Department of Physics and Astronomy

University of Heidelberg, Ruperto Carola

Electronic Vision(s)

Master Thesis in Physics

submitted by

Agnes Korcsák-Gorzó

born in Hamburg, Germany

December 2017

Simulated Tempering in Spiking Neural Networks

This Master Thesis has been carried out by

Agnes Korcsák-Gorzó

at the

Kirchoff-Institute for Physics

University of Heidelberg, Ruperto Carola

under the supervision of

Prof. Dr. Karlheinz Meier

Abstract

In restricted Boltzmann machines, complex high-dimensional data is often represented by a probability distribution which has potentially deep local modes in its energy landscape. These modes lead to problems with mixing when the network performs generation tasks, since conventional sampling algorithms based on a Markov chain may take a long time to escape from them. Specifically for networks with leaky integrate-and-fire neurons, we show that an appropriate modulation of the background Poisson noise poses a solution to these problems. Poisson rate variation leads to a rescaling of the energy landscape analogous to simulated tempering. We introduce a mapping between the temperature defined for networks with abstract units and the Poisson noise rate in LIF networks. A rate variation scheme based on this principle facilitates the network to jump out of local minima and mix between different modes. We thereby suggest a functional role of the macroscopic neural oscillations observed in the cortex with potential applications for artificial generative neural networks.

Zusammenfassung

In Restricted Boltzmann Maschinen werden komplexe hochdimensionale Daten oft mit einer Wahrscheinlichkeitsverteilung dargestellt, die potenziell tiefe lokale Moden in der Energielandschaft aufweist. Diese Moden verursachen Probleme mit Mixing, wenn die Netzwerke die Daten wieder erzeugen. Weil konventionelle auf Markov Ketten basierende Algorithmen eine lange Zeit brauchen, den Moden zu entkommen. Speziell für Netzwerke aus Integrate-and-Fire Neuronen mit Leckstrom (LIF Neuronen), zeigen wir, dass eine passende Modulation des Hintergrund-Poisson-Rauschens eine Lösung für dieses Problem darstellt. Die Variation der Poisson Rate führt zu einer Reskalierung der Energie-Landschaft analog zu Simulated Tempering. Wir führen eine Abbildung zwischen der Temperatur definiert in Netzwerken mit abstrakten Einheiten und der Poisson Rate in LIF Netzwerken ein. Ein Raten-Variations-Schema basierend auf diesem Prinzip erleichtert dem Netzwerk das Entkommen aus dem lokalen Minimum und den Wechsel zwischen verschiedenen Moden. Wir schlagen hiermit eine funktionale Rolle der makroskopischen neuronalen Oszillationen, die im Kortex beobachten wurden, für potenzielle Anwendungen in künstlichen generativen neuronalen Netzen vor.

Contents

1 Introduction

Every night, during sleep, our brains generate dreams. According to latest results, the wake brain is considered to encode memories, which are consolidated during sleep (Rasch and Born, 2013). Memory consolidation is realized by reactivating the newly encoded neuronal memory representations as well as transforming and stabilizing these for embedding into long-term memory (ibid.). For us, dreaming seems to be the most natural phenomenon. But, this ordinary task for the brain poses challenging problems to machine learning. For algorithms, already a set of thousand 800-pixel gray shaded images of handwritten digits count as high-dimensional data, far from being comparable to the dimensionality of neuronal memories. After training algorithms on such images, during "reactivation", problems occur which manifest in the generation of repetitive patterns with high similarity. This thesis proposes a solution to the repetitive pattern generation problem and is inspired by neurophysiological principles that are suggested to play a role in information processing during sleep.

Neural networks with a specific architecture, called *restricted Boltzmann machine (RBM)*, can be trained amongst others on the above mentioned handwritten digits from the MNIST data set (LeCun et al., 1998). Samples from the multivariate probability distribution over the training data, can be obtained by constructing a *Markov chain* as done in *Markov chain Monte Carlo* (MCMC) methods.

RBMs can also be constructed with biologically plausible *leaky integrate-and-fire* (LIF) neurons opposed to the abstract units of conventional neural networks and realize sampling, as demonstrated in Petrovici et al. (2016). With both kinds of RBMs, one property of the sampling algorithms becomes crucial: *mixing* describes the ability of the algorithm to explore different regions of the data space uniformly. However, with both also problems with mixing arise. The reason for these difficulties is that training on high-dimensional data sets creates multi-modal energy landscapes in RBMs. The sampler cannot escape from local minima which are separated by high energy barriers.

In LIF networks, short-term plasticity (STP) with certain parameter ranges facilitates mixing, as presented in Leng et al. (2017). Developing a different solution for LIF networks is the goal of this thesis. For RBMs with abstract units, mixing can be improved by an algorithm called *adaptive simulated tempering (AST)* (Salakhutdinov, 2010) which is inspired from tempering methods as the name suggests. Within the algorithm, the network adaptively visits higher temperatures that flatten the

energy landscape, enabling the network to escape local modes. But, is simulated tempering applicable to LIF networks? What is the analogy for temperature in spiking neural networks?

The essential clue for a possible analogy arises from the fact that each change in temperature affects the slope of the activation function of the abstract units in AST. Similar mechanisms can be achieved in LIF sampling networks by changing the background noise either by varying the synaptic weights or the rates of the noise input. Whilst periodic changes of the synaptic strength of the noise input are rather uncommon in neural tissue, the occurrence of rate oscillations is an ubiquitous phenomenon. This phenomenon carries the name *neural oscillations* and occurs in a range of frequencies and amplitudes. These *brain waves* are widely studied, with the non-invasive observation method, *electroencephalography*, invented almost a century ago (Berger, 1929). Neural oscillations are assumed to play a role in information and memory processing especially during sleep. Interestingly, the distinct frequencies, labeled by Greek letters (γ-, α-, β-, ... wave), apparently encode distinct functions, for instance, in the sleep phase more low-frequent waves are present than in the wake-state (Buzsáki, 2006). Our driving question is, how background noise modulations in an LIF-based RBM will influence its generative properties.

As one of the prerequisites to answering the question, a mapping between temperatures and noise rate is established.

On the technical side, a software framework is constructed that enables the generation of inhomogeneous Poisson processes. We embed the C++ based varying Poisson noise generator introduced in Breitwieser (2015) in the *spike based sampling (sbs)* (ibid.) framework. Then, we wrap the generator into a module to make it usable with the group's primary neural simulation software, NEST *(Gewaltig and Diesmann, 2007)*. The result of this coding work is a Poisson noise generator that is maximally flexible in terms of customized Poisson noise variation patterns: it takes any succession of times and rates and applies the rate changes at the chosen points in time. Due to the limitations of the neural simulation software, the schedule of the rate variation has to be entirely defined in advance to the simulation, i.e., adaptive changes are currently not possible without pausing simulation.

Despite these limitations, we realize different inhomogeneous Poisson noise patterns, especially sinusoidally-shaped oscillations. Part of the study is dedicated to finding optimal sine waves for generating clear and diverse images. Eventually, the mixing performance of spike-based tempering is compared with present mixing facilitation mechanisms.

Outline

The main part of this thesis has three pillars: theory, experiments and software. In Chap. 2, the background knowledge concerning machine learning and biological principles is provided. Amongst others, we thoroughly introduce the previously mentioned concepts of sampling methods, Boltzmann machines (Sec. 2.1) and learning (Sec. 2.2) in the abstract domain. Also, the mixing problem and its solution for abstract networks is discussed (Sec. 2.3). An introduction to spiking neurons (Sec. 2.4) is followed by biologically inspired RBMs and sampling (Sec. 2.5). Subsequently, the mixing problem for LIF networks and the current solution via STP (Sec. 2.6) is presented as well as neural oscillations (Sec. 2.7) discussed in view of our later established alternative solution.

In Chap. 3, the experiments, the mapping between abstract temperatures and Poisson noise rates (Sec. 3.2) is established and the shift in the activation function compensated (Sec. 3.3), followed by rate variation schemes (Sec. 3.4). Eventually, utilizing all of the aforementioned techniques together with the modified software, an LIF-based RBM is stimulated with inhomogeneous noise and its generative properties evaluated (Sec. 3.7). In Chap. 4, the utilized software framework is described as well as the implemented modifications and extensions (Sec. 4.5).

2 Theoretical Background

In this chapter, we introduce concepts and techniques used in the later chapters and provide theoretical background to the experiments. In the first of two major building bocks of the chapter, concepts from machine learning are introduced. In the second building block, these concepts are expressed in a neurobiologically inspired framework. First, methods to sample from probability distributions based on Markov chains are introduced, followed by a specific network architecture, called Boltzmann machine, that has a Boltzmann distribution as stationary distribution and assigns high probabilities to vectors it is trained on (Sec. 2.1). Training algorithms for RBMs are presented in Sec. 2.2. In large abstract networks that process high-dimensional and complex data, the mixing problem arises in sampling and learning. This problem can be mitigated by tempering approaches (Sec. 2.3). The second block starts with a short introduction to LIF neurons (Sec. 2.4), followed by biologically inspired sampling approaches and LIF-based RBMs. The mixing problem in LIF networks and a present solution exploiting short-term plasticity (STP) are described in Sec. 2.6. In later chapters, we develop an alternative approach, which aims to find an analogy of tempering in LIF networks inspired by neural oscillations in the brain (Sec. 2.7). Last but not least, the self-contained Sec. 2.8 comprises evaluation methods to quantify sampling accuracy of the networks and diversity of the generated data.

2.1 Sampling Theory and Boltzmann Machines

In statistics, sampling denotes the action to choose examples from a set in such a way that representative characteristics about the composition of the set can be obtained. Especially, it is a mean to approximate distributions. We present Markov chain Monte Carlo (MCMC) methods (Sec. 2.1.1) as certain type of sampling algorithms as well as two variants thereof: the Metropolis Hastings algorithm (Sec. 2.1.2) and Gibbs sampling (Sec. 2.1.3). Eventually we introduce Boltzmann machines (Sec. 2.1.4) as a neural network that can sample from Boltzmann distributions, solely defined by weights and biases.

2.1.1 Markov Chain Monte Carlo Methods

Markov chain Monte Carlo (MCMC) methods construct a Markov chain that has the target distribution as its equilibrium distribution. It was invented by Ulam in the 1940s and implemented on ENIAC by von Neumann. A Markov chain is a process that obeys the Markov property, i.e., it lacks memory of states further back than a certain number of previous ones. In a first-order Markov chain the probability at a time depends on the previous state

$$p\left(\mathbf{z}^{(t)}|\mathbf{z}^{(0)},...,\mathbf{z}^{(t-1)}\right) = p\left(\mathbf{z}^{(t)}|\mathbf{z}^{(t-1)}\right) . \tag{2.1}$$

With increasing number of sampling steps the desired probability distribution is approached.

2.1.2 Metropolis-Hastings Algorithm

The Metropolis-Hastings algorithm is an MCMC method that iteratively obtains random samples from a probability distribution from which direct sampling is difficult. It was first developed only for symmetric distributions by Metropolis et al. (1953) and later generalized by Hastings (1970). The samples $\mathbf{z}^{(0)}, \mathbf{z}^{(1)}, ..., \mathbf{z}^{(t)}$ create a first order Markov chain. The more samples are produced, the better the target distribution can be approximated. The algorithm is initialized by an arbitrary sample $\mathbf{z}^{(0)}$ and a proposal distribution $g\left(\mathbf{z}'|\mathbf{z}^{(t)}\right)$ constructed similar to the target distribution but easier to sample from. A new state $\mathbf{z}'$ is proposed according to the proposal distribution and accepted with the acceptance probability

$$A\left(\mathbf{z}', \mathbf{z}^{(t)}\right) = \min\left(1, \frac{\tilde{p}\left(\mathbf{z}'\right)}{\tilde{p}\left(\mathbf{z}^{(t)}\right)} \frac{g\left(\mathbf{z}^{(t)}|\mathbf{z}'\right)}{g\left(\mathbf{z}'|\mathbf{z}^{(t)}\right)}\right) , \tag{2.2}$$

where $\tilde{p}\left(\mathbf{z}^{(t)}\right)$ is either the unnormalized target distribution or any function proportional to the target distribution . If the candidate value is accepted, it is used in the next iteration, otherwise the current value is reused.

Intuitively, attempts to move to more probable points in the state space than the current position, i.e., to higher density regions of the target distribution, are always accepted. Attempts to move to less probable points, are the more likely rejected the larger the relative decrease in probability is. The chain hence tends to explore high-density regions with occasional excursions to low-density regions, but overall following the target distribution.

In contrast to simple rejection sampling methods, this algorithm does not suffer from the curse of dimensionality since the partition function is canceled out in the acceptance probability and hence not used to produce the samples. However, in multivariate distributions, new sample points are multi-dimensional. High dimen-

sions form an obstacle in finding the suitable proposal distribution since different individual dimensions behave differently. The jumping width needs to fit all dimensions at once, otherwise the mixing will be very slow. This problem is solved in a variant of Metropolis Hastings, called Gibbs sampling.

2.1.3 Gibbs sampling

Gibbs sampling (GS) is a special case of the Metropolis-Hastings invented by Geman and Geman (1984) and hence also an MCMC method. In Gibbs sampling a new sample is chosen for every dimension, i.e., for every component $z_k^{(t)}$ of the current state vector $\mathbf{z}^{(t)}$ separately according to its conditional distribution

$$g\left(\mathbf{z}'|\mathbf{z}^{(t)}\right) = p\left(z_k'|\mathbf{z}_{\setminus k}^{(t)}\right) . \tag{2.3}$$

Combined with

$$p\left(\mathbf{z}\right) = p\left(z_k|\mathbf{z}_{\setminus k}\right) p\left(\mathbf{z}_{\setminus k}\right) , \tag{2.4}$$

the acceptance probability becomes

$$A\left(\mathbf{z}'|\mathbf{z}^{(t)}\right) = \frac{p\left(\mathbf{z}'\right)}{p\left(\mathbf{z}^{(t)}\right)} \frac{g\left(\mathbf{z}^{(t)}|\mathbf{z}'\right)}{g\left(\mathbf{z}'|\mathbf{z}^{(t)}\right)} = 1 . \tag{2.5}$$

An acceptance probability equal to 1 means that the proposed state is always accepted. One Gibbs step is complete when all components of state $\mathbf{z}$ are updated. The updating order is arbitrary, but usually a fixed periodic order is used. In each Gibbs sampling step the value of one component is replaced by a value drawn from the distribution of that component conditioned on the values of the most recent values of the other components, e.g. for the case of a three-dimensional state vector:

$$\begin{aligned} p\left(z_1|z_2^{(t)}, z_3^{(t)}\right) &\rightarrow z_1^{(t+1)} , \\ p\left(z_2|z_1^{(t+1)}, z_3^{(t)}\right) &\rightarrow z_2^{(t+1)} , \\ p\left(z_3|z_2^{(t+1)}, z_3^{(t+1)}\right) &\rightarrow z_3^{(t+1)} . \end{aligned} \tag{2.6}$$

2.1.4 Boltzmann Machine

A Boltzmann machine (BM) is a neural network of all-to-all connected stochastic binary units developed in Hinton and Sejnowski (1983). The connections are sym-

metric, such that it forms an undirected graphical model. It shares properties with Markov random fields and Ising models with the difference that the interaction weights of the Boltzmann machine are learned and not hand-designed or random. The total input y to a unit i is given by

$$y_i = b_i + \sum_j z_j W_{ij} \,, \tag{2.7}$$

where b_i is the bias of unit i and W_{ij} the weight entry in the weight matrix $\mathbf{W}$ between units i and j. z_j has value 1 if unit j is on and 0 otherwise. The probability that unit i gets activated is given by a logistic function, a sigmoid curve,

$$p\left(z_i = 1\right) = \frac{1}{1 + \exp\left(-y_i\right)} = \sigma(y_i) \,. \tag{2.8}$$

Sequential updates of the units lead to the equilibrium distribution of the network, the Boltzmann distribution. The probability of a state vector is defined by its energy

$$p\left(\mathbf{z}\right) = \frac{1}{Z} \exp\left(-E(\mathbf{z})\right) \,, \tag{2.9}$$

where the partition function Z ensures the correct normalization and is given by

$$Z = \sum_{\mathbf{z}} \exp\left(-E(\mathbf{z})\right) \,, \tag{2.10}$$

with the energy of a state vector defined like in Hopfield networks as

$$E\left(\mathbf{z}\right) = \sum_{i<j} W_{ij} z_i z_j - \sum_i b_i z_i \,. \tag{2.11}$$

Restricted Boltzmann Machine

The fully-connected Boltzmann machine turned out to be impractical in terms of learning. But, with appropriate restrictions imposed on the connections, training becomes easier, as shown in Smolensky (1986). The restricted Boltzmann machine (RBM) consists of a layer of visible and a layer of hidden units, with no intralayer connections, i.e., the weight matrix is still symmetric, but has a vanishing trace:

$$W_{ii} = 0, \; W_{ij} = W_{ji} \,, \tag{2.12}$$

and hence forms a bipartite graph. The probability of a state vector can be expressed by the visible and hidden state vectors

$$p(\mathbf{v}, \mathbf{h}) = \frac{1}{Z}\exp\left[-E\left(\mathbf{v}, \mathbf{h}\right)\right] , \tag{2.13}$$

$$E(\mathbf{v}, \mathbf{h}) = -\sum_i a_i v_i - \sum_j b_j h_j - \sum_{i,j} v_i W_{ij} h_j , \tag{2.14}$$

where a_i are the the biases of the visible units and b_j of the hidden units. The hidden unit activations are independent from each other given the visible unit activations and vice versa. The conditional probabilities are given by

$$p\left(\mathbf{v}|\mathbf{h}\right) = \prod_{i=1}^{m} p\left(v_i|\mathbf{h}\right) , \tag{2.15}$$

$$p\left(\mathbf{h}|\mathbf{v}\right) = \prod_{j=1}^{n} p\left(h_j|\mathbf{v}\right) , \tag{2.16}$$

where m is the number of visible and n the number of hidden units. For single units, the activation probabilities are given by

$$p(h_j = 1|\mathbf{v}) = \sigma\left(b_j + \sum_{i=1}^{m} W_{ij} v_i\right) , \tag{2.17}$$

$$p(v_i = 1|\mathbf{h}) = \sigma\left(a_i + \sum_{j=1}^{n} W_{ij} h_j\right) . \tag{2.18}$$

2.2 Learning Algorithms

In this work, trained networks are utilized, and the training itself is not performed. A detailed discussion about the training process can be found in Leng (2014) . Also, as mentioned earlier, since training for plain BMs is hardly feasible, here the one for RBMs is discussed.

An untrained Boltzmann machine generates samples from the Boltzmann distribution. Training means to present the Boltzmann machine some training data, here mainly the MNIST data set, i.e., clamp the visible layer to the data and update its parameters according to a predefined rule after each presentation. The goal of each update is to maximize the probability of the visible layer $p(\mathbf{v})$ with respect to the input, or more precisely the log-likelihood $\ln p\left(\mathbf{v}\right)$is evaluated. Taking

the average over the training set and exploiting the restrictions of the RBM, the derivatives with respect to the model parameters, the weights W_{ij} and the biases of visible units a_i and hidden units b_j read

$$\langle \frac{\partial \ln p(\mathbf{v})}{\partial W_{ij}} \rangle = \eta \left(\langle v_i h_j \rangle_{\text{data}} - \langle v_i h_j \rangle_{\text{model}} \right) , \tag{2.19}$$

$$\langle \frac{\partial \ln p(\mathbf{v})}{\partial a_i} \rangle = \eta \left(\langle v_i \rangle_{\text{data}} - \langle v_i \rangle_{\text{model}} \right) , \tag{2.20}$$

$$\langle \frac{\partial \ln p(\mathbf{v})}{\partial b_j} \rangle = \eta \left(\langle v_i \rangle_{\text{data}} - \langle v_i \rangle_{\text{model}} \right) . \tag{2.21}$$

These derivatives are used to update the model parameters via gradient ascent with a constant learning rate. In the course of this learning phase, the energy corresponding to the states of the training samples are lowered and their probability increased, enabling the generation of similar data. Unfortunately, the states in the model terms imply the evaluation of the partition function - an unfeasible operation in high-dimensional state spaces. Hence, in the following a method is discussed that approximates the expectation.

2.2.1 Contrastive Divergence

Contrastive divergence (CD) proposed by Hinton (2002) offers a way to avoid the intractable model average over $z_i z_j$, which is computationally cheaper than approximating it via MCMC sampling and has a lower variance than computing

$$\frac{\partial \ln p\left(\mathbf{v}\right)}{\partial W_{ij}} . \tag{2.22}$$

In k-step contrastive divergence, the difference between a term involving $v_i^{(0)}$ the visible layer clamped to the input and one involving $v_i^{(k)}$ the visible layer after k Gibbs sampling steps is evaluated for the

$$\text{CD}_k \left(\mathbf{v}^{(0)}, W_{ij} \right) = p\left(h_j = 1 | \mathbf{v}^{(0)} \right) v_i^{(0)} - p\left(h_j = 1 | \mathbf{v}^{(k)} \right) v_i^{(k)} , \tag{2.23}$$

$$\text{CD}_k \left(\mathbf{v}^{(0)}, a_i \right) = v_i^{(0)} - v_i^{(k)} , \tag{2.24}$$

$$\text{CD}_k\left(\mathbf{v}^{(0)}, b_j\right) = p\left(h_j = 1|\mathbf{v}^{(0)}\right) - p\left(h_j = 1|\mathbf{v}^{(k)}\right) . \tag{2.25}$$

CD-1 is often sufficient.

2.2.2 Persistent Contrastive Divergence

In persistent contrastive divergence (PCD) described in Tieleman and Hinton (2009), as the name implies, a Markov chain is initialized only once in the beginning with the first training data point and afterwards runs persistently. This approach improves the approximation of the model distribution, provided the learning rate is small enough to ensure slow changing of the model parameters.

2.3 Mixing Problem and Solution

During training, an RBM assigns high probabilities to the training data, which correspond to low energy values. In the case of very inhomogeneous classes, in between these energy modes, high energy barriers are created. As a result, the energy landscape becomes increasingly rough during training and the Gibbs sampler is prevented from covering all of the relevant state space. This again, destabilizes the learning dynamics and leads to poor parameter estimates. These difficulties also become apparent in sampling-based generation tasks in the form of a high similarity in consecutively generated patterns. Effectively, it takes a longer time until the BM converges to the target distribution. The ability of generative models based on sampling to travel over energy barriers is termed *mixing*. In the following a method by Salakhutdinov (2010) is described that utilizes tempering in sampling (Sec. 2.3.1) and learning (Sec. 2.3.2) to facilitate mixing.

2.3.1 Adaptive Simulated Tempering

Adaptive Simulated Tempering (AST) is an algorithm that mixes better than Gibbs sampling as demonstrated in Salakhutdinov (2010). It is a combination of the Wang-Landau algorithm in Wang and Landau (2001) and Simulated Tempering from Marinari and Parisi (1992), which are addressed in the following.

Wang-Landau Algorithm

Let us consider a probability distribution

$$p\left(\mathbf{x};\theta\right) = \frac{1}{Z}\exp\left(-E\left(\mathbf{x},\theta\right)\right) , \tag{2.26}$$

defined over a state space X that is segmented into K disjoint sets $\{X_k\}_{k=1}^{K}$. The states in each partition are updated via Gibbs sampling. In order to generate a Markov chain that travels uniformly over the state space, the Wang-Landau (WL) algorithm introduces for each segment a probability adjusting factor g_k into the acceptance probability of the Metropolis-Hastings algorithm. Each time the network remains in one partition, the corresponding adjusting factor increases. Thus, the probability mass of the entire partition decreases and the chain is induced to leave into other parts of the state space. As a result, the chain spends an equal amount of time in each set X_k. In particular, the algorithm consists of three stages:

1. $\mathbf{g}^t$, a K-dimensional vector, holds all the adjusting factors for a certain point in time. Initialize all its elements to 1 at $t = 0$.

2. Generate the consecutive state $\mathbf{x}^{t+1}$ given model state $\mathbf{x}^t$ via Gibbs sampling using the constant probability distribution:

$$p\left(\mathbf{x};\theta,\mathbf{g}^t\right) \sim \sum_{k=1}^{K} \frac{p\left(\mathbf{x},\theta\right)}{g_k^t} I\left(\mathbf{x}\in X_k\right),\ I(x) = \begin{cases} 1 & \text{if } x \text{ is true} \\ 0 & \text{otherwise} \end{cases} . \tag{2.27}$$

3. Calculate the adaptive adjusting factors from the new state

$$g_k^{t+1} = g_k^t \left[1 + \gamma_t I\left(\mathbf{x}^{t+1}\in X_k\right)\right] . \tag{2.28}$$

   Since $\gamma_t > 0$ is involved in the update of g_k, it is called the weight adapting factor.

In the limit of infinite time and vanishing weight adapting factors for all partitions, the probability converges

$$\frac{g_k^t}{\sum_i g_i^t} \to p\left(\mathbf{x}\in X_k\right) , \tag{2.29}$$

signifying that the adaptive adjusting factors asymptotically converge to optimal values, ensuring a uniform exploration of the state space. However, the problem with this algorithm lies in the fact that it is difficult to find a suitable partitioning. Appropriate partitions can be established in a natural way via simulated tempering.

Simulated Tempering (ST)

The principle of simulated tempering (ST) (Marinari and Parisi, 1992) or simulated annealing is borrowed from metallurgy (Kirkpatrick et al., 2007). The power of simulated annealing can be clarified with an illustrative analogy: In a landscape with

various unequally deep modes, conventional algorithms correspond to a particle that bounces over the area, loosing energy until it comes to rest in one of the minima - not necessarily the global one. It is an MCMC algorithm which overcomes a problem of many deterministic search and gradient-based methods, namely to get trapped in local minima. Compared to simulated annealing, temperature in ST becomes a dynamic variable. The energy loss is smaller and happens on longer time scales, such that the particle reaches the deepest mode.

Let us turn to the formulation of simulated tempering in neural networks. There, the goal is to sample from the target distribution, $p(\mathbf{x})$, over the multi-dimensional states $\mathbf{x}$. On that account, a Markov chain is simulated from the joint distribution

$$p\left(\mathbf{x}, k\right) \sim c_k \exp\left(-\beta_k E\left(\mathbf{x}\right)\right) , \tag{2.30}$$

where c_k are constants and $0 < \beta_K < \beta_{K-1} < ... < \beta_1 = 1$ are similar to the inverse temperature $1/(k_\mathrm{B}\, T)$. In particular, this is the combined state space of the states and the temperature. The states are only accepted if k reaches 1. In the sampling process two transition operators are evaluated alternately in a row. First, $\mathbf{x}$ is updated via Gibbs sampling from the conditional probability

$$p\left(\mathbf{x}|k\right) = \frac{1}{Z_k} \exp\left(-\beta_k E\left(\mathbf{x}\right)\right) , \tag{2.31}$$

where Z_k is the partition function. Second, conditioned on $\mathbf{x}$, k is sampled using the Metropolis-Hastings algorithm with a proposal distribution

$$\text{for } 2 \leqq k \leqq K-1 \quad q\left(k+1|k\right) = q\left(k-1|k\right) = \frac{1}{2} , \tag{2.32}$$

where $q\left(2|1\right) = q\left(K-1|K\right) = 1$ and 0 otherwise. The strength of this principle, lies in the enhancement of mixing between local modes due to occasional visits to high temperatures, resembling the increase of transition rates with higher temperatures in thermodynamics. However, a necessary requirement is that the Markov chain uniformly travels over the temperature range ensured by setting c_k proportional to the inverse of the partition sum of that subspace. Since the partition is computationally intractable, the WL algorithm comes into play at this point: it enables the partition of the state space into K sets $\{X_k\}_{k=1}^{K}$ covering the different temperature values and assign each with the appropriate adaptive adjusting factor. As described above, a rejected move, results into a decrease of the probability mass of that temperature, slowly favoring a leave to another temperature.

AST algorithm

The AST algorithm is initialized with a set of adaptive adjusting factors $\{\mathbf{g}_k\}_{k=1}^{K}$ and an initial model state $\mathbf{x}^1$ at the first temperature $k = 1$. For $n = 1, 2, ..., N$ iterations alternatively, given $\mathbf{x}^n$ a new new state $\mathbf{x}^{n+1}$ from $p(\mathbf{x}, k^n)$ is Gibbs-sampled and given k^n, k^{n+1} is sampled from proposal distribution $q(k^{n+1} \leftarrow k^n)$ by Metropolis-Hastings sampling, i.e., it is accepted with

$$A\left(k^{n+1}, k^n\right) = \min\left(1, \frac{p\left(\mathbf{x}^{n+1}, k^{n+1}\right) q\left(k^n \leftarrow k^{n+1}\right) g_{k^n}}{p\left(\mathbf{x}^{n+1}, k^n\right) q\left(k^{n+1} \leftarrow k^n\right) g_{k^{n+1}}}\right) . \tag{2.33}$$

Afterwards the adaptive adjusting factors are updated:

$$g_i^{n+1} = g_i^n \left(1 + \gamma_n I\left(k^{n+1} = i\right)\right) , \; i = 1, 2, ..., K . \tag{2.34}$$

Each time k reaches the value 1, the sample is kept. The resulting sequence approximates the target distribution. The convergence in Eqn. 2.29 and the resulting benefit are still valid.

2.3.2 Coupled Adaptive Simulated Tempering

Coupled AST (CAST) is a learning method introduced in Salakhutdinov (2010), in which the mixing enhancement property of AST is exploited, when it comes to learning complex data. This kind of data is represented by a multi-modal energy landscape, so the part of the training that is performed by Gibbs sampling, is prone to get stuck. In CAST a slow and a fast Markov chain are run in parallel. PCD updates of the model parameters are performed in the slow chain after every Gibbs step. In the fast chain, AST adaptively changes temperatures indexed by k to facilitate mixing. The fast chain triggers an exchange of the current states between the two chains, as soon as it reaches the lowest temperature 1. When the chain reaches the low-temperature region, the frequent visits of the $k = 1$ temperature lead to frequent swaps between the two chains, which does not help mixing. To avoid this case, in practice, a fixed number of Gibbs updates need to elapse before the next swap can happen.

2.4 Spiking Neurons

In the subsequent analytical and experimental considerations of neurons, we use one of the simplest spike-based neuron model called leaky integrate-and-fire (LIF) neuron model. In contrast to multi-compartment models, the complex morphology of nerve cells is entirely neglected here, modeling the neuron as point-like. By

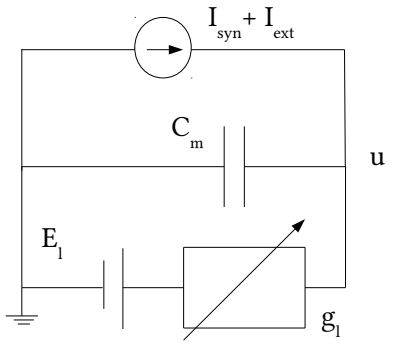


Figure 2.1: Electric circuit modeling the membrane of a nerve cell. C_m, the membrane capacitance, is realized by a capacitor in parallel with a resistor modeling the leak conductance and a voltage source modeling the resting potential E_l. I^ext and I^syn are jointly represented by a parallel current source.

abstracting away a number of other physiological details, it provides a tractable and yet realistic model. First, we introduce the differential equation governing the membrane potential dynamics (Sec. 2.4.1) and extend them by describing mathematical descriptions for corresponding synapse models (Sec. 2.4.2).

2.4.1 Leaky Integrate-and-Fire (LIF) Neurons

The Hodgkin-Huxley model (Hodgkin and Huxley, 1952) introduces a differential equation for the membrane potential, which further depends on a set of three differential equations for ion channels, forming a system of coupled, nonlinear differential equation. The leaky integrate-and-fire (LIF) model is a simplification thereof and models the dynamics of the membrane potential u with only one differential equation

$$C_\mathrm{m}\frac{\mathrm{d}u}{\mathrm{d}t} = g_\mathrm{l}(E_\mathrm{l} - u) + I^\mathrm{syn} + I^\mathrm{ext}\,, \tag{2.35}$$

with C_m being the membrane capacitance. A passive leak term is proportional to u, comprising the leak conductance g_l and leak potential E_l. I^syn represents the synaptic current and I^ext the external input, which make up the total current. The equations of both models result from modeling the membrane of a neuron with an electric circuit (Fig. 2.1). For spike emission two distinct values of the membrane potential, the threshold and the reset potential, ϑ and ϱ, are determined. Whenever

the membrane potential crosses ϑ the neuron spikes

$$\text{neuron spikes at } t = t_{\text{spike}} \Leftrightarrow u(t_{\text{spike}}) = \vartheta\,, \tag{2.36}$$

and is afterwards reset to ϱ for a duration τ_{ref} in which the neuron is refractory

$$u\left(t_{\text{spike}} < t \leq t_{\text{spike}} + \tau_{\text{ref}}\right) = \varrho\,. \tag{2.37}$$

A spike train refers to the temporal sequence of spike events at times t_s and is expressed as

$$\rho(t) = \sum_{\text{spikes } s} \delta(t - t_s)\,. \tag{2.38}$$

In the case of zero synaptic input and a constant external current, Eqn. 2.35 can be solved as

$$u(t) = E_{\text{l}} + \frac{I^{\text{ext}}}{g_{\text{l}}} + \left(u_0 - E_{\text{l}} - \frac{I^{\text{ext}}}{g_{\text{l}}}\right) e^{-\frac{t}{\tau_{\text{m}}}}\,, \tag{2.39}$$

where $u_0 = u(t = 0)$ is the initial value of the membrane potential and $\tau_{\text{m}} = C_{\text{m}}/g_{\text{l}}$ the membrane time constant. If the equilibrium value $E_{\text{l}} + I^{\text{ext}}/g_{\text{l}}$ of u is smaller than ϑ the time course of the membrane potential represents charging or discharging the capacitor. In the supra-threshold case, the membrane potential alternately describes an exponential rise up to the threshold and a jump discontinuity when reset to ϱ. By imposing $u_0 = \varrho$ and $u(T) = \vartheta$, where T is the time between the end of the refractory period and the first crossing of the threshold, as boundary conditions on Eqn. 2.39 the firing steady-state firing rate can be calculated by

$$\nu = (\tau_{\text{ref}} + T)^{-1} = \left[\tau_{\text{ref}} + \tau_{\text{m}} \ln\left(\frac{\varrho - E_{\text{l}} - \frac{I^{\text{ext}}}{g_{\text{l}}}}{\vartheta - E_{\text{l}} - \frac{I^{\text{ext}}}{g_{\text{l}}}}\right)\right]^{-1}\,. \tag{2.40}$$

2.4.2 Current-, Conductance-Based Synapses

Likewise, the biological synaptic dynamics is reduced to its essentials. Common synapse models introduce an interaction kernel $\epsilon(t)$ weighted with w at each spike event. For each neuron these kernels sum up linearly over its pre-synaptic partners k and their spike train, resulting into an overall synaptic effect

$$f^{\text{syn}}(t) = \sum_{\text{synapses } k} \sum_{\text{spikes } s} w_k \epsilon_k (t - t_s)\,.$$

In the case of conductance-based (COBA) neurons, this changes the membrane potential‘s conductance. As a consequence, the membrane potential drifts towards

the respective reversal potential, $E_{\mathrm{e}}^{\mathrm{rev}}$ or E_i^{rev}, depending on the excitatory or inhibitory nature of the synapse. The synaptic current hence reads

$$I^{\mathrm{syn}} = g_{\mathrm{e}}^{\mathrm{syn}} \left(E_{\mathrm{e}}^{\mathrm{rev}} - u\right) + g_{\mathrm{i}}^{\mathrm{syn}} \left(E_{\mathrm{i}}^{\mathrm{rev}} - u\right) , \tag{2.41}$$

which yields the COBA LIF equation when inserted into Eqn. 2.35

$$C_{\mathrm{m}} \frac{\mathrm{d}u}{\mathrm{d}t} = g_{\mathrm{l}} \left(E_{\mathrm{l}} - u\right) + g_{\mathrm{e}}^{\mathrm{syn}} \left(E_{\mathrm{e}}^{\mathrm{rev}} - u\right) + g_{\mathrm{i}}^{\mathrm{syn}} \left(E_{\mathrm{i}}^{\mathrm{rev}} - u\right) + I^{\mathrm{ext}} . \tag{2.42}$$

A closed-form solution of this ODE is in opposition to the plain LIF equation harder, in fact an exact solution cannot be given, but only an approximative, mainly for two reasons. First, the conductances $g_{\mathrm{e,\,i}}^{\mathrm{syn}}$ are temporally evolving which brings about a time dependence of the total conductance $g^{\mathrm{tot}} = g_{\mathrm{l}} + g_{\mathrm{e}}^{\mathrm{syn}} + g_{\mathrm{i}}^{\mathrm{syn}}$. Since the membrane potential is described by the effective time constant $\tau_{\mathrm{eff}} = C_{\mathrm{m}}/g^{\mathrm{tot}}$, its responsiveness becomes time dependent as well. Second, I^{syn} is proportional to the difference between u and the reversal potential. This leads to saturating effects especially for inhibitory post-synaptic potentials (PSPs), since the rest membrane potential is usually closer to the inhibitory reversal potential. By choosing an exponential interaction kernel

$$\epsilon(t) \propto \Theta(t) \exp\left(-\frac{t}{\tau^{\mathrm{syn}}}\right) , \tag{2.43}$$

the synaptic conductances can be expressed by

$$g_{x\in\{\mathrm{e,\,i}\}}^{\mathrm{syn}}(t) = \sum_{\mathrm{syn}\,k} \sum_{\mathrm{spk}\,s} w_k \Theta(t - t_s) \exp\left(-\frac{t - t_s}{\tau^{\mathrm{syn}}}\right) , \tag{2.44}$$

yielding total synaptic current of

$$I^{\mathrm{syn}}(t, u) = \sum_{x\in\{\mathrm{e,\,i}\}} \sum_{\mathrm{syn}\,k} \sum_{\mathrm{spk}\,s} w_k \Theta\left(t - t_s\right) \left(E_x^{\mathrm{rev}} - u\right) \exp\left(-\frac{t - t_s}{\tau^{\mathrm{syn}}}\right) . \tag{2.45}$$

Though COBA synapses are a biologically more plausible model for chemical synapses there also exist current-based (CUBA) models. Both are equally justified in the case of the point neuron model: The membrane potential is influenced by conductance changes whereas the soma is responsive to synaptic current changes. The differential Eqn. 2.35 for u stays the same for the CUBA case, with a synaptic current expressed by exponential interaction kernels:

$$I^{\text{syn}}(t) = \sum_{\text{syn } k} \sum_{\text{spk } s} w_k \Theta\left(t - t_s\right) \exp\left(-\frac{t - t_s}{\tau^{\text{syn}}}\right) . \tag{2.46}$$

Again, the overall synaptic current is formed by a linear summation of PSPs. With that in the CUBA case an exact solution can be derived.

2.5 Sampling in Biologically Inspired Networks

The concepts from 2.1 are translated to networks with spiking neurons. For abstract model neurons this was achieved with neural sampling in Buesing et al. (2011) (Sec. 2.5.1) and for LIF neurons in Petrovici (2016) (Sec. 2.5.2). This lead to the implementation of LIF-based BMs (Sec. 2.5.3) which are utilized later in the experiments.

2.5.1 Neural Sampling

Neural Sampling is an MCMC sampling method in networks of abstract model neurons (AMNs) developed by Buesing et al. (2011). It is the cornerstone to Boltzmann machines based on spiking neurons. In neural sampling, the binary vector $\mathbf{z}$ represents the firing activity of the network at time t

$$z_k(t) = 1 \Leftrightarrow v_k \text{ was active in } (t - \tau, t] , \tag{2.47}$$

i.e., a spike of neuron v_k sets the value of the associated binary value z_k to 1 for a duration of τ. Assuming that the membrane potential of the neuron with index k corresponds to the log odds of z_k being on, the neural computability condition (NCC)

$$u_k(t) = \log\left(\frac{p\left(z_k(t) = 1 | \mathbf{z}_{\setminus k}(t)\right)}{p\left(z_k(t) = 0 | \mathbf{z}_{\setminus k}(t)\right)}\right) , \tag{2.48}$$

is introduced. For the Boltzmann distribution the NCC implies the following membrane potential for neuron k

$$u_k(t) = b_k + \sum_{i=1}^{K} W_{ki} z_i(t) , \tag{2.49}$$

where b_k is the bias, describing the excitability of the neuron, W_{ki} is the synaptic strength between neuron k and i, and together with $z_i(t)$ they form the time course of the PSP induced in neuron k by the firing of neuron i. A number of non-binary internal variables $\zeta_1, ..., \zeta_K$ encode the exact firing time of the neuron in the time

interval $(t-\tau, t]$, which ensures the Markov property of sampling. These variables control the exit from the refractory state. The dynamics of ζ_k is determined by updates in discrete time steps. For a neuron model with absolute refractory mechanism, ζ_k is set to τ (refractory period) when neuron v_k fires and decreases with each update toward zero. The neuron is ready to spike again if $\zeta_k \leq 1$. The spike probability is defined via a logistic function

$$p\left[\zeta_k(t+1) = \tau | \zeta_k(t) \in \{0,1\},\, u_k(t)\right] = \sigma\left(u_k - \log\tau\right) . \tag{2.50}$$

2.5.2 LIF Sampling

The functioning of neural sampling relies on the intrinsic stochasticity of the network's units. A combination with models implying a spiking condition like Eqn. 2.36 is desirable to account for findings of in vitro experiments that reveal the deterministic nature of neurons. As a side remark, neurophysiologically more plausible are escape noise models, where the neurons spike stochastically in a range of membrane potential values (see e.g. in Gerstner et al. (2014)). Petrovici et al. (2016) achieves this goal of generating the required stochasticity in LIF neurons by embedding them in a spiking noisy environment. It is demonstrated that the network is able to approximate sampling from a well-defined target distribution. Further, it enables the realization of an LIF-based Boltzmann machine.

Noisy spiking environment

In the following, some approximations in the case of a noisy spiking environment according to Petrovici et al. (2016) are demonstrated. For the argument, I_{rec} and I_{ext} are neglected for the moment and the focus lies on the noise part of the synaptic current in Eqn. 2.35. A COBA neuron responds to bombardment with high-frequency Poisson noise with accelerated membrane dynamics, transitioning to the so called high-conductance state (HCS). In this state, the following differential equation holds

$$\tau_{\text{eff}} \frac{\mathrm{d}u}{\mathrm{d}t} = u_{\text{eff}} - u\,. \tag{2.51}$$

The introduced effective time constant τ_{eff} and effective membrane potential u_{eff} are related via the total conductance

$$g^{\text{tot}} = g_{\text{l}} + \sum_k g_k^{\text{syn}}\,, \tag{2.52}$$

which is dominated by the noise part of the conductance. For high input, the quotient of the standard deviation and the mean of g^{tot} converges to zero, thus it

can be replaced by its mean value in the equations

$$\tau_{\text{eff}} \approx \langle \tau_{\text{eff}} \rangle = \frac{C_{\text{m}}}{\langle g_{\text{tot}} \rangle}\,, \tag{2.53}$$

$$u_{\text{eff}} = \frac{I^{\text{ext}} + g_{\text{l}}\, E_{\text{l}} + \sum_i g_i^{\text{noise}} E_i^{\text{rev}}}{\langle g_{\text{tot}} \rangle}\,. \tag{2.54}$$

When τ_{eff} is small, the membrane potential is approximately equal to the effective membrane potential. With an approach from Ricciardi and Sacerdote (1979), Petrovici et al. (2016) have shown that under high frequency input, the diffusion approximation holds and the noise contribution of the current and the effective membrane potential can be described by an Ornstein-Uhlenbeck process, i.e., a mean-reverting random walk in continuous time

$$\mathrm{d}u(t) = \theta\,(\mu - u(t))\,\mathrm{d}t + \sigma \mathrm{d}W(t)\,. \tag{2.55}$$

$dW(t)$ is a Wiener process and here the characterizing parameters are expressed by neurophysiological parameters

$$\theta = \frac{1}{\tau_{\text{syn}}}\,, \tag{2.56}$$

$$\mu = \frac{I^{\text{ext}} + g_{\text{l}}\, E_{\text{l}} + \sum_i \nu_i w_i E_i^{\text{rev}} \tau_{\text{syn}}}{\langle g_{\text{tot}} \rangle}\,, \tag{2.57}$$

$$\sigma^2 = \frac{\sum_i \nu_i w_i^2\, (E_i^{\text{rev}} - \mu)^2\, \tau_{\text{syn}}}{\langle g_{\text{tot}} \rangle^2}\,. \tag{2.58}$$

Free membrane potential statistics under noise input

Under high enough Poisson noise input the membrane potential distribution of the CUBA (also COBA) neuron approximates a Gaussian distribution. The expectation value of the CUBA membrane potential is given by

$$E[u] = E_{\text{l}} + \frac{I^{\text{ext}}}{g_{\text{l}}} + \frac{\sum_{k=1}^{n} w_k \nu_k \tau_k^{\text{syn}}}{g_{\text{l}}}\,, \tag{2.59}$$

and its variance by

$$\text{Var}\,[u] = \sum_{k=1}^{n} \left[\frac{\tau_{\text{m}}\, \tau_k^{\text{syn}}}{C_{\text{m}}\,(\tau_{\text{m}} - \tau_k^{\text{syn}})}\right]^2 w_k^2 \nu_k \left(\frac{\tau_{\text{m}}}{2} + \frac{\tau_k^{\text{syn}}}{2} - 2\frac{\tau_{\text{m}}\, \tau_k^{\text{syn}}}{\tau_{\text{m}} + \tau_k^{\text{syn}}}\right)\,. \tag{2.60}$$

The expectation value of the COBA effective potential and effective time constant are given by

$$E[u_{\text{eff}}] = \frac{g_{\text{l}}\, E_{\text{l}} \; + I^{\text{ext}} + \sum_k w_k \nu_k \tau_k^{\text{syn}} E_k^{\text{rev}}}{g_l + \sum_k w_k \nu_k \tau_k^{\text{syn}}} = \langle u_{\text{eff}} \rangle \;, \tag{2.61}$$

$$E\left[\tau_{\text{eff}}\right] = \frac{C_{\text{m}}}{E\left[g^{\text{tot}}\right]} = \frac{C_{\text{m}}}{g_{\text{l}} + \sum_k w_k \nu_k \tau_k^{\text{syn}}} = \langle \tau_{\text{eff}} \rangle \;. \tag{2.62}$$

The expectation value of the COBA membrane potential is given by

$$E[u] = \frac{g_{\text{l}}\, E_{\text{l}} + I^{\text{ext}} + \sum_k w_k \nu_k \tau_k^{\text{syn}} E_k^{\text{rev}}}{g_l + \sum_k w_k \nu_k \tau_k^{\text{syn}}} \,, \tag{2.63}$$

and its variance by

$$\text{Var}\left[u\right] = \sum_{k=1}^{n} \left[\frac{\langle \tau_{\text{eff}} \rangle \, \tau_k^{\text{syn}} \left(E_k^{\text{rev}} - \langle u_{\text{eff}} \rangle \right)}{C_{\text{m}} \left(\langle \tau_{\text{eff}} \rangle - \tau_k^{\text{syn}} \right)} \right]^2 w_k^2 \nu_k \left(\frac{\langle \tau_{\text{eff}} \rangle}{2} + \frac{\tau_k^{\text{syn}}}{2} - 2 \frac{\langle \tau_{\text{eff}} \rangle \, \tau_k^{\text{syn}}}{\langle \tau_{\text{eff}} \rangle + \tau_k^{\text{syn}}} \right) . \tag{2.64}$$

For CUBA neurons, $\text{Var}[u]$ is proportional to the noise input rate ν_k and the squared input weight w_k. Changing these two terms leads to a change of the width of the Gaussian distribution. This results in a slope change in the cumulative function of the Gaussian, which approximates the change of the activation function. This role played by the noises serves as the foundation of the implementation of spike-based tempering.

Analytical LIF-based Activation Function

The goal of the following mathematical considerations is the analytical derivation of a sigmoid shaped activation function for LIF neurons like in neural sampling

$$p(z = 1) = \sigma(v) = \frac{1}{1 + \exp(-v)} \,. \tag{2.65}$$

LIF neurons in the HCS enter one of two modes after one spike: Either the bursting mode, where the free membrane potential stays suprathreshold, eliciting in total n spikes, or the freely evolving mode where the membrane potential is subthreshold again. The probability of an n-spike burst is denoted by P_n. The average time the membrane potential spends in freely evolving mode following this burst is denoted by T_n. The membrane distribution at the end of the first refractory period in a burst is approximately Gaussian distributed in the HCS. The probability that the membrane potential will be suprathreshold can be obtained via the cumulative density function. Together with the parameters in Eqns. 2.56, 2.57 and 2.58 char-

acterizing the OU process, the so called transfer function $p\left(u_n|u_{n-1}\right)$ between the membrane potentials at two consecutive spikes times in a burst can be retrieved. This function enters in the form of recursive integrals into the calculation of P_n and T_n. Finding the average time to reach the threshold from membrane potential value u_n is regarded as a first passage time problem, solvable with common methods from statistical physics and necessary for the calculation of T_n. With these expressions the activation function is evaluated as

$$p\left(z=1\right)=\frac{\sum_n P_n n\tau_{\text{ref}}}{\sum_n P_n\left(n\tau_{\text{ref}}+T_n\right)}\,. \tag{2.66}$$

Detailed derivation of all mentioned variables and further considerations can be found in Petrovici et al. (2013) and the corresponding supplementary material.

2.5.3 LIF-based Boltzmann Machine

Until now, it is not guaranteed that the activation function has the necessary sigmoid shape. This is achieved by fitting a sigmoid to the obtained values, yielding two fit parameters, α that rescales the membrane potentials and $\bar{u}_k^0$ that shifts them, yielding

$$p\left(z_k=1\right)=\sigma\left(\frac{\bar{u}_k-\bar{u}_k^0}{\alpha}\right)\,, \tag{2.67}$$

with $\bar{u}_k^0$ representing the membrane potential value, where the activation probability equals 0.5. When the neuron is disconnected from other neurons, its excitability can be varied by its rest membrane potential or external current. When I^{ext} is zero, for the correct conversion between the LIF and the abstract regime the biases read

$$b_k=\frac{\bar{u}_k-\bar{u}_k^0}{\alpha}\,. \tag{2.68}$$

Weights w_{kj} between LIF neurons k and j are translated into abstract weights W_{kj}. The reason for this lies in the fact, that instead of rectangular PSPs like in neural sampling, here, exponential-shaped PSPs are present and the membrane potentials are rescaled in Eqn. 2.67. The shape mismatch can be corrected for by matching the integral value under both PSPs within the refractory period. Another reason is that the units need to be adjusted.

$$W_{kj}=\frac{w_{kj}\left(E_{kj}^{\text{rev}}-\mu\right)}{\alpha C_{\text{m}}\left(1-\frac{\tau_{\text{syn}}}{\tau_{\text{eff}}}\right)}\left[\tau_{\text{syn}}\left(e^{-1}-1\right)-\tau_{\text{eff}}\left(e^{-\frac{\tau_{\text{syn}}}{\tau_{\text{eff}}}}-1\right)\right]\,. \tag{2.69}$$

Again, detailed derivation of these formulas can be found in Petrovici et al. (2013) plus the supplementary. With these conversions, LIF sampling becomes possible and LIF-based Boltzmann machines can be created.

2.6 Mixing Problem and Solution in LIF Networks

Weight matrices and biases obtained from training an abstract network with CD, PCD or CAST (Sec. 2.2) can be used for LIF networks. The mixing problem mentioned in Sec. 2.3 also occurs in these networks. One approach to facilitate mixing exploits the STP mechanism (Sec. 2.6.1). The differences between this approach and tempering, which is the base for our approach are discussed in Sec. 2.6.2.

2.6.1 Short-Term Plasticity

The synaptic strength of neurons is determined by the amplitude of the elicited postsynaptic potential (PSP). Short-term plasticity (STP) (Zucker and Regehr, 2002) is a change in the synaptic efficacies on timescales of milliseconds to seconds, in contrast to long-term plasticity (minutes to hours) and structural plasticity (days to years). The efficacy changes are caused by physiological processes, modify the synaptic efficacy only temporarily and are present in many cortical areas. STP encodes the history of the presynaptic activity on time scales that corresponds to many mental processes (motor control, speech recognition, etc.) and therefore it is supposed to contribute to information transmission and processing. If STP leads to the weakening of the synaptic strength it is called short-term depression (STD) and if it leads to facilitating it is called short-term facilitation (STF). A phenomenological STP model in Tsodyks et al. (1998), is the Tsodyks-Makram-Model (TSO) which is described in the following.

$$PSP \propto w \cdot U \cdot R\,, \tag{2.70}$$

$$\frac{\mathrm{d}R}{\mathrm{d}t} = \frac{1-R}{\tau_{\text{rec}}} - \sum_{\text{spikes } s} UR\,\delta\left(t - t_s\right)\,, \tag{2.71}$$

$$\frac{\mathrm{d}U}{\mathrm{d}t} = -\frac{U}{\tau_{\text{fac}}} + \sum_{\text{spikes } s} U_0\left(1-U\right)\delta\left(t - t_s\right)\,, \tag{2.72}$$

where w is the synaptic weight, $U \in [0,1]$ the utilized fraction of available synaptic resources $R \in [0,1]$. Upon arrival of a presynaptic spike at time t_s, the synaptic resources are used by a fraction U from R, which recovers exponentially with the time constant τ_{rec}. Facilitation is modeled by a simultaneous increase in U, followed by an exponential decay with time constant τ_{fac}.

Different parameter combinations $(U_0, \tau_{\text{rec}}, \tau_{\text{fac}})$ correspond to different synaptic mechanisms. *Static synapses* correspond to $(1, 0, 0)$ and are characterized by a first increasing PSP amplitude which then converges to a constant value. This is due to exponential shape of the PSPs, which add up in their effects. *Renewing synapses* with $(1, \tau_{\text{syn}}, 0)$ have a constant amplitude by weakening sequential PSPs originating from the same neuron. This effectively models the saturation of the post-synaptic receptor pools after an incoming spike. *Plastic synapses* weakly modulate the PSP amplitude and correspond to many parameter combinations found in a systematic investigation of the parameter triplet (Leng et al., 2017). They were shown to increase the mixing properties of spike-based LIF networks (ibid.).

2.6.2 Plastic Synapses versus Tempering

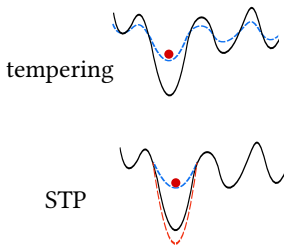


Figure 2.2: STP versus tempering. Tempering modifies the landscape globally in contrast to STP, which affects only the current attractor. Adapted from Leng et al. (2017), Fig. 1D.

Plastic synapses (Sec. 2.6.1) modify the energy landscape locally, i.e., only with respect to the current energy mode, whilst tempering (Sec. 2.3.1) induces a global change of the energy landscape. The reason is that STP solely effects the efferent connections of the active neurons, but tempering changes the spike probability distribution of all neurons by changing the background noise (see Chap. 3) as depicted in Fig. 2.6.2.

Our approach implements tempering in LIF networks with a scheme inspired by neural oscillations (Sec. 2.7). In contrast to plastic synapses, where all samples are valid, in tempering only those states corresponding to the base temperature are considered and the others neglected, which increases the simulation time.

2.7 Neural Oscillations

The spiking activity of individual cortical neurons is usually characterized by Poisson statistics. In contrast to that, neuron groups of different sizes show oscillatory behavior, so called neural oscillations or brain waves. The standard reading in that topic is Buzsáki (2006). A comprehensive review of current research can be found in Draguhn and Buzsáki (2004) on which the information in this section is based if not cited differently. The oscillations are generated and shaped by intrinsic cellular and circuit properties, e.g. time constants. These properties create

frequency bands from 0.05 to 500 Hz. These macroscopic oscillations of many frequency bands are similar to the sinusoidal wave form originating from harmonic oscillators, which inspired our rate varying scheme in Sec. 3.4.1. Combinations of rhythms are supposed to evoke behavioral or cognitive states.

2.7.1 Electroencephalogram

The recording of brain wave patterns is possible since the invention of the electroencephalogram (EEG) in Berger(1929). At the same time, Berger discovered an oscillation of 8-12 Hz and named it alpha wave. This milestone marks the beginning of intensive studies of brain waves. Today, a standard EEG consists of usually 20-256 electrodes with diameters of 0.4 to 1 cm, which are attached uniformly to the scalp plus one or two reference electrodes, e.g. on the ear and a ground electrode e.g. on the nose. Technical details are provided in Nunez and Srinivasan (2007). These electrodes record neural oscillation activities in the brain by detecting and amplifying the electric potential of neurons. The power density of the EEG or the local field potential is inversely proportional to frequency. A majority of the EEG signals have their origin in the cerebral cortex from about 10 million to one billion neurons. Ongoing research is dedicated to the question where exactly in the brain the individual oscillations occur and come from. Despite the occurrence in various parts of the cortex or the hippocampus the oscillations are not limited to these parts as described in Groppe et al. (2013). EEG provides a high enough temporal resolution to reveal further oscillatory patterns mainly during rest and sleep and other unconscious states like anesthesia and epilepsy. The wake state shows rather desynchronized patterns. Today, the availability of other methods than scalp recordings, like invasive techniques enables measurements on smaller scales. These techniques show that the awake state though desynchronized is still rich in rhythms.

2.7.2 Frequency Bands

Due to the locus of the majority of neuronal connections and the limited speed of transmission, the frequency is constrained by the size of the involved neuron subset. High/low frequency oscillations occur especially in smaller/larger subsets, indicating a link between neural oscillations and anatomical structure. Synchrony is crucial in the creation of oscillations and means that a group of neurons fires in the same time window. Frequencies occur in certain bands, called slow 4 (0.025-0.07 Hz), slow 3 (0.07-0.2 Hz), slow 2 (0.2-0.5 Hz), slow 1 (0.5-1.5 Hz), delta (1.5-4 Hz), theta (4-8 Hz), alpha (8-12 Hz), beta (12-30 Hz), gamma (30-80 Hz), fast oscillations (80-200 Hz), ultrafast oscillations (200-600 Hz), depicted in Fig. 2.3. These bands are the foundation of our frequency selection in Sec. 3.4.2.

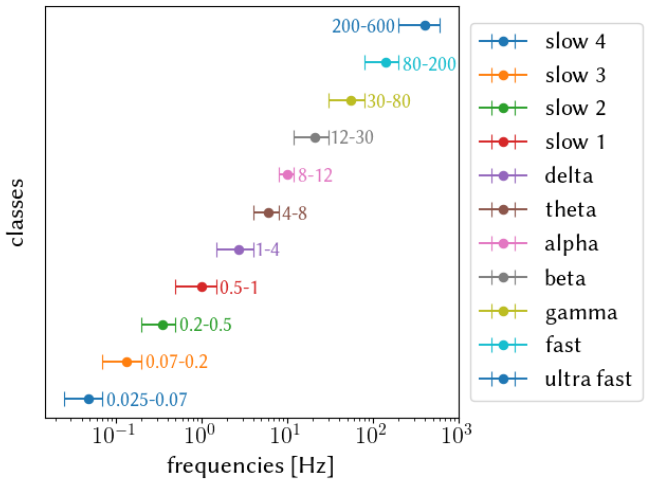


Figure 2.3: Oscillation frequency bands adapted from Fig. 1B in Draguhn and Buzsáki (2004).

2.7.3 Function during Sleep and Wakefulness

Brain waves seem to be functionally relevant in several aspects. Sometimes they fulfill tasks specific to a certain architecture of the underlying neural tissue. Beyond that, some general substrate-independent benefits could be shown. The most interesting of them within the scope of this thesis are consolidation and combination of learned information. Sleep and wakefulness are conditions that humans and animals take on alternately. Sleep is in contrast to wakefulness characterized by unconsciousness, inactivity and limited responsiveness to outer stimuli and can be seen as the default state of the brain since it is disentangled from body and external input. The following information are based on an overview in McCarley and Sinton (2008).

EEG results indicate the succession of different sleep phases, each characterized by the dominance of a certain brain wave. Recordings of eye movements and muscle tone confirm the existence of sleep phases. Wakefulness is characterized mainly by beta and gamma oscillations and closed-eye relaxation by alpha waves. During sleep, higher voltages and smaller frequencies are prevalent. The waves gradually slow down in the course of non-rapid eye movement (NREM) sleep in overall four stages. In stage 1, light sleep immediately after falling sleep, theta waves occur which stay until stage 2. In stage 2, short sleep spindles occur. In stage 3, the transition to deep sleep, delta waves are in the foreground. In stage 4, deep sleep, delta waves make up the majority of the waves. The rapid-eye movement (REM) sleep is named after the occurrence of fast undirected movements of the eye. It is similar to stage 1 in the sense that mostly theta waves are present. The stages 1-4

with final REM sleep are repeated five to seven times per night and the time spent in REM phases increases. The total sleep cycle resembles a damped oscillation. Dreaming mainly happens in the REM phase. There is evidence that during sleep implicit knowledge is transformed to explicit knowledge (Wagner et al., 2004).

2.8 Evaluation

Two properties characterize the performance of the networks in this mixing study: the sampling accuracy, i.e., the proximity of the sampled distribution to the target distribution and the diversity of the generated patterns. For small networks the sampling accuracy can be measured by the Kullback-Leibler divergence (Sec. 2.8.1). A quantitative measure for the homogeneity of the patterns is the indirect sampling likelihood without constraints on the network size (Sec. 2.8.2).

2.8.1 Kullback-Leibler Divergence

The Kullback-Leibler divergence (DKL) measures the discrepancy of two probability distributions. For the discrete probability distribution P with respect to another, Q, the DKL from Q to P is given by:

$$D_{\mathrm{KL}}\left(P||Q\right) = \sum_i P(i) \log\left(\frac{P(i)}{Q(i)}\right) . \tag{2.73}$$

It is only defined if $Q(i) = 0 \Leftrightarrow P(i) = 0 \ \forall i$. If $P(i)$ is zero, the contribution of the ith term is considered to be zero as well, since $\lim_{x\to 0} x\log(x) = 0$. In integral form it reads

$$D_{\mathrm{KL}}(P||Q) = \int_{-\infty}^{\infty} p(x) \log\left(\frac{p(x)}{q(x)}\right) \mathrm{d}x \, . \tag{2.74}$$

One property of the DKL is that it is always positive, which can be shown. The DKL is also called relative entropy since it is composed of the cross entropy $H(P,Q)$ and the entropy $H(P)$:

$$\begin{aligned} D_{\mathrm{KL}}\left(P||Q\right) &= -\sum_x p(x) \log\left(q(x)\right) + \sum_x p(x) \log\left(p(x)\right) \\ &= H\left(P,Q\right) - H(P) \, . \end{aligned} \tag{2.75}$$

2.8.2 Indirect Sampling Likelihood

Indirect Sampling Likelihood (ISL) described in Breuleux et al. (2009) measures the mixing quality of a set of generated samples. ISL estimates the coverage of some test examples, by counting the number of these examples "far" from the generated ones. The algorithm consists of three steps:

1. generation of samples S from a trained generative model
2. use S to train a density model P
3. compute and return the log-likelihood of the test set under P

In particular, P is a non-parametric kernel density estimator. Put differently, P is a mixture model with one component per generated example $\mathbf{x}_i$ and a hyper-parameter β controlling the probability transfer from $\mathbf{x}_i$ to some neighbors. In experiments with d-dimensional binary vectors, each component of the mixture, which corresponds to the kernel probability, can be expressed as a factorized Bernoulli

$$P(\mathbf{y}) = \frac{1}{N} \sum_{i=1}^{N} \prod_{j=1}^{d} \beta^{1_{\mathbf{y}_j = \mathbf{x}_{ij}}} \, (1-\beta)^{1_{\mathbf{y}_j \neq \mathbf{x}_{ij}}} \quad . \tag{2.76}$$

β can have values from 0 to 1 and is chosen to optimize the likelihood of the generated samples. In the case of $\beta = 1$, P corresponds to the empirical distribution associated with the samples $\{\mathbf{x}_i\}$, with decreasing β the distribution is smoothed out until the uniform distribution over the binary vectors is obtained at $\beta = 0.5$. $\mathbf{y}$ are test examples, that neither belong to the training nor to the generated set and N, the number of generated samples. If the curve of $\log(P)$ versus the absolute number of S has a faster increase in comparison to another such curve, it indicates better mixing, since it covers the main modes of the test samples faster.

3 Experiments

The goal of the experimental and theoretical work of this chapter is the implementation of spike-based tempering.

First, the activation functions of an abstract unit under different temperatures adopted in AST are illustrated. Some initial experiments, conducted on the single neuron level, aim to find a temperature analogy for LIF neurons. The logistic activation function can be approximated with LIF neurons with a translation between the abstract and the LIF domain (Sec. 2.5.2). For LIF neurons, the activation functions under Poisson noise of different rates and synaptic weights are recorded and their shape changes explained on the basis of the free membrane potential distribution (Sec. 3.1.2). An analogy to temperature change is achieved by varying the Poisson noise rate and an approximative mathematical mapping between the two quantities is derived (Sec. 3.2). Furthermore, the horizontal shift of the activation function induced by the reset mechanism is compensated (Sec. 3.3). On the network level, we discuss rate variation schemes following sinusoidal patterns inspired from neural oscillations. For small RBMs (for RBM see Sec. 2.1.4 and Fig. 3.1) we select a range of noise rates with similar sampling accuracy which is used for studying the sampling accuracy for different sine waves (Sec. 3.5). Finally, in large RBMs trained on MNIST, the mixing performance under inhomogeneous Poisson noise of the discussed schemes is investigated and compared to networks without mixing facilitation and networks with plastic synapses (for plastic synapse see Sec. 2.6.1) as mixing facilitation (Sec. 3.6). Eventually, the diversity of the patterns measured by ISL is recorded for systematically varied sine wave extrema and period length (Sec. 3.7).

We present the simulation results at the example of current-based neurons. To counteract the deviation between the LIF and the abstract model, induced by additive effects of the PSPs in bursts, the experiments are conducted with renewing synapses implemented by TSO (for TSO see Sec. 2.6.1) - if not mentioned differently. The corresponding neuron parameters (Tab. 6.1) and TSO parameters (Tab. 6.2), follow those proposed for sampling neurons in Petrovici et al. (2016). Some standard software settings for noise, the calibration and the simulations in the experiments are listed in the Appendix (Tabs. 6.4, 6.5 and 6.6). Parameters different from those are explicitly mentioned.

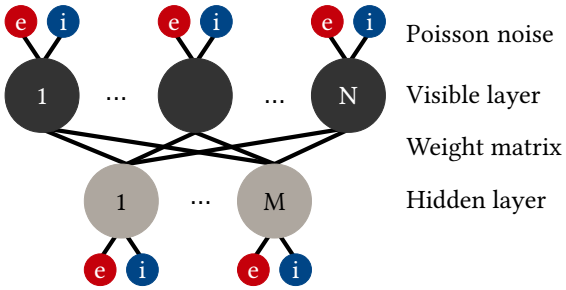


Figure 3.1: Architecture of the LIF-based RBM used here with N visible (black circles) and M hidden neurons (gray circles), each receiving excitatory ("e", red) and inhibitory ("i", blue) Poisson noise input. The weight matrix is symmetric and no intralayer connection exist, leading to the layered structure.

3.1 From Temperatures to Poisson Noise Rates

Before tempering can be realized in LIF networks, a proper mapping between temperatures and noise rates needs to be established. In this section, we show that varying the temperature in abstract neural networks induces a change in the slope of the activation function. In LIF networks, varying either the rate or the weight of the Poisson noise input both affect the membrane potential distribution and thus scale the activation function in a similar manner as in AST. The idea arises to map the variation of one of these two parameters to the AST temperatures. However, continuous changes of synapse strengths are biologically less plausible than background noise modulation. The latter is, for instance, observed in form of periodic oscillations primarily in the cortex, called brain waves (Sec. 2.7).

3.1.1 Activation Functions of Different Temperatures in AST

In AST the inverse temperature β directly enters the calculation of the sigmoid activation function. Hence, β modifies the slope of the abstract activation function: a smaller β multiplicatively decreases the slope. This is demonstrated in Fig. 3.2 at the spiking probability curves evaluating

$$p(z_i = 1) = \frac{1}{1 + \exp\left(-\beta u\right)}\,, \tag{3.1}$$

for different β from 0.1 to 0.9 .

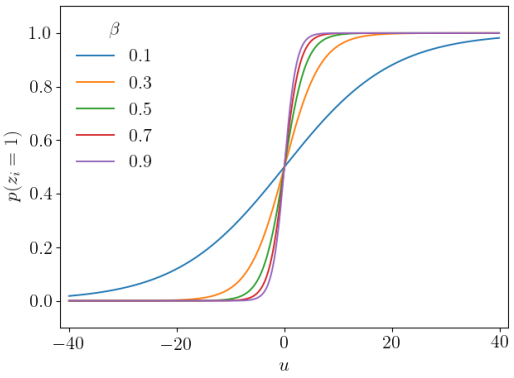


Figure 3.2: AST activation functions of an abstract neuron with different inverse temperature values β: 0.1 (blue), 0.3 (orange), 0.5 (green), 0.7 (red), 0.9 (purple). The spiking probability is plotted over an abstract membrane potential with a range from -40 to 40. With increasing β the slope increases (and vice versa).

3.1.2 Membrane Potential Distribution of Different Noise Rates

High enough Poisson noise input produces a Gaussian shaped free membrane potential distribution. If the input rate is increased, for CUBA neurons the mean of the free membrane potential distribution is unchanged but the variance will increase (Fig. 3.3, left). If the neuron spikes, the reset mechanism distorts the distribution. A comparison of these two different scenarios under varying Poisson noise rate can be seen in Fig. 3.3 (right). The distortion is due to the fact that the membrane potential gets reset once arrived at the threshold. So, the distribution is cut off at the threshold and the values of the missing tail of the Gaussian bell are allocated at the reset potential, visible as the peak. The membrane potential distributions with threshold under different rates are an indication for the shape changes of the activation function, discussed in the subsequent section.

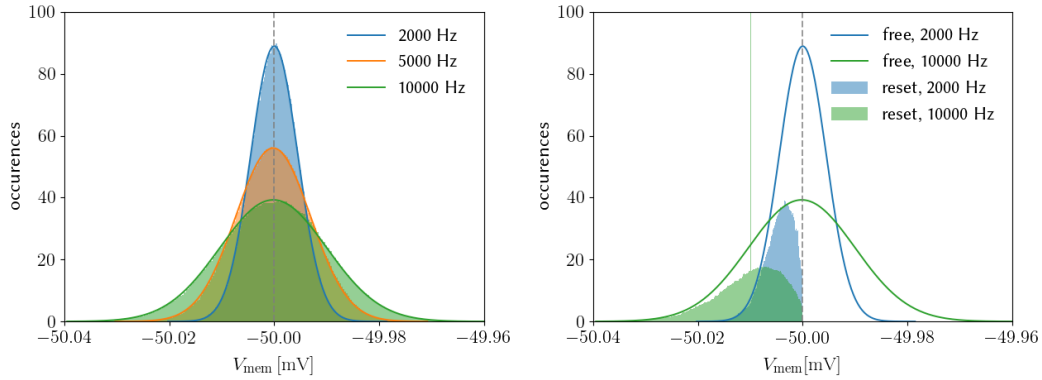


Figure 3.3: *Left*: Histogram of the free membrane potential of an LIF neuron in a simulation for 10 s (transparent colored area) and fitted with a Gaussian (envelope line). The simulation is performed for balanced Poisson noise of 2 (blue), 5 (orange) and 10 kHz (green). The mean lies exactly at the resting potential -50 mV (gray dashed line). With increasing rate the mean stays constant, but the variance increases. *Right:* The free membrane potential distributions (solid line) opposed to the respective distributions with a threshold potential (transparent colored areas) for balanced input of 2 kHz (blue) and 10 kHz (green). The threshold potential is set identical to the resting potential -50 mV (gray dashed line). The vertical line at the reset membrane potential -50.01 mV (green overlays blue) corresponds to the reset states after the spikes. At this reset even more states are accumulated, which lie outside the upper limit of the y-axis for clarity. The cut-off at the threshold plus the accumulation distort the membrane distribution.

3.1.3 Activation Functions of Different Noise Rates

When there is no external input except for the Poisson noise, the activation function of a neuron is the probability to spike dependent on a range of resting potentials. It goes towards 0 if the resting potential is far below the threshold potential, has a steep rise near the threshold and afterwards saturates to 1, following a sigmoid shape. A special symmetry point, we will refer to often, is the potential where the spike probability is exactly 0.5, which we denote as $u_{p0.5}$. The shape can be directly deduced from the membrane distribution for different resting potentials. If the resting potential of the neuron, thus the mean of the distribution, is further below the threshold than one standard deviation, the probability to spike is close to 0. If the resting potential approaches the threshold, the probability increases, since the left tail of the distribution is pushed over the threshold. At the point of identical threshold and resting potential, the spiking probability is 0.5. As soon as it is more than one standard deviation above the threshold, the spiking probability

goes to 1.

In the previous section the broadening of the membrane distribution was related to higher noise input. So, how does a higher input rate influence the activation function? The larger width of the distribution has an effect with two aspects. First, already for smaller resting potentials, the tail is pushed over the threshold, leading to an elevated spiking probability. Second, the normalization of the probability leads to less probability mass in the tails, which decreases the change in probability in dependence of the resting potential. In the activation function this effect manifests in a decreasing slope with higher input. The activation function for a single sampler under noise of different rates as input are recorded and fitted with a sigmoid in Fig. 3.5. The smaller slope corresponds to a higher α fit parameter, see Fig. 3.4 (left). The choices of rate values are explained in Sec. 3.5.1.

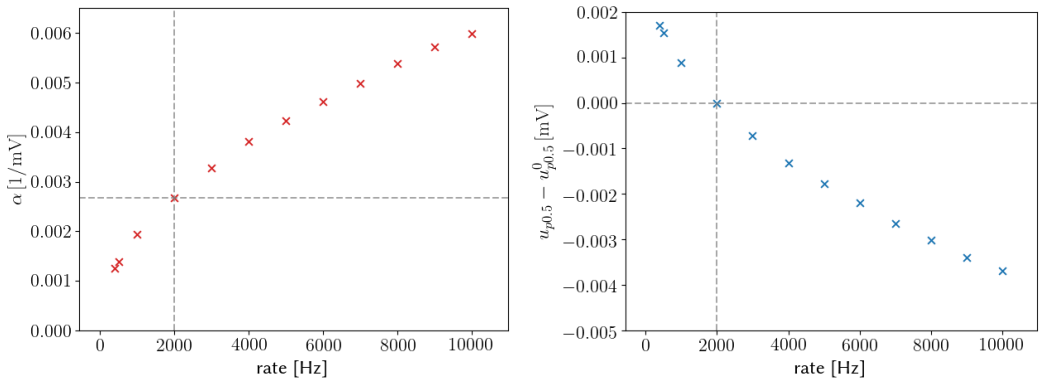


Figure 3.4: Sigmoid fit parameters to the activation functions of an LIF neuron, dependent on the balanced noise rate: 0.4, 0.5, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 kHz. The neuron is calibrated on balanced Poisson noise of 2 kHz (gray dashed lines) with input weight 0.002 nA. *Left:* The slope factor α increases with the rate. *Right*: The shift $u_{p0.5}$ corresponding to the respective rate relative to the $u^0_{p0.5}$ of the reference rate. The difference decreases as the rate increases.

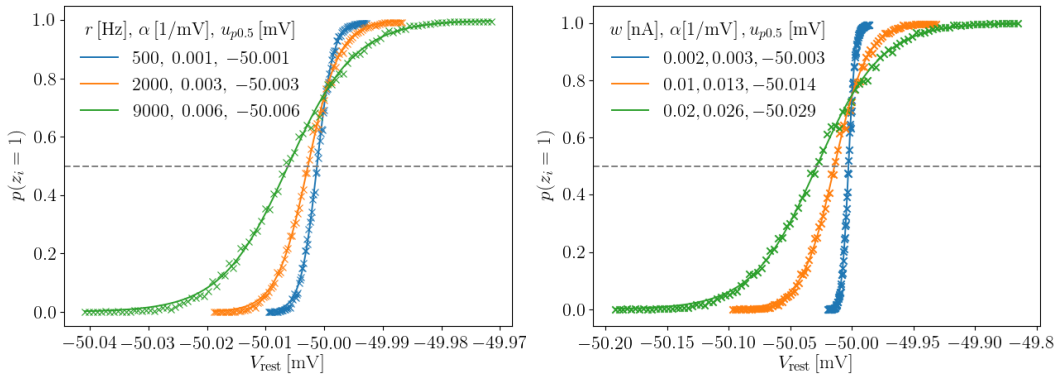


Figure 3.5: Recorded activation functions of an LIF neuron (crosses) and fitted with sigmoid functions (lines). The neuron is calibrated on balanced Poisson noise with rates of 2 kHz and synaptic weights of 0.002 nA. The slope parameter α and the shift parameter $u_{p0.5}$ are given in the legend. The spike probability of 0.5 is marked with gray dashed lines. *Left:* Activation functions under stimulation with balanced noise of 0.5 (blue), 2 (orange), 9 (green) kHz and 0.002 nA synaptic weights. With increasing rate, the slope decreases and the activation function is shifted to the left. *Right*: Activation functions under constant noise rate of 2 kHz and different synaptic weights of 0.002 nA (blue), 0.01 (orange), 0.02 (green) nA. With increasing weight, the slope decreases and the activation function is shifted to the left. Note the different x-axis range.

The reset mechanism introduces a shift of the activation function (Fig. 3.4, right): For higher rates to the left, for smaller to the right, as can be seen in Fig. 3.5 (left). In order to investigate the influence of different synaptic noise weights, we performed the simulations also with weights of several orders of magnitude. This also leads to a rescaling and shifting of the activation function as can be seen in Fig. 3.5 (right). Likewise, the slope gets smaller with increasing weight.

Crossing Point of Activation Functions

We observed that activation functions cross in a particular point at a firing probability of approximately 0.8. At this specific resting potential, the firing probability robustly takes on the same value, independent of the value of the balanced noise rate pair. This phenomenon can be seen in Fig. 3.6. The position of the crossing point will be of importance later, but further investigation is needed to understand how it comes about.

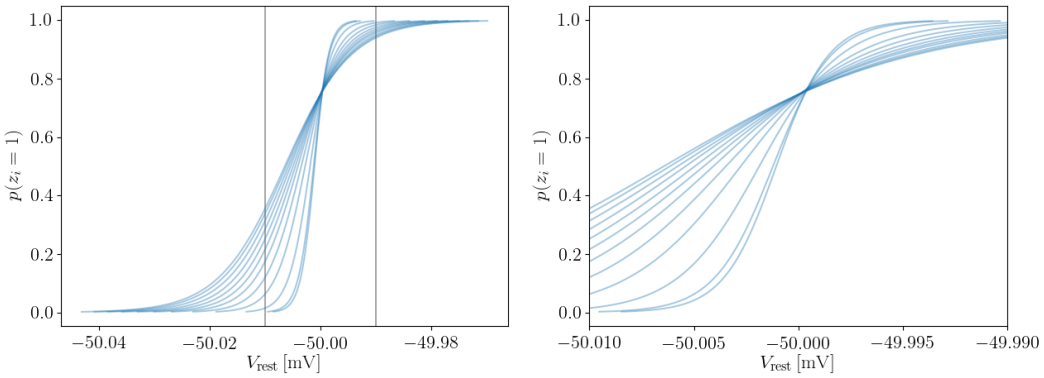


Figure 3.6: Sigmoid functions fitted to activation functions of an LIF neuron, which is calibrated on 2 kHz under different balanced noise rates: 0.4, 0.5, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 kHz. All lines cross at a firing probability of approximately 0.8. A close-up of the area between the gray lines in the left plot is drawn in the right plot.

3.2 Mapping Temperature to Poisson Noise Rate

In the following, we derive the mapping of the AST temperature to the Poisson noise input on a theoretical level. First, the relationship between β and the slope parameter α is demonstrated and then the approximative dependence of α on the rate is derived.

3.2.1 Relationship between Temperature and Alpha

The sigmoid fit to the activation function is given by

$$p(z = 1) = \frac{1}{1 + \exp\left[-\left(u - u_{p0.5}\right)/\alpha\right]} \,. \tag{3.2}$$

A method to eliminate the shift of $u_{p0.5}$ is described in Sec. 3.3.1. So, let us neglect it in the sigmoid function

$$p(z = 1) = \frac{1}{1 + \exp\left(-u/\alpha\right)} \,. \tag{3.3}$$

Comparison with Eqn. 3.1 reveals the equivalence between α and β. In order to establish the ground temperature $\beta_0 = 1$, we relate all slope parameters indexed by n to the one of the reference rate indexed by 0, which yields

$$\beta_n = \frac{1}{T_n} \equiv \frac{\alpha_0}{\alpha_n} \,, \tag{3.4}$$

where T denotes the temperature. This relationship is supported by the observation that the activation function in Fig. 3.2 under higher β_n behaves like an activation function in Fig. 3.5 under smaller noise, i.e., smaller α_n.

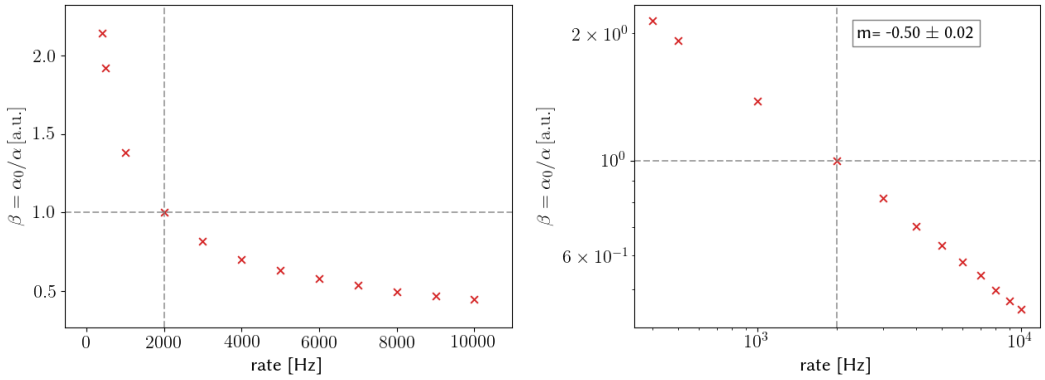


Figure 3.7: Inverse temperature values calculated via. Eqn. 3.4 from the slope factor α of the respective noise rates and the reference rate. The reference rate and $\beta = 1$ are marked with gray dashed lines. *Left:* Display with linear scales on both axes. *Right:* Display with logarithmic scales on both axes.

3.2.2 Relationship between Alpha and Rate

Finally, we derive an approximative formula for the relationship between the slope of the activation function and the noise rate. LIF-based BM dynamics can be approached by two aspects. First, high noise input leads to a Gaussian membrane distribution which has an error function as cumulative distribution function. Second, abstract BMs have a logistic activation function. Matching these two integrals, by equating their derivatives at x=0 leads to the desired relationship between α and the rate. In the following, we provide the detailed calculation.

The probability density function of a Gaussian is given by

$$\mathrm{PDF}\left(x|\mu,\sigma^2\right) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right), \tag{3.5}$$

where x is the random variable, μ the mean and σ^2 the variance. The cumulative distribution function is defined as the integral over the probability density function from minus infinity to x:

$$\mathrm{CDF}\left(x|\mu,\sigma\right) = \frac{1}{2}\left[1 + \mathrm{erf}\left(\frac{x-\mu}{\sigma\sqrt{2}}\right)\right]. \tag{3.6}$$

With the denotation $\tilde{x} = (x - \mu)\,/(\sigma\sqrt{2})$ the error function is given by

$$\operatorname{erf}(\tilde{x}) = \frac{1}{\sqrt{\pi}} \int_{-\tilde{x}}^{\tilde{x}} e^{-t^2} \mathrm{d}t = \frac{2}{\sqrt{\pi}} \int_{0}^{\tilde{x}} e^{-t^2} \mathrm{d}t\,. \tag{3.7}$$

For the use case here, it is valid to consider a centered cumulative distribution function, i.e., $\mu = 0$. The shape of the cumulative density function and the sigmoid activation function

$$\sigma(x) = \frac{1}{1 + e^{-\frac{x}{\alpha}}}\,, \tag{3.8}$$

are very similar and almost parallel in the middle. The slopes at this point can be set approximately equal. They are given by

$$\partial_x \mathrm{CDF}|_{x=0} = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}|_{x=0} = \frac{1}{\sqrt{2\pi}\sigma}\,, \tag{3.9}$$

$$\partial_x \sigma(x)|_{x=0} = \frac{1}{\alpha} \frac{e^{-\frac{x}{\alpha}}}{\left(1 + e^{-\frac{x}{\alpha}}\right)^2}|_{x=0} = \frac{1}{\alpha} \frac{1}{(1+1)^2} = \frac{1}{4\alpha}\,. \tag{3.10}$$

On the one hand, Eqn. 3.9 comprises the variance of the membrane distribution which is dependent on the Poisson rate. On the other hand, Eqn. 3.10 involves the slope of the activation function, i.e., the temperature. So, the 0-th order estimate of the rising sigmoid is given by

$$\frac{1}{\sqrt{2\pi}\sigma} = \frac{1}{4\alpha}\,, \tag{3.11}$$

which can be rearranged to yield the dependence of the slope parameter on the variance

$$\alpha = \frac{1}{4}\sqrt{2\pi}\sigma\,. \tag{3.12}$$

This relation is also used in the coding scheme of sbs as initial value in the calibration. The variance of the free membrane potential for a given rate can be calculated via formulas 2.59 to 2.64. These equations reveal that the standard deviation σ of the free membrane distribution is proportional to the rate r, which yields

$$\alpha \propto \sigma \propto \sqrt{r}\,. \tag{3.13}$$

Therefore, Eqn. 3.12 yields the conversion between α and the rate and we can generate one from the other.

This relationship is also supported by our experiments. The simulated dependence of β on the rate of the Poisson noise is plotted in Fig. 3.7 once with linear

axes scale and once with double logarithmic. The latter line shows a linear dependence which is why we can calculate the underlying power law. Let r denote the rate and m the power in the approach:

$$\frac{\alpha_0}{\alpha} = \beta = \text{const} \cdot r^m \,. \tag{3.14}$$

Taking the logarithm of both sides and solving for m, we can find m as the slope in the double logarithmic plot:

$$m = \frac{\log\left(\frac{\beta_1}{\beta_0}\right)}{\log\left(\frac{r_1}{r_0}\right)} \,, \tag{3.15}$$

which yields a value of approximately -0.5, leading to

$$\beta \sim \frac{1}{\alpha} \sim \frac{1}{\sqrt{r}} \Rightarrow \alpha \sim \sqrt{r} \,. \tag{3.16}$$

This result recovers the theoretical approximation in Eqn. 3.12. The predicted versus simulated results are plotted in Fig. 3.8. They are in good accordance. In summary, this enables the mapping between the temperature and the noise rate.

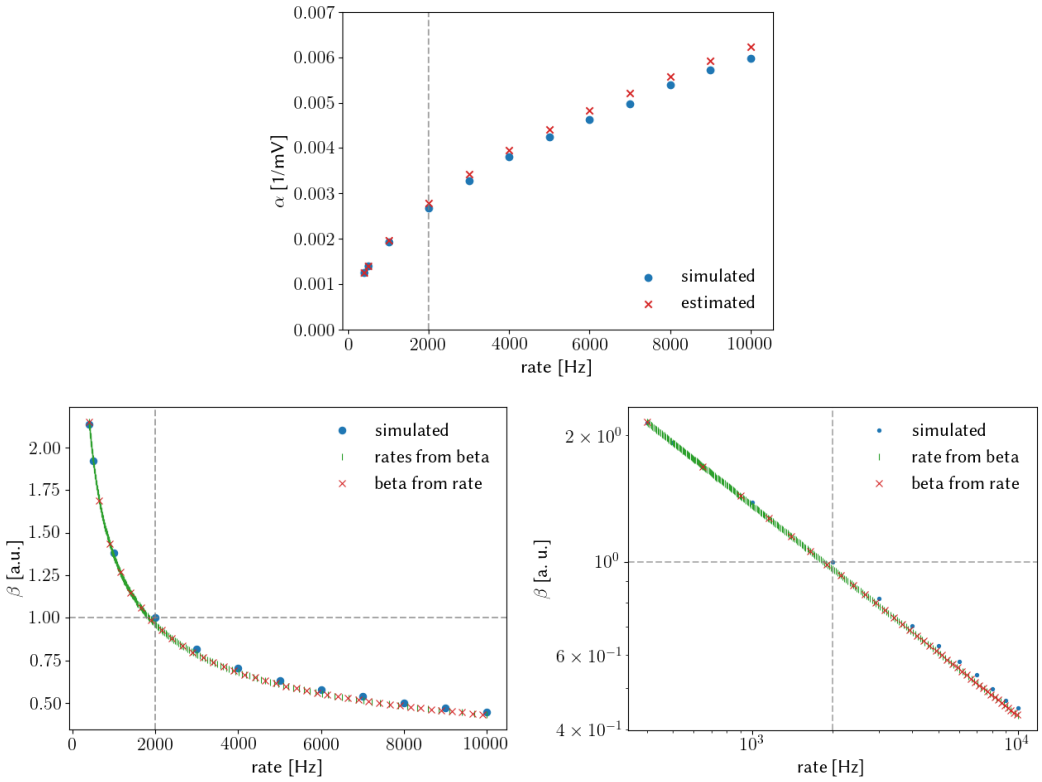


Figure 3.8: *Upper*: Simulated α (blue circles) versus predicted α (red crosses) calculated by the approximation established in Eqn. 3.12. *Lower:* Simulated β values (blue circles) versus approximated predictions of the beta values calculated from the rate range (red crosses) and vice versa (green dashes). They are generally in good accordance. Minor mismatches are due to the approximation. *Lower left:* Display with linear scales on both axes. *Lower right:* Display with logarithmic scales on both axes.

3.3 From Balanced to Shift-Compensated Noise

When comparing the activation function of LIF neurons under increasing noise input (Fig. 3.5) and the one of abstract units with AST (Fig 3.2), one difference is noticeable: The activation functions of the abstract neurons all align with their inflection points at a spiking probability of 0.5. The LIF activation functions have a crossing point (Fig. 3.6) that is at a spiking probability of 0.8 and which deviates from the inflection point. This is due to the shift caused by the reset mechanism. So, in order to create activation functions that only encode the temperature change and enable a proper mapping between temperatures and rates, we have to com-

pensate for this shift. In the following section, we demonstrate how to achieve such a compensation by increasing the rates of the inhibitory Poisson noise.

3.3.1 Shift Compensation Using Inhibitory Noise

Eqn. 2.59 applied to our setup, yields

$$E[u] = E_\mathrm{l} + \frac{I^\mathrm{ext} + w_e \nu_e \tau_e^\mathrm{syn} + w_i \nu_i \tau_i^\mathrm{syn}}{g_\mathrm{l}} \,. \tag{3.17}$$

From this, we can see that the mean membrane potential is determined by the external current input I^ext and the noise input with rate ν, synaptic time constant τ^syn and weight w. Since, the inhibitory weight w_i has a negative sign, the inhibitory noise term cancels the excitatory in the balanced case. Here, we set I^ext to zero and select the inhibitory noise as the means of modulation. We feed an LIF neuron with excitatory noise of the rates from a certain range determined in Sec. 3.5.1. We start from the balanced case, at the benchmark noise rate of 2 kHz, where the shift of $u_{p0.5}$ of other rates are measured from. For other noise rates, we sweep a number of inhibitory rates, below and above the excitatory rate. The resulting shift values for the pairs of excitatory and inhibitory rates are plotted in Fig. 3.9 encoded by color. The blue line corresponds to the balanced case, $r_\mathrm{exc} = r_\mathrm{inh}$. The black circles mark the inhibitory rates that diminish the shift closest to zero. We apply two linear fits to approximate the shifts, one for the inhibitory rates above the reference rate and one for below. As fit functions we obtain for the rates above the reference rate of 2 kHz

$$r_\mathrm{inh}(r_\mathrm{exc}) = 1.04 \cdot r_\mathrm{exc} - 68.72,\ r^2 = 0.99998, \tag{3.18}$$

and below

$$r_\mathrm{inh}(r_\mathrm{exc}) = 1.11 \cdot r_\mathrm{exc} - 215.01,\ r^2 = 0.99983. \tag{3.19}$$

We suspect that the nature of the dependency is not linear, not at least since the intercept of negative rates has no meaning, but for our purposes and regime these approximations are close enough. This linear fits are exploited later on, to determine the shift compensating inhibitory rate corresponding to the chosen excitatory rate. The points with a “red” shift correspond to activation functions that are left from the reference activation function. The points with a ”blue” shift correspond to activation functions that are right from the reference activation function.

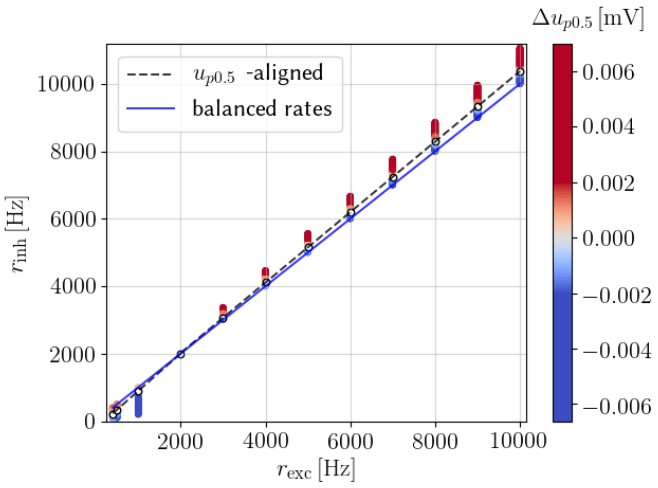


Figure 3.9: Determination of the shift-compensating inhibitory rates. The x-axis corresponds to the excitatory rate and the y-axis to the inhibitory rate. The excitatory rate values are chosen in equal distances between 400 Hz and 10 kHz. A neuron is stimulated by each each noise rate pair and the corresponding $u_{p0.5}$ value of the activation function is plotted encoded by color. The blue line corresponds to the balanced case of equal excitatory and inhibitory rate with shifts plotted in Fig. 3.4. Each black circle denotes the inhibitory rate that reduces the shift closest to 0. The inhibitory rates above the calibration rate are fitted with a linear function and those below with another. The values for the respective fit function slope m, intercept b and quality measure r^2 are for the rates above 2 kHz: m=1.04, b=-68.7 Hz, r^2=0.99998 and below 2 kHz: m=1.11, b=-215.01 Hz, r^2=0.99983.

3.3.2 Shift-Compensated Activation Functions

An example is demonstrated in Fig. 3.10 for three different excitatory rates. The activation function corresponding to the excitatory rate higher than the calibration rate is pushed from the right to the left until their inflection points overlap by applying the appropriate inhibitory and vice versa for the activation function corresponding to a smaller rate. In summary, we are now able to compensate the shift and vary solely the slope of the activation function and hence the “temperature” of the system only.

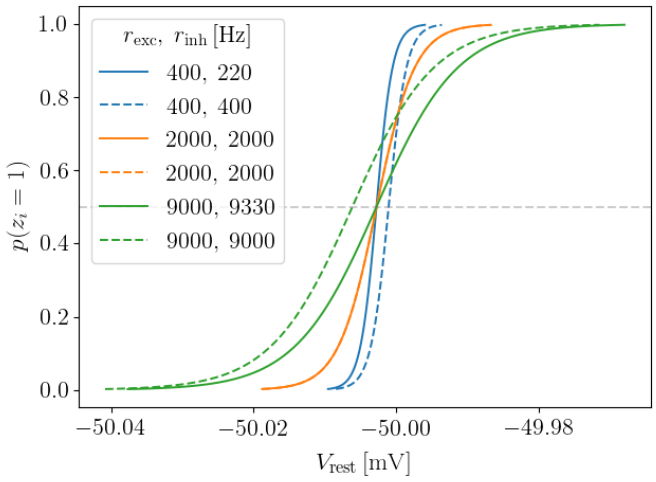


Figure 3.10: Activation functions with shift-compensating inhibitory rates (solid lines) compared to balanced noise input (dashed lines) for different rates: 0.4 (blue), 2 (orange) and 9 kHz (green). The activation function corresponding to the reference rate at 2 kHz, stays unchanged. The other two functions are shifted by adjusting the inhibitory noise rate, until the inflection points overlap at a spike probability of 0.5, marked with a dashed gray line. Thus, we are able to counteract the shift in u_{p05} resulting from an increase and also from a decrease of the Poisson rate.

3.4 Rate Variation Schemes

In this section, the rate variation schemes for later experiments are discussed. The primary usage of sine functions is justified (Sec. 3.4.1) followed by the construction of sine waves with biologically inspired frequencies (Sec. 3.4.2). Moreover, the differences between temperature variation and rate variation are examined and their use cases are (Sec. 3.4.3).

3.4.1 Poisson Rate Following a Sine Function

In AST, the temperature is updated adaptively based on the current state of the network. Since the PyNN-NEST software does not allow for changes during simulation, we fix a variation scheme prior to the simulation. This scheme needs to ensure that not only the reference rate but also higher and lower rates are visited. As the easiest scheme to vary the rates, the reference rate can be alternated with a higher rate forming a square wave. The risk here is that the rate changes are too big and the network cannot adapt properly. To resolve this problem a smoother

transition is needed. Inspired from the wave patterns in neural oscillations, we adopt a sinusoidal rate variation.

A sinusoidal function is given by

$$y = a \cdot \sin\left[b\left(x + c\right)\right] + d\,, \tag{3.20}$$

where d is the shift along the y-axis. a is a scale factor and its absolute value, $|a|$, denotes the amplitude of the sine function. The phase, c, changes the locations of zero values and extreme values. b is the scale factor along the x-axis and encodes the period length via

$$T = \frac{2\pi}{|b|}\,, \tag{3.21}$$

Now we assign to each of these parameters a quantity of the rate sine wave. a corresponds to the amplitude, so we set it to half of the distance between the maximal and minimal rate. It is discussed in depth in later sections, since biological findings require a very short period, but cannot be too short in sampling as the equilibrium state needs to be ensured. d is set to the middle point value between the minimal and maximal value, i.e., the point around which the wave oscillates. Since we are running a long simulation and do not have to relate the sine wave to another signal, we do not shift the curve along the x-axis and set c to zero. In practice, we approximate the sine function with stepwise constant rates. Ideally, the step size converges to zero and the rate is continuously changing. We study in later chapters the effects of different step sizes on the sampling accuracy of the network.

3.4.2 Rate Sine Waves with Parameters from Biology

Synchronized network activity in the form of neural oscillations occurs, in conjunction with different cognitive tasks, in several frequency bands as described in Sec. 2.7. In the following, neural oscillations applied to LIF networks are modeled by sinusoidal rate variations. The inverse of the biological frequencies are reflected in the period of the respective sine wave (Fig. 3.11).

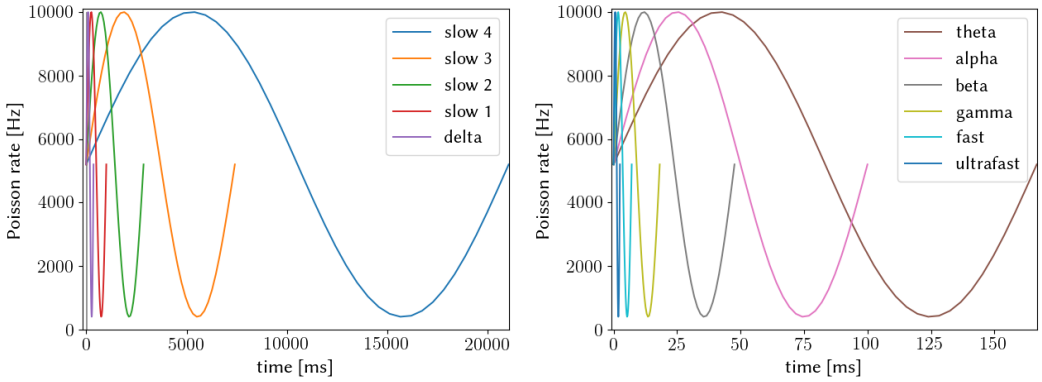


Figure 3.11: Mean values of the neural oscillation ranges (Sec. 2.7.2) transformed to oscillations of Poisson noise rates. The sine maximum is set to 10 kHz and the minimum to 400 Hz. *Left:* Rounded values for low frequencies: slow 4 (0.05 Hz, 21053 ms), slow 3 (0.14 Hz, 7407 ms), slow 2 (0.35 Hz, 2857 ms), slow 1 (1 Hz, 1000 ms) and delta (2.75 Hz, 364 ms). The gray lines indicate the range of the high frequencies. *Right:* Rounded values for high frequencies: theta (6 Hz, 167 ms), alpha (10 Hz, 100 ms), beta (21 Hz, 48 ms), gamma (55 Hz, 18 ms), fast (140 Hz, 7 ms) and ultrafast (400 Hz, 3 ms).

3.4.3 Rate versus Temperature Variation

Until now, the rate was directly varied according to a sinusoidal function (Fig. 3.12). However, in AST, the inverse temperature is varied. Whilst temperature modulation has a rather physical motivation, rate modulation corresponds to biological processes in the brain. With the aid of the established conversion between β and the rate (Sec. 3.2), it is possible to vary directly the inverse temperature according to a sine function and convert the resulting values to rates (Fig. 3.13). Since the conversion follows a power law, the sine wave is distorted.

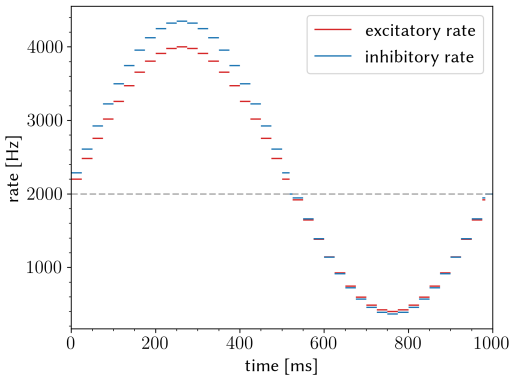


Figure 3.12: Sinusoidal variation of excitatory (red) Poisson noise rate between 0.4 and 4 kHz with a period length of 1s. The corresponding inhibitory rate (blue) compensates for the shift as described in Sec. 3.3.1. It is higher for rates above the reference rate and lower for rates below. At the reference rate of 2 kHz (gray dashed line) inhibitory and excitatory rates overlap. The length of the red and blue dashes, the step size, indicates how long the respective rate is present. In general the step size is 25 ms and for construction reasons sometimes smaller.

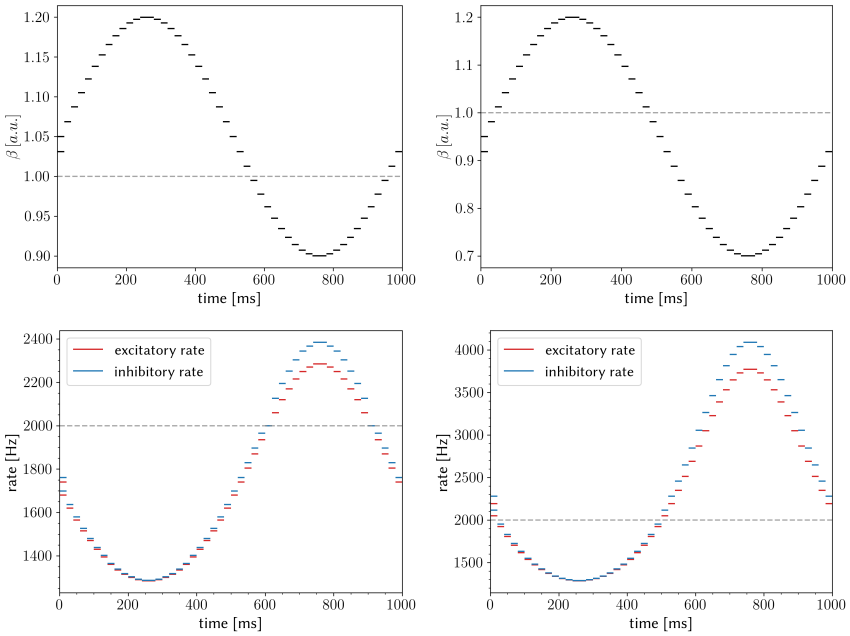


Figure 3.13: *Upper*: Sinusoidal β variation with a sine period of 1s. *Upper Left:* The sine minimum is at 0.9 and the maximum at 1.2. *Upper Right:* The sine minimum is at 0.7 and the maximum at 1.2. *Lower:* Translation of the sinusoidal β variation into excitatory rates (red) and the corresponding shift-compensating inhibitory rates (blue). *Lower Left:* Rate variation corresponding to sinusoidal β variation between 0.9 and 1.2. *Lower Right:* Rate variation corresponding to sinusoidal β variation between 0.7 and 1.2. Due to the non-linear relationship (see Fig. 3.8) between the β and rate, the sine wave is distorted.

3.5 Sampling Accuracy Study

We study the sampling accuracy of small networks with weights and biases following a beta distribution with sinusoidally varied rates (Sec. 3.5.2). As a first prerequisite the Poisson noise rate range with considerations of sampling accuracy is determined (Sec. 3.5.1). Second, the influence of the sine wave discretization on the sampling accuracy is explored (Sec. 3.5.3). For calculating the distributions and DKL in the case of sinusoidal rate variation, we use the conversion from rate to beta described in Sec. 3.2. We use the mapping in the opposite direction in the case of sinusoidal variation of the inverse temperature to obtain the rates corresponding to the temperature values, which we set in the simulation. The conversion is indicated with a right-arrow in Tab. 3.1. In order to measure the quality of the generated samples, the simulated joint distribution over the states is compared with

	sinusoidal β	**sinusoidal** ν
1. **generate**	$\beta_{\text{theo}},\ \beta_{\text{sim}}$	$\nu_{\text{theo}},\ \nu_{\text{sim}}$
2. **convert**	$\beta_{\text{sim}} \rightarrow \nu_{\text{sim}}$	$\nu_{\text{theo}} \rightarrow \beta_{\text{theo}}$
3. calculate numerically theoretical joint distribution (β_{theo})		
4. simulate to obtain simulation joint distribution (ν_{sim})		
5. DKL(simulation joint distribution || theoretical joint distribution)		

Table 3.1: Calculating the DKL between theoretical and simulated joint distribution follows distinct pathways in the beginning for sinusoidal rate and β variation. In the first step, for both quantities, values that follow a specified sine function are created for theoretical calculations and simulation. For simulation, a sine curve of the desired number of periods is generated. Since the simulation requires a rate sine wave and the theoretical distribution requires the inverse temperatures, in both cases the missing quantity is generated. Eventually, the theoretical and simulated joint distributions are retrieved and the DKL evaluated.

the theoretical joint distribution. Since for the theoretical distribution the partition function has to be calculated, this comparison is performed for small networks of five samplers.

Beta Distributed Weights

The weights and biases of the small networks of this section are randomly drawn from a beta distribution. The probability density function of the beta distribution for a random variable x between 0 and 1 is given by

$$\begin{aligned} \mathcal{B}(x;\alpha,\beta) &= \frac{1}{B(\alpha,\beta)} x^{\alpha-1}(1-x)^{\beta-1}\ , \\ B(\alpha,\beta) &= \int_0^1 t^{\alpha-1}(1-t)^{\beta-1}\,\mathrm{d}t\,, \end{aligned} \tag{3.22}$$

where α and β are the non-negative shape parameters. The normalization $B(\alpha,\beta)$ ensures that the integral over the total probability equals 1. The uniform distribution corresponds to an α and β of 1. By changing these parameters, the probability mass of the distribution can either be shifted towards extreme values (at the edge) or central values of the interval. We choose an α and β of 0.5 which pronounces the tails, shift the distribution by $d = 0.5$ and scale it with $a = 1.2$, following the setup in Petrovici et al. (2016) (see Fig. 3.14). These settings ensure dissimilar distributions comprising several orders of magnitude and a linear projection on values in [-0.6, 0.6]. The shifted and rescaled distribution from which we sample

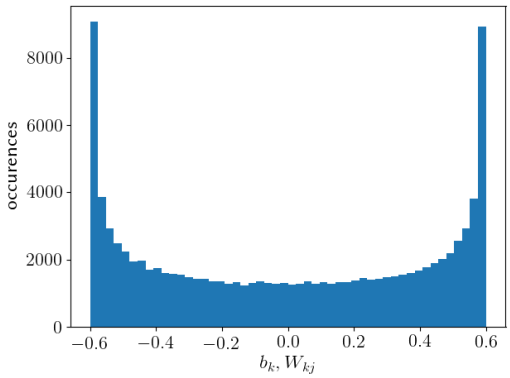


Figure 3.14: Histogram of the beta distribution $1.2 \cdot (\mathcal{B}(0.5, 0.5) - 0.5)$ from which the weights and biases for the 5- and 10-unit networks are sampled. The histogram is retrieved from 10^5 samples and has 50 bins. With these small networks the sampling accuracy is investigated.

the Boltzmann weights and biases reads

$$W_{kj},\, b_k \sim a \cdot (\mathcal{B}(x; \alpha, \beta) - d) = 1.2 \cdot (\mathcal{B}(x; 0.5, 0.5) - 0.5)\ . \tag{3.23}$$

3.5.1 Noise Rate Range

In this subsection, we determine the Poisson noise rate range that covers a β range similar to the one used in Salakhutdinov (2010). We would like to vary the rates to 5 times smaller and 5 times higher values than the reference rate, 2 kHz, hence, from 400 Hz to 10 kHz. We expect that the higher rates facilitate mixing and that the smaller rates stabilize the generated pattern. In AST, the quality of the activation function does not depend on the temperature. However, in spike-based networks, the stochasticity is not intrinsic but rather provided by the amount of noise, thus by the synaptic weight and the noise rate. The theoretical details for this paragraph can be found in Sec. 2.5.2. If the rates become too small, the DKL between the theoretical and the simulated joint distribution worsens. The reason for this is that the neurons are not stochastic anymore and the quality of the diffusion approximation decays. Presumably, the upper boundary is unproblematic, since the high-conductance state and the diffusion approximation improves with increasing noise rate input. However, with the intent to scan possible values for the lower boundary and check the upper boundary, the DKL time course for 10 rates between 10 Hz and 10 kHz is recorded. This rate series also includes the reference rate of 2 kHz, the networks in later experiments are calibrated with. Since

the DKL value can be only retrieved for small networks due to the exploding computation time with increasing network size, we only use network of size 10 here. The statistics is retrieved over 10 different random seed initializations. The result can be seen in Fig. 3.15.

The DKL time course with intrinsically stochastic neurons updated by Gibbs sampling is included for comparison. This theoretically optimal model converges to the Boltzmann distribution for infinite time. Opposed to that, the DKL time course of LIF sampling converges to a higher value, since it only approximates sampling. Rates between 400 Hz and 10 kHz converge to sufficiently close DKL values. Even in light of a recent study where kHz noise is required for the diffusion approximation (Jordan et al., 2017), the whole range is valid for our purposes. So, the 400 Hz marks the lower boundary of the chosen rate range and 10 kHz the upper.

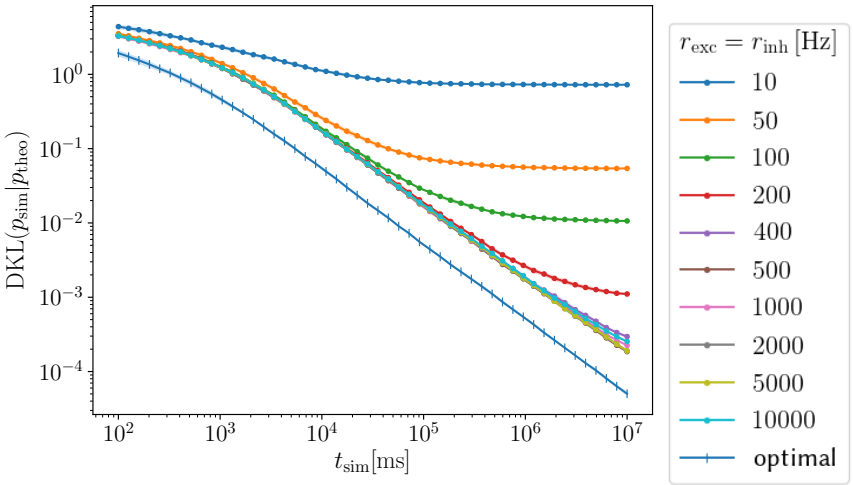


Figure 3.15: DKL time course for a 10-unit LIF network stimulated with 10 rates between 10 Hz and 10 kHz given in the legend. Gibbs sampling is included as a theoretically optimal reference. The lines correspond to the mean. The almost invisible shaded areas correspond to the standard deviation retrieved from 10 different random seed initializations. For rates from 400 Hz to 10 kHz, the converged lines are sufficiently close, which establishes the range for the experiments.

3.5.2 Renewing Synapses under Sinusoidal Noise Input

In order to test the sampling accuracy under sinusoidal noise input, a small network of 5 samplers is stimulated with sinusoidally varying rate. We compare the theoretical with the simulated joint distribution, once averaged over all rates and once specifically at the reference rate in a sequence of sinusoidally varied rates.

The resulting joint distribution averaged over all temperatures and also specifically for $\beta = 1$ is depicted in Fig. 3.16. In both cases, simulated and theoretical results are in good accordance.

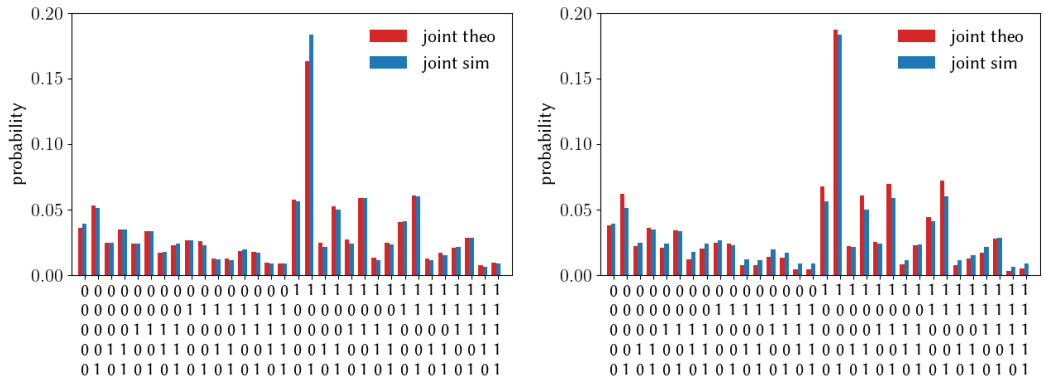


Figure 3.16: Comparison between simulated (blue) and theoretical (red) joint distributions. The joint distributions show the probability of all possible state permutations of the binary states of the 5 samplers. The states are retrieved from a single simulation of a 5-unit LIF-based RBM with renewing synapses, stimulated with sinusoidally varying Poisson noise. The minimal rate is 400 Hz, the maximal rate 10 kHz and the reference rate 2 kHz. 100 sine periods are simulated with equal step size of 10 ms. *Left:* Averaged over all rates in the simulated distribution and all β's in the theoretical distribution. The simulated distribution approximates the theoretical well. *Right:* Specifically for the reference rate corresponding to $\beta = 1$, i.e., 2 kHz. The simulated distribution approximates the theoretical well.

3.5.3 Sine Wave Discretization

Ideally, one would construct a continuous sine wave. In simulation terms "continuously" means changing the rate each simulation step, thus every 0.1 ms. However, the long simulations in Sec. 3.7 with large networks, exceed the available working memory on the cluster nodes already with 58 rate changes for a sine period of 4s. Thus, discretization of the sine wave is necessary for which there exist several possibilities. It is most straightforward to choose either the time or the rate to be linearly spaced, which is illustrated in Fig. 3.17. As a difference between the two, linear spacing of the time leads to an accurate scanning of the valleys, where the rate changes are small, but not on the rising edge with large changes in the rate values. In order to make the scanning accuracy depending on the rate change, linearly spaced rates can be applied. The difference between these two options becomes significant when rate changes are rare.

For small networks, we investigate the influence of the step size on the DKL for a fixed period length, i.e., the number of traversed rates varies between the data points in Fig. 3.18. The variations between the DKL values lie within the standard deviations of the single DKL values. As a conclusion, the step size is not critical for the sampling quality.

These insights help to choose a discretization method in the mixing performance experiments later on. For practical reasons, we linearly space the time. For that, we set a step size at which the difference between the two methods is insignificant, since all step sizes in the investigated range in Fig. 3.18 are equally valid.

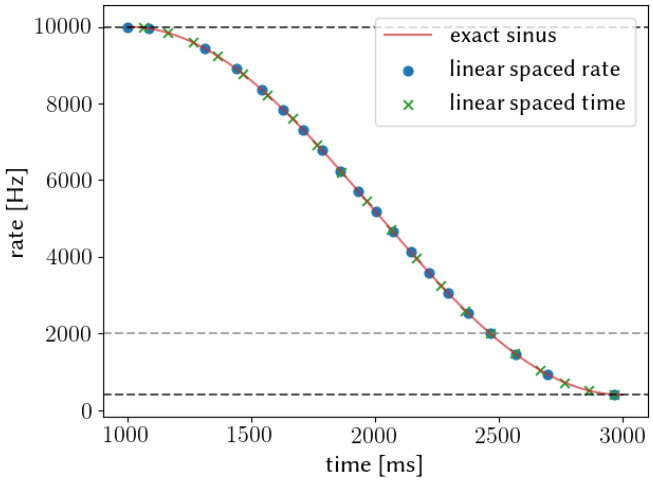


Figure 3.17: The discretization of the exact sine wave can either involve linear spacing of time or the rate. The gray dashed lines mark the reference rate at 2 kHz, which needs to be exactly captured by the method. The minimum rate, here about 400 Hz, and the maximum rate, here about 10 kHz (black dashed lines), are flexible. Linearly spacing the time (green crosses) has the risk that the maximums, where the rate changes are small, the curve is scanned accurately, but the rising edges, where the changes are large, are hardly scanned. This problem is avoided with linearly spaced rates (blue circles), where the time steps are flexible.

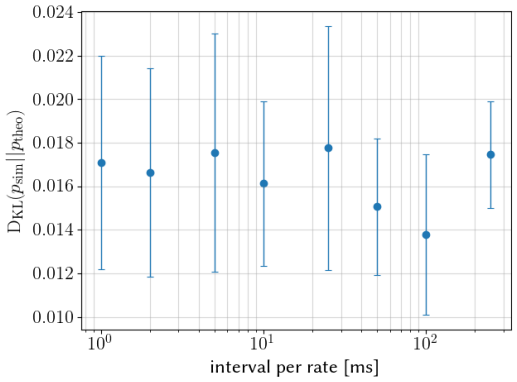


Figure 3.18: We stimulate an LIF RBM with noise of sinusoidally varying rate, which is discretized linearly in time in a 1000 s long simulation. The sine period is 1s, the minimal rate is at 400 Hz and the maximal rate at 10 kHz. In order to study the influence of the time interval between two rate changes on sampling accuracy, the DKL between the final simulated and the theoretical joint distribution is plotted over the step sizes 1, 2, 5, 10, 25, 50, 100 and 250 ms. The standard deviations (bars) are gained from 10 different random seed initializations. The DKL values are distributed in a close range, leading to the conclusion that the step size is not critical for sampling accuracy.

3.6 Mixing Performance in Generation Tasks

In this section, for large networks of 784 visible units, 400 hidden units and a trained weight matrix, the MNIST digit generation task is performed. First, as a reference, no mixing facilitation mechanism is applied (Sec. 3.6.1). Then, as a first mixing benchmark, a network with plastic synapses as mixing facilitation is simulated (Sec. 3.6.2). As a second mixing benchmark, the performance of a network with abstract units and AST as mixing facilitation is presented (Sec. 3.6.3). Finally, the mixing performance of an LIF network with spike-based tempering as mixing facilitation is simulated and compared to the reference and the two benchmark simulations (Sec. 3.6.4). To inspect the quality and diversity of the generated patterns, we plot the firing probabilities of the visible units from the hidden unit activation

$$p(\mathbf{v}) = \frac{1}{1 + \exp\left(-\mathbf{h}\mathbf{W} - \mathbf{b}\right)}, \tag{3.24}$$

where $\mathbf{v}$ and $\mathbf{h}$ denote the vectors of the visible and hidden states, $\mathbf{W}$ the weight matrix and $\mathbf{b}$ the bias vector.

Trained Weight Matrix

The weight matrices of the large RBMs are created from CAST training on 1000 MNIST digits in a network of abstract units, performed by L. Leng [1] (see Sec. 2.3.2) instead of randomly drawn from a beta distribution. Each unit of the visible layer corresponds to one pixel of a 28×28 grid, on which the training digit patterns are clamped.

3.6.1 Reference: No Mixing Facilitation

In order to illustrate the mixing problem, we perform the handwritten digit generation task with a network of standard sampling neurons without a mixing facilitation mechanism, thus with renewing synapses and under homogeneous Poisson noise input. We retrieve the firing probability of the visible layer at evenly chosen points of the simulation time. The first second of the 1000 s simulation is discarded, such that the network dynamics can adapt. The generated patterns of this network for the 100 times of the simulation can be seen in Fig. 3.19. They show recognizable handwritten digits. The network does not mix well as it seems to be stuck in the "0" mode.

[1] For the training, a subset of 1000 images from the MNIST training data set is used. For a total of 200 000 steps, the network with binary units is trained in 100 mini-batch sizes for each step according to the CAST algorithm. The fast chain contains 20 different temperature values between 0.9 and 1. The weight adapting factor, gamma, is chosen as $1 + 90/(100 + t + 5)$, where t is the training step during learning. The learning rate is set to $20/\left(2000 + t\right)$.

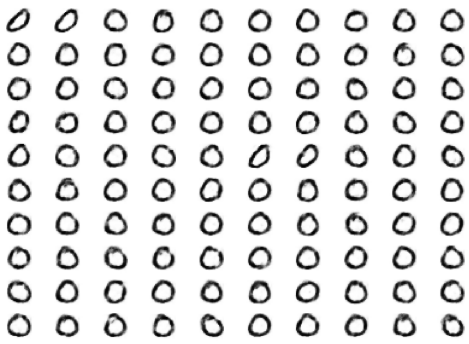


Figure 3.19: Visible layer activation probabilities of an LIF network consisting of 1194 units at 100 times of the simulation of 1000 s in steps of 990 ms after a burn-in of 1s. The RBM has a weight matrix trained on 1000 training samples of the MNIST data set of handwritten digits. The neurons with renewing synapses are stimulated with homogeneous Poisson noise input of 2 kHz. Since the network uses no mechanism which facilitates mixing, it gets stuck in the "0" mode, i.e. it mixes badly.

3.6.2 Mixing Benchmark I: Plastic Synapses

In this subsection, the network stimulated with homogeneous noise is equipped with a mixing facilitating mechanism, namely plastic synapses. Plastic synapses correspond to certain parameter settings of the TSO mechanism (Sec. 2.6.1). Here, we choose $U = 0.1$, $\tau_{\text{rec}} = 70$ ms and $\tau_{\text{fac}} = 0$ ms. Furthermore, the weight matrix is divided by a so called weight division factor of 0.11 to recover the first PSP height. These settings specifically correspond to the RBM used here. They improve mixing as reported in Leng et al. (2017). The generated patterns of this network for 100 evenly chosen times of the simulation can be seen in Fig. 3.20. The images are not as clear as in the simulation with renewing synapses in Fig. 3.19, but a lot more diverse. With plastic synapses, the network is able to escape the local modes and mix well.

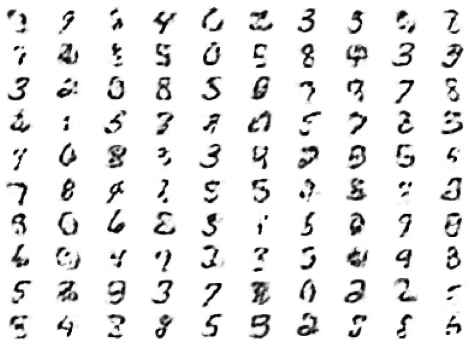


Figure 3.20: Visible layer activation probabilities of an LIF network consisting of 1194 units at 100 evenly chosen times of the simulation, every 990 ms after the first 1s is burned in. The network parameters are trained on the MNIST data set of handwritten digits. The use of plastic synapses allows the network to escape local energy minima faster than with renewing synapses, thus leading to more diverse patterns, i.e., the network mixes better.

3.6.3 Mixing Benchmark II: Abstract Units with AST

The handwritten digit generation task is repeated in an abstract RBM with AST (Sec. 2.3.1) as mixing facilitation. This experiment is described in Salakhutdinov (2010) and serves as a reference in terms of mixing quality in tempered networks. Different from the original publication, we choose 10 different temperatures here, instead of 20, which corresponds to the number of rate changes in later inhomogeneous noise experiments. A larger number of temperatures smooths the transitions at the cost of simulation time, but does not significantly improve the mixing properties. 100 states corresponding to $\beta = 1$ are chosen from the simulation and displayed in Fig. 3.21 (left). More precisely, from the adaptively developed temperature evolution, all β unequal to 1 which facilitate mixing are discarded (Fig. 3.21, right). Moreover, the first 100 $\beta = 1$ are likewise neglected, such that the network dynamics can adapt.

The images are more diverse than those produced with the LIF network with renewing synapses and homogeneous noise input (Fig. 3.19). Occasionally, the abstract network gets stuck in one specific mode belonging to one digit, for instance the six "2" in the fifth row show a large similarity (Fig. 3.21, left). In general, the algorithm mixes well between modes corresponding to different digits.

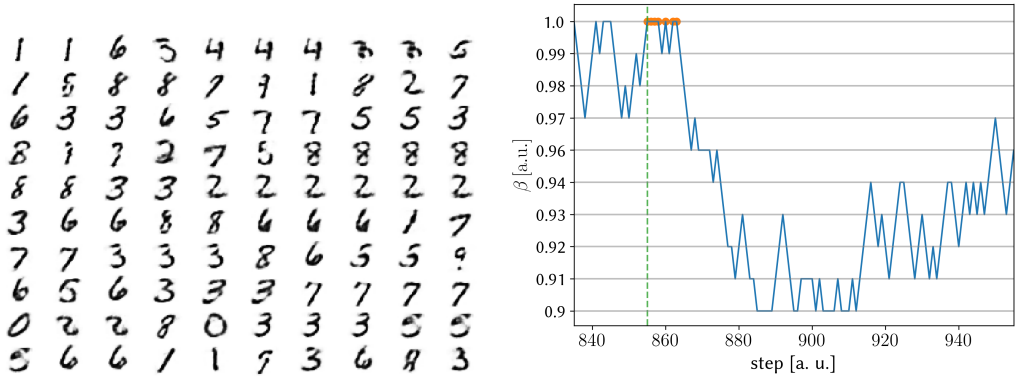


Figure 3.21: *Left:* With AST generated patterns of an RBM with abstract units that is trained on the MNIST training set. *Right:* 10 inverse temperatures 0.9, 0.91, ..., 1 (gray lines) are visited adaptively. Here, only a subset (step 1010-1130) is displayed from in total 100100 steps. The first 100 occurrences of the base temperature are considered as burn-in (green dashed line) and discarded - all further are considered as valid (orange dots). The spiking probabilities plotted on the left, correspond to every 100th occurrence of a valid base temperature.

3.6.4 Renewing Synapses under Sinusoidal Noise Input

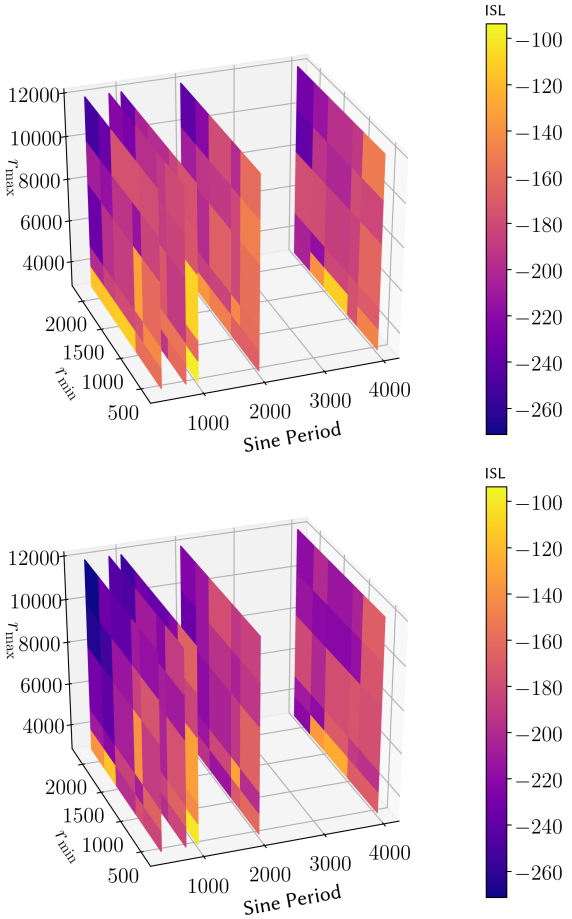
The handwritten digit generation task is repeated with LIF networks under sinusoidal noise input. The wave is characterized by a period length of 4000 ms, a minimum at 1.5 kHz and a maximum at 3 kHz and repeated for 100 periods. In this sine wave the reference rate is not the minimal rate, i.e., it is crossed twice. We read out at the first occurrence. The 100 images obtained at the reference rate are depicted in Fig. 3.22. They are easily recognizable and diverse. Compared to the network output without mixing facilitation (Fig. 3.19), the images are much more diverse. Several modes belonging to one digit are explored as well as modes belonging to different digits. Moreover, the mixing quality is comparable to the benchmark in LIF networks (Fig. 3.20) and the tempering benchmark (Fig. 3.21).

Figure 3.22: 100 consecutive visible layer outputs of an LIF RBM stimulated with sinusoidal noise. The period length is 400 ms, the minimum of the sine wave is at 1.5 kHz and the maximum at 3 kHz. The reference rate is at 2 kHz. Since the minimum of the sine wave and the reference rate are not identical, the reference is crossed twice. The images are read out at the first occurrence of the reference rate. The handwritten digits are clear and the network is able to escape each mode.

3.7 ISL Study

Different neural oscillations are primarily characterized by different sine periods (Sec. 3.4.2). This section aims to find an optimal combination of further constants of the rate sine wave for each fixed period - optimal in the sense of pattern diversity. Alongside the biologically inspired aspect of this experiment, a sinusoidal rate variation instead of a temperature variation is applied (see Sec. 3.4.3). In particular, the minimum, the maximum and implicitly the amplitude of the sine wave are of interest. Thus, for the individual periods, 2D sweeps over different values for sine extrema are performed in the following. The step size is fixed to 10 ms, which entails a varying number of rate changes dependent on the period. As a measure of diversity, the ISL value (Sec. 2.8.2) is evaluated, which returns higher values for higher diversity. The result is shown in Fig. 3.23, where the x-axis corresponds to the period length, the y-axis to the rate minimum and the z-axis to the rate maximum. The ISL value of each triplet is encoded with color. The distinct 4D surface plots correspond to distinct specific points of the sine wave. Since in the underlying sine wave, the reference rate is not the minimum, it appears twice per sine period. This leads to three interesting points in terms of diversity: the first crossing of the reference rate (first 4D plot), the minimum (second 4D plot) and the second crossing of the reference rate (third 4D plot). 1000 generated samples are required for a fully converged ISL value with this network size, as suggested in Leng et al. (2017). Consequently, a reliable ISL evaluation requires the simu-

lation of 1000 periods. For computation time reasons, here only a single run is performed for each parameter set. This method reveals those parameter combinations that create the most diverse images. In the minimum, the ISL values are lower, meaning that the networks mix worse than in the reference rates. Based on this plot, the trend that small maximal rates corresponding to beta values around 0.9 produce better ISL values is visible, which would match with the setup in AST. Nevertheless, more statistics is needed to reveal potential trends in this sweep.



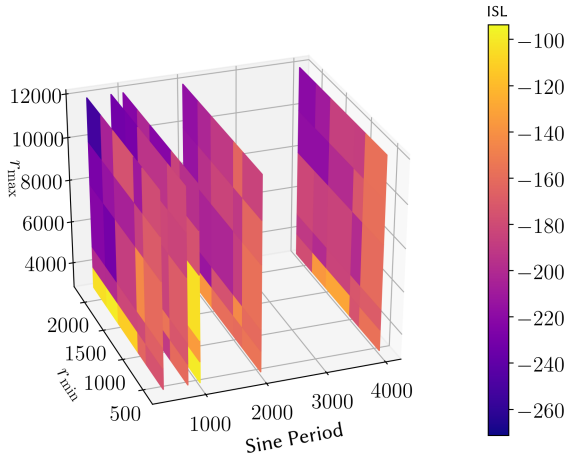


Figure 3.23: 2D sweeps over the sine wave‘s minimal rate in Hz (y-axis) and maximal rate in Hz (z-axis) for several period lengths in ms (x-axis). The minimal rate values are 400, 800, 1000, 1500 and 1800 Hz. The maximal rate values are 3000, 4000, 5000, 8000 and 10000 Hz. These values lie at the center of the squares. The period length values are 400, 800, 1000, 2000 and 4000 ms. The ISL value, as a measure of the diversity of the images, is represented by the color-coding. In the underlying sine wave the minimal rate is not identical to the reference rate. Thus, the ISL values are evaluated from states corresponding to three points of the sine wave: first crossing of the reference rate (*first plot*), the minimum (*second plot*) and the second crossing of the reference rate (*third plot*). For computation time reasons for each square a single simulation is run, but not with several random seeds. The ISL values in the minimum are in general lower than in the two reference rates.

4 Software

First, this chapter provides an overview over the software used within this thesis to conduct the experiments in Chap. 3. Aspects relevant for the subsequently described contributions are specifically highlighted. The utilized framework includes the open-source softwares Neural Simulation Tool (NEST) (Sec. 4.1) developed by the NEST-initiative and the interface PyNN (Sec. 4.2). Furthermore, a NEST module allowing for time-varying Poisson noise input (Sec. 4.3) and the library spike-based-sampling (sbs) (Sec. 4.4) are described, both developed in this group. In this thesis, sbs is extended by a class that integrates the inhomogeneous Poisson noise generator (Sec. 4.5).

4.1 Neural Simulation Tool

The Neural Simulation Tool (NEST) enables the user to simulate networks consisting of point like neurons or neurons with few compartments introduced in Gewaltig and Diesmann (2007). The simulator is specialized for large networks. The fields of application are dynamics and network structure studies. One main building block of NEST are nodes that transmit events via connections and can be connected to networks. These instances also form the main objects in the NEST code.

Nodes are either neurons, devices or smaller networks. A variety of models for neurons, devices, synapses or events are available. Neuron models range from simple leaky integrate-and-fire to Hodgkin-Huxley type neurons. As synapses, for instance, in the experiments current- and conductance-based synapse models are utilized. Device examples are the Poisson spike train generator as stimulus device as well instruments to record the membrane potential or to perform the simulation.

Connections can be set up in various high-level patterns. They take a sending and a receiving node, a weight, i.e., the strength of the stimulus, and a delay, i.e., an artificial travel time of the neurological signal. Connections forward an event to the synaptic input of the neurons according to a selectable synaptic kernel. Events are either spikes or counts marked by the time-stamp of their generation and the weight of the connection. The arrival time at the target neuron is determined by this time stamp plus the delay of the connection. Handle functions for the events are implemented in the neuron models.

The original simulation language of NEST is SLI, a high-level scripting language. Underneath, the core code of the simulator is written in C++ . The interpreted programming language Python has several advantages over SLI: it is easier to realize simulations, stimuli and evaluate the results. Furthermore, SLI is stack-based. PyNEST (Eppler et al., 2009) is the corresponding wrapping-layer to access NEST from Python alongside PyNN (discussed below), which also uses PyNEST (also see Fig. 4.1). This is the user interface, we use.

The modular structure of NEST enables the extension with own modules. In the course of this work the parts of sbs that are needed for the experiments are updated to NEST 2.12.0, a more recent version which came along with profound technical changes in the synapse architecture, described in Kunkel et al. (2014).

4.2 PyNN

PyNN is a Python based programming interface to various simulators of spike-based neural networks, described in Davison et al. (2009). The supported simulators are NEURON (Hines and Carnevale, 1997), NEST, PCSIM (Pecevski et al., 2009), Brian (Goodman and Brette, 2008), Neuroml (Crook et al., 2005) and the Heidelberg neuromorphic hardware Schemmel et al. (2010) (Fig. 4.1). It allows to mix PyNN and native simulator code and only supports features that are provided by at least two simulators. It consists of a high-level object oriented interface plus a low-level procedural interface. Neurons are organized in Populations which are connected via Projections. Here, we use PyNN 0.8.3.

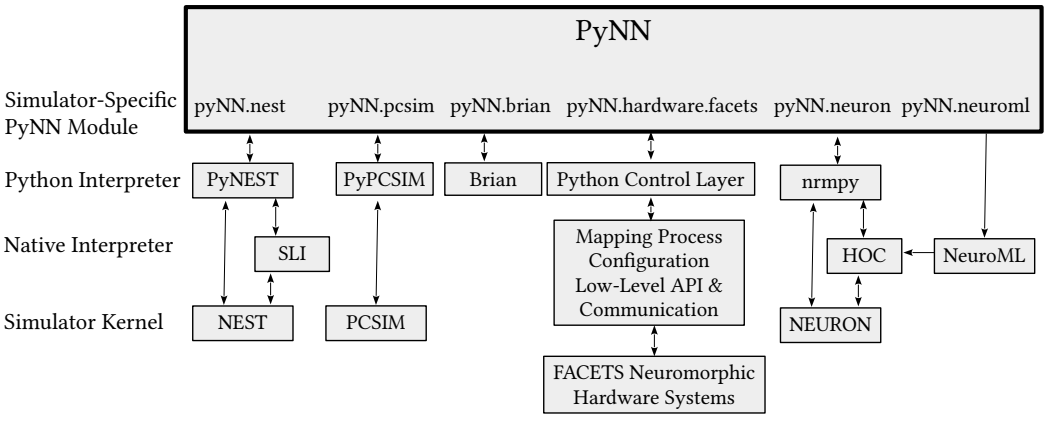


Figure 4.1: Simulators with PyNN support. Similar structures as demonstrated for NEST exist also for the other simulators, which are not used here and thus not closer described. Adapted from Fig. 7 in Brüderle et al (2011).

4.3 Inhomogeneous Poisson Noise Generator

NEST provides a C++ based Poisson noise generator, poisson_generator. This is a static noise source, i.e., the rates and weights are constant over the simulation and cannot be changed without stopping the simulation. It functions by drawing a sample from a Poisson distribution of the specific rate for each time step. There exists one other Poisson noise generator, called sinusoidal_poisson_generator, which creates sinusoidally changing rates. It allows to set parameters of the sine function. However, this generator has amongst others the disadvantage that custom rate variation schemes deviating from a sine function are not possible, which are e.g. needed to create the rate changes calculated from sinusoidal beta variations (Sec. 3.4.3).

Due to the need of time-varying Poisson noise sources in learning experiments, several new NEST modules for this necessity are introduced in Breitwieser (2015). For our use case, we focus on the poisson_generator_var_rate_multi. Here, for each time step, the total number of spikes is drawn and distributed according to a multinomial distribution. As input, it requires three arrays: rates, times and indices. The k-th entry in the time array marks the onset of a Poisson noise stimulation with the k-th rate, sent to the target corresponding to the k-th index. In order to increase the efficiency of the noise generator, the number of targets need to be specified via num_outputs, such that the generator can already allocate the required sizes for the internal arrays.

This generator enables maximal flexibility in creating rate variation schemes. To make it usable for this work, this model is wrapped into a NEST module and made available in sbs via a source configuration (for the details see 4.5).

4.4 Spike-Based Sampling

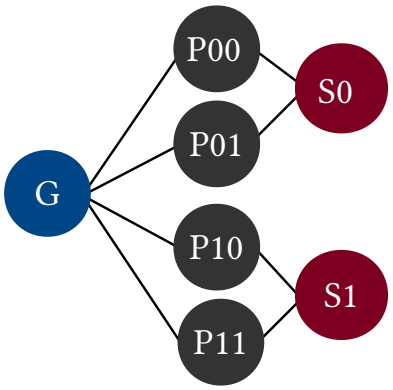
Spike-based sampling (sbs) is a library enabling fast spike-based inference simulations. It is based on PyNN with several performance enhancing specializations for NEST introduced in Breitwieser (2015). It provides two main classes. First, LIFsampler enables the setup of the LIF neurons with user defined specifications, like neuron parameters, as well as the calibration. Second, BoltzmannMachine connects the samplers to networks that sample from Boltzmann distributions. In the course of this thesis an extension to the source configuration is implemented and described in the following.

4.5 Source Configuration

The SourceConfiguration class takes care of the setup of sources in sbs and separates the noise sources from the sampling network. One of the subclasses is PoissonSourceConfiguration which sets up the standard Poisson noise generator poisson_generator (Sec. 4.3) or a more efficient version thereof if specified by the user. We implement a further subclass that sets up the connections between the time-varying Poisson noise generator poisson_generator_var_rate_multi (Sec. 4.3) and the network. The purpose of this subclass, named MultiPoissonVarRateSourceConfiguration, is to pre-process the settings of the user for the generator.

For each Poisson source, potentially several per sampler, the framework creates a parrot neuron. A parrot neuron is a model that re-emits all the received spikes. The Poisson noise generator is connected to the series of intermediate parrot neurons. In addition, the subclass derives the appropriate indices array for the multi_poisson_generator_var_rate from the input matrix. It specifies for each time which target parrot receives which spike train from the unique spike train of the generator. In a last step, the parrot neurons are connected to the samplers (Fig. 4.2).

The class takes a matrix of the source course with all the Poisson sources for each individual sampler. In this, each sampler can have an individual number of Poisson sources specified by a three column matrix. The first column defines the time points at which the rate changes occur, the second column the desired rates, the third column takes the weight (see code snippet in Fig. 4.2). As a constraint, per Poisson source only one weight is supported and with that one synapse type, determined by the sign of the weight entries. But, it leaves the freedom of future implementation of changing weights. Both synapse models, COBA and CUBA, can be utilized within this framework. So, the supplied matrix has as dimensions: number of samplers times number of Poisson sources which might vary from sampler to sampler times the number of changes times the three entries time, rate and weight.




```
import sbs.db as db
import numpy as np

poisson_source_1 = np.array([
    # [ <time>,<rate>,<weight>],
    [ 0., 1000., 0.002 ],
    [ 2000., 100., 0.002 ],
    [ 3000., 4000., 0.002 ]])

poisson_source_2 = np.array([
    # [ <time>,<rate>,<weight>],
    [ 0., 1000., -0.002 ],
    [ 2000., 100., -0.002 ],
    [ 5000., 300., -0.002 ]])

src_courses = np.array(
    [poisson_source_1, poisson_source_2, ])

sampler_config = db.SamplerConfiguration.load(
    "calibration.json")

sampler_config.source_config = \
    db.MultiPoissonVarRateSourceConfiguration(
    src_courses=src_courses)
```

Figure 4.2: *Left*: Projection scheme from the Poisson noise spike train generator ("G") to parrots ("P") and further to samplers ("S"). In this example, each sampler receives input from two parrot neurons, for instance, from an excitatory and an inhibitory one. *Right:* Example program code to set up a time varying Poisson source. For an excitatory and an inhibitory Poisson source (poisson_source_1, poisson_source_2), matrices with a times, rates and a weights column are defined. The two sources are pooled into one array (source_courses) and later passed to the generator. As usual, the configuration of the samplers (sampler_config) is loaded back from calibration. In the last step, the source configuration of the sampler configuration is replaced by the time varying Poisson source configuration (MultiPoissonSourceConfiguration).

5 Discussion

In this thesis, we constructed a spike-based tempering framework using inhomogeneous Poisson noise input. The study begins with investigations on the single neuron level, in which for various balanced rates, i.e., excitatory equaling the inhibitory noise rate, the activation function is recorded. Two parameters are retrieved: the slope factor α and the inflection point $u_{p0.5}$ which result from fitting a sigmoid function. The same experiment is repeated for synaptic noise weights. Compared to the activation function under the reference rate, the one the neuron is calibrated on, an increase in the rates or the weights decreases the slope and induces a negative shift. A decrease in the quantities has the opposite effects on the parameters. These observations are explained by the influence of the noise amount on the membrane distribution. We find the slope to encode temperature in the system and introduce the mapping

$$\beta = \frac{1}{T} \equiv \frac{\alpha_0}{\alpha}\,,$$

where β is the inverse of temperature T, α the slope factor of the considered activation function and α_0 the slope factor of the reference activation function. Afterwards, an approximative mathematical relationship between α and the variance of the membrane potential distribution is derived (Sec. 3.2.1). The variance depends on the rate, so, the conversion between inverse temperature and the noise rate ν is established which reveals the dependency

$$\beta \propto \frac{1}{\sqrt{\nu}}\,.$$

This relationship is supported by experimental findings (Sec. 3.2.2). Whilst the slope is an indication for temperature, the shift is an artifact deriving from the reset mechanism and hinders a clean temperature variation. Thus, we establish a technique to counteract this shift by applying an inhibitory rate that is smaller/higher than the excitatory rate for the rates below/above the reference rate. We find the shift-compensating inhibitory rates dependent on the excitatory rates. For this study it is sufficient to approximate the dependency with two linear fits.

Next, we create a sinusoidal variation scheme for the rates. On the technical side, we embed an existing C++ based time-varying Poisson noise generator into

the spike-based sampling (sbs) framework by implementing a Python based class that takes care of the configuration of these specific sources in advance. The user provides a matrix with the desired time points of the rate changes, the rate values and weights to characterize the Poisson sources for each sampler. In our case, each sampler receives an excitatory and an inhibitory input with simultaneous rate changes. This endeavor goes along with an update of those parts of sbs essential for the experiments to a newer NEST version.

In order to examine the sampling accuracy, we employ this framework to stimulate an RBM of five units with sinusoidally varying rates or inverse temperatures. From this, a comparison between the theoretical and the simulated joint distribution is retrieved. The calculation of the theoretical distribution requires β's, whereby the simulation requires rates, so we make frequent use of the introduced mapping. First, the simulated joint distribution of the whole simulation is contrasted to the theoretical distribution being an average over those corresponding to the distinct betas. Second, the simulated joint distribution over the states when the reference rate is present, is compared with the theoretical distribution calculated with the base temperature at a β of 1. In both cases, the distributions are in good accordance, meaning that the chosen sine wave properties, most crucially the period length and the step size between two rate updates, do not hinder the network to sample from the correct target distribution. We fix the boundaries of possible values of the sine wave to 400 Hz and 10 kHz. The corresponding considerations are described in Sec. 3.5.1. We set up a series of sine waves by systematically varying its properties in this range, i.e., different combinations of the minimal and maximal values for distinct period lengths from 0.4 ms to 4 s which is part of the neural oscillations regime. A larger RBM with 784 visible and 400 hidden units and a weight matrix trained on MNIST handwritten digits is successively run under these constructed sinusoidal rate inputs. In each experiment, the ISL value, a measure for the diversity of the generated images, is evaluated in 1000 states, where the reference rate is present (Fig. 3.23). With sine wave settings which are amongst those that produced the best, i.e., the highest ISL values, images are generated. We show that the generated patterns are as readable as under homogeneous noise input, meaning that the network samples from the correct target distribution. More importantly, the images show a diversity comparable to those achieved with mixing facilitating mechanism like plastic synapses or AST for abstract units. In conclusion, for a series of sine wave period lengths, pairs of minimal and maximal rate are found that escape the repetitive pattern generation problem and perform similar to established mixing facilitating mechanisms in terms of pattern diversity.

6 Outlook

As an immediate improvement of the search for sine wave properties generating the most diverse images, the ISL sweep could be performed with more statistics, the sweep values refined and lower sine frequencies visited, all at the cost of time. The implementation of the high-frequency domain of neural oscillations is more challenging, since the sine periods are of the same order as other time constants of the neurons, which requires considerations about the sampling accuracy. But, it promises the advantage to take less time for the collection of a certain number of valid states, which would avoid long simulation time.

As one possibility to improve the visualization of the high-dimensional data set of the generated images, t-Distributed Stochastic Neighbor embedding (t-SNE) developed by Van Der Maaten and Hinton (2008) can be used, which helps to reduce the dimensionality. Besides that, ISL needs to be complemented with a quantity measuring the quality of the images which is challenging in large networks due to the exploding computation time to calculate the target distribution. Another goal is the application of inhomogeneous Poisson noise in learning, for instance in the form of coupled spike-based tempering for LIF networks, where the time-varying Poisson noise plays the role of the fast, mixing facilitating chain as AST in CAST.

In this work, the amplitude and the period length of the sine wave were held constant for the whole simulation. The flexibility of the software framework features the potential to implement more advanced and biologically relevant patterns in the future. It is imaginable to construct a wave pattern with a decreasing amplitude, resembling a damped oscillator. This is established in simulated tempering methods and would possibly help the exploration of the energy landscape. Moreover, in the brain, frequencies do not occur in an isolated fashion but rather in superposition. Depending on the state of the brain, like wake or sleep, certain frequencies become dominant in the frequency spectrum. Conceivably, a scheme can be constructed that mimics this coexistence of several frequencies or forms a series of different oscillations. The latter can be concretely inspired by the frequency succession during sleep that also resembles a damped oscillation.

Another continuative research aspect is the combination of the two mixing facilitation methods for LIF networks, namely spike-based tempering and plastic synapses. As a yet to implement third method, neural adaptation based on adaptive exponential integrate-and-fire (AdEx) neurons might also enhance mixing. The integration of different mixing facilitating mechanisms is the far goal of this study.

Appendix

Parameters

neuron		
membrane capacitance	cm	0.2 nF
excitatory reversal potential	e_rev_E	0 mV
inhibitory reversal potential	e_rev_I	-100 mV
offset current	i_offset	0.0 nA
membrane time constant	tau_m	0.1 ms
refractory time constant	tau_refrac	10 ms
excitatory synaptic time constant	tau_syn_E	10 ms
inhibitory synaptic time constant	tau_syn_I	10 ms
reset potential	v_reset	-50.01 mV
rest potential	v_rest	-50 mV
threshold potential	v_thresh	-50 mV

Table 6.1: Default neuron parameters for all experiments if not explicitly changed.

synapse		
PyNN synapse model	synapse_model	IF_cond_exp, IF_curr_exp
TSO model	proper_tso	true
enable renewing synapses	saturating_synapses	true

Table 6.2: Default synapse parameters in sbs for all experiments if not explicitly changed.

TSO		renewing	plastic
utilization	U	1	0.1
recovery time constant	tau_rec	10 ms	70 ms
facilitation time constant	tau_fac	0 ms	0 ms
weight division factor	wdf	1	0.11

Table 6.3: Default TSO parameters for all experiments if not explicitly changed.

noise	
weights	-0.002, 0.002 nA/μS
rates	variable, reference at 2000 Hz

Table 6.4: Default noise parameters for all experiments if not explicitly changed.

calibration		
burn-in time at start	burn_in_time	500 ms
time step	dt	0.001 ms
calibration time	duration	1e4 ms
number of samples	num_samples	100
range of bias	bias_range	-6 to 6
grid-constrained time scheme	spike_precision	on_grid

Table 6.5: Default calibration parameters for all experiments if not explicitly changed.

simulation		
neuron simulator	sim_name	pyNN.nest
burn-in time at start	burn_in_time	500 ms
time step	dt	0.1 ms
simulation time	duration	1e6 ms
grid-constrained time scheme	spike_precision	on_grid

Table 6.6: Default simulation parameters for all experiments if not explicitly changed.

AST		
min-batch size		1
inverse temperature	β	0.9...1.
number of temperatures		3
valid samples		100000
adaptive weight factor	γ	$\frac{90}{100+t+5} + 1$
learning rate		$\frac{20}{2000+t}$

Table 6.7: Default AST parameters for all experiments if not explicitly changed.

Nomenclature

AST	adaptive simulated tempering
BM	Boltzmann machine
CAST	coupled adaptive simulated tempering
CD	contrastive divergence
COBA	conductance-based
CUBA	current-based
DKL	Kullback-Leibler divergence
EEG	electroencephalogram
GS	Gibbs sampling
ISL	indirect sampling likelihood
LIF	leaky integrate-and-fire
MCMC	Markov chain Monte Carlo
NEST	Neural Simulation Tool
PCD	persistent contrastive divergence
PSP	post-synaptic potential
RBM	restricted Boltzmann machine
sbs	spike-based sampling
ST	simulated tempering
STP	short-term plasticity
WL	Wang-Landau

Bibliography

Hans Berger. Über das Elektrenkephalogramm des Menschen. *Arch. Psychiatr. Nervenkr.*, 1929.

Oliver Julien Breitwieser. Towards a Neuromorphic Implementation of Spike-Based Expectation Maximization. *Master thesis*, 2015.

Olivier Breuleux, Yoshua Bengio, and Pascal Vincent. Unlearning for Better Mixing. *Technical Report*, 2009.

Lars Buesing, Johannes Bill, Bernhard Nessler, and Wolfgang Maass. Neural dynamics as sampling: a model for stochastic computation in recurrent networks of spiking neurons. *PLoS Computantioal Biology*, 2011.

György Buzsáki. Rhythms of the Brain. *Oxford University Press, U.S.A.*, 2006.

Sharon Crook, David Beeman, Padraig Gleeson, and Fred Howell. XML for Model Specification in Neuroscience. *brains, minds, media*, 2005.

Andrew P. Davison, Daniel Brüderle, Jochen Martin Eppler, Jens Kremkow, Eilif Muller, Dejan Pecevski, Laurent Perrinet, and Pierre Yger. PyNN: a common interface for neuronal network simulators. *Frontiers in Neuroinformatics*, 2009.

Andreas Draguhn and György Buzsáki. Neuronal oscillations in cortical networks. *Science*, 2004.

Jochen Martin Eppler, Moritz Helias, Eilif Muller, Markus Diesmann, and Marc-Oliver Gewaltig. PyNEST: A convenient interface to the NEST simulator. *Frontiers In Neuroinformatics*, 2009.

Stuart Geman and Donald Geman. Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1984.

Wulfram Gerstner, Werner M. Kistler, Richard Naud, and Liam Paninski. Neuronal Dynamics: From Single Neruons to Networks and Models of Cognition. *Cambridge University Press New York*, 2014.

Marc-Oliver Gewaltig and Markus Diesmann. NEST (NEural Simulation Tool). *Scholarpedia*, 2007.

Dan Goodman and Romain Brette. Brian: a simulator for spiking neural networks in Python. *Frontiers in Neuroinformatics*, 2008.

David M. Groppe, Stephan Bickel, Corey J. Keller, Sanjay K. Jain, Sean T. Hwang, Cynthia Harden, and Ashesh D. Mehta. Dominant frequencies of resting human brain activity as measured by the electrocorticogram. *NeuroImage*, 2013.

W. K. Hastings. Monte Carlo Sampling Methods Using Markov Chains and Their Applications. *Biometrika Trust*, 1970.

Michael L. Hines and Nicolas Ted Carnevale. The NEURON Simulation Environment. *Neural Computation*, 1997.

Geoffrey E. Hinton. Training Products of Experts by Minimizing Contrastive Divergence. *Neural Computation*, 2002.

Geoffrey E. Hinton and Terrence J. Sejnowski. Optimal perceptual inference. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1983.

A. L. Hodgkin and A. F. Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *Bulletin of Mathematical Biology*, 1952.

Jakob Jordan, Mihai Alexandru Petrovici, Oliver Julien Breitwieser, Johannes Schemmel, Karlheinz Meier, Markus Diesmann, and Tom Tetzlaff. Stochastic neural computation without noise. *Arxiv*, 2017.

S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by Simulated Annealing. *Science*, 2007.

Susanne Kunkel, Maximilian Schmidt, Jochen Martin Eppler, Hans E. Plesser, Masumoto, Jun Igarashi, Shin Ishii, Tomoki Fukai, Abigail Morrison, Markus Diesmann, and Moritz Helias. Spiking network simulation code for petascale computers. *Frontiers in Neuroinformatics*, 2014.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 1998.

Luziwei Leng. Deep Learning Architectures for Neuromorphic Hardware. *Master thesis*, 2014.

Luziwei Leng, Roman Martel, Oliver Julien Breitwieser, Ilja Bytschok, Walter Senn, Johannes Schemmel, Karlheinz Meier, and Mihai Alexandru Petrovici. Spiking neurons with short-term synaptic plasticity form superior generative networks. *Arxiv*, 2017.

E. Marinari and G. Parisi. Simulated Tempering: A New Monte Carlo Scheme. *Europhysics Letters (EPL)*, 1992.

Robert W. McCarley and Christopher M. Sinton. Neurobiology of sleep and wakefulness. *Scholarpedia*, 2008.

Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics*, 1953.

Paul Nunez and Ramesh Srinivasan. Electroencephalogram. *Scholarpedia*, 2007.

Dejan Pecevski, Thomas Natschläger, and Klaus Schuch. PCSIM: a parallel simulation environment for neural circuits fully integrated with Python. *Frontiers in Neuroinformatics*, 2009.

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

Mihai Alexandru Petrovici, Johannes Bill, Ilja Bytschok, Johannes Schemmel, and Karlheinz Meier. Stochastic inference with deterministic spiking neurons. *Arxiv*, 2013.

Mihai Alexandru Petrovici, Johannes Bill, Ilja Bytschok, Johannes Schemmel, and Karlheinz Meier. Stochastic inference with spiking neurons in the high-conductance state. *Phys. Rev. E*, 2016.

Björn Rasch and Jan Born. About Sleep's Role in Memory. *Physiological Review*, 2013.

Luigi M. Ricciardi and Laura Sacerdote. The Ornstein-Uhlenbeck Process as a Model for Neuronal Activity. *Biological Cybernetics*, 1979.

Ruslan R. Salakhutdinov. Learning Deep Boltzmann Machines using Adaptive MCMC. *Proceedings of the 27th International Conference on Machine Learning*, 2010.

Johannes Schemmel, Daniel Brüderle, Andreas Grübl, Matthias Hock, Karlheinz Meier, and Sebastian Millner. A Wafer-Scale Neuromorphic Hardware System for Large-Scale Neural Modeling. *Proceedings of the 2010 IEEE international symposium on circuits and systems (ISCAS)*, 2010.

Paul Smolensky. *Information Processing in Dynamical Systems: Foundations of Harmony Theory*. 1986.

Tijmen Tieleman and Geoffrey E. Hinton. Using Fast Weights to Improve Persistent Contrastive Divergence. *Proceedings of the 26th Annual International Conference on Machine Learning (ICML)*, 2009.

Misha V. Tsodyks, Klaus Pawelzik, and Henry Markram. Neural Networks with Dynamic Synapses. *Neural Computation*, 1998.

Laurens J. P. Van Der Maaten and Geoffrey E. Hinton. Visualizing high-dimensional data using t-sne. *Journal of Machine Learning Research*, 2008.

Ullrich Wagner, Steffen Gais, Hilde Haider, Rolf Verleger, and Jan Born. Sleep inspires insight. *Nature*, 2004.

Fugao Wang and D. P. Landau. Efficient, multiple-range random walk algorithm to calculate the density of states. *Physical Review Letters*, 2001.

Robert S. Zucker and Wade G. Regehr. Short-Term Synaptic Plasticity. *Annual Review of Physiology*, 2002.

Acknowledgments

I am using this opportunity to express my gratitude to everyone who helped me overcome energy barriers in the course of this master thesis:

To Prof. Karlheinz Meier of giving me the chance of being part of the *BrainScaleS* project. For conducting the examination, I would like to thank him and Prof. Daniel Durstewitz.

To Wei for supervising this work and thoroughly reviewing the writing.

To further giants on whose shoulders I was allowed to stand:
Mihai for his crystal clear ideas and explanations. Oliver for his high coding standards and for tirelessly answering my software questions. To the other awesome TMAs: Akos, Andreas, Dominik, Max and Nico. I highly appreciate all the proofreading and feedback.

I am thankful to past and present office inhabitants and Electronic Vision(s) for making tough workdays actually pleasant.

My warm thanks are due to my lovely and helping flat mates and friends, my parents and siblings for their endless support and Gerd for being everything.

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, December 8, 2017 .....

(signature)