options pose challenges for computational neuroscience.

In the next chapter we will discuss these challenges and propose solutions. In developing these solutions, we will address the question whether neural networks might even perform stochastic computations entirely without external noise.

7 Stochastic neural networks without stochastic input

In the course of this work, we have investigated two types of networks. First, we have studied the computational properties of cortical layer networks in Chapter 3 where biological neural architectures were transferred into AdEx topologies to perform stochastic inference. Due to the difficulty in interpreting the network dynamics for high pattern densities, we have started to develop a stochastic inference framework by characterizing the response of LIF neurons, which are easier to describe than AdEx-based neurons. The description of the LIF neural response to synaptic stimuli was used to build LIF-based Boltzmann machines (BMs) in Chapter 5. An important component common in both inference networks, the AdEx-based cortical networks and the LIF-based BMs, was presynaptic noise that stimulated each network neuron. In both types of networks, each neuron received completely independent Poisson input generated by random number generators in software. For these stochastic networks we have emphasized the benefits of using neuromorphic computing. Like any physical system, neuromorphic systems have bandwidth limitations, setting an upper bound on potential signal transmission. Considering the amount of stochastic input that a large-scale stochastic network requires for operation, it becomes difficult to support such networks with uncorrelated noise exclusively from external sources. First we will discuss this problem from a practical point of view for the specific case of the BrainScaleS wafer system and present an approach to implement LIF-based BMs despite this limitation. Although this appears to be an exclusively hardware-related problem setting, it will give us deeper insight into the role of noise sources in spiking neural networks. This will allow conclusions that relate to general principles of computational networks in general.

7.1 Stochasticity on neuromorphic hardware

We concluded the previous chapter with an introduction to the BrainScaleS wafer system and will proceed now by looking at the two basic possibilities to supply the networks with stochasticity.

- **Noise supply from external host:** The stochastic input is generated in software on a host computer and transmitted to the neuromorphic device and then relayed via digital communication into the synaptic circuits. This is identical to our approach on the Spikey (Section 6.3.1), where Poisson noise was generated on a host PC to supply the emulated BMs on the chip.

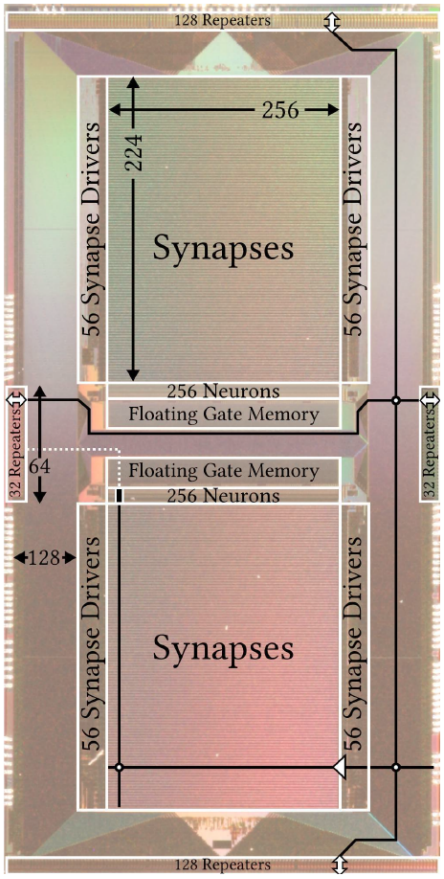

Figure 7.1: The photograph shows the HICANN chip, which is sized 5x10mm^2. The different building blocks are marked. The largest building block is made up of synaptic arrays, allowing flexible connectivity between hardware neurons. The 512 neuron circuits are placed at the center edges of the synapse arrays. The Layer 1 network implements the digital spike transmission and relays the signals between neurons and synaptic arrays. The Repeaters at the edges of the HICANN are connection nodes to other HICANN chips. Figure is taken from *Jeltsch* (2014).

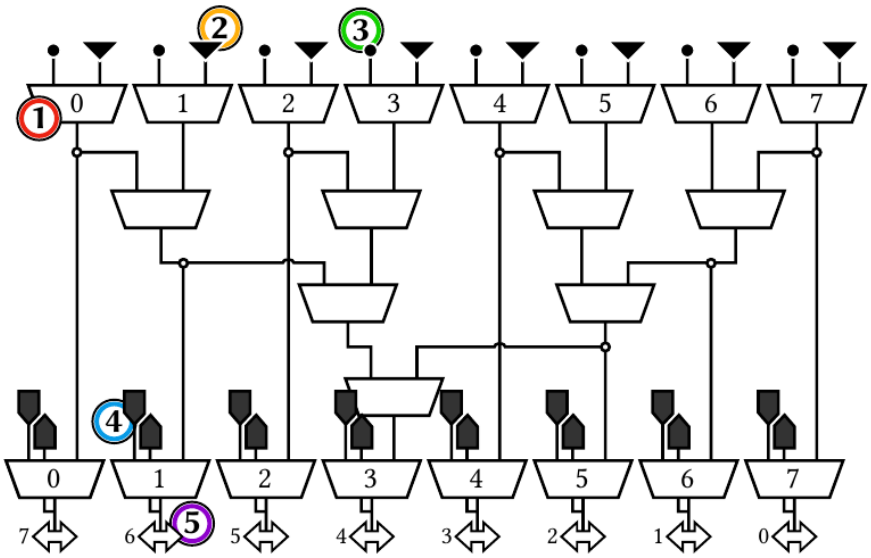

Figure 7.2: The schematic shows the merger tree of the HICANN chip (Figure 7.1). The merger tree is part of the Layer 1 routing architecture and merges incoming digital signals from different sources. The event flow starts at the top and follows to the bottom. Events from the HICANN-internal background generator (2) and other HICANN neurons (3) are merged or forwarded in (1). External events entering the routing network are merged in (4). Figure is taken from *Jeltsch* (2014).

- **Noise generation on the neuromorphic system:** The noise is generated on the neuromorphic chips (HICANNs) and distributed to the network circuits.

The first option is the preferable one, since it allows to control all noise parameters in software. After generating the spike trains in software, the generated spike trains are transmitted via ethernet to the wafer. On the wafer, the spike trains are relayed via FPGA to the neuron circuits on the HICANN chips through the so-called *merger tree*. In Figure 7.2, we see a merger tree routing schematic. The externally generated spike events can be fed through the FPGA channels in the DNC mergers (4). They are then relayed to the HICANN denmems (dendritic membranes). The data throughput of these channels is limited to 24 MHz per HICANN in real-time. For bandwidths $\geq$ 20 MHz, event data loss occurs, i.e., a portion of externally transmitted spikes is lost during transmission in a non-deterministic way. On a biological time scale, the maximum bandwidth per HICANN amounts to $\nu_{\max} = 25 \cdot 10^{-4}\,\text{MHz} = 2500\,\text{Hz}$ (*Koke and Vogginger*). If we modestly assume that we can use approximately 100 well-suited neurons per HICANN, we would have a maximum noise input rate of $\frac{2500\,\text{Hz}}{100} = 25\,\text{Hz}$ available for each neuron. If we separate this total bandwidth into an inhibitory and excitatory part, we would achieve a bandwidth of 12.5 Hz per source. Even with such a low number of usable neurons (100), the input bandwidth is not sufficient to sustain a high-conductance state in each neuron and establish the necessary stochastic dynamics in the system. Therefore, running networks with high-conductance neurons driven by externally pre-generated noise is difficult to accomplish.

The second proposed option is to generate noise input on the wafer system itself. In Figure 7.2, a background generator (3) is marked, whose purpose is to generate pseudo-random numbers during each hardware clock cycle. The generator is a *Linear feedback shift register* (LFSR) and consists of a logic gate that calculates a linear function of an input bit. The output of this linear function is set as the new binary state and depends on the current state. A spike event is generated if a state gives a larger number than a number defined by the binary state. This binary state serves as a spiking threshold of the background source. Consequently, the rate of a stochastic process is then set by this predefined threshold and can be adjusted to achieve the desired stochastic rate. The number of states that can be generated is bounded by the length of the register. On the HICANNs, the length of each register is 16 bit, yielding $2^{16} - 1$ possible states [1]. After this deterministic state sequence has been generated, the deterministic cycle repeats the same $(2^{16} - 1)$-state sequence. We can briefly summarize the properties of the LFSR generators on the HICANN.

- **Total spike rate of LFSR-generated noise:** There are 8 LFSRs on one HICANN and each LFSR can generate spike rates of up to 5000 Hz in biological network time. For 100 functional neurons per HICANN, this results in a spike rate of $\frac{40000\,\text{Hz}}{100} = 400\,\text{Hz}$ (*Koke*, 2016). These frequencies are high enough to induce a stochastic high-conductance state.

[1] Here we exclude the zero-state because it generates identical states.

- **Mathematical properties of the LFSR output:** Up to this point we have only considered Poisson processes as viable stochastic inputs. Although LFSR-generated temporal sequences are autocorrelated, existing work shows that they do not impair sampling statistics (*Großkinski*, 2016).
- **Length of the LFSR sequence:** We can calculate how long networks on the hardware are supplied with noise by LFSRs in terms of network runtime. For a $(2^{16}-1)$-state sequence and a HICANN clock of $\nu_{\text{op}} = 100\,\text{MHz}$, a number of 65535 states is generated during a biological network runtime of $65535 \cdot \nu_{\text{op}} \cdot 10^4 = 6.56\,\text{s}$. This means that the LFSR-generated pseudo-random number sequence covers a relatively short network runtime, which is not long enough to ensure convergence to stationary distributions. After a runtime of 6.56 s, the noise pattern is repeated.
- **Relaying the generated noise to all neurons:** Since there are only 8 LFSRs per HICANN, the noise has to be split and distributed to all network neurons on each HICANN chip. Since there are limited routing lines, a significant portion of network neurons is bound to share noise inputs with other network neurons. This becomes significant for large-scale networks that we have presented in the previous chapter.

All in all, the LFSR noise generators are suited to support a fraction of the network with noise input. The limited length of the state sequence causes temporal correlations, since the noise input pattern is repeated after a relatively short period of network runtime. More importantly, since all sampling neurons require a certain amount of noise input, a fraction of neurons is bound to share these noise sources among each other. This inevitably introduces spatial correlations between these neurons, which is not accounted for in our LIF sampling framework, where we have used independent Poisson noise sources. For such networks, these correlations affect the stochastic dynamics. This problem will be the focal point of this chapter. In particular, we will restrict our analysis to pairwise correlations of neuron pairs (Figure 7.3), since these correlations constitute a substantial part of higher-order correlations in neuron dynamics and can be analyzed using straightforward methods.

The problem of having spatial correlations on a wafer-scale system due to shared noise sources has been worked on for several years, started by Johannes Bill and Mihai A. Petrovici (*Petrovici and Bill*, 2009). The original idea to deal with this problem was to minimize correlated firing of network neurons by minimizing the number of shared inputs between neuron pairs. that share noise input ((*Petrovici and Bill*, 2009)). We will refer to this type of correlations as *shared input correlations* and will introduce approaches in which the networks itself counteract spatial correlations.

7.2 Compensation of correlations with neural networks

A possible solution to deal with spatial correlations on hardware is to implement additional networks whose purpose it is to counteract these correlations (*Jordan et al.*, 2014, 2016).

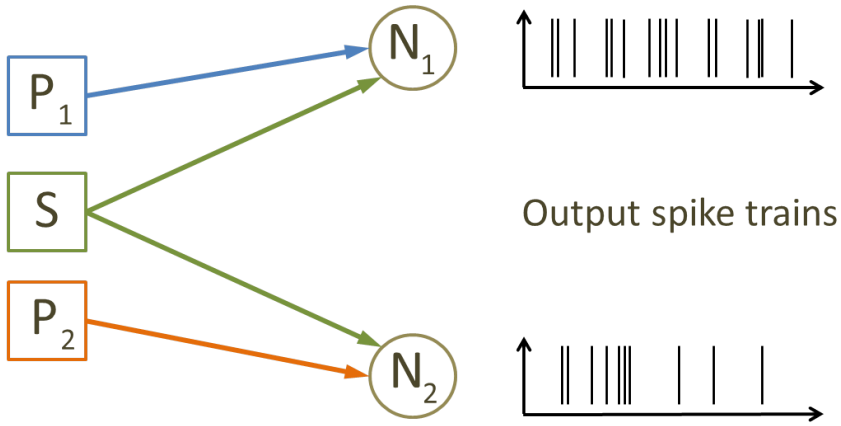

Figure 7.3: A practical problem that is bound to neuromorphic implementations is the bandwidth limitation of external signals. This problem is crucial for stochastic computing networks, as it affects stochastic sources in particular. Commonly, stochastic sources are used to drive network activity in a biological regime (see Section 4.3). The illustration shows two neurons, N_1 and N_2, that are connected to a private source each, P_1 and P_2, and also share one noise source, S. As a result from the shared noise source, their spike output is correlated. Since the spike output encodes our probabilistic states, the correlations impair the network performance.

For large-scale networks on the hardware, such decorrelation networks demand substantial neuron and synapse resources on the wafer. This increases the complexity of mapping the complete network topology on the wafer and thereby also limits the size of possible inference networks. On a more fundamental level, running computational and decorrelating networks in parallel appears to be an arbitrary separation between neural computation and decorrelation. In contrast to this, biological networks can perform decorrelation, as well as computational tasks at the same time (*Gütig et al.*, 2003; *Helias et al.*, 2008). We aim to resolve this discrepancy by embedding a decorrelational function into our stochastic sampling networks.

7.3 Equivalence of weights and noise

In principle, not only hardware substrates, but also biological systems show bandwidth and resource constraints, since the number of synapses and neurons is limited in the neural substrate. One of the main questions is whether the measured correlation traces in biological networks are mere artefacts resulting from the consequences of shared inputs or whether shared inputs could be used as means of computation in neural circuitry. The relationship between correlation traces and encoding of information is a major topic in computational neuroscience (*Moreno-Bote et al.*, 2014; *Rosenbaum et al.*, 2014; *Tchumatchenko et al.*, 2011). Our goal is to build networks that use shared noise to encode information by sampling from Boltzmann probability distributions, thereby impementing shared noise as a computational component of stochastic networks. To characterize correlations in spiking neural networks, we can make a distinction between two key causes for correlations in firing activity.

- **Shared feedforward input:** Neurons in a layered network receive stochastic input from presynaptic sources (Figure 7.4A). Neurons that receive input from the same source (marked in color), also exhibit correlated output. The correlations resulting from these shared inputs are propagated to the next layer. Note that there are no lateral (intralayer) connections to propagate these correlations within the layer.
- **Lateral synaptic connections:** Each neuron receives a private source (Figure 7.4B) without shared inputs. The correlations in the neuron output are caused by intralayer synaptic connections.

We can interpret the relayed shared noise as shared background input, which is propagated through Boltzmann connections to other neurons in the BM. We aim to modify the network connectivity to compensate the feedforward correlations by introducing additional lateral synaptic connections. These additional synaptic connections will be implemented to counteract the correlations originating from shared noise.

We will describe spatial correlations in the context of the sampling framework that was introduced in Chapter 5. Again, the firing activity of each neuron can be described

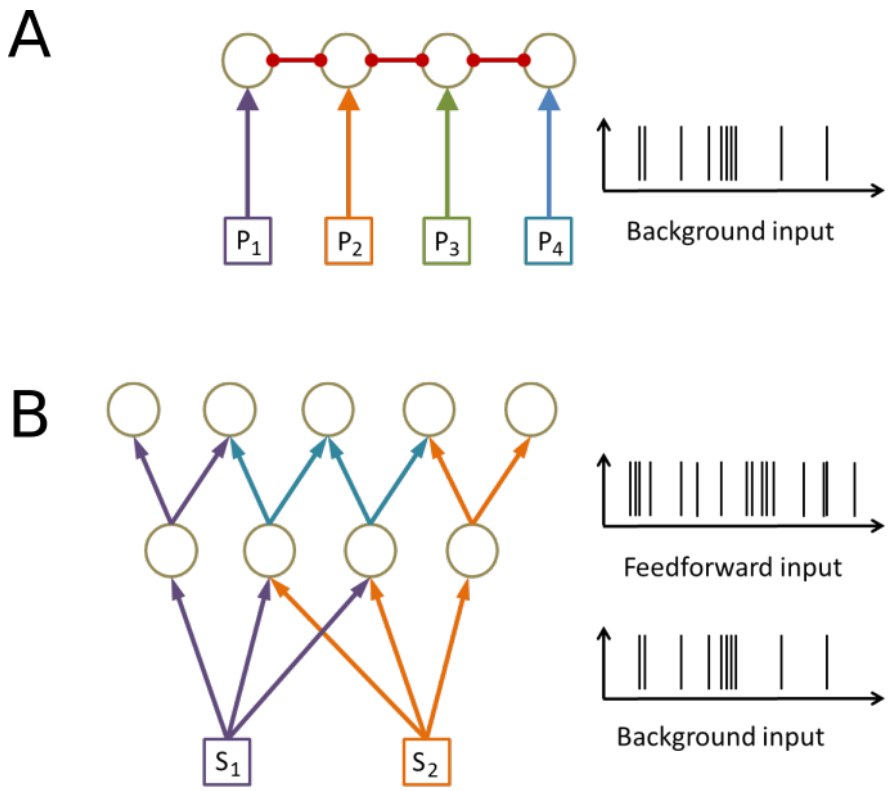

Figure 7.4: The figures show two possible causes of correlations in a network. **(A)** Each neuron receives independent input and is laterally connected to other neurons in the network. The interneuron connections induce correlated firing patterns. **(B)** In a feedforward architecture, the neurons in the layer are not connected via synapses, but receive shared input from the lower layer, which propagates the signal, causing output correlations in the upper layer.

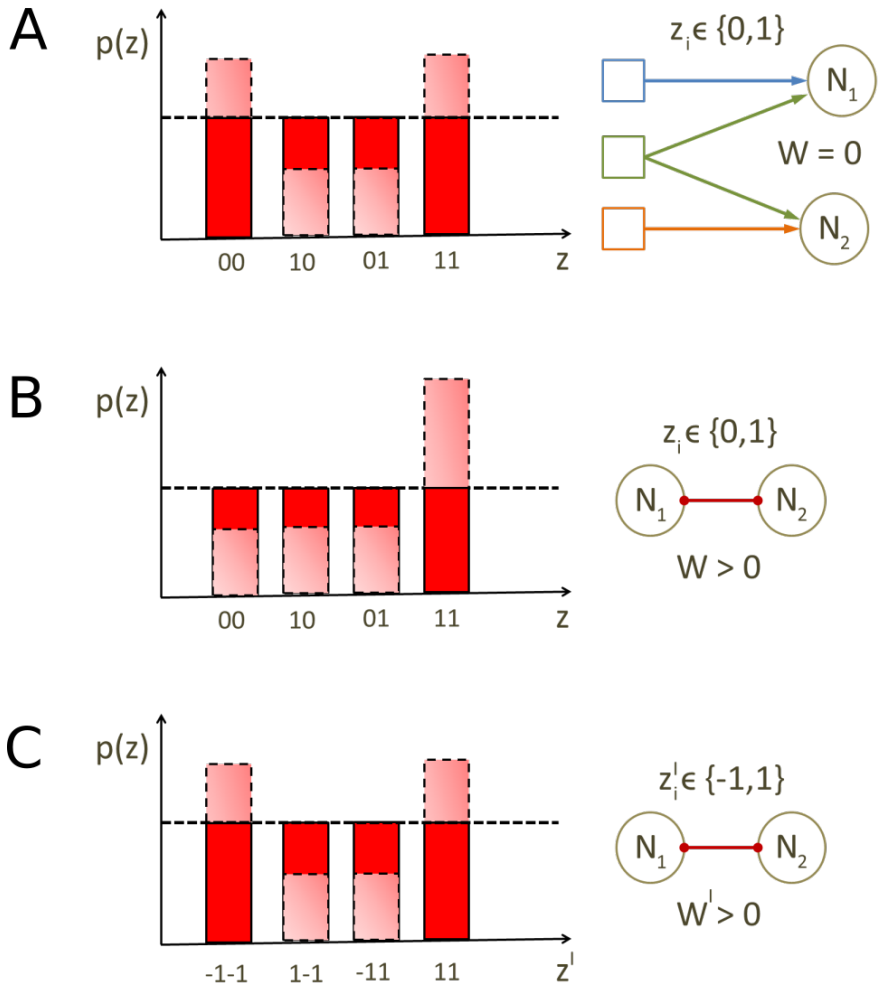

Figure 7.5: The effect of shared noise and synaptic interaction on pairwise joint probabilities. **(A)** Given a two-neuron BM encoding a uniform Boltzmann distribution, we see the effect of positive spike train correlations in shared noise. The probabilities of the (00) and (11) states increase by the same quantity, since neurons are correlated during active and inactive states by the same amount. **(B)** An increase in excitatory weight increases correlations only for the (11) state. The neurons are correlated only during synaptic transmission, which happens only during the $z = 1$ state. **(C)** The same probability distribution can be modeled using states $z^{\mathrm{I}} \in \{-1, 1\}$ that are known from the Ising model. In the Ising domain, correlations due to synaptic weights are equivalent to correlations resulting from shared input noise (A). To use this equivalence, the BM needs to be translated into the Ising domain, characterized by states $z^{\mathrm{I}} \in \{-1, 1\}$. This translation is done by rescaling weights and biases, as described in Equations 7.7, 7.8.

as a binary random variable,

$$z_i(t) = \begin{cases} 1, & \text{neuron has spiked during } (t - \tau_{\text{ref}}, t) \\ 0, & \text{else} \quad . \end{cases} \tag{7.1}$$

Since we are interested in pairwise correlations, we look at pairwise joint activity states z_{ij}. The corresponding pairwise joint probabilities, $p(z_i, z_j)$, are affected by shared inputs and synaptic connections and reflect the quantity of correlated pairwise activity of the neuron pair. We can exploit this relationship and introduce additional Boltzmann weights to change these joint probabilities. Since shared noise gives positive correlations, we can connect positively correlated neuron pairs with inhibitory Boltzmann weights to achieve a net pairwise correlation coefficient of zero. In the following we will investigate the applicability of this intuitive approach.

It is important to note that the impact of weights and shared inputs on the probability distribution is different. In Figure 7.5 we can see the effect on the state distribution for both, shared inputs (Figure 7.5A) and (excitatory) synaptic weights (Figure 7.5B). The two correlation patterns cannot be mapped onto each other in a straightforward way. In the case of present shared noise (A), the neuron pair synchronizes its dynamics perpetually, not only during interaction periods. As a result, both probabilities $p(z_i = 1, z_j = 1)$ and $p(z_i = 0, z_j = 0)$ change in the same way. In contrast to this, correlations due to synaptic interaction require both neurons to be active at the same time (i.e., be in the $z = 1$ state), since synapses are active only during $z = 1$ states (B). A strong excitatory weight therefore increases the probability $p(z_i = 1, z_j = 1)$ and decreases all remaining state probabilities. Analogously, a strong inhibitory weight would increase the probabilities $p(z_i = 1, z_j = 0)$ and $p(z_i = 0, z_j = 1)$ by the same amount and decrease remaining ones.

This means that we cannot compensate shared noise by using weights only. We have seen in Figure 7.5 that an increase in shared noise ratio is inherently different from an increase in excitatory interaction. During interaction, an asymmetric relocation of probability mass towards the $p(z_i = 1, z_j = 1)$ probability occurs. In contrast to this, shared noise shifts the probability mass in a symmetric manner. If we want to show how synaptic weights can be applied to compensate for shared noise, we need a modification of our theoretical sampling framework.

7.4 Sampling in the Ising domain

A key point of our theoretical framework is the abstraction of neuron dynamics to binary states. The binary states $\{0, 1\}$ intuitively reflect the (non)refractory neuron state, as "0" defines an inactive (subthreshold) state and "1" defines a neuron during its refractory period. We have already pointed out that in a sampling network governed by these states, the state probability is not shifted in a symmetric manner in case of positive synaptic weights. A network model that fulfills the symmetric relocation of probability mass is the Ising model (*Ising*, 1925). The Ising model was proposed by Ernst Ising in 1925 and was originally used to model magnetic dipoles in a spin system with spin states

$\{-1, 1\}$. Although being very similar to Boltzmann machine networks, the network of spin particles exhibits an important difference. Since the interaction between spin particles occurs during both spin states, -1 and 1, the resulting correlations in the spin states are symmetric in $p_{\text{Ising}}(1, 1)$ and $p_{\text{Ising}}(-1, -1)$. Therefore, we aim to map our system of $z \in \{0, 1\}$ states to a system of $\{-1, 1\}$ states and validate the equivalence of shared input correlations and weight-induced correlations. To perform this mapping, we will first look at the energy landscape of the Ising model, which is described by a Hamiltonian H of the spin system,

$$\boldsymbol{H} = -\sum_{\langle ij \rangle} J_{ij} z_i^{\text{I}} z_j^{\text{I}} - \mu \sum_i h_i z_i^{\text{I}} \quad . \tag{7.2}$$

In a solid state model we consider a set of lattice sites where spin particles are located. In the original Ising model on a two-dimensional grid, the spin interaction occurs only between the adjacent (indicated by $\langle . \rangle$) sites i, j with magnetic interaction $\boldsymbol{J}$. The strength of the external field is modeled by $\boldsymbol{h}$ and the particles on the sites are all identical with spin states denoted as $z^{\text{I}} \in \{-1, 1\}$. The interaction with an external field is modeled by the second term in Equation 7.2, where μ represents the induced magnetic moment of the particles. The probability of finding a particle in a particular spin state is then described by a Boltzmann distribution,

$$p_{\text{Ising}}(\boldsymbol{z}^{\text{I}}) = \frac{e^{-\beta H(\boldsymbol{z}^{\text{I}})}}{Z_\beta} \quad , \tag{7.3}$$

$$Z_\beta = \sum_{\boldsymbol{z}^{\text{I}}} e^{-\beta H(z^{\text{I}})} \quad . \tag{7.4}$$

Here, β is the known inverse temperature and Z_β denotes the partition function, which is the normalization of the probability density. The statistics of the Hamiltonian in Equation 7.2 are governed by a Boltzmann distribution as well. As already noted before, a key difference is that spin states have positive and negative interaction states in $z^{\text{I}} \in \{-1, 1\}$, whereas BM states only map a positive interaction state in $z \in \{0, 1\}$. Consequently, for an interaction $\boldsymbol{J}$, both probabilities $p_{\text{Ising}}(z_i^{\text{I}} = 1, z_j^{\text{I}} = 1)$ and $p_{\text{Ising}}(z_i^{\text{I}} = -1, z_j^{\text{I}} = -1)$ are symmetric.

Since the states of both systems, z and z^{I}, are sampled from Boltzmann distributions, it is possible to find a mapping between the Ising domain and the BM domain (*Baumbach*, 2016). To keep the network notation consistent, we will move away from the notation of physical spin systems and will denote the Ising weights and biases as W^{I} and b^{I}, respectively.

A detailed derivation of the mapping rule can be found in *Baumbach* (2016) and here we will provide only the resulting translation rules for a given Boltzmann machine energy function,

$$E(\boldsymbol{z}) = -\frac{1}{2} \sum_{i,j} W_{ij} z_i z_j + \sum_i b_i z_i \quad , \tag{7.5}$$

and an Ising energy function,

$$H(\boldsymbol{z}^{\mathrm{I}}) = -\frac{1}{2}\sum_{i,j} W_{ij}^{\mathrm{I}} z_i^{\mathrm{I}} z_j^{\mathrm{I}} + \sum_i b_i^{\mathrm{I}} z_i^{\mathrm{I}} \quad . \tag{7.6}$$

We translate Ising parameters $(\boldsymbol{W}^{\mathrm{I}}, \boldsymbol{b}^{\mathrm{I}})$ into Boltzmann parameters $(\boldsymbol{W}, \boldsymbol{b})$ by using

$$\boldsymbol{b} = 2\boldsymbol{b}^{\mathrm{I}} + 2\boldsymbol{W}^{\mathrm{I}} \quad , \tag{7.7}$$

$$\boldsymbol{W} = 4\boldsymbol{W}^{\mathrm{I}} \quad . \tag{7.8}$$

By translating the parameters, we can map the statistics of the Ising system onto our Boltzmann machines and vice versa. The dynamics of LIF-based Boltzmann machines do not change, as it still operates according to the LIF-based differential equations. The refractoriness is still modeled as a $z = 1$ state during the simulation and the results are interpreted according to the BM energy function (Equation 7.5). Therefore, we will define our networks in the Ising domain and translate the parameters to the BM domain according to Equations 7.7, 7.8 to run the simulation. A key point in these translation rules is that the weight $\boldsymbol{W}^{\mathrm{I}}$ is used to convert the bias $\boldsymbol{b}^{\mathrm{I}}$. This means that part of the interaction $\boldsymbol{W}^{\mathrm{I}}$ is stored in a BM bias $\boldsymbol{b}$ in order to compensate for the lack of the "-1"-interaction parts in the Boltzmann domain. In an example calculation we can see the distribution of states for both cases, $z^{\mathrm{I}} \in \{-1, 1\}$ (Ising domain) and $z \in \{0, 1\}$ (Boltzmann domain) for a simple network of 2 neurons:

$$\begin{aligned} b_1 = b_2 &= 0 \\ W_{12} &= 2 \\ b_1^{\mathrm{I}} = b_2^{\mathrm{I}} &= 0 \\ W_{12}^{\mathrm{I}} &= 2 \end{aligned} \tag{7.9}$$

From these parameters we obtain the resulting probabilities,

$$p(z_1 = 0, z_2 = 0) = \frac{e^{(0+0+0)}}{Z^{BM}} \quad , \tag{7.10}$$

$$p(z_1 = 1, z_2 = 0) = p(z_1 = 1, z_2 = 0) = \frac{e^{(0+0+0)}}{Z^{BM}} \quad , \tag{7.11}$$

$$p(z_1 = 1, z_2 = 1) = \frac{e^{W}}{Z^{BM}} \quad , \tag{7.12}$$

$$Z^{BM} = 3 \cdot e^{0} + e^{W} \quad . \tag{7.13}$$

We see that $p(z_1 = 1, z_2 = 1)$ is different from the remaining three state probabilities. On the other hand, in the Ising domain,

$$p_{\text{Ising}}(z_1^{\text{I}} = -1, z_2^{\text{I}} = -1) = \frac{e^J}{Z^I} \quad , \tag{7.14}$$

$$p_{\text{Ising}}(z_1^{\text{I}} = 1, z_2^{\text{I}} = -1) = p_{\text{Ising}}(z_1^{\text{I}} = -1, z_2^{\text{I}} = 1) = \frac{e^{-J}}{Z^I} \quad , \tag{7.15}$$

$$p_{\text{Ising}}(z_1^{\text{I}} = 1, z_2^{\text{I}} = 1) = \frac{e^J}{Z^I} \quad , \tag{7.16}$$

$$Z^I = 3 \cdot e^0 + e^J \quad , \tag{7.17}$$

the probabilities $p_{\text{Ising}}(z_1^{\text{I}} = -1, z_2^{\text{I}} = -1)$ and $p_{\text{Ising}}(z_1^{\text{I}} = 1, z_2^{\text{I}} = 1)$ are identical. This simple example shows that the probability mass is shifted symmetrically, as can be seen in Figure 7.5. We can now use these translation rules defined in Equations 7.7, 7.8 to validate our proposed compensation method by introducing additional inhibitory synaptic interaction $\boldsymbol{W}^{\text{I}}$ to counteract correlation.

7.5 Compensation of shared noise correlations

We will first restrict ourselves to two-neuron LIF networks, where we characterize the shared input correlations of two cases, namely synaptic connections and shared Poisson input. We use the default parameters, as stated in Appendix A.3. We will define our networks in the Ising domain with parameters ($\boldsymbol{W}^{\text{I}}$, $\boldsymbol{b}^{\text{I}}$) but translate them into the BM domain to run the LIF simulations. After the simulations we will compute the pairwise correlations, using the well-known measure *Pearson product-moment correlation coefficient* (CC),

$$\rho_{XY} = \frac{\text{Cov}(X, Y)}{\sigma_X \sigma_Y} \tag{7.18}$$

$$= \frac{\text{E}[(X - \mu_X)(Y - \mu_Y)]}{\sigma_X \sigma_Y} \quad . \tag{7.19}$$

Here, $\text{Cov}(X, Y)$ is the covariance of random variables X and Y. The population average and standard deviation is denoted as E[.] and σ, respectively.

The random variables X and Y are interpreted in terms of Boltzmann neuron states z. We look at joint activity states z_{ij} that are governed by pairwise joint probabilities $p(z_i, z_j)$ with binary states $(z_i, z_j) \in \{0, 1\}^2$.

We can rewrite Equation 7.19, using the known relationship,

$$\text{E}[(X - \mu_X)(Y - \mu_Y)] = \text{E}[X, Y] - \text{E}[X]\text{E}[Y] \quad ,$$

and z_i, z_j as random variables of neurons i and j,

$$\rho_{XY} = \frac{\mathrm{E}[z_i, z_j] - \mathrm{E}[z_i]\mathrm{E}[z_j]}{\sigma_i \sigma_j} \tag{7.20}$$

$$= \frac{\sum_{z_i, z_j \in \{0,1\}} p(z_i, z_j) z_i z_j - \sum_{z_i \in \{0,1\}} p(z_i) \sum_{z_j \in \{0,1\}} p(z_j)}{\sqrt{(\mathrm{E}[z_i^2] - \mathrm{E}^2[z_i])(\mathrm{E}[z_j^2] - \mathrm{E}^2[z_j])}} \tag{7.21}$$

$$= \frac{\sum_{z_i, z_j \in \{0,1\}} p(z_i, z_j) z_i z_j - \sum_{z_i \in \{0,1\}} p(z_i) \sum_{z_j \in \{0,1\}} p(z_j)}{\sqrt{\prod_{i,j} \left[\sum_{z_n \in \{0,1\}} p(z_n) z_n^2 - \left(\sum_{z_n \in \{0,1\}} p(z_n) z_n \right)^2 \right]}} \quad . \tag{7.22}$$

Here we can simplify above expression by eliminating all terms with $z_n = 0$. This results in

$$\rho_{ij} = \frac{p(z_i = 1, z_j = 1) - p(z_i = 1) \cdot p(z_j = 1)}{\sqrt{p(z_i = 1) - p(z_i = 1)^2}} \quad . \tag{7.23}$$

The above measure will be used to quantify pairwise correlations of neuron states. Note that this procedure is equivalent to evaluating spike train correlations where the spike convolution kernel is a rectangle with length τ_{ref}.

7.5.1 Impact of weights on shared input correlations

We apply ρ_{ij} to evaluate the correlation impact of shared input s and compare it to the correlations caused by synaptic weights of an LIF neuron pair, as described in Figure 7.5. To compare the functional form of ρ_{ij} of both cases, we sweep over excitatory synaptic weights in the range $W^{\mathrm{I}} \in [0, 1]$ and the ratio of shared to private Poisson inputs, $s \in [0, 1]$ for a single neuron pair. The ratio s determines the percentage of shared input, given a total input rate. In the sweeps, the total rate that an LIF neuron receives, is $\nu_{\mathrm{exc}} = \nu_{\mathrm{inh}} = 5000\,\mathrm{Hz}$. Then, the Poisson rate that is shared with the other neuron is set as $\nu_s = s \cdot \nu_{\mathrm{exc}} = s \cdot \nu_{\mathrm{inh}}$. The independent, private part of the Poisson input rate is then $\nu_p = (1 - s) \cdot \nu_{\mathrm{exc}} = (1 - s) \cdot \nu_{\mathrm{inh}}$. The rest of the parameters is set to standard values, as described in Appendix A.3. We define our networks in the Ising domain where we set $\boldsymbol{b}^{\mathrm{I}} = \mathbf{0}$. Note that the Boltzmann bias $\boldsymbol{b}$ is still nonzero, as given in translation rules in Equations 7.7, 7.8. Results of the sweep can be seen in Figure 7.6. The fitted curves show a qualitatively different shape, since the underlying causes for correlations are different. It is important to note that fully correlated dynamics can be achieved using shared noise, $s = 1$, resulting in a maximum correlation coefficient, $\rho_{ij}^{s} = 1$. On the other hand, applying excitatory synaptic interaction, the correlation coefficient converges to its maximum, $\rho_{ij}^{W^{\mathrm{I}}} \longrightarrow 1$, for increasing weights, $W^{\mathrm{I}} \longrightarrow \infty$. This means that perfectly correlated states cannot be achieved using synaptic weights in the sampling framework. This is also reflected in Equation 7.17.

Still, in principle, it is possible to compensate for shared inputs that induce correlation coefficients $0 < \rho_{ij}^{s} < 1$ by using inhibitory weights $W_{ij}^{\mathrm{I}} < 0$, resulting in

$$\rho_{ij}^{s} + \rho_{ij}^{W^{\mathrm{I}}} = 0 \quad . \tag{7.24}$$

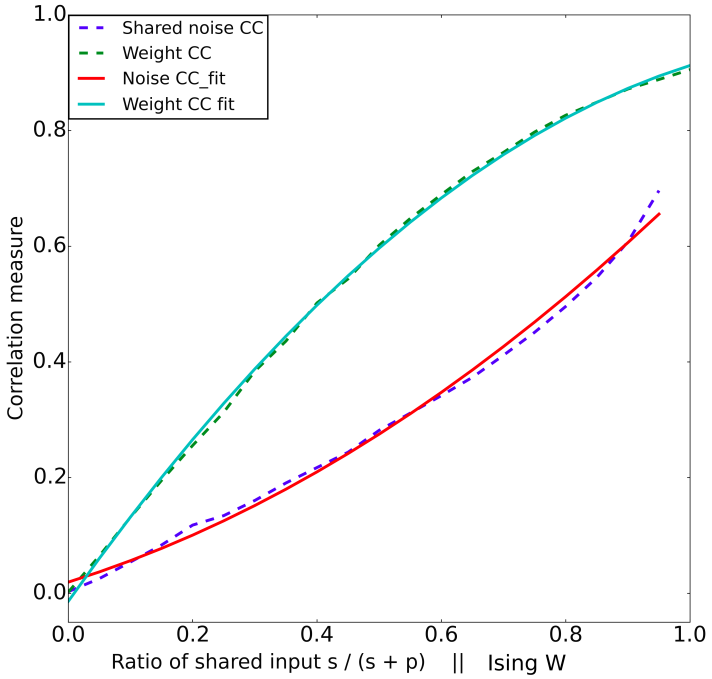

Figure 7.6: The pairwise correlation coefficient was measured for a two-neuron network with synaptic connections $\boldsymbol{W}^{\mathrm{I}} \in [0, 1]$ (dashed green) and the ratio of shared input noise in the range $s \in [0, 0.95]$ (dashed blue). Note that the curves have different functional shapes, but it is still possible to map a correlation coefficient value from one curve to the other.

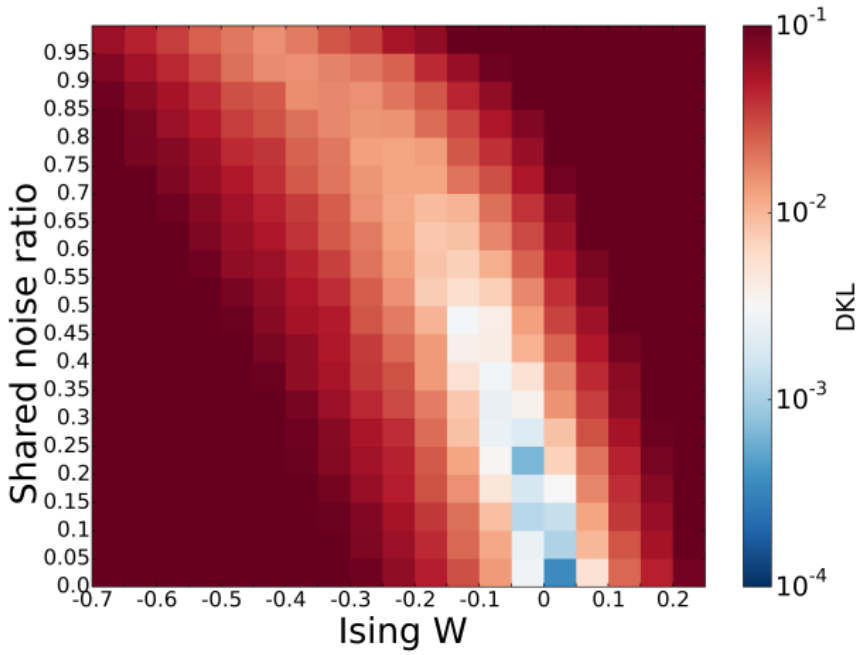

Figure 7.7: The two-neuron network samples from a uniform distribution, i.e., the target BM has zero biases and weights. We introduce shared input correlations by increasing s on the y-axis, inducing positive correlations. This leads to a decrease in sampling quality, which is measured by the DKL and encoded in color in the heatmap. We sweep over inhibitory connection weights ${\boldsymbol{W}^{\mathrm{I}}}_c$ on the x-axis to find a compensation weight for the shared noise correlation. Uncompensated shared input correlations impair the sampling quality, as indicated by red DKL data points. If correlations are compensated, the quality improves (blue). Hence, we see a continuous valley of low DKL values, starting at $(\boldsymbol{W}^{\mathrm{I}} = 0, s = 0)$. This valley marks the parameter pairs $(\boldsymbol{W}^{\mathrm{I}}, s)$ at which the correlation coefficients are cancelled out, resulting in good sampling. Strikingly, we see that this valley of low DKL values becomes less pronounced for larger compensating weights. The reason is that for a high shared input ratio s, large compensating weights are necessary. In these cases, systematic deviations occur, as explained in Section 5.3. The DKL is averaged over ten simulation runs with a runtime of $T_{\mathrm{sim}} = 10^5$ ms each.

To validate this, we set up a two-neuron network that is set to sample from a uniform distribution, where $p_{\text{target}}(z_i^{\text{I}}, z_j^{\text{I}}) = \frac{1}{2^2} \forall z_i^{\text{I}}, z_j^{\text{I}}$. We have chosen this setup because this distribution is encoded without connections and biases, therefore, there are no pairwise correlations which would complicate the analysis. As soon as shared noise is implemented, the LIF-sampled distribution $p_{\text{LIF}}(\boldsymbol{z}^{\text{I}})$ is significantly impaired. As a result, the Kullback-Leibler divergence $\text{D}_{\text{KL}}\left(p_{\text{LIF}} \,||\, p_{\text{target}}\right)$ increases. In Figure 7.7, we implement shared noise ratios $s \in [0, 1]$ and connect the neuron pair with weights $W^{\text{I}} \in [-0.7, 0]$ to find the inhibitory weights that compensate for these ratios s. The results in Figure 7.7 show a valley of low DKL values. This valley of low DKL values indicates a good network performance and suggests that the harming spatial correlations have been cancelled out by inhibitory weights, according to Equation 7.24. Note that the shape of the DKL valley is very similar to the correlation curve $\rho_{ij}^{W^{\text{I}}}$ shown in Figure 7.6, where positive correlations were induced by excitatory connections.

One concerning observation in Figure 7.7 is the substantial flattening of the valley for larger Ising weights. This indicates a decreasing impact of synaptic weights and can be explained by systematic deviations due to large weights, as discussed in Section 5.3. We still can conclude that compensation with small weights is possible to a certain degree.

7.5.2 Connectivity as shared noise

In our original goal we aimed to compensate positive shared noise correlations, $\rho_{ij}^{s} > 0$, by introducing inhibitory connections. We have seen in the previous section that compensation of correlations decreases in quality as compensating weights become larger. In practice we can compensate small shared input correlations with synaptic connections. It is important that the problem resulting from large weights is not related to the compensation of correlation, but a consequence of the LIF sampling framework. We have already investigated and benchmarked this effect in Section 5.3. This means that mapping high pairwise correlations by using synaptic weights leads to systematic errors. Since we are able to model strong correlations by using a high shared input ratio s, it should also be possible to use shared noise instead of weights to model strong correlations. This would lift the necessity to introduce large weights into networks and improve sampling quality, while keeping the same pairwise correlation pattern. We invert direction of the compensation from $s \longrightarrow \boldsymbol{W}^{\text{I}}$ to $\boldsymbol{W}^{\text{I}} \longrightarrow s$ and model the impact of large weights via shared inputs. If both parameters impact the network in the same way, i.e., induce the same correlation coefficient, then connectivity patterns could be substituted by shared input patterns allowing the network to sample well from the target probability distribution without synaptic weights. This could be particularly useful for a hardware network setup where strong pairwise correlations are necessary, but constrained by upper bounds of synapse strengths (*Schmidt*, 2014).

We want to investigate this case by sampling with two LIF neurons from a probability distribution which is not uniform, but defined by $(\boldsymbol{W}^{\text{I}} > 0, \boldsymbol{b}^{\text{I}} = \boldsymbol{0})$. The bias $\boldsymbol{b}^{\text{I}}$ will be zero to simplify analysis. The weight $\boldsymbol{W}^{\text{I}}$ is randomly drawn from a beta distribution, as described before. Although these parameters define the target distribution, the neuron pair will not be connected by $\boldsymbol{W}^{\text{I}}$, but only share noise sources with ratio s. The DKL

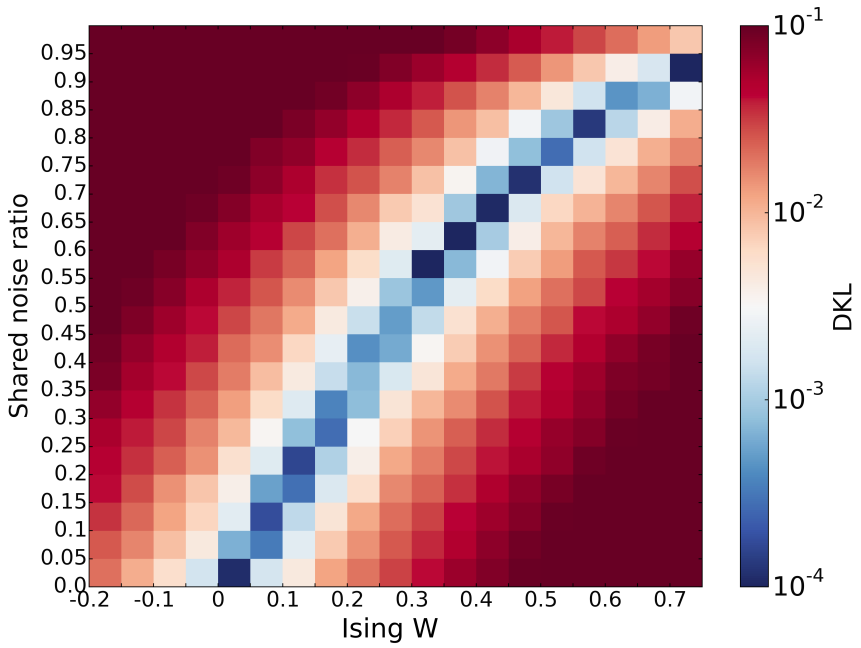

Figure 7.8: The sampling performance of a two-neuron network without connections. The network is set up to sample from a target Boltzmann distribution defined by $(\boldsymbol{W}^{\mathrm{I}} > 0, \boldsymbol{b}^{\mathrm{I}} = \mathbf{0})$. We sweep over these theoretical weights $\boldsymbol{W}^{\mathrm{I}}$ on the x-axis. The neuron pair shares Poisson noise with a ratio s, mapping the pairwise correlations. The quality of the resulting sampling is measured by the DKL, which is shown in color in the heatmap. We find good sampling quality (blue) in regions where the shared noise parameter s is set to match the pairwise correlation structure defined by the target Boltzmann parameters. This region can be identified as a valley of low DKL, starting at $(\boldsymbol{W}^{\mathrm{I}} = \mathbf{0}, s = 0)$. The quality of sampling is preserved in this region because s implements precisely the pairwise correlations that correspond to the weights, $\boldsymbol{W}^{\mathrm{I}}$, as predefined in theory. Note that, in contrast to shared noise compensation shown in Figure 7.7, we do not implement any synaptic connections. Therefore, the sampling performance remains good even for large s, since we avoid systematic weight-induced errors. The DKL has been averaged over ten simulation runs with a runtime of $T_{\mathrm{sim}} = 10^5$ ms each.

measures the distance between both distributions, the targeted one, p^{W^I}, and the one sampled by the neuron pair, p^s. The DKL will indicate whether networks operating on shared noise can carry out the same computations as synaptic weights. In the simulations we will use the standard LIF sampling parameters (Appendix A.3) and sweep over the target weights $W^\mathrm{I} \in [0, 0.75]$ while using shared noise ratios of $s \in [0, 1]$. Note that weights W^I are positive because they are equivalent to a positive correlation coefficient resulting from shared input ratio s. Negative shared input correlations can be implemented by inverting the synapse types where the shared noise is injected. In such a case the one neuron would receive the shared spike trains through excitatory synapses, the other neuron would receive them through inhibitory synapses. However, we will limit our study case to positive $\boldsymbol{W}^\mathrm{I}$ only, since both cases are analogous.

The results of the two-neuron sweeps are shown in Figure 7.8. Here we see a valley of low DKLs for pairs of excitatory target weights $\boldsymbol{W}^\mathrm{I}$ and shared noise ratios s. This means that the network performance remains high for a subset of $(\boldsymbol{W}^\mathrm{I}, s)$-pairs, since the distance between p^{W^I} and p^s is small. The valley of low DKL values does not flatten, since we use shared input to model pairwise correlations instead of synaptic weights. Hence there are no systematic errors resulting from the weight translation described in Section 5.4. Excluding the effects of large weights on the sampling dynamics, the sampling performance remains the same if we interchange lateral weights and shared noise, as indicated by the DKL. Since the DKL is our measure of the functional properties of sampling networks, we can conclude that the network functionality can be maintained regardless of the origin of network correlations. This, in turn, suggests that we can characterize the network parameter pair $(\boldsymbol{W}^\mathrm{I}, s)$ as a single free parameter, namely the correlation coefficient $\rho_{W^\mathrm{I},s}$. This parameter fully defines the pairwise correlation structure.

7.5.3 Equivalence of interaction weights and shared noise

Assuming that we only need $\rho_{W^\mathrm{I},s}$ to describe the pairwise correlation topology, we can configure networks to arbitrary noise-weight configurations $(\boldsymbol{W}^\mathrm{I}, s)$. This yields the same correlation structure and sampling performance if the implemented weights $\boldsymbol{W}^\mathrm{I}$ are not too strong. We have already demonstrated that it is possible to implement a purely weight-driven network with an additional weight W_c^I that incorporates the computational part of the shared noise ratio s. This means that a network with $(\boldsymbol{W}^\mathrm{I} + \boldsymbol{W}^\mathrm{I}{}_c, 0)$ would achieve the same network functionality as $(\boldsymbol{W}^\mathrm{I}, s)$, if $\rho_{\boldsymbol{W}^\mathrm{I},s} = \rho_{\boldsymbol{W}^\mathrm{I}+\boldsymbol{W}^\mathrm{I}{}_c,0}$. In the same way, it is also possible to achieve the same functionality by using only additional shared noise s_c, resulting in a $(\boldsymbol{0}, s + s_c)$ configuration with

$$\rho_{\boldsymbol{W}^\mathrm{I},s} = \rho_{\boldsymbol{W}^\mathrm{I}+\boldsymbol{W}^\mathrm{I}{}_c,0} = \rho_{\boldsymbol{0},s+s_c} \quad . \tag{7.25}$$

Since the correlation coefficient $\rho_{\boldsymbol{W}^\mathrm{I},s}$ cannot be computed analytically, it is necessary to sweep over weights and shared ratios to find compensation parameters for a specific correlation setup. Such a measurement over $\rho_{\boldsymbol{W}^\mathrm{I},s}$ of value pairs in $W^\mathrm{I} \in [-1, 1]$ and $s \in [0, 1]$ is shown in Figure 7.9. For a given set of LIF parameters, the correlation coefficients on the hyperplane in Figure 7.9 define any correlation pattern of a neuron

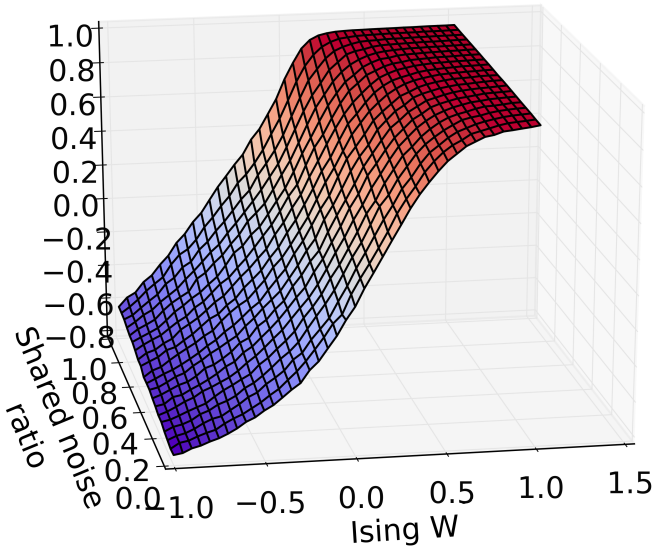

Figure 7.9: The hyperplane in the three-dimensional space shows the correlation coefficient, $\rho_{\boldsymbol{W}^{\mathrm{I}},s}$, for a neuron pair for varied shared noise ratios (y-axis) and synaptic weights $\boldsymbol{W}^{\mathrm{I}}$ (x-axis). The color, as well as the z-axis value denote the value of the correlation coefficient $\rho_{\boldsymbol{W}^{\mathrm{I}},s}$. Blue color marks negative, red color marks positive correlation coefficients. The correlation coefficient hyperplane could be used to map a desired correlation pattern, defined by $\rho_{\boldsymbol{W}^{\mathrm{I}},s}$ for a variety of weights and shared noise ratios. For networks on neuromorphic hardware, it could be possible to combine shared noise and weight configurations to adapt to hardware-imposed parameter constraints. Each data point is a result of ten averaged simulation runs with a runtime of $T_{\mathrm{sim}} = 10^5$ ms each.

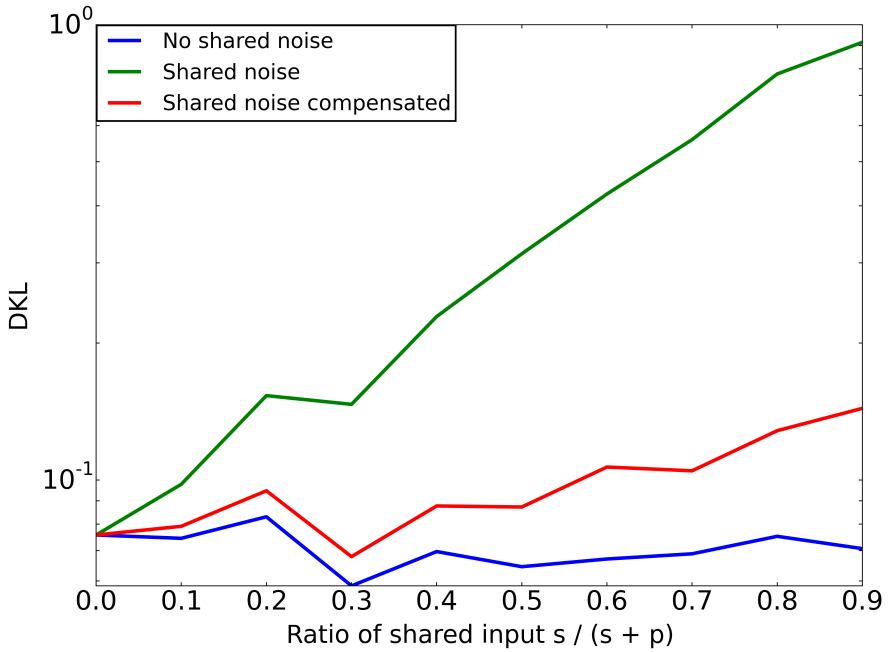

Figure 7.10: A ten-neuron network is set to sample from a uniform distribution with ($\boldsymbol{W}^{\mathrm{I}} = \mathbf{0}, \boldsymbol{b}^{\mathrm{I}} = \mathbf{0}$). Each neuron shares noise with only one other neuron. We increase the shared noise input (x-axis) and at the same time introduce an additional negative weight matrix W_c^{I} to decorrelate the network. The red curve shows the DKL of the network in comparison to a benchmark with independent noise (blue) and the same network without decorrelation of the noise sources (green). Although the decorrelation does not achieve the results of the benchmark, there is a strong improvement compared to the network without decorrelating weights. The simulations were run for $T_{\mathrm{sim}} = 10^5$ ms.

pair, regardless whether the correlations result from weights or shared noise. We see that, in principle, shared noise and synaptic weights can be interchanged to yield an identical correlation coefficient. The network components ($\boldsymbol{W}^{\mathrm{I}}, s$) can be freely modified, defining the same quality of network computation as long as the resulting correlation coefficient $\rho_{\boldsymbol{W}^{\mathrm{I}},s}$ is identical. This could benefit hardware implementations of sampling networks, as it provides a method to obtain a desired pairwise correlation topology, respecting hardware-imposed constraints on shared inputs and synaptic weights.

7.5.4 Compensation of shared noise in networks

We can apply the results that are illustrated on the hyperplane in Figure 7.9 to extend our compensation of shared noise to bigger networks. We will again compensate undesired shared noise correlations for pairs of neurons in a ten-neuron sampling network.

The parameters for the target distribution are again drawn from a beta distribution, as described in A.4. We implement shared input noise with $s \in [0, 1]$ and counteract the resulting increase of correlations by setting additional compensation weights W_c^{I}. Ideally, this would result in the elimination of correlations, $(\boldsymbol{W}^{\mathrm{I}}, s) \longrightarrow (\boldsymbol{W}^{\mathrm{I}} + \boldsymbol{W}^{\mathrm{I}}{}_c, 0)$. Since we have measured the correlation coefficient $\rho_{\boldsymbol{W}^{\mathrm{I}},s}$ for a range of weight-noise configurations $(\boldsymbol{W}^{\mathrm{I}}, s)$, we can read out the necessary compensation weights W_c^{I} to cancel out the shared noise correlations.

In Figure 7.10 we see the result of the simulations, where we have compared the DKL values for three cases. In a case without shared input, the DKL remains flat (blue), as performance stays constant. For an increase of the shared noise ratio (x-axis), the performance deteriorates if the network correlations are not compensated for (green). By adding compensating synaptic weights to the network (red), $(\boldsymbol{W}^{\mathrm{I}}, s) \longrightarrow (\boldsymbol{W}^{\mathrm{I}} + \boldsymbol{W}^{\mathrm{I}}{}_c, 0)$, the DKL decreases significantly. But for higher shared noise ratios ($s > 0.4$), the DKL increases noticeably. Although it is possible to compensate for low shared noise ratios ($s \leq 0.3$), higher ratios require higher inhibitory synaptic weights W_c^{I}. In such cases systematic errors arise again, as we have already observed in Figure 7.8. The increase of the total presynaptic weight input per neuron due to additional compensation weights, $\boldsymbol{W}^{\mathrm{I}}{}_c$, leads to higher systematic errors. This impact has also been illustrated in Section 5.3 (e.g., Figure 5.5). Even if the single compensating weights are small, they sum up in large networks where many pairwise correlations need to be compensated for. For instance, in the classification networks in Section 6.2, the connectivity pattern includes up to 784x1200 connections (Figure 6.4). Realistically, it requires a complicated connectivity scheme to embed compensational weights or shared noise for each neuron to achieve a good network performance.

Overall, cases in which connectivity and shared noise ratios are small can be successfully decorrelated using compensational weights without substantial loss of performance. For larger networks, we will use the obtained results to develop a more practical method to deal with shared noise correlations.

7.5.5 Training correlations

In Section 6.2 we have used a simple training method to find Boltzmann parameters that map the MNIST data set on a Boltzmann distribution. We will use a training approach to adapt the network to shared noise correlations such that it can sample from target Boltzmann distributions. In this method, the knowledge of the desired correlation coefficent $\rho_{\boldsymbol{W}^{\mathrm{I}},s}$ is not necessary, since the parameters Boltzmann parameters $(\boldsymbol{W}^{\mathrm{I}}, b^{\mathrm{I}})$ are updated according to a predefined learning rule. In the training rule, the target pairwise correlations are defined by the pairwise joint probabilities of the neurons. We can implement this information into the already known contrastive divergence training algorithm (Section 6.1.1). In the update rules we will still use the Ising parameters $(\boldsymbol{W}^{\mathrm{I}}, \boldsymbol{b}^{\mathrm{I}})$. This does not change the shape of the CD equations,

$$\Delta W^{\mathrm{I}}_{ij} = \eta \left[\langle z_i z_j \rangle_{\mathrm{data}} - \langle z_i z_j \rangle_{\mathrm{model}} \right] \quad , \tag{7.26}$$

$$\Delta b^{\mathrm{I}}_{j} = \eta \left[\langle z_i \rangle_{\mathrm{data}} - \langle z_i \rangle_{\mathrm{model}} \right] \quad . \tag{7.27}$$

Note that the terms $\langle z_i z_j \rangle$ are probabilities of correlated pairwise firing, $p(z_i = 1, z_j = 1)$, so we can rewrite both equations according to

$$\Delta W^{\mathrm{I}}_{ij} = \eta \left[p(z_i = 1, z_j = 1)_{\mathrm{data}} - p(z_i = 1, z_j = 1)_{\mathrm{model}} \right] \quad , \tag{7.28}$$

$$\Delta b^{\mathrm{I}}_{j} = \eta \left[p(z_i)_{\mathrm{data}} - p(z_i)_{\mathrm{model}} \right] \quad . \tag{7.29}$$

This means that the updates adjust the network firing probabilities ($p(z_i = 1, z_j = 1)_{\mathrm{model}}$, $p(z_i)_{\mathrm{model}}$) to the desired firing probabilities ($p(z_i = 1, z_j = 1)_{\mathrm{data}}$, $p(z_i)_{\mathrm{data}}$). The correlation structure of each neuron pair in the network is reflected in the pairwise joint probabilities. Therefore, the training is conducted to adjust the parameters $(W^{\mathrm{I}}, b^{\mathrm{I}})$ to sample from a target distribution $p_{\mathrm{target}}(\boldsymbol{z}^{\mathrm{I}})$.

We will validate the training method (Equations 7.28,7.28) by training Boltzmann machines consisting of ten neurons to sample from a uniform Boltzmann distribution. We will set a ratio of pairwise shared Poisson noise input, $s = 0.3$. The LIF parameters are set according the standard parameters, as defined in Appendix A.3.

In our simulation we will train the LIF network directly, in contrast to the CD training scheme in Section 6.1.1, where we trained the ANM and mapped the parameters to LIF-based Boltzmann machines. In our LIF training scheme, we will perform simulation runs of duration $T_{\mathrm{sim}} = 5000\,\mathrm{ms}$ and calculate $p(z_i, z_j)$ after each run. Then we apply update rules 7.28, 7.29 to modify the parameters. Each of these simulation runs is then considered a training step. The learning rate $\eta(t_{\mathrm{train}})$ is set as a function of training steps t_{train},

$$\eta(t_{\mathrm{train}}) = \frac{0.05}{1 + \cdot t_{\mathrm{train}}} \quad . \tag{7.30}$$

The parameters for $\eta(t_{\mathrm{train}})$ in the equation above have been chosen after several parameter sweeps and are robust to variations of up to $\approx 10\%$. The decline of the learning rate $\eta(t_{\mathrm{train}})$ asserts that the updates cause small changes in the energy landscape do not overshoot the parameters that represent the desired probability distribution. Since we use only a small BM, we can use the DKL as a function of t_{train} to assess the sampling performance of the trained network, as shown in Figure 7.11. The benchmark DKL line (green) shows the performance of a network without shared noise, initialized with the target weights. The trained network in the presence of shared noise has a decreasing DKL (red) with increasing training time (x-axis). Most importantly, it also falls below the benchmark measurement. This means that the trained network, despite shared noise,

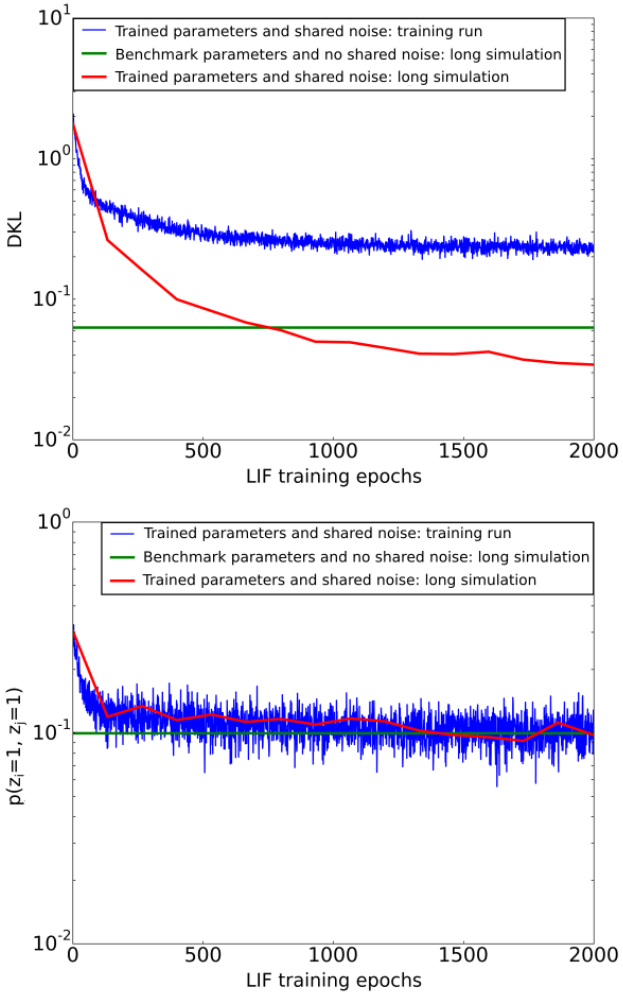

Figure 7.11: **(Top)** Sampling performance of a ten-neuron Boltzmann machine with pairwise shared inputs ($s = 0.3$) and synaptic weights drawn from a beta distribution (Appendix A.4). The green curve indicates the benchmark with independent inputs and the red curve shows the DKL of a CD-trained network with shared noise. The blue curve shows the sampling performance during the training iterations, which are set to $T = 5000\,\text{ms}$. We see that the performance of the trained network (red) even exceeds the benchmark that is simulated with independent inputs. This means that the training does not only adapt the network to shared input correlations, but also modifies the parameters to mitigate the systematic errors resulting from parameter translation from the BM domain to the LIF domain. **(Bottom)** The pairwise firing rate $p(z_i^{\text{I}}, z_j^{\text{I}})$ is trained to yield the target probability of $\frac{1}{2^{10}}$. The sampling runtime of the trained network (red) in each data point was $T_{\text{sim}} = 5 \cdot 10^4\,\text{ms}$.

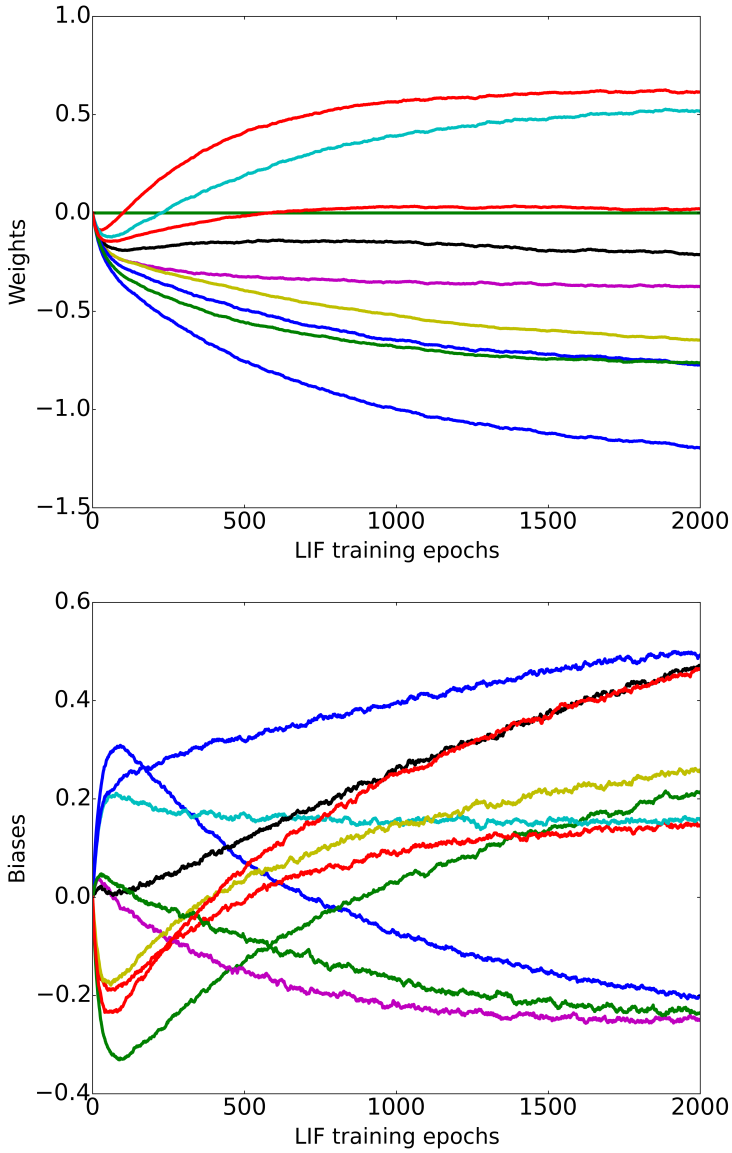

Figure 7.12: Both plots show parameters of the ten-neuron Boltzmann machine from Figure 7.11, trained to sample from a randomly drawn probability distribution in presence of a shared input ratio of $s = 0.3$. Each training iteration lasts 5000 ms. The correlated firing probabilities are averaged over the runtime after each training iteration. **(Top)** The weights $\boldsymbol{W}^{\mathrm{I}}$ are trained to adjust the network to shared background input. **(Bottom)** The biases are additional free parameters and are also trained with the same training algorithm. The preliminary results are promising, but the convergence of both, weights and biases, can be improved by using more elaborate training algorithms.

performs better than an LIF network initialized with the target Boltzmann parameters without shared noise.

A significant shared noise ratio of $s = 0.3$ requires a certain amplitude of compensating Boltzmann weights for each of the ten neurons. Using a simple training algorithm like CD, the network successfully compensates for shared input correlations by training weights and biases, overcoming the inherent weight-related systematic errors of the LIF sampling framework. The trained BM weights and biases can be seen in Figure 7.12. In this preliminary result we can conclude that it is possible to adapt the network to the underlying correlation pattern. At this point we conclude the preliminary studies in this section and will use the results to address the shared noise correlation problem for larger networks. The success of the preliminary training results show potential to run networks which have even higher shared noise ratios than what we have seen here. In fact, we will show that it is possible to construct LIF-based Boltzmann machines which perform sampling without independent noise sources.

7.6 Sea of Boltzmann machines: stochastic computing without noise sources

In this work we have investigated two different types of neural networks in terms of their potential to perform stochastic inference. For both network architectures, the L23 networks in Chapter 3 as well as LIF-based sampling networks introduced in Chapter 5, a common functional component was external stochastic input in form of presynaptic Poisson stimuli. In both systems, stochasticity is vital to perform meaningful computation. For the sampling framework, we have shown potential applications using LIF-based Boltzmann machines and highlighted the importance of using an efficient substrate for sampling. We have further emphasized the potential of mixed-signal neuromorphic computing, in particular boasting a 10^4 speed-up compared to biological runtime.

For an emulation of stochastic networks, the neuromorphic platform needs to be supplied with stochasticity. In particular, in the studied large-scale stochastic networks (Section 6.2), each of the 1984 neurons ideally would require independent Poisson sources. Due to bandwidth and routing limitations, external noise and on-chip generated noise is still not enough to sustain such network sizes (*Kungl*, 2016). This means that the existing background noise needs to be shared among network neurons to provide a basic level of stochastic drive to the networks. The resultung shared input correlations are harmful for our sampling networks, but it is possible to adjust the networks by training, as seen in the previous section.

In this section we will go a step further and extend our LIF sampling framework to sample from Boltzmann machines without providing any external noise. These networks will be referred to as *sea of Boltzmann machines* throughout this chapter, since they consist of a pool of interconnected BMs that share their output spikes as noise (illustration in Figure 7.13). Since we aim to omit external input, such a system can be

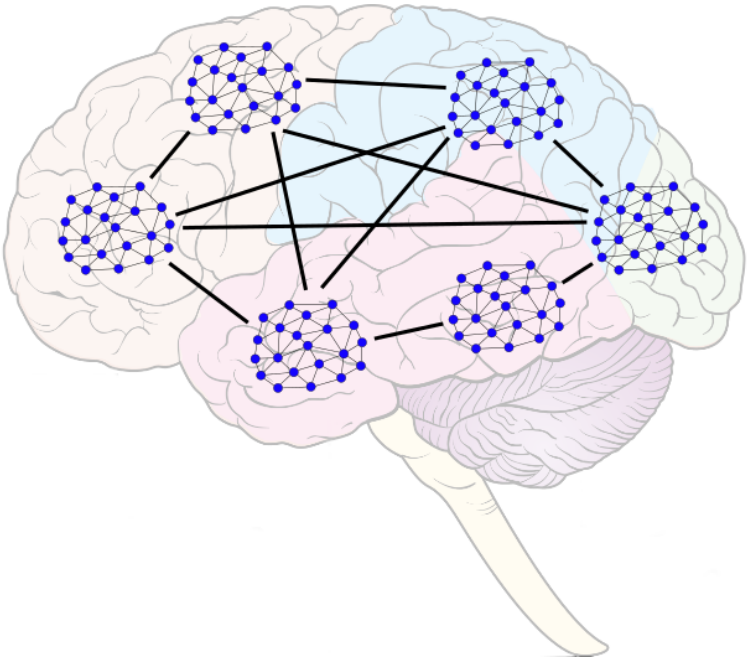

Figure 7.13: The closed neural system illustrates the goal of this section, which is to build a self-supplying network that does not need external stochastic spike trains for stochastic computing. In our functional approach, the network consists of computational nodes which sample from Boltzmann distributions. In a sparse connectivity scheme, the spiking output of the stochastic computing nodes will serve as stochastic drive for the remaining BMs in the network. This way, no external input is necessary to maintain network ability. This does not only solve practical issues of neuromorphic bandwidth limitations, but also steps forward to model functional networks which encode states and generate stochasticity in the same operational regime. Figure is taken from *Dold* (2016).

considered a closed system where the spatial correlations have the same cause as the pairwise correlations that we have studied so far. Results in the previous section suggest that pairwise correlations are not prohibitive for sampling performance if the networks are adjusted to the correlations by training. One main difference which we will have to look at is that the correlations result from BM-generated noise, not from Poisson statistics.

The results in Section 7.6 were achieved in collaboration with Dominik Dold and are published in the Master's thesis, *Dold* (2016), which was supervised by the author of this thesis. The simulations are conducted by Dominik Dold.

An important aspect of such closed neural systems is the difficulty to maintain a network activity. One of the common attributes of such closed networks is that they

cannot sustain stable, irregular activity. The activity is not regulated with outside control parameters, but is fed back through the other network components. In many cases, the firing activity either recedes or synchronizes the network, destroying the stochasticity that is vital to computation. These are common problems that are also discussed in literature (*Kriener et al.*, 2014; *Kumar et al.*, 2008b). Before we investigate these problems, we will first look at the properties of the BM-generated noise compared to the Poisson spike trains that we have used thus far as background input.

7.6.1 Correlations in Boltzmann machine spike trains

In a sea of Boltzmann machines each LIF-based BM samples from a predefined Boltzmann distribution. The output spike trains of these BMs are at the same time used as background input for other BMs in the network. Each background input that a neuron receives, consists of multiple spike trains from other LIF neurons. The neurons in the BMs that generate the noise are connected via Boltzmann weights, and therefore generate spike trains that have autocorrelations. These autocorrelations can be measured within any spike train that originates from a neuron that is connected to other neurons in a BM (Figure 7.14).

First, we will compare the difference between BM-generated spike trains and Poisson background noise (Figure 7.14). We will not yet connect multiple BMs, but only investigate the impact of autocorrelations inside a spike train that is generated by an LIF neuron in a BM. We will look at the activation function of an LIF neuron that is stimulated by BM-generated noise input to assess the impact of these autocorrelations. We remember that the activation function describes the firing statistics of an LIF neuron as a firing probability $p(z = 1)$, encoding the probability of its binary states. There are two properties in BM-generated output that are important for the LIF firing statistics. The first property is the minimum interspike interval τ_{ref} which can be found in any LIF-generated spike train. Since a neuron is deactivated for a refractory period after every output spike, there is a minimum interval that regularizes the spike trains. The second important property is the occurrence of activity bursts of a BM-neuron in case of strong BM interactions. This also increases regularity, which is harmful for stochastic inference.

The impact of these effects will be evaluated by running a BM where one neuron is stimulated by BM-generated background, as illustrated in Figure 7.14. This background input is generated from a fully functional LIF-based BM (blue) consisting of three neurons. The generated output is applied as background input into each neuron of a BM (red) that samples from a distribution. For now, we will refer to the noise-generating BMs as BM_n, and the background-receiving, computing BMs as BM_c. Note that both types of BMs sample from well-defined Boltzmann distributions. The distributions for the BMs were again drawn from a beta distribution, as described in Appendix A.6. The Poisson input rates and weights of the BM_n networks were chosen $\nu_{\text{inh}} = \nu_{\text{exc}} = 400\,\text{Hz}$ and $w_{\text{inh}} = w_{\text{exc}} = 0.001\,\mu\text{S}$ respectively to suffice the high-conductance state.

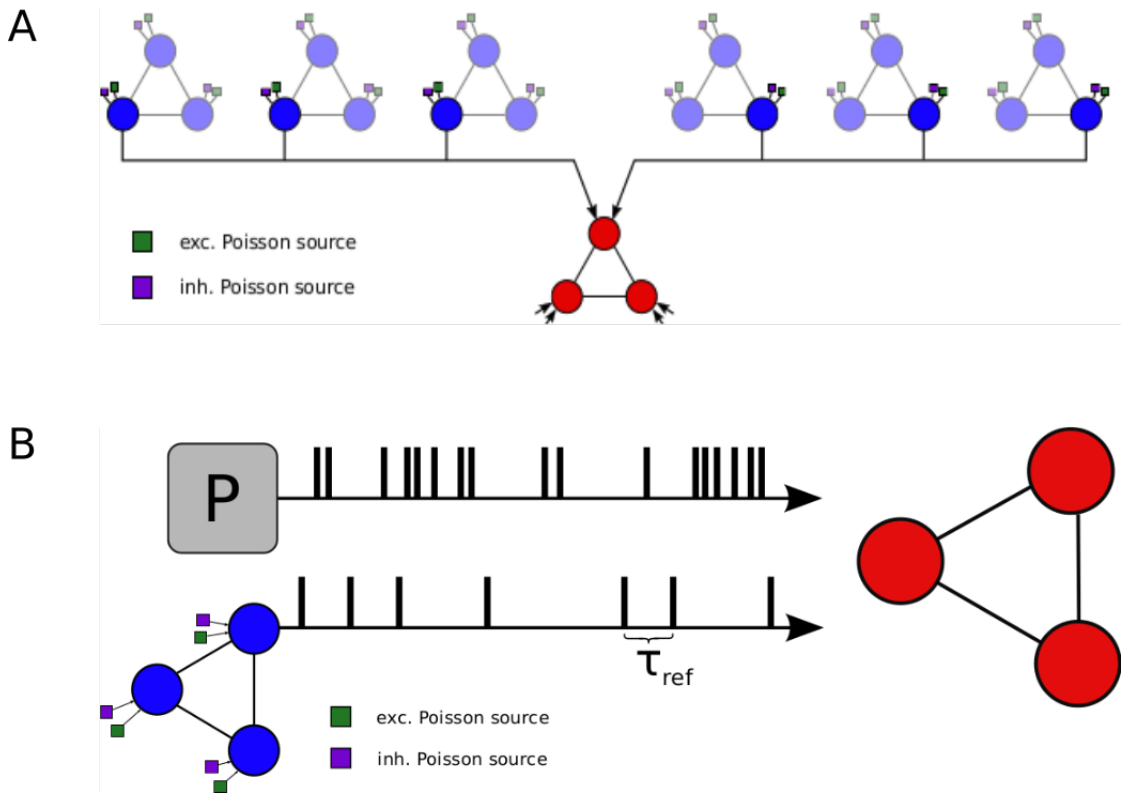

Figure 7.14: **(A)** We evaluate the sampling properties of neurons in the red Boltzmann machine where each neuron (red) receives excitatory and inhibitory background input from merged spike trains from different BMs (blue). The marked input lines on the other two red neurons also consist of balanced inhibitory and excitatory connections. In this setup the blue BMs serve as background noise generators and receive uncorrelated Poisson noise, consisting of excitatory (green) and inhibitory (purple) input. **(B)** The difference between a Poisson background generator and background generated by a BM neuron are interspike intervals of at least τ_{ref}. These interspike intervals regularize the background spike trains. Figure is taken from *Dold* (2016).

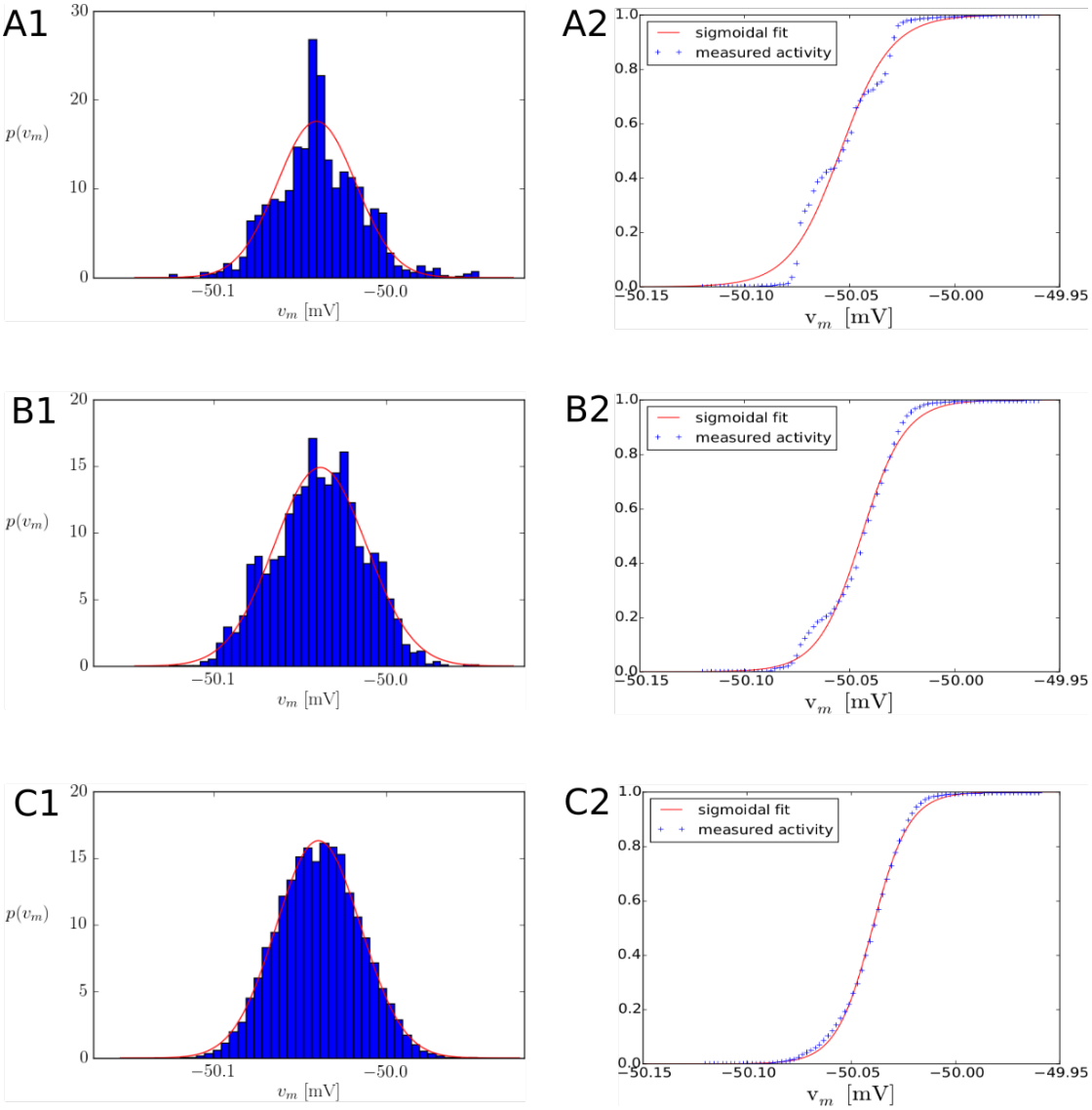

Figure 7.15: On the left (A1, B1, C1) we see free membrane potential (no spiking mechanism) distributions of an LIF neuron receiving BM-generated noise ($\nu_{\text{noise}} = 0.9997\tau_{\text{ref}}^{-1}$) with strong autocorrelations. On the right (A2, B2, C2) we see the corresponding activation functions. **(A)** For a simulation runtime of $T_{\text{sim}} = 10^4$ ms we see artefacts in the shape of the distribution (left) and in the curve of activation function (right). **(B)** As we increase the runtime to $T_{\text{sim}} = 5 \cdot 10^4$ ms, the distribution resembles a Gaussian and the activation function becomes smoother. **(C)** The simulation time is further increased to $T_{\text{sim}} = 10^5$ ms. If the simulation time is sufficiently long, we can reliably fit a logistic function (red curve) to the measured firing rates $p(z = 1)$, as can be seen in the bottom right plot. Figure is adapted from *Dold* (2016).

In the measurement results of the activation function ($T_{\text{sim}} = 10^5$ ms) we see that high firing activity in BM_n indeed impacts the quality of the firing statistics of BM_c neurons (Figure 7.15). The reason for this is the increased regularity of the calibration spike trains, which distorts the firing statistics. However, a significantly longer calibration time recovers a smooth activation function, since regularity artefacts in the spike trains are averaged out during longer simulation runs. The activation functions, as well as the membrane potential distributions resulting from the BM-generated noise (Figure 7.15) can be fitted to a logistic function, σ, and Gaussian distributions, respectively.

To assess the quality of sampling of a BM_c, we set up a six-neuron BM_c that is supplied with noise from a BM_n. We compare the sampling performance of BM_c with the performance from Poisson-driven BMs by using the DKL. The parameters for both, BM_c and BM_n, were drawn from a beta distribution,

$$W \propto W_0 \cdot \mathcal{B}\left[(0.5, 0.5) - 0.5\right] \quad , \tag{7.31}$$

$$b \propto 1.2 \cdot \mathcal{B}\left[(0.5, 0.5) - 0.5\right] \quad . \tag{7.32}$$

The simulation results can be seen in Figure 7.16 and confirm that the amplitude of weights, W_0, affects the sampling quality of BM_c. Simulations in (*Dold*, 2016, Chapter 3) show that the weight amplitude in the noise-generating BMs, BM_n, does not affect the sampling quality of BM_c.

These results already show potential for sea of BMs, since the autocorrelations do not prohibit a good sampling performance. The implemented network setups thus far do not reflect a real use case of interconnected BMs, as only one BM_c is supplied with correlated noise and there are no interconnections between different BMs yet (Figure 7.14).

After we have evaluated the impact of autocorrelations in BM-generated spike trains we will increase the amount of spatial correlations by including all spike trains generated in a BM_n as background noise for BM_c networks. In three-neuron BM_c networks, two neurons will receive BM_n-generated background noise, while the third neuron will still receive independent Poisson input(illustrated in Figure 7.17). There are still no connections from BM_c to BM_n. In this setup we will test the influence of the additional crosscorrelations between neurons in the BM_c network, since many of them emerge from the same BM_n network. To quantify only the background-propagated correlations, we do not connect neurons in the BM_c network, and set the parameters ($\boldsymbol{W}_{\text{comp}} = \mathbf{0}, \boldsymbol{b}_{\text{comp}} = \mathbf{0}$). The lack of synaptic connections guarantees that there are no correlations introduced in BM_c networks and the zero bias is set for simplicity, without loss of generality. The number of BM_n networks is chosen such that their output rate supplies each BM_c neuron with at least 800 Hz for excitation and inhibition.

Since we use multiple background spike trains from the same BM_n, we introduce additional correlations. These crosscorrelations between noise spike trains result in correlations in the BM_c ouput in Figure 7.18. It shows three distinctly colored distributions

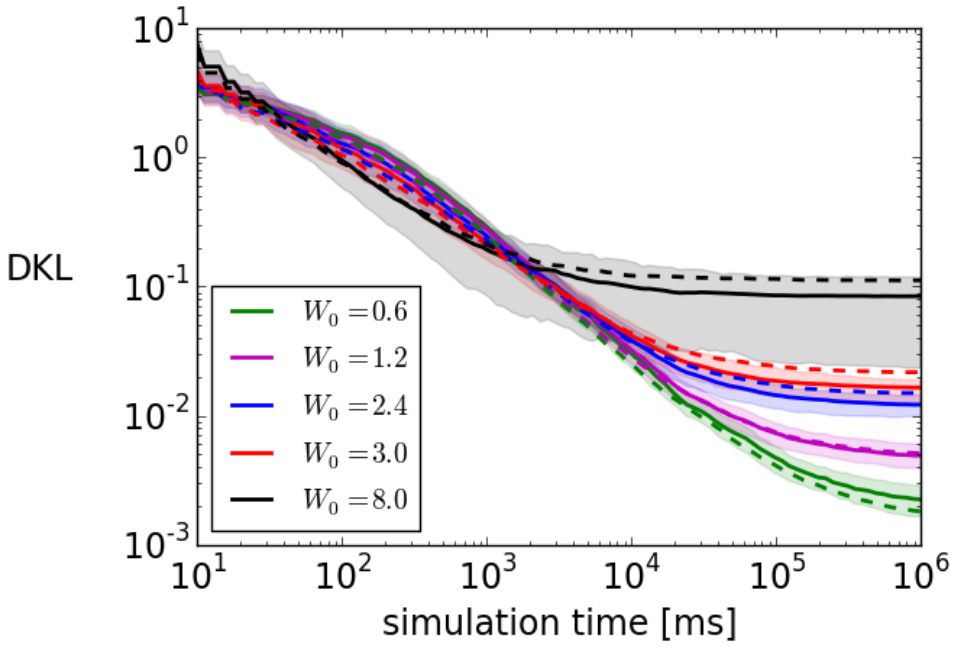

Figure 7.16: The six-neuron BM_c networks are initialized with target Boltzmann parameters drawn from a beta distribution. The colored full lines show the sampling performance of networks receiving BM_n-generated noise and the dashed curves show the DKL of Poisson-driven BMs. The DKL curves measure the sampling quality for different weight amplitudes W_0, where each curve is averaged over 24 different BM_c networks. We see that the performance heavily depends on the synaptic weights of the distributions in the BM_c networks, W_0. Lower synaptic weights lead to less systematic deviations, resulting in a lower DKL. Interestingly, for the smallest weight, $W_0 = 0.6$, the networks on average perform better than their Poisson counterparts. The synaptic weights for the noise-generating BM_n have the same amplitude, $W_0 = W_0^{\text{noise}} = 1.2$, in all simulations. The shaded areas enclose the 15th and 85th percentile. Figure is taken from *Dold* (2016).

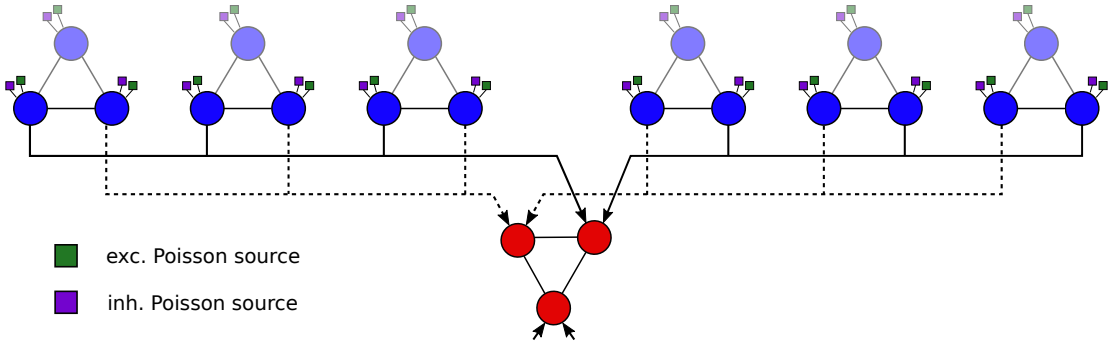

Figure 7.17: The setup shows noise-generating BMs (blue) and computing BMs (red). Two of the three neurons in the red network receive merged spike trains that are generated in the same BM_n. This introduces correlations between different background spike trains that are injected into the BM_c networks. The third neuron in BM_c receives independent noise sources. Figure is taken from *Dold* (2016).

of measured correlation coefficients, resulting from three different simulation setups. All three setups have in common that the Boltzmann weights between the BM_n pair of neurons are negative. Therefore, the background spike trains have a negative correlation coefficient. The simulation setups differ in the way the BM_n-generated background is connected to the BM_c neurons. Depending on the synapse configuration of the background spike trains to the BM_c neurons, the negative correlations will remain or invert into positive ones. In the left histogram, the negative correlation from the BM_n spike trains remains negative since the background is connected to identical synapses of the neuron pair in BM_c, "EE" or "II". The right histogram in Figure 7.18 shows the case where the generated negative correlation in the background is inverted into a positive one, since the synapses of the BM_c neuron pair differ (either "EI" or "IE"). This propagation of correlation is reflected in the location of the histograms, as they show either a negative mean correlation (same synapse type) or a positive mean correlation (different synapse type). This shows that we can modulate the correlations exhibited by BM_c-generated output by adjusting the synapse types to the background noise.

In the histogram in the center (Figure 7.18), the synaptic connections $BM_n \longrightarrow BM_c$ are random, resulting in a correlation mean of $\rho = 0$. On average, the negatively correlated background spike trains are connected inhibitorily or excitatorily with the same probability to the BM_c neuron pairs, therefore yielding a net correlation of zero. This is an interesting and encouraging finding, since random connectivity is the simplest method and does not demand for sophisticated routing schemes on the hardware to decrease correlations. A drawback in this result is the large spread of the center histogram in Figure 7.18, which suggests that the network performance is still suffering from correlations.

To verify this assumption, we will use the randomized connectivity scheme between BM_n and BM_c neuron pairs in a three-neuron BM_c network. The BM parameters are again drawn from the same beta distribution as before. The sampling results were quanti-

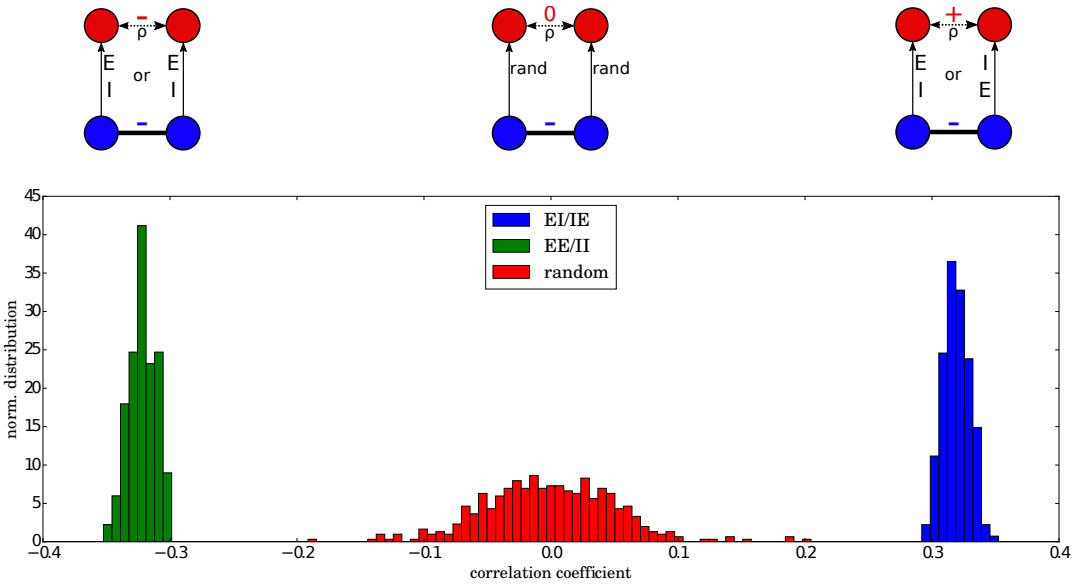

Figure 7.18: We connect three-neuron BM_c networks with BM_n-generated noise, where two BM_c neurons receive background input with at least 800 Hz. The weights from BM_n are drawn from a beta distribution, according to $W^{\text{noise}} \propto 6 \cdot [\mathcal{B}(0.5, 0.5) - 0.5]$. The W^{noise} weights are all negative, therefore the BM_n output spike trains propagate negative correlations to the BM_c networks. Inside the BM_c network there are no connections, so the net correlations between BM_c neurons result from the background noise injected to two neurons. The bottom histograms show the correlation coefficient resulting from three different types of connectivity of BM_c to the BM_n-generated background connections (same synapses, different synapses, randomized synapses). We see that random connectivity yields a mean pairwise correlation of zero, which suggests that decorrelation works by randomizing the connectivity to the BM_c neurons. The distribution shows still a large spread, indicating that there is still shared noise correlation in the network. Each histogram includes 200 simulations. Figure is taken from *Dold* (2016).

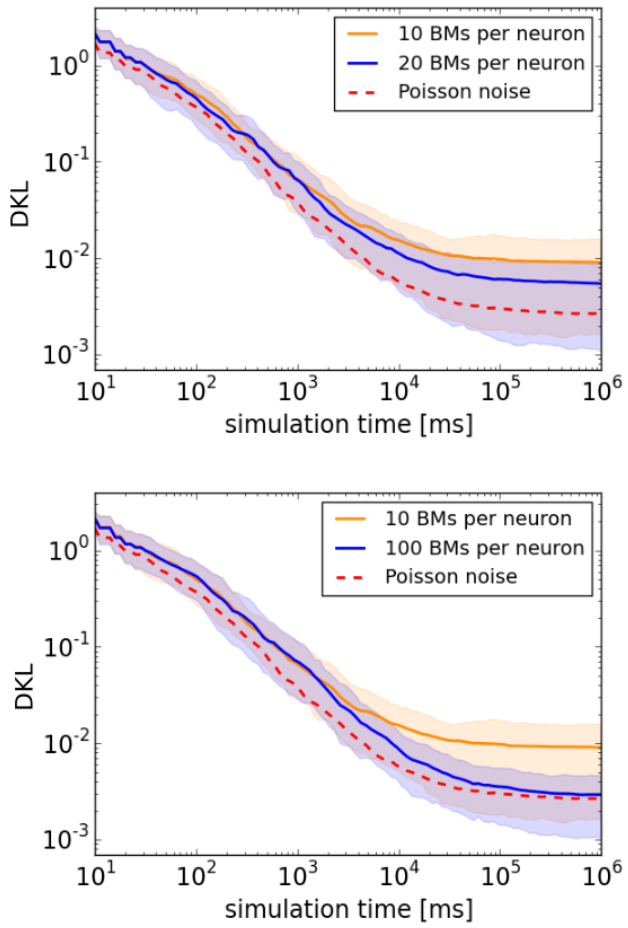

Figure 7.19: We simulate three-neuron BM_c networks with the Boltzmann parameters drawn from a beta distribution, according to Equations 7.31, 7.32. The parameters for the BM_n networks are identical to the ones in Figure 7.18. **(Top)** The yellow DKL curve on the y-axis shows the sampling quality of BM_c networks in runtime on a logarithmic scale for 10 BM_n networks. The blue DKL curve shows the sampling quality of BM_c supplied by 20 BM_n networks. The colored areas mark the interval between the 15th and 85th percentile of DKL curves, which were averaged over 24 different Boltzmann machines. We see that an increased number of noise-supplying BM_n networks causes more accurate sampling (i.e., lower DKL) for the blue curve. Still, we do not achieve the quality of purely Poisson-driven networks (dashed red), which show a significantly lower DKL. **(Bottom)** We increase the number of noise-supplying BM_n networks to 100 (blue curve). This drastically improves sampling, as the DKL average almost reaches the quality of Poisson-driven networks (dashed red). We have plotted the DKL curve for 10 BM_n nets (yellow) again for comparison. The error bands and the number of simulation runs are the same as in the upper plot. Figure is taken from *Dold* (2016).

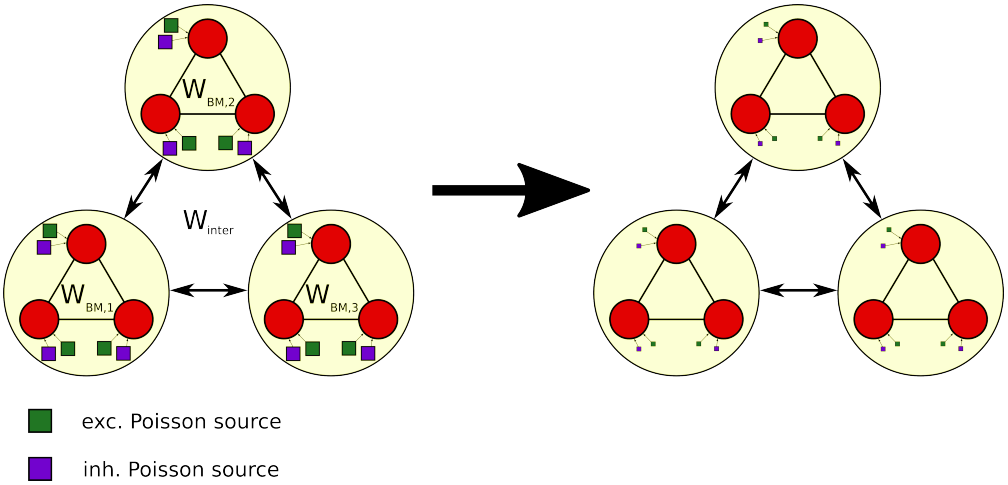

Figure 7.20: Two types of network architectures are shown. On the left, a network of BMs is shown, which has internal connections $W_{\mathrm{BM,\,i}}$ and interconnections W_{int}. The internal connections and the biases define the target Boltzmann distributions. The function of the interconnections W_{int} is to transmit the output spikes between each BM to apply it as background sources. Additionally, each neuron receives excitatory (green) and inhibitory (blue) Poisson sources. These sources serve as kickstart mechanism to provide a reasonable activity to the network. On the right, we see the same network where the frequencies of the Poisson sources are decreased. In our simulations, we will start with both, Poisson input as well as BM-generated input to stabilize the network activity. The Poisson rate is then reduced continuously during runtime, until the BM networks supply each other with BM-generated background noise only. Figure is adapted from *Dold* (2016).

fied via DKL and are shown in Figure 7.19. The results suggest that random connectivity is a viable method to mitigate correlations from $\mathrm{BM_n}$-generated spike trains. An increase of noise-supplying $\mathrm{BM_n}$ networks has a positive effect on the sampling performance, as the probability for pairwise correlation patterns decreases for an increasing background source pool.

7.6.2 A closed network of Boltzmann machines: removing external sources

We have now verified that sampling quality of $\mathrm{BM_c}$ networks improves if the network size grows. Since BM networks can cope with shared noise correlations in the background noise, we will now interconnect all BMs such that there will be no distinction anymore between noise-generating BMs ($\mathrm{BM_n}$) and computing BMs ($\mathrm{BM_c}$). All neurons will be grouped into BMs which will be interconnected. First, each neuron will receive Poisson background, as well as BM-generated background from the network-generated noise. After initialization we will gradually reduce the external Poisson frequency and assess

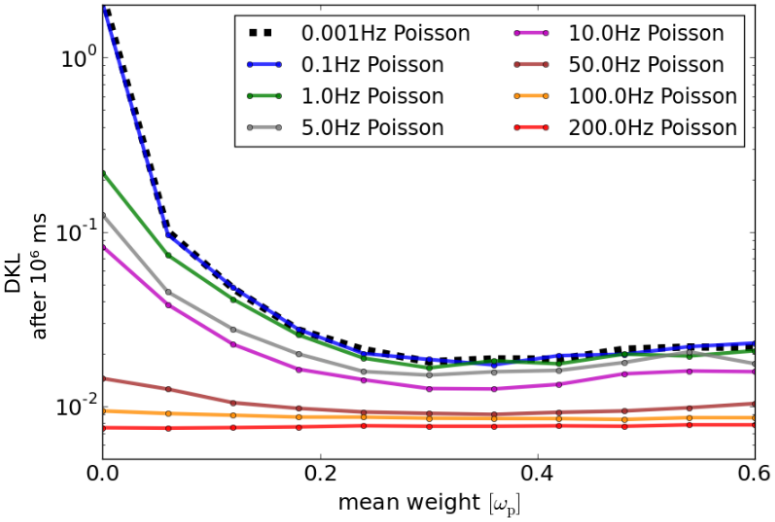

Figure 7.21: The plot shows the sampling performance results of 200 three-neuron BMs after a sampling time of $T_{\text{sim}} = 10^6$ ms with Poisson rate reduction (see Figure 7.20). We varied the final Poisson frequencies ν_{target} and the mean interconnection weights μ_{int} in the unit of Poisson weights, $[w_p]$. For a mean weight of zero, $\mu_{\text{int}} = 0$, the DKL becomes higher, as the sampling performance decreases. Strong weights, in turn, still guarantee good sampling quality, even for decreasing Poisson rates. As long as the interconnection weights remain strong enough, the neurons stay in the high-conductance state, ensuring the LIF sampling properties. If the weights become too large, though, the DKL rises again, with declining quality. In such a scenario, the mean interconnection weights, μ_{int}, become high enough to distort the interaction within each BM. In such a case, the synaptic impact of the background noise is in the order of magnitude of the interactions within a BM. Figure is taken from *Dold* (2016).

the performance of the network operating only on BM-generated background input (Figure 7.20).

We set up a network of 200 three-neuron BMs with randomly drawn weights (beta distribution) and biases that define each Boltzmann machine.

We define an initial background frequency ν_{start}, which serves as an approximation of the total target frequency of the sampling network. To define the parameter translation from the BM domain to the LIF domain, we first measure the activation function of the BMs without interconnections, although connections within the BMs are present. The background noise is external Poisson background. We measure the resulting output spike trains and set the LIF parameters according to this Poisson-based calibration (see Section 5.2.1). To perform sampling, we then interconnect the BMs but still add Poisson noise

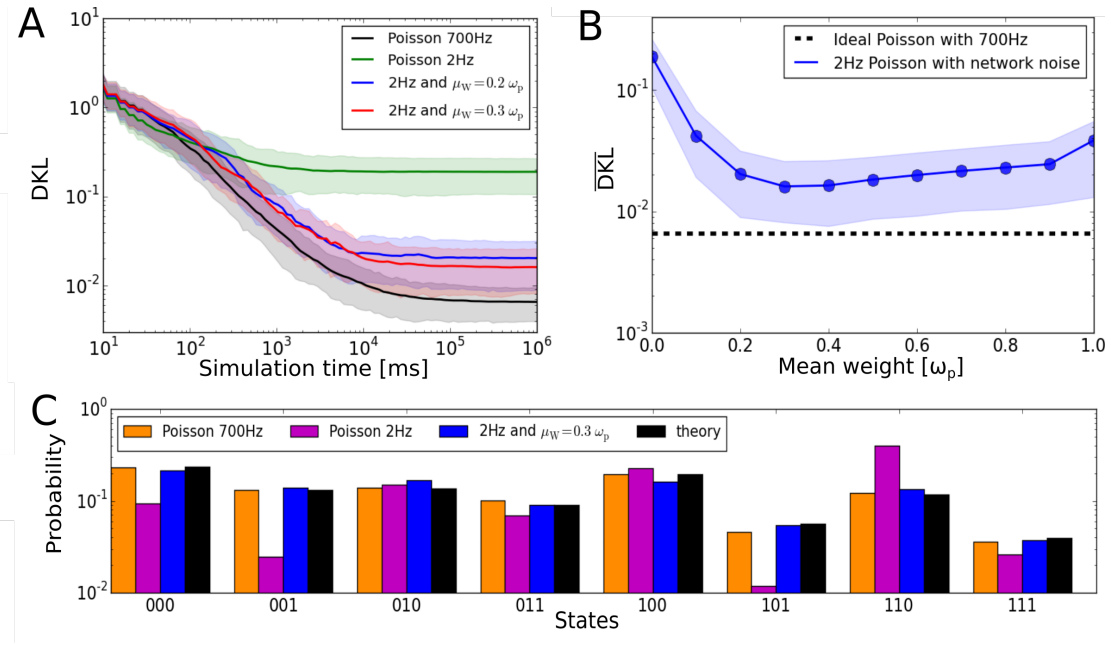

Figure 7.22: The sampling quality of a network consisting of 200 three-neuron BMs with reduced Poisson noise is compared to a Poisson-driven network after a runtime of $T_{\text{sim}} = 10^6$ ms. **(A)** The DKL curves indicate the performance of the sampling networks. For a Poisson-driven network with only 2 Hz input (green), the network does not receive enough stochastic input to reliably sample from the complete state space. This misrepresentation of the targeted probability distribution causes the relatively high DKL. For a BM with interconnections and mean interconnection weights $\mu_W = 0.2w_p$ and $\mu_W = 0.3w_p$, we see significant improvement (blue, red curves). Still, the Poisson-driven network (black curve) with a 700 Hz input shows the best sampling performance. The colored areas indicate the interval between the 15th and 85th percentile from the mean. **(B)** We vary the mean interconnection weights to find the optimal value with only 2 Hz Poisson input and network-generated input. This value lies close to $\mu_W^{\text{opt}} \approx 0.3w_p$. For weaker interconnections, the high-conductance state is not guaranteed and increasingly strong synapses distort the dynamics within the BMs. **(C)** The sampled distribution of a three-neuron network after a runtime of $T_{\text{sim}} = 10^6$ ms shows that the sea of BM networks (blue) achieves very good sampling using a Poisson rate of only 2 Hz and an interconnection strenght $\mu_W = 0.3w_p$. The purely Poisson-driven networks with a frequency of 700 Hz (yellow) and the BM-driven networks are both very close to the theoretical computation of the probabilities (black). This suggests that the removal of Poisson rates does not impair the sampling and the network can operate on BM-generated background noise. Figure is taken from *Dold* (2016).

with a lower frequency ν_{target}. This frequency defines the target frequency of the network. The network will now be calibrated again with an approximate rate of $\nu_{\text{start}} + \nu_{\text{target}}$, resulting from the BM interconnections and Poisson noise. Here, ν_{target} is the rate of the Poisson process and ν_{start} is the approximate rate of the background generated by interconnected BMs.

To start sampling, we again connect the BM networks and initialize with a Poisson frequency of ν_{start}. Since the network has been set up to sample with an approximate rate of $\nu_{\text{start}} + \nu_{\text{target}}$, the network will initially sample erroneously, as it receives background with an approximate rate of $2 \cdot \nu_{\text{start}}$ from both the BM interconnections and Poisson noise. We accept this intentional discrepancy because the higher rate at the beginning serves as a kickstart effect and asserts that network activity is maintained. After this rate stability is assured at the beginning, the Poisson rate is reduced, according to

$$\nu(t) = \nu_{\text{start}} - \frac{\nu_{\text{start}} - \nu_{\text{target}}}{1 + \exp\left(-\frac{t - t_\nu}{\sigma_\nu}\right)} \quad . \tag{7.33}$$

The function describes a logistic decrease of the frequency $\nu(t)$, where ν_{target} defines the target frequency, which will be set at the end of the process. The parameter $t_\nu = 5 \cdot 10^3$ ms approximately defines the midpoint of the rate reduction and $\sigma_\nu = 10^2$ ms determines the smoothness of the frequency reduction. As the Poisson rate decreases, the BMs provide the main supply of stochasticity. Note that the network is set up to sample with a total background Poisson rate of ν_{target} and the output spikes from other BMs, which are approximated as ν_{start}. After the Poisson rate reduction period we arrive at the correct network firing frequency, as shown in Figure 7.21 (top). Remarkably, it is possible to reduce the Poisson rate considerably almost without loss of sampling quality as long as the networks have sufficiently strong network connections. Depending on the interaction weights between the BMs, the performance can get close to the ideal Poisson-only case, as seen in Figure 7.21(center, bottom).

These results suggest that a replacement of Poisson noise by BM-generated output yields good sampling results, but the BM-output driven network still achieves visibly lower performance than the Poisson-driven networks. The main cause for this are shared noise correlations in the network, as indicated by the large spread of the histograms. Although the network has a very sparse BM-to-BM connectivity, there are still correlations left that distort the sampling mechanism. From the simulation of three-BM networks we can see the evaluated distribution of correlation coefficients in Figure 7.23. The correlations decrease with the increase of network size, leading to improved sampling. This can be formalized by looking at the connectivity ϵ. In a sea of BMs with N neurons, each neuron receives an average of $\epsilon \cdot N$ inputs. A neuron pair then shares $\epsilon^2 \cdot N$ inputs on average. In our networks, the total number of received inputs per neuron is set to

$$n_{\text{in}} = \epsilon \cdot (N_{\text{BM}} - 1) \cdot N_{\text{nrn}} = 20 \quad . \tag{7.34}$$

In Figure 7.23 we see histograms for BM networks with a BM count from $N_{\text{BM}} = 10$ up to $N_{\text{BM}} = 150$. Since the number of total inputs per neuron remains fixed at $n_{\text{in}} = 20$,

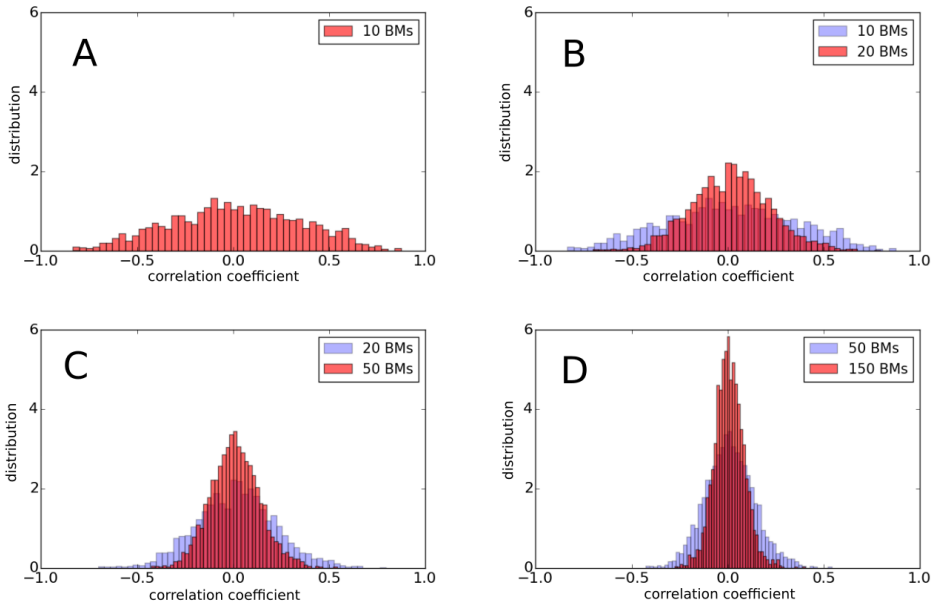

Figure 7.23: Shown are the pairwise correlation coefficient histograms of a sea of BMs. **(A)** The sea of BMs generate their own background noise and the BM interconnections are governed by a random connectivity pattern. The correlation coefficient distribution is centered around zero due to the random connectivity between BMs, but is very broad due to shared input correlations. The correlation coefficients were evaluated for random neuron pairs within a BM. **(B)** A doubling in network size from 10 BMs (blue) to 20 BMs (red) decreases the spread of the distribution, since the number of interconnections is kept constant and the probability of pairwise correlations within a BM decreases. **(C)** With an increasing number of BMs and a constant number of interconnections in the network, the histogram width (red) narrows further. **(D)** For a network consisting of 150 BMs, the interconnectivity ($\epsilon = 5\%$) and the spread decrease further. The blue distributions serve as comparison to networks with higher connectivity. For all distributions, up to 500 neuron pair correlations were evaluated. Figure is taken from *Dold* (2016).

the connectivity decreases with $\epsilon \propto \frac{1}{N_{\mathrm{BM}}}$. Then the number of shared inputs decreases with $\epsilon^2 \cdot N_{\mathrm{BM}} \propto \frac{1}{N_{\mathrm{BM}}}$. As a result, the correlation coefficient also decreases with an increasing number of N_{BM}, as seen in Figure 7.23.

These results already indicate a successful implementation of BM networks that can sample without Poisson noise. We will aim to further improve the performance of sampling without Poisson inputs by using the results from Section 7.1. In particular, we have seen that the harmful impact shared input correlations can be mitigated by training Boltzmann parameters to suffice the pairwise firing statistics. Since our preliminary training results in Section 7.1 showed potential, our goal will be to adapt the closed BM networks to the inevitable shared noise correlations in closed networks.

7.6.3 Training Boltzmann machines without external noise sources

In all our simulations thus far we have included external noise sources at least at the beginning of the simulation. We have started to slowly remove the Poisson input during a simulation run, slowly adjusting the network to BM-generated background noise. Even with only a negligible amount of external Poisson noise left, the network performed very well. We will now run the network entirely without external noise and remove the shared input correlations by training.

As stated before, a main concern regarding closed networks are the risks of network activity divergence from the target activity. There are two potential scenarios that can occur in a closed neural network. If the network activity is not sufficient to sustain all network neurons, the activity will subside and eventually cease. Therefore, we need to assert continuous network firing. In the second scenario, network activity can increase disproportionately due to excessive firing until regular firing patterns persist and prohibit stochastic inference. Such regular firing modes reduce stochasticity until sampling is not possible anymore. To avoid both scenarios, we need to set up the interconnectivity of BMs accordingly, considering two important network parameters. The first important parameter is the ratio between inhibitory and excitatory connections that an LIF neuron receives from other BMs as background, $\eta = \frac{N_{\mathrm{inh}}^{\mathrm{syn}}}{N_{\mathrm{exc}}^{\mathrm{syn}}}$. The second one is the connectivity of the network, ϵ, which we discussed in the previous section. The lower the connectivity, the sparser the network becomes and the less likely neuron pairs share a background source. Combinations of both these parameters can be used to set up an appropriate firing regime. Different firing modes of LIF-based Boltzmann machines were investigated in (*Korcsak-Gorzo*, 2015) and showed that different configurations of these two parameters enable a variety of different firing modes. We will use these results and evaluate the sampling quality of a closed network by varying η and ϵ. More general studies of sparse network firing dynamics can be found in (*Amit and Brunel*, 1997b; *Brunel*, 2000; *Kumar et al.*, 2008b; *Kriener et al.*, 2014; *Deco and Jirsa*, 2012).

In our simulations we will use smaller networks than in the previous section, since training demands a time-consuming iterative simulation scheme. Our networks will con-

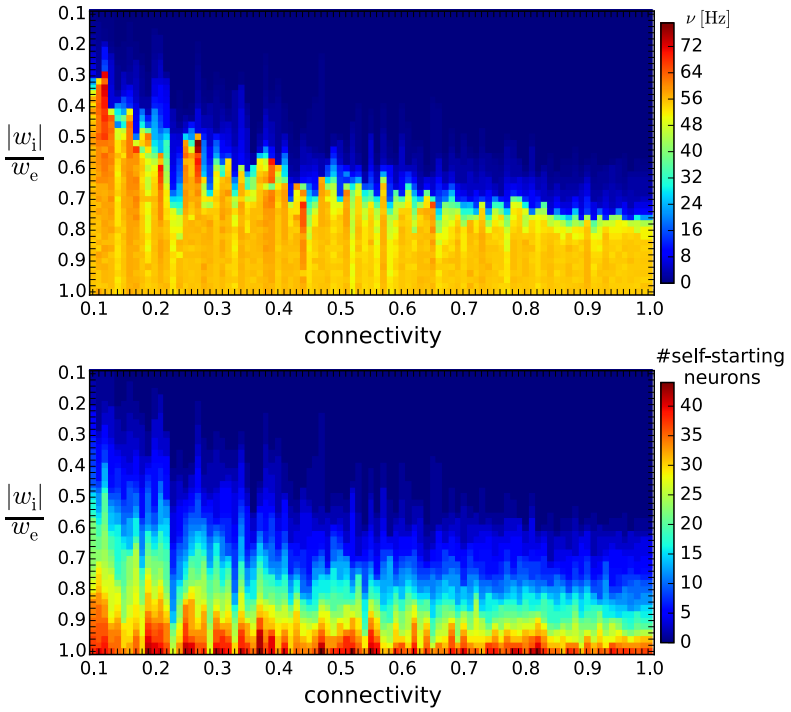

Figure 7.24: **(Top)** The heatmap shows the network firing rate ν for a variation of the connectivity (x-axis) and the ratio of excitatory and inhibitory synapse strengths (inverted y-axis). The network firing rates are set to $\approx 50\,\text{Hz}$. For a constant connectivity on the x-axis, the yellow regions of these stable firing rates are found for higher inhibition, i.e., higher $g = \frac{|w_i|}{w_e}$. As we increase connectivity, these regions become smaller because synchronization of inhibition occurs. The smaller $g = \frac{|w_i|}{w_e}$ is chosen, the fewer kickstarting neurons are initialized (i.e., neurons with above-threshold resting potential) to ensure a reasonable firing activity. For low g values, the probability increases for activity to cease (blue area). A higher connectivity ϵ on the x-axis synchronizes the network and thereby limits the parameter region where the target firing rate is robust. **(Bottom)** The number of kickstarting neurons can be seen on the y-axis. For decreasing g, the number of kickstarting neurons decreases. If g becomes too low, there is not sufficient excitation at the beginning of the network simulation because there are too few neurons with leak potential above threshold. This results in a low network firing rate, as seen in the upper plot. For high g, there is an abundance of kickstarting neurons, enabling the network to converge to the target firing rate ν. All in all, the quantity of kickstarting neurons coincides well with the robustness of the firing rate. Figure is taken from *Dold* (2016).

sist of 10 ten-neuron BMs that will be initialized without external input. We will vary the parameters ϵ and g to investigate network dynamics. Again, the BM parameters are drawn randomly from a beta distribution (see Appendix table A.10). To measure the activation function, we initialize Poisson inputs with rates $\nu_{\text{inh}} = \nu_{\text{exc}} = 700\,\text{Hz}$ without BM interconnections. After measuring the activation function of each network neuron, we set the connectivity ϵ and inhibition/excitation ratio g. The connections are chosen randomly, while the ratio g is set by randomly drawn excitatory weights from a beta distribution within the interval $\mu_W \pm \sigma_W$. Here, μ_W is the mean of the distribution and σ_W the interval of the beta distribution. We draw the absolute values of the inhibitory weights from the same interval and rescale them according to ratio g. More information on these parameters can be found in *Dold* (2016, Appendix 8.8.8).

We use the previously recorded spike trains as input for the interconnected network to adjust the activation function to the added interconnections. The activation functions are determined to a large degree by ϵ and g, since they influence the firing statistics of every LIF neuron in the network. For a predominantly inhibitory setup with $g > 1$, the network activity can only be maintained if the leak potential E_l is set to a value above the spiking threshold θ. This is automatically adjusted during the mapping of the BM parameters $(\boldsymbol{W}, \boldsymbol{b})$ to the LIF domain, as described in Section 5.2. Since the inhibitory feedback is present during measurement of the activation function, the LIF leak potential is set correspondingly to match the firing activity. For the parameter pair (ϵ, g), we have found configurations at which stable network activity is asserted. Most importantly, the ratio g determines the number of neurons starting with an above-threshold leak potential, $E_\text{l} > \theta$. The higher g is set, the more inhibition the network receives. The LIF neurons are initialized with a high leak potential to counteract this high inhibition rate from the interconnections. We can see the number of these kickstarting neurons in Figure 7.24 (bottom) and their impact on the network firing rate. For a large parameter regime of g and ϵ, the networks can be kickstarted reliably. In general, $g > 1$ yields a stable network activity because there are more above-threshold neurons.

We have confirmed that networks can maintain activity without external noise, so we can now address our initial concern of this section, namely reducing the correlations by training to improve sampling. We evaluate the training algorithm by setting up a network in which shared input correlations are the only source of correlations, i.e., $(\boldsymbol{W} = \boldsymbol{0}, \boldsymbol{b} = \boldsymbol{0})$. This way we do not introduce BM-weight-related correlations, which could mask the correlations introduced by shared inputs from BM interconnections. We use 40 ten-neuron BMs in the unconnected network. The interconnections between the BMs will have connectivity $\epsilon = 5.2\%$ and weight ratio $\eta = 4$. The measurement protocol of the activation function will be according to the description in the previous section. First, we will use Poisson sources with a frequency of $\nu_{\text{inh}} = \nu_{\text{exc}} = 700\,\text{Hz}$ without setting BM interconnections and record the resulting network spike trains. We then set interconnections and use the spike trains acquired from the previous Poisson-driven simulation run. Then we again record the activation function and readjust the LIF parameters according to the simulation run with interconnections. Now there will be no external Poisson sources during the sampling process, and only BM interconnections pro-

vide background input. The network activity is kickstarted by above-threshold neurons with predominantly inhibitory connections ($\eta = 4$). We know from the simulations in Figure 7.23 that correlations persist. To cancel out these correlations, we will train the BM parameters $(\boldsymbol{W}, \boldsymbol{b})$ with the CD algorithm as described in Section 7.5.5,

$$\Delta \boldsymbol{W}_{ij} = \eta(t_{\text{train}}) \left[p(z_i = 1, z_j = 1)_{\text{target}} - p(z_i = 1, z_j = 1)_{\text{net}} \right] \quad , \tag{7.35}$$

$$\Delta \boldsymbol{b}_j = \eta(t_{\text{train}}) \left[p(z_i = 1)_{\text{target}} - p(z_i = 1)_{\text{net}} \right] \quad . \tag{7.36}$$

Again, we refer to $p(z_i = 1, z_j = 1)$ as probabilities of correlated pairwise firing. We do not use Ising states here, since we already know that training $(\boldsymbol{W}^{\text{I}}, \boldsymbol{b}^{\text{I}})$ and $(\boldsymbol{W}, \boldsymbol{b})$ parameters is equivalent for the CD scheme. The learning rate $\eta(t_{\text{train}})$ is a function of training iterations t_{train},

$$\eta(t_{\text{train}}) = \frac{400}{2000 + t_{\text{train}}} \quad . \tag{7.37}$$

The functional shape and parameters in $\eta(t_{\text{train}})$ have been set after correspondance with Luziwei Leng and are a result of parameter sweeps. The differences in $\eta(t_{\text{train}})$ compared to the learning rate in Equation 7.30 result from different network sizes and connectivity. The decline of the learning rate asserts that the parameter changes do not overshoot into a particular direction after a training step. This lowers the risk of paramter fluctuation during training.

As in Section 7.5.5 we train the pairwise firing probabilities $p(z_i = 1, z_j = 1)$ between neuron pairs i and j. In a Poisson-driven network, the probability distribution is uniformly distributed and the correlation coefficient is zero, since we have set $(\boldsymbol{W} = \boldsymbol{0}, \boldsymbol{b} = \boldsymbol{0})$ for now. We train the LIF network for 500 training steps according to Equations 7.35, 7.36 to eliminate the correlations. The results in Figure 7.25 (top) show the evolution of pairwise correlation coefficients between randomly drawn neuron pairs from the same BM. The correlations decrease significantly already after several hundred training iterations. The trained network has a correlation coefficient distribution that becomes very similar to the one from the ideal Poisson case.

After we have minimized the correlations in our closed network, we will use a similar setup to reevaluate the sampling performance of the trained network. This time we set up a practical case with nonzero BM parameters, $(\boldsymbol{W}_{\text{target}}, \boldsymbol{b}_{\text{target}})$, drawn from the beta distribution (see Appendix A.8) with a weight amplitude $W_0 = 2$. Again, the total number of background inputs per neuron is set to 20 with a connectivity $\epsilon = 5.2\%$. The setup of LIF parameters via measurement of the activation function, as well as the training scheme, are performed in the same way as in the previous simulation. A comparison of the resulting DKL to Poisson-driven sampling can be seen in Figure 7.26. Note that the trained BM parameters $(\boldsymbol{W}_{\text{net}}, \boldsymbol{b}_{\text{net}})$ are different from $(\boldsymbol{W}_{\text{target}}, \boldsymbol{b}_{\text{target}})$, since they were modified during training to minimize the impact of shared input correlations. We have also compared our trained networks to trained

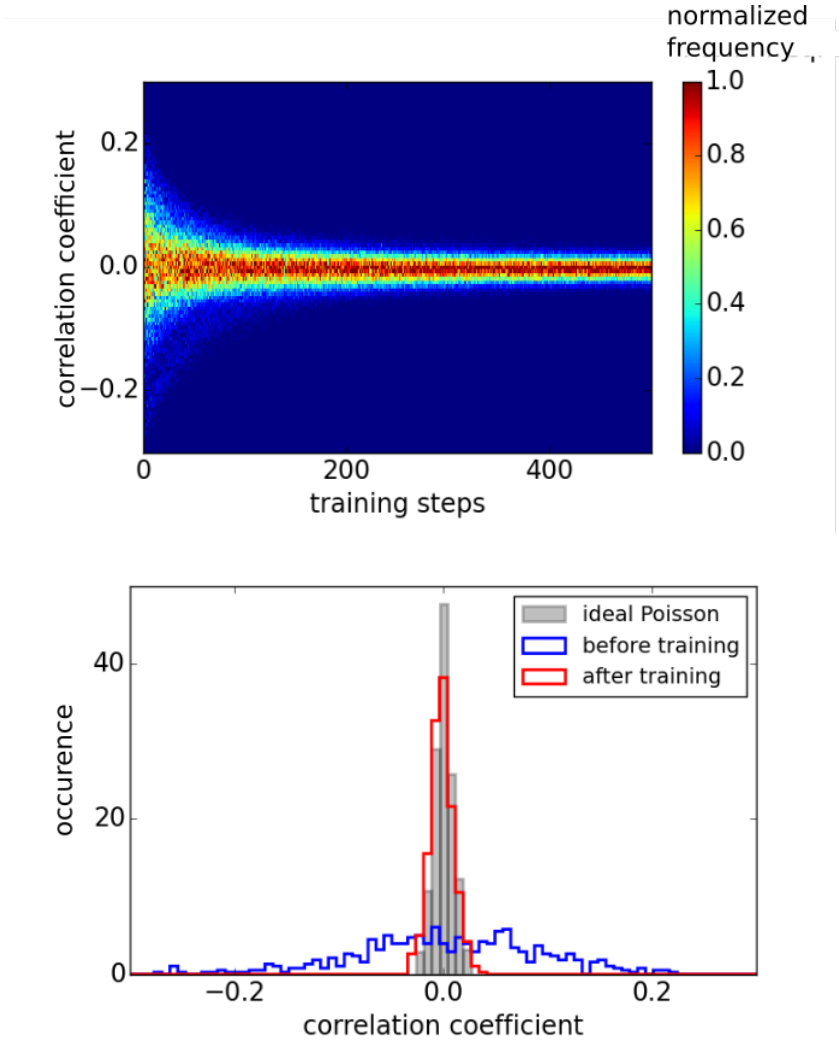

Figure 7.25: **(Top)** The evolution of the pairwise correlation coefficient is shown during the training of 40 ten-neuron Boltzmann machines with ($\boldsymbol{W} = \mathbf{0}, \boldsymbol{b} = \mathbf{0}$). The training was performed using the contrastive divergence algorithm (CD, Equations 7.35, 7.35). Since the LIF neurons in a BM have no connections, the only sources of correlation are shared background inputs. Training the Boltzmann parameters in 500 training steps minimizes these correlations significantly already after 200 training iterations. **(Bottom)** The histograms show a comparison of the correlation coefficients before (blue) and after (red) training the network. The training after 500 iterations shows substantial improvement and yields results similar to the quality of a Poisson-driven network. Figure is taken from *Dold* (2016).

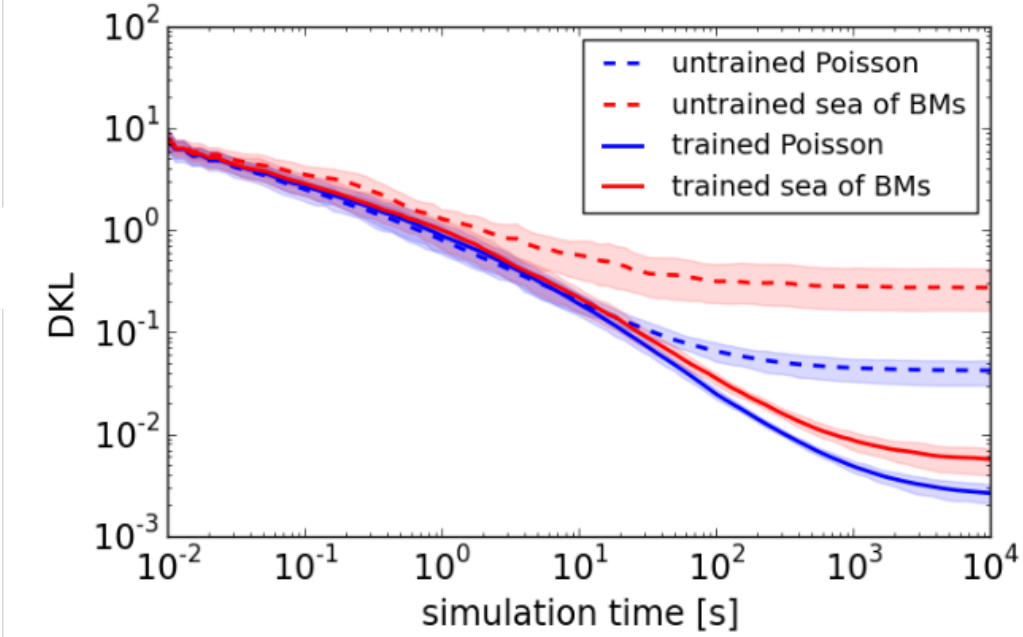

Figure 7.26: Comparison of the sampling performance of different network realizations. The DKL serves as a measure for performance on the log-log scale. An untrained sea of BMs (dashed red) shows poor performance, as untreated correlations distort the sampled probability distributions. An untrained Poisson-driven network (dashed blue), although performing significantly better than an untrained sea of Boltzmann machine, still suffers from weight-induced systematic deviations. The trained networks show a remarkable improvement compared to the untrained networks. As expected, trained Poisson networks (blue) show the lowest DKL. The trained sea of BMs (red), however, still performs better than untrained Poisson-driven networks and a close to the Poisson-driven case in terms of sampling performance. The results are averaged over all BMs in the network. The colored areas mark the interval between the 15th and 85th percentile over all DKL measurements. Figure is taken from *Dold* (2016).

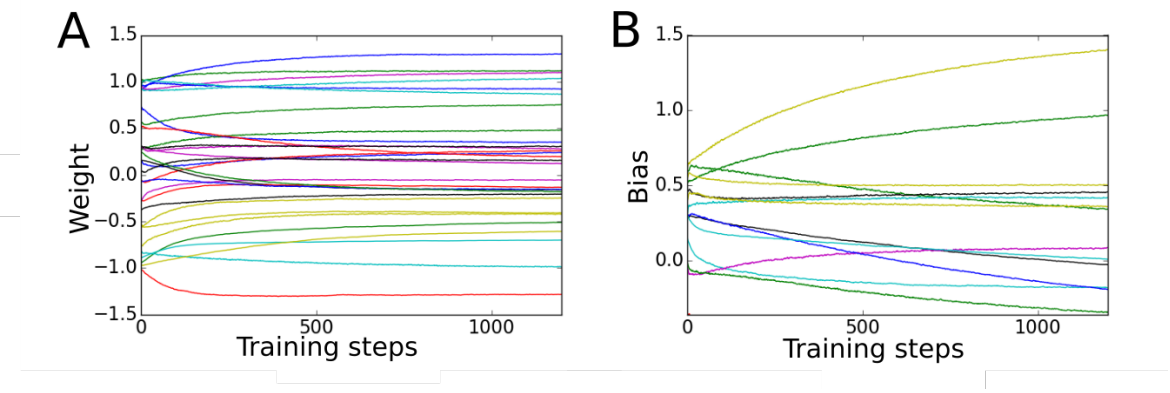

Figure 7.27: **(Left)** Shown are the weights and biases from the simulation runs of Figure 7.26. The parameters were selected randomly from the set of trained parameters $(\boldsymbol{W}_{\text{net}}, \boldsymbol{b}_{\text{net}})$. The weights converge after only ≈ 100 training steps and remain stable. **(Right)** The convergence of the biases could be improved, potentially by introducing a more elaborate training scheme while keeping the number of training iterations low. Still, the trained networks perform very well, as shown in Figure 7.26. Figure is taken from *Dold* (2016).

Poisson-driven networks and conclude that the quality is very similar. As we found out in Section 7.5.5, training not only removes shared input correlations, but also mitigates systematic deviations caused by the obligatory BM-to-LIF parameter translation. Therefore, even a Poisson-driven network benefits immensely from training. We can see the convergence of the trained weights and biases of the sea of BM network in Figure 7.27.

In conclusion, we can state that it is possible to sample from BMs without external noise sources, using only network-inherent activity. Here, the BM-generated spike trains that are used for encoding probability distributions are also used as stochastic background sources. This underlines the unification of two important aspects in computational neuroscience that are often regarded as separate network components: information encoding on one side, and the network-driving stochasticity on the other side. We have developed a framework where sampling is accomplished in a closed system without external stochasticity. Further studies regarding the stability of network activity can be pursued to characterize the robustness of these closed systems extensively. In terms of network parameters, high g ratios increase the number of above-threshold neurons (kickstarting neurons), whose activity increasess the robustness of the network. Note that our network configurations are different from network setups in studies of so-called self-sustained networks (*Kumar et al.*, 2008b; *Kriener et al.*, 2014). These studies implemented the leak potentials of the LIF neurons below spiking threshold and incorporated strong inhibition with delays. This guaranteed asynchronous irregular firing modes, which are important firing modes observed in biology. Since delays are not compatible with the LIF sampling

paradigm (see Figure 6.12), we cannot use them to decorrelate the network to a substantial degree. We have already seen in the hardware implementation in Section 6.4 that delays distort the sampled distribution. However, in (*Korcsak-Gorzo*, 2015) it was shown that asynchronous irregular firing modes can also be established in Boltzmann machine networks without delays.

At this point we can conclude that the training of the network, even in its simplest form, can fundamentally restructure the network to adapt to persistent correlation structures of the underlying substrate.

8 Conclusion & Outlook

In the present work we focused on stochastic computing as a key component of neural computation. Our aim was to explore the potential of biological architectures and mechanisms to perform stochastic inference using spiking neurons. In these networks, we have also seen how dynamics facilitated by adaptation and short-term plasticity improve their performance in the computational task they solve.

8.1 Cortical attractor networks

Our initial approach in Chapter 3 was to evaluate the capability of an AdEx-based cortical network to solve inference tasks. In this model, the network stores and retrieves information by switching between an irregular firing mode and so-called attractor states (Section 3.1). To quantify the performance of the network on pattern completion tasks, we have introduced the so-called memory characteristic (Section 3.3). The memory characteristic is a meaningful measure because it characterizes a general attribute of networks, namely their proficiency to store and retrieve information after stimulus is received. In this measure, the network size and the number of stimulated columnar units during the task are taken into account. For an increasing number of stored patterns, the interpretation of network states, and conclusively, the assessment of the performance became increasingly difficult (Section 3.5.1). By changing the existing conditions for attractor state detection, we were able to evaluate the network performance even for high pattern densities. The memory characteristic could be applied as an universal benchmark tool to compare the storage and retrieval performance of different network architectures.

8.2 The behavior of leaky integrate-and-fire neurons under stochastic stimulus

In Chapter 3 the evaluation of cortical network states was difficult to achieve for high pattern densities. To understand network dynamics, a reasonable approach is to first fully characterize single neuron dynamics in the irregular, spontaneous cortical firing regime. Since an analytical description of AdEx firing activity is not possible, our goal in Chapter 4 was to formalize the stochastic firing dynamics of single LIF neurons in the biological high-conductance state, as such states are observed in cortical in-vivo networks (Section 4.3). We have provided a novel description of the firing statistics of a LIF neuron, which enabled us to predict the activation function for a wide range of parameter regimes, for which a prediction was not possible by using existing approaches (*Brunel and Sergi*,

1998; *Moreno-Bote and Parga*, 2004). In particular, our approach allows us to consider burst firing modes that are characterized by strong autocorrelations due to long synaptic time constants. This characterization could even be extended to provide an analytical description of cross-correlations between neuron pairs resulting from shared input.

8.3 Probabilistic computing with leaky integrate-and-fire neurons

The properties of LIF neurons in the high-conductance state enabled us to build networks that perform stochastic inference, as described in Chapter 5. We have derived precise translation rules between an abstract sampling domain and the LIF domain. This configuration requires a translation between the neuron and synaptic parameters of LIF networks and Boltzmann parameters (Section 5.2). This allows us to use LIF neurons to sample from the general class of Boltzmann probability distributions (Section 5.2). As we have shown, LIF-sampled distributions converge reliably to their target distribution even in more extreme regimes, such as strong weights and high connectivities. Moreover, we have also shown their robustness to variations of many LIF neuron and synapse parameters as long as the membrane potential reacts sufficiently fast to synaptic input. This is an attribute of the membrane potential during the high-conductance state (Sections 5.4, 5.5). It should be pointed out that an important difference to biological constraints is the violation of Dale's principle in Boltzmann machines, since Boltzmann units can exhibit excitation and inhibition to their postsynaptic partners. In cortical architectures, such as the ones described in Chapter 3, inhibitory and excitatory interaction is carried out by different populations in the minicolumns, thereby obeying Dale's principle. To adapt to such biological constraints, implementations of LIF-based sampling networks in cortical L23 architectures can be explored. Such network architectures could potentially provide a conceptual link between both domains, sampling spiking neurons and cortical computation.

8.4 Applications of leaky integrate-and-fire Boltzmann machines

Using the LIF-based sampling framework, we have built generative models of the MNIST data set using a restricted Boltzmann machine topology (Section 6.2). The classification rates of LIF-based RBMs (96.4%) are similar to Gibbs-sampled networks (96.7%), using one hidden layer for feature encoding. While the discriminative performance of both models is similar, the Gibbs-based model suffers from a low state transition rate during the training. The potential walls in the energy landscape impede transitions towards other high-probability states. To compensate for this disadvantage, the Gibbs algorithm is often extended by tempering mechanisms, such as CAST. This requires additional computational effort, as it is necessary to reshape the energy landscape to improve the generative properties (Section 6.2.1). The LIF-based generative model uses

short-term plasticity to locally reshape the energy landscape and overcome potential barriers. This enables the LIF-based network to traverse between high-probability modes more efficiently and achieve a better generative performance. Moreover, the short-term plasticity mechanism enables LIF networks to adapt to heterogeneous data sets. In such data sets, some classes are overrepresented, including more training data. LIF networks sample from all classes with the same frequency because short-term plasticity reshapes the energy landscape as soon as the network stays too long in a certain mode that represents a class (Section 6.2.2). In the Gibbs sampling case, such homogenization requires additional preprocesing of the target data set.
Although the performance of LIF-based sampling is already very good, it could be further improved by introducing modulation of the noise input to LIF samplers. This would be analogous to an annealing mechanism. We have seen that the inverse temperature $\beta = \frac{1}{kT}$ is an important property in Boltzmann distributions (Section 6.2.1). For single Boltzmann units, it determines the slope of the activation function and thereby the probability of state transitions. In background-stimulated LIF networks, the slope of the activation function depends on the amount of presynaptic noise input. This establishes an analogy between the functional role of temperature in physical systems and noise in neural networks. For instance, to facilitate improbable state transitions during sampling, the amplitude of background noise could be modulated, controlling the transition rate in the networks. Such a mechanism could be used to identify the functional role of time-dependent modulation of diffusive input in neural networks.

Above we have pointed out the benefits of spiking neurons with biology-inspired mechanisms compared to conventional sampling algorithms. A key reason for the efficiency of biological networks is a substrate which respects the parallel computing architectures of biological networks. In contrast to simulations, the dynamics of the sampling units evolves in parallel, allowing to increase the network sizes while maintaining a constant network runtime. Our platform of choice, the Spikey chip, offers a speed-up factor of 10^4 compared to biological networks. For our stochastic networks, this speed-up increases the convergence speed of the sampled distributions towards the target distributions. Different to simulation, physical delays are inevitable on neuromorphic hardware and fundamentally violate the representation of states assumed in the sampling framework (Section 6.4). To adjust our sampling networks to finite delays, we have modified our network architecture by implementing an artificial refractory mechanism that is controlled by auxiliary neuron pools. We have implemented Boltzmann machines of three random variables and demonstrated fast convergence of the sampled probability towards the target probability distribution (Figure 6.15). Despite the limitations imposed by physical delays, neurons on the Spikey chip could still be used as the hidden layer of an RBM architecture, since there are no intra-layer connections. In such a setup, the device could be used as a classifier for data provided by a connected host computer.
For a neuromorphic implementation of multiple layers, a wafer-scale platform offers several advantages. In the case of the BrainScaleS system (Section 6.4.2), the refractory times can be set sufficiently long to neglect the present physical delays during sampling. The network runtime is also accelerated by a 10^4 speed-up and offers a flexible config-

urability of the neuron and synapse parameters. Due to the large number of parameters in large scale networks, there is a need for automatized training procedures during which the hardware parameters are adjusted to suffice a predefined functionality metric. This so-called *in-the-loop* software framework is currently developed in the VISIONS group and implements such a training procedure. A representational network, such as the LIF-based RBM, could be trained to discriminate images, with the functionality metric being the classification rate. We have already demonstrated the contrastive divergence update rules for an iterative adjustment of network parameters. However, it would be beneficial to explore more biology-oriented training mechanisms, like the so-called spike-timing-dependent plasticity (STDP) mechanism. Combining on-line training with contrastive divergence update rules could further boost the efficiency of inference on hardware, since the network would not need to be reset after a parameter update. Instead, synapses would be adjusted during a single hardware emulation run according to correlations in post- and presynaptic firing patterns. To combine the STDP mechanism with training methods, work has already been done and is currently ongoing (*Weilbach*, 2015).

8.5 Noise correlations

In the previous sections we have presented our results in implementing sampling-based inference using spiking neurons. So far, in the investigated networks, the functional role of noise consisted in generating a stochastic firing regime for LIF neurons in high conductance. In general, temporal noise is either an intrinsic property of the network constituents (i.e., neurons and synapses) or is introduced externally. In our software simulations in both the L23 network in Chapter 3, and in the LIF-based Boltzmann machines, the stochasticity was implemented via presynaptic Poisson sources. In Chapter 7, our initial motivation was to transfer large-scale networks on a wafer-scale system. Due to bandwidth limitations on the hardware, it is not possible to supply each of the network neurons with temporal noise (Section 7.1). As a result, network neurons receive common noise sources and exhibit correlated spiking dynamics as a consequence. This raises the question whether these correlations reduce the sampling performance of the networks. This question can be regarded in a more general context, since it affects every stochasticity-driven network. Therefore, we investigate how shared noise-induced correlations affect the computational properties of noise-driven spiking networks. In Section 7.3, we show that temporal noise does not only serve as means to maintain stochastic firing modes in networks, but that it can also play an active role to transmit information. More precisely, the functional role of shared noise can be regarded the same as synaptic weights. We show that an LIF neuron pair that shares background noise has the same computational properties as a pair with a synaptic connection. To show this for sampling units, we translate the network parameters $(\boldsymbol{W}, \boldsymbol{b})$ with states $z \in \{0, 1\}$ to the Ising model domain with states $z^{\mathrm{I}} \in \{-1, 1\}$ (Section 7.4). Due to this functional equivalence between synaptic weights and shared sources, the effects of both can be described by one correlation coefficient (Figure 7.9). We demonstrate the equivalence in two cases. Firstly, we compensate for present shared noise-induced correlations by in-

troducing additional, counteracting synaptic weights (Figure 7.7). Secondly, we define a target distribution which implies nonzero connectivity ($W^{\mathrm{I}} \neq 0$) and successfully sample from the distribution without connecting the neuron pair (Figure 7.8). This means that noise decorrelation and computation do not need to be carried out by separate network architectures, but could be carried out in one network where shared noise and synaptic weights fulfill corresponding roles.

The equivalence of synaptic properties and shared noise also implies that a network can adapt to an imprinted noise architecture to fulfill a certain computational task. We confirm this assumption in Section 7.5.5, where we train LIF networks to adjust to a predefined shared noise configuration. We show that it is possible to adapt the network to an environment with an imprinted correlation structure by training its network parameters, $(\boldsymbol{W}, \boldsymbol{b})$ (Figure 7.11). Additionally, by training the network to sample from a target distribution $p(\boldsymbol{z})$, we also remove LIF sampling-specific systematic deviations that we have discussed in Section 5.2.1. During training, the network parameters are adjusted to minimize the distance to the target distribution, measured by the Kullback-Leibler divergence (DKL).

Addressing the initial problem of how to deal with the inevitable shared noise correlations on neuromorphic hardware, a potential solution would consist in introducing weights and train the network to adapt to the present noise input topology. This differs from existing decorrelation approaches, where additional decorrelating networks are implemented (Section 7.2).

In the results above, we have argued that shared noise sources can assume the same computational role as synaptic weights. Also, we have shown that networks can be trained to sample in the presence of shared noise in Section 7.5.5. This leads to the question whether external noise sources are needed at all for computation. Since networks are able to adapt to shared noise correlations, in principle, networks could be trained to adapt to a setup in which the noise is generated within the network itself without any external noise sources. In Section 7.6 we propose the so-called sea of Boltzmann machines, which is a network consisting of many interconnected Boltzmann machines. In such a network, each Boltzmann machine samples from a target distribution and provides its output spike trains as background noise for other Boltzmann machines in the sea. Of course, shared input correlations arise in the network, potentially affecting the sampling properties. In Section 7.6.3 we train the network to sample without external noise sources, but only using spike trains that are generated within the network. We retain a sampling quality that is similar to a network which receives external Poisson sources (Figure 7.26) and thereby show that sampling networks can adapt to shared input correlations even in such an extreme case (Figure 7.25).

In the sea of Boltzmann machines, we unify all three network functionalities - computation, generation of stochasticity and adaptation to correlations. The network performs computation by sampling and utilizes the output spike trains to preserve the stochastic

firing modes which are crucial for sampling. The same firing patterns can be used to iteratively train the BM parameters to adjust to correlated firing patterns. We show that temporal noise is not merely a prerequisite to provide the ground for stochastic computation in spiking networks, but is a substantial part of computation.

A Appendix

A.1 Acronyms

AdEx Adaptive exponential leaky integrate-and-fire

AI Asynchronous irregular

ANM Abstract neuron model

API Application programming interface

AST Adapted simulated tempering

BAS Basket cell

BM Boltzmann machine

CAST Coupled adapted simulated tempering

CC Pearson product-moment correlation coefficient

CD Contrastive divergence

CV Coefficient of variation

DAC Digital-to-analog converter

denmem Dendritic membrane

DKL Kullback-Leibler divergence

DNC Do not connect

EPSP Excitatory postsynaptic potential

FPGA Field-programmable gate arrays

FPT First-passage time

HC Hypercolumn

HCS High-conductance state

HICANN High Input Count Analog Neural Network

IPSP	Inhibitory postsynaptic potential
ISI	Interspike interval
LFSR	Linear feedback shift register
LHS	Left-hand side
LIF	Leaky integrate-and-fire
L23	Cortical layer 2/3 network
MC	Minicolumn
MCMC	Markov chain Monte Carlo
MNIST	Mixed National Institute of Standards and Technology
NCC	Neural computability condition
NEST	Neural Simulation Tool
ODE	Ordinary differential equation
OU	Ornstein-Uhlenbeck
PCD	Persistent contrastive divergence
PSC	Postsynaptic conductance
PSP	Postsynaptic potential
PYR	Pyramidal cells
RBM	Restricted Boltzmann machine
RHS	Right-hand side
RSNP	Regular spiking non-pyramidal cells
RV	Random variable
SBS	Spike-based sampling
STD	Short-term depression
STP	Short-term potentiation
TSO	Tsodyks-Markram model
t-SNE	t-distributed stochastic embedding
VLSI	Very-large-scale integration
WTA	Winner-take-all

A.2 Cortical layer 2/3 simulation parameters

We provide simulation parameters for the presented results in Chapter 3. Note that the technical information in this section is taken from *Rivkin* (2014). The technical information in Section A.2 has been taken from the supplementary material in *Rivkin* (2014).

Table A.1: Fitted neuron parameters for the L23 model. In each MC module there are 30 PYR cells, two RSNP cells and one BAS cell.

Parameter	PYR	RSNP	BAS	Unit
C_{m}	0.179	0.0072	0.00688	nF
$E_{\mathrm{exc}}^{\mathrm{rev}}$	0.0	0.0	0.0	mV
$E_{\mathrm{inh}}^{\mathrm{rev}}$	-80.0	-	-	mV
τ_{m}	16.89	15.32	15.64	ms
τ_{ref}	0.16	0.16	0.16	ms
$\tau_{\mathrm{syn,\ exc}}$	17.5	66.6	6.0	ms
$\tau_{\mathrm{syn,\ inh}}$	6.0	-	-	ms
ρ	-60.7	-72.5	-72.5	mV
	-61.71	-57.52	-56.0	mV
a	0.0	0.28	0.0	nS
b	0.0132	0.00103	0.0	nA
θ	0.0	0.0	0.0	mV
τ_w	196.0	250.0	0.0	ms
E^{spike}	-53.0	-51.0	-52.5	mV
V_T	-	-	-	mV

Table A.2: Connectivities between L23 populations, synaptic and plasticity time constants.

Pre-Post	type	weight [μS]	τ_{syn} [ms]	U	τ_{rec} [ms]	τ_{facil} [ms]
PYR-PYR (local)	exc	0.004125	17.5	0.27	575.	0.
PYR-PYR (global)	exc	0.000615	17.5	0.27	575.	0.
PYR-BAS	exc	0.000092	6.0	-	-	-
PYR-RSNP	exc	0.000024	66.6	-	-	-
BAS-PYR	inh	0.0061	6.0	-	-	-
RSNP-PYR	inh	0.0032	6.0	-	-	-
background-PYR	exc	0.000224	17.5	-	-	-

Table A.3: Stimulus parameters for the L23 model.

Background	
# of sources per PYR	1
rate	300 Hz
weight	0.000224 μS
Stimulus during pattern completion	
# of sources per MC	5
$p_{\text{L4}\to\text{PYR}}$	0.75
weight	0.0012375 μS
	(30% local PYR→PYR)

Table A.4: Original network structure: connection probabilities.

within an MC	
PYR → PYR	0.25
RSNP → PYR	0.70
between MCs inside the same HC	
PYR → BAS	0.70
BAS → PYR	0.70
between MCs in different HCs	
PYR → PYR	0.30
PYR → RSNP	0.17

Table A.5: Parameters applied during the evaluation of pattern completion.

Parameter	Value or formula
time frame after stimulus	500 ms
minimum on-state duration	100 ms
pre-stimulus buffer time	500 ms
typical on-state duration	300 ms

Table A.6: Parameters applied during the evaluation of pattern completion.

Parameter	Value or formula
time frame after stimulus	500 ms
minimum on-state duration	100 ms
pre-stimulus buffer time	500 ms
typical on-state duration	300 ms
cone - lower border	RSNP rate = 4.5·PYR rate
cone - upper border	RSNP rate = 30·PYR rate
tolerance amount	14 MC states

A.3 The behavior of leaky integrate-and-fire neurons under stochastic stimulus

The technical information in this section is taken from the supplementary material of *Petrovici et al.* (2013). The simulations have been performed with the parameters below, if not stated otherwise in the main text. The choice of parameters is similar to parameters in *Naud et al.* (2008).

C_m	0.1 nF	membrane capacitance
g_l	5 nS	leak conductance
E_l	-65 mV	leak potential
ρ	-53 mV	reset potential
$E_\mathrm{exc}^\mathrm{rev}$	0 mV	excitatory reversal potential
$E_\mathrm{inh}^\mathrm{rev}$	-90 mV	inhibitory reversal potential
θ	-52 mV	threshold voltage
τ_syn	10 ms	synaptic time constant
τ_ref	10 ms	refractory time constant

Table A.7: Neuron parameters used for the simulations in the main manuscript. The parameters apply to simulations in Chapters 4, 5, 7.

The synaptic background noise was implemented as presynaptic spike stimulus by inhibitory and excitatory Poisson stimuli with rates $\nu_\mathrm{inh} = \nu_\mathrm{exc} = 5000\,\mathrm{Hz}$. The excitatory synaptic weight for the noise stimuli was set to $w_\mathrm{p\ exc} = 0.0035\,\mu\mathrm{S}$. The inhibitory weight $w_\mathrm{p\ inh}$ was adjusted to result in $p(z_k = 1) \approx 0.5$ without current stimulus. For above parameters, this happens at an average free membrane potential of $V_g = -55\,\mathrm{mV}$. This determines $w_\mathrm{p\ inh}$ according to

$$\left|\frac{E_\mathrm{inh}^\mathrm{rev} - V_g}{E_\mathrm{exc}^\mathrm{rev} - V_g}\right| = \frac{w_\mathrm{p\ exc}}{w_\mathrm{p\ inh}} \quad . \tag{A.1}$$

A.4 Probabilistic computing with leaky integrate-and-fire neurons

We draw the Boltzmann parameters $(\boldsymbol{W}, \boldsymbol{b})$ from a beta distribution $\mathcal{B}(a, b)$, where we vary constants a and b. These constants determine the inclination and symmetry of the probability density of the distribution. We mainly use symmetric distributions in this work because we want symmetric conditions between positive and negative parameters $(\boldsymbol{W}, \boldsymbol{b})$.

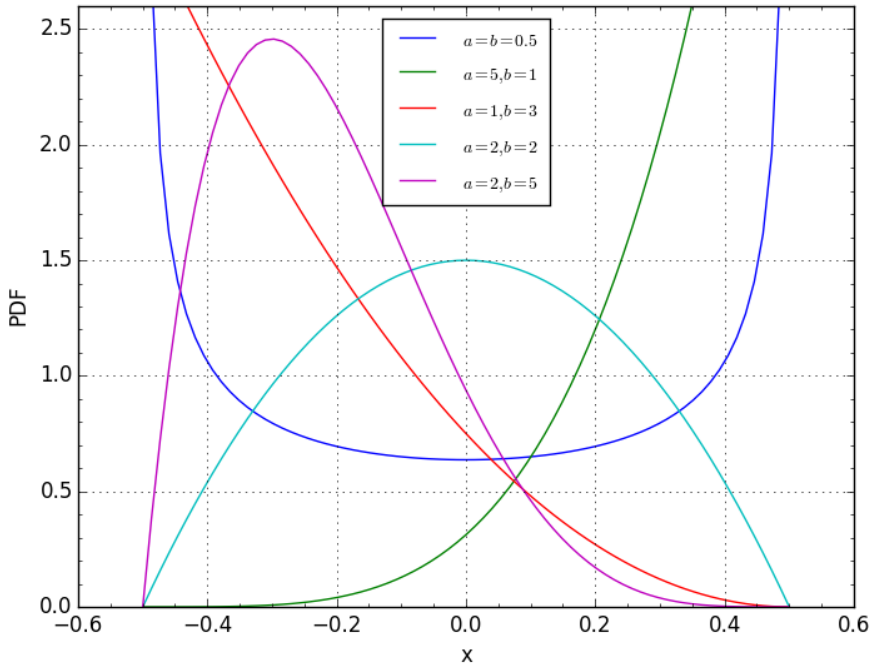

Figure A.1: The beta distribution $\mathcal{B}(a, b) - 0.5$ is shown for different parameters a and b. These parameters determine the slope and and inflection of the probability density. For our setups, we draw our Boltzmann parameters from distributions with $a = b$, since it ensures a symmetry of positive and negative Boltzmann parameters. In particular, for $a = b = 0.5$, the distribution yields BM parameters that are predominantly drawn close to the edges of the parameter ranges. The probability to draw BM parameters close to the origin is low. This way we ensure that the mapped BMs are not trivial uniform distributions. Figure is adapted from *Commons* (2014).

We map the drawn weight and bias values to a parameter interval, $(W_{ij}, b_i) \in [c, d]$. This means that we are mapping the beta distribution to a specific parameter range. The purpose of the beta distribution is to yield parameter values that are bounded to a certain

interval. We choose parameters from a distribution to collect a randomized sample size of the quality of BM networks. To draw random parameters, we need to consider a suitable probability function. Unbounded probability densities (e.g. Gaussians) are inadequate for this purpose, since samples are not bounded. Drawing from unbounded probability distributions could yield arbitrarily large weights and biases. This would destroy network dynamics. In the beta distribution, the constants c and d are set to match the interval in the distribution $\mathcal{B}(0.5, 0.5)$ and are mapped linearly to the interval $[-0.6, 0.6]$. More specifically, the parameters are drawn according to

$$b_i, W_{ij} \sim 1.2 \cdot [\mathcal{B}(0.5, 0.5) - 0.5] \quad . \tag{A.2}$$

Here, we draw from a beta distribution $\mathcal{B}(0.5, 0.5)$ and shift it by -0.5 to center the distribution to the origin. We increase the amplitude of sampled weights and biases (without increasing the range) by 1.2. This makes the probability distribution symmetric so that positive and negative weights and biases have an identical probability to be drawn.

For the large-scale distributions shown in Figure 5.5 we have also drawn from a beta distribution. In Figure 5.5, for each simulation we have drawn the biases according to $b_i \sim 1.2 \cdot [\mathcal{B}(0.5, 0.5) - 0.5]$. For small weights (Figure 5.5A) we have drawn $W_{ij} \sim 0.6 \cdot [\mathcal{B}(0.5, 0.5) - 0.5]$, for intermediate weights (Figure 5.5B) $W_{ij} \sim 1.2 \cdot [\mathcal{B}(0.5, 0.5) - 0.5]$ and for large weights $W_{ij} \sim 2.4 \cdot [\mathcal{B}(0.5, 0.5) - 0.5]$ (Figure 5.5C).

A.5 Stochastic computing with leaky integrate-and-fire Boltzmann machines

The training algorithm used to produce the results in Figure 6.2 C,D was a contrastive divergence algorithm which was trained on a Gibbs sampler, initialized with ($\boldsymbol{W} = \boldsymbol{0}, \boldsymbol{b} = \boldsymbol{0}$). The number of training steps amounted to $T_{\text{train}} = 20000$. We used a linearly decreasing learning rate $\eta = 5 \cdot 10^{-4}$ until $T_{12} = 10000$, and then a constant learning rate $\eta_{12} = 10^{-4}$ was applied.

In Figure 6.2C the network states $\boldsymbol{z}(t)$ from the simulation are projected onto a basis $\boldsymbol{B}$:

$$\boldsymbol{z}^{034}(t) = (\boldsymbol{B}^0 \cdot z(t), \boldsymbol{B}^3 \cdot z(t), \boldsymbol{B}^4 \cdot z(t))^{\mathsf{T}} \quad . \tag{A.3}$$

For a representation in two dimensions, the vector $\boldsymbol{z}^{034}(t)$ is projected onto plane coordinates,

$$\boldsymbol{z}^{\text{proj}}(t) = \begin{pmatrix} \sin(\phi_{\text{B}}^0) & \sin(\phi_{\text{B}}^3) & \sin(\phi_{\text{B}}^4) \\ \cos(\phi_{\text{B}}^0) & \cos(\phi_{\text{B}}^3) & \cos(\phi_{\text{B}}^4) \end{pmatrix} \boldsymbol{z}^{034}(t) \tag{A.4}$$

with $(\phi_{\text{B}}^0, \phi_{\text{B}}^3, \phi_{\text{B}}^4) = (0, \frac{2\pi}{3}, \frac{4\pi}{3})$ being parallel to the basis vectors. On these basis vectors, the three digits are projected. In Figure 6.2C, the network is run in total for 4000 ms, where we take samples of $\boldsymbol{z}(t)$ taken every 2 ms. This means that on the projection plane,

we see a total of 2000 states (blue dots). We have confirmed that this simulation runtime is sufficiently long to represent a meaningful distribution of the Markov Chain.

We can also confirm that most states in the prior distribution are clustered around the digit states $(0, 3, 4)$. To show the state transitions of the Markov Chain, we have illustrated the network states $\boldsymbol{z}(t)$ as a red trajectory. The red trajectory connects 100 network states that are projected on the plane. The total time interval is 200 ms.

The snapshots at the left of Figure 6.2C represent averaged states taken during the projection time interval of 200 ms and show the states of the red trajectory. We average the pixels of the snapshots over a time window of 25 ms with

$$\bar{z}_k(t) = \frac{1}{25\,\mathrm{ms}} \int_{t-12.5\,\mathrm{ms}}^{t+12.5\,\mathrm{ms}} z_k(t')\, dt'. \tag{A.5}$$

The intervals between the snapshots were 25 ms during the total simulation time.

In Figure 6.2D we add incomplete input into the network. We activate four pixels with a positive current, which are located at the center of the image. More precisely, the pixel indices are $\mathcal{I} = \{77, 78, 79, 80\}$ if the pixel counter starts at the top-left and proceeds row-wise. The current amplitude is constant for all four pixels, $I_{\mathrm{px}} = 0.831\,\mathrm{nA}$. For Figure 6.2D we have used the same projection methods that we have described above for Figure 6.2C. We can refer to this input as "ambiguous" in the sense that it activates the 3- and 4-modes. Since positive current causes an excitation, the pixels in the center indicate that the inferred digit is either a "3" or a "4". This is what the network infers and what is seen in the projected states, which are more concentrated in the "3" and "4" domains.

A.6 Stochastic neural networks without stochastic input

The single neuron parameters in Sections 7.5.5, 7.2, 7.3, 7.4, 7.5.1, 7.5.2, 7.5.3, 7.5.4 and 7.5.5 were chosen according to the default parameters in Table A.7, if not stated otherwise in the main text. The parameters for the Boltzmann machine have been drawn from a beta distribution, as described in A.4.

A.6.1 Sea of Boltzmann machines: stochastic computing without noise sources

The full parameter list of Section 7.6 are taken from *Dold* (2016). The parameters used for Section 7.6 will be described in the following tables.

Table A.8: The single LIF neuron parameters below are used in Section 7.6.1. The respective results are shown in Figures 7.14, 7.14, 7.15 and 7.16

C_{m}	0.2 nF	membrane capacitance
τ_{m}	0.1 ms	membrane time constant
$E_{\mathrm{exc}}^{\mathrm{rev}}$	0.0 mV	exc. reversal potential
$E_{\mathrm{inh}}^{\mathrm{rev}}$	-100.0 mV	inh. reversal potential
θ	-50.0 mV	threshold potential
τ_{syn}	10.0 ms	exc. synaptic time constant
E_L	-50.0 mV	leak potential
ρ	-50.01 mV	reset potential
τ_{ref}	10.0 ms	refractory time
I_{offset}	0.0 nA	offset current
$w_{\mathrm{inh/exc}}$	0.001 μS	Poisson noise weights

Table A.9: For Figures 7.22, 7.21, 7.25, 7.26 and 7.27 the below LIF neuron parameters are used. The simulations can be found in Sections 7.6.1, 7.6.2 and 7.6.3.

C_{m}	0.2 nF	membrane capacitance
τ_{m}	1.0 ms	membrane time constant
$E_{\mathrm{exc}}^{\mathrm{rev}}$	0.0 mV	exc. reversal potential
$E_{\mathrm{inh}}^{\mathrm{rev}}$	-100.0 mV	inh. reversal potential
θ	-50.0 mV	threshold potential
E_L	-50.0 mV	rest/leak potential
τ_{syn}	10.0 ms	inh. synaptic time constant
ρ	-50.1 mV	reset potential
τ_{ref}	10.0 ms	refractory time
I_{offset}	0.0 nA	offset current
$w_{\mathrm{inh/exc}}$	0.001 μS	Poisson noise weights

Table A.10: For the closed-network simulations in Figures 7.25, 7.26, 7.27 we used the below simulation parameters. The simulations are described in Section 7.6.3. Note that there are no Poisson sources used. The beta distribution beta(a, b) is described in Section A.4.

#BMs	40
#neurons per BM	10
g	4.0
ϵ	5.13%
η	0.5
μ_{W}	0.001 μS
σ_{W}	0.001 μS
weight distr.	0.0 or $2.0 \cdot \big(\mathcal{B}(0.5,\ 0.5) - 0.5\big)$
bias distr.	$1.2 \cdot \big(\mathcal{B}(0.5,\ 0.5) - 0.5\big)$
calibration time	10^5ms
sampling time	10^5ms
training steps	1200

Bibliography

Abbott, L., J. Varela, K. Sen, and S. Nelson, Synaptic depression and cortical gain control, *Science*, *275*(5297), 221–224, 1997.

Abbott, L. F., and S. B. Nelson, Synaptic plasticity: taming the beast, *Nature neuroscience*, *3*, 1178–1183, 2000.

Abeles, M., *Corticonics: Neural Circuits of the Cerebral Cortex*, Cambridge University Press, New York, 1991.

Abeles, M., G. Hayon, and D. Lehmann, Modeling compositionality by dynamic binding of synfire chains, *Journal of computational neuroscience*, *17*(2), 179–201, 2004.

Ackley, D. H., G. E. Hinton, and T. J. Sejnowski, A learning algorithm for Boltzmann machines, *Cognitive Science*, *9*, 147–169, 1985.

Alberts, B., D. Bray, J. Lewis, M. Raff, K. Roberts, and J. D. Watsonx, *Molecular Biology of the Cell, third edition*, Garland Publishing, Inc, 1994.

Amit, D. J., and N. Brunel, Model of global spontaneous activity and local structured activity during delay periods in the cerebral cortex., *Cerebral cortex*, *7*(3), 237–252, 1997a.

Amit, D. J., and N. Brunel, Model of global spontaneous activity and local structured activity during delay periods in the cerebral cortex, *Cereb Cortex*, *7*(3), 237–52, 1997b.

Barbieri, F., and N. Brunel, Irregular persistent activity induced by synaptic excitatory feedback, *BMC Neuroscience*, *8*(2), 1, 2007.

Baumbach, A., Magnetic phenomena in spiking neural networks, Master thesis, Ruprecht-Karls-Universität Heidelberg, 2016.

Berkes, P., G. Orbán, M. Lengyel, and J. Fiser, Spontaneous cortical activity reveals hallmarks of an optimal internal model of the environment, *Science*, *331*(6013), 83–87, 2011.

Bi, G. Q., and M. M. Poo, Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type., *The Journal of neuroscience: the official journal of the Society for Neuroscience*, *18*(24), 10,464–10,472, 1998.

Bill, J., and M. A. Petrovici, Neural sampling with leaky integrate-and-fire neurons on neuromorphic hardware systems, *Tech. rep.*, 2011.

Binzegger, T., R. J. Douglas, and K. A. C. Martin, A quantitative map of the circuit of cat primary visual cortex, *J Neurosci*, *24*(39), 8441–53, 2004.

Bishop, C. M., *Pattern recognition and machine learning*, vol. 1, springer New York, 2009.

BrainScaleS, Research project, `http://brainscales.kip.uni-heidelberg.de/public/index.html`, 2012.

Brascamp, J. W., R. Van Ee, A. J. Noest, R. H. Jacobs, and A. V. van den Berg, The time course of binocular rivalry reveals a fundamental role of noise, *Journal of vision*, *6*(11), 8, 2006.

Breitwieser, O., Investigation of a cortical attractor-memory network, Bachelor thesis, Ruprecht-Karls-Universität Heidelberg, hD-KIP 11-173, 2011.

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

Brüderle, D., et al., A comprehensive workflow for general-purpose neural modeling with highly configurable neuromorphic hardware systems, *Biological Cybernetics*, *104*, 263–296, 2011.

Brunel, N., Dynamics of sparsely connected networks of excitatory and inhibitory spiking neurons, *Journal of Computational Neuroscience*, *8*(3), 183–208, 2000.

Brunel, N., and S. Sergi, Firing frequency of leaky integrate-and-fire neurons with synaptic current dynamics, *Journal of Theoretical Biology*, *195*, 87–95, 1998.

Bruno, R. M., and B. Sakmann, Cortex Is Driven by Weak but Synchronously Active Thalamocortical Synapses, *Science*, *312*(5780), 1622–1627, 2006.

Buesing, L., J. Bill, B. Nessler, and W. Maass, Neural dynamics as sampling: A model for stochastic computation in recurrent networks of spiking neurons, *PLoS Computational Biology*, *7*(11), e1002,211, 2011.

Burkitt, A. N., A review of the integrate-and-fire neuron model: Ii. inhomogeneous synaptic input and network properties, *Biological cybernetics*, *95*(2), 97–112, 2006.

Buxhoeveden, D., and M. Casanova, The minicolumn and evolution of the brain, *Brain Behav Evol*, *60*, 125–151, 2002a.

Buxhoeveden, D. P., and M. F. Casanova, The minicolumn hypothesis in neuroscience, *Brain*, *125*(5), 935–951, 2002b.

Bytschok, I., From shared input to correlated neuron dynamics: Development of a predictive framework, Ph.D. thesis, Diploma thesis, University of Heidelberg, 2011.

Cáceres, M. O., and A. A. Budini, The generalized ornstein-uhlenbeck process, *Journal of Physics A: Mathematical and General*, *30*(24), 8427, 1997.

Chan, K. C., G. A. Karolyi, F. A. Longstaff, and A. B. Sanders, An empirical comparison of alternative models of the short-term interest rate, *The journal of finance*, *47*(3), 1209–1227, 1992.

Chance, F. S., L. Abbott, and A. D. Reyes, Gain modulation from background synaptic input, *Neuron*, *35*(4), 773–782, 2002.

Commons, W., Probability density function for the beta distribution, 2014.

Compte, A., C. Constantinidis, J. Tegnér, S. Raghavachari, M. V. Chafee, P. S. Goldman-Rakic, and X.-J. Wang, Temporally irregular mnemonic persistent activity in prefrontal neurons of monkeys during a delayed response task, *Journal of neurophysiology*, *90*(5), 3441–3454, 2003.

Davison, A., E. Muller, D. Brüderle, and J. Kremkow, A common language for neuronal networks in software and hardware, *The Neuromorphic Engineer*, doi:doi:10.2417/1201001.1712, 2010.

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

Deco, G., and V. K. Jirsa, Ongoing cortical activity at rest: criticality, multistability, and ghost attractors, *The Journal of Neuroscience*, *32*(10), 3366–3375, 2012.

Destexhe, A., 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*, *3*, 493 – 506, 2009.

Destexhe, A., M. Rudolph, J. M. Fellous, and T. J. Sejnowski, Fluctuating synaptic conductances recreate in vivo-like activity in neocortical neurons., *Neuroscience*, *107*(1), 13–24, 2001.

Destexhe, A., M. Rudolph, and D. Pare, The high-conductance state of neocortical neurons in vivo, *Nature Reviews Neuroscience*, *4*, 739–751, 2003.

Diesmann, M., and M.-O. Gewaltig, NEST: An environment for neural systems simulations, in *Forschung und wisschenschaftliches Rechnen, Beiträge zum Heinz-Billing-Preis 2001*, *GWDG-Bericht*, vol. 58, edited by T. Plesser and V. Macho, pp. 43–70, Ges. für Wiss. Datenverarbeitung, Göttingen, 2002.

Diesmann, M., M.-O. Gewaltig, and A. Aertsen, Stable propagation of synchronous spiking in cortical neural networks, *Nature*, *402*, 529–533, 1999.

Djurfeldt, M., M. Lundqvist, C. Johansson, M. Rehn, O. Ekeberg, and A. Lansner, Brain-scale simulation of the neocortex on the ibm blue gene/l supercomputer, *IBM Journal of Research and Development*, *52*(1.2), 31–41, 2008.

Dold, D., Stochastic computation in spiking neural networks without noise, Master thesis, Ruprecht-Karls-Universität Heidelberg, 2016.

Durrett, R., *Stochastic calculus: a practical introduction*, vol. 6, CRC press, 1996.

Durstewitz, D., J. K. Seamans, and T. J. Sejnowski, Neurocomputational models of working memory, *Nature neuroscience*, *3*, 1184–1191, 2000.

Eccles, J. C., P. Fatt, and K. Koketsu, Cholinergic and inhibitory synapses in a pathway from motor-axon collaterals to motoneurones, *J. Physiol.*, *126*, 524–562, 1954.

Ermentrout, G. B., and D. H. Terman, *Mathematical foundations of neuroscience*, vol. 35, Springer Science & Business Media, 2010.

FACETS, Research project, `http://http://facets.kip.uni-heidelberg.de/`, 2010.

Faisal, A. A., L. P. Selen, and D. M. Wolpert, Noise in the nervous system, *Nature reviews neuroscience*, *9*(4), 292–303, 2008.

Finch, S., Ornstein-uhlenbeck process, 2004.

Fiser, J., C. Chiu, and M. Weliky, Small modulation of ongoing cortical dynamics by sensory input during natural vision, *Nature*, *431*(7008), 573–578, 2004.

Fiser, J., P. Berkes, G. Orbán, and M. Lengyel, Statistically optimal perception and learning: from behavior to neural representations, *Trends in cognitive sciences*, *14*(3), 119–130, 2010.

Friedmann, S., A new approach to learning in neuromorphic hardware, Ph.D. thesis, Ruprecht-Karls-Universität Heidelberg, 2013.

Fuster, J. M., G. E. Alexander, et al., Neuron activity related to short-term memory, *Science*, *173*(3997), 652–654, 1971.

Geman, S., and D. Geman, Stochastic relaxation, gibbs distributions, and the bayesian restoration of images, *IEEE Transactions on pattern analysis and machine intelligence*, (6), 721–741, 1984.

Gerstner, W., and W. Kistler, *Spiking Neuron Models: Single Neurons, Populations, Plasticity*, Cambridge University Press, 2002.

Gerstner, W., W. Kistler, R. Naud, and L. Paninski, *Neuronal Dynamics*, Cambridge University Press, 2014.

Gibbs, J. W., *Thermodynamics*, vol. 1, Dover Publications, 1961.

Gnadt, J. W., and R. A. Andersen, Memory related motor planning activity in posterior parietal cortex of macaque, *Experimental brain research*, *70*(1), 216–220, 1988.

Gray, H., *Anatomy of the Human Body*, vol. 20th US Edition, Public domain, 1918.

Großkinski, M., Neural sampling with linear feedback shift registers as a source of noise, Bachelor thesis, Ruprecht-Karls-Universität Heidelberg, 2016.

Gütig, R., R. Aharonov, S. Rotter, and H. Sompolinsky, Learning input correlations through nonlinear temporally asymmetric hebbian plasticity, *The Journal of Neuroscience*, *23*(9), 3697–3714, 2003.

Hartel, A., Implementation and characterization of mixed-signal neuromorphic asics, Ph.D. thesis, Universität Heidelberg, 2016.

HBP SP9 partners, *Neuromorphic Platform Specification*, Human Brain Project, 2014.

Helias, M., S. Rotter, M.-O. Gewaltig, and M. Diesmann, Structural plasticity controlled by calcium based correlation detection, *Front. Neuroinform.*, *2*(7), 2008.

Helias, M., S. Kunkel, G. Masumoto, J. Igarashi, J. M. Eppler, S. Ishii, T. Fukai, A. Morrison, and M. Diesmann, Supercomputers ready for use as discovery machines for neuroscience, *Frontiers in Neuroinformatics*, *6*(26), doi:10.3389/fninf.2012.00026, 2012.

Hertäg, L., D. Durstewitz, and N. Brunel, Analytical approximations of the firing rate of an adaptive exponential integrate-and-fire neuron in the presence of synaptic noise, *Frontiers in computational neuroscience*, *8*, 2014.

Heyman, D. P., and M. J. Sobel, *Stochastic models in operations research: stochastic optimization*, vol. 2, Courier Corporation, 2003.

Hinton, G. E., Training products of experts by minimizing contrastive divergence, *Neural computation*, *14*(8), 1771–1800, 2002.

Hinton, G. E., and R. R. Salakhutdinov, Reducing the dimensionality of data with neural networks, *Science*, *313*(5786), 504–507, 2006.

Hô, N., and A. Destexhe, Synaptic background activity enhances the responsiveness of neocortical pyramidal neurons., *J Neurophysiol*, *84*(3), 1488–1496, 2000.

Hopfield, J. J., Neural networks and physical systems with emergent collective computational abilities, *Proceedings of the National Academy of Sciences*, *79*, 2554–2558, 1982.

Horton, J. C., and D. L. Adams, The cortical column: a structure without a function, *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, *360*(1456), 837–862, 2005.

Hubel, D., and T. Wiesel, Receptive fields, binocular interaction and functional architecture in the cat's visual cortex, *J.Physiology*, *160*, 106–154, 1962.

Hubel, D., and T. Wiesel, Receptive fields and functional architecture in two non-striate visual areas (18 and 19) of the cat, *J. Neurophysiology*, *28*, 229–289, 1965.

Ising, E., A contribution to the theory of ferromagnetism, *Z. Phys*, *31*(1), 253–258, 1925.

Jeltsch, S., A scalable workflow for a configurable neuromorphic platform, Ph.D. thesis, Universität Heidelberg, 2014.

Johansson, C., and A. Lansner, Towards cortex sized artificial neural systems., *Neural Networks*, *20*(1), 48–61, 2007.

Johansson, C., M. Rehn, and A. Lansner, Attractor neural networks with patchy connectivity, *Neurocomputing*, *69*(7-9), 627–633, 2006.

Jordan, J., M. Petrovici, O. Breitwieser, J. Schemmel, K. Meier, M. Diesmann, and T. Tetzlaff, Stochastic neural computing without random numbers, 2016.

Jordan, J., et al., Neural networks as sources of uncorrelated noise for functional neural systems, *Tech. rep.*, Computational and Systems Neuroscience, 2014.

Kandel, E. R., J. H. Schwartz, T. M. Jessell, S. A. Siegelbaum, and A. Hudspeth, *Principles of neural science*, vol. 4, McGraw-hill New York, 2000.

Kersten, D., P. Mamassian, and A. Yuille, Object perception as bayesian inference, *Annu. Rev. Psychol.*, *55*, 271–304, 2004.

Koke, C., Personal communication, 2016.

Koke, C., Device variability in synapses of neuromorphic circuits, Ph.D. thesis, Universität Heidelberg, 2017.

Koke, C., and B. Vogginger, Linear feedback shift register hicann specification, `https://brainscales-r.kip.uni-heidelberg.de/projects/tns/wiki/PoissonSources`.

Kolmogorov, A., Über die analytischen methoden in der wahrscheinlichkeitsrechnung, *Mathematische Annalen*, *104*(1), 415–458, 1931.

Korcsak-Gorzo, A., Firing states of recurrent leaky integrate-and-fire networks, bachelor thesis, 2015.

Körding, K., and D. Wolpert, Bayesian integration in sensorimotor learning, *Nature*, *427*(6971), 244–247, 2004.

Kriener, B., H. Enger, T. Tetzlaff, H. E. Plesser, M.-O. Gewaltig, and G. T. Einevoll, Dynamics of self-sustained asynchronous-irregular activity in random networks of spiking neurons with strong synapses, *Frontiers in computational neuroscience*, *8*, 2014.

Krizhevsky, A., and G. Hinton, Learning multiple layers of features from tiny images, 2009.

Kumar, A., S. Schrader, A. Aertsen, and S. Rotter, The high-conductance state of cortical networks, *Neural Computation*, *20*(1), 1–43, 2008a.

Kumar, A., S. Schrader, A. Aertsen, and S. Rotter, The high-conductance state of cortical networks, *Neural computation*, *20*(1), 1–43, 2008b.

Kungl, A., Sampling with leaky integrate-and-fire neurons on the hicannv4 neuromorphic chip, 2016.

Kunkel, S., et al., Spiking network simulation code for petascale computers, *Frontiers in neuroinformatics*, *8*, 78, 2014.

Le Roux, N., and Y. Bengio, Representational power of restricted boltzmann machines and deep belief networks, *Neural Computation*, *20*(6), 1631–1649, 2008.

LeCun, Y., and C. Cortes, The mnist database of handwritten digits, 1998.

LeCun, Y., Y. Bengio, and G. Hinton, Deep learning, *Nature*, *521*(7553), 436–444, doi: http://dx.doi.org/10.1038/nature1453910.1038/nature14539, 2015.

Leng, L., Deep learning architectures for neuromorphic hardware, Master thesis, Ruprecht-Karls-Universität Heidelberg, hD-KIP 14-26, 2014.

Leng, L., M. A. Petrovici, R. Martel, I. Bytschok, O. Breitwieser, J. Schemmel, and K. Meier, Short-term plasticity helps spiking neural networks to become good discriminitive and generative models (working title), *to be published*, 2016.

Lin, I.-C., M. Okun, M. Carandini, and K. D. Harris, The nature of shared cortical variability, *Neuron*, *87*(3), 644–656, 2015.

Lindsay, A., K. Lindsay, and J. Rosenberg, Increased computational accuracy in multicompartmental cable models by a novel approach for precise point process localization, *Journal of computational neuroscience*, *19*(1), 21–38, 2005.

López, J. C., Neurophysiology: Noises on, *Nature Reviews Neuroscience*, *2*(11), 756–756, doi:http://www.nature.com/nrn/journal/v2/n11/full/nrn1101-756b.html, 2001.

Lundqvist, M., M. Rehn, M. Djurfeldt, and A. Lansner, Attractor dynamics in a modular network of neocortex, *Network: Computation in Neural Systems*, *17:3*, 253–276, 2006.

Lundqvist, M., A. Compte, and A. Lansner, Bistable, irregular firing and population oscillations in a modular attractor memory network, *PLoS Comput Biol*, *6*(6), 2010.

Maaten, L. v. d., and G. Hinton, Visualizing data using t-sne, *Journal of Machine Learning Research*, *9*(Nov), 2579–2605, 2008.

Marder, E., and J.-M. Goaillard, Variability, compensation and homeostasis in neuron and network function, *Nature Reviews Neuroscience*, *7*(7), 563–574, 2006.

Markram, H., The blue brain project, *Nature Reviews Neuroscience*, *7*(2), 153–160, 2006.

Markram, H., and M. Tsodyks, Redistribution of synaptic efficacy between neocortical pyramidal neurons, *Nature*, *382*, 1996.

Markram, H., A. Gupta, A. Uziel, Y. Wang, and M. Tsodyks, Information processing with frequency-dependent synaptic connections., *Neurobiol Learn Mem*, *70*(1-2), 101–112, 1998.

Markram, H., et al., Reconstruction and simulation of neocortical microcircuitry, *Cell*, *163*(2), 456 – 492, doi:http://dx.doi.org/10.1016/j.cell.2015.09.029, 2015.

Martel, R., Generative properties of lif-based boltzmann machines, Ph.D. thesis, Master thesis, University of Heidelberg, 2015.

Mead, C. A., *Analog VLSI and Neural Systems*, Addison Wesley, Reading, MA, 1989.

Mead, C. A., Neuromorphic electronic systems, *Proceedings of the IEEE*, *78*, 1629–1636, 1990.

Miller, K. D., Understanding layer 4 of the cortical circuit: a model based on cat v1, *Cerebral cortex*, *13*(1), 73–82, 2003.

Moreno-Bote, R., and N. Parga, Role of synaptic filtering on the firing response of simple model neurons, *Physical Review Letters*, *92*(2), 028,102, 2004.

Moreno-Bote, R., J. Beck, I. Kanitscheider, X. Pitkow, P. Latham, and A. Pouget, Information-limiting correlations, *Nature neuroscience*, *17*(10), 1410–1417, 2014.

Mountcastle, V. B., An organizing principle for cerebral function: The unit module and the distributed system, in *Neuroscience, Fourth Study Program*, edited by F. O. Schmitt, pp. 21–42, MIT Press, Cambridge, MA, 1979.

Nair, V., and G. E. Hinton, Rectified linear units improve restricted boltzmann machines, in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pp. 807–814, 2010.

Naud, R., N. Marcille, C. Clopath, and W. Gerstner, Firing patterns in the adaptive exponential integrate-and-fire model, *Biological Cybernetics*, *99*(4), 335–347, doi:10.1007/s00422-008-0264-7, 2008.

Neftci, E., S. Das, B. Pedroni, K. Kreutz-Delgado, and G. Cauwenberghs, Event-driven contrastive divergence: neural sampling foundations, *Frontiers in neuroscience*, *9*, 2015.

Nessler, B., M. Pfeiffer, L. Buesing, and W. Maass, Bayesian computation emerges in generic cortical microcircuits through spike-timing-dependent plasticity, *PLoS Computational Biology*, *9*(4), e1003,037, 2013.

Oster, M., R. Douglas, and S.-C. Liu, Computation with spikes in a winner-take-all network, *Neural computation*, *21*(9), 2437–2465, 2009.

Pare, D., E. Shink, H. Gaudreau, A. Destexhe, and E. J. Lang, Impact of spontaneous synaptic activity on the resting properties of cat neocortical pyramidal neurons in vivo, *J Neurophysiol*, *79*(3), 1450–60, 1998.

Paul, W., and J. Baschnagel, *Stochastic processes*, Springer, 1999.

Pearl, J., Reverend bayes on inference engines: A distributed hierarchical approach, in *AAAI*, pp. 133–136, 1982.

Pesaran, B., J. S. Pezaris, M. Sahani, P. P. Mitra, and R. A. Andersen, Temporal structure in neuronal activity during working memory in macaque parietal cortex, *Nature neuroscience*, *5*(8), 805–811, 2002.

Petkov, V., Calibration of synaptic drivers on the spikey chip, *Tech. rep.*, 2012a.

Petkov, V., Toward belief propagation on neuromorphic hardware, Diploma thesis, Ruprecht-Karls-Universität Heidelberg, hD-KIP 12-23, 2012b.

Petrovici, M. A., *Form Versus Function: Theory and Models for Neuronal Substrates*, Springer, 2016.

Petrovici, M. A., and J. Bill, A new method for quantifying and predicting neural response correlations: Internal report., 2009.

Petrovici, M. A., J. Bill, I. Bytschok, J. Schemmel, and K. Meier, Stochastic inference with deterministic spiking neurons, *arXiv preprint arXiv:1311.3211*, 2013.

Petrovici, M. A., I. Bytschok, J. Bill, J. Schemmel, and K. Meier, The high-conductance state enables neural sampling in networks of lif neurons, *BMC Neuroscience*, *16*(Suppl 1), O2, 2015a.

Petrovici, M. A., D. Stöckel*, I. Bytschok, J. Bill, T. Pfeil, J. Schemmel, and K. Meier, Fast sampling with neuromorphic hardware, 2015b.

Petrovici, M. A., J. Bill, I. Bytschok, J. Schemmel, and K. Meier, Stochastic inference with spiking neurons in the high-conductance state, *Phys. Rev. E*, *94*, 042,312, doi: 10.1103/PhysRevE.94.042312, 2016.

Petrovici, M. A., et al., Characterization and compensation of network-level anomalies in mixed-signal neuromorphic modeling platforms, *PloS one*, *9*(10), e108,590, 2014.

Pfeil, T., Configuration strategies for neurons and synaptic learning in large-scale neuromorphic hardware systems, Diploma thesis, Ruprecht-Karls-Universität Heidelberg, hD-KIP-11-34, 2011.

Pfeil, T., et al., Six networks on a universal neuromorphic computing substrate, *Frontiers in Neuroscience*, *7*, 11, doi:10.3389/fnins.2013.00011, 2013.

Pfister, J.-P., T. Toyoizumi, D. Barber, and W. Gerstner, Optimal Spike-Timing-Dependent Plasticity for Precise Action Potential Firing in Supervised Learning, *Neural Comput.*, *18*(6), 1318–1348, 2006.

Plesser, H. E., and S. Tanaka, Stochastic resonance in a model neuron with reset, *Physics Letters A*, *225*(4), 228–234, 1997.

Potjans, T. C., and M. Diesmann, The cell-type specific cortical microcircuit: Relating structure and activity in a full-scale spiking network modela, *Cereb. Cortex*, *24*, 785–806, doi:10.1093/cercor/bhs358, 2012.

Pouget, A., J. M. Beck, W. J. Ma, and P. E. Latham, Probabilistic brains: knowns and unknowns, *Nature Neuroscience*, *16*(9), 1170–1178, 2013.

Qu, L., Y. Akbergenova, Y. Hu, and T. Schikorski, Synapse-to-synapse variation in mean synaptic vesicle size and its relationship with synaptic morphology and function, *Journal of Comparative Neurology*, *514*(4), 343–352, 2009.

Reyes, A. D., Synchrony-dependent propagation of firing rate in iteratively constructed networks in vitro, *Nature neuroscience*, *6*(6), 593–599, 2003.

Ricciardi, L. M., and L. Sacerdote, The ornstein-uhlenbeck process as a model for neuronal activity, *Biological Cybernetics*, *35*, 1–9, 1979.

Ricciardi, L. M., and S. Sato, First-passage-time density and moments of the ornstein-uhlenbeck process, *Journal of Applied Probability*, *25*, 43–57, 1988.

Rivkin, B., On the memory characteristic of a cortical atractor network, Bachelor thesis, Ruprecht-Karls-Universität Heidelberg, 2014.

Rolls, E. T., and G. Deco, *The noisy brain: stochastic dynamics as a principle of brain function*, vol. 34, Oxford university press Oxford, 2010.

Rosenbaum, R., T. Tchumatchenko, and R. Moreno-Bote, Correlated neuronal activity and its relationship to coding, dynamics and network architecture, *Frontiers in computational neuroscience*, *8*, 2014.

Roudi, Y., and P. E. Latham, A balanced memory network, *PLoS Comput Biol*, *3*(9), e141, 2007.

Salakhutdinov, R., Learning deep boltzmann machines using adaptive mcmc, in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pp. 943–950, 2010.

Sasaki, T., N. Matsuki, and Y. Ikegaya, Action-potential modulation during axonal conduction, *Science*, *331*(6017), 599–601, 2011.

Schemmel, J., A. Grübl, K. Meier, and E. Muller, Implementing synaptic plasticity in a VLSI spiking neural network model, in *Proceedings of the 2006 International Joint Conference on Neural Networks (IJCNN)*, IEEE Press, 2006.

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

Schmidt, D., Automated characterization of a wafer-scale neuromorphic hardware system, Master thesis, Ruprecht-Karls-Universität Heidelberg, 2014.

Schmuker, M., T. Pfeil, and M. P. Nawrot, A neuromorphic network for generic multivariate data classification, *Proceedings of the National Academy of Sciences*, *111*(6), 2081–2086, 2014.

Schottky, W., Über spontane stromschwankungen in verschiedenen elektrizitätsleitern, *Annalen der physik*, *362*(23), 541–567, 1918.

Schwartz, M.-O., *PhD thesis*, University of Heidelberg, in preparation, 2012.

Shafi, M., Y. Zhou, J. Quintana, C. Chow, J. Fuster, and M. Bodner, Variability in neuronal activity in primate cortex during working memory tasks, *Neuroscience*, *146*(3), 1082 – 1108, doi:10.1016/j.neuroscience.2006.12.072, 2007.

Shipp, S., Structure and function of the cerebral cortex, *Current Biology*, *17*(12), R443–R449, 2007.

Shu, Y., A. Hasenstaub, M. Badoual, T. Bal, and D. A. McCormick, Barrages of synaptic activity control the gain and sensitivity of cortical neurons, *Journal of Neuroscience*, *23*(32), 10,388–10,401, 2003.

Silver, D., et al., Mastering the game of go with deep neural networks and tree search, *Nature*, *529*(7587), 484–489, 2016.

Smolensky, P., Information processing in dynamical systems: Foundations of harmony theory, *Tech. rep.*, DTIC Document, 1986.

Song, S., and L. F. Abbott, Cortical development and remapping through spike timing-dependent plasticity, *Neuron*, *32*(2), 339–350, 2001.

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

Stevens, C. F., and Y. Wang, Facilitation and depression at single central synapses, *Neuron*, *14*(4), 795–802, 1995.

Stöckel, D., Boltzmann sampling with neuromorphic hardware, Bachelor thesis, Ruprecht-Karls-Universität Heidelberg, 2015.

Stöckel, D., Boltzmann sampling with neuromorphic hardware, 2015.

Sutskever, I., G. E. Hinton, and G. W. Taylor, The recurrent temporal restricted boltzmann machine, in *Advances in Neural Information Processing Systems*, pp. 1601–1608, 2009.

Tchumatchenko, T., T. Geisel, M. Volgushev, and F. Wolf, Spike correlations–what can they tell about synchrony?, *Frontiers in neuroscience*, *5*, 68, 2011.

Tieleman, T., Training restricted boltzmann machines using approximations to the likelihood gradient, in *Proceedings of the 25th international conference on Machine learning*, pp. 1064–1071, ACM, 2008.

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

Tsodyks, M. V., and T. Sejnowski, Rapid state switching in balanced cortical network models, *Network: Computation in Neural Systems*, *6*(2), 111–124, 1995.

van Albada, S. J., M. Helias, and M. Diesmann, Scalability of asynchronous networks is limited by one-to-one mapping between effective connectivity and correlations, *PLoS Comput Biol*, *11*(9), e1004,490, 2015.

van Vreeswijk, C., and H. Sompolinsky, Chaos in neuronal networks with balanced excitatory and inhibitory activity, *Science*, *274*(5293), 1724–6, 1996.

Villiere, V., and E. M. McLachlan, Electrophysiological properties of neurons in intact rat dorsal root ganglia classified by conduction velocity and action potential duration, *Journal of Neurophysiology*, *76*(3), 1924–1941, 1996.

Weilbach, C., An online learning algorithm for lif-based boltzmann machines, Bachelor thesis, Ruprecht-Karls-Universität Heidelberg, 2015.

Yang, H., and M. A. Xu-Friedman, Stochastic properties of neurotransmitter release expand the dynamic range of synapses, *The Journal of Neuroscience*, *33*(36), 14,406–14,416, 2013.

Yang, T., and M. N. Shadlen, Probabilistic reasoning by neurons, *Nature*, *447*(7148), 1075–1080, 2007.

Yang, Y., and A. Raine, Prefrontal structural and functional brain imaging findings in antisocial, violent, and psychopathic individuals: a meta-analysis, *Psychiatry Research: Neuroimaging*, *174*(2), 81–88, 2009.

Acknowledgments

The work reported was funded by the Manfred Stärk foundation.

I would like to thank everybody who supported this work:

Karlheinz Meier I am grateful for the opportunity to work in this exceptional group. I have learned a lot.

Manfred Stärk Ich danke Ihnen von ganzem Herzen dafür, dass Sie durch Ihre Finanzierung diese Arbeit ermöglicht haben. Es war immer eine Freude, sich mit Ihnen über unsere Arbeit und Forschung allgemein zu unterhalten. Wie Sie bereits sagten: "Forschung ist nie zu Ende." Ich bin trotzdem froh, mit der Fertigstellung dieser Arbeit viele Sachen zum Abschluss gebracht zu haben. Ich hoffe, dass Sie auch weiterhin viel Freude an unseren neuromorphen Forschungsprojekten haben werden!

Johannes Schemmel It is always reassuring to count on supportive and encouraging words from the VISIONS head. I thank Johannes for many insightful comments on modelling and biology.

Daniel Durstewitz, Ulrich Schwarz, Michael Hausmann Thank you for agreeing to review this thesis and showing so much interest for the work done in our group.

Mihai A. Petrovici Thanks for throwing in all-nighters to proofread (or rather paint red) my diploma thesis and for proofreading this one under more humane circumstances. Also, thanks for "I know what you mean" during discussions and for all the helpful advice during our research projects. Thanks for helping me and many other people in this group by investing inhuman working hours to get things done. Speaking of challenges: thanks for the film debates, the badminton rage, the politics rage, road rage, cake bet (rage?) and GSP. Lastly, thanks for readily playing the devils advocate at any possible opportunity, you're a lot better at it than Keanu!

Dominik Dold Thanks for your dedication and great ideas on building noiseless networks and... video games! You truly have a grossly incandescent taste in video games, I must say. It always causes a nostalgy boost when discussing the most relevant topic of all: which FF installment was the best one? (It's the seventh.)

Boris Rivkin We had intriguing discussions on cortical coding, spurious attractors and a productive time during the L23 network stuff. Also, now I know how to really hear my voice.

David Stöckel Thank you for harnessing the power of Spikey and generating immortal results with only four random variables!

Luziwei Leng Thank you for giving me a hard time & mocking me during our table tennis matches and for the fine tea! Also, we've had plenty of awkward (therefore enjoyable) conversations in the Mensa, which I always appreciate.

Agnes Korcsak-Gorzo Thank you for your patience to go through so many topic changes and still staying in our group. Also, Federer!

Johannes Bill Thanks for some great football matches in CapoCaccia 2013 and bringing even more Eintracht into our office. Your advice on scientific writing was greatly appreciated.

Dimitri Probst I wish we had more opportunities to play football together. Thanks for all the meat from the Schwenker during the World Cup 2014 opening ceremony!

Paul Müller As a fellow Stärkian, I thank you for your support on hardware knowledge during the BrainScaleS Demo projects. Also, thanks for your monumental amount of work hours to make the BSS Demos happen at the end of the project phase. Lastly, thank you for proofreading my thesis!

Mitja Kleider Thanks for proofreading my thesis and providing our office with all necessary supplies. Our alliance against Christoph to enforce a darkening of the office works every time. At some point I am going to break this alliance and steal all your startup ideas.

Christoph Koke Seperated by an impenetrable calibration-modeling barrier, we had only little conversation until I moved into 01.312. Sharing the pain of writing the thesis, I think I became more productive. This actually means that the panic monster took over and there was no more instant gratification. Also thanks for proofreading my thesis and for enduring all the procastrinating conversations during my writing time.

Eric Müller Thank you for dealing with my obnoxious tech problems ("verranzt") without making too much fun of me. I admire your impeccable timing at bringing up random topics with slightly political bias at random times. You never know when Eric starts talking about Vladimir!

Sebastian Schmitt Thanks for proofreading that chapter in record time, and in

general for doing great stuff with the wafer. Your Peter Parker reference is one of the legendary lines from the BrainScaleS plenaries!

Andreas Hartel Aside of trying to break the enemy's defense in Badminton, I always enjoy our casual encounters in the KIP. Beyond that, the drive back from Frankfurt Hahn with you and Christian after the Vilnius workshop was a great experience. Also, thanks for proofreading that nasty chapter from my thesis. Your input was extremely helpful to improve it.

Oliver Breitwieser Thank you for the great software and your lifetime of support on technicals that I have regularly annoyed you with. Also the unforgettable Fürberg sauna experience, hah!

Vitali Karasenko Thank you for playing Maxwell's demon in our office. Congratulations, you postponed the heat death of our universe by 2048 years!

Tom Pfeil Thank you for patiently explaining and troubleshooting Spikey every time I had issues. You were always helpful when I was dealing with unforeseen, hardware-related consequences. Of course I am grateful for the memorable Badminton matches with Wei, Andi and Mihai.

Christian Weilbach Thanks for the music, the coffee and all the Lenin.

Christian Pehle Thanks for all the enjoyable conversations about pretty much anything and a good time in Vilnius as roommates.

Björn Kindler Plain and simple: thanks for being the best organizer one could imagine.

Andreas Grübl Thanks for taking care of so many hardware-related things and always being helpful when I had problems with Spikey.

Kai Husmann After several years of enduring that old `kataklysm` machine, I was incredibly happy for an upgrade to something that can display pdf documents effortlessly. Then I knew I was ready to write this thesis. Thanks for that!

Dan Husmann & Maurice Güttler Thanks for taking time to look for hardware figures!

Syed Ahmed Aamir Well, thanks for accidentally finding my "lost" keyring. I have no idea what I would have done if you hadn't "found" it.

Venelin Petkov Thank you for showing me the ropes to survive here when I still was fresh: proxies, ssh configs, git, emacs and general Unix knowledge. Also thanks for not

telling anyone that I didn't already know by myself!

Sven Schrader Our discussion about music during the drive to Fürberg in 2012 was one of the reasons I had finally picked up an instrument. Sure, sooner or later I would have done it anyway, but still!

Meine Eltern & Familie Danke fur die jahrzehntelange Unterstützung bei allem, was ich getan habe und insbesondere dafür, dass ich immer alles auf meine Art machen konnte.

Sergej Danke für die Wetten, die Spiele und die etlichen Runden in CSGO, die immer viel zu früh vorbei waren...

For knowing what matters, for the guanakos, the guitars, the waves and for being the last. Thank you, **Luana**.

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, November 14, 2016

.....
(signature)