Heidelberg University
Department of Physics and Astronomy

Master's Thesis

in Physics

submitted by

Dominik Dold

born in Titisee-Neustadt, Germany

September 2016

Stochastic Computation in Spiking Neural Networks Without Noise

This Master's Thesis has been carried out by Dominik Dold at the

Kirchhoff-Institute for Physics

Ruprecht-Karls-Universität Heidelberg

under the supervision of

Prof. Dr. Karlheinz Meier

Stochastic Computation in Spiking Neural Networks Without Noise

Nowadays, it is generally assumed that cortical neurons perform stochastic computations. In computational models of cortical neural networks, stochasticity is often implemented by feeding every neuron with Poisson noise. However, this approach runs into difficulties as soon as we consider physical implementations of networks, e.g. on neuromorphic hardware. Such systems always have limited bandwidth for external input, thus rendering a scalable implementation of stochastic computation that relies on external noise unfeasible.

In the cerebral cortex, one major source of stochasticity is the ongoing background activity of the cortex itself. Inspired by this observation, we implement a novel approach for providing functional networks with noise. For this purpose, so-called Boltzmann machines consisting of Leaky Integrate-and-Fire neurons are utilized to construct a small and sparsely connected network of functional units. The key idea is that every Boltzmann machine in this network receives noise from other Boltzmann machines, which generate noise as a byproduct of their functional task.

By combining computational principles inspired from biology as well as methods coming from modern machine learning, we demonstrate that it is indeed possible to implement such networks of Boltzmann machines that function reliably without any external noise input.

Stochastisches Rechnen in Spikenden Neuronalen Netzen Ohne Rauschen

Heutzutage wird allgemein angenommen, dass Neuronen im Kortex stochastische Methoden zur Informationsverarbeitung verwenden. Um dieses stochastische Verhalten in numerischen Modellen kortikaler Netzwerke zu berücksichtigen, speist man diese meist mit Poisson-Rauschen. Dies führt jedoch zu Problemen bei physikalischen Umsetzungen solcher Netzwerke, z.B. auf neuromorpher Hardware. Hierbei ist die für externe Signale verfügbare Bandbreite immer beschränkt, wodurch eine skalierbare Implementierung stochastischer Rechenmethoden, welche externes Rauschen benötigen, nicht möglich ist.

Im Kortex stellt die permanente Hintergrundaktivität des Kortex selbst eine maßgebliche Rauschquelle dar. Hiervon inspiriert untersuchen wir eine neue Methode um funktionale Netzwerke mit Rauschen zu versorgen. Dabei verwenden wir sogenannte Boltzmann-Maschinen, bestehend aus Leaky Integrate-and-Fire Neuronen, welche zu kleinen, spärlichen Netzwerken verbunden werden. Die Hauptidee hierbei ist, dass jede Boltzmann-Maschine durch andere sich im Netzwerk befindliche Boltzmann-Maschinen mit Rauschen versorgt wird, deren Aktivität ein Resultat der Bewältigung funktionaler Aufgaben ist.

Durch Kombinieren von Prinzipien und Konzepten aus Biologie und Maschinellem Lernen demonstrieren wir, dass es in der Tat möglich ist solche Netzwerke aus Boltzmann-Maschinen zu implementieren, welche ohne externe Rauschquellen funktionale Aufgaben lösen können.

Contents

It is an important and popular fact that things are not always what they seem. For instance, on the planet Earth, man had always assumed that he was more intelligent than dolphins because he had achieved so much – the wheel, New York, wars and so on – whilst all the dolphins had ever done was muck about in the water having a good time. But conversely, the dolphins had always believed that they were far more intelligent than man – for precisely the same reasons.

. . . In fact there was only one species on the planet more intelligent than dolphins, and they spent a lot of their time in behavioural research laboratories running round inside wheels and conducting frighteningly elegant and subtle experiments on man. The fact that once again man completely misinterpreted this relationship was entirely according to these creatures' plans.

Douglas Adams, The Hitchhiker's Guide to the Galaxy

1 Introduction

The fundamental building blocks of information processing in the mammalian brain are believed to be single, vastly interconnected cells called *neurons*. These transmit information via short all-or-nothing events called *action potentials* or *spikes*. However, spike trains recorded from single cortical neurons have been observed to exhibit strong *trial-to-trial variability* in vivo when repeatedly exposed to identical visual stimuli (*Henry et al.*, 1973; *Schiller et al.*, 1976; *Snowden et al.*, 1992; *Vogels et al.*, 1989; *Holt et al.*, 1996; *Azouz and Gray*, 1999; *Fiser et al.*, 2004). Contrary, in vitro measurements showed that cortical neurons deterministically transform time-dependent synaptic input currents into output spike trains (*Mainen and Sejnowski*, 1995). Thus, the variability observed in vivo can mainly be attributed to the ongoing cortical background activity (*Arieli et al.*, 1996) which is absent in vitro, but whether the seemingly stochastic and irregular behavior of cortical neurons plays any role in neural coding is still an open debate.

If we assume that noise serves no functional purpose, the neural code used by cortical neurons has to be able to cope with the observed variations. One possibility are *rate* or *population codes*, which disregard individual spiking times and only take into account the average frequency of either single neurons or populations thereof (*Paradiso*, 1988; *Vogels*, 1990; *Shadlen and Newsome*, 1994; *Lee et al.*, 1998). Another coding scheme are *temporal codes*, i.e., information is rather encoded in the exact spike times of each neuron instead of averaged rates. Of course, this way of encoding information is highly susceptible to noise. However, it is believed that the underlying noise might enable cortical neurons to perform stochastic computations (*Hoyer and Hyvarinen*, 2003; *Körding and Wolpert*, 2004; *Rolls and Deco*, 2010; *Roumani and Moutoussis*, 2012), turning neuronal noise into a crucial and even necessary part of information processing in the cortical area of the brain.

Recently, it was shown that the spike-based neural sampling framework with abstract neurons introduced by *Buesing et al.* (2011) can be mapped to a network of *Leaky Integrate-and-Fire* neurons (*Petrovici et al.*, 2013, 2015a; *Petrovici*, 2016), enabling sampling from Boltzmann distributions over binary variables with a population of neurons using biologically inspired mathematical models. Promising applications are fast and energy-efficient implementations of *Boltzmann machines* (*Ackley et al.*, 1985) on state-of-the-art accelerated neuromorphic hardware (*Stöckel*, 2015; *Petrovici et al.*, 2015b; *Kungl*, 2016) as well as software implementations of large *restricted Boltzmann machines* (*Salakhutdinov et al.*, 2007) that are trained on real-world problems, for example image classification and generation of the MNIST handwritten digit database (*Leng*, 2014; *Martel*, 2015; *Leng et al.*, 2016).

Since Leaky Integrate-and-Fire neurons are inherently deterministic, every neuron is commonly fed with high-frequency excitatory and inhibitory *Poisson noise* to realize stochastic dynamics. This is also the usual method of reproducing the stochastic behavior of neurons found in the cerebral cortex (*Gerstner and Kistler*, 2002). In software simulations, even though adding external Poisson noise increases the total simulation time, the number of neurons of a finite-sized network that can be provided with independent Poisson noise is otherwise not limited. This changes if we, for instance, consider *very-large-scale integration* (VLSI) of neural networks in silicon. In such devices, the network is commonly restricted to a certain volume and can only be accessed or fed with external inputs via its surrounding surface area, i.e., the available interfaces or buses attached to the device. This strictly limits the maximum bandwidth of external input that can be provided to the network (see Fig. 1.1). Therefore, unlike in software simulations, the amount of available noise is limited, making it a valuable resource that has to be managed efficiently.

Prominent examples of such state-of-the-art neuromorphic hardware are the *Spikey* neuromorphic chip (*Schemmel et al.*, 2006; *Pfeil et al.*, 2013) and the *HICANN* (High Input Count Analog Neural Network) wafer system (*Schemmel et al.*, 2008, 2010), both developed at Heidelberg University. The HICANN chip is equipped with a high-bandwidth on-chip and on-wafer bus infrastructure for inter-neuron communication. However, these buses cannot be directly connected to external components. Instead, an additional interface, with limited bandwidth due to the high integration density on-chip, is used for off-chip and off-wafer communication (for details, see App. 8.4). In fact, in a recent implementation of small Boltzmann machines on a single HICANNv4 chip, the external bandwidth was only able to support a 2-neuron Boltzmann machine (*Kungl*, 2016).

Currently, there exist two approaches to deal with bandwidth limitations:

- The available pool of noise inputs may simply be distributed among the neurons used in the experiment, allowing each input to connect to several on-chip neurons instead of just one. To reach adequate noise frequencies, some neurons will therefore receive similar inputs, leading to shared-input correlations (*Bytschok*, 2011). Even though these additional correlations have a negative effect on the sampling quality of Boltzmann machines, it was recently demonstrated by I. Bytschok and M. Petrovici (personal communication) that this can be compensated by adjusting the weights and biases of Boltzmann machines appropriately, for example via training.

- Instead of using Poisson sources, the spike trains of neurons from a balanced recurrent network with strong inhibitory feedback can be used (*Jordan*, 2013; *Jordan et al.*, 2015), see Fig. 1.1. Such recurrent networks have the advantage that shared-input correlations get compensated by negative correlations between neurons of the network, caused by strong inhibitory feedback. This way, a small pool of neurons forming a recurrent network (colloquially dubbed a "sea of noise") can be used to feed a much larger functional network with noise.

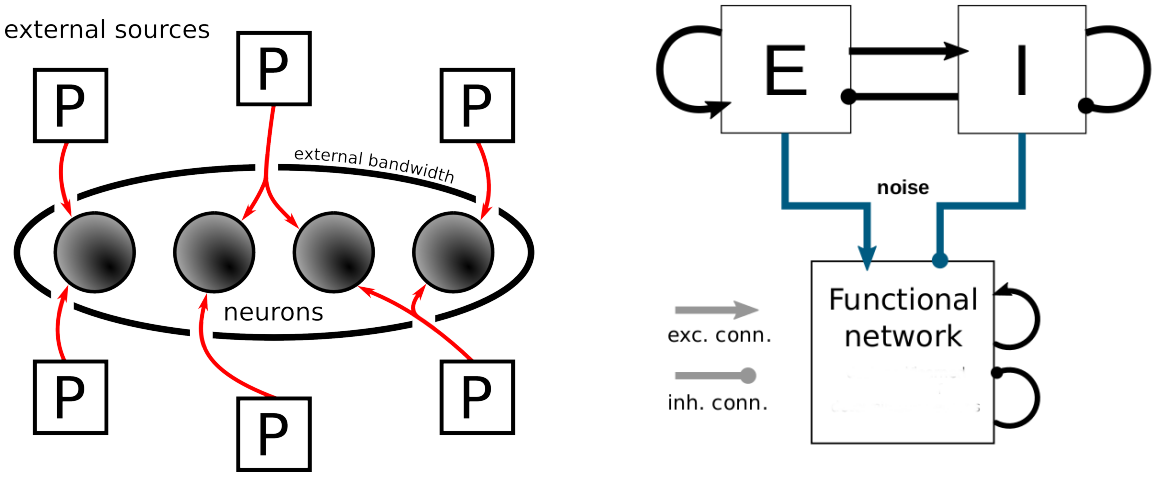

Figure 1.1: **(left)** The number of independent external sources available to a pool of neurons (circles) is limited by the external bandwidth (ellipse). Therefore, some neurons have to share Poisson inputs (P), introducing shared-input correlations. **(right)** A recurrent neural network with an excitatory (E) and inhibitory (I) population of neurons can be used as an effective noise source for a functional network. Image taken from *Jordan et al.* (2015).

In this thesis, we will explore an alternative approach of providing noise to functional neural networks that does not utilize sources which are specialized on generating noise, as is the case in the two previously presented methods. Our approach is inspired by the mammalian brain, where cortical neurons exhibit seemingly noisy behavior, but are not subject to any external Poisson sources. In fact, the brain most probably has no functional areas that are specialized on generating noise input nor does it receive noise from any external sources (*Gerstner and Kistler*, 2002). However, every cortical neuron constantly receives input from over 10^4 presynaptic partners and is thus exposed to a high-frequency bombardment of spikes originating from the ongoing background activity of the cortical network alone (*Arieli et al.*, 1996; *Fourcaud and Brunel*, 2002). Consequently, it appears plausible that different functional components of the brain supply each other with noise. The question then arises whether a similar setup can be realized with Leaky Integrate-and-Fire Boltzmann machines. In fact, it was recently shown in *Korcsák-Gorzó* (2015) that sparse and inhibition-dominated Boltzmann machines produce uncorrelated and Poisson-like spike trains that might be used as stochastic input for other functional networks. This is a promising result and supports the idea that spikes from Boltzmann machines could be used as noise for other Boltzmann machines.

The goal of this thesis is to show that a network of Boltzmann machines can be used to reduce the number of needed Poisson sources by compensating the missing stochastic input with the intrinsic background activity of the network. Broadly speaking, every Boltzmann machine of the network uses the spike trains of neighbouring Boltzmann

machines as noise input (see Fig. 1.2). This would mimic the continuous bombardment observed in the cerebral cortex and allow functional networks to perform stochastic computations by utilizing the ongoing background activity of other functional networks. Moreover, returning to our initial bandwidth problem on neuromorphic hardware, such a "sea of Boltzmann machines" would allow us to exploit the high-bandwidth infrastructure available for on-chip and on-wafer communication on HICANN, circumventing the bottleneck for external stimuli.

This thesis is structured as follows: First, a short introduction of the necessary theoretical background will be given. Afterwards, we will study whether spike trains generated by arbitrary Boltzmann machines can be used to replace Poisson sources. Finally, networks of Boltzmann machines will be implemented to study how slowly reducing the strength, frequency or number of external inputs affects the sampling quality. As a consequence of these studies and a final result, we will indeed be able to show that it is possible to build networks of Boltzmann machines without any external input that are able to sample from their target Boltzmann distributions almost as well as if each neuron was receiving noise from independent Poisson sources.

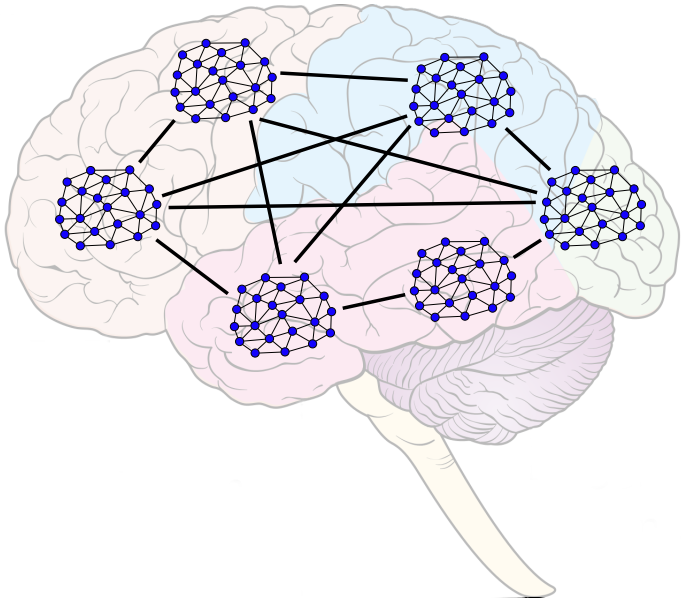

Figure 1.2: Illustration of the "sea of Boltzmann machines", which consists of many interconnected Boltzmann machines (connected blue dots). Inspired by the cerebral cortex, stochasticity is provided by the intrinsic noise or background activity of the surrounding network instead of feeding every neuron with external Poisson spikes. In our case, every Boltzmann machine of the network receives spikes from the surrounding Boltzmann machines as noise. Image modified from *Wikimedia Commons* (2014).

2 Theoretical Background

In the following sections, the theoretical requirements needed for this thesis are outlined.

First, a brief overview on neuron models, especially the Leaky Integrate-and-Fire (LIF) neuron model will be given. For a more in-depth discussion of neuron and synapse dynamics, see *Gerstner and Kistler* (2002); *Bytschok* (2011); *Petrovici* (2016).

Afterwards, LIF sampling as a computational method in spike-based networks and its application to state-of-the-art machine learning problems will be highlighted. Again, for further details see *Leng* (2014); *Martel* (2015); *Breitwieser* (2015); *Petrovici* (2016).

2.1 Leaky Integrate-and-Fire (LIF) Neuron Model

A central principle of modern neuroscience is the so-called *neuron doctrine*, introduced by Santiago Ramón y Cajal and shortly after named by Wilhelm von Waldeyer in 1891 (*Waldeyer*, 1891; *Guillery*, 2005; *De Carlos and Borrell*, 2007). It states that the elemental building blocks or atomic units of information processing in the brain are single cells called *neurons*. Interconnected through *synapses*, they build a complex network consisting of approximately 86 billion neurons in the human brain (*Azevedo et al.*, 2009).

In a simplified view, a biological neuron consists of three distinct parts (see Fig. 2.1):

- the *dendritic tree*, which passes on incoming signals from neighbouring neurons,
- the *soma*, where the accumulation of these incoming signals may trigger an outgoing signal,
- the *axon*, which transmits outgoing signals to other neurons in the network.

Instead of using continuous-time signals, neurons interact via discrete "all-or-nothing" events called *Action Potentials* (AP) or *spikes*. An AP is characterized by an abrupt depolarization of the membrane potential of the neuron, followed by a phase of strong hyperpolarization (called refractory period) during which it cannot be depolarized again. The first detailed explanation of APs was given by Alan Lloyd Hodgkin and Andrew Huxley in 1952 (*Hodgkin and Huxley*, 1952) based on observations in the squid giant axon (see Fig. 2.2).

Even though neurons have already been proposed in 1891 as the fundamental unit of information processing, it remains unclear how information is actually encoded and

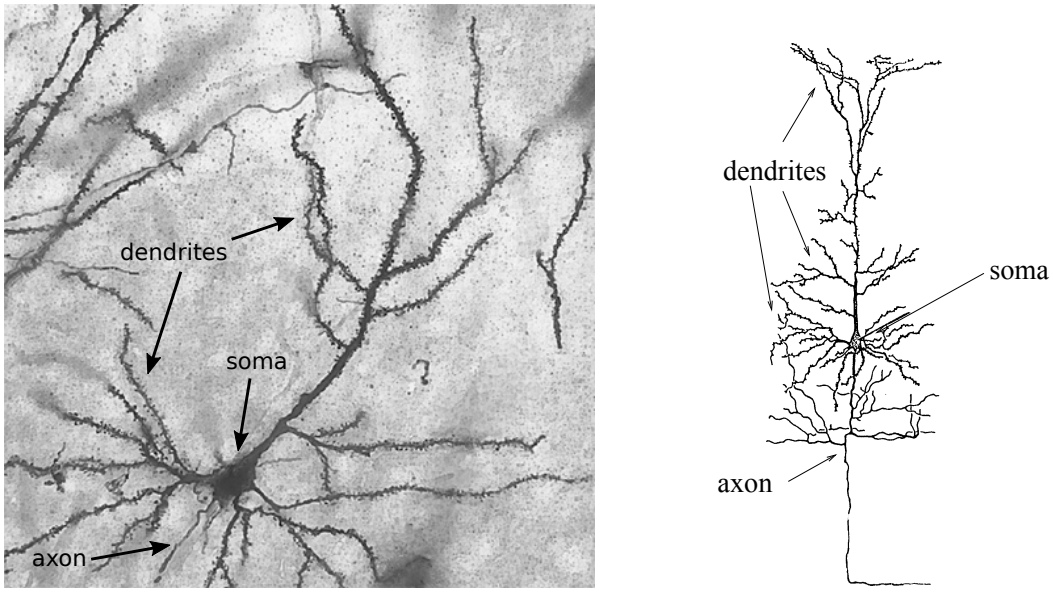

Figure 2.1: **(left)** Pyramidal neuron of the rat prefrontal cortex stained with the Golgi-Cox method. Image modified from *Perez-Costas et al.* (2007). **(right)** Drawing of a single neuron by Cajal. It consists of three major parts: (i) The soma containing the cell nucleus, (ii) dendrites which extend from the soma and collect input from other neurons and (iii) the outgoing axon which transmits signals to other neurons. Image modified from *Gerstner and Kistler* (2002).

processed in the human brain (*Gerstner and Kistler*, 2002). One possibility to improve our understanding of neuronal coding is to complement biological studies with software simulations and hardware emulations of neural networks. However, since realistic biological systems are rather complex and contain a multitude of dynamic variables, we have to use simpler models that encompass the most fundamental properties of neuronal dynamics. One such model is the *Leaky Integrate-and-Fire* (LIF) neuron model (*Lapique*, 1907), which is described by a simple first-order ordinary differential equation (ODE) for the membrane potential of the neuron and a threshold condition for triggering spikes:

$$c_\mathrm{m}\frac{\mathrm{d}u}{\mathrm{d}t} = g_\mathrm{L}\left(E_\mathrm{L} - u(t)\right) + I_\mathrm{syn}\,, \tag{2.1}$$

$$u(t_s) \geq \vartheta \;\;\Rightarrow\;\; u(t \in [t_s, t_s + \tau_\mathrm{ref}]) = u_\mathrm{reset}\,. \tag{2.2}$$

Similarly to the Hodgkin-Huxley model (see Fig. 2.2), the cell membrane of the neuron is modelled as a capacitor with capacitance c_m connected to a resistor R with conductance $g_\mathrm{L} = R^{-1}$. The effects of incoming spikes from neighbouring neurons are represented as charge fluxes I_syn onto the capacitor. The resulting change in membrane potential due to a presynaptic spike is called a *postsynaptic potential* (PSP). Note that throughout this

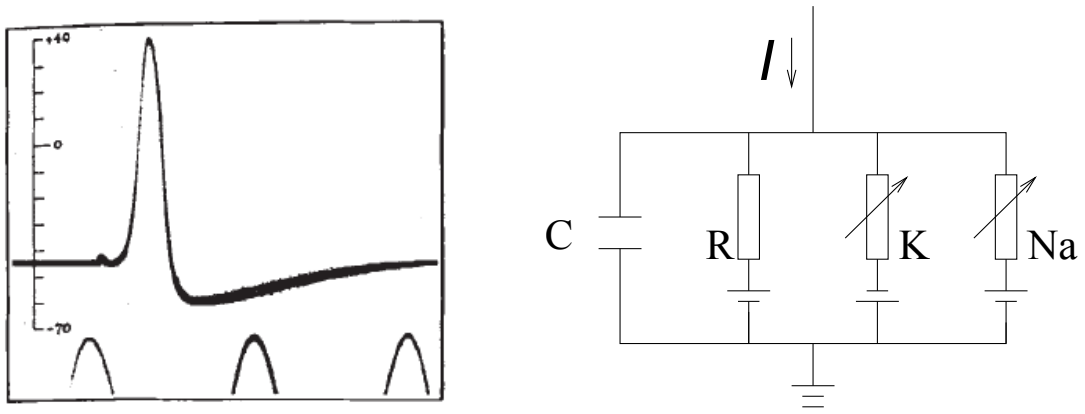

Figure 2.2: **(left)** Recording of an AP of a squid giant axon by *Hodgkin and Huxley* (1939). Note the strong overshoot followed by a phase of hyperpolarization. The y-scale is given in mV, the time scale in terms of cycles (bottom) with 2ms per cycle. **(right)** Abstract model used by Hodgkin and Huxley to describe the creation of APs. The cell membrane of the neuron is modelled by a capacitor C that can leak charges through a conductance R^{-1}. In addition, there are two channels for K^+ and Na^+ ions with voltage-dependent conductance. A strong step-current I leads to an opening of the two ion gates on different time scales. For instance, the strong overshoot is produced by an initial inflow of Na^+ into the cell. The hyperpolarization originates from a delayed inactivation of the Na^+ gates and opening of the K^+ gate leading to a strong outflow of K^+ ions. Image taken from *Gerstner and Kistler* (2002).

thesis, neurons will be treated as being point-like, i.e., possible effects due to dendritic transmission of PSPs will be neglected. If the membrane potential reaches the threshold ϑ, it emits a spike and becomes refractory for a time period τ_{ref}. During this time, the neuron is not able to emit another spike. This is implemented by clamping the membrane potential to the reset potential u_{reset}, mimicking the period of hyperpolarization observed in biology. Further note that the characteristic shape of APs is reduced to the spike time t_s, which corresponds to the rising flank of the AP, and the clamping mechanism. Therefore, we completely neglect the shape of the AP. This is a valid simplification as it is often generally assumed that the shape of all APs is identical and contains no information (*Gerstner and Kistler*, 2002; *Dayan and Abbott*, 2001).

The dynamics of a LIF neuron is governed by two terms: A drift term that leads to an exponential decay towards the *leak potential* E_{L} and the synaptic input I_{syn}. The latter can be described in two ways:

- *Current-Based Synapses* (CUBA): Each incoming spike leads to an influx of charge, i.e., a direct increase or decrease of the membrane potential. These PSPs add up linearly.

- *Conductance-Based Synapses* (COBA): Incoming spikes increase the conductance of the dendritic membrane, leading also to a temporary increase in charge flux. In contrast to the CUBA model, they induce a nonlinear interaction of PSPs.

The CUBA model neglects more complex dynamics at the *synaptic cleft* (the location where *presynaptic* - and *postsynaptic terminal* meet, see Fig. 2.3) and describes the resulting charge flux into the soma. Contrary, the COBA model is directly inspired from *chemical synapses*, where incoming spikes lead to an opening of ion gates at the postsynaptic terminal, i.e., the membrane conductance of the postsynaptic cell is locally increased. This increase in conductance leads to a passive flow of charges through the membrane.

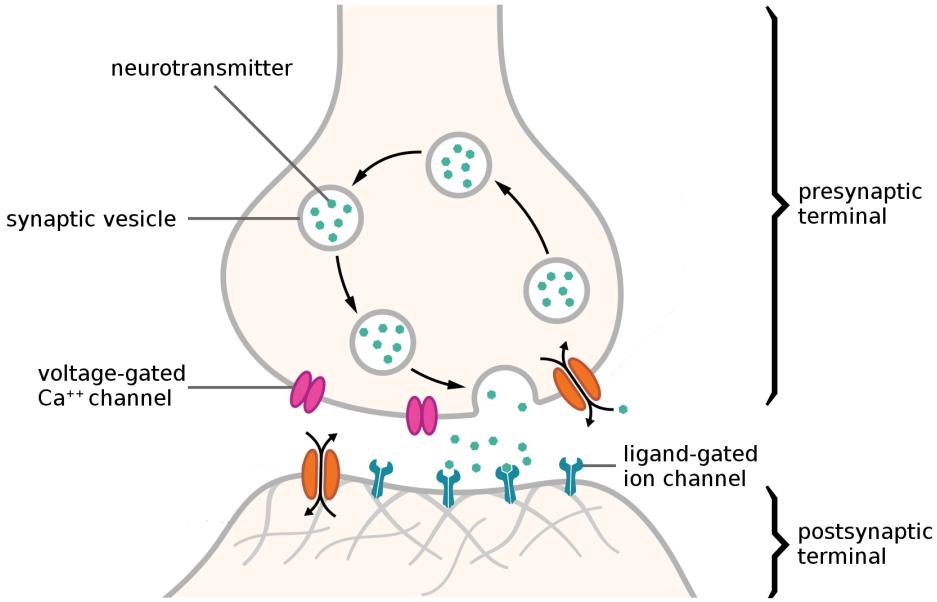

Figure 2.3: Schematic representation of a chemical synapse. A spike from the presynaptic neuron opens the voltage-gated calcium channels at the presynaptic terminal. The inflowing Ca^{++} ions bind to the synaptic vesicles containing neurotransmitters and let them fuse with the cell membrane. This way, the neurotransmitters are released into the synaptic cleft and diffuse towards the receptors of the ligand-gated ion channels of the neighbouring dendritic membrane. As soon as they dock onto the receptors, the ion channels are opened, locally increasing the conductance of the postsynaptic cell. Image modified from *Wikimedia Commons* (2015).

In the LIF model, the synaptic term is implemented as follows:

$$I_{\text{syn}}^{\text{CUBA}} = \sum_{\text{syn. k}} \sum_{\text{spikes s}} w_k \epsilon(t - t_s)\,, \tag{2.3}$$

$$I_{\text{syn}}^{\text{COBA}} = \sum_{x\in\{\text{e,i}\}} \sum_{\text{syn. k}} \sum_{\text{spikes s}} w_k \epsilon(t - t_s)(E_x^{\text{rev}} - u)\,. \tag{2.4}$$

In both cases, we sum up the inputs coming from all synapses. However, whether a presynaptic spike leads to an inhibitory or excitatory PSP is decided differently. In the CUBA model, this is determined by the sign of the synapse or interaction strength w_k. But in the COBA model, the presynaptic spikes only lead to a change in conductance

$$g_x^{\text{syn}}(t) = \sum_{\text{syn. k}} \sum_{\text{spikes s}} w_k \epsilon(t - t_s)\,, \tag{2.5}$$

where the direction of charges flowing passively through the membrane is governed by the *reversal potential* E_x^{rev}. More precisely, an increase of the conductance g_x^{syn} pulls the membrane potential towards the respective reversal potential E_x^{rev}. As in the Hodgkin-Huxley model, excitatory PSPs can mostly be attributed to the inflow of Na^+ ions and inhibitory PSPs to the outflow of K^+ ions. Thus, the excitatory and inhibitory reversal potentials are very close to those of Na^+ and K^+ gates, i.e., $E_{\text{Na}^+}^{\text{rev}} \simeq 50\text{mV}$ and $E_{\text{K}^+}^{\text{rev}} \simeq -77\text{mV}$ (*Gerstner and Kistler*, 2002). Note that w_k is always positive in case of the COBA model as it is a conductance.

The resulting effect of a presynaptic spike at time t_s is described by an interaction kernel $\epsilon(t - t_s)$. As already discussed, in the COBA model a presynaptic spike leads to an increase in conductance which will then exponentially decay back to its initial value on a time scale governed by the *synaptic time constant* τ_{syn}. This can be described by a simple exponential interaction kernel

$$\epsilon(t - t_s) = \theta(t - t_s) \exp\left(-\frac{t - t_s}{\tau_{\text{syn}}}\right). \tag{2.6}$$

A more thorough discussion of synaptic interactions and possible interaction kernels can be found in *Petrovici* (2016). Throughout this thesis, the COBA model with exponential interaction kernel will be used. The resulting ODE for COBA LIF neurons reads:

$$c_{\text{m}} \frac{\text{d}u}{\text{d}t} = g_{\text{L}} \left(E_{\text{L}} - u\right) + g_{\text{e}}^{\text{syn}}(E_{\text{e}}^{\text{rev}} - u) + g_{\text{i}}^{\text{syn}}(E_{\text{i}}^{\text{rev}} - u)\,, \tag{2.7}$$

$$g_x^{\text{syn}}(t) = \sum_{\text{syn. k}} \sum_{\text{spikes s}} w_k\ \theta(t - t_s)\ \exp\left(-\frac{t - t_s}{\tau_{\text{syn}}}\right), \quad x \in \{\text{e}, \text{i}\}\,, \tag{2.8}$$

with the threshold condition given by Eq. 2.2.

2.2 Tsodyks-Markram Model

In the previous section, the interaction between neurons via chemical synapses was introduced (see Fig. 2.3). However, when the presynaptic neuron is firing with a high frequency, the resources in the presynaptic terminal will be slowly depleted and less neurotransmitters will be released per presynaptic event. This corresponds to a time modulation of the interaction strength w_{syn} between pre - and postsynaptic neuron.

One model that describes such *short-term plasticity* (STP) phenomenologically is the *Tsodyks-Markram model* (TSO) (*Tsodyks and Markram*, 1997). In this thesis, we will use a simplified version thereof by *Fuhrmann et al.* (2002); *Maass and Markram* (2002). To incorporate TSO, each synapse gets an additional parameter $R \in [0, 1]$ which corresponds to the amount of resources still available, e.g. vesicles containing neurotransmitters. It is governed by the following ODE:

$$\frac{\mathrm{d}R}{\mathrm{d}t} = \frac{1 - R}{\tau_{\text{rec}}} - U_{\text{SE}} R \delta(t - t_s)\,. \tag{2.9}$$

The first term describes the recovery of used resources $(1 - R)$ with time constant τ_{rec}. The second one gives the fraction of resources $U_{\text{SE}}R$ that are depleted whenever a presynaptic spike arrives, e.g. at time t_s. U_{SE} can take values between $[0, 1]$, whereas $U_{\text{SE}} = 1$ stands for a complete utilization of all resources in one spike.

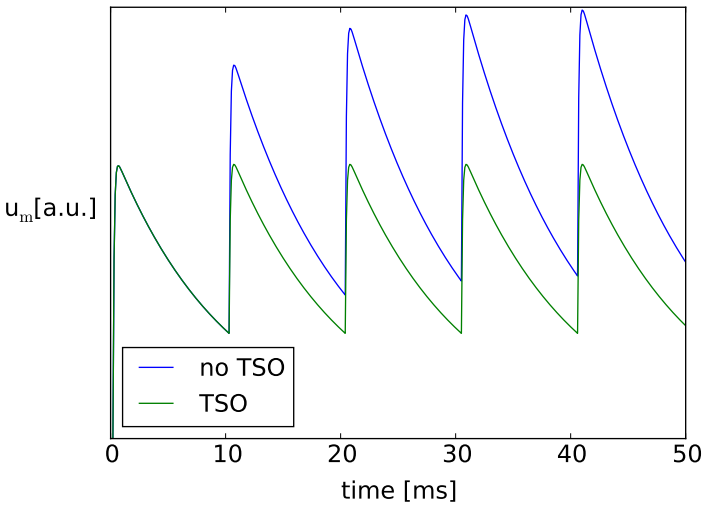

Figure 2.4: Membrane potential of an exponential-shaped COBA LIF neuron that receives a presynaptic spike every refractory period ($\tau_{\text{ref}} = 10\text{ms}$). Without TSO (blue curve), successive PSPs add up and the maximum membrane potential grows in the beginning. With TSO (green curve, $\tau_{\text{rec}} = \tau_{\text{syn}} = 10\text{ms}$), this effect can be mitigated and the initial build-up declines.

The synaptic connection strength is then simply multiplied by the fraction of resources used up during the spike

$$w_{\text{syn}} \to w_{\text{syn}} \cdot U_{\text{SE}} R\,. \tag{2.10}$$

Note that after a presynaptic event, the PSP has an exponential tail that lasts longer than the refractory period. Hence, a series of incoming presynaptic spikes, all with a distance of τ_{ref}, will lead to an initial build-up of the total PSP height. TSO can be used to compensate this effect by setting $\tau_{\text{rec}} = \tau_{\text{syn}}$ and $U_{\text{SE}} = 1$, see Fig. 2.4, and will be utilized later in Chap. 2.3.2 for this purpose.

2.3 Stochastic Computing in Spiking Networks

Over the past decades, experimental findings showed that cortical neurons in the mammalian brain behave inherently stochastic in vivo. It has been proposed that this stochasticity is a hallmark of ongoing probabilistic computations (*Hoyer and Hyvarinen*, 2003; *Körding and Wolpert*, 2004). In such models, it is generally assumed that neural networks utilize this inherent noise to deal with incomplete or noisy data and prevent decision deadlocks, e.g. the ability to re-evaluate a previous decision or classification (*Rolls and Deco*, 2010; *Roumani and Moutoussis*, 2012).

One prominent example of such behavior is *perceptual bistability*, where an image allows several interpretations of its content. For instance, in Fig. 2.5 the animal on the left image can either be identified as a duck or a rabbit. Similarly, the *Necker cube* (right image) can either be seen from above or from below. Between both cases, the quadratic surfaces exchange their position in the cube, i.e., from back - to front side and vice versa.

Figure 2.5: Two well-known examples of perceptual bistability. In both images, there are two ways to interpret the image's content. However, instead of seeing a superposition of both interpretations, the brain is swapping back and forth between the two. **(left)** One can either see a duck or a rabbit. Image modified from *Wikimedia Commons* (2012). **(right)** The cube can either be seen from above or from below. Image taken from *Wikimedia Commons* (2007).

Note that one does not see a superposition of both images, e.g. the duck superimposed with the rabbit, but that our perception switches back and forth in a random manner between both possibilities. A similar phenomenon is *binocular rivalry*, where a different image is presented to each eye. Instead of seeing a superposition of both images, the brain again switches back and forth between the two (see Fig. 2.6).

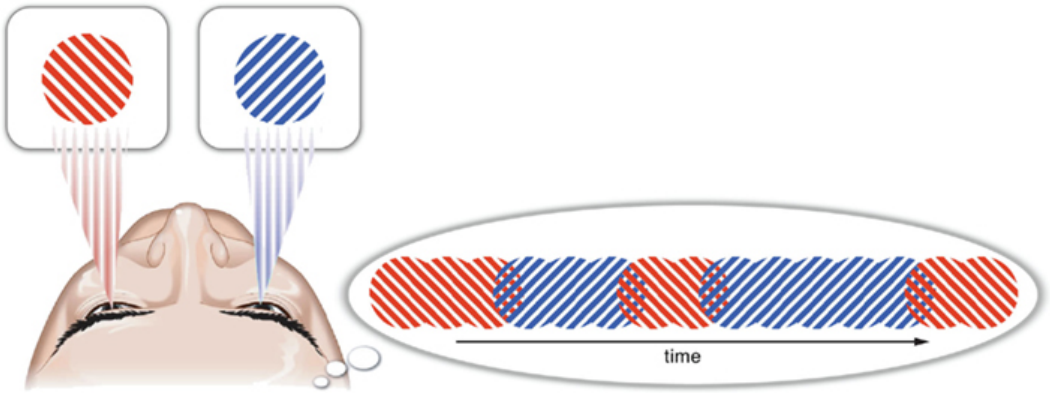

Figure 2.6: A common setup for binocular rivalry. The left eye sees red stripes, tilted by $-45°$ and the right eye sees blue stripes, tilted by $45°$. Instead of seeing a superimposed image, i.e., both the red and blue stripes overlapping, the observed image switches back and forth between the two presented stripe patterns. Image taken from *Dieter and Tadin* (2011).

One possibility to explain this behavior is the following: First, the brain does not simply choose the most likely interpretation of the image. It seems to be switching between interpretations with high probabilities, in our case 'rabbit' and 'duck'. This can be described as *sampling* from a *posterior distribution* that encodes the possible interpretations of the picture.

Sampling is a well-known technique to draw representative samples from a given probability distribution. This is particularly useful for high-dimensional distributions that cannot be computed analytically or numerically. Popular examples of sampling algorithms are the *Metropolis-Hastings algorithm* (*Metropolis et al.*, 1953; *Hastings*, 1970), *Gibbs sampling* (*Geman and Geman*, 1984) and *Hamiltonian Monte Carlo sampling* (*Duane et al.*, 1987).

In our example, the probability landscape of the posterior distribution would consist of two large maxima, one for the rabbit and one for the duck. The network then samples from this distribution, i.e., it starts to explore one of the maxima (e.g. the duck / red stripes), then randomly jumps into the other maximum (the rabbit / blue stripes), exploring it until it jumps back, etc. (see Fig. 2.7).

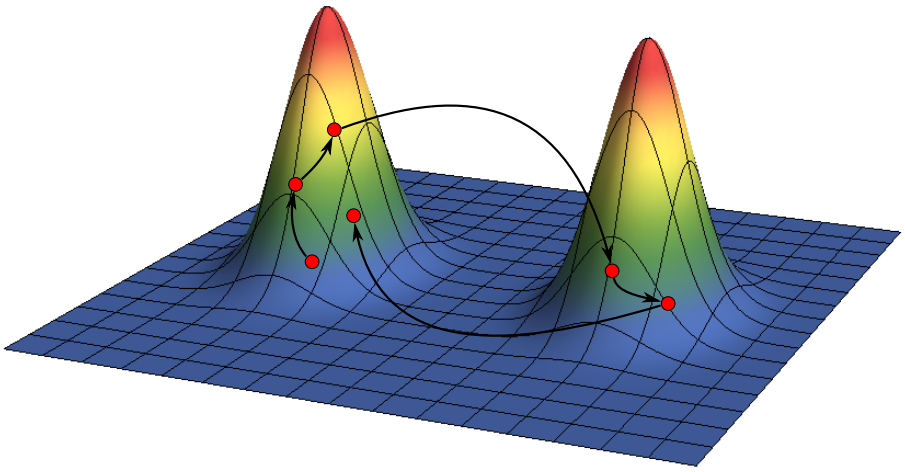

Figure 2.7: Exemplary probability landscape of a posterior distribution. Here, the two maxima would represent the probability of the stimulus being identified as the duck/red stripes or the rabbit/blue stripes. Instead of choosing the interpretation with the highest probability, the neural network explores the probability landscape by sampling from it (red dots). This enables the network to switch back and forth between different maxima.

2.3.1 Neural Sampling

A method to realize stochastic inference in networks of spiking neurons was introduced by *Buesing et al.* (2011). The so-called *neural sampling* is a generalisation of Gibbs sampling and implements a network of abstract neurons with refractory period τ that sample from a *Boltzmann distribution*

$$p(\mathbf{z}) = \frac{1}{Z} \exp(-\beta E(\mathbf{z})) \tag{2.11}$$

with normalization Z and binary state variables $\mathbf{z} = (z_1, z_2, ...)$, $z_k \in \{0, 1\}$. The state of a neuron is encoded as $z_k = 1$ if it is refractory and otherwise $z_k = 0$. The membrane potential of an abstract neuron is given by:

$$u_k(t) = b_k + \sum_{i=1}^{K} W_{ki} z_i(t)\,, \tag{2.12}$$

where K is the total number of neurons, b_k is the bias of neuron k and W_{ki} is the synaptic weight between neuron i and k. If we compare this with the LIF model, the bias can be seen as an offset current or the equilibrium value of the membrane potential, respectively.

The interaction term corresponds to the synaptic term, but with a rectangular PSP instead of an integral over an exponential function. Note that, since we are sampling from a Boltzmann distribution, the weights W_{ij} are symmetric, i.e., $W_{ij} = W_{ji}$ and self-interactions are prohibited, $W_{ii} = 0 \; \forall i$.

The key property of this network is the *neural computability condition*, which relates the membrane potential of a neuron to the probability distribution we want to sample from:

$$u_k(t) = \ln\left(\frac{p(z_k = 1|z_{\setminus k})}{p(z_k = 0|z_{\setminus k})}\right), \tag{2.13}$$

where $\setminus k$ denotes all indices but the kth one. In case of a Boltzmann distribution with energy function

$$E(\mathbf{z}) = -\frac{1}{2}\sum_{i,j} W_{ij} z_i z_j - \sum_i b_i z_i \tag{2.14}$$

and $\beta = 1$, we recover Eq. 2.12 for the membrane potential. From the neural computability condition, we can derive the conditional probability of a single neuron to be in the state $z_k = 1$ depending on its membrane potential u_k:

$$p(z_k = 1|z_{\setminus k}) = \frac{1}{1 + \exp(-u_k)} = \sigma(u_k)\,. \tag{2.15}$$

Before we can discuss the actual sampling mechanism, we have to introduce an additional variable ζ_k which simply counts how long neuron k has been refractory. If neuron k spikes, we set $z_k = 1$ and $\zeta_k = \tau$. Afterwards, ζ_k will count downwards in each subsequent time step until it reaches 0, where it remains until the neuron spikes again. This is illustrated in Fig. 2.8.

Finally, sampling is done in the following way:

1. Initialize the network in a state $\mathbf{z}' = (z'_1, z'_2, ..., z'_K)$.

2. Now update in a fixed order the state of each neuron. The transition probability $T^k(z_k, \zeta_k|z'_k, \zeta'_k)$ for neuron k to change its state $(z'_k, \zeta'_k) \to (z_k, \zeta_k)$ is given by:

$$T^k(1,\ \tau \mid (z'_k,\ \zeta'_k) \in [(0,0),\ (1,1)]) = \sigma(u'_k - \ln\tau)\,, \tag{2.16a}$$
$$T^k(0,\ 0 \mid (z'_k,\ \zeta'_k) \in [(0,0),\ (1,1)]) = 1 - \sigma(u'_k - \ln\tau)\,, \tag{2.16b}$$
$$T^k(1,\ \zeta'_k - 1 \mid 1,\ \zeta'_k \in [2,\tau]) = 1\,, \tag{2.16c}$$
$$\text{otherwise} \ = 0\,,$$

where already updated neurons are taken into account. Note that $\ln \tau$ compensates for longer refractory periods, as we will have τ 1-states after spiking (see Fig. 2.8). This has to be fixed by increasing the number of 0-states accordingly, i.e., by reducing the spiking probability. In the special case of $\tau = 1$, neural sampling reduces to Gibbs sampling.

3. After updating all neurons in sequence, the new state of the network is $\mathbf{z}$. This state is a fair sample of the underlying Boltzmann distribution. By repeating step 2, one obtains additional subsequent samples.

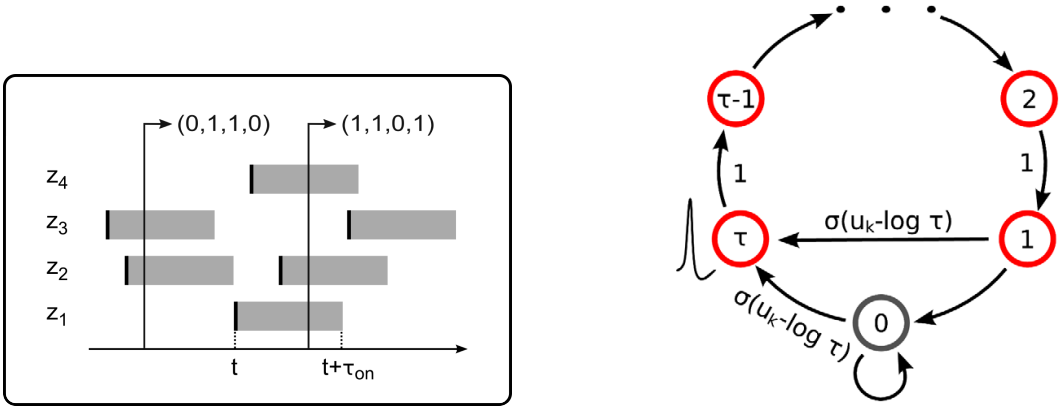

Figure 2.8: **(left)** The state of the neural network is determined by the refractory periods of the neurons. If a neuron is refractory, it is in the 1-state, otherwise in the 0-state. Image taken from *Petrovici et al.* (2013). **(right)** Update scheme of neural sampling. If the neuron is refractory, it will simply decrease to 1 in subsequent time steps. After the neuron leaves the refractory period, it can either spike again immediately or transition into the 0-state, where it remains until it spikes again with probability $\sigma(u_k - \ln \tau)$. Image taken from *Buesing et al.* (2011).

2.3.2 LIF Sampling

Recently, it was proven that neural sampling can be carried over to a network of CUBA and COBA LIF neurons (*Petrovici et al.*, 2013, 2015a; *Petrovici*, 2016). It is therefore possible to perform stochastic computations in a biologically inspired network of spiking LIF neurons by sampling from a Boltzmann distribution with the states again being encoded via the refractory and non-refractory state of the neurons.

To find a mapping between the parameters of the abstract neurons used in neural sampling and the more complex COBA LIF neurons, two preconditions have to be met:

1. LIF neurons are inherently deterministic and only spike when trespassing the threshold (see Eq. 2.7). The probability of spiking at a certain membrane potential,

also called the *activation function*, is zero below the threshold. Therefore, we have to add a source of randomness to our network in order to make the LIF neurons stochastic.

2. The activation function encodes the conditional probabilities of the target Boltzmann distribution. Hence, we also want the activation function of our LIF neurons to have the same shape as the logistic activation function used in neural sampling (see Eq. 2.15).

For COBA LIF neurons, it can be shown that both conditions are met when we push the neurons into the *high-conductance state*. This is done by adding excitatory and inhibitory high-frequency *Poisson noise* to each neuron.

First, the Poisson noise (or *shot noise*) is generated by a discrete *Poisson process* $\eta(t)$ which is defined by the following two equations:

$$\langle \eta(t) \rangle = \nu \,, \tag{2.17a}$$

$$\langle \eta(t)\eta(t') \rangle = 2\pi\delta(t - t') \,. \tag{2.17b}$$

A Poisson source emits spikes at times $(t_1, t_2, t_3, ...)$ with frequency ν. The number of spikes n in a given time interval T is distributed according to a Poisson distribution

$$p_\nu(n) = \frac{(\nu T)^n}{n!} \exp(-\nu T) \tag{2.18}$$

and the actual spike times are all completely independent, i.e., the generated noise is completely uncorrelated as demanded by Eq. 2.17b.

It can be shown that, as long as we are in the high-conductance state, the COBA LIF equation takes the form of the Langevin equation of the Ornstein-Uhlenbeck process (*Uhlenbeck and Ornstein*, 1930)

$$\mathrm{d}u(t) = \theta \cdot (\mu - u(t))\mathrm{d}t + \sigma \mathrm{d}W \tag{2.19}$$

with the following parameters (*Petrovici*, 2016):

$$\theta = \frac{1}{\tau_{\mathrm{syn}}} \,, \tag{2.20a}$$

$$\mu = \frac{g_{\mathrm{L}} E_{\mathrm{L}} + \sum_j \nu_j w_j E_j^{\mathrm{rev}} \tau_{\mathrm{syn}}}{\langle g^{\mathrm{tot}} \rangle} \,, \tag{2.20b}$$

$$\frac{\sigma^2}{2} = \frac{\sum_j \nu_j w_j^2 (E_j^{\mathrm{rev}} - \mu)^2 \tau_{\mathrm{syn}}}{2 \langle g^{\mathrm{tot}} \rangle^2} \,, \tag{2.20c}$$

$$g^{\mathrm{tot}}(t) = g_{\mathrm{L}} + \sum_j g_j^{\mathrm{syn}}(t) \,. \tag{2.20d}$$

The Ornstein-Uhlenbeck process describes the Brownian motion $\mathrm{d}W$ of a particle (also known as *Wiener process*) with an additional drift or attractor term, respectively. In our case, the drift term is given by the membrane's leak and the Brownian motion or random walk originates from the high-frequency bombardment of the excitatory and inhibitory Poisson noise, where each spike increases or decreases the membrane potential by a small amount.

The stationary solution of the Ornstein-Uhlenbeck process is a Gaussian distribution for the membrane potential with mean μ and standard deviation $\sigma \cdot \sqrt{2\theta}^{-1}$. Thus, the membrane is now symmetrically fluctuating around a mean value allowing the neuron to spike even though the mean membrane potential is below threshold (or not spike, even though the mean membrane potential is above threshold). The general solution of the Ornstein-Uhlenbeck process for finite times t given an initial membrane potential u_0 is given by:

$$p(u,t|u_0) = \sqrt{\frac{\theta}{\pi\sigma^2(1-e^{-2\theta t})}} \exp\left(-\frac{\theta}{\sigma^2} \frac{(u-\mu+(\mu-u_0)e^{-\theta t})^2}{1-e^{-2\theta t}} \right). \tag{2.21}$$

For $t = 0$, this corresponds to a delta peak at u_0, i.e., the membrane potential is perfectly localized at an arbitrary initial value u_0. Due to the drift term, it is drawn to the mean value μ. The diffusion leads to a widening of the distribution until the final width $\sigma \cdot \sqrt{2\theta}^{-1}$ is reached (see Fig. 2.9).

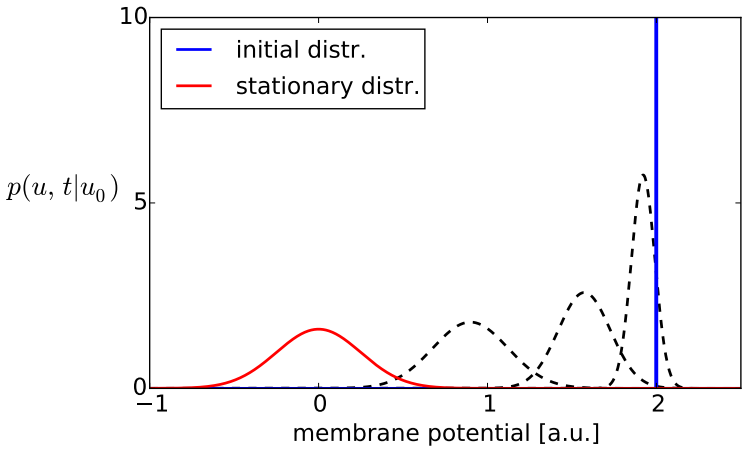

Figure 2.9: Solution of the Ornstein-Uhlenbeck process at different times for $u_0 = 2.0$, $\mu = 0$, $\theta = 2.0$ and $\sigma = 0.5$, all unitless. The initial distribution is a delta peak at $u = u_0$ (blue). Firstly, because of the drift term the particle will be drawn to the equilibrium value μ. Secondly, the distribution widens due to Brownian motion (dashed). Since we have an attracting force towards the equilibrium value and a diffusion term acting outwards, the distribution will settle at a stationary solution with finite width for $t \to \infty$ (red).

The ODE of the COBA LIF neuron can also be rewritten in the following way:

$$\tau_{\text{eff}}(t)\frac{\mathrm{d}u}{\mathrm{d}t} = u_{\text{eff}}(t) - u(t)\,, \tag{2.22a}$$

$$u_{\text{eff}}(t) = \frac{g_{\text{L}}E_{\text{L}} + \sum_j g_j^{\text{syn}}(t)E_j^{\text{rev}}}{g^{\text{tot}}(t)}\,, \tag{2.22b}$$

$$\tau_{\text{eff}}(t) = \frac{c_{\text{m}}}{g^{\text{tot}}(t)}\,. \tag{2.22c}$$

These equations describe the dynamics of $u(t)$ as the decay towards a time-dependent equilibrium value $u_{\text{eff}}(t)$ with an effective membrane time constant $\tau_{\text{eff}}(t)$. Also note that the synaptic input changes the total conductivity $g^{\text{tot}}(t)$. If we increase the synaptic input, the total conductivity rises and the effective membrane time constant becomes smaller. Hence, in the high-conductance state, the membrane dynamics are very fast due to the high conductance. Also note that $u(t)$ is a low-pass filtered version of $u_{\text{eff}}(t)$, but $u_{\text{eff}}(t)$ is not clamped to the reset potential after spiking. In addition, we obtain $\tau_{\text{eff}} \to 0$ and thus $u(t) \simeq u_{\text{eff}}(t)$ in the limit of high input rates (see Fig. 2.10).

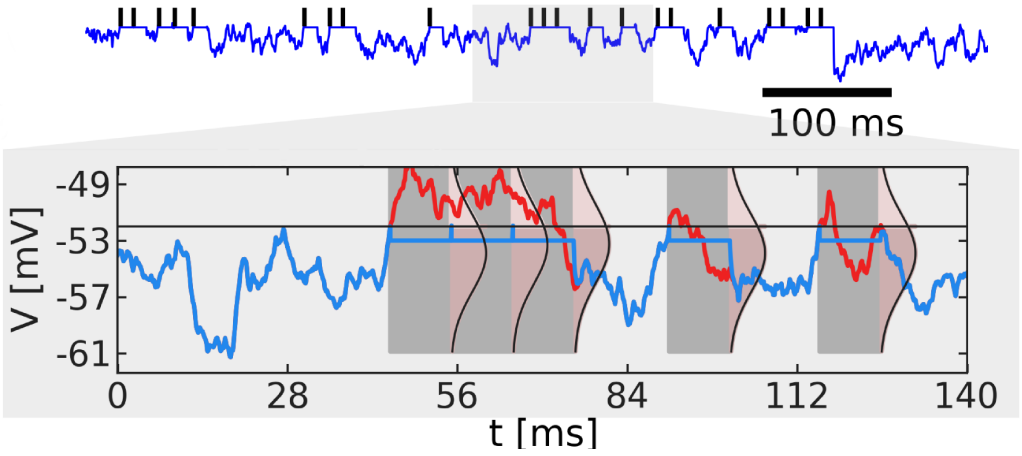

Figure 2.10: Membrane potential of a COBA LIF neuron in the high-conductance state. **(top)** Trace of the membrane potential. Spikes are highlighted by black bars, followed by the refractory period where the membrane potential is clamped to the reset potential. **(bottom)** Zoom in of the picture above. In blue, the membrane potential of the neuron and in red the trace of the effective or free membrane potential are shown. After a refractory period, the membrane potential decays back towards the effective membrane potential. If u_{eff} is above the threshold, this leads to another spike. If it is below the threshold, the neuron remains silent. The gray areas highlight the refractory period. After each refractory period, the expected distribution of the free membrane potential (a Gaussian) is drawn. Image modified from *Petrovici* (2016), original from *Petrovici et al.* (2013).

Combined with the solution of the Ornstein-Uhlenbeck process, this can be used to prove that the activation function in the high-conductance state can indeed be approximated by a logistic function. The method used is called *autocorrelation propagation formalism*, see *Petrovici* (2016) for a thorough discussion.

To translate the theoretical bias and weights from neural sampling, we now have to do the following:

1. First, we have to determine the activation function of the LIF neuron. This is done by measuring the activity at different leak potentials E_L of the neuron with strong excitatory and inhibitory noise, whereas the activity is defined as the ratio of the time the neuron was refractory and the total simulation time. After obtaining the activation curve, we can perform a logistic fit

$$\sigma(u, \alpha, u_{\mathrm{p}0.5}) = \frac{1}{1 + \exp\left(-\frac{u - u_{p0.5}}{\alpha}\right)}\,, \tag{2.23}$$

   where $u_{\mathrm{p}0.5}$ is the membrane potential at which the neuron is refractory with a probability $p(z = 1) = 0.5$. α is a factor determining the slope of the activation function.

2. The bias b_k can be translated by demanding that, without any synaptic input from other neurons, the activities of abstract neuron and LIF neuron have to be equal, i.e.,

$$\frac{1}{1 + \exp\left(-b_k\right)} \stackrel{!}{=} \frac{1}{1 + \exp\left(-\frac{u - u_{\mathrm{p}0.5}}{\alpha}\right)} \tag{2.24}$$

   and therefore

$$E_\mathrm{L} = (\alpha b + u_{\mathrm{p}0.5}) \frac{\langle g^\mathrm{tot} \rangle}{g_\mathrm{L}}\,. \tag{2.25}$$

3. The weights can be translated by setting the integral of the PSPs of abstract and LIF neurons equal:

$$\int_0^{\tau_\mathrm{ref}} \mathrm{PSP}_\mathrm{abstract}\ \mathrm{d}t \stackrel{!}{=} \int_0^{\tau_\mathrm{ref}} \mathrm{PSP}_\mathrm{LIF}\ \mathrm{d}t\,. \tag{2.26}$$

   For abstract neurons, the PSPs have a simple rectangular shape with height W_{ij} and length τ_ref. The PSPs of COBA LIF neurons are more complicated since we have to integrate over exponential functions with time-dependent exponents. However, in case of the high-conductance state, an accurate approximation of this

integral can be found analytically and we obtain a difference of exponentials as PSP shape. Hence, by equalizing the integrals over the refractory period of the PSPs, we can derive a translation rule for the weights:

$$w_{ij}^{\mathrm{LIF}} = \beta W_{ij}^{\mathrm{abstract}}\,, \tag{2.27}$$

$$\beta = \frac{\alpha c_{\mathrm{m}} \tau_{\mathrm{ref}} \left(\frac{1}{\tau_{\mathrm{syn}}} - \frac{1}{\tau_{\mathrm{eff}}}\right)}{E_{ij}^{\mathrm{rev}} - \langle u \rangle} \left[\tau_{\mathrm{syn}} \left(e^{-\frac{\tau_{\mathrm{ref}}}{\tau_{\mathrm{syn}}}} - 1\right) - \tau_{\mathrm{eff}} \left(e^{-\frac{\tau_{\mathrm{ref}}}{\tau_{\mathrm{eff}}}} - 1\right)\right]^{-1}. \tag{2.28}$$

The intuitive explanation for this rule is that a presynaptic spike should have the same net effect on the membrane potential of both abstract and LIF postsynaptic neurons, i.e., the same net flux of 'charges'.

Finally, the PSPs of LIF neurons decay exponentially, thus we will always have interaction remaining after the refractory period. To compensate for unwanted build-ups from overlapping PSPs during bursts, one can use STP (see Chap. 2.2). After translating the theoretical parameters, the LIF network can be used to sample from the given target distribution, as demonstrated in Fig. 2.11.

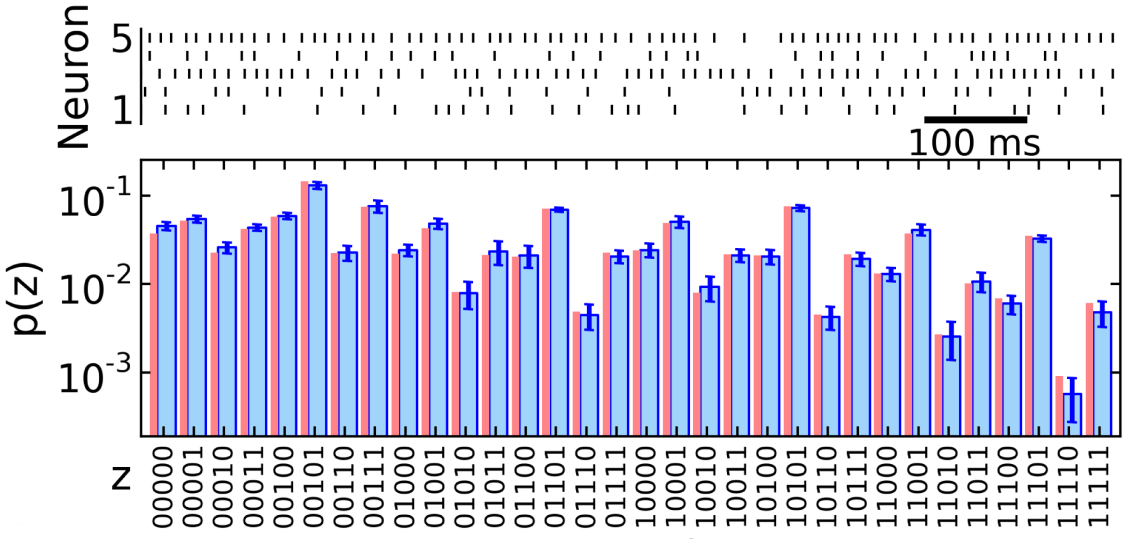

Figure 2.11: Sampling from a Boltzmann distribution over binary variables with COBA LIF neurons. **(top)** Raster plot showing the spike times of a five-neuron network. The spike times are sufficient to calculate the states of the network. **(bottom)** The target distribution is given in red, the sampled one in blue. As we can see, the LIF neurons produce a close approximation of the desired target distribution. Note the logarithmic scale on the y-axis. Image modified from *Petrovici et al.* (2013).

2.3.3 Kullback-Leibler Divergence

At this point, we need a measure that quantifies how accurately our LIF networks are actually sampling from their respective target distribution.

A standard measure is the *Kullback-Leibler divergence* (*Kullback and Leibler*, 1951), also denoted as D_{KL} and sometimes called *relative entropy*, which can be motivated as follows:

First, we assume a set of symbols $X = (x_1, x_2, ...)$ and the probabilities of their occurrence $p_X = (p_{x_1}, p_{x_2}, ...)$. For instance, X could be the English alphabet and p_X the frequency of those letters in English literature. If we want to store, for example, a text file or a book, the minimal mean memory required per sign is given by the *Shannon entropy* (*Shannon*, 1948)

$$H(p_X) = -\sum_i p_{x_i} \ln(p_{x_i})\,. \tag{2.29}$$

$\ln(p_{x_i})$ is the information content of the symbol x_i. Therefore, the entropy gives us a measure for the mean information content of the set of symbols X. Now, if we do not know the underlying distribution p_X but only an approximation q_X, the mean storage needed per symbol is

$$H(p_X||q_X) = -\sum_i p_{x_i} \ln(q_{x_i})\,, \tag{2.30}$$

where we assign the information content $\ln(q_{x_i})$ to each symbol, but the real occurrence of the symbol is given by p_{x_i}. The redundant amount of storage that we need because we are not using the correct distribution to encode the symbols is given by the Kullback-Leibler divergence:

$$\begin{aligned} D_{\mathrm{KL}}(p_X||q_X) &= H(p_X||q_X) - H(p_X) \\ &= \sum_i p_{x_i} \ln\left(\frac{p_{x_i}}{q_{x_i}}\right). \end{aligned} \tag{2.31}$$

Note that the D_{KL} is only zero for $p_X = q_X$, however it is not a metric (for instance it is not symmetric). To allow sampled probabilities of zero and to ensure that the D_{KL} remains monotonic during sampling, we calculate it in the following way:

$$D_{\mathrm{KL}}^{\mathrm{used}} = \sum_i p_i^{\mathrm{sampled}} \ln\left(\frac{p_i^{\mathrm{sampled}}}{p_i^{\mathrm{target}}}\right). \tag{2.32}$$

A typical D_{KL} trace for LIF sampling is shown in Fig. 2.12. In the beginning, we have almost no samples and therefore the approximation of the target distribution is

unreliable, which is reflected in the high D_{KL} value. During sampling, the D_{KL} decreases but saturates after a certain time. This saturation does not occur for other sampling methods, like Gibbs sampling or sampling with abstract neurons. There are several reasons for this behavior:

1. The activation function does not exactly match a logistic function, even though it can be fitted by one, and resembles rather an error function.

2. The translation rules are only approximations. For example, we replaced time-dependent variables by their expectation values. Furthermore, the weight translation was derived by matching the PSP shapes of abstract and LIF neurons, which is also merely an approximation.

3. The PSP shape of LIF neurons is not rectangular and has a contribution for times larger than the refractory period, i.e., when the interaction is already over in case of abstract neurons.

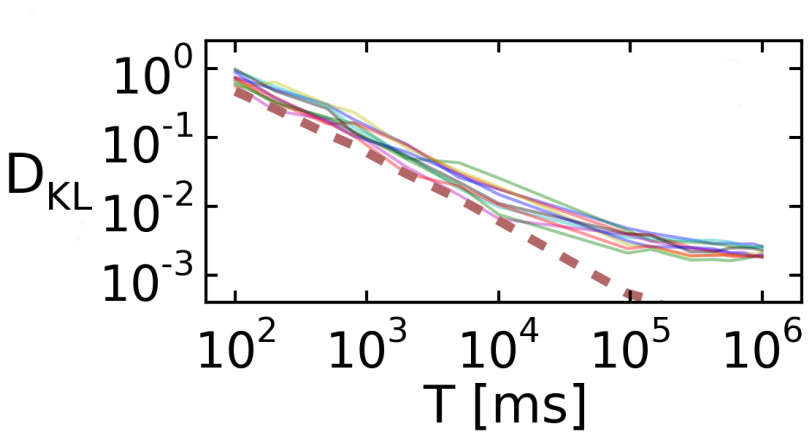

Figure 2.12: Typical D_{KL} curves obtained from LIF sampling (solid lines). For early readout times, when we have only a small amount of samples, the sampled distribution does not agree with the target distribution. This is reflected in the high D_{KL}. During sampling, the D_{KL} decreases as we are getting closer to the target distribution. However, LIF sampling does not converge onto the target distribution due to systematic differences in the theory of neural sampling and LIF sampling. In contrast, algorithms like Gibbs sampling or neural sampling converge towards the target distribution by design, i.e., towards $D_{\mathrm{KL}} = 0$ (dashed line). Image modified from *Petrovici et al.* (2013).

2.4 Training Networks of Spiking Neurons

2.4.1 Boltzmann Machines

Until now, we have shown how to set the parameters of a LIF network in order to sample from a predefined Boltzmann distribution over binary variables. However, in real-world applications, for instance image recognition, the underlying distribution is not known in advance. Thus, it would be preferable to train the network on certain tasks, for instance image classification, auditive recognition or as an artificial intelligence in e.g. (video) games with human-like behavior.

There is a powerful model in machine learning that utilizes binary Boltzmann distributions to perform such tasks: *Boltzmann machines* (*Ackley et al.*, 1985), commonly denoted as BMs. A *fully visible Boltzmann machine* is an undirected graph that consists of several binary units, all connected via symmetric weights $W_{ij}\ \forall i, j$, $W_{ii} = 0\ \forall i$. The units can take the states 0 and 1, and the state vectors are distributed according to a Boltzmann distribution with the already mentioned weights W_{ij} and biases $b_i\ \forall i$. Note that LIF networks which sample from such Boltzmann distributions are in fact Boltzmann machines.

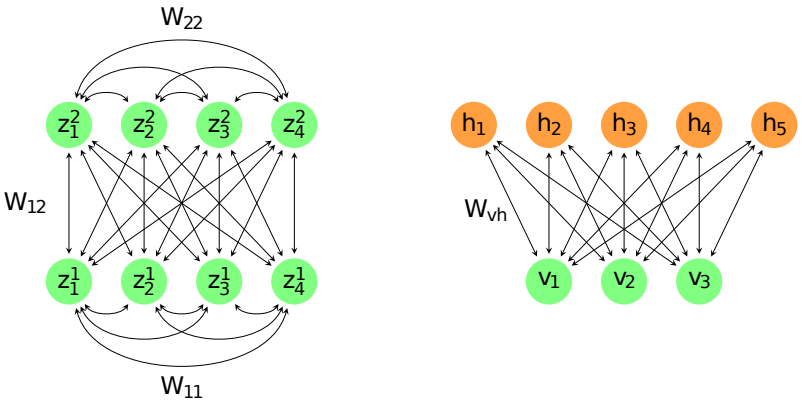

Figure 2.13: Graphical representation of Boltzmann machines. **(left)** A fully visible BM has connections between all neurons. **(right)** A restricted BM consists of two layers, a so-called visible (v) and hidden (h) layer. Connections between neurons within a layer are prohibited and only neurons of neighbouring layers are connected. The neurons of the visible layer represent the network's input and output, for instance pixel values of an image we want to classify and the corresponding image labels. The hidden layer increases the representative power of the BM and acts as a feature detector. Image taken from *Petrovici* (2016).

However, it turns out that training fully visible Boltzmann machines is rather inefficient due to the sheer number of connections. For most applications, so-called *restricted Boltzmann machines* (*Salakhutdinov et al.*, 2007) are commonly used. They exhibit a

layered structure, reducing the number of possible connections since neurons of the same layer are completely unconnected. Furthermore, the representational power of these networks can be easily increased by adding more layers or increasing the number of units in already existing ones. Such networks are known as *deep Boltzmann machines* (*Salakhutdinov and Hinton*, 2009) due to their deep layered structure.

LIF BMs classifying data sets of the MNIST database of handwritten data (*LeCun et al.*, 1998) have been successfully implemented in *Leng* (2014). Not only did LIF networks achieve state-of-the-art classification rates, it was also demonstrated that TSO helps the network to switch faster between image classes while dreaming (*Leng*, 2014; *Martel*, 2015; *Leng et al.*, 2016). When dreaming, the visible layer is completely unclamped and the whole network is allowed to develop freely, generating images similar to those it was trained on. Conventional Gibbs sampling, which is usually used to sample from Boltzmann machines, shows comparatively bad mixing properties, i.e., only very few of the learned images are visited while dreaming, see Fig. 2.14. Thus LIF sampling cannot only be used for image classification, but also as a very good generating model of the trained data.

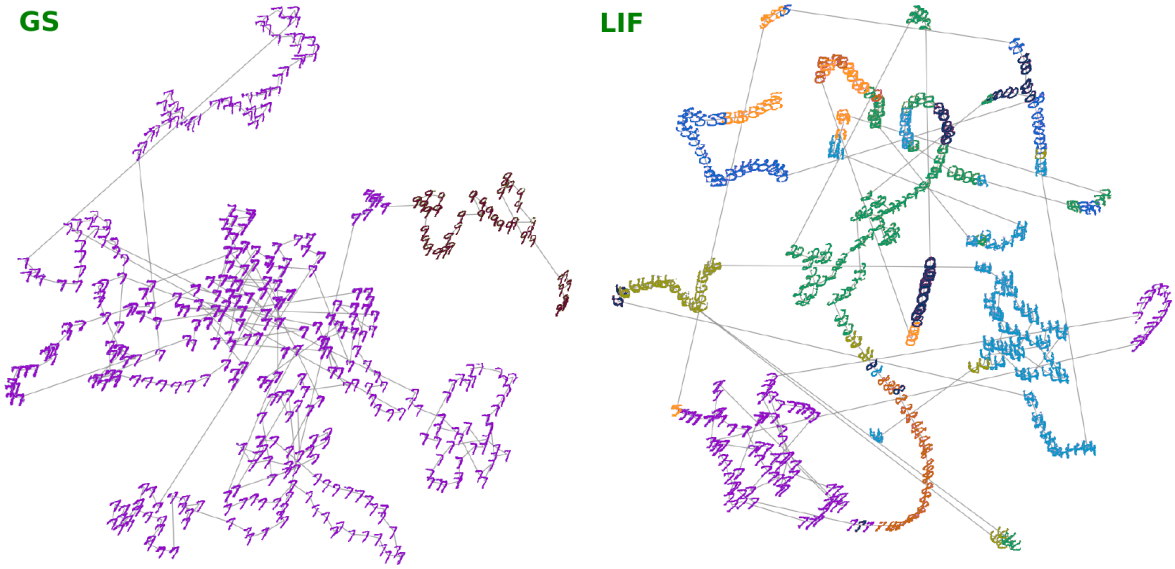

Figure 2.14: Output of a dreaming BM, generated with Gibbs sampling (left) and LIF sampling (right). Different image classes are shown in different colors, the gray line connects consecutively generated images. Gibbs sampling primarily stays in one mode, i.e., produces mainly samples from one of the trained images. LIF networks, however, can have very good mixing properties due to short-term plasticity acting similarly to local temperature changes (β in Eq. 2.11). The visualisation of the generated data shown here is called *t-Distributed Stochastic Neighbor Embedding* (*Maaten and Hinton*, 2008). Image taken from *Leng et al.* (2016).

2.4.2 Contrastive Divergence

BMs can be trained with a very efficient algorithm called *Contrastive Divergence* (CD) (*Hinton*, 2002, 2010), making them particularly attractive for machine learning tasks. We will only derive the update scheme for fully visible BMs. The equivalent scheme for restricted BMs, however, can be derived in a similar way. Again, see *Petrovici* (2016) for more details.

The idea is the following: After running the network for a specified runtime, we compare the generated state statistics with our target distribution or training data. If there are differences (for example, the network was trained on the digit '1', but never or rarely produces it), we adjust the parameters of the Boltzmann distribution such that the desired states become more probable and undesired states less probable. The change in probability of a state $\mathbf{z}$ dependent on the weights w_{ij} is:

$$\begin{aligned}\frac{\partial p(\mathbf{z})}{\partial w_{ij}} &= \frac{\partial}{\partial w_{ij}}\left(\frac{e^{-E(\mathbf{z})}}{\sum_{\mathbf{z'}} e^{-E(\mathbf{z'})}}\right) \\ &= p(\mathbf{z}) z_i z_j - p(\mathbf{z})\left(\frac{\sum_{\mathbf{z'}} z'_i z'_j e^{-E(\mathbf{z'})}}{\sum_{\mathbf{z'}} e^{-E(\mathbf{z'})}}\right). \end{aligned} \tag{2.33}$$

Dividing by $p(\mathbf{z})$, we obtain the derivative of the log-likelihood. Note that maximizing the log-likelihood is equivalent to maximizing the probability itself, since the log function is strictly monotonic. Updating parameters this way is commonly known as *Maximum Likelihood learning*. The ratio of sums on the right-hand side is simply the expectation value of $z_i z_j$ over our current Boltzmann distribution, which we will call the model average of $z_i z_j$:

$$\frac{\partial \ln p(\mathbf{z})}{\partial w_{ij}} = z_i z_j - \langle z_i z_j \rangle_{\text{model}} \,. \tag{2.34}$$

Averaging over the training data, we get:

$$\left\langle \frac{\partial \ln p(\mathbf{z})}{\partial w_{ij}} \right\rangle_{\text{data}} = \langle z_i z_j \rangle_{\text{data}} - \langle z_i z_j \rangle_{\text{model}} \,. \tag{2.35}$$

This result can be used to perform *gradient ascent* learning, i.e., to change the weights into the particular direction in weight space that locally maximizes the log-likelihood:

$$\Delta w_{ij} = \eta_{\text{CD}} \cdot \left(\langle z_i z_j \rangle_{\text{data}} - \langle z_i z_j \rangle_{\text{model}} \right) \tag{2.36a}$$

$$= \eta_{\text{CD}} \cdot \left(p(z_i = 1, z_j = 1)_{\text{data}} - p(z_i = 1, z_j = 1)_{\text{model}} \right). \tag{2.36b}$$

Thus, the weight update can be rewritten as the difference of joint probabilities between the target distribution (data) and the distribution of the network (model). $\eta_{\mathrm{CD}} \in [0, 1]$ is the learning rate. The update rules for the biases can be derived analogously:

$$\Delta b_i = \eta_{\mathrm{CD}} \cdot \left(\langle z_i \rangle_{\mathrm{data}} - \langle z_i \rangle_{\mathrm{model}} \right) \tag{2.37a}$$

$$= \eta_{\mathrm{CD}} \cdot \left(p(z_i = 1)_{\mathrm{data}} - p(z_i = 1)_{\mathrm{model}} \right). \tag{2.37b}$$

For large Boltzmann machines, the model averages have to be obtained via sampling. However, it is unclear how long one has to sample for convergence. *Hinton* (2002) therefore proposed an update scheme where the averages are approximated after just a few sampling steps n, for instance even $n = 1$. This is known as CD or CD_n and can be used to train LIF BMs on arbitrary machine learning applications.

3 A Sea of Boltzmann Machines: The Simplest Case

The final goal of this thesis is to significantly reduce the external Poisson stimuli needed for LIF neurons. In the end, we will see how it is possible to run networks of BMs reliably even without any external noise input. However, to arrive at this goal, we first have to take a look at a very simple setup and gradually introduce more complexity until we obtain a network where the external noise can essentially be turned off. This approach will help us understand the subtle differences between running a BM with completely independent external noise sources and intrinsic background noise from a network of BMs itself.

The simplest possible setup consists of one Boltzmann machine without Poisson sources that receives its noise spike trains from an uncorrelated sea of BMs, i.e., in this case a pool of independent BMs as illustrated in Fig. 3.1. To exclude possible cross-correlations between the spike trains which would lead to correlated noise, only one neuron of each BM contributes for now.

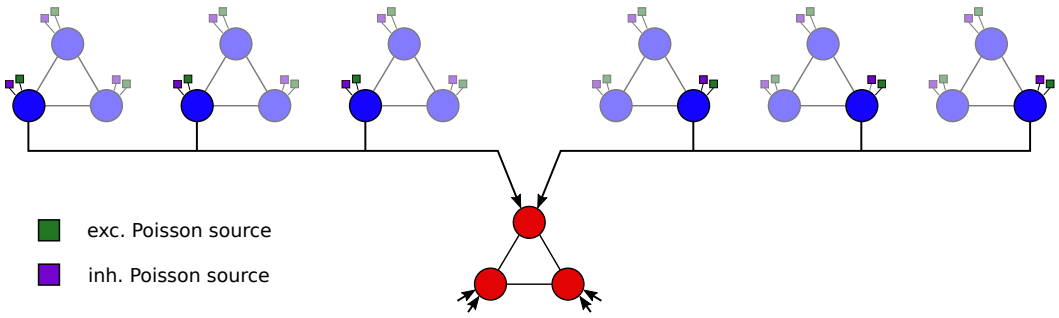

Figure 3.1: Schematic of the setup used throughout this chapter. Instead of using Poisson sources, the neurons of the red BM are fed with spikes coming from a sea of BMs (blue). For example, the three blue BMs on the left could provide inhibitory noise input, and the right ones excitatory. The other two neurons in the red BM are fed in the same way with noise, not shown in this image. The blue BMs are still driven by Poisson sources (small colored boxes).

Note that the noise-generating neurons are still driven by Poisson sources. In the following chapter, the main differences concerning the dynamics of a neuron driven by Poisson or network noise will be discussed. Furthermore, it will be illustrated that LIF

sampling is possible while replacing all Poisson sources with noise coming from other BMs.

3.1 Autocorrelations in Noise Spike Trains

In ordinary LIF sampling, the neurons become stochastic due to high-frequency inhibitory and excitatory Poisson noise. By definition (Eq. 2.17b), the spike times originating from a Poisson source are completely uncorrelated. In contrast, the minimal distance between two spikes coming from a LIF neuron is limited by the refractory period (see Fig. 3.2). Furthermore, from intermediate to high activities, small bursts of consecutive spikes may appear due to the free membrane potential being above threshold for a time longer than the refractory period (as can be seen in Fig. 2.10).

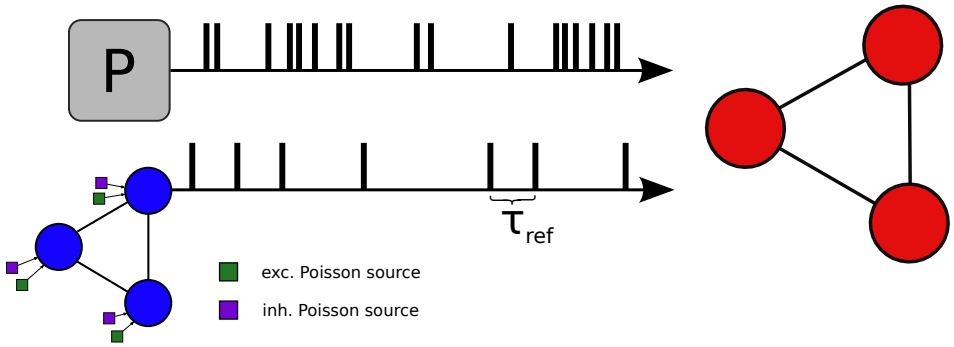

Figure 3.2: Schematic representation of the difference between BM-generated noise and ideal Poisson sources. For Poisson sources, the individual spike times are all completely independent. However, the spikes of a LIF neuron cannot have a distance smaller than the refractory period τ_{ref}. Furthermore, small bursts of consecutive spikes with interspike distances of τ_{ref} may happen, giving the spike train a deterministic structure.

Since we are not only using a single neuron as a noise source but many, the restriction of limited spike distances is not problematic as e.g. the spikes of a second neuron can occur at all times, including during the refractory period of the first neuron. However, small bursts introduce autocorrelations at times $\neq 0$ to the noise spike trains, making the noise colored (correlated) instead of white (completely uncorrelated), which is a notable difference from Poisson sources.

The normalized autocorrelation function ρ_{a} of a function, for instance a spike train $\eta(t) = \sum_{\text{spikes s}} \delta(t - t_s)$, is defined as:

$$\rho_{\text{a}}(\Delta) = \frac{\langle (\eta(t+\Delta) - \langle \eta \rangle)(\eta(t) - \langle \eta \rangle) \rangle}{\text{Var}(\eta)}\,, \tag{3.1}$$

where $\langle\cdot\rangle$ denotes the expectation value and $\mathrm{Var}(\cdot)$ the variance. Intuitively, the autocorrelation function encodes how similar the spike train is to itself some time lag Δ later. Since spikes are only characterized by their spike time, we have to bin the spike train $\eta(t)$ in order to evaluate its autocorrelation function, i.e., discretize it into time intervals Δt and count the number of occurred spikes in each interval. Hence, the equation becomes

$$\rho_{\mathrm{a},n} = \frac{\sum_i (\eta_{i+n}^{\mathrm{bin}} - \langle\eta^{\mathrm{bin}}\rangle)(\eta_i^{\mathrm{bin}} - \langle\eta^{\mathrm{bin}}\rangle)}{\mathrm{Var}(\eta^{\mathrm{bin}})} \tag{3.2}$$

with η_i^{bin} being the number of spikes during the time interval $i\Delta t$ and $(i+1)\Delta t$. Due to the *Wiener-Khintchine theorem* (*Wiener*, 1930; *Khintchine*, 1934), an efficient way of calculating an autocorrelation function is via the fast Fourier transform $\mathcal{F}$:

$$\rho_{\mathrm{a},n} = \mathcal{F}^{-1}(\mathcal{F}(\eta^{\mathrm{bin}})\mathcal{F}^*(\eta^{\mathrm{bin}}))|_n \,. \tag{3.3}$$

The complexity of the fast Fourier transform applied to an array with N entries is $\mathcal{O}(N \log N)$ (*Cooley and Tukey*, 1965), and using several Fourier transforms in series simply adds a constant multiplicative factor to the complexity. But directly evaluating the sum in Eq. 3.2 has a complexity of $\mathcal{O}(N^2)$, becoming greatly inferior to the fast Fourier transform for large arrays.

To investigate the influence of correlated noise on the dynamics of a single LIF neuron, the parameters of the noise-generating BMs will be set constant and equal for the time being. Consequently, each BM has the same weight matrix and bias vector. Also, since we are only interested in the influence of bursting neurons on the noise quality, all neurons in a BM will share the same bias. By increasing or decreasing the bias, the activity of the neurons inside the BM can thus be easily adjusted. Of course, this rather unrealistic restriction will be dropped in later simulations. The used parameters can be found in App. 8.2. For all simulations in this thesis, the software packages NEST, PyNN and SBS have been used, see App. 8.3 for more details.

3.1.1 Correlation Patterns in the Free Membrane Potential

First, we can take a look at the autocorrelation functions of the used noise (see Fig. 3.3). In case of Poisson noise, the autocorrelation function is given by a delta peak at $\Delta = 0$. But for noise coming from BMs, additional peaks at multiples of the refractory period emerge. Depending on how active the neurons of the noise-generating BMs are, i.e., how many bursts they emit and how long these are, the additional sidepeak structure will be more or less pronounced.

For instance, if all neurons rarely spike and hence almost no bursts appear, the autocorrelation function will return to a single delta peak at $\Delta = 0$. However, increasing the activity creates sidepeaks at increasingly large time lags. Note that, since bursts consist of multiple consecutive spikes with distance τ_{ref}, the sidepeaks only appear at multiples of τ_{ref}. Also, the sidepeak structure decays on a certain time scale τ_{c} and goes

to zero for large time lags, meaning that noise spikes become uncorrelated for large time differences.

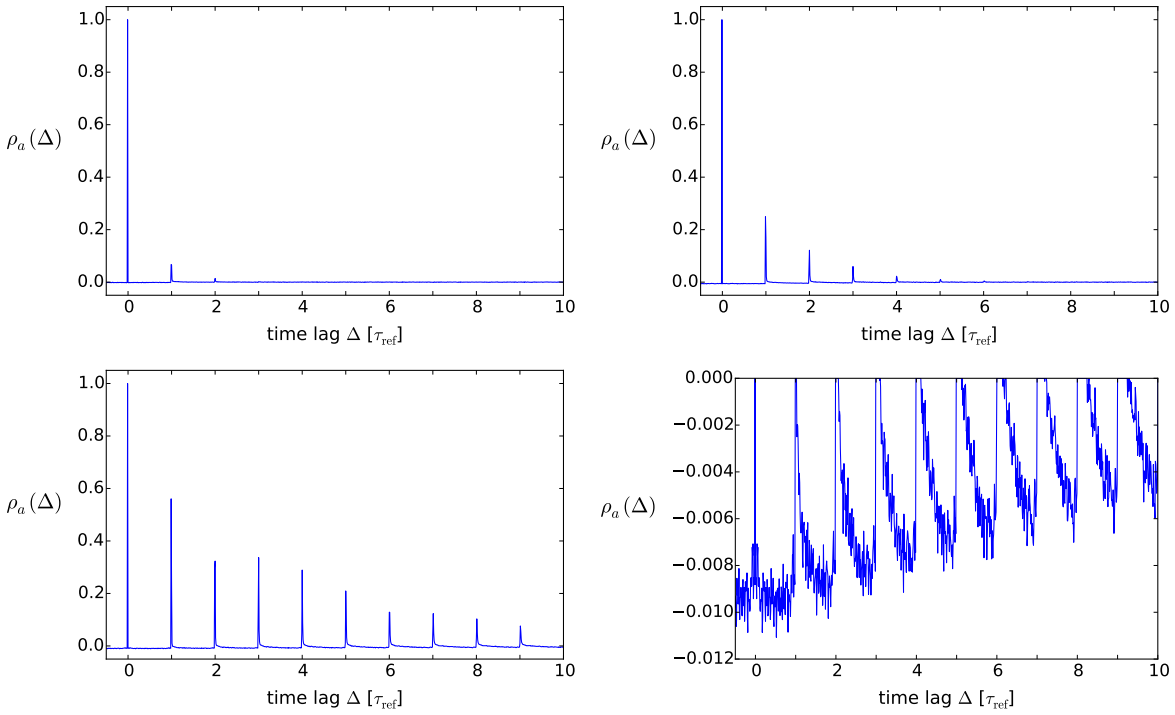

Figure 3.3: Autocorrelation function of spike trains generated by neurons with different mean frequency $\bar{\nu}$. (top left) $\bar{\nu} = 0.14\ \tau_{\text{ref}}^{-1}$, (top right) $\bar{\nu} = 0.53\ \tau_{\text{ref}}^{-1}$ and (bottom left) $\bar{\nu} = 0.89\ \tau_{\text{ref}}^{-1}$. For higher mean frequencies, the neurons are more likely to emit small bursts of consecutive spikes, with each spike having a distance of τ_{ref} to the following spike. This leads to an increase in autocorrelation at multiples of the refractory period τ_{ref}. Furthermore, bursting also introduces an absence of spikes during the refractory period, leading to negative parts in the autocorrelation function. This is shown for $\bar{\nu} = 0.89\ \tau_{\text{ref}}^{-1}$ in the bottom right plot.

Further note that between sidepeaks, the autocorrelation function becomes slightly negative. This is again an effect of the refractory mechanism of the neurons that make up the noise spike train. Each time a neuron spikes, it cannot spike again during its refractory period. Therefore, the probability to observe a noise spike during this time interval is reduced, leading to a negative autocorrelation. This will become important later on when looking at the autocorrelation function of the free membrane potential of a neuron driven by these noise spike trains.

The noise autocorrelations have a significant effect on the dynamics of the free membrane potential. In the Poissonian case, the trace of the free membrane potential shows

no fixed patterns or structures since the noise spike times responsible for PSPs are all random and independent. However, if we replace Poisson noise with noise coming from BMs, patterns emerge dependent on the activity of the noise neurons (see Fig. 3.4). For instance, in case of extremely bursting neurons, blocks of fixed patterns appear as the membrane potential is repeatedly going through the same trajectory for some time. If the activity of the noise neurons is reduced, these blocks start to disappear until we reach the uncorrelated Poissonian limit again.

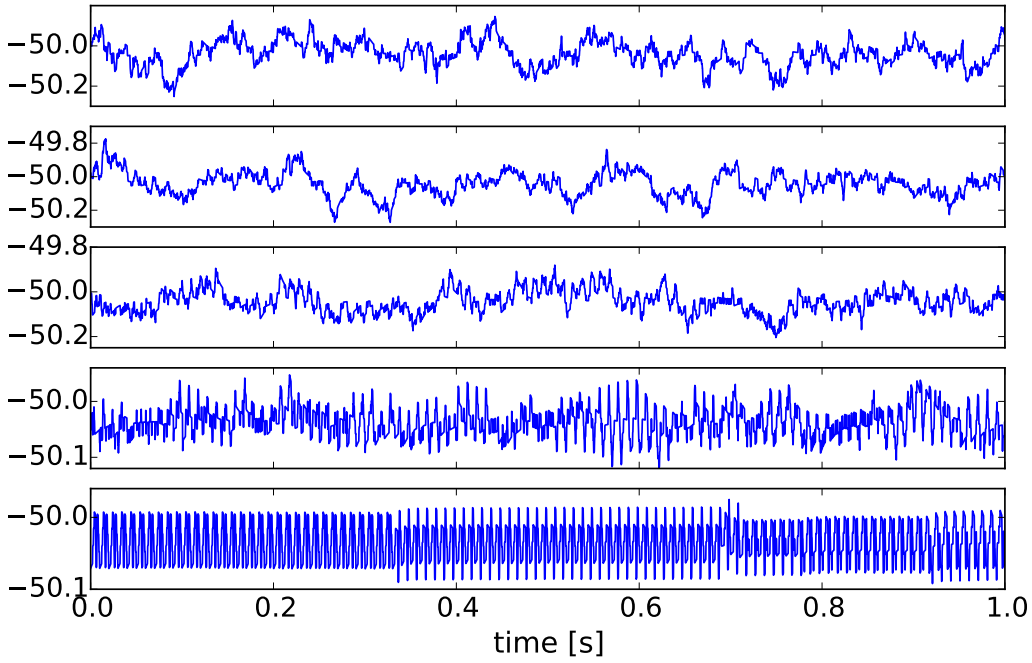

Figure 3.4: Trace of the free membrane potential of a neuron driven by (top) Poisson noise and (others) BM-generated noise from neurons with different mean frequencies, from top to bottom: $\bar{\nu} = 0.14,\ 0.53,\ 0.98,\ 0.9997\ \tau_{\mathrm{ref}}^{-1}$. For a neuron driven by Poisson noise, the membrane is performing a random walk around its mean resulting in a trace without repeated patterns. This is still the case when low or medium activity neurons are providing noise. But already for $\bar{\nu} = 0.53\ \tau_{\mathrm{ref}}^{-1}$, small substructures can be seen in the membrane trace. This effect becomes even stronger for higher activities until we obtain large blocks with fixed patterns in the extremely high activity case. Note that the dynamic range of the membrane potential is reduced for higher activities.

The origin of the blocks is quite intuitive: Assuming the inhibitory noise is, for example, generated by five neurons which are all bursting strongly, the noise spike train will show

the same sequence of spike times until one of the neurons stops spiking for a short period, resulting in a slight change of the sequence. After some time, all neurons will have stopped bursting briefly and the pattern the free membrane potential is traversing will be completely uncorrelated with the initial one.

An interesting question to ask now is whether we are still able to obtain a logistic activation function with extremely correlated noise. In fact, as can be seen in Fig. 3.5, the free membrane distribution becomes Gaussian after sampling long enough. Additionally, the activation function gradually becomes logistic as well. But due to the fixed patterns in the free membrane trace, this process of convergence towards a Gaussian takes much longer as compared to a neuron driven by Poisson noise. More precisely, the time scale on which independent random fluctuations act on the membrane potential is larger than for Poisson stimuli. This has an important consequence for LIF sampling, as we may have to calibrate longer with BM-generated noise than with Poisson noise to obtain the correct activation function. A demonstration of this effect is shown in App. 8.8.1 and will not be discussed here.

Note that the neuron is not performing an Ornstein-Uhlenbeck (OU) process anymore due to the noise being colored. In fact, the new process can be described with the same Langevin equation as the OU process, with the exception that the noise term is now itself generated by an OU process instead of a Poisson process.

In case of a linear Langevin equation

$$\frac{\mathrm{d}x}{\mathrm{d}t} = \gamma(\mu - x) + \eta(t) \tag{3.4}$$

with Gaussian colored noise $\eta(t)$ and position variable x, an analytical solution can be found. For example, with a delta peak as the initial distribution, the stationary distribution is given by a Gaussian with mean μ and finite variance (*Hanggi and Jung*, 1995; *Cáceres*, 1999)

$$\lim_{t\to\infty} \sigma^2(t) = \frac{1}{2\gamma} \int_0^\infty \langle \eta(0)\eta(\tau) \rangle \, e^{-\gamma\tau} \, \mathrm{d}\tau \,, \tag{3.5}$$

as will be demonstrated in the next section. The difference to the OU process can be seen very nicely by looking at the autocorrelation function of the free membrane potential (see Fig. 3.6). In case of an OU process, this is given by an exponential decay with time constant τ_{syn} and can be interpreted as the 'memory' of the synapse. For noise spike trains with autocorrelations on time scales $\tau_{\mathrm{c}} \ll \tau_{\mathrm{syn}}$, we obtain a similar result. Increasing the activity of the noise neurons, spike structures at multiples of τ_{ref} appear for $\tau_{\mathrm{c}} \simeq \tau_{\mathrm{syn}}$. In the limit of very strongly bursting neurons, i.e., $\tau_{\mathrm{c}} \gg \tau_{\mathrm{syn}}$, the autocorrelation function of the free membrane potential decays on the time scale τ_{c} and shows negative parts between multiples of the refractory time.

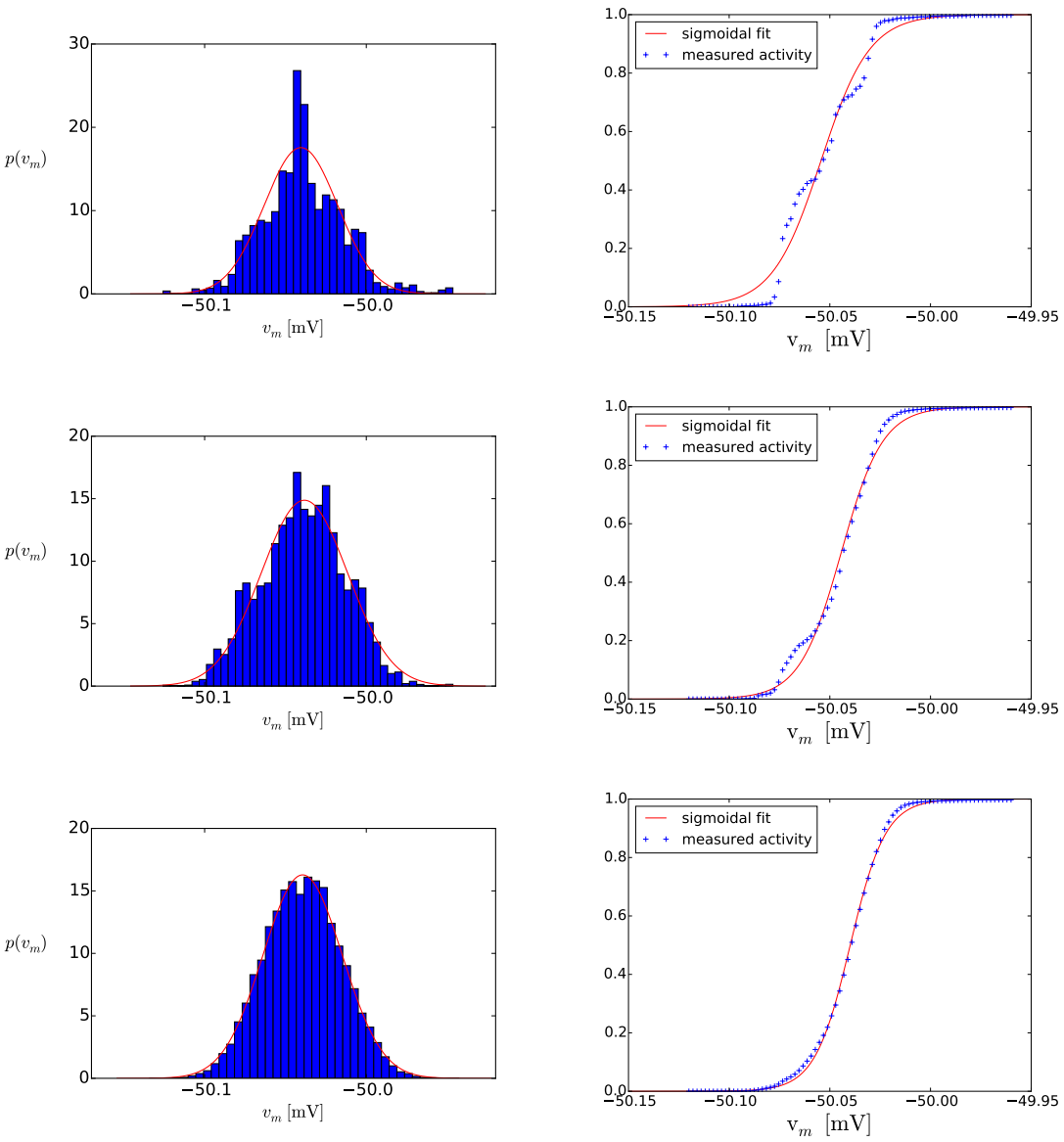

Figure 3.5: Free membrane potential distribution of a neuron driven by strongly autocorrelated noise ($\bar{\nu} = 0.9997\ \tau_{\text{ref}}^{-1}$) and the corresponding activation function after running the simulation for (top) $1 \cdot 10^4$ms, (middle) $5 \cdot 10^4$ms and (bottom) $5 \cdot 10^5$ms for each data point. On the left, we see that for short simulation durations the free membrane distribution does not resemble a Gaussian. However, increasing the run time leads to better results. A Gaussian fit to each distribution is included in red. On the right side, the measured activation functions are presented in blue, with the corresponding logistic fit in red. Again, for short simulation times, the activation function does not resemble a logistic function and has additional systematic deviations coming from the autocorrelations. Measuring over longer times recovers the desired symmetric shape.

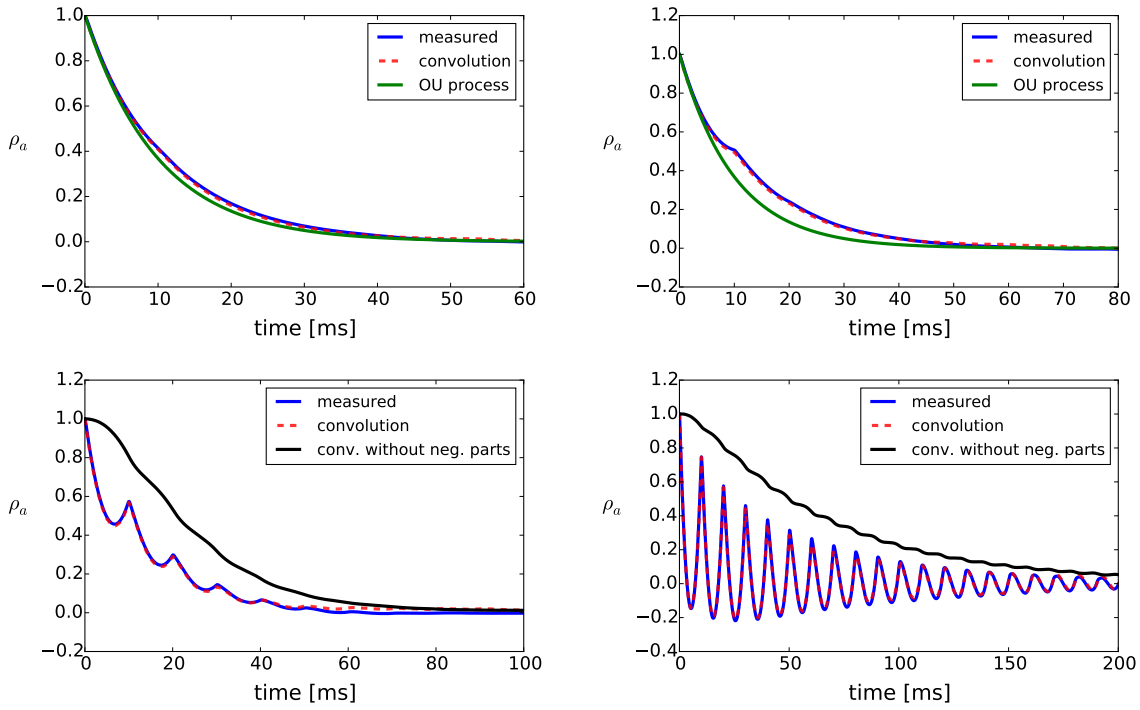

Figure 3.6: Autocorrelation function of the free membrane potential for a neuron with $\tau_{\text{syn}} = 10\text{ms}$ driven by BM-generated noise with different activities. The noise is generated by neurons with a mean frequency of (top left) $\bar{\nu} = 0.014\ \tau_{\text{ref}}^{-1}$, (top right) $\bar{\nu} = 0.14\ \tau_{\text{ref}}^{-1}$, (bottom left) $\bar{\nu} = 0.53\ \tau_{\text{ref}}^{-1}$ and (bottom right) $\bar{\nu} = 0.89\ \tau_{\text{ref}}^{-1}$. In blue, the measured autocorrelation is given. The red dashed lines show the result of the normalized convolution of the noise autocorrelation function and the exponential decay kernel $\exp\left(-\frac{|t|}{\tau_{\text{syn}}}\right)$ (see Eq. 3.6). In green, the autocorrelation function $\exp\left(-\frac{t}{\tau_{\text{syn}}}\right)$ of the OU process is shown. For low activities, we return to a similar result as for the OU process. Finite autocorrelations in the noise, however, lead to a structure with spikes. This effect is even more pronounced in the two bottom plots. The sharp spikes originate from an accelerated decay due to the negative parts of the noise autocorrelation function combined with the positive delta peaks that lead to a sudden rise in correlation. The black curves in the bottom plots are calculated the same way as the red dashed lines, but with all negative parts of the noise autocorrelation function set to 0. This leads to an absence of the strongly decaying part between multiples of the refractory period.

This also demonstrates that we could return to the OU case by increasing the synaptic time constant τ_{syn}, such that correlations in the spike train occur on time scales that are smaller than the inherent autocorrelation of the OU process coming from the synaptic interaction. As before, this would slow down the time evolution of the membrane potential, changing the time scale on which independent dynamics can happen.

It was further observed that the autocorrelation function of the normalized free membrane potential $\bar{u}(t) = u(t) - \langle u \rangle$ can be obtained by convolving the spike train autocorrelation function with the synaptic exponential decay $\exp(-\frac{|t|}{\tau_{\text{syn}}})$:

$$\lim_{t\to\infty} \langle \bar{u}(t)\bar{u}(t+\Delta)\rangle \propto \int_{-\infty}^{\infty} \langle \eta(0)\eta(\tau)\rangle \exp\big(-\frac{|\Delta-\tau|}{\tau_{\text{syn}}}\big)\,\mathrm{d}\tau\,. \tag{3.6}$$

Eq. 3.6 can be motivated as follows: An incoming spike leads to a rapid change of the membrane potential which then starts to decay exponentially. This way, self-similarity is introduced to the trace of the membrane potential resulting in an exponentially shaped autocorrelation function. However, bursting adds even more structure, further propagating these exponential autocorrelations to larger time lags. Convolution takes this propagation of correlations into account. For a proof of Eq. 3.6, see App. 8.7.

Since the spike train autocorrelations have negative values between multiples of the refractory period (see Fig. 3.3), we obtain small minima in the free membrane potential autocorrelations that might even become negative, as demonstrated in Fig. 3.6. This is a clear difference to the correlations induced by the OU process on the membrane dynamics.

3.1.2 Effect on the Free Membrane Potential Distribution

Another major difference between a neuron driven by Poisson noise and BM-generated noise is the width of the free membrane potential distribution. In case of BM-generated noise, the width is narrower as with equivalent Poisson sources, i.e., Poisson noise with the same frequency and weights. The effect becomes more pronounced if we increase the activity of the noise neurons.

The width of the stationary free membrane potential distribution is given by Eq. 3.5 and can be derived analytically for COBA LIF neurons in the high-conductance state. The ansatz presented here is inspired from *Schwarz* (2012).

For simplicity, we assume that both inhibitory and excitatory noise sources have approximately the same frequency, weight and autocorrelation function. The ODE of COBA LIF neurons in the high-conductance state can be solved perturbatively (*Petrovici*, 2016):

$$u(t) = u_0 + \sum_{k\in\{\mathrm{e,i}\}} \sum_{\text{spikes s}} \Lambda_k \Theta(t-t_s) \left[\exp\left(-\frac{t-t_s}{\tau_k^{\mathrm{syn}}} \right) - \exp\left(-\frac{t-t_s}{\langle \tau_{\mathrm{eff}} \rangle} \right) \right], \tag{3.7a}$$

$$\Lambda_k = \frac{\tau_k^{\mathrm{syn}} \omega_k \left(E_k^{\mathrm{rev}} - \langle u_{\mathrm{eff}} \rangle \right)}{\langle g^{\mathrm{tot}} \rangle \left(\tau_k^{\mathrm{syn}} - \langle \tau_{\mathrm{eff}} \rangle \right)}, \tag{3.7b}$$

with an offset u_0. Choosing the reversal potentials symmetrically around $\langle u_{\mathrm{eff}} \rangle$, u_0 can be identified as $\langle u \rangle$. Moreover, since we are in the high-conductance state, which is characterized by a very fast membrane $\langle \tau_{\mathrm{eff}} \rangle \to 0$, we get

$$\bar{u}(t) \propto \sum_{\text{spikes s}} \Lambda_{\mathrm{e}} \Theta(t-t_s) \exp\left(-\frac{t-t_s}{\tau_{\mathrm{e}}^{\mathrm{syn}}} \right) + \sum_{\text{spikes s}} \Lambda_{\mathrm{i}} \Theta(t-t_s) \exp\left(-\frac{t-t_s}{\tau_{\mathrm{i}}^{\mathrm{syn}}} \right) \tag{3.8}$$

with $\bar{u}(t) = u(t) - \langle u \rangle$. Then, using $S_x(t') = \sum_{\text{spikes s}} \delta(t'-t_s)$ with $x \in \{\mathrm{e,i}\}$ to represent the noise spike trains and neglecting the constant prefactor $\Lambda = \Lambda_{\mathrm{e}} = -\Lambda_{\mathrm{i}}$, we get:

$$\bar{u}(t) \propto \int_0^t \mathrm{d}t' S_{\mathrm{e}}(t') \exp\left(-\frac{t-t'}{\tau_{\mathrm{e}}^{\mathrm{syn}}} \right) - \int_0^t \mathrm{d}t' S_{\mathrm{i}}(t') \exp\left(-\frac{t-t'}{\tau_{\mathrm{i}}^{\mathrm{syn}}} \right). \tag{3.9}$$

Note that the Θ functions were absorbed into the integration boundaries. With this, we can now calculate $\langle \bar{u}(t)^2 \rangle$. Since S_{e} and S_{i} are assumed to have the same autocorrelation function $\langle S(t')S(t'') \rangle$ as well as $\tau_{\mathrm{syn}} = \tau_{\mathrm{e}}^{\mathrm{syn}} = \tau_{\mathrm{i}}^{\mathrm{syn}}$, we obtain:

$$\langle \bar{u}(t)^2 \rangle \propto \int_0^t \mathrm{d}t' \int_0^t \mathrm{d}t'' \, \langle S(t')S(t'') \rangle \, e^{-\frac{t-t'}{\tau_{\mathrm{syn}}}} e^{-\frac{t-t''}{\tau_{\mathrm{syn}}}}. \tag{3.10}$$

In our case, the autocorrelation function of the noise spike trains is symmetric and depends only on the time difference $\langle S(t')S(t'') \rangle = f(|t'-t''|)$. This can be used to split the integral into two terms for $t' > t''$ and $t'' < t'$:

$$\begin{aligned} \langle \bar{u}(t)^2 \rangle &\propto \int_0^t \mathrm{d}t' \int_0^{t'} \mathrm{d}t'' f(t'-t'') \; ... + \int_0^{t''} \mathrm{d}t' \int_0^t \mathrm{d}t'' f(t''-t') \; ... \\ &= 2 \int_0^t \mathrm{d}t' \int_0^{t'} \mathrm{d}t'' f(t'-t'') e^{-\frac{t-t'}{\tau_{\mathrm{syn}}}} e^{-\frac{t-t''}{\tau_{\mathrm{syn}}}}. \end{aligned} \tag{3.11}$$

Here, the second term was brought into the same form as the first one by relabeling $t' \to t''$ and $t'' \to t'$. One of the integrals can be evaluated with a change of variables $\tau = t' - t''$ and $\kappa = t''$:

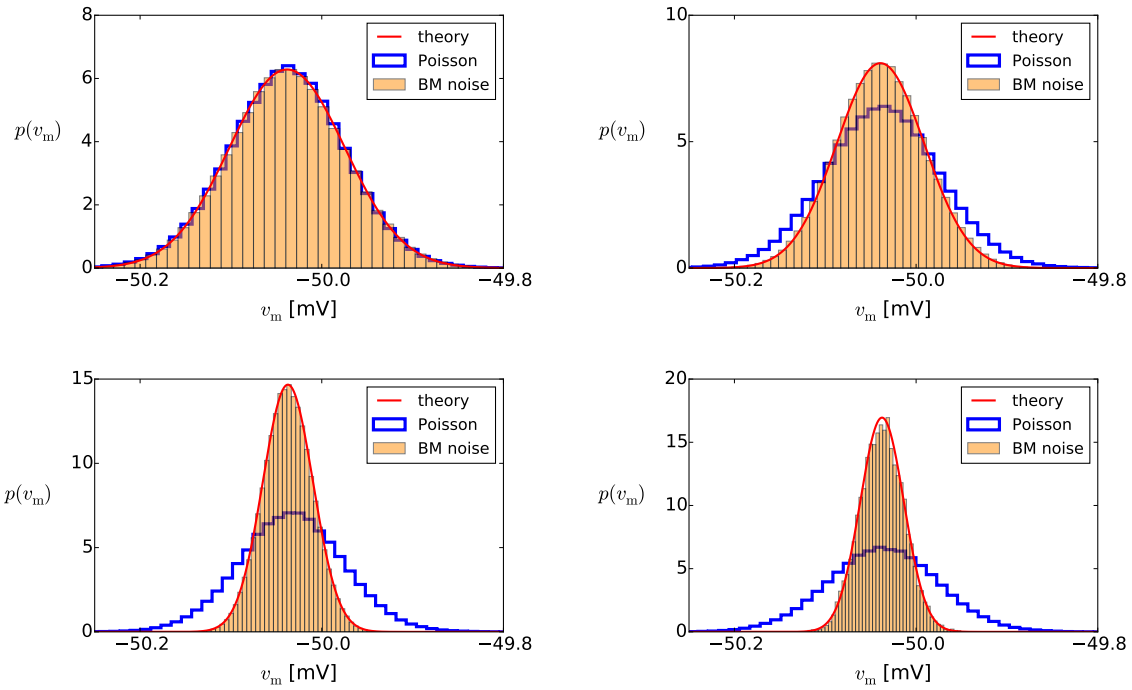

Figure 3.7: Free membrane potential distribution for (orange) a neuron driven by BM-generated noise from neurons with different mean frequencies and (blue) the same neuron driven by equivalent Poisson sources, i.e., same frequencies and weights as the BM-generated noise input. In red, the theoretical distribution is plotted, where the width of the distribution with Poisson sources was simply rescaled by the factor given in Eq. 3.13. The mean frequencies of the noise neurons are (top left) $\bar{\nu} = 0.14\ \tau_\mathrm{ref}^{-1}$, (top right) $\bar{\nu} = 0.53\ \tau_\mathrm{ref}^{-1}$, (bottom left) $\bar{\nu} = 0.89\ \tau_\mathrm{ref}^{-1}$ and (bottom right) $\bar{\nu} = 0.9997\ \tau_\mathrm{ref}^{-1}$. With increased activity the width of the free membrane potential distribution becomes narrower.

$$\begin{aligned}\langle \bar{u}(t)^2 \rangle &\propto e^{-\frac{2t}{\tau_\mathrm{syn}}} \int_0^t \mathrm{d}\tau \int_0^{t-\tau} \mathrm{d}\kappa f(\tau) \exp\left(\frac{2\kappa+\tau}{\tau_\mathrm{syn}}\right) \\ &= \frac{\tau_\mathrm{syn}}{2} \int_0^t \mathrm{d}\tau f(\tau) \left[\exp\left(-\frac{\tau}{\tau_\mathrm{syn}}\right) - \exp\left(\frac{\tau-2t}{\tau_\mathrm{syn}}\right)\right]. \end{aligned} \tag{3.12}$$

For the stationary solution, we have to set $t \to \infty$. Thus, the final result is:

$$\lim_{t\to\infty} \langle \bar{u}(t)^2 \rangle \propto \int_{-\infty}^{\infty} \mathrm{d}\tau f(\tau) \exp\left(-\frac{|\tau|}{\tau_\mathrm{syn}}\right) = \Gamma\,. \tag{3.13}$$

For Poisson noise, $f(\tau)$ is a delta peak at 0 and therefore Γ is 1. However, for autocorrelated noise, we obtain results different from 1, translating to a rescaling of the width

of the membrane distribution. Hence, we can obtain the width of the free membrane distribution for autocorrelated noise by rescaling the width generated by equivalent but uncorrelated Poisson sources (see Fig. 3.7).

A more intuitive explanation for the reduced width can be found by looking at the k-th neighbour distance distribution of the noise spike trains. It measures how probable it is to find exactly k spikes in a time interval Δt. For instance, for $k = 1$ it represents the interspike interval distribution, i.e., the distribution of interspike distances.

Due to autocorrelations, there is a distinct preference for spikes having a distance that is a multiple of τ_{ref} to their k-th neighbour. In the low-activity regime, where the autocorrelations are weak and bursting rarely appears, the distribution is similar to the one of a Poisson source, with the exception of small peaks at multiples of the refractory period. For higher activities, the distribution becomes distorted when reaching a multiple of the refractory period because spikes that belong to the same bursting neuron coincide.

Furthermore, notice that in the right plot in Fig. 3.8, the distribution quickly vanishes for $k < 5$ and $\Delta t > \tau_{\text{ref}}$. In this case, the noise spike train was generated by five network BMs. Therefore, exactly five bursting network neurons are contributing to the noise spike train. Since the spikes of every network neuron have a distance of approximately τ_{ref}, after one network neuron spikes the first four neighbouring spikes have to come from the other network neurons. Hence, their distance cannot be much larger than τ_{ref}. Similarly, for $k = 6$ all spikes have distances larger than τ_{ref}. This introduces an absence of close-distance spike bursts in the noise. These small bursts lead to the large and rapid increase or decrease of the membrane potential observed in the Poisson case and is absent for BM-generated noise in the high activity regime, resulting in the smaller width of the free membrane potential distribution. This effect also occurs if one increases the number of BMs that contribute to one noise source (see App. 8.8.2).

Again, another illustration is that, in case of strongly bursting neurons, the instantaneous frequency of the spike train varies less than for sparsely firing neurons. For example, if the noise neurons are very active, they are very close to their maximum frequency of τ_{ref}^{-1} which also strictly limits the maximum frequency of the noise spike train (see Fig. 3.9). Larger variations in frequency, however, allow periods of strong excitation or strong inhibition leading to a larger membrane potential distribution width as observed for Poisson sources.

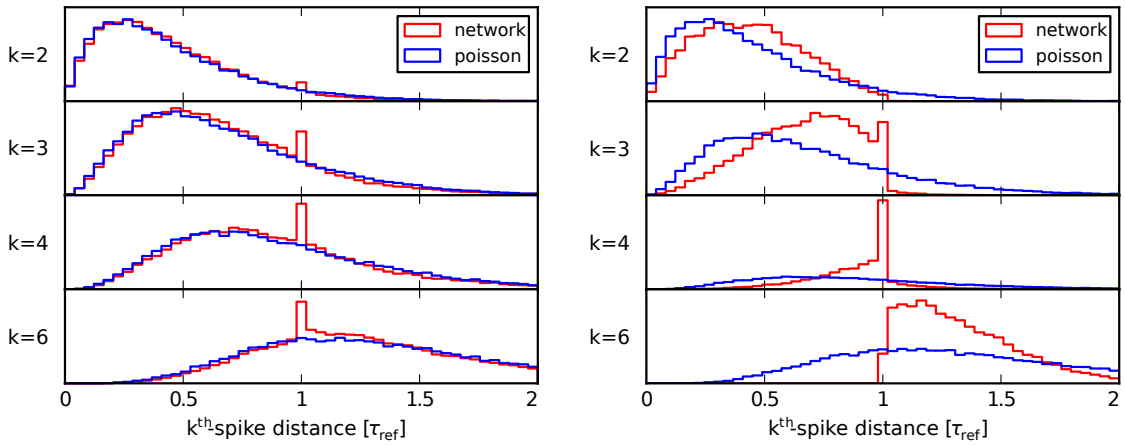

Figure 3.8: k-th neighbour interval distribution for (red) BM-generated noise from neurons with mean frequencies (left) $\bar{\nu} = 0.14\ \tau_{\text{ref}}^{-1}$, (right) $\bar{\nu} = 0.89\ \tau_{\text{ref}}^{-1}$ and (blue) noise from an equivalent Poisson source. For $k = 1$ we simply obtain the interspike interval distribution. Note that for a Poisson process the interspike distances are exponentially distributed. Hence, for $k = 2$, we sum up two exponentially distributed intervals, for $k = 3$ three intervals and so on. However, summing up exponentially distributed random variables leads to a Gamma-distributed random variable. Therefore, the Poissonian k-th neighbour distance distributions are Gamma distributions. Due to the refractory period, BM-generated noise prefers a distance of multiples of the refractory period. This leads to an absence of short noise bursts, i.e., many spikes in a very small time interval, reducing the dynamic range of the free membrane potential.

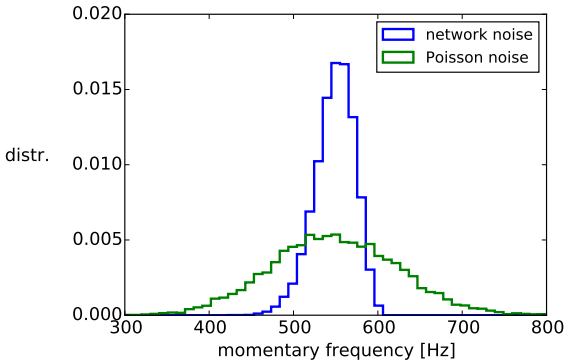

Figure 3.9: Distribution of the momentary noise frequency estimated by binning the noise spike train into intervals of 100ms. If the noise is generated by neurons with a mean frequency close to their maximum frequency (here: $\bar{\nu} = 0.89\ \tau_{\text{ref}}^{-1}$), the distribution becomes skewed.

3.2 LIF Sampling with Boltzmann Machine Noise

As was shown in the previous section, there are several differences between ideal Poisson sources and noise generated by neurons of network BMs. Nevertheless, we have also seen that the resulting activation functions can still be well-fitted by logistic functions, only with very different fit parameters. Therefore, before sampling, the neurons of the sampling BM have to be calibrated to their individual noise sources (with a long calibration time of $5 \cdot 10^5$ms in order to capture the full statistics of the process, see Fig. 3.5). Then, the parameters of the calibration can be used to translate the theoretical weights and biases for each neuron individually. The target Boltzmann parameters are drawn from beta distributions as follows:

$$W \propto W_0 \cdot \big(\text{beta}(0.5,\ 0.5) - 0.5\big)\,, \tag{3.14a}$$

$$b \propto 1.2 \cdot \big(\text{beta}(0.5,\ 0.5) - 0.5\big)\,, \tag{3.14b}$$

with a weight scaling factor W_0 that determines the maximal allowed weights. The beta distribution has been chosen because it can be parametrized to provide large but still limited weights (in contrast to, e.g., a Gaussian) leading to more irregular Boltzmann distributions. Therefore, we can avoid extreme weights and at the same time easily change the overall structure of the distribution by adjusting W_0. Illustrations of the beta distribution can be found in App. 8.5.

The extremely regular spiking of noise neurons discussed in the last section, which was mostly responsible for systematic deviations from the Poisson-stimulated case, is actually rather untypical. In fact, since each noise-generating BM is also sampling from a target distribution, their weights and biases will be drawn from beta distributions as well. This mimics the fact that in the end, all BMs will be performing some specific sampling task and are in no way optimized to provide noise. The autocorrelation functions of noise spike trains generated by randomly initialized BMs can be found in App. 8.8.4.

The simulation setup is the following: A 6-neuron BM is fed with noise coming from a network of 3-neuron BMs. The weights and biases of the noise-generating BMs are all drawn individually from beta distributions. In each simulation, the generated noise was used to sample from 24 target distributions with parameters drawn from a specific beta distribution. Furthermore, each simulation was repeated 10 times with different random seeds in order to take into account different realizations of the noise. For comparison, identical simulations were done using Poisson noise.

As can be seen in Fig. 3.10, if we do not use the BM noise sources for calibration, but equivalent Poisson noise instead, the sampling quality is worse than the ideal one. This is a systematic error, as the activation function used to translate weights and biases is not the one actually observed while sampling with BM-generated noise.

If the calibrations are done correctly, sampling with noise generated from random BMs comes very close to the quality of LIF sampling with Poisson sources, as shown in Fig. 3.11 and 3.12. Even more so, for large weights it becomes slightly better. However,

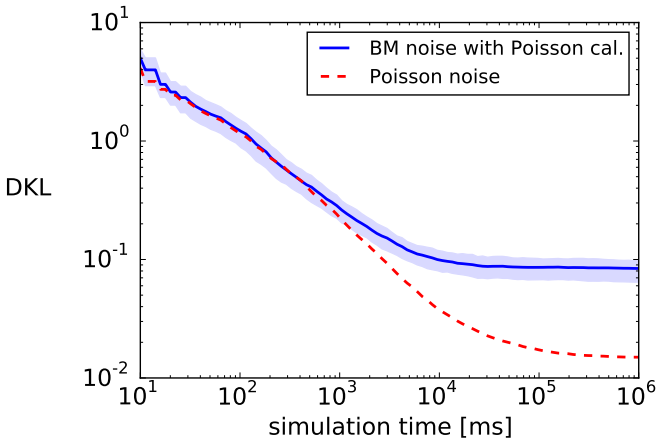

Figure 3.10: Mean D_{KL} over time for a BM sampling from different Boltzmann distributions with $W_0 = 2.4$. The noise-generating BMs used $W_0^{\mathrm{noise}} = 1.2$. If the neurons of the sampling BM are calibrated with equivalent Poisson noise, the final D_{KL} is around one order of magnitude higher than the D_{KL} obtained with Poisson sources. The shaded area marks the interval between the 15th and 85th percentile (see App. 8.2).

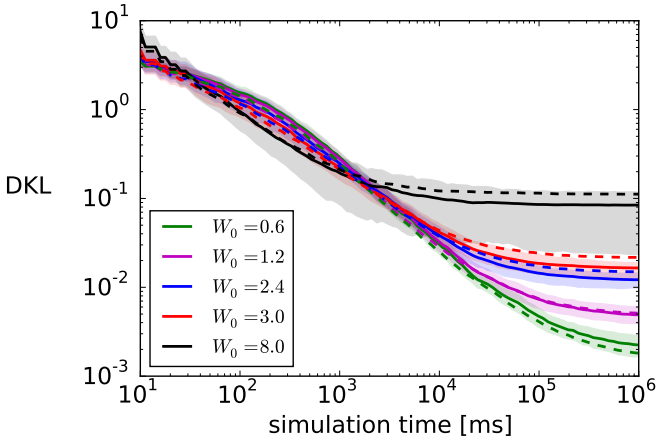

Figure 3.11: Mean D_{KL} over time for a network sampling from different Boltzmann distributions with different weight scaling W_0. The noise-generating BMs have $W_0^{\mathrm{noise}} = 1.2$. The respective case with Poisson noise is included as dashed lines. The shaded areas are bounded by the 15th and 85th percentile. If the neurons of the sampling BM are calibrated with BM-generated noise, the LIF sampling quality is comparable to the Poisson case. For larger weights W_0, the $D_{\mathrm{KL}}^{\mathrm{BM}}$ is even better than the $D_{\mathrm{KL}}^{\mathrm{Poisson}}$. However, this difference is only very small and does not increase for extremely large weights.

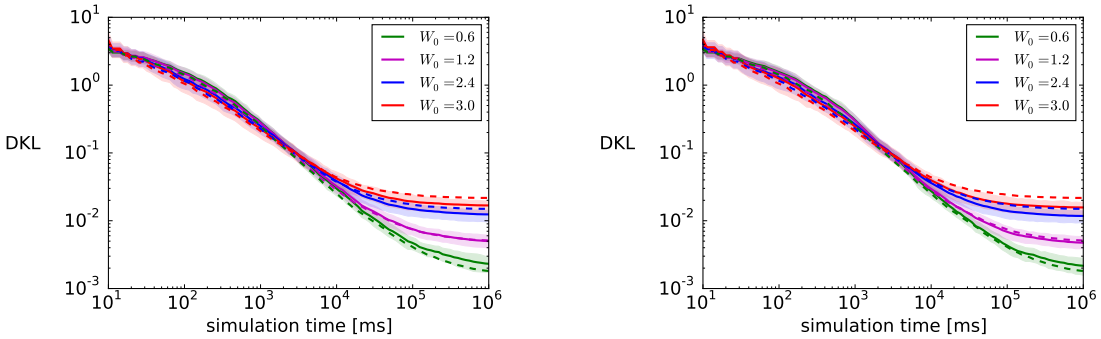

Figure 3.12: Same setup as in Fig. 3.11, but with (left) $W_0^{\text{noise}} = 0.6$ and (right) $W_0^{\text{noise}} = 2.4$. The obtained D_{KL} curves are all very similar (note the logarithmic ordinate), which is not surprising since scaling up the weights of the noise-generating BMs has a negligible effect on the autocorrelation function of the noise spike trains.

as for Poisson noise, the quality gets rapidly worse when increasing the weight scaling factor (due to PSPs not being rectangular). Because the difference in D_{KL} between the case with BM-generated noise and Poisson noise is relatively small and since the final setup will be completely different from the idealized setup studied here, this has not been followed up any further. Interestingly, changing the weight scale factor for the noise-generating BMs does not alter the sampling quality, as shown in Fig. 3.12. This result is expected since changing the absolute value of the weights has only a small effect on the autocorrelations of the noise spike trains (see App. 8.8.4).

It is important to note that a D_{KL} of 10^{-1} does not mean that the network performs bad at sampling on an absolute scale. To provide an illustration of the involved D_{KL} values, bar plots of exemplary distributions used in Fig. 3.11 are presented in Fig. 3.13. It can be seen that even for rather extreme weight distributions, the LIF networks fed with input from noise-generating BMs still capture the main modes of the target distribution very well.

To summarize, there are several differences between Poisson noise and noise generated from Boltzmann machines. First of all, the BM-generated noise spike trains are autocorrelated due to the refractory mechanism of LIF neurons. This leads to a reduced width of the free membrane potential. Furthermore, the membrane autocorrelation is different from the one obtained by a pure OU process because we are using colored (autocorrelated) noise instead of white (uncorrelated) noise.

This also means that the activation function obtained from BM-generated noise and equivalent Poisson noise are different. However, when calibrating each neuron with spike trains from the actually used noise sources, LIF sampling with noise coming from BMs can be successfully implemented without losing sampling quality. Moreover, the sampling

Figure 3.13: Illustration of exemplary distributions that were sampled from. The weights of the noise-generating BMs were scaled with $W_0^{\text{noise}} = 1.2$. The sampled distributions shown here were randomly generated with (top) $W_0 = 0.6$, (middle) $W_0 = 2.4$ and (bottom) $W_0 = 8.0$. Each pair of bars represents the probability of a network state, once given by the sampled distribution and once by the theoretical target distribution. Note that the probabilities here are shown on a log-scale. For higher weight scale factors, the theoretical target distribution becomes more irregular and the differences between sampled and target distribution increase. However, the network continues to sample consistently from the highest-mass probability modes.

quality of ideal LIF sampling with Poisson sources is achieved over a very large range of biases of the noise neurons (for instance, even in the $\bar{\nu} = 0.89\ \tau_{\text{ref}}^{-1}$ case of the previous sections), as demonstrated in App. 8.8.3.

4 Using Correlated Spike Trains from the Sea of Boltzmann Machines

In the previous chapter, a BM was driven by noise coming from a pool of independent BMs instead of ideal Poisson sources. However, to avoid unnecessary cross-correlations, only one neuron out of each of these BMs contributed to the noise spike trains.

This restriction will now be softened by allowing two neurons from each BM to contribute. Such neuron pairs are connected via a synaptic weight, correlating their spike times and introducing additional correlations into the noise spike trains. The two setups investigated in the following sections are shown in Fig. 4.1.

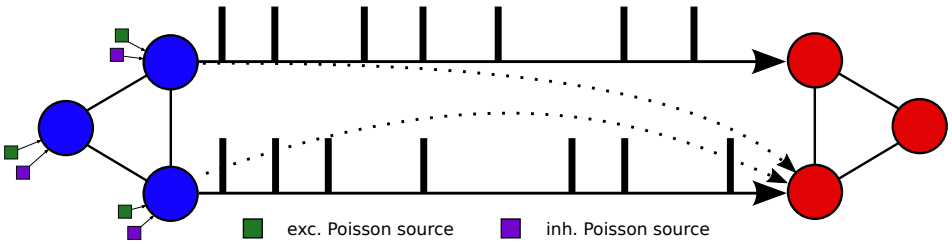

Figure 4.1: Schematic of the setups investigated in this chapter. Instead of using just one neuron from every sea BM as in the previous chapter, a second one is now added. The spike trains can either be connected to two different neurons of the red BM or to a single neuron (dotted line). In both cases, since the blue neurons are connected via weights, correlations are introduced between the noise spike trains. For example, the spike trains depicted here are positively correlated, i.e., the two blue neurons prefer spiking synchronously. Again, none of the red neurons are receiving Poisson input.

If neurons of a BM are driven by cross-correlated noise (as would, in general, be the case for noise neurons that are themselves interconnected), the probability distribution the BM is sampling from becomes distorted. The reason for this is that the additional correlations change the degree of synchronous and asynchronous spiking of neurons in the BM, which is normally defined by the synaptic weights alone (which correspond to Boltzmann weights).

Throughout the next sections, it will be demonstrated how the effect of noise correlations can be reduced. This will enable us to use all neurons in the pool of noise-generating BMs and is a necessary step before introducing interconnections between sea BMs in the next chapter.

4.1 Merging Correlated Spike Trains

First, we will look at a slightly modified version of the uncorrelated case discussed in the previous chapter. Instead of just taking the spike train of one neuron per noise-generating BM, we now take two and merge them. This way, the cross-correlation function of both spike trains will also contribute to the final autocorrelation of the merged spike train.

Of course, there is also the possibility of connecting the two correlated spike trains to the same neuron but with different synapse types. However, this will only have a negligible effect on the noise quality. For instance, if the two spike trains are positively correlated, the sampling neuron will have a higher chance of receiving an excitatory and an inhibitory spike at approximately the same time. Thus, the net effect of the noise on the membrane potential will just cancel, which is equivalent to a reduction of the effective noise frequency. In Chap. 4.2.1, it will be shown that such correlations can be compensated by additionally connecting negatively correlated spike trains, which counter the effects of the positively correlated ones. The setups are illustrated in Fig. 4.2.

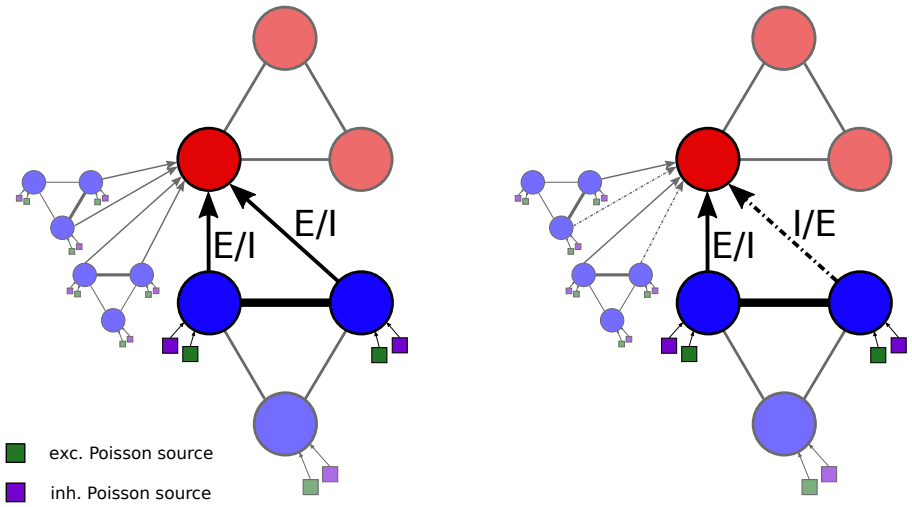

Figure 4.2: Instead of using just one neuron from every sea BM (blue), each neuron of the sampling BM (red) gets two spike trains from every BM. These can either be merged, i.e., both spike trains are either connected to the excitatory (E) or inhibitory (I) synapse (left), or they are connected to different synapses, i.e., one excitatory and the other one inhibitory (right). Every red neuron gets spike trains from many noise-generating BMs.

The autocorrelation function of a spike train that was merged together from two separate spike trains can be calculated analytically. Firstly, let us call the binned version of the spike trains $s^1(t)$ and $s^2(t)$. Merging spike trains is then equivalent to adding up the binned versions $n(t) = s^1(t) + s^2(t)$. This also means that the mean rates simply add up, i.e., $\langle n \rangle = \langle s^1 \rangle + \langle s^2 \rangle$. In the following, we drop the explicit time dependence of the spike trains for clarity. The autocorrelation function of the resulting spike train n is given by:

$$\mathrm{corr}(n,n)_k \cdot \mathrm{Z} = \sum_i \big(n_i - \langle n \rangle\big)\big(n_{i+k} - \langle n \rangle\big)\,, \tag{4.1}$$

with a normalization constant $\mathrm{Z} = \mathrm{Var}(n)$ that will be determined later. Using $n = s^1 + s^2$, this can be rewritten:

$$\begin{aligned}
&\sum_i \big(n_i - \langle n \rangle\big)\big(n_{i+k} - \langle n \rangle\big) \\
=&\sum_i \big(s_i^1 + s_i^2 - \langle s^1 \rangle - \langle s^2 \rangle\big)\big(s_{i+k}^1 + s_{i+k}^2 - \langle s^1 \rangle - \langle s^2 \rangle\big) && \text{(4.2a)} \\
=&\sum_i \big(s_i^1 - \langle s^1 \rangle\big)\big(s_{i+k}^1 - \langle s^1 \rangle\big) + \sum_i \big(s_i^2 - \langle s^2 \rangle\big)\big(s_{i+k}^2 - \langle s^2 \rangle\big) && \text{(4.2b)} \\
+&\sum_i \big(s_i^1 - \langle s^1 \rangle\big)\big(s_{i+k}^2 - \langle s^2 \rangle\big) + \sum_i \big(s_i^2 - \langle s^2 \rangle\big)\big(s_{i+k}^1 - \langle s^1 \rangle\big) \\
=&\ \mathrm{corr}(s^1,s^1)_k\ \mathrm{Var}(s^1) + \mathrm{corr}(s^2,s^2)_k\ \mathrm{Var}(s^2) && \text{(4.2c)} \\
&+ \big(\mathrm{corr}(s^1,s^2)_k + \mathrm{corr}(s^2,s^1)_k\big)\sqrt{\mathrm{Var}(s^1)\mathrm{Var}(s^2)}\,,
\end{aligned}$$

where we used the definition of the cross-correlation function, a generalization of the autocorrelation function defined in Eq. 3.1:

$$\mathrm{corr}(s^1,s^2)_k = \frac{\sum_i \big(s_i^1 - \langle s^1 \rangle\big)\big(s_{i+k}^2 - \langle s^2 \rangle\big)}{\sqrt{\mathrm{Var}(s^1)\mathrm{Var}(s^2)}}\,. \tag{4.3}$$

The normalization Z can be found by using the condition $\mathrm{corr}(n,n)_0 \stackrel{!}{=} 1$, which has to be fulfilled by every autocorrelation function:

$$\begin{aligned}
\mathrm{Z} &= \mathrm{Var}(s^1) + \mathrm{Var}(s^2) + \big(\mathrm{corr}(s^1,s^2)_0 + \mathrm{corr}(s^2,s^1)_0\big)\sqrt{\mathrm{Var}(s^1)\mathrm{Var}(s^2)} \\
&= \mathrm{Var}(s^1) + \mathrm{Var}(s^2) + 2\ \mathrm{cov}(s^1,s^2)\sqrt{\mathrm{Var}(s^1)\mathrm{Var}(s^2)}\,,
\end{aligned} \tag{4.4}$$

where the *covariance* $\mathrm{cov}(s^1,s^2) = \mathrm{corr}(s^1,s^2)_0 = \mathrm{corr}(s^2,s^1)_0$ was introduced. Thus, the autocorrelation function of n can be rewritten in terms of the autocorrelation and cross-correlation functions of s^1 and s^2:

$$\begin{aligned}\mathrm{corr}(n,n)_k = \mathrm{Z}^{-1}\Big(&\ \mathrm{corr}(s^1,s^1)_k\ \mathrm{Var}(s^1) + \mathrm{corr}(s^2,s^2)_k\ \mathrm{Var}(s^2) \\ &+ \big(\mathrm{corr}(s^1,s^2)_k + \mathrm{corr}(s^2,s^1)_k\big)\sqrt{\mathrm{Var}(s^1)\mathrm{Var}(s^2)}\Big)\end{aligned} \tag{4.5}$$

and in the special case of $\mathrm{Var}(s^1) = \mathrm{Var}(s^2)$:

$$\mathrm{corr}(n,n)_k = \frac{\mathrm{corr}(s^1,s^1)_k + \mathrm{corr}(s^2,s^2)_k + \mathrm{corr}(s^1,s^2)_k + \mathrm{corr}(s^2,s^1)_k}{2 + 2\ \mathrm{cov}(s^1,s^2)}\,. \tag{4.6}$$

Now, if s^1 and s^2 are not correlated, the correlation and covariance terms are zero and the resulting autocorrelation of the merged spike train n is simply the weighted mean of both. However, existing correlations either lead to an increase or decrease of autocorrelation as demonstrated in Fig. 4.3.

Also note that the correlation terms for non-zero delay (numerator) as well as the (co)variance terms for zero delay (denominator) are averaged with the corresponding autocorrelations. Hence, if the spike trains are already strongly autocorrelated, positive cross-correlations will only slightly affect the final autocorrelation.

Since the binning of spike trains leads to strong artifacts in the calculated correlation functions, the binned spike trains were first smoothed with a Gaussian kernel. For example, choosing the bin width too small leads to an underestimation of negative correlations. But for larger bin widths, strong oscillations to negative values occur in strictly positive correlation functions. Gaussian smoothing not only removes these artifacts, it also allows us to use a consistent and comparable method to calculate correlations. This approach is a common one and was, for example, also used in *Breitwieser* (2011); *Petrovici et al.* (2014); *Stöckel* (2015). Thus, the spike trains were first binned with a bin width $\Delta = 1\mathrm{ms}$ and further convolved with a Gaussian $\exp\big(-\frac{t^2}{2\sigma^2}\big)$ with width $\sigma = 5\mathrm{ms}$ before calculating the correlation function. An example of the binned spike train and its smoothed version can be found in Fig. 4.4.

Because adding up correlated spike trains does not increase the already present autocorrelation by a significant amount, the quality of LIF sampling should not be negatively affected. To test this, several setups were used. First, the case where both spike trains are connected to the same synapse was investigated (Fig. 4.2, left) for three different cases:

1. The weights between the noise-providing neurons are drawn from a symmetric beta distribution, $W^{\mathrm{noise}} \propto 6 \cdot \big(\mathrm{beta}(0.5,\ 0.5) - 0.5\big)$.

2. The weights between the noise-providing neurons are drawn from an asymmetric beta distribution to provide mainly positive weights, $W^{\mathrm{noise}} \propto 6 \cdot \big(\mathrm{beta}(5,\ 0.5) - 0.5\big)$.

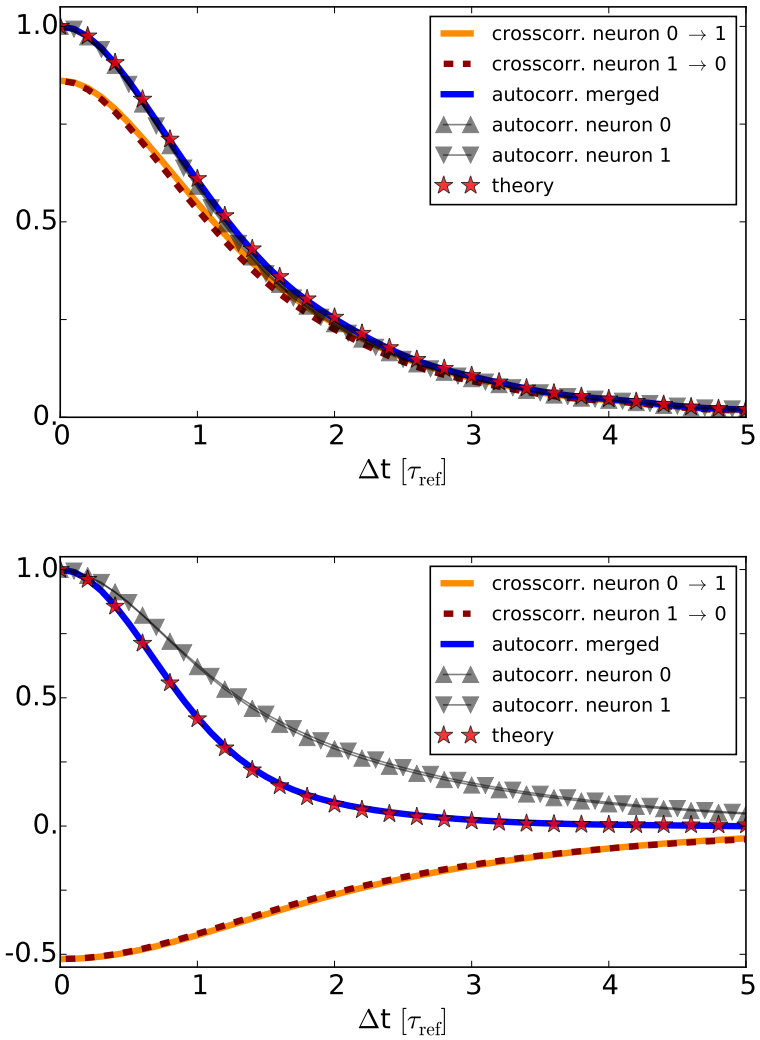

Figure 4.3: Spike train correlation functions of two neurons and their merged spike train, compared with the theoretical result. The spike trains have first been binned and afterwards smoothed with a Gaussian kernel. Both neurons are connected with a weight drawn from an asymmetric beta distribution with strong positive (top) and negative (bottom) weights. This leads either to strong positive (top) or negative (bottom) cross-correlations, shown in orange and dark red. Furthermore, both neurons have finite autocorrelations due to the refractory mechanism (gray triangles). Using Eq. 4.5, the theoretical autocorrelation obtained from merging both spike trains is given by the red stars and agrees very well with the experimentally obtained one (blue). **(top)** Note that, even though we have strong auto - and cross-correlations, the merged spike train has almost the same autocorrelation function as the initial individual spike trains. **(bottom)** A negative cross-correlation reduces autocorrelation after merging the individual spike trains.

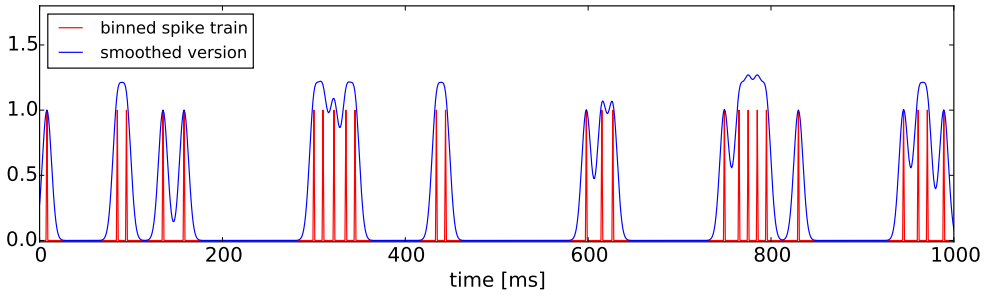

Figure 4.4: In red, the binned version of a spike train is shown, which is discontinuous whenever a spike occurs. Convolving the binned spike train with a Gaussian kernel smooths out the abrupt changes and makes it easier to calculate high-resolution correlation functions between spike trains without any disturbing artifacts (e.g. oscillations).

3. The weights between the noise-providing neurons are drawn from an asymmetric beta distribution to provide mainly negative weights, $W^{\text{noise}} \propto 6 \cdot \big(\text{beta}(0.5,\ 5) - 0.5\big)$.

In all cases, the biases were drawn from a beta distribution $b^{\text{noise}} \propto 1.2 \cdot \big(\text{beta}(0.5,\ 0.5) - 0.5\big)$. The number of used sea BMs was automatically adjusted to guarantee a minimum noise frequency of around 800Hz. Case 2. and 3. are limiting cases where the noise neurons are either strongly positively or negatively correlated. This either leads to an increase or decrease in the autocorrelation function of the resulting noise spike trains. Case 1. mixes spike trains coming from positively and negatively correlated neurons and is therefore not a corner case.

Secondly, a general setup was investigated where the spike train of each neuron is connected randomly inhibitorily or excitatorily with equal probability (i.e., a mix of the two cases presented in Fig. 4.2). The results of sampling from 24 Boltzmann distributions drawn from

$$W \propto 2.4 \cdot \big(\text{beta}(0.5,\ 0.5) - 0.5\big)\,, \tag{4.7a}$$

$$b \propto 1.2 \cdot \big(\text{beta}(0.5,\ 0.5) - 0.5\big)\,, \tag{4.7b}$$

and repeating the whole experiment 20 times with different random seeds is shown in Fig. 4.5. Note that each neuron is again calibrated to its respective noise source, i.e., the noise-generating BMs. As expected, the LIF sampling quality is in all cases close to the ideal case with Poisson sources. Hence, feeding a single neuron with correlated spike trains does not negatively affect LIF sampling.

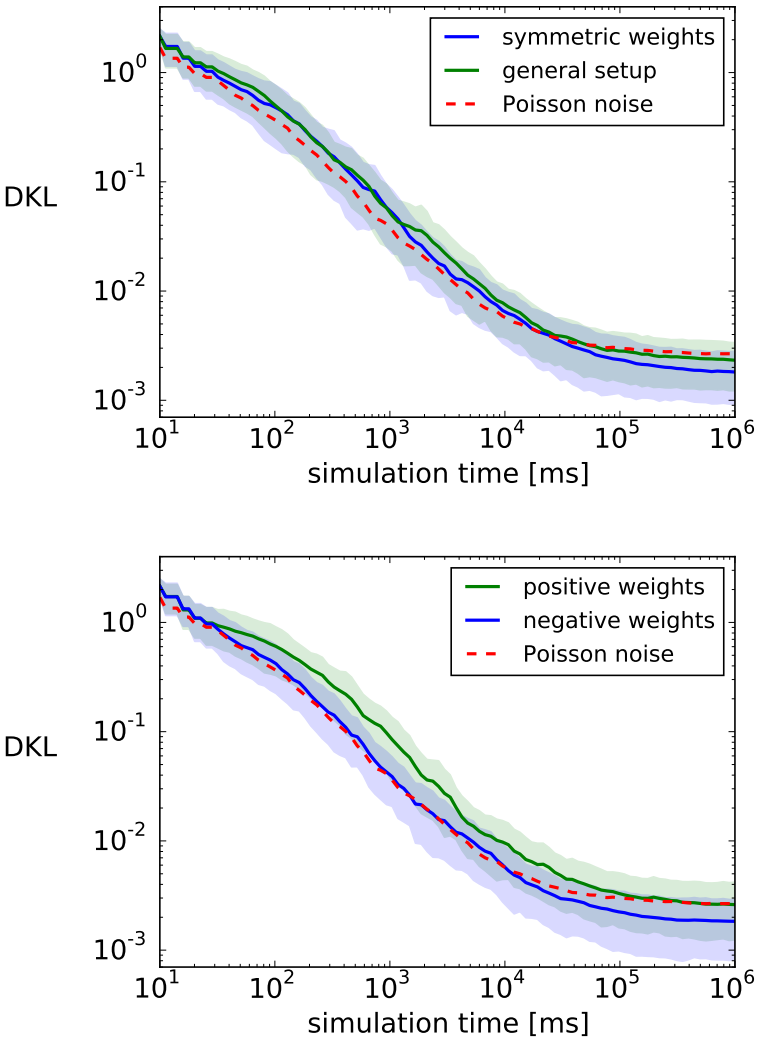

Figure 4.5: Mean D_{KL} after sampling from 24 different Boltzmann distributions with single neurons being fed by correlated spike trains. The experiment was repeated 20 times. The shaded areas mark the interval between the 15th and 85th percentile over all D_{KL} curves. In all cases, the sampling quality is in the same range as the ideal Poissonian one. **(top)** Here, two cases are presented: (blue) The weights of the noise-generating BMs are drawn from a symmetric beta distribution and spike trains of correlated neuron pairs are connected to the same synapse. This is the previously mentioned case 1. (green) The general setup mentioned earlier. Again, weights are drawn from a symmetric distribution, but each spike train is connected with a random synapse type. **(bottom)** Similar to case 1, but once with strong positive and once with strong negative weights (cases 2. and 3. discussed previously).

4.2 Distributing Correlated Spike Trains

Instead of connecting noise spike trains coming from the same BM to a single neuron, we can connect them to neighbouring neurons. This will introduce additional correlations between neurons of the sampling BM.

The setup is as follows: A sampling BM with three neurons receives its noise from a pool of Poisson-driven BMs. Two neurons of the sampling BM get strongly correlated noise, i.e., each noise-generating BM provides its two strongest correlated spike trains as noise. The third neuron gets its noise from independent BMs as in Chapter 3. Whether a spike train is connected inhibitorily or excitatorily is chosen randomly with equal probability.

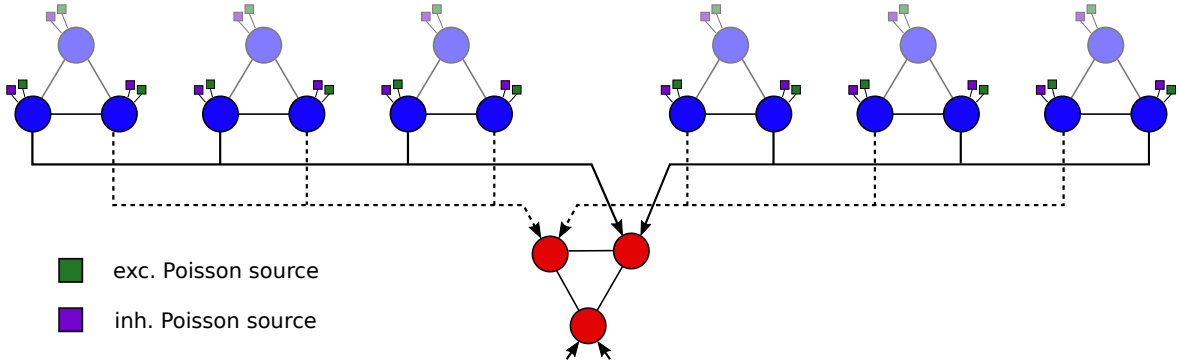

Figure 4.6: The setup investigated in this section. Two of the sampling BM neurons get their noise from a common pool of network BMs and the third one gets independent noise as in Chapter 3. This introduces correlations between the noise spike trains of the top two red neurons.

There are two ways to reduce correlations between noise spike trains. Since several neurons from different noise-generating BMs contribute to each noise spike train, we can simply merge positively and negatively correlated pairs such that the resulting correlation will average out to zero. In this scenario, the noise input itself will be uncorrelated because positive and negative contributions, coming from neurons connected via positive and negative synaptic weights, cancel each other out.

Furthermore, whether the spike trains coming from two correlated neurons are both connected to the same or different synapse types changes the resulting correlation of the receiving neurons. For example, if the two neurons produce positively correlated spike trains, connecting these with a symmetric pattern, i.e., both excitatory or both inhibitory, leads to positive correlations between the two noise-driven neurons since common input facilitates synchronous spiking. However, using an asymmetric pattern, i.e., one excitatory/inhibitory and the other one inhibitory/excitatory, the former positive correlation is translated to a negative one since one neuron gets excitatory input when the other one gets inhibitory input, leading to a preference for non-synchronous spiking.

Now, even if the input noise spike trains are strongly correlated, the membrane dynamics of the sampling BM neurons can be decorrelated by randomly choosing the synapse types.

In the following sections, these cases will be discussed in more detail.

4.2.1 Mixing Input Correlations

The first way to use correlated spike trains to obtain uncorrelated input noise is by choosing the weights of the noise-generating BMs randomly. The procedure works in the following way: We have several 3-neuron BMs that are still driven by Poisson noise. In each of these, we have one strong synaptic weight and two weak synaptic weights that are randomly drawn from a beta distribution

$$W_{\text{strong}}^{\text{noise}} \propto 4.0 \cdot \big(\text{beta}(0.5,\ 0.5) - 0.5\big)\,, \tag{4.8a}$$
$$W_{\text{weak}}^{\text{noise}} \propto 1.0 \cdot \big(\text{beta}(5.0,\ 5.0) - 0.5\big)\,. \tag{4.8b}$$

The strong weights are chosen to guarantee strong correlations such that we can study the decorrelation mechanism presented here under extreme conditions. From each noise-generating BM, the spike trains of the two neurons connected by the strong weight are fed into the neurons with the largest absolute weight of the sampling BM. This is repeated for all noise-generating BMs (see Fig. 4.7). The third neuron of the sampling BM receives its noise from independent noise-generating BMs as in Chapter 3.

In fact, the resulting correlation function between two spike trains $(n^1,\ n^2)$ that were merged together from two sets of correlated spike trains $(s^1,\ s^2)$ and $(t^1,\ t^2)$ can be easily derived. First, the correlation function with normalization constant Z is given by:

$$\text{corr}(n^1, n^2)_k \cdot \text{Z} = \sum_i \big(n_i^1 - \langle n^1 \rangle\big)\big(n_{i+k}^2 - \langle n^2 \rangle\big)\,. \tag{4.9}$$

As in the previous section, n^1 is the sum of the binned spike trains s^1 and t^1, i.e., $n^1 = s^1 + t^1$ and similarly $n^2 = s^2 + t^2$. Inserting these, we get

$$\begin{aligned}
&\sum_i \big(n_i^1 - \langle n^1 \rangle\big)\big(n_{i+k}^2 - \langle n^2 \rangle\big) \\
=&\sum_i n_i^1 n_{i+k}^2 - \sum_i n_i^1 \langle n^2 \rangle - \sum_i n_{i+k}^2 \langle n^1 \rangle + \sum_i \langle n^1 \rangle \langle n^2 \rangle && \text{(4.10a)} \\
=&\sum_i \big(s_i^1 + t_i^1\big)\big(s_{i+k}^2 + t_{i+k}^2\big) - \sum_i \big(s_i^1 + t_i^1\big)\big(\langle s^2 \rangle + \langle t^2 \rangle\big) && \text{(4.10b)} \\
&- \sum_i \big(s_{i+k}^2 + t_{i+k}^2\big)\big(\langle s^1 \rangle + \langle t^1 \rangle\big) + \sum_i \big(\langle s^1 \rangle + \langle t^1 \rangle\big)\big(\langle s^2 \rangle + \langle t^2 \rangle\big)
\end{aligned}$$

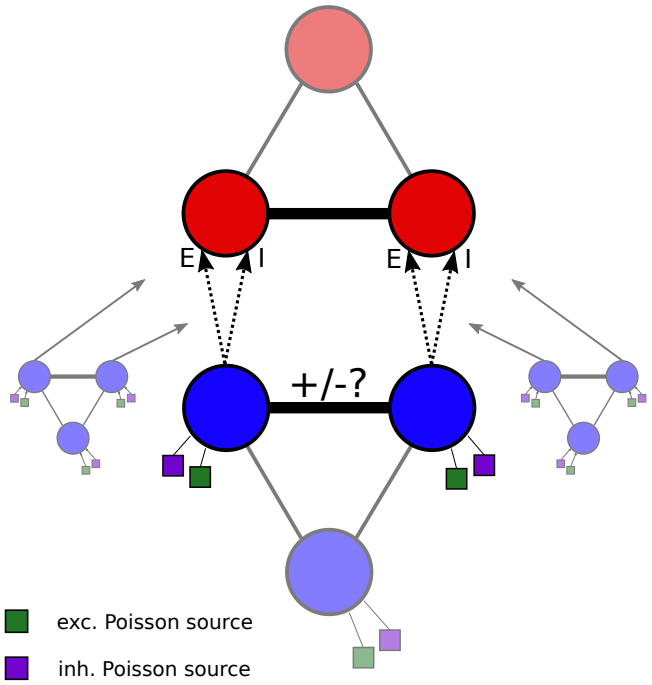

Figure 4.7: The two neurons with the largest absolute weight of the red BM used for sampling get correlated noise from the blue BMs. Whether a spike train is connected inhibitory or excitatory is decided randomly with equal probability. The weights of the blue neurons are chosen randomly as well but from a single beta distribution to assert extreme values. Choosing the weights randomly and using spike trains from different, independent noise-generating BMs will lead to a cancellation of correlations in the resulting overall noise.

$$= \sum_i \left(s_i^1 - \langle s^1 \rangle\right)\left(s_{i+k}^2 - \langle s^2 \rangle\right) + \sum_i \left(s_i^1 - \langle s^1 \rangle\right)\left(t_{i+k}^2 - \langle t^2 \rangle\right) \tag{4.10c}$$
$$+ \sum_i \left(t_i^1 - \langle t^1 \rangle\right)\left(s_{i+k}^2 - \langle s^2 \rangle\right) + \sum_i \left(\langle t_i^1 \rangle - \langle t^1 \rangle\right)\left(\langle t_{i+k}^2 \rangle + \langle t^2 \rangle\right)$$
$$= \mathrm{corr}(s^1, s^2)_k \sqrt{\mathrm{Var}(s^1)\mathrm{Var}(s^2)} + \mathrm{corr}(s^1, t^2)_k \sqrt{\mathrm{Var}(s^1)\mathrm{Var}(t^2)} \tag{4.10d}$$
$$+ \mathrm{corr}(t^1, s^2)_k \sqrt{\mathrm{Var}(t^1)\mathrm{Var}(s^2)} + \mathrm{corr}(t^1, t^2)_k \sqrt{\mathrm{Var}(t^1)\mathrm{Var}(t^2)}$$

after reordering terms. The normalization Z is given by

$$\mathrm{Z} = \sqrt{\mathrm{Var}(n^1)\mathrm{Var}(n^2)} \tag{4.11}$$

and the variation of e.g. n^1, with N the number of bins, can be rewritten as follows:

$$\begin{aligned}
\mathrm{Var}(n^1) &= \frac{1}{N}\sum_i \left(n_i^1 - \langle n^1\rangle\right)^2 && \text{(4.12a)}\\
&= \frac{1}{N}\sum_i \left(s_i^1 - \langle s^1\rangle + t_i^1 - \langle t^1\rangle\right)^2 && \text{(4.12b)}\\
&= \frac{1}{N}\sum_i \left(s_i^1 - \langle s^1\rangle\right)^2 + \frac{1}{N}\sum_i \left(t_i^1 - \langle t^1\rangle\right)^2 && \text{(4.12c)}\\
&+ \frac{2}{N}\sum_i \left(s_i^1 - \langle s^1\rangle\right)\left(t_i^1 - \langle t^1\rangle\right) && \text{(4.12d)}\\
&= \mathrm{Var}(s^1) + \mathrm{Var}(t^1) + 2\ \mathrm{cov}(s^1, t^1)\,. && \text{(4.12e)}
\end{aligned}$$

Since s^1 and t^1 as well as s^2 and t^2 are uncorrelated, all correlation or covariance terms between them vanish. Thus, the correlation function of n^1 and n^2 finally takes the following form:

$$\mathrm{corr}(n^1, n^2)_k = \frac{\mathrm{corr}(s^1, s^2)_k\sqrt{\mathrm{Var}(s^1)\mathrm{Var}(s^2)} + \mathrm{corr}(t^1, t^2)_k\sqrt{\mathrm{Var}(t^1)\mathrm{Var}(t^2)}}{\sqrt{\mathrm{Var}(s^1) + \mathrm{Var}(t^1)}\sqrt{\mathrm{Var}(s^2) + \mathrm{Var}(t^2)}} \tag{4.13}$$

and for equal variances

$$\mathrm{corr}(n^1, n^2)_k = \frac{\mathrm{corr}(s^1, s^2)_k + \mathrm{corr}(t^1, t^2)_k}{2}\,. \tag{4.14}$$

The final cross-correlation of the two resulting spike trains is hence approximately given by the average correlation of the individual spike trains. This is illustrated in Fig. 4.8, where the theoretical result is compared with simulation results.

Adding even more correlated spike trains will simply increase the terms included in the average. Hence, adding up many of these spike trains with random correlations drawn from identical but independent distributions, the *central limit theorem* tells us that the final cross-correlation of the resulting spike trains will be Gaussian-distributed. Since the weight distributions are symmetric around zero, i.e., their mean is zero, the average value of the Gaussian will also be zero. Furthermore, according to the central limit theorem the width of the resulting Gaussian distribution scales reciprocal to the square root of the number of added spike trains. Thus, if the weights of the noise-generating BMs are drawn from a distribution with zero mean, we can diminish the correlation of the input noise by increasing the number of sea BMs. In the limit of infinitely many, the input will be completely uncorrelated. In Fig. 4.9, the distributions of final correlations are shown for different numbers of noise-generating BMs.

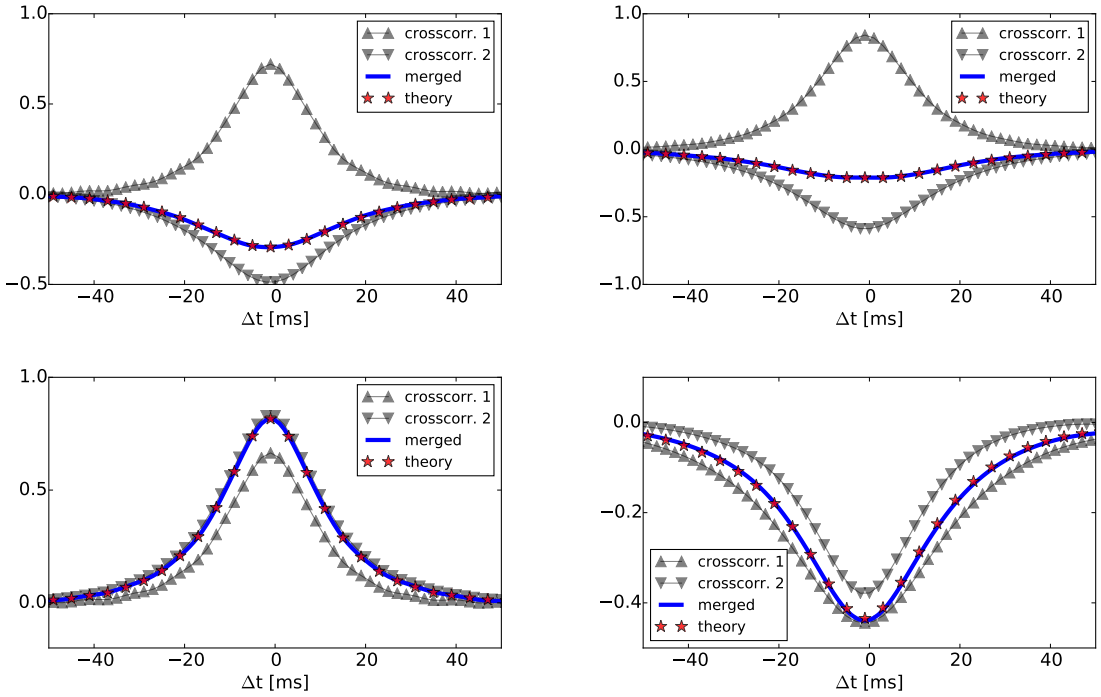

Figure 4.8: Correlation functions of two pairs of spike trains that are merged. Since neurons of a BM are connected via a synaptic weight, their spike trains are correlated. If we take the spike trains from two such neuron pairs, i.e., from the first pair the spike trains (s^1, s^2) and from the second one (t^1, t^2) and merge them to $n^1 = s^1 + t^1$ and $n^2 = s^2 + t^2$, the correlation function of the resulting spike trains (blue) can be calculated from the correlation functions of the individual pairs (gray). Note that this way, adding up negatively and positively correlated spike trains reduces the final correlation (top). Adding up spike trains with similar correlations leads to a weighted average as the final correlation (bottom). The theoretical result (red) is calculated by inserting the individual correlation functions (gray) into Eq. 4.13.

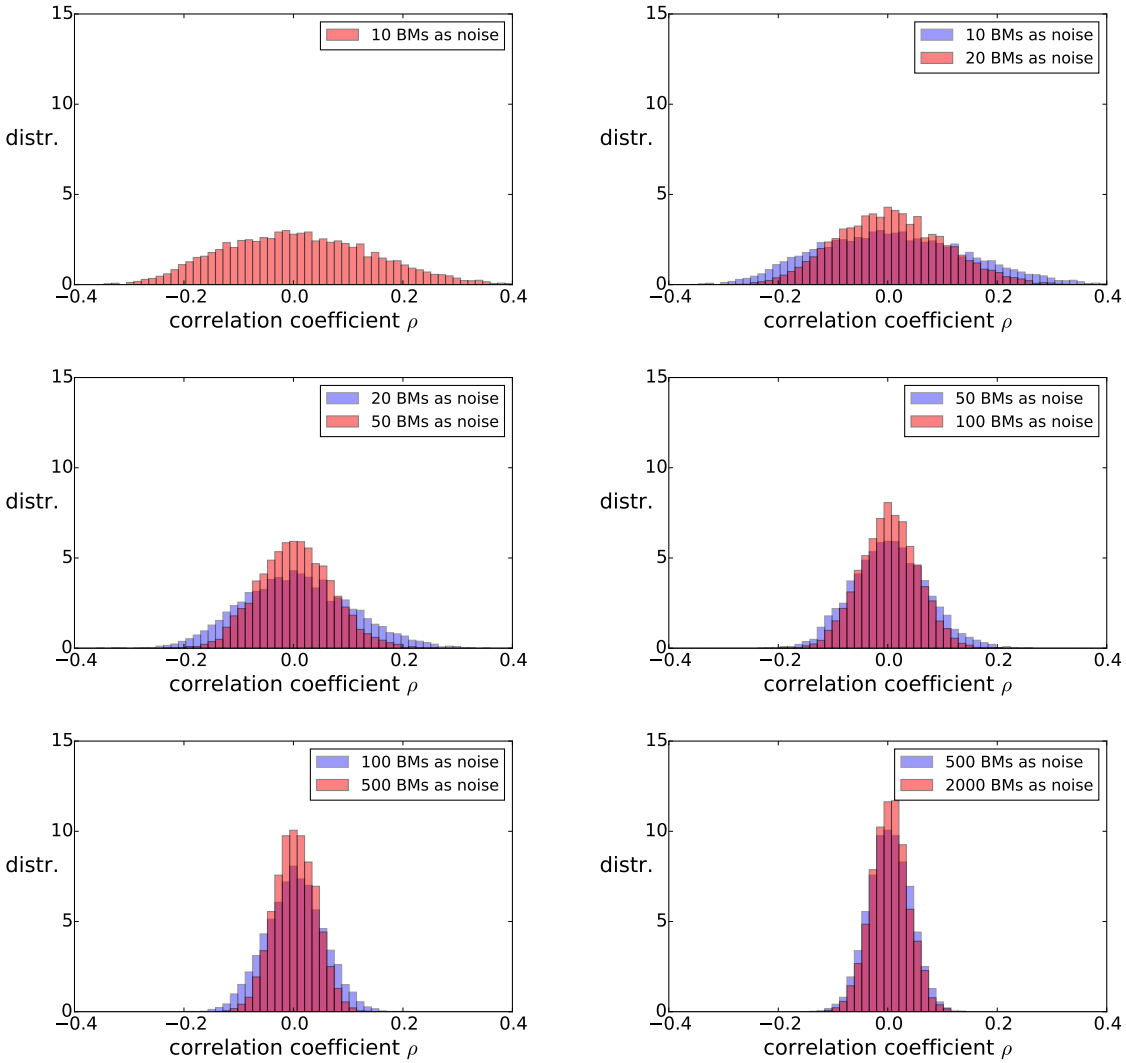

Figure 4.9: Distribution of correlation coefficients between the noise inputs of two connected neurons n_1 and n_2 that receive noise from a sea of BMs. From each noise-generating BM, one spike train is fed to the first neuron n_1 with randomly chosen synapse type and the second spike train to neuron n_2, also with random synapse type. This was repeated 2000 times and plotted in histograms for different numbers of BMs providing noise. In each figure, the previous histogram is included in blue to emphasize that increasing the number of BMs leads to a narrower width of the correlation distribution. This way, the correlations due to synaptic weights in the noise-giving BMs can be reduced by including many positively and negatively correlated spike trains.

4.2.2 Mixing Synapse Types

It turns out that not only the sign of the weights between noise-generating neurons is important, but also the synapse type of connections. We will look at two types of patterns:

1. A symmetric connection pattern, i.e., both spike trains are either connected excitatorily (EE) or both are connected inhibitorily (II).
2. An asymmetric connection pattern, i.e., both spike trains are connected with different synapse types (EI or IE).

The first pattern will result in a direct translation of the input correlation to a similar correlation between the outgoing spikes. Thus, if the incoming spikes are positively correlated, the outgoing spikes will also be positively correlated (and the same for negative correlations). However, the second pattern inverts input correlations, i.e., positive input correlations lead to negative output correlations and vice versa.

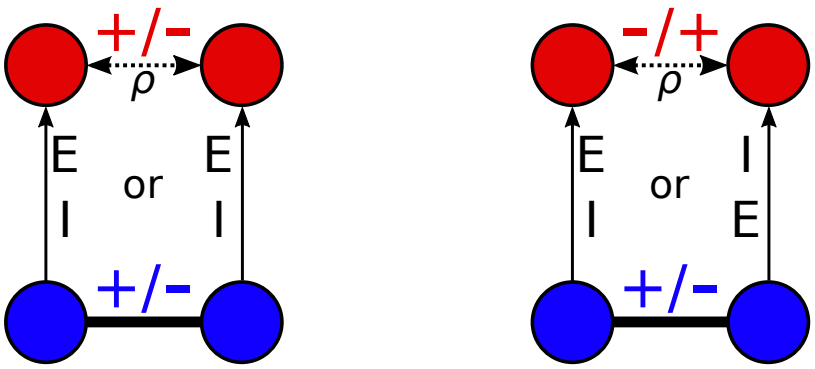

Figure 4.10: Poisson-driven BMs (blue) provide noise to two unconnected neurons (red). **(left)** If the sign of the weight between the blue neurons is fixed and their spike trains are connected with a symmetric pattern to the two red neurons, i.e., both with the same synapse type, then a positive/negative input correlation is translated to a positive/negative correlation ρ between the two red neurons. **(right)** However, if the connection pattern is asymmetric, i.e., both spike trains are connected with different synapse types, then the sign of the correlation is swapped from the blue to the red neurons.

Consequently, even if we only have positive (or negative) input correlations, by adding up spike trains of many sea BMs with randomly chosen connection patterns, we again obtain a sum of positively and negatively correlated spike trains as in the previous section which will average out if the number of noise-generating BMs is large enough. This is demonstrated in Fig. 4.11.

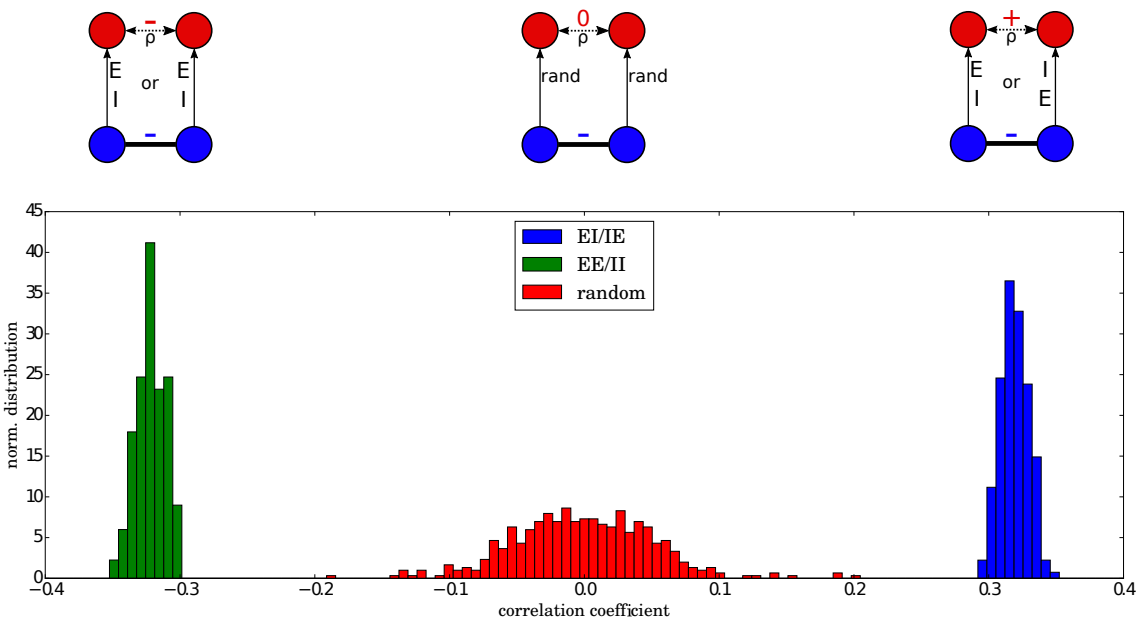

Figure 4.11: Histogram over the final correlation ρ between the two red neurons after receiving noise from the blue BMs with (left) symmetric patterns, (right) asymmetric patterns and (middle) randomly chosen connections of both patterns. For each simulation, the number of noise-generating BMs was automatically chosen to reach a minimum noise frequency of 800Hz. The weight of each noise-generating BM is randomly chosen from a beta distribution $W^{\text{noise}} \propto 6 \cdot \big(\text{beta}(0.5,\ 0.5) - 0.5\big)$. 200 simulation results are included in the histogram. **(left)** Negative input correlations are simply translated to a negatively correlated output. **(right)** Similarly, negative input correlations are reversed to positively correlated output for asymmetric patterns. **(middle)** If each BM is randomly connected with either a symmetric or asymmetric pattern, the positive and negative correlations in the output will cancel out. Of course, the same holds true for positive input correlations. Note that *E* stands for *Excitatory* and *I* for *Inhibitory*.

Note that our previous way of measuring correlations will not work here anymore, since the input noise spikes are still strongly correlated and only become decorrelated due to synaptic interaction. Hence, it is not sufficient to consider the input spike train correlations, but instead we need to look at the correlation in the spiking activity of the neurons that receive these inputs. In order to facilitate this, the (correlated) noise trains are fed to two unconnected neurons, i.e., with synaptic weight $W_{01} = W_{10} = 0$ and $b_0 = b_1 = 0$. This corresponds to sampling from a Boltzmann distribution with probabilities $p_{11} = p_{00} = p_{10} = p_{01} = 0.25$ with indices denoting the state of the two neurons. The correlation coefficient of the states can then be used as a natural way of measuring how correlated the neurons are spiking, taking into account possible decorrelations coming from the synaptic interaction.

The correlation coefficient of the states is given by (*Bytschok*, 2011)

$$\rho = \frac{p_{11} - p_1^1 p_1^2}{\sqrt{\left(p_1^1 - (p_1^1)^2\right)\left(p_1^2 - (p_1^2)^2\right)}} \tag{4.15}$$

with p_1^i being the marginal spiking probability of neuron i.

4.2.3 LIF Sampling with Correlated Noise

As it turns out, if we average over enough input spike trains, the effect of input correlations can be easily reduced. Correlations between spike trains of the same BM are unavoidable as they are imprinted by synaptic connections. However, the spike trains of independent BMs, i.e., of unconnected neurons, are completely uncorrelated. Hence, spike pairs from independent BMs can be mixed and, if we allow both positive and negative weights, this will lead to a cancellation of input correlations.

Additionally, the translated correlation from the input neurons to the states of the output neurons can be modulated by randomly choosing the synapse type for each noise spike train. This way, even though the input is strongly correlated, the output can still be uncorrelated.

In Fig. 4.12, the LIF sampling quality for a BM of 3 neurons is compared for different numbers of noise-generating BMs. Two neurons of the sampling BM receive correlated noise from BMs with weights randomly generated from a symmetric beta distribution $W^{\text{noise}} \propto 6 \cdot \big(\text{beta}(0.5,\ 0.5) - 0.5\big)$. Furthermore, each spike train is connected to a randomly chosen synapse type, so both mechanisms of decorrelation apply. The third neuron receives its noise from a network of independent noise-generating BMs. As expected, the D_{KL} improves if we increase the number of inputs, since input correlations are averaged out.

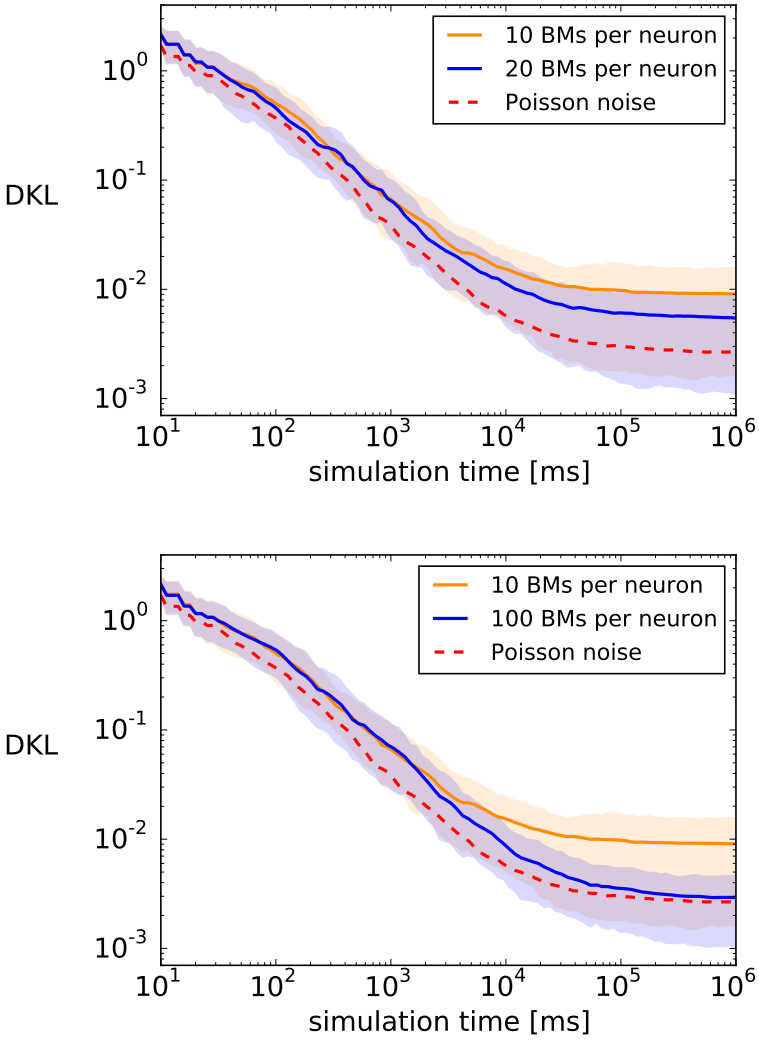

Figure 4.12: D_{KL} for sampling from a 3-neuron BM with correlated noise coming from a sea of BMs. Again, the BM was used to sample from 24 different Boltzmann distributions, all drawn randomly with Eq. 4.7a and 4.7b. The simulation was repeated 20 times for different random seeds. The shaded areas mark the interval between the 15th and 85th percentile over all single D_{KL} curves. For a rather small number of BMs in the sea, the final D_{KL} is worse than for ideal LIF sampling with Poisson sources due to correlations between noise spike trains. However, by increasing the sea size (for instance from 10 to 20 BMs (top) and from 10 to 100 BMs (bottom)), we can again reach similar results as with Poisson noise since possible input correlations are averaged out.

5 Connecting the Sea of Boltzmann Machines to a Large Network

Up until now, the sea BMs were still driven by Poisson noise and only a single BM received noise from the sea instead of Poisson sources. Even though this setup is rather impractical, it gives us some insights into what to expect when exchanging Poisson background with intrinsic network connections in the sea of BMs.

First of all, the case of one BM getting noise from completely independent BMs can be reproduced in the limit of infinitely many BMs ($N \to \infty$), all interconnected with a small connectivity ϵ such that $N \cdot \epsilon \gg 1$ guarantees large enough noise frequencies for the high-conductance state. In Chap. 3 it was demonstrated that a BM driven by independent spike trains coming from other BMs can perform LIF sampling without any restrictions. Thus, if we remove all Poisson sources from the network, each BM will still receive a finite number of completely independent noise inputs coming from the network itself and therefore be able to correctly sample from its target Boltzmann distribution.

However, this limit is not a practical one, since all sensible physical systems are limited in size, from software simulations of neural networks to the brain itself. In the particularly interesting case of neuromorphic hardware, for example, the network's size is in general limited by hardware constraints, as for instance the number of available neurons and synapses. Therefore, the main concern of the following chapter lies in whether the needed Poisson stimuli in a finite-sized network of BMs can be reduced via intrinsic noise coming from within the network itself.

Firstly, we will connect a number of Poisson-driven BMs to a network by introducing interconnections between them. Each BM samples from another distribution with Boltzmann parameters randomly drawn from beta distributions

$$W \propto W_0 \cdot \big(\text{beta}(0.5,\ 0.5) - 0.5\big)\,, \tag{5.1a}$$

$$b \propto 1.2 \cdot \big(\text{beta}(0.5,\ 0.5) - 0.5\big)\,, \tag{5.1b}$$

again with a weight scaling factor W_0. The interconnections W_{int} are also drawn from a modified beta distribution that resembles a bimodal Gaussian – with one mode for negative and one for positive weights. The beta distribution has been chosen to provide a more general setup with randomly selected noise weights that are still bounded, i.e., they cannot be arbitrarily large. Furthermore, a parameter $\eta = \frac{\#\text{inh. conn.}}{\#\text{exc. conn.}}$ determining

the ratio of total inhibitory to excitatory connections as well as a parameter $g = -\frac{\langle w_{\text{inh}} \rangle}{\langle w_{\text{exc}} \rangle}$ for adjusting the ratio of connection strengths was added. The noise weight from neuron i to j is then drawn from

$$W_{\text{int},ij} \propto \Big[\big(\text{beta}(4.0,\ 4.0) - 0.5\big) \cdot 2\sigma_{\text{W}} + \mu_{\text{W}}\Big] \cdot g_{ij} \cdot (-1)^{S_{ij}} \cdot B_{ij} \tag{5.2}$$

with

$$S_{ij} = \text{binomial}\big(1,\ \eta\big)\,, \tag{5.3a}$$

$$g_{ij} = \frac{1}{\max(1,\ g)}\big[\big(S_{ij} \cdot (g-1)\big) + 1\big]\,, \tag{5.3b}$$

$$B_{ij} = \begin{cases} 0 & \textbf{if} \quad i \geq j \quad \textbf{and} \quad i - j < i \bmod N_{\text{neurons}} + 1\,, \\ 0 & \textbf{if} \quad i < j \quad \textbf{and} \quad j - i < j \bmod N_{\text{neurons}} + 1\,, \\ 1 & \textbf{else}, \end{cases} \tag{5.3c}$$

where S_{ij} is 1 with probability η and 0 with probability $1 - \eta$. The S_{ij} factor decides if a connection is inhibitory or excitatory and is further used to automatically rescale the weights with g. For example, setting $\eta = 1$ would result in only inhibitory noise connections, and $\eta = 0$ in only excitatory ones. The parameters of the beta distribution were chosen to obtain a Gaussian-like shape. Further, μ_{W} and σ_{W} are the mean and width of either the positive or negative noise weights (depending on g). g simply changes the ratio between the mean excitatory and mean inhibitory weights. For instance, if $g > 1$, inhibitory weights are drawn from a Gaussian-shaped distribution with mean μ_{W} and width σ_{W}, and excitatory weights from a distribution with mean $\frac{\mu_{\text{W}}}{g}$ and $\frac{\sigma_{\text{W}}}{g}$. Similarly, if $g < 1$, excitatory weights are drawn with mean μ_{W} and width σ_{W}, and inhibitory weights with $g \cdot \mu_{\text{W}}$ and $g \cdot \sigma_{\text{W}}$. Note that by demanding $\sigma_{\text{W}} \leq \mu_{\text{W}}$, the distribution for positive and negative weights are always bounded by 0. Furthermore, rescaling the weight matrix W_{int} by a constant factor leaves the mean-to-width ratio of the distributions invariant, which ensures that even when rescaling, the distributions for negative and positive weights are bounded by 0. B_{ij} is a block diagonal matrix with zeroes on the block diagonal and ones everywhere else in order to ensure that neurons of the same BM do not "provide noise" to each other. N_{neurons} is the number of neurons per BM. Note that by choosing $\sigma_{\text{W}} = 0$, the positive and negative weights will be drawn from delta-peaked distributions, i.e., all excitatory weights are identical and all inhibitory weights are identical.

The interconnection matrix W_{int} is furthermore not symmetric. This construction has been chosen since, even if some neurons get the same noise spike trains, having random weights will diminish the effect of shared-input correlations as was demonstrated in the previous chapter. Illustrations of Eq. 5.2 can be found in App. 8.6.

The connectivity ϵ is implemented by setting a fraction $1-\epsilon$ of the noise connections of each neuron to 0. For example, the input noise neurons that feed neuron i are given by $C_i = \{j \mid j:\ W_{\text{int},ij} \neq 0\}$. From this list, $|C_i| \cdot \epsilon$ entries are randomly deleted, with $|C_i|$ being the number of elements in C_i. For the remaining entries $\bar{C}_i$, all interconnection weights are set to 0, i.e., $W_{ij} = 0$ if $j \in \bar{C}_i$.

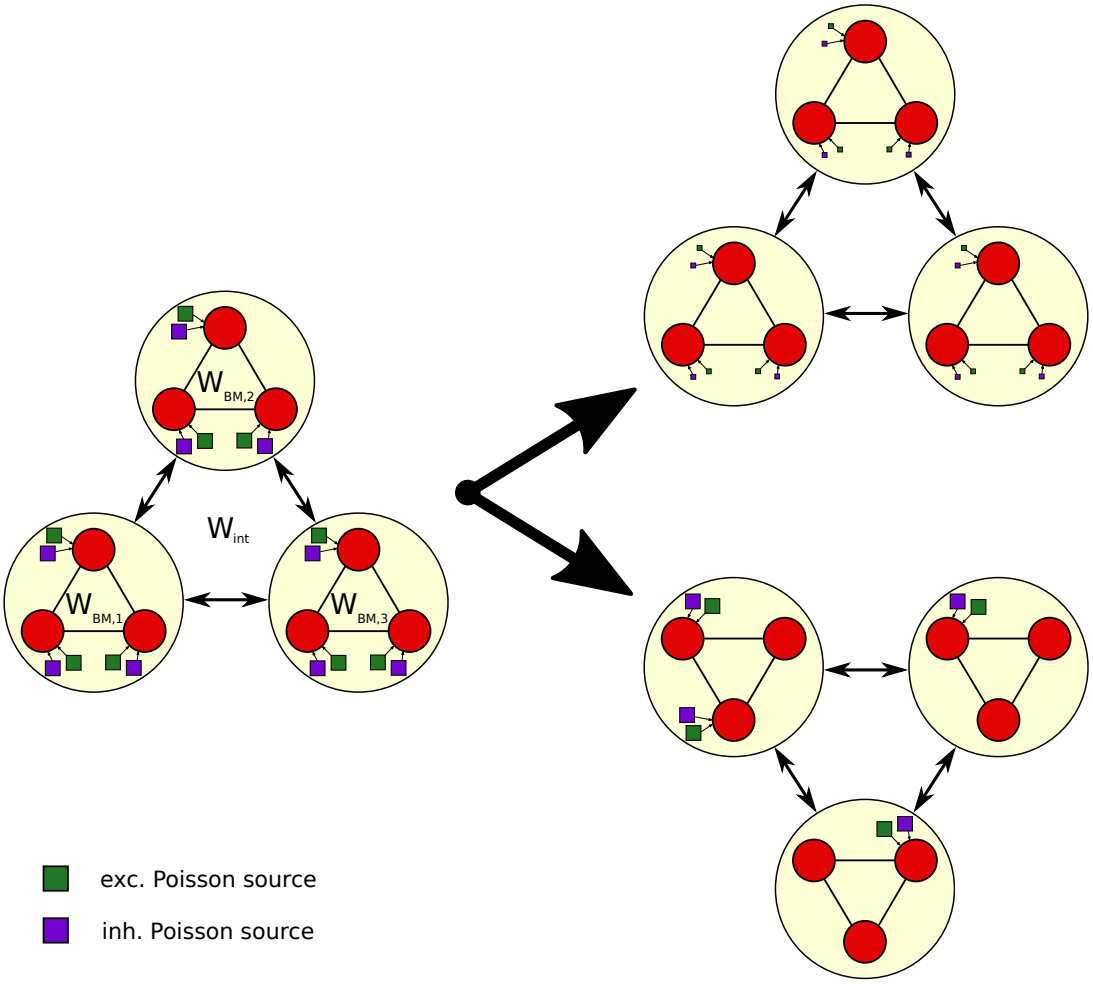

Figure 5.1: The three setups discussed in the following sections. **(left)** Several BMs driven by Poisson noise that sample from different Boltzmann distributions are interconnected. These additional connections are not part of the Boltzmann distributions and are regarded as *background noise*. Afterwards, there are two ways to continue: **(top right)** *Fading out* the Poisson noise frequencies, i.e., reducing the frequency of all Poisson sources during runtime, which in turn reduces the overall bandwidth needed to run the BMs. **(bottom right)** The second way to reduce the required noise bandwidth is to *cut off* some of the Poisson sources, i.e., we reduce the number of external connections plugged into the network.

After connecting the network BMs, two ways of removing Poisson noise will be used: Firstly, the input frequency of all Poisson inputs will be smoothly reduced, i.e., the Poisson noise gets slowly faded out. Secondly, instead of fading out all frequencies, a certain number of Poisson connections will be cut away altogether but the frequency remains constant. This will enable us to safely move to a network without any Poisson stimuli by either fading out the Poisson frequencies to 0 or by cutting off all external Poisson sources. A schematic of all three setups discussed in the following sections can be found in Fig. 5.1.

5.1 Introducing Interconnections in the Sea

5.1.1 Calibrating on Intrinsic Noise

One major issue when adding intrinsic network noise to the already available Poisson sources is that the activation function of each neuron changes depending on the intrinsic connection strengths. Furthermore, it is not at all trivial to find the correct activation function of each neuron since all neurons act, on one hand, as a noise source for other neurons, and receive noise from network neurons on the other. Hence, adding such intrinsic noise connections introduces *feedback loops* into the network, meaning that changing the spike behavior of neuron a, for instance by changing its calibration function, has a significant effect on the behavior of other network neurons, which in turn has an effect on neuron a again as it receives noise from the network.

Even though this problem is hard to solve, having sparse networks helps to get close to the ideal calibration function of each neuron, since its effect on the overall network and, most significantly, the effect on neurons providing feedback is less pronounced. Having Poisson noise in addition to intrinsic noise connections further stabilizes the calibration scheme for weak intrinsic connections. By slowly increasing the interconnection strengths, we can transition into a regime where the network is mainly driven by intrinsic noise.

Note that if the network BMs are still able to sample while being mainly driven by spike trains coming from the network itself, it should in principle be possible to reduce the overall Poisson bandwidth instead of the weight ratio between intrinsic and Poisson connections. In doing so, we should end up in a similar network setup where the intrinsic noise is compensating for the loss of Poisson stimuli.

To calibrate each neuron, an iterative scheme (see Fig. 5.2) was used at first. The idea was to compensate for possible changes due to feedback by slowly increasing the interconnection weights from 0 to their final desired value. This way, we would start by calibrating each BM only with Poisson noise, sample from every Boltzmann distribution, take the spike trains, and use these as an approximation of the spike trains we expect to see when introducing a very small increase in the interconnection weights. It is then possible to calibrate every neuron on its Poisson input and on a representation of what

to expect as input coming from the network.

However, it turned out that this perturbative ansatz leads to unstable results for interconnection weights larger than the Poisson weights and can be replaced by a greedy simplification that is stable for large interconnection weights and leads to similar results as the iterative scheme for small weights (see Fig. 5.3).

The instabilities occur because of strong synchronizations in the network originating from a drastic difference between the noise input used for calibration and the actually observed noise while sampling. This way, if a neuron provides, for instance, excitatory noise to a subset of network neurons, but it is more active than anticipated during calibration, this will result in a synchronization of the subset of neurons and the noise neuron, i.e., they get more active than expected as well. For strong interconnections, this leads to subsets of neurons that either burst or remain completely silent as shown in App. 8.8.5.

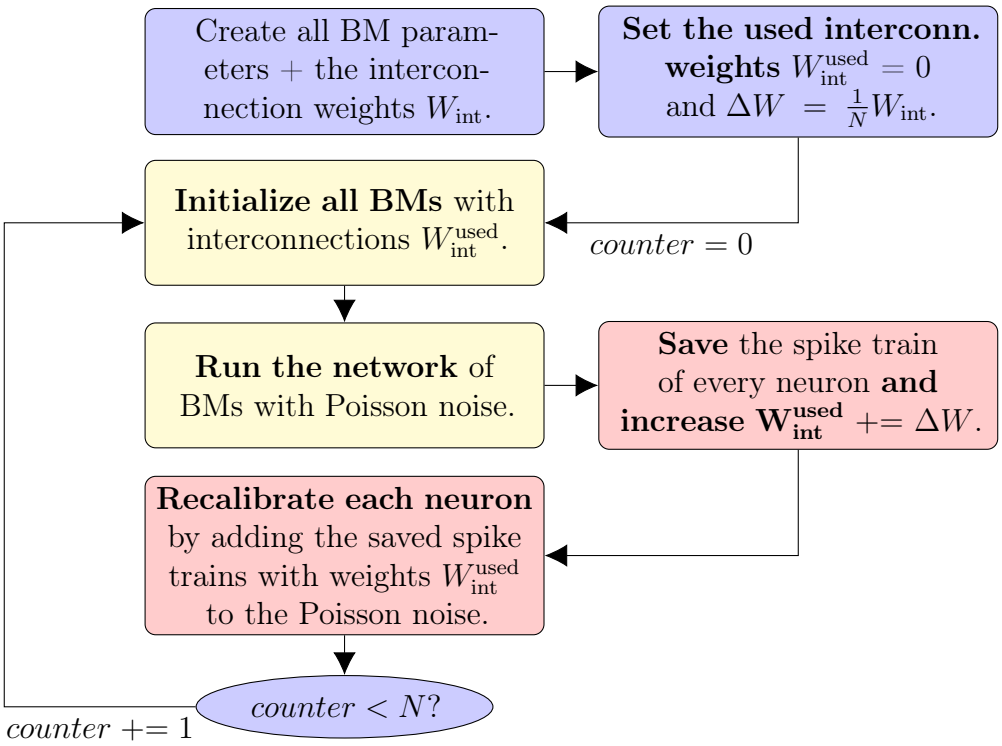

Figure 5.2: Iterative calibration scheme. In each iteration, the spike trains from the previous step are used for calibration to approximate the intrinsic noise each neuron will see during sampling. The scheme starts with no interconnection weights and gradually increases these weights throughout each step. It turns out that, because of correlations between network neurons, this scheme becomes unstable for weights much larger than the used Poisson weights.

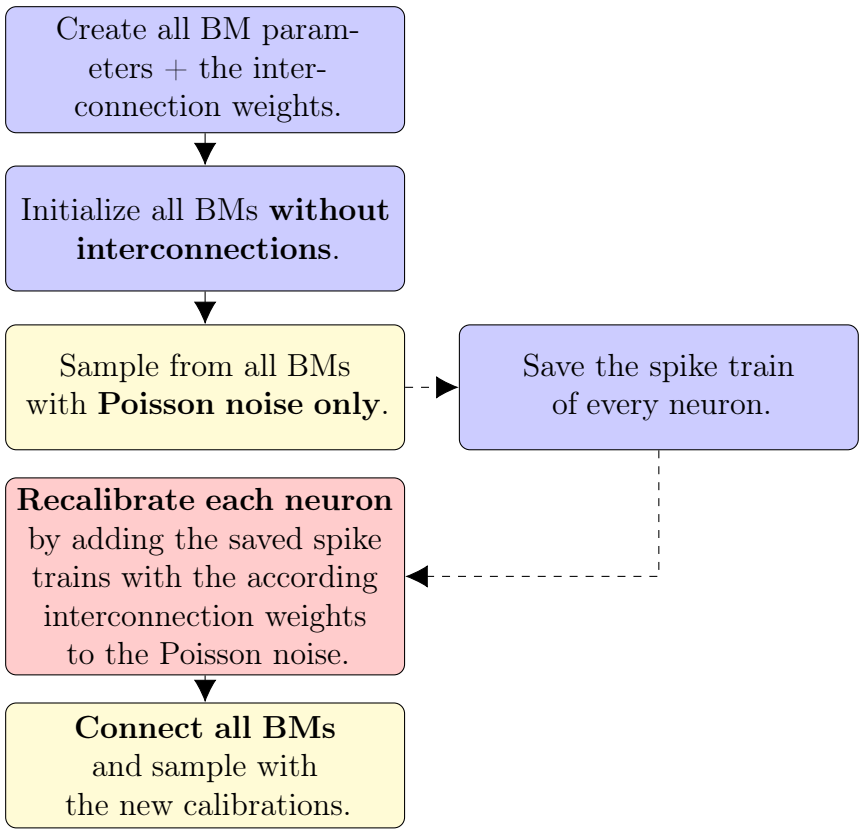

Figure 5.3: Greedy calibration scheme. In principle, it is very similar to the iterative scheme presented in Fig. 5.2. However, the whole iteration process is left out and the interconnection strengths are immediately set to their final value after a single iteration. Since we expect LIF sampling to work for sparse enough networks, this is a valid approach as the spike train statistics of every neuron should remain the same with and without interconnection weights. Hence, the spike trains obtained from ideal LIF sampling with Poisson noise only should be a relatively good approximation of the intrinsic noise each neuron observes while sampling.

In the greedy approach, the spike trains obtained from ideal LIF sampling with Poisson noise only are used as an approximation for the intrinsic noise each neuron will see while sampling with interconnections turned on. Hence, we assume that all BMs will still be able to correctly sample from their target Boltzmann distribution with intrinsic noise and therefore, the spike train statistics of each neuron should not change in any significant way from the ideal Poisson case. Of course, this is not entirely true since shared-input correlations or correlations between neurons of independent BMs in the network distort the Boltzmann distribution each BM is sampling from. These network-wide correlation effects are also the reason why the iterative scheme fails for large weights as they cannot be compensated by recalibrating every neuron, since neurons are

completely unconnected and independent during calibration. Therefore, in the iterative scheme, errors introduced by correlations add up over several iteration steps, leading to the unstable behavior for large interconnection weights.

For this reason, only the greedy calibration scheme will be used in all following chapters. Moreover, neglecting all iteration steps increases the speed of the calibration process, enabling us to look at larger networks without running into problems with large simulation durations.

5.1.2 Dealing with Network-Wide Correlations

In Chap. 4 it was demonstrated that input correlations can be dealt with by averaging over enough input spike trains. However, this only worked since we had no feedback connections, i.e., an isolated BM got noise from a pool of completely independent BMs.

If we have a network of BMs which provide noise to each other via additional interconnections, the situation becomes more complex. For instance, since each BM provides noise to many other neurons in the network, but also receives noise from neurons inside the network, (higher order) feedback loops can have a significant impact on the network dynamics as a whole.

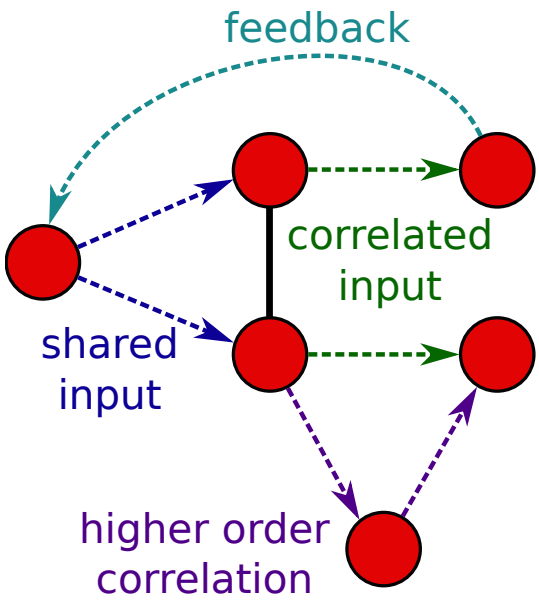

Figure 5.4: Possible ways how correlations can be introduced and propagated through the network. Since the network size is finite, some neurons have to share the same input (blue), leading to shared-input correlations. Furthermore, two correlated neurons can provide noise to independent or connected neurons, altering their initial correlation (green). Finally, we can also have feedback connections, i.e., a pair of correlated neurons provides noise to each other (cyan), and many ways of higher order correlations (magenta).

The simplest kind of correlations are shared-input correlations, appearing naturally because every neuron is getting its noise from a finite pool of neurons. Correlations due to input from neurons that are connected via a synapse or are somehow otherwise correlated, for example because of shared-input correlations, are also possible and have been discussed in Chap. 4. In addition, feedback is possible as well since every single neuron is feeding noise into the same system it is receiving its own noise from. Some possible example cases are presented in Fig. 5.4. Introducing interconnections thus leads to a propagation and possibly amplification of correlations in the network.

A similar problem was analysed in *Kumar et al.* (2008). They investigated whether it is possible to have a recurrent network of LIF COBA neurons that are able to sustain their firing state without any external input over some time. Interestingly, they found out that the network can spontaneously die out to a zero-rate state due to strong network synchronizations. As in our case, these network synchronizations originate from unavoidable overlaps of input noise sources as well as pairwise and higher order correlations in the network.

However, the effect of these correlations can be diminished by increasing the network size but keeping the number of inputs every neuron receives fixed. This way, the number of shared inputs decreases and (higher order) feedback loops become less problematic because of the reduced connectivity in the network, i.e., the network of BMs becomes sparser and correlations can propagate less. For instance, if two neurons obtain on average $\epsilon \cdot N$ inputs from a pool of N neurons, they will share on average $\epsilon^2 \cdot N$ inputs. Increasing N while decreasing $\epsilon \propto \frac{1}{N}$ will reduce the number of shared inputs $\epsilon^2 \cdot N \propto \frac{1}{N}$. This procedure is demonstrated in Fig. 5.5 where the total number of inhibitory and excitatory inputs every neuron receives is fixed to $\epsilon \cdot (N_{\mathrm{BMs}} - 1) \cdot N_{\mathrm{neurons}} = 20$. N_{neurons} is the number of neurons per BM and was set to 3, N_{BMs} is the number of BMs in the network and was varied from 10 to 150. The Boltzmann parameters were randomly drawn from the beta distributions given in Eq. 5.1a and 5.1b with $W_0 = 1.2$. Also note that the neuron parameters are still the same as in the previous chapters. The synaptic ratios η and g as well as the mean-to-width ratio of the interconnection distribution were all set to 1.

As can be seen in Fig. 5.5, the overall LIF sampling quality increases over the range of investigated interconnection strengths for sparser networks. Increasing the network size while decreasing the connectivity leads to less correlations in the network. We can actually look at two kind of correlations: The correlation of states between neurons of the same BM and the correlation between neurons of different BMs. The former is predetermined by the synaptic weight connecting the two neurons, but will differ because of correlated input. Neurons of different BMs should be completely independent as no functional synaptic weight connects them. Therefore, the states of such neuron pairs should be completely uncorrelated which is however not the case for finite-sized networks with intrinsic noise connections.

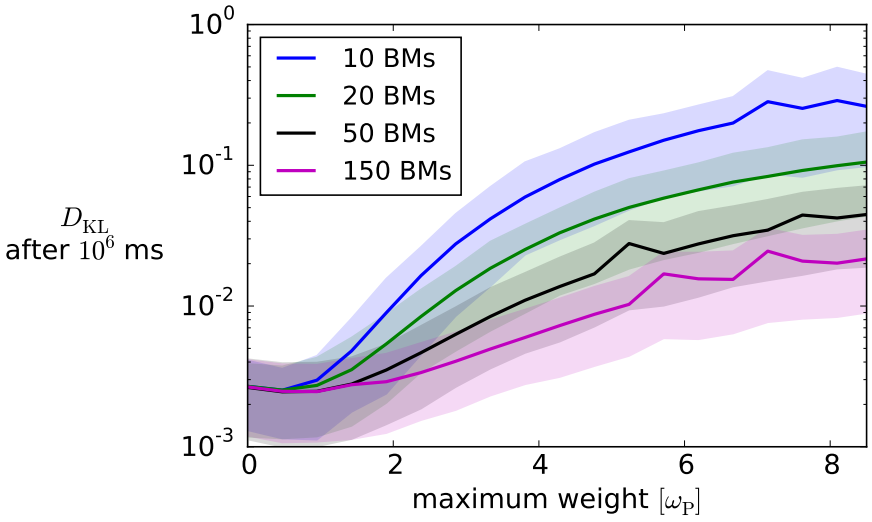

Figure 5.5: Mean D_{KL} of all network BMs after sampling for 10^6ms shown for different network sizes. The connectivity is adjusted such that the mean number of inputs every neuron gets is held constant. First note that the D_{KL} saturates for high interconnection weights as correlations in the network are maximized. Also, the small abrupt deviations as seen for instance in the magenta curve are systematic and result from failed calibrations. This demonstrates that even a small number of wrongly calibrated neurons in the network can have a significant effect on the sampling quality. Furthermore, for weights that are small compared to the Poisson noise weights ω_{p}, the sampling quality stays constant. As expected, the overall sampling quality improves for sparser networks. The simulation was repeated for five different random seeds. The connectivities used are (blue) $\epsilon = 74\%$, (green) $\epsilon = 36\%$, (black) $\epsilon = 14\%$ and (magenta) $\epsilon = 5\%$.

In Fig. 5.6, the state correlation coefficients between neurons belonging to the same BM from the simulations of Fig. 5.5 were plotted in a histogram. This was once done for the completely unconnected network, i.e., every neuron receiving Poisson noise and the correlations only depending on the synaptic weights connecting them, and for the network with maximum interconnection weights.

For small network sizes, the correlation coefficients of the unconnected and connected case differ drastically. This explains the bad sampling quality, as the neurons of a BM are not able to follow the target state distribution because of the additional correlations. But decreasing the connectivity of the network while increasing the network size reduces the shared-input correlations as expected.

The same can be observed for the correlations between neurons of different BMs, as demonstrated in Fig. 5.7.

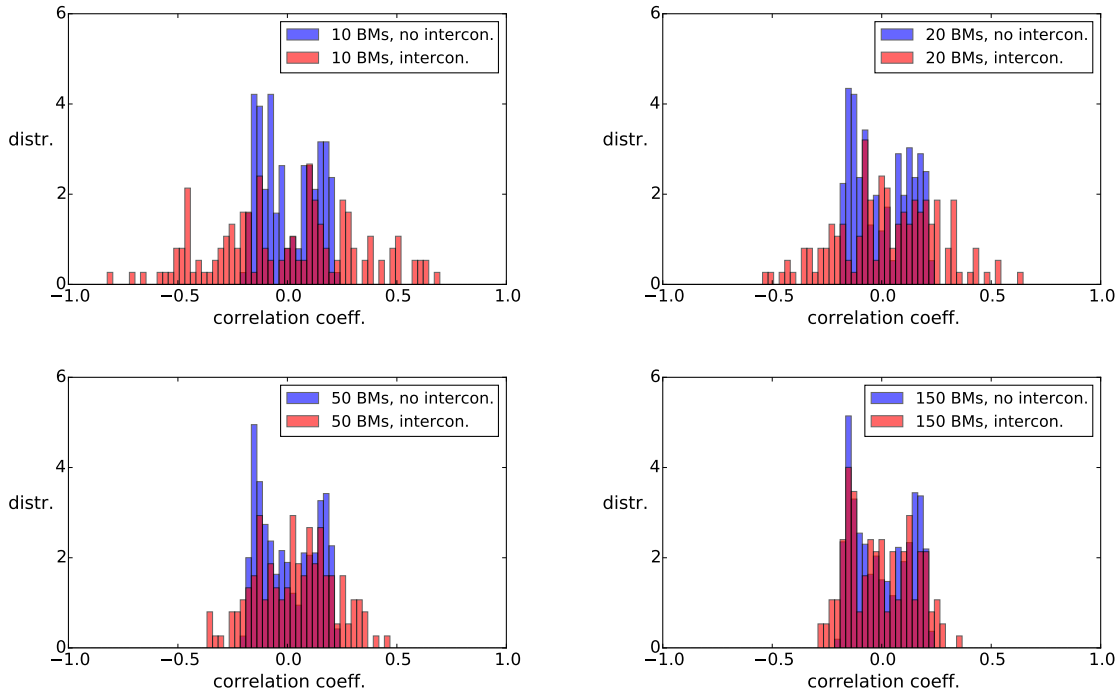

Figure 5.6: State correlations of neuron pairs in a network of BMs. The correlations are only evaluated from neuron pairs that belong to the same BM. If the network of BMs is unconnected, the correlation coefficients are determined by the synaptic weights of the BMs only (blue). Introducing intrinsic noise connections will distort the correlation structure between neurons of the same BM (red), reducing the overall sampling quality. One possible source for correlations are shared-input correlations which appear naturally due to the finite number of neurons in the network. These can be reduced by increasing the network size while keeping the input each neuron receives constant. This way, the network becomes sparser, i.e., less connected, and network-wide correlations are weaker.

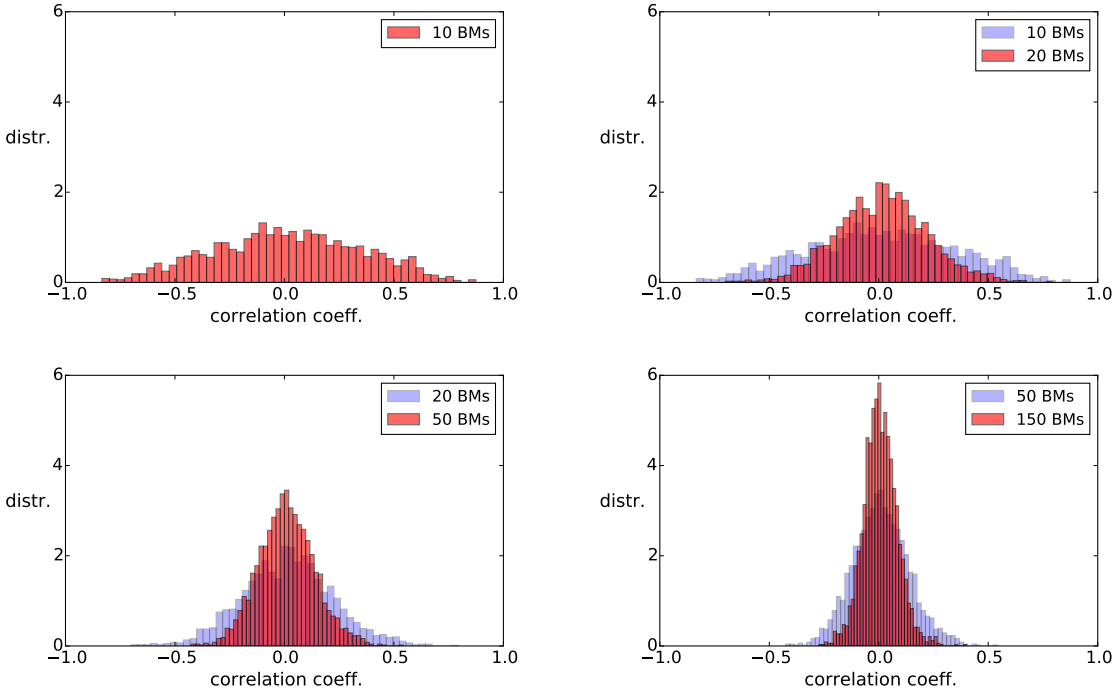

Figure 5.7: Similar setup to the one in Fig. 5.6, but for neuron pairs of different BMs. At most 500 pairs were evaluated for each histogram. If two neurons of the network are not part of the same BM, no functional synaptic weight connects them and their dynamics should be completely independent. But again, shared inputs and feedback loops lead to synchronizations between otherwise independent neurons. For example, if such a neuron pair is positively correlated, they will both spike preferably together. However, both neurons are part of a distinct BM and should follow a certain distribution of state overlaps determined by the respective Boltzmann parameters. The correlations between neurons of different BMs therefore distort these state overlaps, resulting in a loss of sampling quality.

Moreover, in all cases, the quality can be adjusted by choosing the interconnection weights small enough in comparison to the Poisson noise weights ω_p. This is not surprising since the Poisson noise acts as a source of uncorrelated input. It therefore diminishes the effect of correlations induced by the interconnections. By increasing or decreasing the strength of the Poisson stimuli, the interconnection weight strength at which a deterioration of the sampling quality is observed can be increased or decreased, as demonstrated in Fig. 5.8.

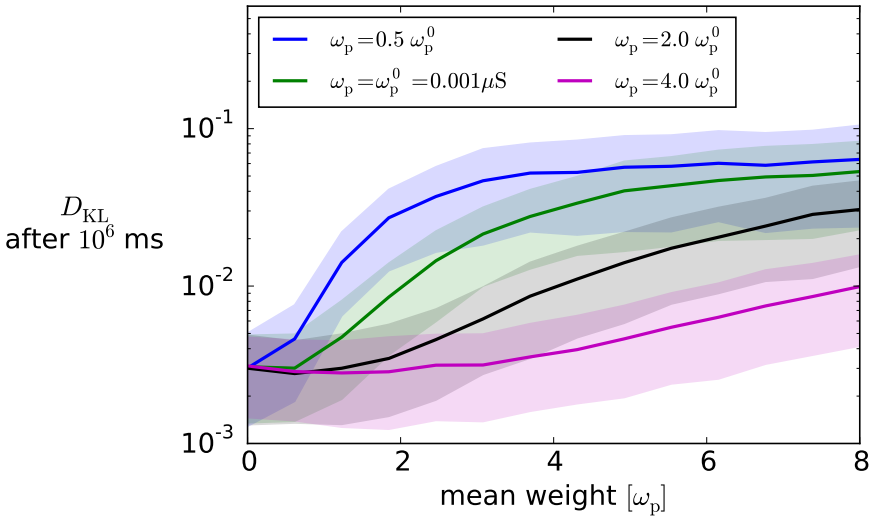

Figure 5.8: Similar setup to the one in Fig. 5.5. In a network of 50 3-neuron BMs, the strength of the Poisson sources was varied. This leads to differently drastic changes in D_KL when increasing the interconnection weight strengths. Since the Poisson sources act as an uncorrelated input, the influence of the correlated input coming from the network depends on the relative strength between Poisson and network noise. If the Poisson strength is reduced, the sampling quality already worsens for weaker interconnection weights. Similarly, increasing the Poisson weights allows larger interconnection weights. All in all, the strength of the Poisson input ω_p sets the scale of usable interconnection weights.

It is worth noting that after connecting and newly calibrating every neuron, their spiking activity becomes more irregular. This can be quantified by calculating the *coefficient of variation* of the interspike intervals CV_ISI for each neuron. It is defined as:

$$CV_\mathrm{ISI} = \frac{\sigma_\mathrm{ISI}}{\mu_\mathrm{ISI}}\,, \tag{5.4}$$

where μ_ISI is the mean distance of spikes and σ_ISI the corresponding standard deviation.

For instance, if the CV_{ISI} is larger than 1, the spike times are very irregular as the standard deviation of spike distances is larger than the mean distance. Similarly, small values of CV_{ISI} indicate very regular spiking as most spike distances are very similar. A Poisson source creates spike trains with a CV_{ISI} of exactly 1. Normally, since LIF neurons can have small periods of bursting with fixed refractory times, the CV_{ISI} of their spike trains will be between 0 and 1. But introducing background noise increases the irregularity of spiking, as seen in Fig. 5.9. Because every neuron provides noise to the network, and the network noise to every neuron, small changes in the input noise a neuron receives will slightly change its spiking behavior which in turn has an effect on the network again. Therefore, because of this 'back-action', it is actually quite intuitive that interconnections will lead to a more irregular spiking activity.

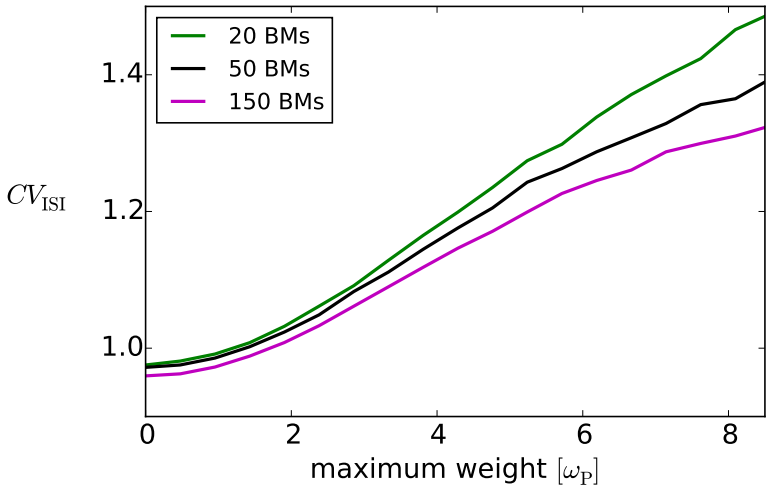

Figure 5.9: Coefficient of variation of the interspike intervals. Using the background activity of the network as an additional noise source results in more irregular spiking activity as can be seen by the increasing CV_{ISI} for stronger network weights.

We also have to ensure that the calibration used to translate theoretical Boltzmann parameters to LIF parameters is approximately correct. In general, we will always have the problem that we can only approximate the input a neuron receives from the network as noise. In Fig. 5.10, the used calibration and the actually observed activation function during the network run are compared for the case of 20 BMs and the maximum weights shown in Fig. 5.5. Even though the approximated calibrations agree quite well with the real activation functions, the shape of the activation function and the rest potential u_0 at which the probability of being refractory is $p(u_0) = 0.5$ are always slightly different.

These differences can again be reduced by decreasing the network connectivity as shown in Fig. 5.11, where the number of BMs was increased to 150. However, small

deviations between the used activation function and the real one still exist and are a non-negligible source of error concerning LIF sampling in such networks.

Finally, as a side remark, it turns out that the mean-to-width ratio of the interconnection weight distribution has no significant influence on the LIF sampling quality, see App. 8.8.6. E.g. for high widths, we obtain both strong and weak weights that translate to strong and weak correlations in the network. However, on average the total correlation in the network does not change. It is therefore not necessary to tune the absolute noise weights in any way as long as they scatter symmetrically around some mean value.

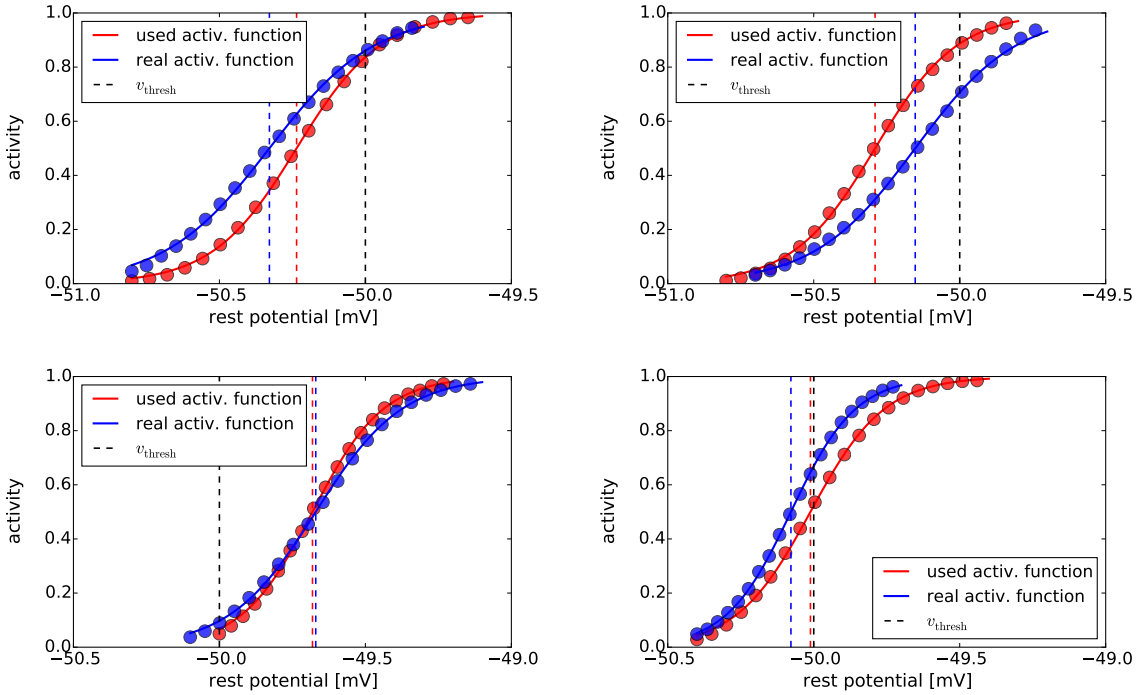

Figure 5.10: Activation functions of exemplary neurons with background noise coming from the network. In red, the measured activity (dots) and the fitted activation function (line) obtained from the greedy calibration scheme are shown. In blue, the measured activity and resulting calibration function with the real noise observed while running the interconnected network is shown. The black dashed line marks the threshold potential. Even though the guessed and real activities are rather similar, the differences are large enough to affect the LIF sampling quality in a significant way. **(top)** Imprecise calibration functions can result in a wrongly translated bias because of differences in the rest potential for which the activity is 50%, marked in the plots as vertical dashed lines. **(bottom)** Furthermore, a difference in slope will lead to wrongly translated weights, affecting the correlation structure between neurons of a BM.

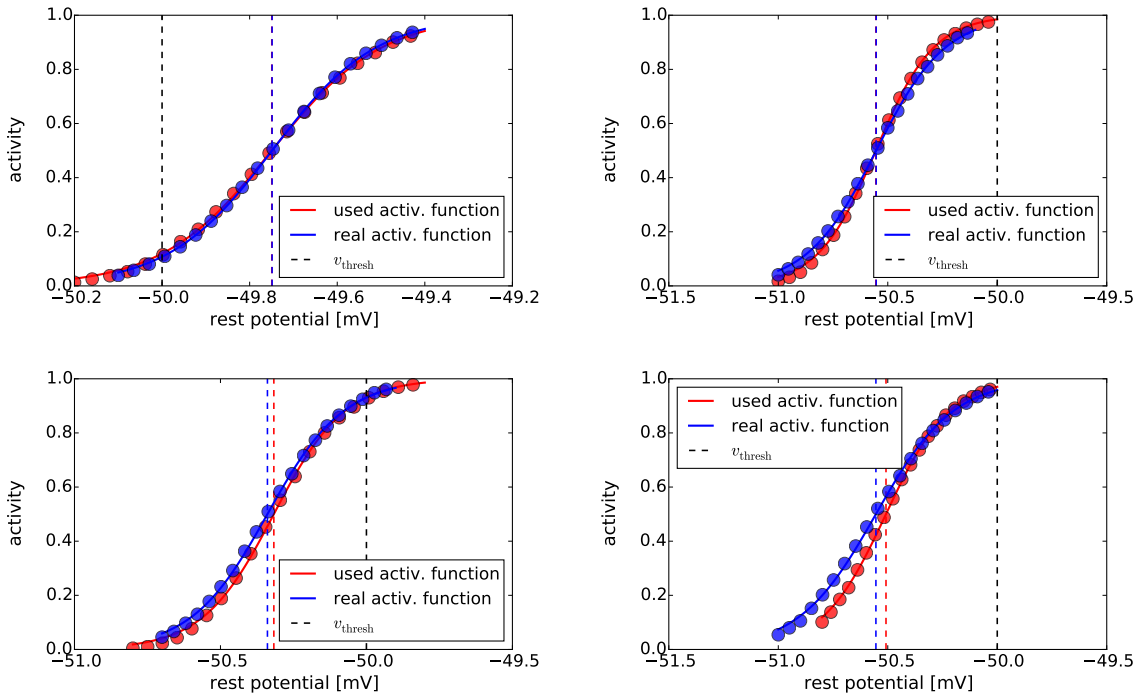

Figure 5.11: The same as in Fig. 5.10, but for a larger network that consists of 150 BMs. A sparser network does not only decrease network-wide correlations, but also improves the agreement between the approximated and actual activities consequently (top left). However, minor differences of varying severity are still possible (top right and bottom).

5.2 Intrinsic Noise Restoring Stochasticity

In the previous section, we found that the D_{KL} obtained from LIF sampling improves if the network of BMs is sparse, i.e., consists of many BMs with low connectivity. In this regime, the network activity can be described as being *asynchronous irregular* (AI) as the mean CV_{ISI} is around or above one, depending on the strength of the interconnections, and unwanted correlations resulting from noise spike trains are rather weak in comparison to the functional correlations between neurons of the same BM.

Even though the setup is quite different, these results are similar to the observations of *Korcsák-Gorzó* (2015), who recommended using inhibitorily dominated and sparsely connected BMs operating in the AI state. Note that network connections are still balanced in our case, but possible effects of inhibitorily dominated connections will be discussed in Chap. 6. Furthermore, the strength of the network connections can be varied to adjust the degree of additional correlations in the network. This will be important in the following sections when reducing the Poisson bandwidth.

5.2.1 The Poisson Fade-Out

The first ansatz to reduce Poisson bandwidth consists of globally fading out the Poisson frequencies to a negligible level during runtime. This has been implemented by changing the Poisson frequency from a starting frequency ν_{start} to a target frequency $\nu_{\text{target}} < \nu_{\text{start}}$ with a logistic function

$$\nu(t) = \nu_{\text{start}} - \frac{\nu_{\text{start}} - \nu_{\text{target}}}{1 + \exp\left(-\frac{t - t_\nu}{\sigma_\nu}\right)}, \tag{5.5}$$

where $t_\nu = 5 \cdot 10^3$ms and $\sigma_\nu = 10^2$ms are used to adjust when and how smooth or rapid the frequency is faded out (see Fig. 5.12). A continuous transition from start to target frequency has been chosen to ensure that the network can smoothly adjust from a Poisson-driven to a background-noise-driven network activity. Moreover, *Kumar et al.* (2008) reported unstable behavior for abrupt changes of external input frequencies in recurrent networks and used an exponential decay to turn off the external sources.

The calibration is done the following way: First, as in the greedy scheme, every BM is run with Poisson noise with frequency ν_{start}. Afterwards, the spike trains obtained from sampling are used to calibrate every neuron with both background and Poisson noise, however now with a reduced Poisson frequency ν_{target}. While sampling, the network will start with strong Poisson input, ensuring that the neurons spike with a higher mean frequency as dictated by the theoretical Boltzmann parameters. Decreasing the Poisson frequency will smoothly change the spike behavior of every neuron until the target frequency is reached and all neurons settle at an activity similar to the one used in the calibration (see Fig. 5.12). This way, every neuron is sampling from its intended Boltzmann distribution and provides correct noise to the rest of the network. This is a fixed point in terms of the network activity as each neuron provides the correct spike train statistics and receives the correct input noise to keep on sampling from the correct distribution. Note that sampled states are only recorded after the fade-out.

In the following simulations, the neuron parameters have been changed to the values found in App. 8.2, such that noise frequencies below several hundred Hertz have a significant impact on the LIF sampling quality.

First, a small one-shot experiment was performed with a network of 200 3-neuron BMs. The Boltzmann parameters were again randomly drawn from the beta distributions given in Eq. 5.1a and 5.1b with $W_0 = 2.0$. The size of the network was chosen this way to lower correlations coming from the interconnections as was demonstrated in the previous section. The remaining parameters were set to $g = 1.0$, $\eta = 0.5$ and $\frac{\sigma_W}{\mu_W} = 1.0$. The average number of inputs each neuron gets was set to 28, which translates to a mean excitatory and inhibitory background noise frequency of 700Hz. As already mentioned in Fig. 5.12, the Poisson frequencies were set to $\nu_{\text{start}} = 700$Hz and $\nu_{\text{target}} = 2$Hz.

For 2Hz Poisson input, the LIF neurons are all almost completely deterministic. This can be easily seen by measuring their activation function as it has a non-logistic and

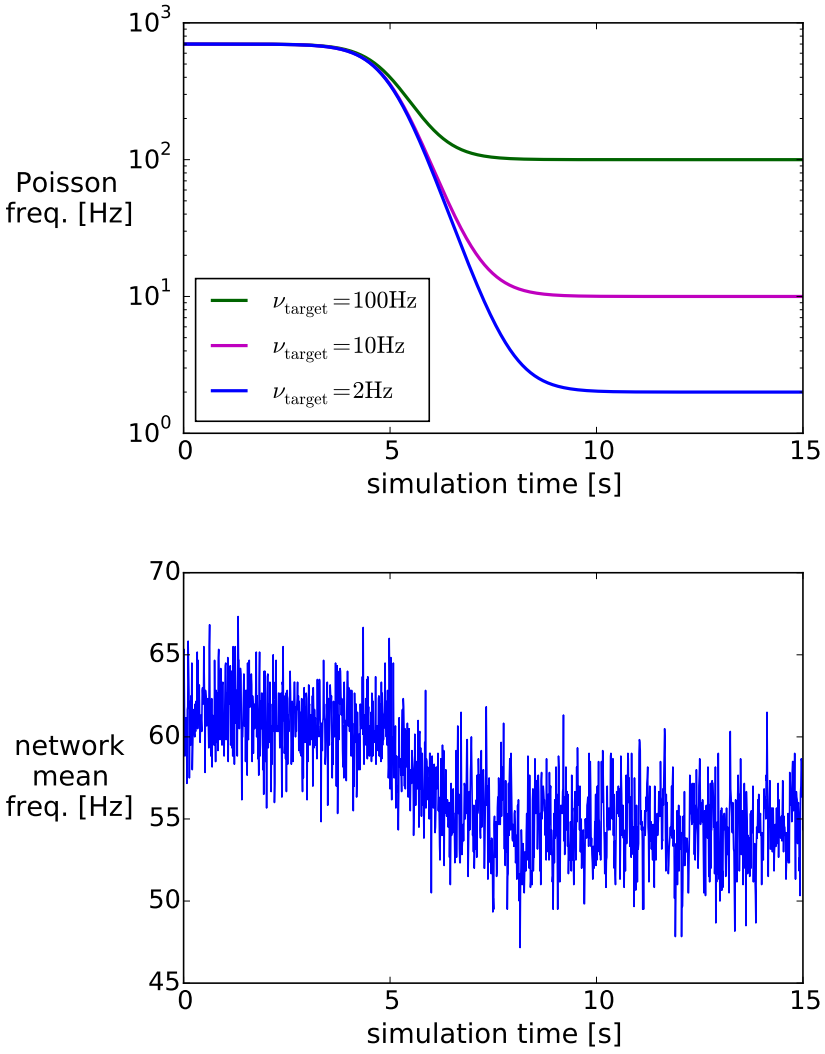

Figure 5.12: **(top)** To reduce the Poisson bandwidth, the frequency of every Poisson source is reduced with a logistic function, demonstrated here for several different target frequencies ν_{target} and starting frequency $\nu_{\text{start}} = 700\text{Hz}$. **(bottom)** Since the network starts with a higher Poisson frequency than it was calibrated on, the mean network activity or mean frequency of all network neurons is initially higher than during LIF sampling. Smoothly decreasing the external Poisson frequency then lowers the mean frequency of the network to the desired level specified by the neuron calibrations. This is a fixed point where every neuron samples correctly from its distribution and receives the correct noise input to have a similar activation function as the one used while calibrating. The mean frequency was calculated by averaging the total number of network spikes over 10ms intervals for the case of $\nu_{\text{target}} = 2\text{Hz}$, which is further discussed in Fig. 5.14.

asymmetric shape (see Fig. 5.13). The activation function is not quite a Heaviside theta function because of the finite time the neurons need to get from the reset potential to the threshold again. Obviously, this has a drastic influence on the sampling quality as the activation function encodes the conditional probability of a LIF neuron to be refractory. However, if we add intrinsic network noise to the weak 2Hz Poisson sources, the symmetric logistic shape of the activation function is recovered. Hence, the loss of external noise gets compensated by the background activity of the network itself.

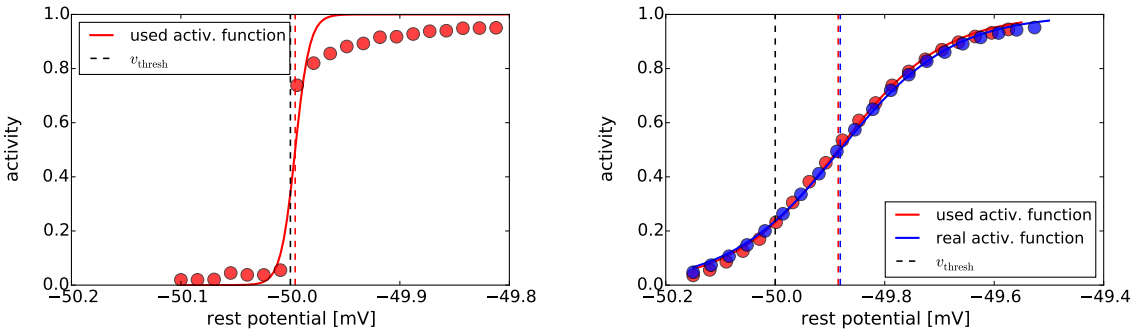

Figure 5.13: Activation functions without and with background noise coming from the network itself. **(left)** Activation function of a neuron with 2Hz Poisson input (dotted) and the corresponding logistic fit (line). The shape is reminiscent of a deterministic neuron which only starts spiking when the rest potential is set above threshold. The slow convergence to the maximum firing rate is caused by the finite time the neuron needs to get from the reset potential to the threshold again after spiking. **(right)** Adding background noise to the 2Hz Poisson input recovers the familiar logistic shape of a stochastic LIF neuron. Both the activation function used for calibration and the observed one during sampling are shown here.

This is also reflected in the mean D_{KL} evolution while sampling (see Fig. 5.14). For only 2Hz Poisson input, the lack of stochasticity prevents the BMs to traverse every state according to its Boltzmann distribution. This results in much worse D_{KL} values on average than for high frequency Poisson input, e.g. 700Hz input. But introducing interconnections recovers the stochasticity and the sampling quality improves notably. Even though there is still a small difference to the quality reached by ideal Poisson sources, it is quite surprising that sampling with almost no Poisson noise works so well out of the box. Again, as in the previous section, the quality can be further improved by making the network sparser.

Also note that the mean of the interconnection weights has to be adjusted carefully. Values which are too high will lead to unnecessary correlations and synchronizations in the network which distort the distributions sampled from. If the weights are chosen

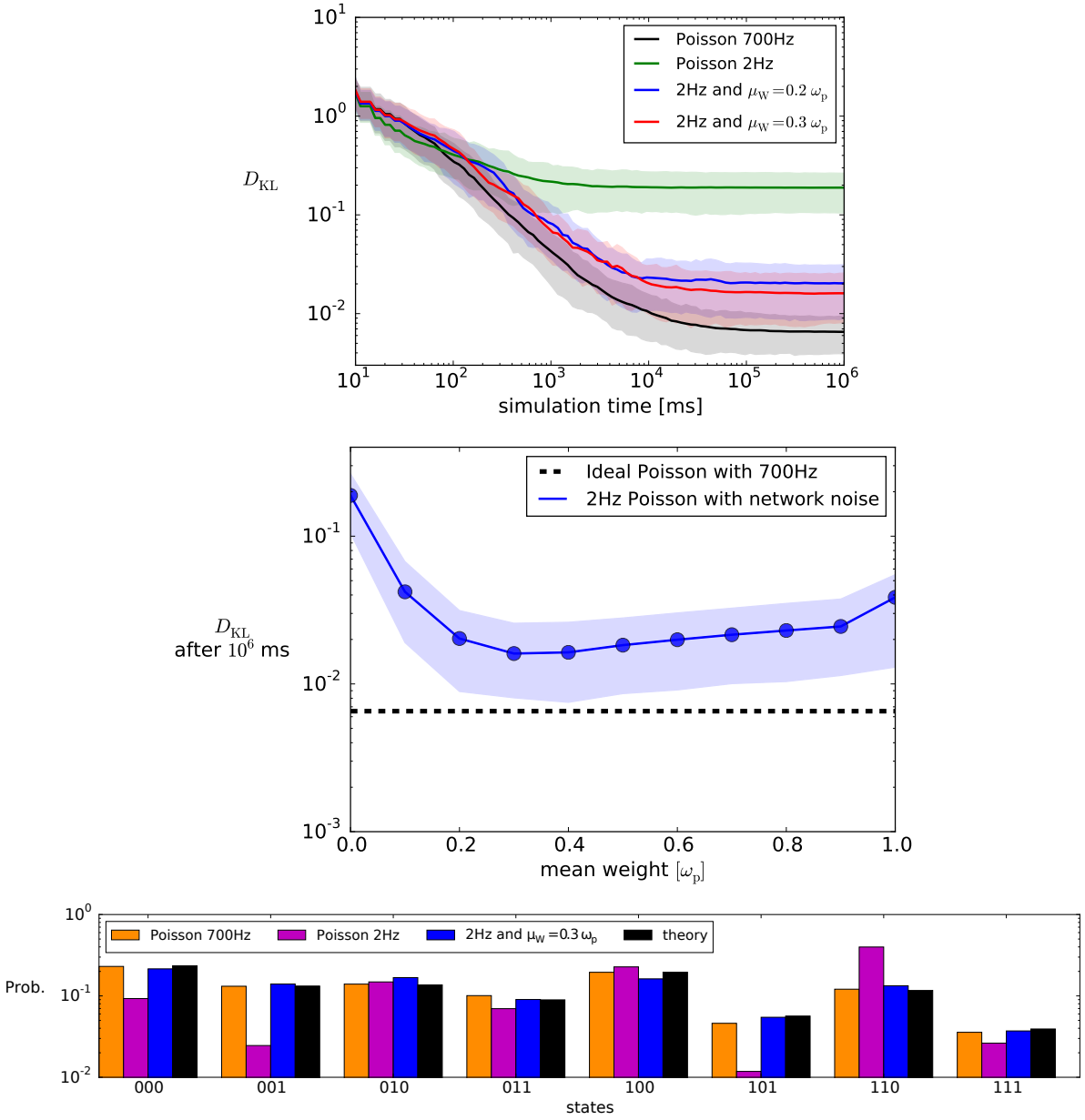

Figure 5.14: Mean sampling quality of a network consisting of 200 3-neuron BMs. **(top)** Mean D_{KL} over all BMs in the network as a function of the sampling time. If we only use 2Hz Poisson sources, the sampling converges rather quickly at much worse values than for the 700Hz Poisson sources. However, adding intrinsic noise coming from the network improves the sampling quality significantly. **(middle)** The sampling quality can be adjusted by changing the strength of the interconnections. Weak connections prevent the network neurons to reach the high-conductance state and too large weights introduce strong correlations which distort the sampled distributions. **(bottom)** Illustration of the sampled distribution of a single network BM with and without intrinsic noise.

too small, the background noise is not strong enough to heave the LIF neurons into the high-conductance state, resulting in asymmetric activation functions.

Of course this also works for different target frequencies. Somewhat surprisingly, the resulting sampling quality saturates for vanishing Poisson frequencies at D_{KL} values very close to the 2Hz case investigated before (see Fig. 5.16). Therefore, we are actually able to completely fade-out the Poisson sources and keep on sampling stably by compensating the missing external sources with intrinsic noise from the network.

The limiting case studied here damps the Poisson frequencies down to $\nu_{\mathrm{target}} = 0.001\mathrm{Hz}$, at which point LIF sampling will break down completely. If interconnection weights are introduced, but they are not strong enough, the neurons inside the network will not be able to enter the high-conductance state, see Fig. 5.15. But again, increasing the weights further pushes the neurons back into the high-conductance state, enabling them to sample from their correct conditional probability.

This is a very important result as it allows us to exchange nearly all external noise input with spike trains coming internally from the network of BMs. This is very similar to the observation of the previous section where the final D_{KL} saturated for very large interconnection weights at non-extreme values. Similarly to the approach here, this also represents a limiting case with Poisson input having almost no effect on the activation function.

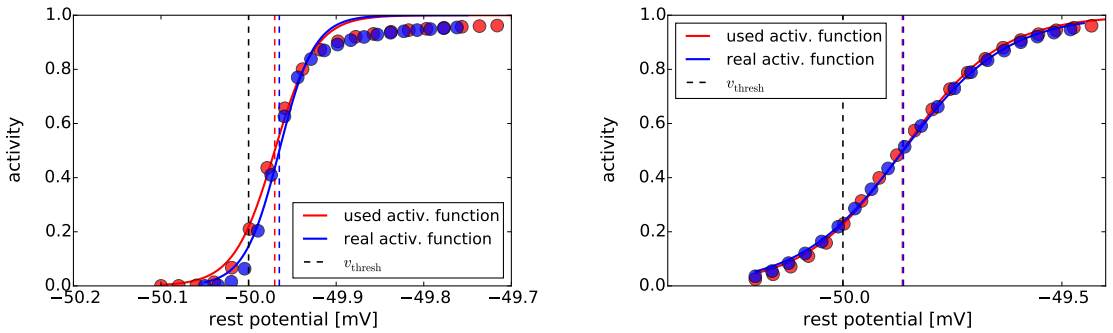

Figure 5.15: Activation function for a Poisson source with $\nu_{\mathrm{target}} = 0.001\mathrm{Hz}$ and background noise with varying connection strengths. **(left)** The mean strength of the interconnection weights is 0.06 ω_{p} and therefore too weak to push the neurons into the high-conductance state. Even though the neuron is not deterministic anymore, the activation function is not yet symmetric. **(right)** By increasing the mean value of the interconnection weights, for instance to 0.36 ω_{p}, the symmetric shape can be regained despite the very low-frequency Poisson sources.

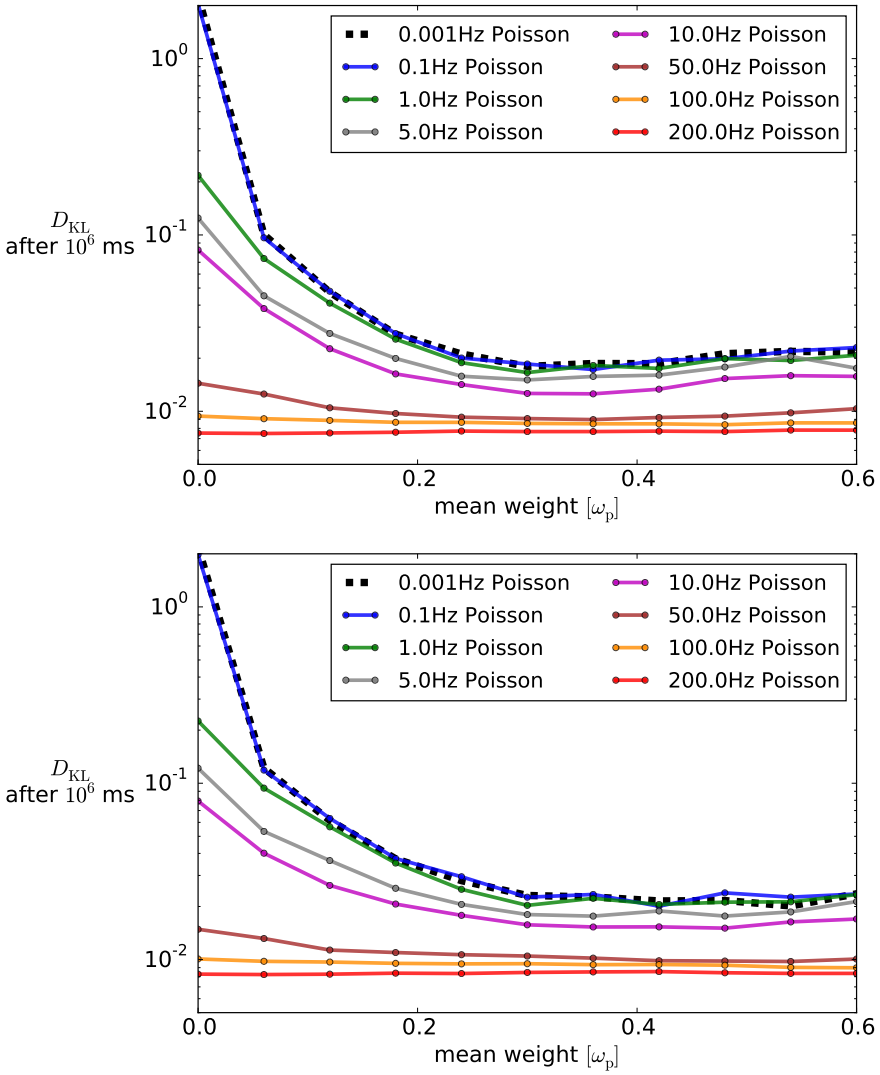

Figure 5.16: Mean D_{KL} of a 200 3-neuron BM network after sampling for 10^6ms for different target frequencies ν_{target} and different mean interconnection weights. The simulation was repeated for two different random seeds, i.e., two different network realizations, which are presented in the top and bottom plot. For a mean weight of zero, we can clearly see how the sampling quality gradually decreases for smaller Poisson frequencies. However, in all cases, the loss of external noise can be compensated by increasing the interconnection weights enough. Interestingly, the decrease in sampling quality saturates at non-extreme D_{KL} values, e.g. the D_{KL} curves for $\nu_{\mathrm{target}} = 0.1$Hz and $\nu_{\mathrm{target}} = 0.001$Hz are not distinguishable. Error bars have been excluded to ensure readability.

5.2.2 The Poisson Cut-Off

Another method to reduce the external bandwidth consists of cutting off Poisson sources of single neurons. By doing so, some neurons will only receive noise coming from the network, while some others will still get Poisson input in addition. For a neuromorphic implementation, this means that only a subset of neurons need to be provided with external noise.

With this approach, it is easy to gradually cut off all the external Poisson inputs until none remain and the network has to generate its own stochasticity again. This is demonstrated in Fig. 5.17. Note that, if we use the greedy calibration scheme, each neuron only needs to be calibrated twice, once with and without Poisson sources on top of the network background noise.

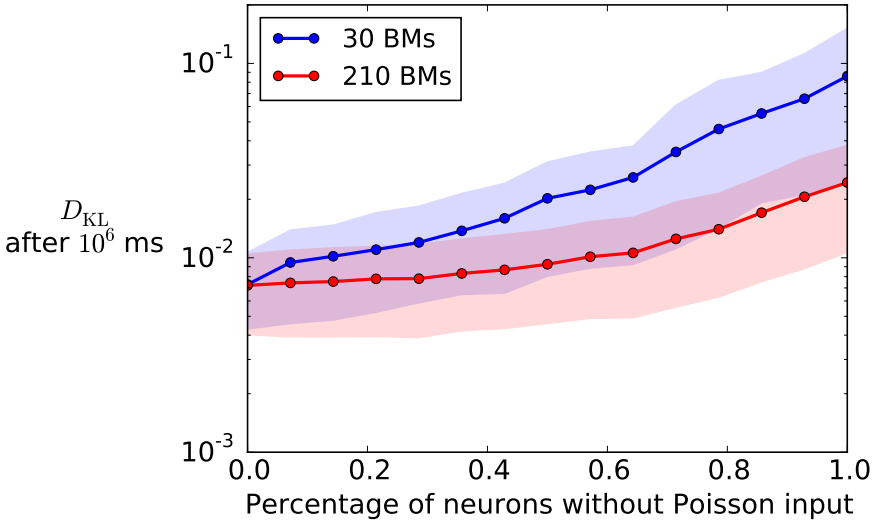

Figure 5.17: Mean D_{KL} after sampling for 10^6ms as a function of the percentage amount of cut off Poisson sources. Again, as in the previous sections, it turns out that the overall effect of cutting off external sources becomes less dominant for larger and sparser networks. Furthermore, even when only half the neurons still receive external noise, the effect on the sampling quality remains small. Surprisingly, cutting off all external noise still leads to good and stable results as long as the network is sparse enough. The parameters used in the simulation are the same as in the previous section. The simulation was repeated five times for different random seeds and the shaded areas represent the interval between the 15th and 85th percentile over all measured D_{KL} values.

Again, as in the previous sections, the sampling quality will slightly decrease if we cut off too many external noise sources. However, the change is quite small up to around

50-60% cut-off for a network of 210 3-neuron BMs. Comparing the similar setups in Fig. 5.5, 5.16 and 5.17, note that we arrive at similar D_{KL} values in the limit cases of all three different methods presented throughout the last sections. Even though this has not been studied in detail, it is a very reassuring observation that we obtain similar results for different methods of how to transition from a primarily Poisson-driven network to a network driven by intrinsic noise.

Finally, all results presented here demonstrate that it is indeed possible to either strongly reduce the Poisson input or even cut off all external noise sources without drastically impairing the sampling results. This represents a core result of this thesis and is a first convincing confirmation of the initial assumptions that have motivated it in the first place. In the following chapter, we will go one step further by analysing the dynamics of a network without any external Poisson sources in detail.

6 Stochastic LIF Networks Without External Noise

6.1 Deterministic Start of Networks

In the previous chapter, we were able to demonstrate that the amount of external Poisson sources needed for LIF sampling can not only be drastically reduced, but removed altogether. The stochasticity is then solely provided by the background activity of the network of BMs itself. However, it is interesting to ask how such a network reaches this state of stable operation without any external noise sources to kick-start the network in the first place.

This can be answered by looking at the case of $g = 4$, meaning the interconnections between BMs have four times stronger inhibitory weights than excitatory ones. Because the background noise is inhibitorily dominated, it will effectively generate a negative bias. In order to compensate for this, neurons that require a higher bias will receive a suprathreshold leak potential as a result of their calibration. Therefore, when the activity in the network tends to die out, these neurons will receive no background noise and they will be dominated by their suprathreshold leak, causing them to spike again, thereby kickstarting the activity of the entire network.

Setting up such a network with initial membrane potentials below the threshold, the start-up dynamics will only consist of the deterministic drift towards each neurons rest potential. As soon as the membrane potential of a neuron reaches the threshold, it emits a spike which is used as inhibitory noise input for other neurons in the network. Thus, in the beginning, the mean activity of the network will have a large burst starting from zero due to the deterministic drift, followed by a strong damping of the network activity due to the strong inhibitory connections (see Fig. 6.1). However, as discussed above, inhibition does not extinguish the network activity, but rather causes it to settle at the desired activity needed for LIF sampling. Note that due to the strong inhibition, neither the one-state with maximum mean activity τ_{ref}^{-1} nor the zero-state are fixed points of the network.

In fact, the network still starts for smaller values of g and stable sampling can be guaranteed even for excitatory dominated interconnections, i.e., $g < 1$. However, in these cases it is important to adjust the connectivity of the network to assure that enough neurons have their rest potential above threshold. These are then able to excite the remainder of the network with subthreshold rest potentials due to the strong excitatory

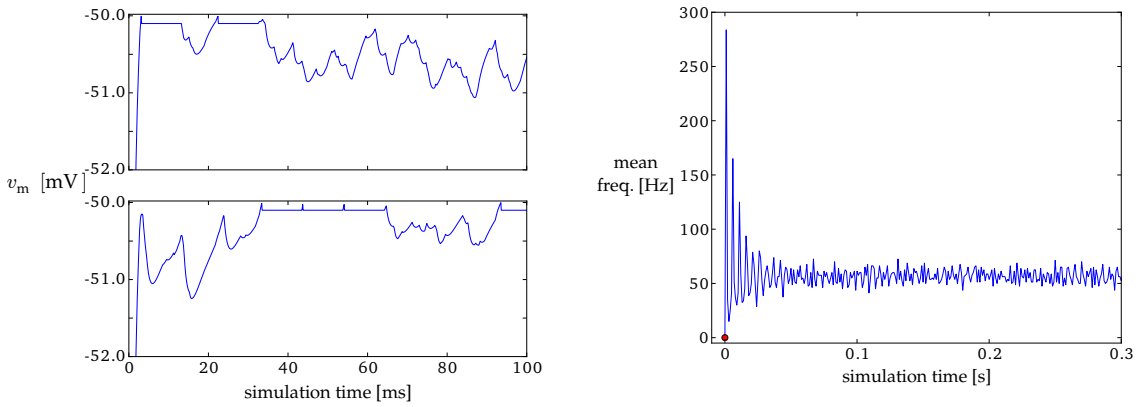

Figure 6.1: Deterministic start of a sea of BMs with $g = 4$. **(left)** Membrane potential of two network neurons. Initially, all membrane potentials are set below threshold. Since many neurons have a rest potential above threshold (here -50mV), they start drifting towards it. After some neurons have spiked (top) other neurons in the network are inhibited (bottom), keeping them from spiking in the beginning. **(right)** Because of the deterministic drifts, many neurons initially hit the threshold resulting in a strong burst in mean activity. Afterwards, the strong inhibitory spikes dampen the network until it reaches a stable mean activity.

connections (see Fig. 6.2). Hence, instead of the initial strong burst observed for $g > 1$, the network slowly starts by activating more and more neurons, therefore slowly increasing the mean network activity until the desired target frequency, predetermined by the underlying Boltzmann distributions, is reached (see App. 8.8.8).

For $g < 1$, we can also start the network by using strong Poisson input which gets slowly faded out as in Chap. 5.2.1. However, in cases where the network is not able to start on its own before, most neurons are now either silent or active, see Fig. 6.3. In the former case, after fading out the external Poisson sources, the network activity is not strong enough to keep all neurons active. This is similar to what happens in Fig. 6.2 as the network is not able to restart on its own after hitting the zero-state. However, if the connectivity is large enough, the network can also synchronize due to the strong excitatory connections and fall into the one-state, i.e., almost all neurons start bursting. It might be possible to avoid this behavior by improving the calibration quality. But since this method of starting heavily relies on strong external Poisson input, this has not been further investigated. Furthermore, even after improving the calibration, small variations towards higher activities might still lead to a synchronization of the network due to the strong excitatory interconnections. The parameters for the simulations discussed here can be found in App. 8.2.

Moreover, setting rest potentials above threshold can also be implemented via an ex-

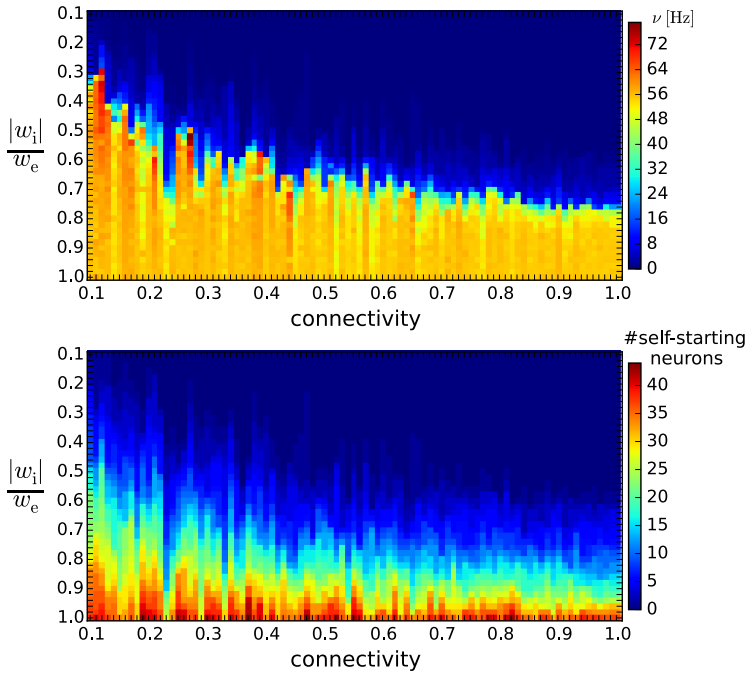

Figure 6.2: **(top)** Mean activity of a sea of BMs consisting of 10 10-neuron BMs after starting the network without external noise, dependent on $g = \frac{|\omega_i|}{\omega_e}$ (note the inverted axis) and the connectivity ϵ. Even for small values of g, the network is able to start and remain stable. Different to the case of $g > 1$, only a small fraction of neurons has its rest potentials set above threshold. Consequently, the network starts in a low activity state, gradually exciting more and more neurons until the target mean frequency is reached. However, there exists a distinct transition to $(g,\ \epsilon)$-values where the remaining self-starting neurons are too few and too weak to stably start the rest of the network. **(bottom)** Number of self-starting neurons for each configuration. In cases where the network remains silent, only a small number of neurons have their rest potential above threshold.

ternal current. It is therefore not surprising that at least sustained activity can be guaranteed. But it is still interesting that a purely deterministic mechanism can be employed to start the network – with stochasticity only arising from the spiking dynamics of the BMs. Further, for $g > 1$ there is a notable similarity to the Poisson fade-out in Chap. 5.2.1, where the network was also slowly placed into the correct activity state starting from a high mean activity.

Finally, it should be added that the kind of self-sustained network presented here is different from the ones investigated in *Kumar et al.* (2008); *Kriener et al.* (2014). There, recurrent networks of COBA/CUBA neurons with strong inhibitory connections and synaptic delays are used to put the network into a self-sustained asynchronous irregular state. These networks do not have "pacemaker" neurons, which can spike

without external input. Thus, strong rises in activity can lead to a sudden death of the network since the rest potentials are all set subthreshold.

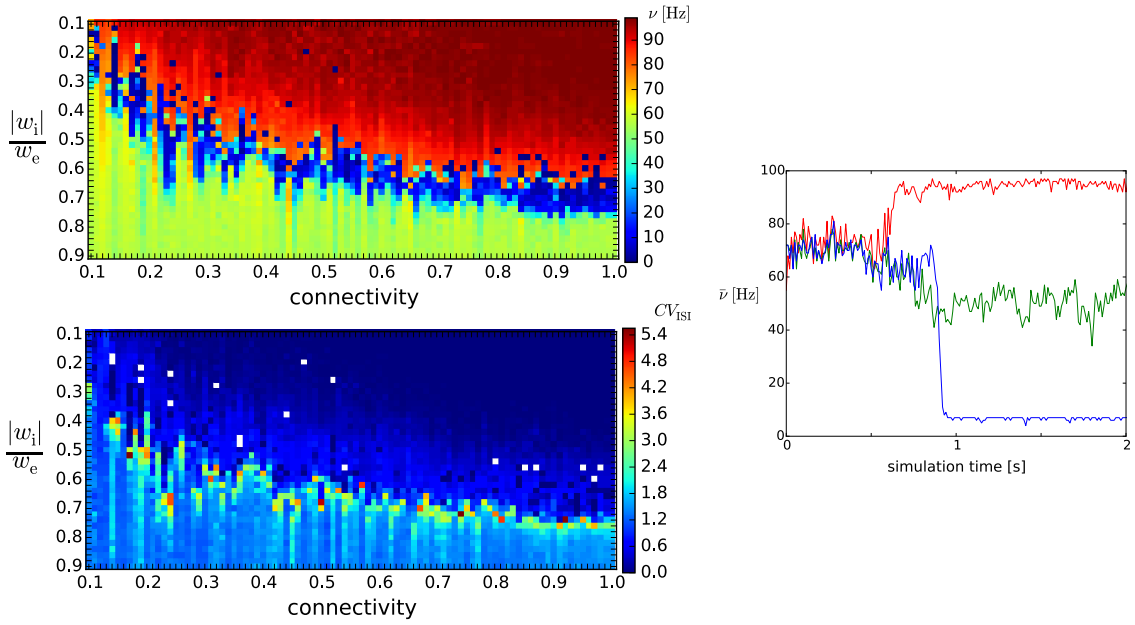

Figure 6.3: Same setup as in Fig. 6.2, but the sea of BMs is started with strong Poisson input that is slowly faded out. **(top left)** Three distinct cases can be seen: Stable network activity is observed for the same values of $g = \frac{|\omega_i|}{\omega_e}$ and ϵ as in Fig. 6.2 (green-yellow area). However, decreasing g leads to less neurons having their rest potentials above threshold until the remaining ones are not strong enough to keep the entire network alive (blue area). By further decreasing g, the excitatory connections become so strong that short rises in activity kick the network in a trivial bursting state where the excitatory input that each neuron observes is much stronger than initially assumed during calibration (red area). **(bottom left)** Mean CV_{ISI} of the network. If the network runs stably, the activity is irregular with CV_{ISI} values mostly above 1 (light blue area). In the bursting state it is 0 as expected (dark blue area). The CV_{ISI} cannot be calculated if the network does not spike at all, marked in white. **(right)** Mean activity of the network during start-up. The colors represent the three cases of stable activity (green), trivial bursting (red) and network death (blue) after removing the external Poisson noise.

6.2 Removing Correlations by Training

In Chap. 5, it was shown that the sampling quality increases for larger and sparser networks. However, this process becomes very tedious as we have to increase the network

size over orders of magnitudes to achieve notable changes. Another approach is inspired by recent work of I. Bytschok and M. Petrovici (personal communication), who showed that shared-input correlations, which naturally occur if neurons of a single BM get noise from a shared pool of Poisson sources, can be compensated by adjusting the Boltzmann weights and biases accordingly. Finding the correct weights and biases can be done by training, for instance with CD (see Chap. 2.4.2). In the following section, CD will be utilized to drastically increase the sampling quality of LIF BMs without external Poisson sources. The learning rate was set to

$$\eta_{\mathrm{CD}}(s) = \frac{400}{s + 2000} \tag{6.1}$$

after conducted parameter sweeps, where s counts the number of training steps. It was further observed that initial steps should be rather large to allow the training algorithm to correct for strong correlations. Note that the noise interconnections remain unchanged throughout the training. Detailed simulation parameters can be found in App. 8.2.

6.2.1 Special Case: Boltzmann Machines with Zero Weights

Since the quality of LIF sampling is mainly reduced due to network correlations, an intuitive ansatz is to further train each BM on its target distribution. This way, the distorted correlation structure between neurons of the same BM coming from shared input and network-wide correlations will be corrected.

To check if this is really the case, a simple setup of 40 10-neuron Boltzmann machines without Poisson sources, all with internal weights set to 0, is used. With Poisson background, these BMs would have no internal connections. But since in our setup neurons have to share background sources, the synaptic weights in these BMs have to be nonzero to compensate for their shared-input correlations. To this end, each BM was first calibrated as in the previous chapter and used to sample for 10^5ms. The samples were then used to update all BMs at once with the CD update scheme. In total, 1200 CD updates were performed.

In Fig. 6.4, the correlation coefficients between the states of 500 randomly selected neuron pairs belonging to the same BM have been plotted in a histogram for every update step. As we see, even after the first 100-200 update steps, the width of the correlation distribution decreases substantially. After 1200 steps, the correlation coefficients are actually distributed almost identically to the ideal Poissonian case. For Poisson sources, the difference from zero correlation appears due to (i) the finite time over which the correlation coefficients were calculated and (ii) the usage of sampled probabilities in the calculation, which depend on the LIF sampling quality. The obtained correlation coefficients for both cases range from around -0.02 to 0.02 and are therefore rather low compared to the initial correlations in the untrained sea of BMs. This demonstrates that CD can be used to further reduce harmful correlations without increasing the network size.

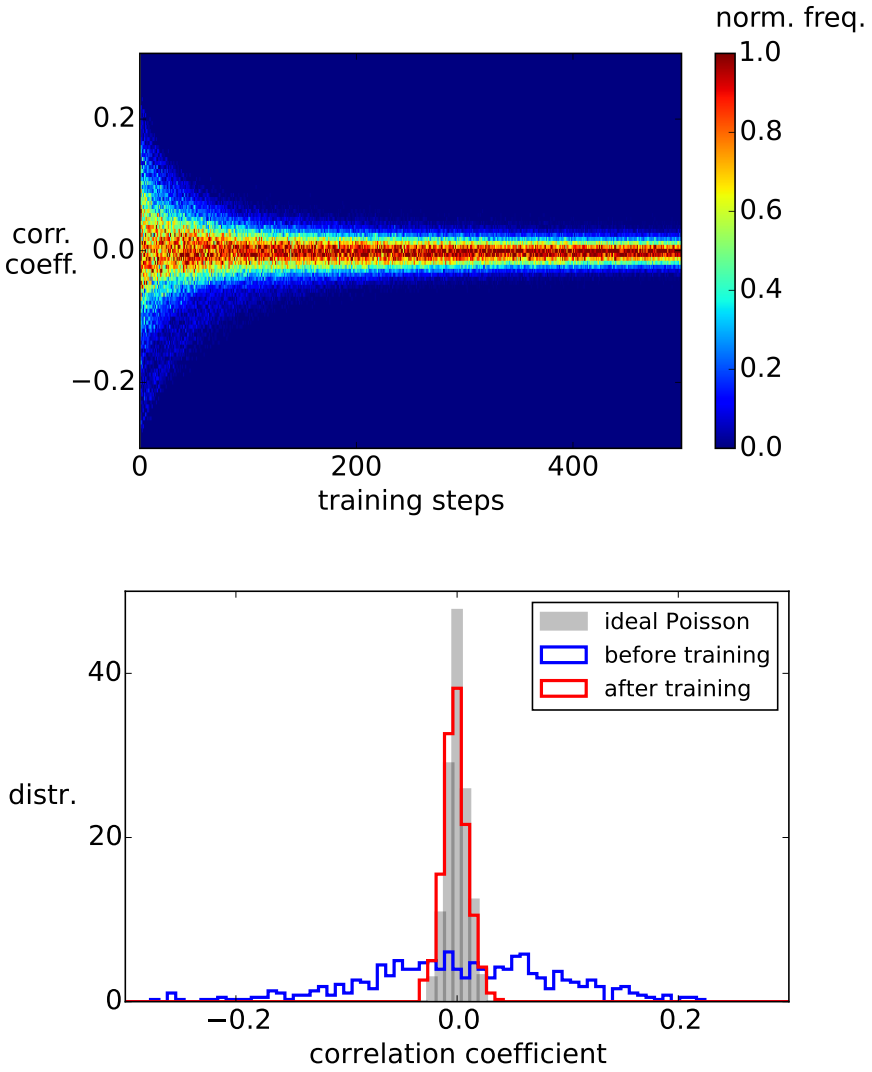

Figure 6.4: Correlation of states of 500 randomly selected pairs of neurons. Each pair consists of neurons belonging to the same BM. **(top)** The normalized distribution of correlation coefficients is shown for all training steps. Each distribution is normalized on its maximum value. Already after a few hundred training steps, the network correlations have been drastically reduced. **(bottom)** Histogram of correlations before and after training. After 1200 training steps, the observed correlations are on the same order of magnitude as in the ideal Poissonian case with correlation coefficients ranging from around -0.02 to 0.02.

6.2.2 General Case: Boltzmann Machines with Random Weights

After assuring that training indeed reduces unwanted correlations in the network, we can use CD to improve the sampling quality of a general setup. For this, again a network of 40 10-neuron Boltzmann machines was used, but this time with finite weights randomly drawn from Eq. 5.1a with $W_0 = 2.0$. The number of inputs each neuron receives from the network was set to 20, corresponding to a connectivity of around 5%.

If we compare the D_{KL} curves before and after training with those of the BMs driven by ideal Poisson noise before and after training, we can clearly see that the difference between the sea of BMs and the ideal case shrinks drastically (see Fig. 6.5). Training with Poisson noise also improves the sampling quality because the rules for translating the theoretical Boltzmann parameters to LIF parameters are only approximations (see Section 2.3.2). In this case, the samples for training were again obtained after 10^5ms. However, it is possible to obtain results of similar quality for smaller sampling times, for instance only 10^4ms, as demonstrated in App. 8.8.7. This decreases the total simulation time needed for the whole training process.

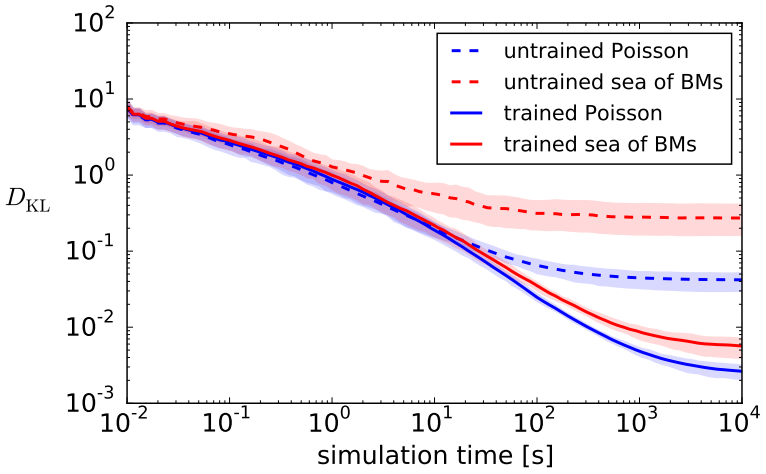

Figure 6.5: Mean D_{KL} of a network of 40 10-neuron BMs driven by Poisson noise only (blue) and sea noise only (red) before (dashed) and after training with CD (line). After training, both the sea of BMs and the Poisson-driven network perform almost equally well. The sampling quality of the sea of BMs might even be further improved by continuing the training. Also note that CD improves the final mean D_{KL} obtained from the sea of BMs over two orders of magnitude. The shaded areas mark again the 15th and 85th percentile over all single D_{KL} curves.

Looking at the training results of randomly chosen weights and biases (see Fig. 6.6), we can see that their evolution is very stable and smooth even though all BMs get

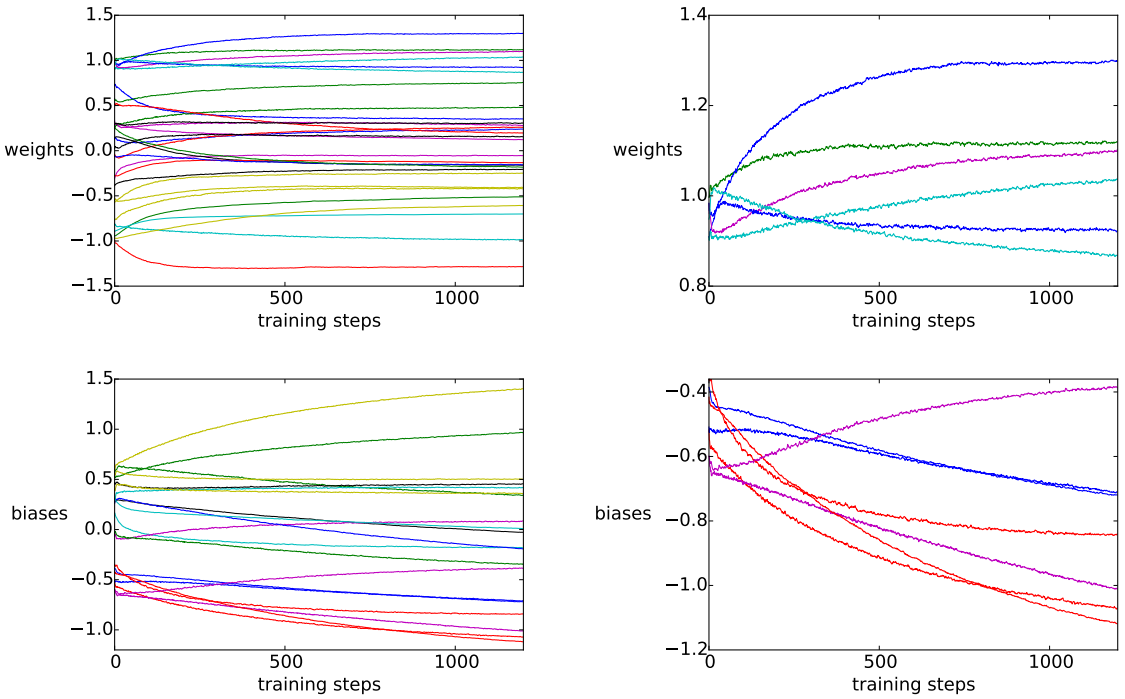

Figure 6.6: Randomly selected weights and biases during training from the sea of BMs shown in Fig. 6.5. The plots on the right are enlarged versions of a small region in the respective left plots. Both weights and biases evolve very smoothly and are stable during training. Even though most weights have already converged, some biases can still be optimized. This might further improve the sampling quality of the sea of BMs.

updated simultaneously in each update step. This is not at all obvious, as updating all BMs at the same time leads to slight changes in the background noise. However, these seem to have a negligible influence on the training since all weights and biases converge without irregularities, as for example jumps between successive training steps. Also note that not all biases have converged yet, so further training might still improve the overall LIF sampling quality.

This confirms that the intrinsic noise of a network of BMs can be used to replace external Poisson sources without losing any significant amount of sampling quality in terms of D_{KL}. This result is not limited to networks with $g = 4$ and works equivalently well for $g < 1$, as demonstrated in App. 8.8.8. This represents the second major result of this work. While in the previous chapters we have shown how, in the thermodynamic limit ($N \to \infty$), large ensembles of BMs can provide each other with uncorrelated noise with virtually no external input needed, here we have shown that training allows very practical implementations of these results, with the required N being as small as 40 – or even lower, as we discuss in the following.

6.3 Towards Small and Fully Connected Networks

Training the sea of BMs with CD opens up the possibility to use small and fully connected instead of large and sparse networks. This was not possible before, since network-wide correlations rendered individual BMs useless and the only way to cope with this problem was to make the network sparser by increasing the number of BMs. Additionally, CD lifts the necessary restriction between sparseness and the desired minimal mean input each neuron has to get from the sea to reach the high-conductance state. That's because we have now access to higher connectivities ϵ without negatively affecting the LIF sampling quality of the network BMs. Thus, even for very small networks, for instance 4 BMs with 10 neurons each, all neuron inputs can reach the desired noise frequency needed for sampling by choosing a sufficiently large connectivity. This will be demonstrated in the following sections for several exemplary cases.

6.3.1 Setup 1: 11 Boltzmann Machines with 3 Neurons Each

First, a simple setup consisting of 11 BMs with 3 neurons each was investigated. The number of BMs chosen here is the minimal number at which stable network runs were observed throughout the whole training (see Section 6.3.2 for details). The connectivity was set to 93.33%, leading to an average input of 28 spike trains per neuron which equals an average excitatory and inhibitory input frequency of 700Hz each.

The simulation was repeated for three different cases:

1. A general setup with $g = 4.0$, $\eta = 0.5$ and $\frac{\sigma_\mathrm{W}}{\mu_\mathrm{W}} = 1.0$ (shown in red in Fig. 6.7).
2. A purely inhibitory setup with $\eta = 1.0$ and $\frac{\sigma_\mathrm{W}}{\mu_\mathrm{W}} = 1.0$ (shown in green).
3. A setup with fixed inhibitory and excitatory weight strengths, i.e., $\eta = 0.5$ and $\frac{\sigma_\mathrm{W}}{\mu_\mathrm{W}} = 0.0$ (shown in blue).

Before training, all three cases perform badly with final D_KL values of around 10^{-1} to 10^{0}. The sampling quality of case 2. is especially bad, since due to the lack of mixing between excitatory and inhibitory synapse types, neurons sharing the same input spikes have very similar membrane traces and thus spike dynamics.

However, after training, case 1. and 3. perform very well with D_KL values on the order of 10^{-4}. This is a drastic improvement of the sampling quality over three orders of magnitude. Furthermore, not only is the sampling result very impressive, we have to keep in mind that the noise provided to each neuron originates from 93.33% of the whole network. Thus, every neuron sees approximately the same input spike trains. If all noise synapses are inhibitory, this leads to poor sampling qualities as observed for case 2. Also, training becomes very noisy and convergence towards ideal Boltzmann parameters that compensate for network-wide correlations is not guaranteed (see Fig. 6.8). But still,

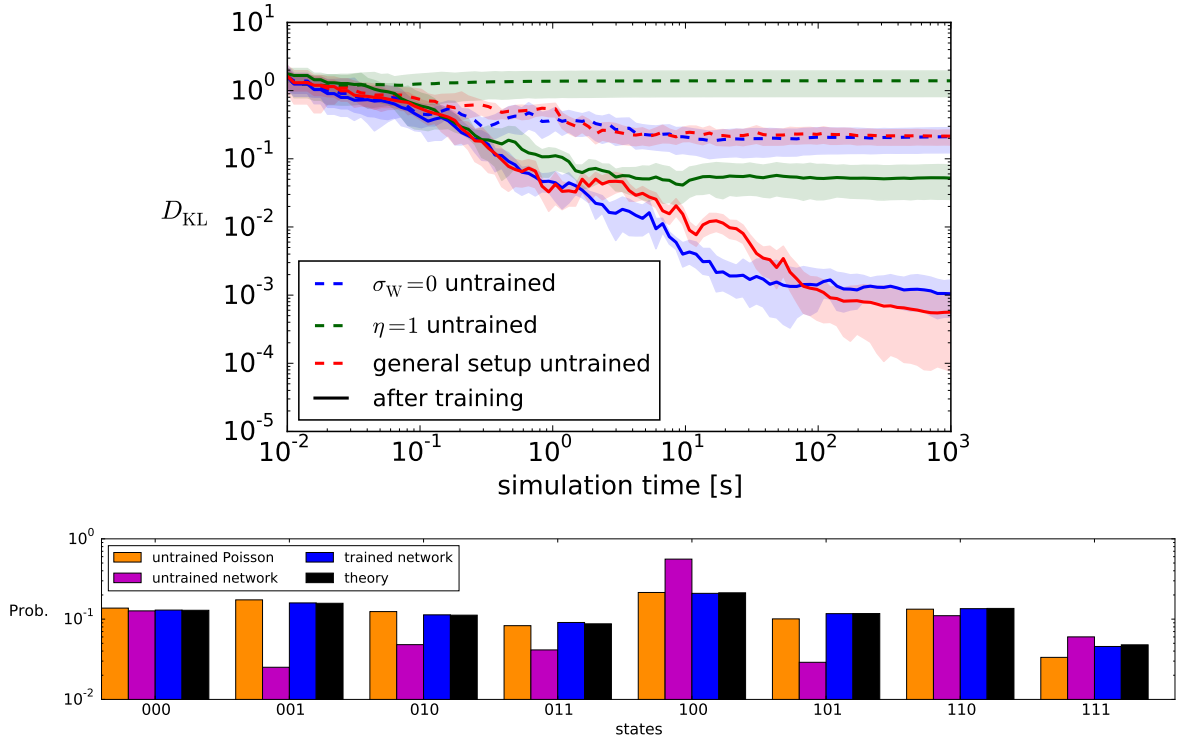

Figure 6.7: **(top)** Mean D_{KL} of the sea of BMs before (dashed) and after (line) training for different setups. Even though the network is very small, CD improves the final mean D_{KL} over three orders of magnitude. Changing the width of the interconnection weight distribution σ_{W} does not notably change the results (blue). However, taking away the randomness of synapse types, for example by making all noise weights negative (green), seriously affects the sampling quality after training. The general setup shown in red is defined via $\mu_{\mathrm{W}} = \sigma_{\mathrm{W}} = 0.001\mu\mathrm{S}$ and $\eta = 0.5$. **(bottom)** Illustration of the obtained state distribution of a single network BM, taken from the general network setup. Before training, the differences between theoretical and sampled state probabilities can be large. But after training, the theoretical distribution is very well approximated by the sea of BMs.

because the absolute weight values of the noise connections are chosen randomly, which also weakens correlations, the training does improve the D_{KL} to around 10^{-2}.

In case 1. and 3., the initial overall correlation in the network is reduced by the randomized noise synapse types. Hence, even though two neurons might get exactly the same input spike trains, assigning each of the incoming spike trains with a random synapse type will result in both neurons having different membrane traces. This mitigates the correlations between neurons in the network and facilitates convergence towards an appropriate set of theoretical Boltzmann parameters.

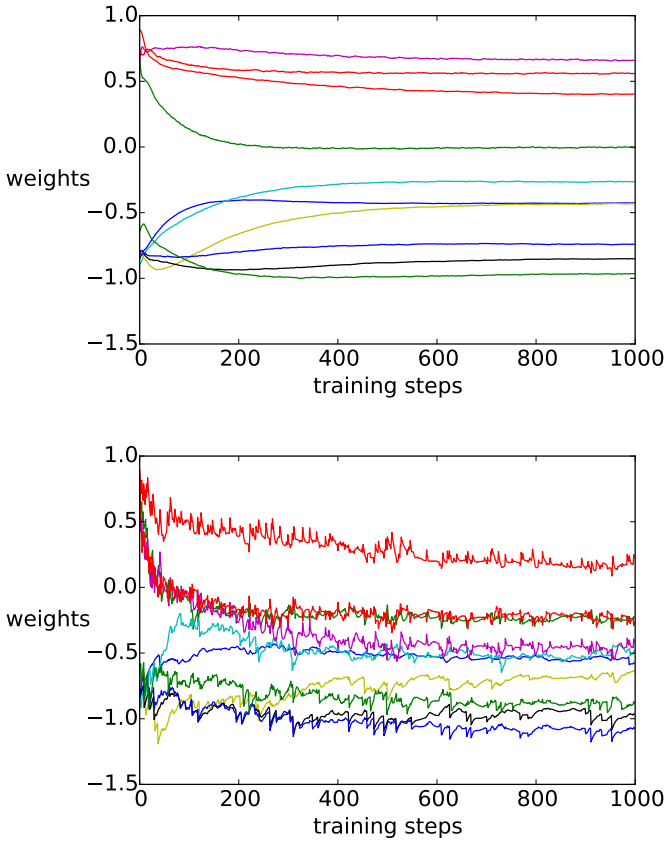

Figure 6.8: Randomly selected weights during training for the general case presented in Fig. 6.7 with $\eta = 0.5$ (top) and $\eta = 1.0$ (bottom). If the noise synapse types are fixed to only one type, the weight updates become very abrupt and noisy. Furthermore, a lot of weights are not able to converge to a value much different from the initial one. Thus, the strong initial correlations cannot be compensated by training with CD. If we choose $\eta = 0.5$, initial correlations are weaker in the network and CD is able to stably converge to a good solution.

As a minimum requirement, it has been observed that, after calibration, the network should at least be able to run stable in a non-trivial state, i.e., it should not fall into the zero or one-state (all neurons silent or all neurons bursting). Otherwise the training process is not guaranteed to converge to a good solution.

Finally, it should again be highlighted that two major properties of the interconnection weight matrix are crucial for CD to converge properly. Firstly, as already discussed, the synapse types of all interconnection weights have to be chosen randomly to weaken shared-input correlations. Secondly, the weight matrix is not symmetric but asymmetric, reducing the amount of direct feedback loops drastically. For instance, in App. 8.8.9 the 40 BM network of the previous section with 5% connectivity was implemented, but with a symmetric interconnection weight matrix. Thus, the whole sea of BMs is again a large BM and every neuron provides noise to neurons it gets noise from. This makes training the network very hard since symmetric connections introduce strong correlations between neurons of different BMs. Again, the weights change only slightly during training and therefore, the final mean D_{KL} of the network does not improve considerably as well.

6.3.2 Setup 2: Small Networks with 10 Neurons Each

Applying the knowledge gained in the previous section, we can easily go to small networks of larger BMs with 10 neurons each. Interestingly, we can even train a fully connected network consisting of only two BMs. However, such small networks have a serious drawback: They are susceptible to getting stuck in a subspace of the entire possible state space. More precisely, it can happen that the network will at some point start traversing almost deterministically through a small and closed portion of the entire available states only. To get out of this subspace, an external random kick is needed, which, however, is not available to the network.

This is demonstrated in Fig. 6.9, where the D_{KL} of a single sampling run of the network after 958 training steps is shown. First, the training obviously increased the LIF sampling quality of the two BMs inside the network, even though the amount of available noise sources is limited to the 10 neurons of the neighbouring BM. But after running the network for around 30s, it gets stuck, immediately leading to an increase in D_{KL} as states in the subspace become overemphasized. This is also demonstrated in Fig. 6.10, where the sampled state distributions are shown after sampling for 31s and 100s. After 31s, the sampled distributions look similar to the desired theoretical target distributions and capture their overall structure astonishingly well. However, after 100s, the sampled distributions are reduced to a few peaks, showing that both BMs started traversing through a small and fixed number of states only. During training, this has no significant influence on the weight and bias updates. But it often happens while sampling for the pairwise and marginal probabilities that the network gets stuck, leading to a spontaneous drop in D_{KL}. This is shown in Fig. 6.11 and can be used as an easy way to identify unstable network configurations during training.

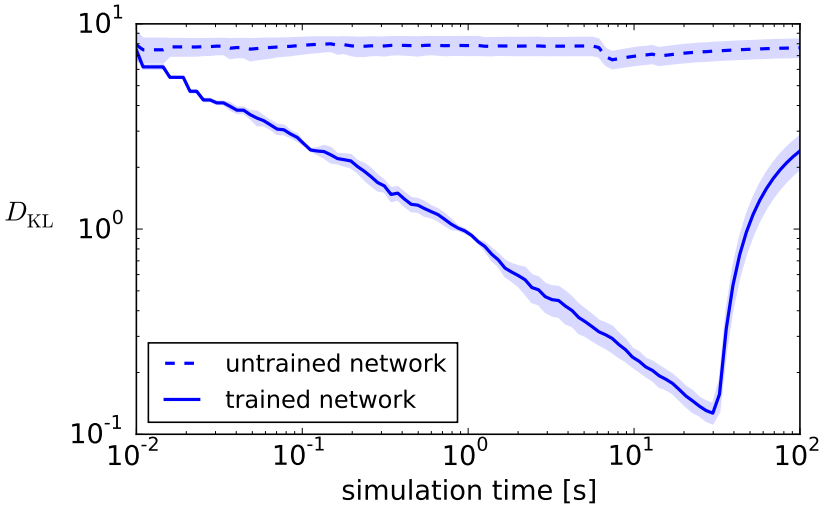

Figure 6.9: Network consisting of 2 BMs with 10 neurons each sampling before training (dashed) and after 958 training steps (line). CD does indeed improve the sampling quality for this minimalistic network. However, due to the limited size and therefore lack of randomness, the network can get stuck in a small subspace of states, which leads to an immediate increase of D_{KL}.

We can get rid of this effect by increasing the number of BMs in the sea. This way, we will have more noise sources available and the chance of getting stuck in a certain subspace of states is reduced. Already for 4 BMs with 10 neurons each, the network remains stable throughout the whole training process. Note that - even though the final D_{KL} reached with this setup is worse compared to the ideal Poissonian one of around 10^{-3} shown in Fig. 6.5 - it improves over two orders of magnitude from 10^0 before training to 10^{-2} after training (see Fig. 6.12). Furthermore, as can be seen in Fig. 6.13, all four sampled distributions capture the important features of their target distributions very well.

Again, this demonstrates that using CD opens up the possibility for small and strongly connected networks of BMs which can operate reliably without any external noise. But still, the size of the network can have a large influence on the LIF sampling performance, as for example very small networks can get stuck in certain states due to a lack of randomness.

Nevertheless, it is rather surprising that LIF sampling can be realized without any external Poisson sources, even for small and non-sparsely connected networks. Thus, it is indeed possible to implement a network of well-performing BMs without having to utilize any external noise inputs.

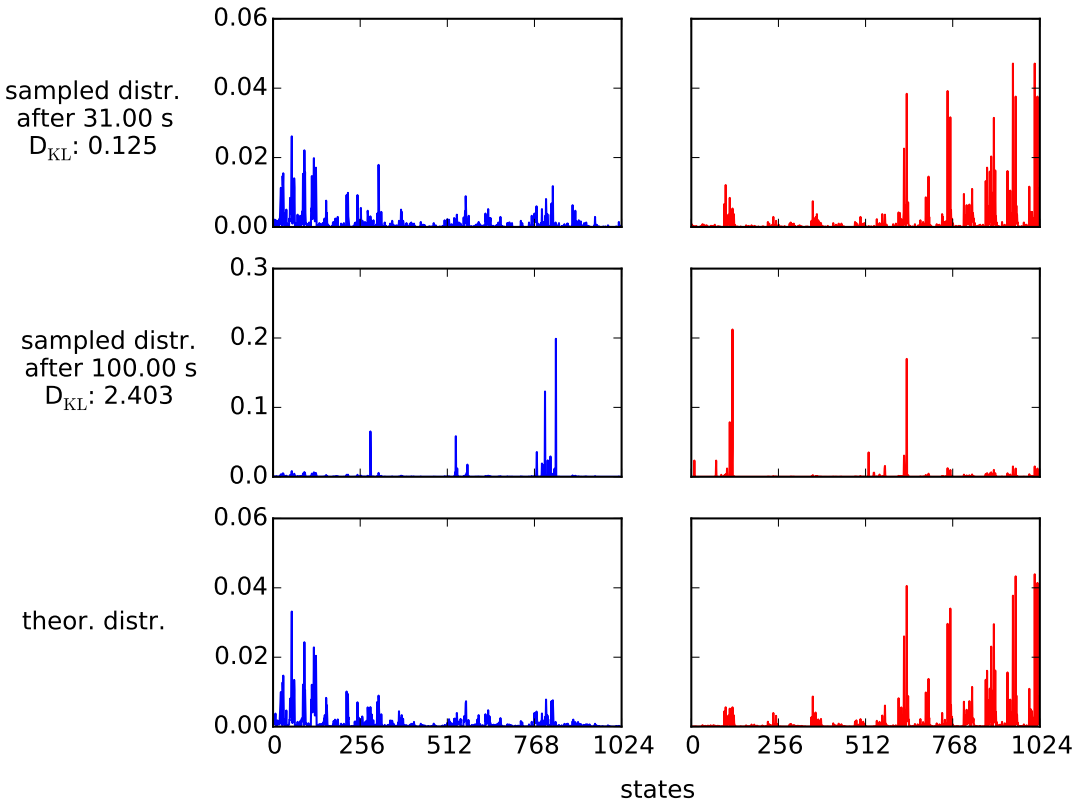

Figure 6.10: State distribution of the 2^{10} possible states of the two BMs. After sampling for 31s (top), the overall structure of the sampled distributions agrees remarkably well with the theoretical one (bottom). Note that no external noise sources are used and every neuron gets its noise from the 10 neurons of the neighbouring BM. Due to the network being so small, the neurons can get stuck and start traversing only through a small subspace of states. This can be seen very well after sampling for 100s (middle) as only a very small number of states is highly emphasized in the sampled distribution.

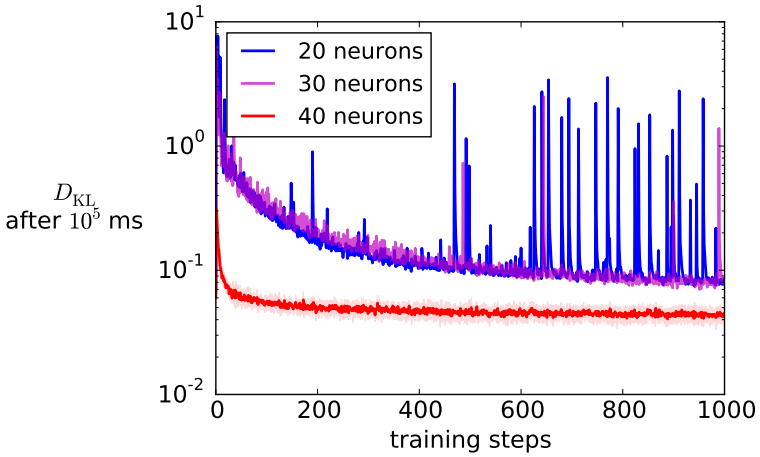

Figure 6.11: Final mean D_{KL} obtained during training for a network of 2 BMs (blue), 3 BMs (magenta) and 4 BMs (red). The already mentioned instability of the network getting stuck also appears during training and leads to spikes in the D_{KL} curve. For 3 BMs, this effect is highly reduced but can still appear. Increasing the number of BMs to 4 eventually leads to completely stable network runs during training. The red shaded area marks the interval between the 15th and 85th percentile. The errors for 20 and 30 neurons are rather small and have been excluded to guarantee readability.

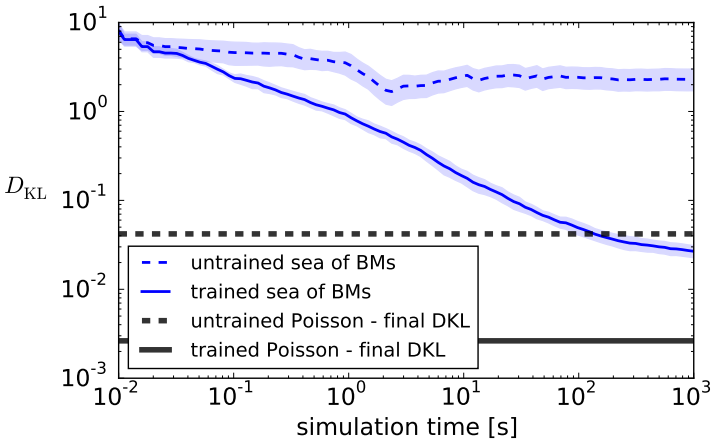

Figure 6.12: Sampling quality of a 4 10-neuron BM network (dashed) before and (line) after training with CD. Even though the network is small and fully connected, the sampling quality can be improved over two orders of magnitude via training. This shows that the initial distorting correlations coming from the network interconnections can be compensated with CD even in case of small networks. However, the final D_{KL} is worse than for sampling with Poisson sources post-training.

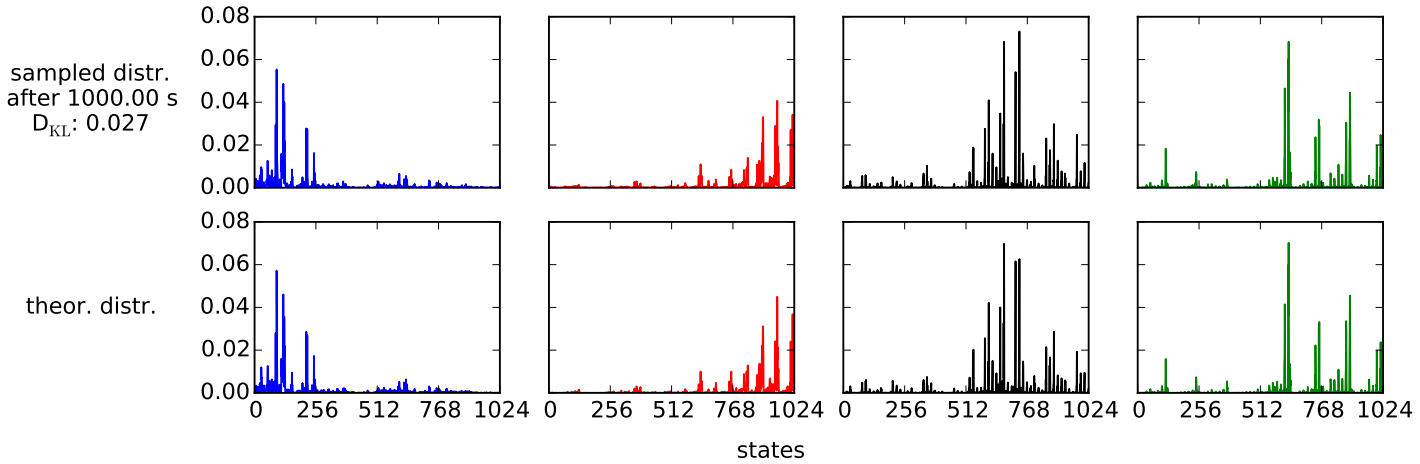

Figure 6.13: Comparison of the sampled distribution after 1000s and the underlying theoretical Boltzmann distribution for a network of 4 10-neuron BMs. As we can see, the sea of BMs captures the structure of the theoretical distributions very well. Again, note that no external noise sources were used here and stochasticity was only provided by the BMs themselves.

7 Summary and Outlook

Throughout this thesis, we gradually developed and evaluated a framework to eliminate external Poisson input for LIF sampling. In the end, we succeeded in completely cutting off all external stimuli and demonstrated that even small networks of Boltzmann machines can exploit the ongoing network activity as an intrinsic noise source.

In the first two chapters, the focus was set on the question of whether spike trains originating from a pool of Boltzmann machines can be utilized as a replacement for Poisson sources. To find an answer, instead of running a single Boltzmann machine with Poisson spikes, we used spike trains coming from an adequately sized pool of Boltzmann machines. As opposed to a Poisson source, the spike train generated by a Leaky Integrate-and-Fire neuron is colored instead of white, i.e., they have a short-time correlation structure due to consecutive spikes having a minimal distance of a single refractory period. Furthermore, since neurons of the same Boltzmann machine are connected via synaptic weights, additional cross-correlations between noise spike trains connected to different neurons are introduced, leading to a distortion of the distribution sampled from.

However, not only did it turn out that the coloring of the noise is completely unproblematic for the sampling quality, it was also demonstrated that undesired cross-correlations can be mitigated by increasing the number of noise inputs per neuron and choosing the synapse type of each connected noise spike train randomly.

These results allowed us to go one step further. Instead of looking at a single, completely isolated Boltzmann machine, we chose an open systems approach and embedded the Boltzmann machine into a sea of Boltzmann machines. This is analogous to the introduction of the *canonical ensemble* in statistical physics, which places a previously isolated physical system into contact with a so-called *heat bath* with fixed temperature T, commonly reflecting the environment. By interacting with this heat bath, the system can undergo state changes induced by thermal fluctuations on an energy scale of the order $k_\mathrm{B}T$, where k_B denotes the *Boltzmann constant*.

In our case, Boltzmann machines that were previously treated as completely isolated entities are embedded into a network of Boltzmann machines. Here, the network itself acts as the natural environment or heat bath, and simply by allowing each Boltzmann machine to interact with this environment, as in the case of the canonical ensemble in statistical physics, stochastic behavior arises in a completely natural way.

This way, we end up at an interesting duality: Spike times acting as carriers of function and information, but also as noise input for other Boltzmann machines in the network, providing the necessary stochasticity to guarantee functionality in the first place. To our knowledge, this is the first implementation of small, functional neural networks which operate without external idealized Poisson sources, but use the background activity of the surrounding neural networks to enable stochastic computations.

At this point, it should be noted that the sea of Boltzmann machines is actually a completely deterministic system. Every network start with identical initial conditions will result in the same sequence of spikes and states. Stochastic behavior arises due to small changes in the sea having a large impact on the network dynamics and hence on future network states. Therefore, the sea of Boltzmann machines is a deterministic *chaotic system* and can be compared with a deterministic *pseudorandom number generator*, like the ones used in software simulations to generate Poisson noise, instead of a source of true randomness.

Finally, the LIF sampling theory is the centerpiece of all results presented throughout this thesis. First, it allows us to use spike-based neural networks to sample from binary Boltzmann distributions, giving each spike time a concrete, functional purpose. Furthermore, neurons are pushed into a stochastic regime of operation by bombarding them with Poisson spikes. Coming from biology, it is completely natural to replace this artificial bombardment with spikes from a large pool of functional networks.

It also paves the way to the implementation of Leaky Integrate-and-Fire Boltzmann machines which can then be trained with contrastive divergence, enabling us to realize small and non-sparsely connected networks of Boltzmann machines that run stably and sample from their respective target distributions without any external noise sources.

Starting from these results, we are now able to tackle several interesting issues:

- **A sea of large-scale, multifunctional networks:**
  One natural progression is towards larger (restricted) Boltzmann machines that are not set to sample from predefined theoretical distributions anymore, but are trained on small real world problems, as for instance image classification. This allows us to study the impact on functionality due to unforeseen changes or deficiencies in the background activity. Moreover, biologically inspired plasticity rules might be used to allow the sea of Boltzmann machines to adjust to such defects and preserve functionality.

- **Inducing a change in function by changing the background activity:**
  Related to the previous point is the following idea: Might a sudden change in the background activity, for instance because some neurons of one or two Boltzmann machines were clamped, induce a useful functional change or 'association' in some of the remaining network Boltzmann machines? This would further reduce the

artificial distinction between noise and information in the network, since noise connections can induce functional changes as well.

- **Getting rid of calibrations:**
  Up until now, every neuron had to be calibrated at least on some approximation of the network noise it will see during sampling. For large networks, it is unrealistic to first run every Boltzmann machine with Poisson noise to get these approximations. However, it may be the case that these calibrations do not need to be accurate for learning to converge. Thus, a generic activation function might be used for all neurons in the ensemble.

- **A sea of Boltzmann machines on HICANN:**
  After improving the calibration scheme, it is theoretically possible to start small-scale experiments with the sea of Boltzmann machines on HICANN. Even though *Kungl* (2016) demonstrated that LIF sampling is in principle possible on HICANNv4, the external bandwidth is not high enough to increase the number of neurons to values significantly larger than two. Since the sea of Boltzmann machines is completely independent of any external noise input, it directly enables hardware implementations of larger LIF-based Boltzmann machine ensembles.

- **Adding plasticity to noise connections:**
  In this thesis, the network interconnections providing intrinsic noise were implemented as static synapses. However, adding short-term plasticity to these synapses or including them into the training process, for example by adding *Spike Timing Dependent Plasticity*, might help in the aforementioned issues. Moreover, this would again reduce the artificially introduced distinction between synapses connecting neurons of the same and synapses connecting neurons of different Boltzmann machines.

- **A sea of Boltzmann machines as a Boltzmann machine:**
  Preliminary simulations have shown that it is much harder to train a sea of Boltzmann machines with symmetric weights than with asymmetric ones. In fact, up until now, only the asymmetric case was successfully trained with contrastive divergence. However, it should be further investigated if changes in the training algorithm might improve sampling results for the symmetric case as well.

To summarize, it was successfully demonstrated that LIF-based Boltzmann machines can be used as noise sources for functional networks, in our case again Boltzmann machines, replacing commonly used idealized Poisson sources. Moreover, Boltzmann machines can be interconnected to large networks that utilize intrinsic network spikes as background noise, eliminating any external Poisson sources. This opens up many interesting possibilities for further studies with networks that function without any external stimuli.

8 Appendix

8.1 Acronyms

AI Asynchronous Irregular

BM Boltzmann Machine

CD Contrastive Divergence

COBA Conductance-Based

CUBA Current-Based

CV Coefficient of Variation

DKL Kullback-Leibler Divergence

HICANN High Input Count Analog Neural Network

ISI Interspike Interval

LIF Leaky Integrate-and-Fire

MNIST Mixed National Institute of Standards and Technology

NEST Neural Simulation Tool

ODE Ordinary Differential Equation

OU Ornstein-Uhlenbeck

PSP Post-Synaptic Potential

SBS Spike-Based Sampling

SLURM Simple Linux Utility Resource Management

STP Short-Term Plasticity

TSO Tsodyks-Markram Model

8.2 Parameters

The integration time step used for simulations was initially set to $dt = 0.01$ms, however it turned out that the quality of the results is not impaired by choosing $dt = 0.1$ms, which leads to an immense speedup. In most D_{KL} plots, uncertainties are shown as percentiles instead of standard deviations to avoid values below 0 on the logarithmic scale.

The neuron parameters used in various simulations and some specific simulation setups are given in the following tables.

Table 8.1: COBA LIF neuron parameters used throughout Chap. 3 to 5.1.

c_{m}	0.2 nF	membrane capacitance
τ_{m}	0.1 ms	membrane time constant
$E^{\mathrm{e}}_{\mathrm{rev}}$	0.0 mV	exc. reversal potential
$E^{\mathrm{i}}_{\mathrm{rev}}$	-100.0 mV	inh. reversal potential
u_{thresh}	-50.0 mV	threshold potential
$\tau^{\mathrm{e}}_{\mathrm{syn}}$	10.0 ms	exc. synaptic time constant
u_{rest}	-50.0 mV	rest/leak potential
$\tau^{\mathrm{i}}_{\mathrm{syn}}$	10.0 ms	inh. synaptic time constant
u_{reset}	-50.01 mV	reset potential
τ_{refrac}	10.0 ms	refractory time
I_{offset}	0.0 nA	offset current
$w_{\mathrm{inh/exc}}$	0.001 μS	Poisson noise weights

Table 8.2: Theoretical Boltzmann parameters used for the network Boltzmann machines generating the noise spike trains in Chap. 3. The mean frequency of the neurons were changed by increasing the bias. Note that for very low activities, a lot of Boltzmann machines (#BMs = number of BMs used) are needed for each network to achieve the desired frequency of at least 600Hz.

bias b	$\bar{\nu}$ $[\tau_{\mathrm{ref}}^{-1}]$	#BMs
-4.0	$(1.42 \pm 0.03)\%$	330
-2.0	$(14.14 \pm 0.15)\%$	50
0.0	$(52.95 \pm 0.26)\%$	12
2.0	$(89.08 \pm 0.10)\%$	6
5.0	$(99.97 \pm 0.01)\%$	6
W_{12}	-0.36	
W_{13}	0.59	
W_{23}	0.22	

Table 8.3: COBA LIF neuron parameters used throughout Chap. 5.2 to 6.

c_{m}	0.2 nF	membrane capacitance
τ_{m}	1.0 ms	membrane time constant
$E^{\mathrm{e}}_{\mathrm{rev}}$	0.0 mV	exc. reversal potential
$E^{\mathrm{i}}_{\mathrm{rev}}$	-100.0 mV	inh. reversal potential
u_{thresh}	-50.0 mV	threshold potential
$\tau^{\mathrm{e}}_{\mathrm{syn}}$	10.0 ms	exc. synaptic time constant
u_{rest}	-50.0 mV	rest/leak potential
$\tau^{\mathrm{i}}_{\mathrm{syn}}$	10.0 ms	inh. synaptic time constant
u_{reset}	-50.1 mV	reset potential
τ_{refrac}	10.0 ms	refractory time
I_{offset}	0.0 nA	offset current
$w_{\mathrm{inh/exc}}$	0.001 μS	Poisson noise weights

Table 8.4: Simulation parameters used in Chap. 6.1.

#BMs	10
#neurons per BM	10
η	0.5
μ_{W}	0.001 μS
σ_{W}	0.001 μS
weight distr.	$2.0 \cdot \big(\mathrm{beta}(0.5,\ 0.5) - 0.5\big)$
bias distr.	$1.2 \cdot \big(\mathrm{beta}(0.5,\ 0.5) - 0.5\big)$
sampling time	10^4ms

Table 8.5: Simulation parameters used in Chap. 6.2. The sampling time is used while calculating the update steps. No Poisson sources are used.

#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 $2.0 \cdot \big(\mathrm{beta}(0.5,\ 0.5) - 0.5\big)$
bias distr.	$1.2 \cdot \big(\mathrm{beta}(0.5,\ 0.5) - 0.5\big)$
calibration time	10^5ms
sampling time	10^5ms
training steps	1200

Table 8.6: Simulation parameters used in Chap. 6.3.1. The sampling time is used while calculating the update steps. No Poisson sources are used.

#BMs	11
#neurons per BM	3
g	4.0
ϵ	93.33%
μ_{W}	0.001 μS
weight distr.	$2.0 \cdot \big(\mathrm{beta}(0.5,\ 0.5) - 0.5\big)$
bias distr.	$1.2 \cdot \big(\mathrm{beta}(0.5,\ 0.5) - 0.5\big)$
calibration time	10^5ms
sampling time	10^5ms
training steps	1000

Table 8.7: Simulation parameters used in Chap. 6.3.2. The sampling time is used while calculating the update steps. No Poisson sources are used.

#BMs	2, 3, 4
#neurons per BM	10
g	4.0
ϵ	100%
μ_{W}	0.001 μS
σ_{W}	0.001 μS
weight distr.	$2.0 \cdot \big(\mathrm{beta}(0.5,\ 0.5) - 0.5\big)$
bias distr.	$1.2 \cdot \big(\mathrm{beta}(0.5,\ 0.5) - 0.5\big)$
calibration time	10^5ms
sampling time	10^5ms
training steps	1000

8.3 Simulation Software

All simulations have been done in the programming language *Python 2.7.3* (*Van Rossum and Drake Jr*, 1995). The implementation and numerical computation of neuron models, synapse types and noise sources were done with *NEST 2.4.2* (*Gewaltig and Diesmann*, 2007), which is an abbreviation for *Neural Simulation Tool*. NEST is a simulator for spiking neural networks and provides a multitude of neuron models and synapse types for large scale simulations. It can be accessed in Python via the pyNEST interface.

Instead of using NEST directly, *PyNN 0.8* (*Davison et al.*, 2009), pronounced *pine*, a Python module which implements a simulator-independent language for building and connecting populations of neurons has been utilized.

Furthermore, the Python module *SBS 1.3.2* (*Breitwieser*, 2015), called *Spike-Based Sampling*, allowed effortless realizations of LIF Boltzmann Machines by providing methods for calibrating neurons and translating theoretical Boltzmann parameters to networks of LIF neurons with PyNN and NEST.

Throughout this thesis, SBS and PyNN were mainly supplemented with code written by myself to attain the desired network setups and evaluate simulation runs accordingly. Also, in some cases, SBS code has either been modified or newly implemented as separate functions.

Finally, *SLURM* (*Simple Linux Utility Resource Management*, *Yoo et al.* (2003)) was used to distribute and manage independent jobs on the group internal computer cluster in a straightforward way.

8.4 HICANN Wafer System

The HICANN wafer system is currently under development as part of the *Human Brain Project* (HBP), a so-called FET Flagship (European Commission Future and Emerging Technologies Flagship). The goal of the HICANN wafer system is to provide a general-purpose emulation platform for biologically inspired neural networks. This opens up the possibility to develop and implement novel computational methods beyond the commonly used von Neumann architecture.

A single wafer contains 384 HICANN chips, all made up of two symmetric halves containing both neuron circuits and a large synapse array occupying most of the space on-chip (see Fig. 8.1). In total, a single chip holds 2x256 neuron circuits, so-called *dendritic membranes* (DenMems), grouped into 8 blocks of 64 DenMems that can either be short-circuited or combined to form larger neurons or multi-compartment models. Each circuit implements the *Adaptive Exponential Integrate-and-Fire* neuron model (*Brette and Gerstner*, 2005) with conductance-based synapses. The synapse arrays are made up of 224 rows and 256 columns each and are used to route synaptic input transmitted via 2x128 vertical buses to the neuron circuits. The synaptic weights have either a resolution of 4 bits or, by combining two neighbouring synapse drivers, a maximum resolution of 8 bits. Note that due to the high integration density of analog neurons and synapses on the chip, all components are sped up by a factor of 10^4 compared to biology.

If a neuron spikes, a time-stamped digital spike pulse package with the corresponding 6 bit neuron address gets fed into one of the 64 horizontal buses and can then be routed towards another neuron via the vertical buses. This transport of action potentials with horizontal and vertical buses spanning over the whole wafer is called layer 1 (L1) communication. The L1 architecture cannot be directly connected to an external component, for example a host PC or other wafers. Therefore, each HICANN has a layer 2 (L2) interface with the same bandwidth as a single L1 bus. This L2 interface can be fed with up to eight L1 buses and gets connected to the host PC via a hierarchical packet-based network (*Millner*, 2012). This obviously limits the available bandwidth for external spike inputs. The high acceleration factor further reduces the amount of external input that can be provided, since noise frequencies are coupled to the intrinsic time scale of the on-chip neurons. Every HICANN is further equipped with its own noise generators, so-called *linear-feedback shift registers*, but due to the limited space available on chips, the number of shift registers per HICANN was restricted to eight only.

Note that the bandwidth limitation ultimately results from hard physical constraints, because the information flux into every physical device is strictly limited by its surrounding surface area and the transmission speed of the incoming signal. Therefore, it is an inherent property of such systems and does not only apply to the HICANN chip.

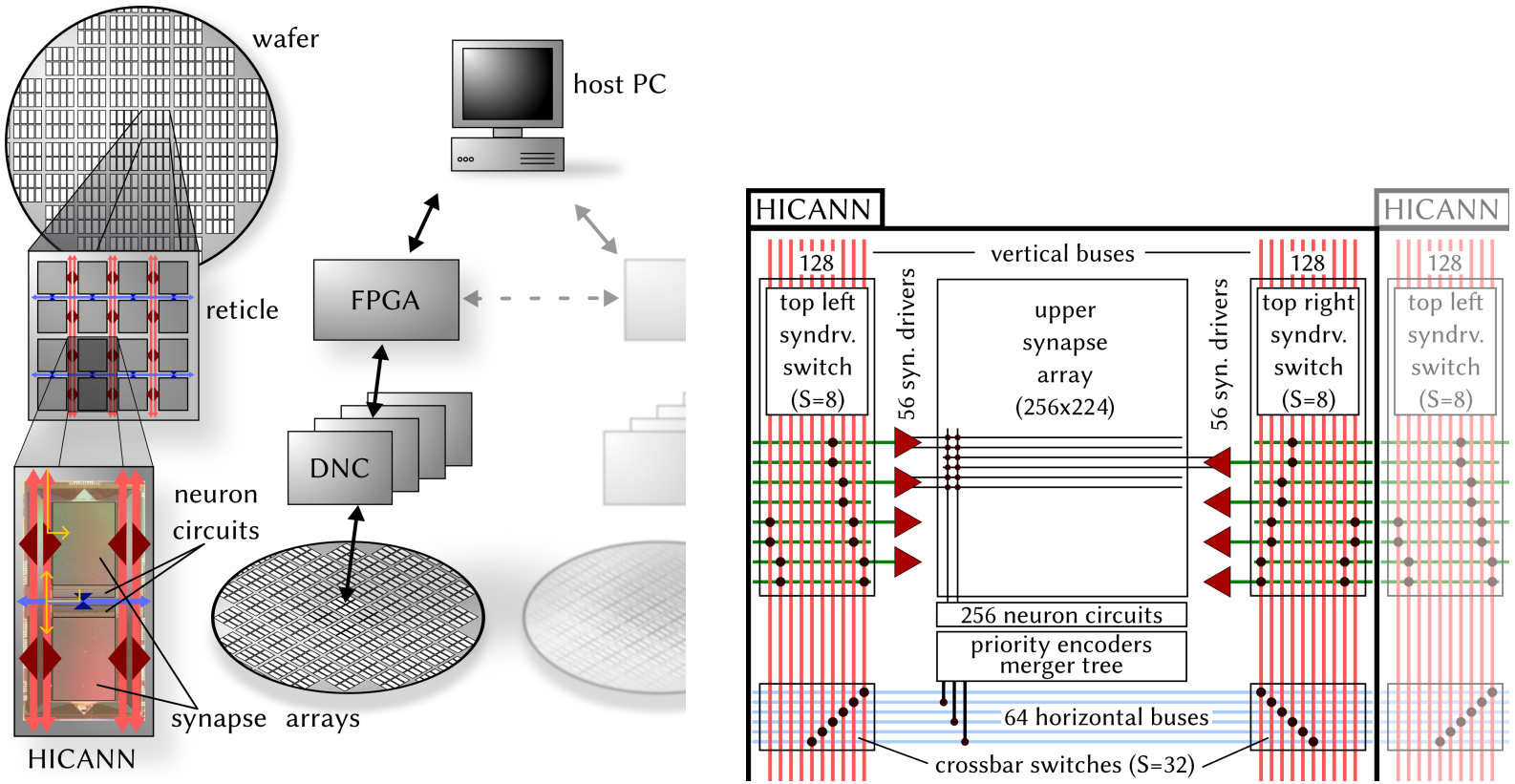

Figure 8.1: Schematics of the HICANN wafer system. **(left)** A single wafer consists of 48 reticles with 8 HICANN chips each. The size of the reticles is limited by the maximum surface that can be illuminated at once during the manufacturing process. Every HICANN consists of two symmetric halves holding synapse arrays and neuron circuits. On-chip as well as wafer-wide communication between neurons is implemented via horizontal (blue) and vertical (red) buses stretching all over the wafer. The yellow arrows show exemplary routes of spikes. Off-wafer communication uses an additional interface that connects the wafer via a hierarchical packet-based network, four Digital Network Chips (DNC) and one Field-Programmable Gate Array (FPGA), to a host PC. **(right)** Schematic of the upper half of a single HICANN chip. If a neuron spikes, a 6 bit digital signal holding the neuron's address is injected into the horizontal bus. Routing paths can be statically set with crossbar and synapse driver switches. In the synapse array, ingoing digital signals are translated into a postsynaptic conductance and transmitted to the neuron circuit if the 6 bit address of the ingoing signal matches the one previously set in the synapse. Images taken from *Petrovici* (2016).

8.5 Illustration of the Beta Distribution

The beta distribution is defined on the bounded interval $x \in [0, 1]$ as

$$\text{beta}(\alpha,\ \beta) = \frac{1}{B(\alpha,\ \beta)} x^{\alpha-1}(1-x)^{\beta-1} \tag{8.1}$$

with the normalization constant $B(\alpha,\ \beta) = \int_0^1 x^{\alpha-1}(1-x)^{\beta-1}\ \mathrm{d}x$. The parameters α and β determine the shape of the distribution. Note that having a bounded set of values for x is very desirable if we want to draw random synaptic weights, as for instance a Gaussian distribution would allow problematically huge weights with a low but non-zero probability. The shapes of the beta distributions used throughout this thesis are shown in Fig. 8.2.

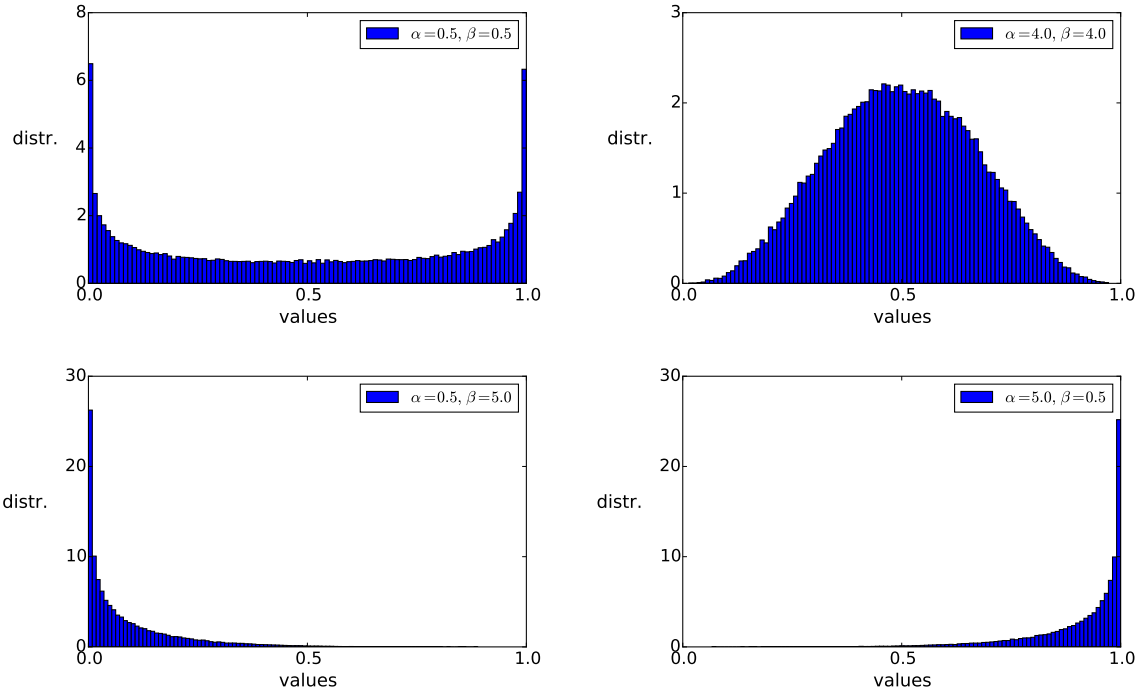

Figure 8.2: Demonstration of the typical beta distributions used throughout this thesis. The shape of the distribution can be adjusted by changing α and β. **(top)** If α and β are equal, the distribution is symmetric. The mass of the probability distribution can be shifted to the boundaries for parameters < 1 (left) and to the center for parameters > 1 (right). Note that for $\alpha = \beta = 1$, we obtain a uniform distribution on the interval $[0, 1]$. **(bottom)** If α and β have different values, the distribution becomes skewed.

8.6 Illustration of the Interconnection Weight Distribution

The distribution introduced in Chap. 5 that generates the noise interconnections between network BMs is demonstrated in Fig. 8.3 for several parameter sets.

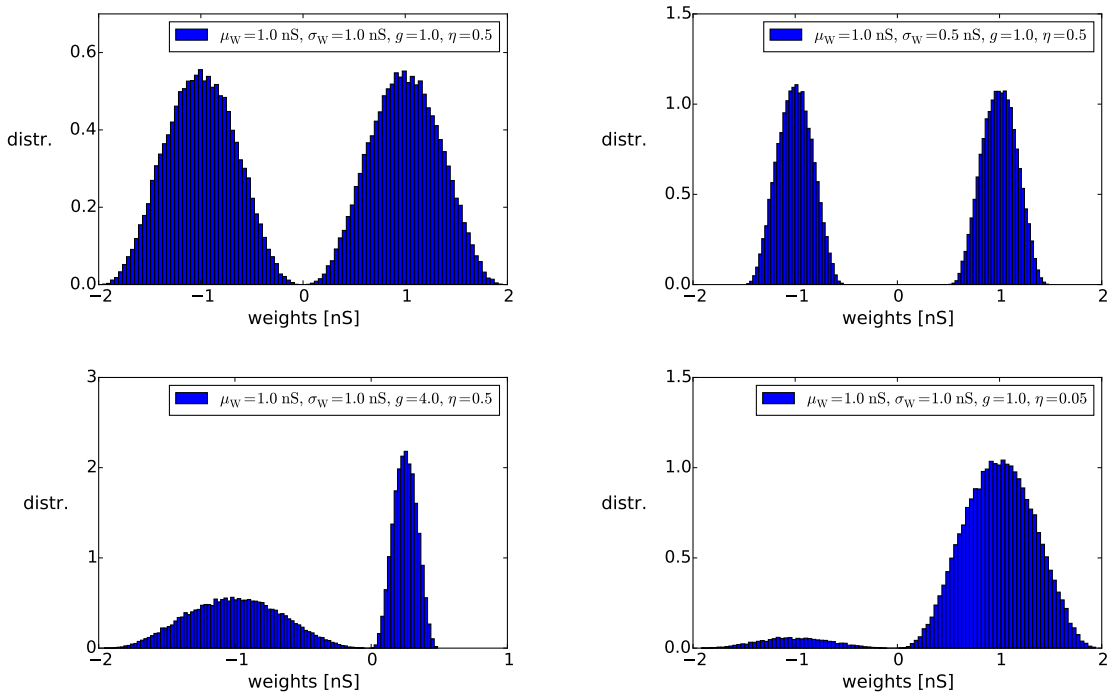

Figure 8.3: Some examples of how the interconnection weight distribution can be adjusted by changing the mean-to-width ratio, g and η. **(top left)** Standard weight distribution used in this thesis which yields symmetric interconnections. **(top right)** By increasing the mean-to-width ratio, the distribution gets centered narrower around $\pm\mu_W$. **(bottom left)** Adjusting g leads to a rescaling of the relative strength of inhibitory and excitatory weights. The distribution has been chosen such that changing g leaves the mean-to-width ratio constant. **(bottom right)** η can be used to set the relative abundance of excitatory and inhibitory synaptic connections in the network.

8.7 Proof for the Free Membrane Potential Autocorrelation Function

We can prove Eq. 3.6 by using the Wiener-Khintchine theorem (W), see Eq. 3.3, and the well-known convolution theorem (C), which states that convolutions in real space are equal to point-wise multiplications in Fourier space and vice versa, i.e.,

$$\mathcal{F}(a * b) = \mathcal{F}(a) \cdot \mathcal{F}(b)\,, \tag{8.2a}$$
$$\mathcal{F}(a \cdot b) = \mathcal{F}(a) * \mathcal{F}(b)\,, \tag{8.2b}$$

where Fourier transforms are denoted by $\mathcal{F}$ and convolutions by $*$. a and b are, in this case, arbitrary functions. In the following, the autocorrelation function of a function f is denoted by ρ_f. If the neuron is fed with a spike train η, the trace of the free membrane potential can be obtained by convolving η with the PSP shape κ, i.e., $u = \eta * \kappa$. With this, the autocorrelation function can be calculated as follows:

$$\rho_u(\Delta) \overset{\text{W}}{=} \mathcal{F}^{-1}\big(\mathcal{F}(u)\mathcal{F}^*(u)\big)(\Delta) \tag{8.3a}$$
$$\overset{\text{C}}{=} \mathcal{F}^{-1}\big(\mathcal{F}(\eta)\mathcal{F}(\kappa)\mathcal{F}^*(\eta)\mathcal{F}^*(\kappa)\big)(\Delta) \tag{8.3b}$$
$$\overset{\text{C}}{=} \big[\mathcal{F}^{-1}\big(\mathcal{F}(\eta)\mathcal{F}^*(\eta)\big) * \mathcal{F}^{-1}\big(\mathcal{F}(\kappa)\mathcal{F}^*(\kappa)\big)\big](\Delta) \tag{8.3c}$$
$$\overset{\text{W}}{=} [\rho_\eta * \rho_\kappa](\Delta)\,. \tag{8.3d}$$

In case of a COBA LIF neuron which is in the high-conductance state, we obtain Eq. 3.6 because the PSP shape κ can be approximated by an exponential decay with time constant τ_{syn}.

8.8 Additional Results

8.8.1 Sampling Quality Dependence on the Calibration Time

For the following simulation results, the weights and biases of the target distributions were drawn from beta distributions:

$$W \propto 1.2 \cdot \big(\text{beta}(0.5,\ 0.5) - 0.5\big)\,, \tag{8.4a}$$

$$b \propto 1.2 \cdot \big(\text{beta}(0.5,\ 0.5) - 0.5\big)\,. \tag{8.4b}$$

In total, the generated noise was used to sample from 24 different Boltzmann distributions with 3 neurons. To take the changing quality of the generated noise into account, the experiment was repeated 10 times. Thus, in total, noise from BMs was generated 10 times and each of these were used to sample from 24 Boltzmann distributions. The results are shown in Fig. 8.4.

8.8.2 k-th Neighbour Interval Distribution for Very Large Networks

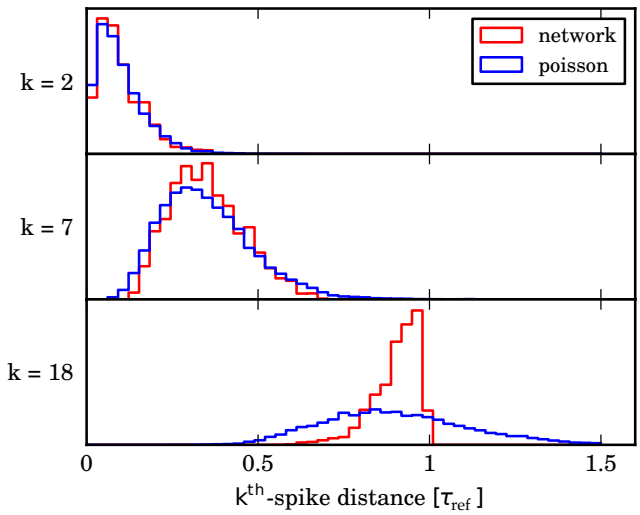

Figure 8.5: k-th neighbour interval distribution for a noise-generating network consisting of 20 Boltzmann machines. Even though for small k there are only minor differences, as soon as the distribution is shifted towards τ_{ref}, the interval distribution becomes distorted again. The mean frequency of the neurons generating the noise is $\bar{\nu} = 0.9997\ \tau_{\text{ref}}^{-1}$.

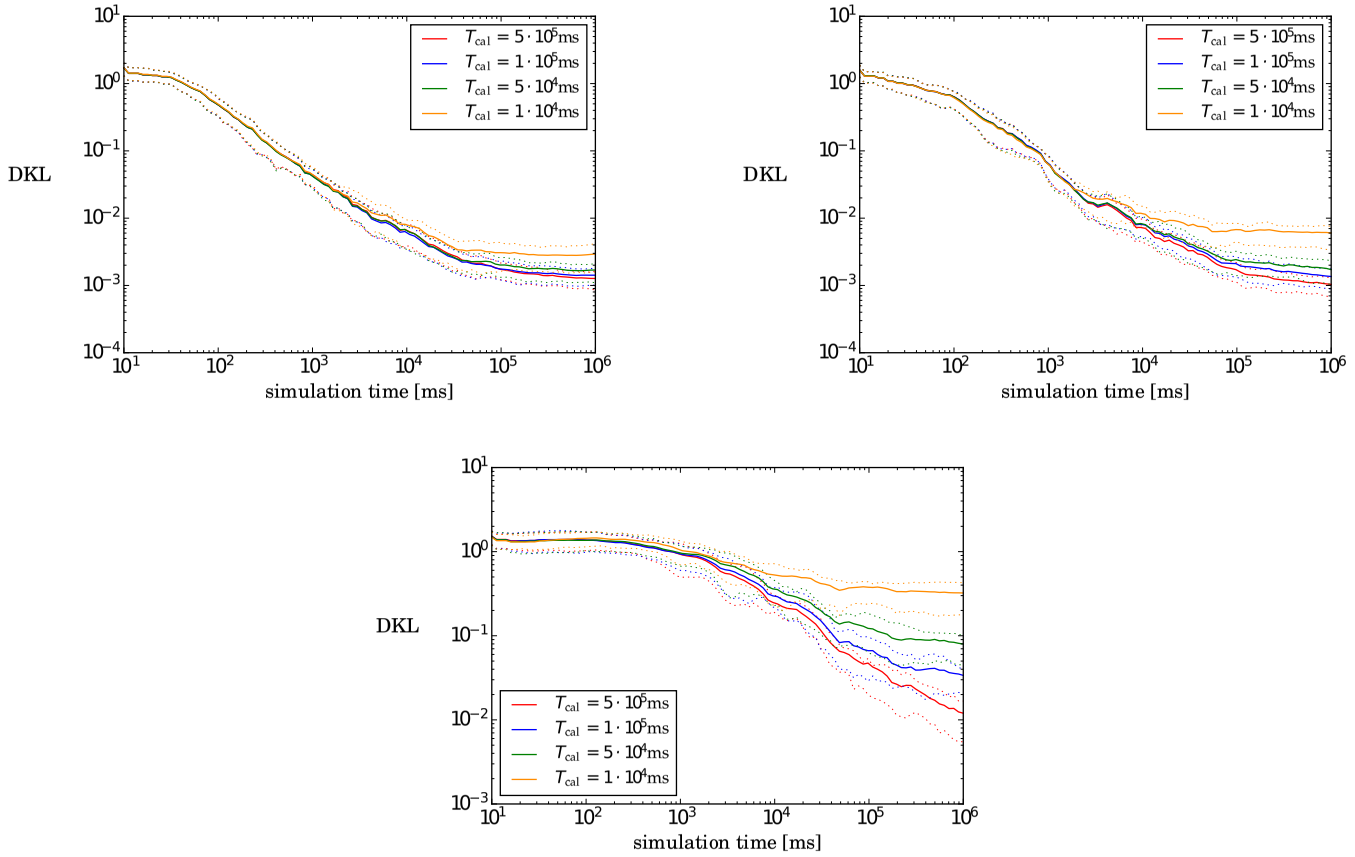

Figure 8.4: Kullback-Leibler divergence for sampling with noise coming from Boltzmann machines. The noise-generating neurons have an activity of (top left) $\bar{\nu} = 0.14\ \tau_{\text{ref}}^{-1}$, (top right) $\bar{\nu} = 0.89\ \tau_{\text{ref}}^{-1}$ and (bottom) $\bar{\nu} = 0.9997\ \tau_{\text{ref}}^{-1}$. The thick lines represent the mean and the dotted lines the 15th and 85th percentile of all measurements. First, for low activities, the calibration time has no significant effect on the sampling quality, as can be seen in the top left figure. However, if the noise is generated by neurons with high activity, i.e., neurons which spike close to their maximum frequency τ_{ref}^{-1}, calibrating longer improves the sampling results tremendously. Also note the plateau in the bottom plot, demonstrating that sampling with very strong autocorrelations in the noise indeed slows down the effective sampling speed as uncorrelated random kicks only appear on large time scales $\gg \tau_{\text{syn}}$.

8.8.3 Sampling Quality with Bursting Noise Neurons

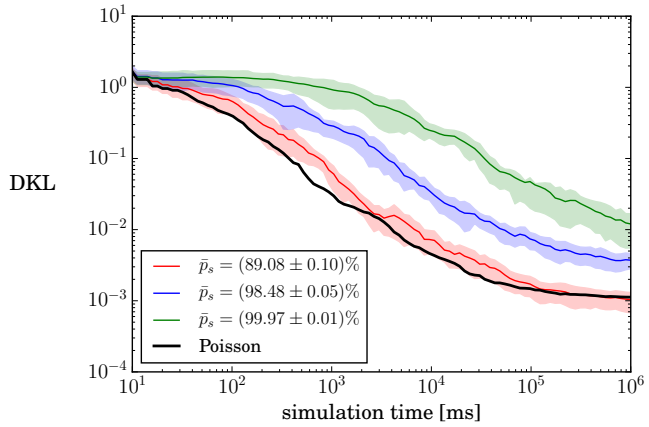

Figure 8.6: LIF sampling quality for a 3-neuron BM fed with noise coming from very active neurons. As can be seen, even for noise neurons with mean rates up to around $\bar{\nu} = 0.9\ \tau_{\text{ref}}^{-1}$, D_{KL} values similar to those obtained from ideal LIF sampling with Poisson sources are reached. Hence, LIF sampling works over a large range of biases in the noise BMs.

8.8.4 Autocorrelation of Noise from Randomized Boltzmann Machines

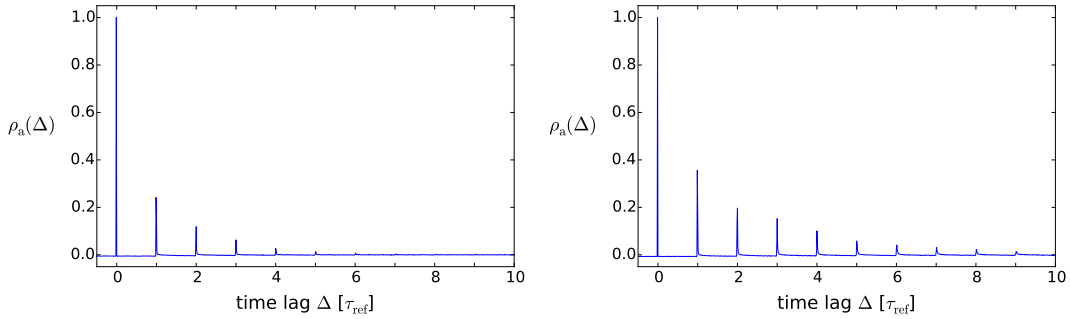

Figure 8.7: Autocorrelation function of spike trains generated by neurons with different weight scale factor W_0^{noise}. (left) $W_0^{\text{noise}} = 0.6$ and (right) $W_0^{\text{noise}} = 2.4$. Both cases are rather similar. Therefore, the influence of the weight scale factor on the autocorrelation of the generated noise spike trains is only marginal. Further, note that the strength of the autocorrelations is very similar to those obtained for neurons with intermediate activity, e.g. $\bar{\nu} = 0.53\ \tau_{\text{ref}}^{-1}$, meaning that we are not in the limit of extreme bursting.

8.8.5 Comparison of the Two Calibration Schemes

To compare the iterative and non-iterative calibration scheme presented in Chap. 5, a fully connected network of 10 BMs which consist of 3 neurons each was used. The minimum step width of the interconnections was set to $\frac{\omega_\mathrm{p}}{10}$. The mean-to-width ratio for the noise weight distribution and the noise weight strength ratio g were both set to 1. Furthermore, the weights and biases of the target Boltzmann distributions were drawn from the beta distributions given in Eq. 5.1a and 5.1b with $W_0 = 1.2$.

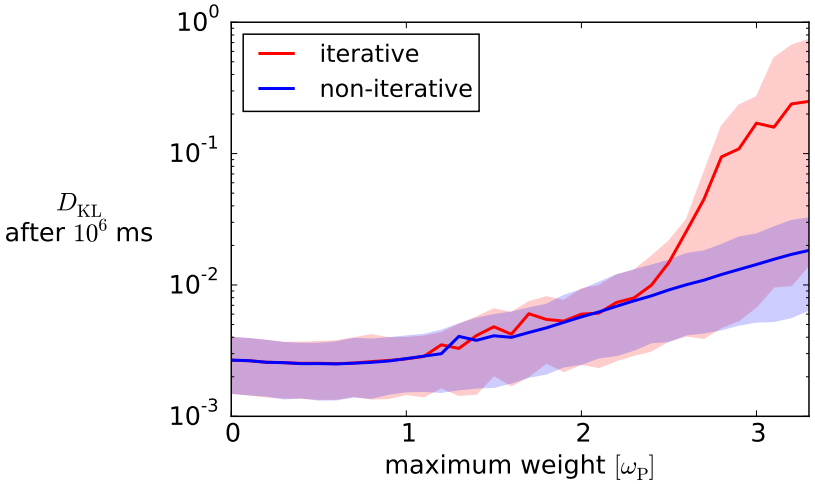

Figure 8.8: Comparison of the iterative (red) and non-iterative or greedy (blue) scheme for different interconnection weights given as multiples of the Poisson weights ω_p. The drawn lines represent the average D_KL over five network realizations with different random seeds. The shaded area gives the 15th and 85th percentile over the D_KL values of all BMs. For the greedy calibration scheme, the final D_KL values smoothly increase with growing intrinsic connections. However, for the iterative scheme, a sudden rise to very bad D_KL values occurs between $2\omega_\mathrm{p}$ and $3\omega_\mathrm{p}$. Before this sudden rise in D_KL, both schemes perform equally well.

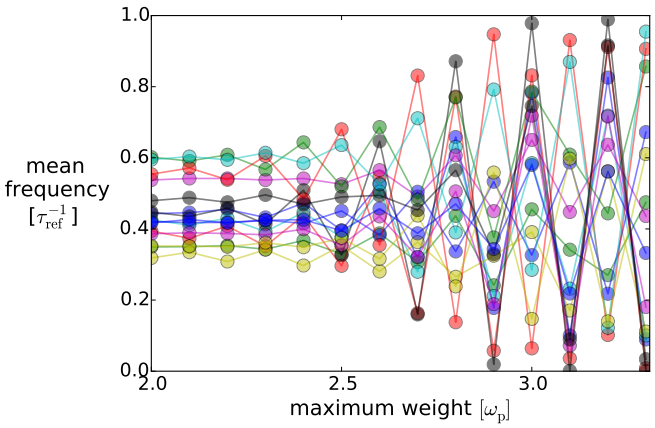

Figure 8.9: Mean frequency of 15 network neurons from a single simulation run using the iterative calibration scheme. For large weights, subsets of neurons start to oscillate between different firing states, e.g. bursting or remaining completely inactive. The oscillating behavior is a direct consequence from taking the spike trains of the previous iteration step for the calibration of the current iteration step.

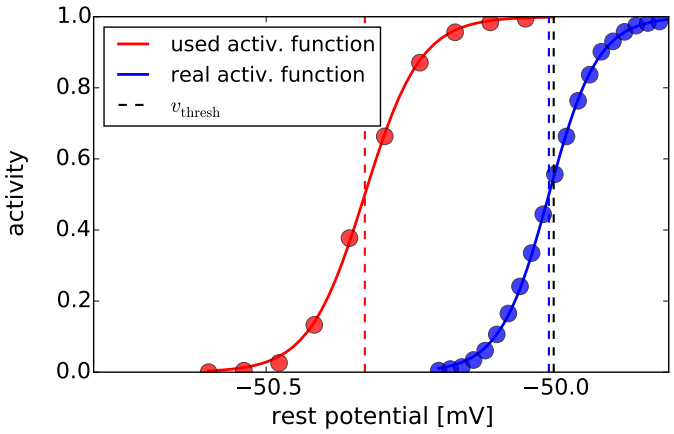

Figure 8.10: Calibration function of a single neuron after all iteration steps. The red line is the activation function obtained from the spike trains of the previous iteration step. The blue curve shows the actually observed activation function while running the network. The dashed lines mark the rest potential at which the probability to be refractory is 0.5. The used calibration function is much farther to the left than the actually observed activity, leading to a very low activity of the neuron even for high biases due to the wrong translation of the theoretical parameters.

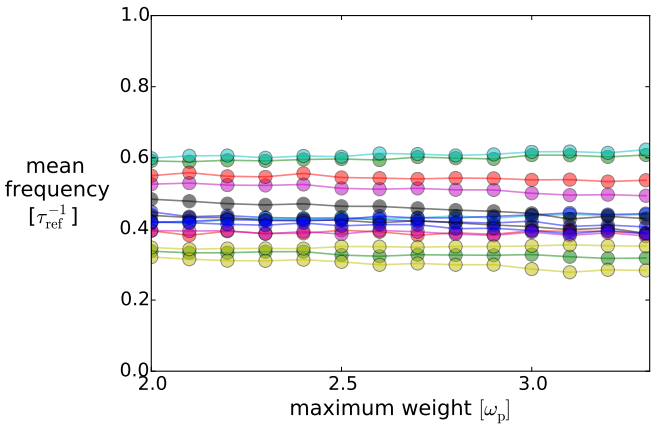

Figure 8.11: Mean frequency of 15 network neurons from a single simulation run using the greedy calibration scheme. Throughout different interconnection weights the mean frequency of each neuron stays approximately the same and the network remains stable.

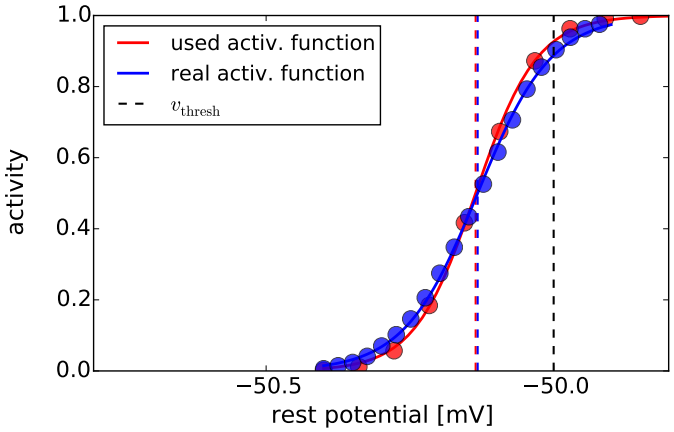

Figure 8.12: Equivalent setup as in Fig. 8.10, but the calibration was done with the greedy, non-iterative scheme. Both the approximation and the activity observed while sampling with network connections are quite similar, demonstrating that the non-iterative scheme should be used instead of the iterative one.

8.8.6 Mean-to-Width Ratio of the Interconnections

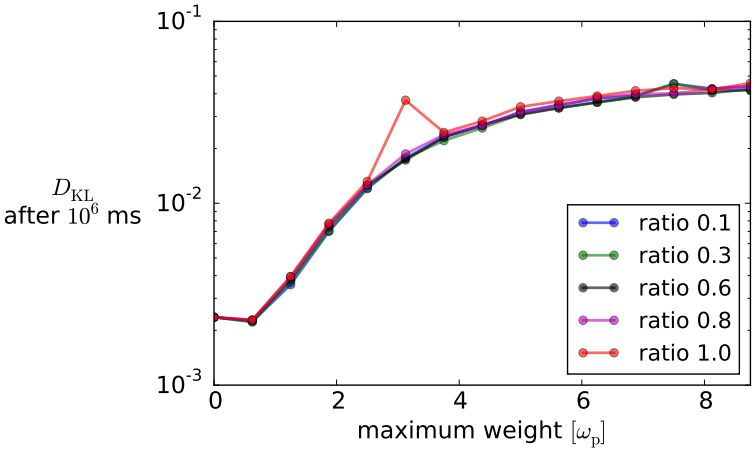

Figure 8.13: Mean D_{KL} of all BMs in a network after sampling for 10^6ms. The network consists of 100 3-neuron BMs with a connectivity $\epsilon = 0.1$ and $g = 1.0$, $\eta = 0.5$. Changing the mean-to-width ratio $\frac{\sigma_{\mathrm{W}}}{\mu_{\mathrm{W}}}$ of the distribution from which the interconnection weights are drawn has no significant effect on the LIF sampling quality. Error bars are rather small and have been excluded to provide readability in the plot. The outlier results from bad calibrations.

8.8.7 Shorter Sampling Time for Training Updates

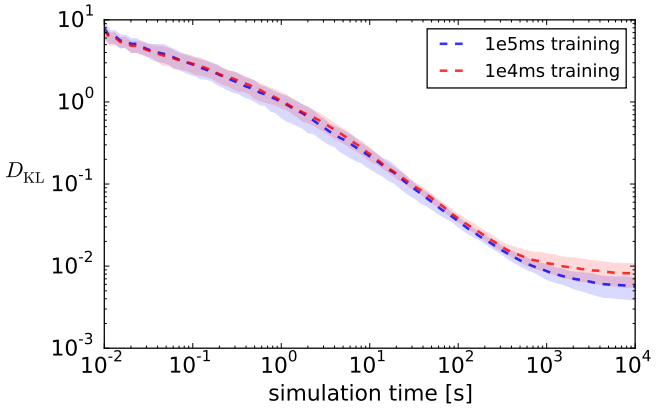

Figure 8.14: Same setup as in Fig. 6.5, but the probabilities used to calculate the weight and bias updates every step were either obtained after sampling for 10^5ms and 1200 training steps (blue) and 10^4ms and 2000 training steps (red). Both setups lead to similar results and the small differences occurring can be reduced by further training the 10^4ms case.

8.8.8 DKL after Training for Different Weight Ratios

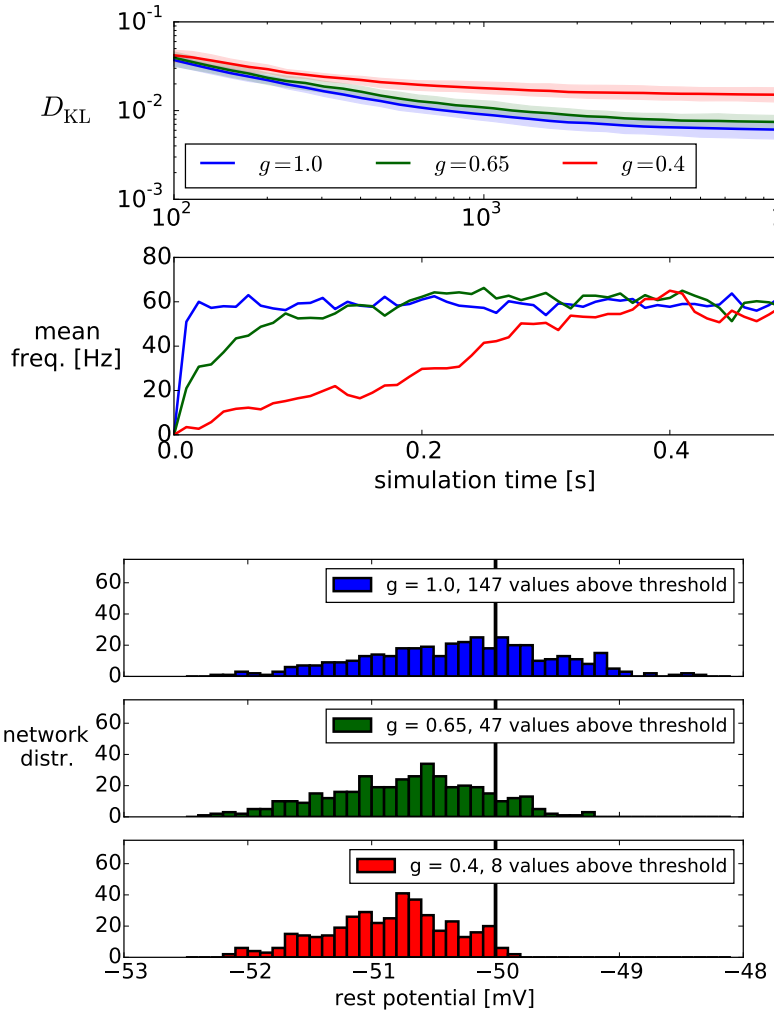

Figure 8.15: Same setup as in Fig. 6.5, but with updates obtained after sampling for 10^4ms and 2000 training steps. The sea of BMs was realized for three different values of g. **(top)** Even for excitation dominated interconnections, i.e., $g = 0.65$ or $g = 0.4$, training leads to very good sampling results. Note that $g = 0.4$ was only trained for 1200 steps, resulting in a slightly worse D_{KL} as in the other cases. Additionally, for smaller values of g the network needs longer during start-up to reach the desired target mean network activity. **(bottom)** Distribution over rest potentials set in the network after translating the theoretical biases. The black vertical line marks the threshold potential. Note that even a very small number of neurons (e.g. 8 out of 400) with strong excitatory noise connections is able to stably start the network.

8.8.9 Sea of Boltzmann Machines as a Large Boltzmann Machine

Again, the same setup as in Fig. 6.5 was used, but with a symmetric interconnection weight matrix. To keep the initially symmetric weights even after cutting off a connection W_{ij}, the transposed entry W_{ji} has to be set to 0 as well. While cutting off weights in this way, one has to be careful not to cut off too many inputs of certain neurons due to the need of symmetrizing the weight matrix after each step.

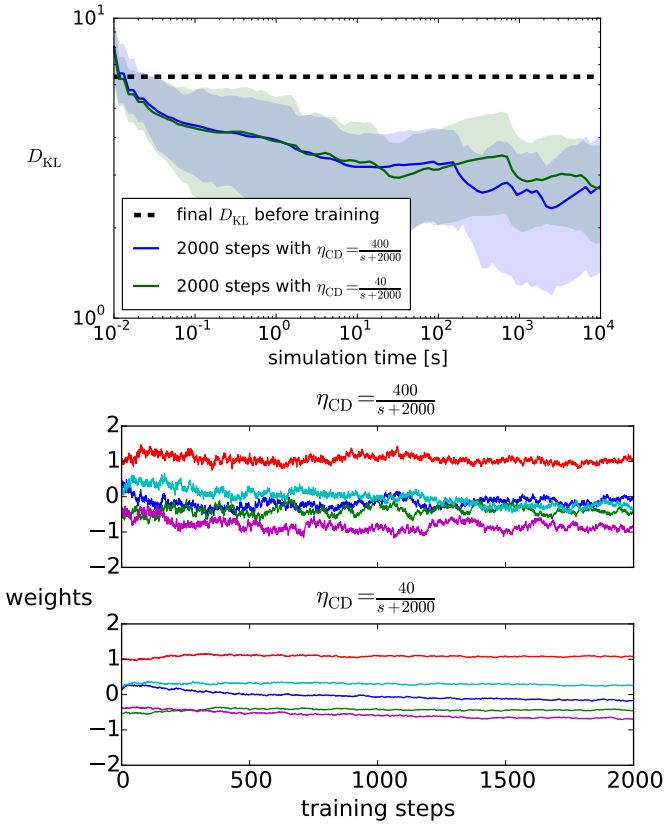

Figure 8.16: Same setup as in Fig. 6.5, but with a symmetric interconnection weight matrix. Hence, the whole sea of BMs is a BM itself. **(top)** Even after training for 2000 update steps as before, the sampling quality does not improve considerably and remains in a very bad regime of D_{KL} values. **(bottom)** The reason for the bad sampling behavior originates from the many feedback connections enforced by symmetrizing the weight matrix. Consequently, every neuron receives only noise from neurons it provides noise to, introducing strong correlations into the network. These cannot be trained away with CD and lead to the flat weight evolution observed here even for smaller learning rates.

Bibliography

Ackley, D. H., G. E. Hinton, and T. J. Sejnowski, A learning algorithm for Boltzmann machines, *Cognitive science*, *9*(1), 147–169, 1985.

Arieli, A., A. Sterkin, A. Grinvald, and A. Aertsen, Dynamics of ongoing activity: explanation of the large variability in evoked cortical responses, *Science*, *273*(5283), 1868, 1996.

Azevedo, F. A., L. R. Carvalho, L. T. Grinberg, J. M. Farfel, R. E. Ferretti, R. E. Leite, R. Lent, S. Herculano-Houzel, et al., Equal numbers of neuronal and nonneuronal cells make the human brain an isometrically scaled-up primate brain, *Journal of Comparative Neurology*, *513*(5), 532–541, 2009.

Azouz, R., and C. M. Gray, Cellular mechanisms contributing to response variability of cortical neurons in vivo, *The Journal of neuroscience*, *19*(6), 2209–2223, 1999.

Breitwieser, O., Investigation of a Cortical Attractor-Memory Network, Bachelor thesis, Heidelberg University, 2011.

Breitwieser, O., Towards a Neuromorphic Implementation of Spike-Based Expectation Maximization, Master's thesis, Heidelberg University, 2015.

Brette, R., and W. Gerstner, Adaptive Exponential Integrate-and-Fire Model as an Effective Description of Neuronal Activity, *Journal of neurophysiology*, *94*(5), 3637–3642, 2005.

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 Comput Biol*, *7*(11), e1002,211, 2011.

Bytschok, I., From Shared Input to correlated Neuron Dynamics: Development of a Predictive Framework, Diploma thesis, Heidelberg University, 2011.

Cáceres, M. O., Harmonic potential driven by long-range correlated noise, *Physical Review E*, *60*(5), 5208, 1999.

Cooley, J. W., and J. W. Tukey, An algorithm for the machine calculation of complex Fourier series, *Mathematics of computation*, *19*(90), 297–301, 1965.

Davison, A., D. Brüderle, J. Kremkow, E. Muller, D. Pecevski, L. Perrinet, and P. Yger, PyNN: a common interface for neuronal network simulators, 2009.

Dayan, P., and L. F. Abbott, *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems*, vol. 10, Cambridge, MA: MIT Press, 2001.

De Carlos, J. A., and J. Borrell, A historical reflection of the contributions of Cajal and Golgi to the foundations of neuroscience, *Brain research reviews*, *55*(1), 8–16, 2007.

Dieter, K., and D. Tadin, Understanding attentional modulation of binocular rivalry: A framework based on biased competition, *Frontiers in Human Neuroscience*, *5*, 155, doi:10.3389/fnhum.2011.00155, 2011.

Duane, S., A. D. Kennedy, B. J. Pendleton, and D. Roweth, Hybrid Monte Carlo, *Physics letters B*, *195*(2), 216–222, 1987.

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.

Fourcaud, N., and N. Brunel, Dynamics of the firing probability of noisy integrate-and-fire neurons, *Neural computation*, *14*(9), 2057–2110, 2002.

Fuhrmann, G., I. Segev, H. Markram, and M. Tsodyks, Coding of temporal information by activity-dependent synapses, *Journal of neurophysiology*, *87*(1), 140–148, 2002.

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. M. Kistler, *Spiking neuron models: Single neurons, populations, plasticity*, Cambridge university press, 2002.

Gewaltig, M.-O., and M. Diesmann, NEST (NEural Simulation Tool), *Scholarpedia*, *2*(4), 1430, 2007.

Guillery, R., Observations of synaptic structures: origins of the neuron doctrine and its current status, *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, *360*(1458), 1281–1307, 2005.

Hanggi, P., and P. Jung, Colored noise in dynamical systems, *Advances in chemical physics*, *89*, 239–326, 1995.

Hastings, W. K., Monte Carlo sampling methods using Markov chains and their applications, *Biometrika*, *57*(1), 97–109, 1970.

Henry, G., P. Bishop, R. Tupper, and B. Dreher, Orientation specificity and response variability of cells in the striate cortex, *Vision research*, *13*(9), 1771–1779, 1973.

Hinton, G., A practical guide to training restricted boltzmann machines, 2010.

Hinton, G. E., Training products of experts by minimizing contrastive divergence, *Neural computation*, *14*(8), 1771–1800, 2002.

Hodgkin, A. L., and A. F. Huxley, Action potentials recorded from inside a nerve fibre, *Nature*, *144*(3651), 710–711, 1939.

Hodgkin, A. L., and A. F. Huxley, A quantitative description of membrane current and its application to conduction and excitation in nerve, *The Journal of physiology*, *117*(4), 500, 1952.

Holt, G. R., W. R. Softky, C. Koch, and R. J. Douglas, Comparison of discharge variability in vitro and in vivo in cat visual cortex neurons, *Journal of Neurophysiology*, *75*(5), 1806–1814, 1996.

Hoyer, P. O., and A. Hyvarinen, Interpreting neural response variability as Monte Carlo sampling of the posterior, *Advances in neural information processing systems*, pp. 293–300, 2003.

Jordan, J., Deterministic recurrent networks as a source of uncorrelated noise for functional neural systems, Master's thesis, University of Munich, 2013.

Jordan, J., T. Tetzlaff, M. Petrovici, O. Breitwieser, I. Bytschok, J. Bill, J. Schemmel, K. Meier, and M. Diesmann, Deterministic neural networks as sources of uncorrelated noise for probabilistic computations, p. Suppl 1: P62, 2015.

Khintchine, A., Korrelationstheorie der stationären stochastischen Prozesse, *Mathematische Annalen*, *109*(1), 604–615, 1934.

Korcsák-Gorzó, A., Firing States of Recurrent Leaky Integrate-and-Fire Networks, Bachelor thesis, Heidelberg University, 2015.

Körding, K. P., and D. M. 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.

Kullback, S., and R. A. Leibler, On information and sufficiency, *The annals of mathematical statistics*, *22*(1), 79–86, 1951.

Kumar, A., S. Schrader, A. Aertsen, and S. Rotter, The high-conductance state of cortical networks, *Neural computation*, *20*(1), 1–43, 2008.

Kungl, A. F., Sampling with leaky integrate-and-fire neurons on the HICANNv4 neuromorphic chip, Master's thesis, Heidelberg University, 2016.

Lapique, L., Recherches quantitatives sur l'excitation electrique des nerfs traitee comme une polarization, *J Physiol Pathol Gen*, *9*, 620–635, 1907.

LeCun, Y., C. Cortes, and C. J. Burges, The MNIST database of handwritten digits, 1998.

Lee, D., N. L. Port, W. Kruse, and A. P. Georgopoulos, Variability and correlated noise in the discharge of neurons in motor and parietal areas of the primate cortex, *The Journal of neuroscience*, *18*(3), 1161–1170, 1998.

Leng, L., Deep Learning Architectures for Neuromorphic Hardware, Master's thesis, Heidelberg University, 2014.

Leng, L., M. A. Petrovici, R. Martel, I. Bytschok, O. Breitwieser, J. Bill, J. Schemmel, and K. Meier, Spiking neural networks as superior generative and discriminative models, 2016.

Maass, W., and H. Markram, Synapses as dynamic memory buffers, *Neural Networks*, *15*(2), 155–161, 2002.

Maaten, L. v. d., and G. Hinton, Visualizing data using t-SNE, *Journal of Machine Learning Research*, *9*(Nov), 2579–2605, 2008.

Mainen, Z. F., and T. J. Sejnowski, Reliability of spike timing in neocortical neurons, *Science*, *268*(5216), 1503, 1995.

Martel, R., Generative Properties of LIF-based Boltzmann Machines, Master's thesis, Heidelberg University, hD-KIP 15-86, 2015.

Metropolis, N., A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, Equation of state calculations by fast computing machines, *The journal of chemical physics*, *21*(6), 1087–1092, 1953.

Millner, S., Development of a Multi-Compartment Neuron Model Emulation, Dissertation, Heidelberg University, 2012.

Paradiso, M., A theory for the use of visual orientation information which exploits the columnar structure of striate cortex, *Biological cybernetics*, *58*(1), 35–49, 1988.

Perez-Costas, E., M. Melendez-Ferro, and R. Roberts, Pyramidal neuron in the prefrontal cortex of the rat stained with the Golgi-Cox method, *Neuropsychopharmacology*, *32*(10), `http://www.nature.com/npp/journal/v32/n10/covers/index.html`, 2007.

Petrovici, M. A., *Form Versus Function: Theory and Models for Neuronal Substrates*, Springer, 2016.

Petrovici, M. A., J. Bill, I. Bytschok, J. Schemmel, and K. Meier, Stochastic inference with deterministic spiking neurons, *arXiv*, 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, p. 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, in *Advances in Neural Information Processing Systems 28*, 2015b.

Petrovici, M. A., et al., Characterization and compensation of network-level anomalies in mixed-signal neuromorphic modeling platforms, *PLOS ONE*, doi:dx.doi.org/10.1371/journal.pone.0108590, 2014.

Pfeil, T., et al., Six Networks on a Universal Neuromorphic Computing Substrate, *Frontiers in Neuroscience*, *7*, 11, 2013.

Rolls, E., and G. Deco, The Noisy Brain: Stochastic Dynamics as a Principle of Brain Function, 2010.

Roumani, D., and K. Moutoussis, Binocular rivalry alternations and their relation to visual adaptation, *Frontiers in human neuroscience*, *6*, 2012.

Salakhutdinov, R., and G. E. Hinton, Deep Boltzmann Machines., in *AISTATS*, vol. 1, p. 3, 2009.

Salakhutdinov, R., A. Mnih, and G. Hinton, Restricted boltzmann machines for collaborative filtering, in *Proceedings of the 24th international conference on Machine learning*, pp. 791–798, ACM, 2007.

Schemmel, J., A. Grübl, K. Meier, and E. Mueller, Implementing synaptic plasticity in a VLSI spiking neural network model, in *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, pp. 1–6, IEEE, 2006.

Schemmel, J., J. Fieres, and K. Meier, Wafer-scale integration of analog neural networks, in *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, pp. 431–438, IEEE, 2008.

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 2010 IEEE International Symposium on Circuits and Systems*, pp. 1947–1950, IEEE, 2010.

Schiller, P. H., B. L. Finlay, and S. F. Volman, Short-term response variability of monkey striate neurons, *Brain research*, *105*(2), 347–349, 1976.

Schwarz, U., Stochastic Dynamics, lecture notes, Heidelberg University, `https://www.thphys.uni-heidelberg.de/~biophys/PDF/Skripte/StochasticDynamics.pdf`, 2012.

Shadlen, M. N., and W. T. Newsome, Noise, neural codes and cortical organization, *Current opinion in neurobiology*, *4*(4), 569–579, 1994.

Shannon, C. E., A mathematical theory of communication., *The Bell System Technical Journal*, *27*, 1948.

Snowden, R. J., S. Treue, and R. A. Andersen, The response of neurons in areas V1 and MT of the alert rhesus monkey to moving random dot patterns, *Experimental Brain Research*, *88*(2), 389–400, 1992.

Stöckel, D., Boltzmann Sampling with Neuromorphic Hardware, Bachelor thesis, Heidelberg University, 2015.

Tsodyks, M. V., and H. Markram, The neural code between neocortical pyramidal neurons depends on neurotransmitter release probability, *Proceedings of the National Academy of Sciences*, *94*(2), 719–723, 1997.

Uhlenbeck, G. E., and L. S. Ornstein, On the theory of the Brownian motion, *Physical review*, *36*(5), 823, 1930.

Van Rossum, G., and F. L. Drake Jr, *Python tutorial*, Centrum voor Wiskunde en Informatica Amsterdam, The Netherlands, 1995.

Vogels, R., Population coding of stimulus orientation by striate cortical cells, *Biological cybernetics*, *64*(1), 25–31, 1990.

Vogels, R., W. Spileers, and G. A. Orban, The response variability of striate cortical neurons in the behaving monkey, *Experimental brain research*, *77*(2), 432–436, 1989.

Waldeyer, W., Ueber einige neuere Forschungen im Gebiete der Anatomie des Centralnervensystems, *DMW-Deutsche Medizinische Wochenschrift*, *17*(44), 1213–1218, 1891.

Wiener, N., Generalized harmonic analysis, *Acta mathematica*, *55*(1), 117–258, 1930.

Wikimedia Commons, A Necker cube, `https://commons.wikimedia.org/wiki/File:Necker_cube.svg`, uploaded by Fibonacci, 2007.

Wikimedia Commons, "Kaninchen und Ente" ("Rabbit and Duck"), the earliest known version of the duck–rabbit illusion, from the 23 October 1892 issue of Fliegende Blätter, https://commons.wikimedia.org/wiki/File:Kaninchen_und_Ente.png, uploaded by Gemena, 2012.

Wikimedia Commons, Diagram showing some of the main areas of the brain, https://commons.wikimedia.org/wiki/File:Diagram_showing_some_of_the_main_areas_of_the_brain_CRUK_188.svg, created by Cancer Research UK, 2014.

Wikimedia Commons, Synapse schematic (unlabeled), https://commons.wikimedia.org/wiki/File:SynapseSchematic_lines.svg, created by Thomas Splettstoesser, 2015.

Yoo, A. B., M. A. Jette, and M. Grondona, Slurm: Simple linux utility for resource management, in *Workshop on Job Scheduling Strategies for Parallel Processing*, pp. 44–60, Springer, 2003.

Acknowledgments

First of all, I'd like to thank Micha, Wei, Oli, Agnes and Anna for proof-reading parts of my thesis.

Also many thanks to Ilja for all those helpful discussions, inspiring ideas, for your guidance in general and for correcting the entire thesis! You were right that writing a thesis takes a hell of a lot of time, thanks for that advice. And you will certainly enjoy Dark Souls III, be excited!

A general shout-out is dedicated to Akos and Andi for 'making the office great (again)'. I really enjoyed working and discussing with you guys, I hope we can continue this in the years to come.

Furthermore, my sincerest gratitude to Mihai for giving me the opportunity to work on such an epic topic. For all the discussions. For all the time you invested in us, correcting each and every of our presentations. For actively supporting our endeavour of becoming PhD students. And of course for proof-reading my thesis. All your efforts have not been taken for granted. Thanks!

Many thanks to Karlheinz Meier for allowing me to join his workgroup and for giving me the opportunity to continue working on something I really enjoy and support with all my heart. Also many thanks to Manfred Stärk for supporting my further studies!

My deepest gratitude and thanks to Katja, for always being at my side and supporting me, even through times of doubt. May Chtugel's mercy (or the Force?) always be with you! And many thanks for proof-reading the whole thesis in such a short time!

Last but certainly not least, many thanks to my family, especially my parents. Thanks for supporting me throughout my whole life. You are the best!

Finally, there remains only one thing to be said:

Praise the sun!

Solaire of Astora, Dark Souls

Statement of Originality (Erklärung):

I certify that this thesis, and the research to which it refers, are the product of my own work. Any ideas or quotations from the work of other people, published or otherwise, are fully acknowledged in accordance with the standard referencing practices of the discipline.

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

Heidelberg, September 13, 2016

.....
(signature)