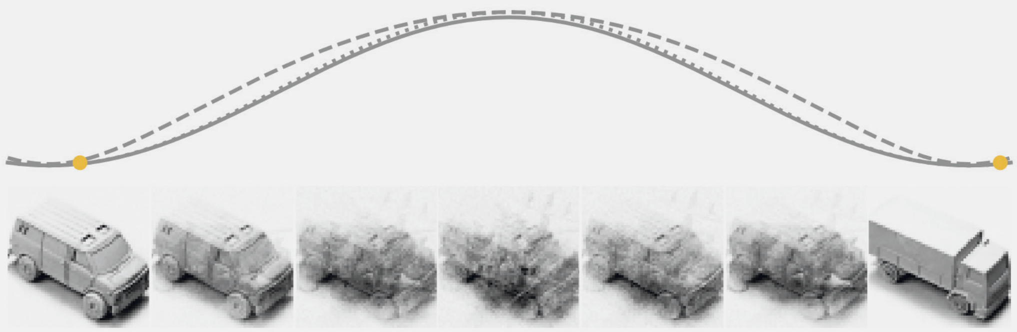


Functions of spiking neural networks constrained by biology

Agnes Korcsak-Gorzo

Information
Band / Volume 119
ISBN 978-3-95806-876-6

Schriften des Forschungszentrums Jülich
Reihe Information / Information Band / Volume 119

Forschungszentrum Jülich GmbH
Institute for Advanced Simulation (IAS)
Computational and Systems Neuroscience (IAS-6)

Functions of spiking neural networks constrained by biology

Agnes Korcsak-Gorzo

Schriften des Forschungszentrums Jülich
Reihe Information / Information Band / Volume 119

ISSN 1866-1777 ISBN 978-3-95806-876-6

Bibliografische Information der Deutschen Nationalbibliothek.
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der
Deutschen Nationalbibliografie; detaillierte Bibliografische Daten
sind im Internet über http://dnb.d-nb.de abrufbar.

Herausgeber und Vertrieb:	Forschungszentrum Jülich GmbH Zentralbibliothek, Verlag 52425 Jülich Tel.: +49 2461 61-5368 Fax: +49 2461 61-6103 zb-publikation@fz-juelich.de www.fz-juelich.de/zb
Umschlaggestaltung:	Grafische Medien, Forschungszentrum Jülich GmbH
Druck:	Grafische Medien, Forschungszentrum Jülich GmbH
Copyright:	Forschungszentrum Jülich 2025

Schriften des Forschungszentrums Jülich
Reihe Information / Information, Band / Volume 119

D 82 (Diss. RWTH Aachen University, 2025)

ISSN 1866-1777
ISBN 978-3-95806-876-6

Vollständig frei verfügbar über das Publikationsportal des Forschungszentrums Jülich (JuSER)
unter www.fz-juelich.de/zb/openaccess.

SUPERVISORS
Prof. Dr. Markus Diesmann
Prof. Dr. Sacha van Albada

REVIEWERS
Prof. Dr. Markus Diesmann
Prof. Dr. Andreas Offenhäusser

Day of Examination: August 27, 2025

To all the spiking neurons that enabled this journey.

SUMMARY

Artificial intelligence (AI) solutions are increasingly taking on tasks traditionally performed by humans. However, their rising computational demands and energy consumption are unsustainable, highlighting the need for more efficient designs. The human brain, evolved to function effectively even when energy is scarce, offers inspiration. Since learning is central to both artificial intelligence and the brain, insights about its underlying principles can deepen our understanding of human learning while informing the development of algorithms that transcend purely engineering-based methods.

This thesis investigates biological learning through two studies, examining it from mechanistic and functional perspectives at an abstraction level commonly employed in neurophysics and computational neuroscience. These fields distill complex neural systems and phenomena into tractable mathematical and computational models, enabling insights beyond the reach of traditional biological approaches. Recognizing that synapses — the connections between neurons — are fundamental to learning, the thesis begins with a review of state-of-the-art computational neuroscience methods for modeling synaptic organization. This review highlights critical aspects of synaptic signaling, including connectivity, transmission, plasticity, and heterogeneity.

In the first study, a synaptic plasticity model is integrated into a spiking neural network simulator and extended with biologically plausible features, for example, continuous dynamics and increased locality. The effectiveness of this enhanced model is demonstrated by training it on a standard neuromorphic benchmark task, incorporating biologically realistic sparse connectivity and weight constraints.

The second study demonstrates that the sampling efficiency of pre-trained spiking neural networks can be enhanced by exposing them to oscillating background spiking activity. Analogous to simulated tempering, these rhythmic oscillations modulate state space exploration, facilitating transitions between high-probability states within the learned representation. These findings establish a link between cortical oscillations and sampling-based computations, offering new insights into memory retrieval and consolidation from a computational perspective.

The research involves developing mathematical and computational models, which are simulated on high-performance computing systems, evaluating learning and sampling performance using standard machine learning metrics, and assessing computational efficiency by analyzing runtime. This thesis shows how biologically inspired mechanisms enhance the functional capabilities of spiking neural networks and how they can guide the development of scalable and efficient AI systems.

ZUSAMMENFASSUNG

Künstliche Intelligenz (KI) übernimmt zunehmend Aufgaben, die traditionell von Menschen ausgeführt wurden. Allerdings sind der steigende Rechenaufwand und Energieverbrauch dieser Systeme nicht nachhaltig, was die Notwendigkeit effizienterer Designs unterstreicht. Das menschliche Gehirn, das sich entwickelt hat, um auch unter Energieknappheit effektiv zu funktionieren, dient hier als Inspiration.

Da Lernen sowohl für die Künstliche Intelligenz als auch für das Gehirn zentral ist, können Einblicke in die zugrunde liegenden Prinzipien unser Verständnis des menschlichen Lernens vertiefen und gleichzeitig die Entwicklung von Algorithmen fördern, die über rein ingenieurwissenschaftliche Ansätze hinausgehen.

Diese Dissertation untersucht biologisches Lernen durch zwei Studien, die es aus mechanistischer und funktionaler Perspektive auf einer Abstraktionsebene betrachten, wie sie häufig in der Neurophysik und der computergestützten Neurowissenschaft verwendet wird. Diese Disziplinen reduzieren komplexe neuronale Systeme und Phänomene auf handhabbare mathematische und rechnergestützte Modelle, welche Einblicke ermöglichen, die über den Rahmen traditioneller biologischer Ansätze hinausgehen. In der Erkenntnis, dass Synapsen — die Verbindungen zwischen Neuronen — für das Lernen grundlegend sind, beginnt die Arbeit mit einem Überblick über aktuelle Methoden in der rechnergestützten Neurowissenschaft zur Modellierung der Organisation von Synapsen. Dieser Überblick beleuchtet wesentliche Aspekte der synaptischen Signale, einschließlich Konnektivität, Übertragung, Plastizität und Heterogenität.

In der ersten Studie wird ein Modell der synaptischen Plastizität in einen Simulator für spikende neuronale Netzwerke integriert und mit biologisch plausiblen Eigenschaften erweitert, wie etwa kontinuierlicher Dynamik und erhöhter Lokalität. Die Effektivität des erweiterten Modells wird durch das Training auf einer standardisierten neuromorphen Benchmark-Aufgabe demonstriert, unter Berücksichtigung von biologisch realistischer spärlichen Konnektivität und Gewichtsbeschränkungen.

Die zweite Studie zeigt, dass die Sampling-Effizienz vortrainierter spikender neuronaler Netzwerke verbessert werden kann, indem sie oszillierenden Hintergrundaktivitäten ausgesetzt werden. Analog zum „Simulated Tempering“ modulieren diese rhythmischen Oszillationen die Erkundung des Zustandsraums und erleichtern Übergänge zwischen hochwahrscheinlichen Zuständen in der gelernten Repräsentation. Diese Ergebnisse stellen eine Verbindung zwischen kortikalen Oszillationen und auf Sampling basierenden Berechnungen her und bieten neue Einblicke in die Gedächtnisabruf- und -konsolidierungsprozesse aus einer rechnergestützten Perspektive.

Die Forschung umfasst die Entwicklung mathematischer und rechnergestützter Modelle, die auf Hochleistungsrechnersystemen simuliert werden. Die Lern- und Sampling-Leistung wird mit Standardmetriken des maschinellen Lernens bewertet, und die Recheneffizienz wird durch Analyse der Laufzeit beurteilt. Diese Dissertation zeigt, wie biologisch inspirierte Mechanismen die funktionalen Fähigkeiten spikender neuronaler Netzwerke verbessern und wie sie die Entwicklung skalierbarer und effizienter KI-Systeme anleiten können.

AUTHOR'S LIST OF PUBLICATIONS

Cortical oscillations support sampling-based computations in spiking neural networks
Agnes Korcsak-Gorzo*, Michael G. Müller *, Andreas Baumbach, Luziwei Leng, Oliver J. Breitwieser, Sacha J. van Albada, Walter Senn, Karlheinz Meier, Robert Legenstein, Mihai A. Petrovici (2022) *PLOS Computational Biology*.
DOI: 10.1371/journal.pcbi.1009753

Phenomenological Modeling of Diverse and Heterogeneous Synaptic Dynamics at Natural Density
Agnes Korcsak-Gorzo*, Charl Linssen*, Jasper Albers, Stefan Dasbach, Renato Duarte, Susanne Kunkel, Abigail Morrison, Johanna Senk, Jonas Stapmanns, Tom Tetzlaff, Markus Diesmann, Sacha J. van Albada (2024) Book chapter in *New Aspects in Analyzing the Synaptic Organization of the Brain. Springer*.
DOI: 10.1007/978-1-0716-4019-7_15

Event-driven eligibility propagation in large sparse networks: efficiency shaped by biological realism
Agnes Korcsak-Gorzo, Jesús A. Espinoza Valverde, Jonas Stapmanns, Hans Ekkehard Plesser, David Dahmen, Matthias Bolten, Sacha J. van Albada†, Markus Diesmann† (2025) *arXiv preprint*.
DOI: 10.48550/arXiv.2511.21674

* Shared first authorship.
† Shared last authorship.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

"Nature is a harmonious mechanism where all parts, including those appearing to play a secondary role, cooperate in the functional whole. "

Santiago Ramón y Cajal in *Advice to a young investigator*, translated 1999

In recent years, artificial intelligence (AI) has emerged as a transformative field, with innovations rapidly changing our everyday lives. AI solutions increasingly perform tasks traditionally carried out by humans. At the heart of AI's success are the principles of neural network learning. The 2024 Nobel Prize in Physics was awarded to Geoffrey E. Hinton and John J. Hopfield in recognition of their pioneering contributions to this topic.

As the name suggests, early neural networks were inspired by the brain, particularly the principle that neurons communicate through connections, whose strengths are modulated during learning. However, modern artificial neural networks, with their intricate architectures and abstract mathematical dynamics, are a product of engineering and have significantly diverged from their biological origins. Training these powerful networks is computationally intensive, resulting in high energy consumption levels that are becoming unsustainable for society.

Optimizing algorithms for energy efficiency has emerged as a critical challenge in advancing AI systems. One promising direction involves revisiting the roots of neural networks and drawing inspiration from biology. Over millions of years, evolution has fine-tuned brain circuits for rapid and efficient learning, enabling skill acquisition, even under conditions of scarce energy. The brain's architecture may hold the key to effective and energy-efficient learning.

Solutions developed by evolution differ fundamentally from those employed in artificial neural networks. In biological networks, activity is sparse and spike-based, functioning in an event-driven manner, whereas artificial networks rely on dense, time-driven communication. Similarly, cortical networks are characterized by sparse, recurrent synaptic connections, shaped by rewiring during evolution and development. In contrast, artificial neural networks are typically dense and feedforward in structure.

Training in artificial neural networks — adjusting weights to generate appropriate outputs based on inputs — shares similarities with functional plasticity in the brain. However, while AI systems often start with randomly initialized weights, the brain's initial structures are fine-tuned through evolutionary processes, providing a significant advantage in efficiency and functionality.

Many tasks targeted by AI systems are already performed by the human brain under comparable constraints. Skills essential for these tasks, explored in this thesis, include managing uncertainty in inputs, generating sequences, and classifying perceived objects. By understanding human learning mechanisms more deeply, we can design novel algorithms that may surpass those developed solely based on engineering principles [1].

This thesis explores the intersection of machine learning and neurobiology through distinct studies addressing specific problems. While each chapter delves into different aspects of this interdisciplinary domain, the work as a whole is guided by several overarching questions:

1. *How can machine learning principles be expressed through biological mechanisms and incorporated into biologically plausible models?*
2. *How does function emerge from both biological and artificial mechanisms?*
3. *How can principles underlying biological mechanisms be leveraged to advance machine learning algorithms?*

These questions are not intended to be exhaustively or systematically addressed or fully resolved within this work. Instead, they serve as thematic anchors, providing a broader conceptual framework for the research.

Despite the biophysical complexity of synapses [2], their behavior can often be effectively described using a few differential equations. Chapter 2 explores the synaptic organization of the brain through the lens of computational neuroscience, systematically reviewing state-of-the-art methods for studying this organization with an emphasis on phenomenological modeling grounded in physics. This chapter provides a comprehensive overview of integrating empirical data into mathematical models, implementing these models in software, and performing simulations that mirror experimental setups. Simulations, regarded as the third pillar of scientific exploration alongside biological experiments and mathematical theory [3], play a crucial role in validating theoretical predictions and experimental findings. The chapter introduces key aspects of synaptic signaling in a progressive manner. It begins with static binary brain connectivity, advances to weighted connections, and finally incorporates dynamics into the connectivity (structural plasticity) and the weights of these connections (functional plasticity). In practice, each step of the modeling and simulation workflow presents unique challenges and potential pitfalls, which are highlighted and addressed.

Chapter 3 builds on a biologically inspired algorithm for spiking networks that approximates the performance of state-of-the-art machine learning methods for training recurrent neural networks. We extend this model by incorporating biological features, analyzing each feature's impact on computational accuracy and efficiency through simulations on a widely recognized benchmark task for neuromorphic hardware and algorithms.

Chapter 4 examines the critical ability of both biological and artificial neural networks to rapidly switch between plausible interpretations of ambiguous or uncertain input. This task becomes increasingly difficult in high-dimensional data spaces due to the curse of dimensionality, where distinct interpretations act as strong and competing attractors. Inspired by a stochastic sampling technique, we hypothesize that cortical oscillations play a similar role in overcoming this challenge. To test this hypothesis, we employ a framework based on the concept that cortical networks perform sampling-based probabilistic inference through their dynamics. We develop theoretical and computational models that incorporate oscillatory activity into this framework, exploring its potential functional roles. Using pre-trained spiking networks, we evaluate how oscillatory spiking activity influences probabilistic sampling, demonstrating its impact on the network's ability to navigate and resolve competing attractor states.

This thesis explores how biological features influence learning capabilities and the ability to sample from a learned distribution, particularly in high-dimensional datasets that mimic real-world inputs. These challenges are especially pronounced in biologically plausible spiking neural networks, where learning depends on the interplay of multiple biological mechanisms. To evaluate the simulated networks, we use various metrics from machine learning.

Ultimately, the algorithms developed in this thesis will reach their full potential for energy efficiency when implemented on specialized hardware designed to emulate brain-like learning, known as neuromorphic hardware. With this in mind, this research primarily addresses the constraints of the interconnected domains of machine learning and neurobiology, while the requirements of neuromorphic hardware are considered as an outlook for future advancements.

2 PHENOMENOLOGICAL MODELING

Author Contributions

Phenomenological Modeling of Diverse and Heterogeneous Synaptic Dynamics at Natural Density. A. Korcsak-Gorzo*, C. Linssen*, J. Albers, S. Dasbach, R. Duarte, S. Kunkel, A. Morrison, J. Senk, J. Stapmanns, T. Tetzlaff, M. Diesmann, S. J. van Albada (2024) Book chapter in *New Aspects in Analyzing the Synaptic Organization of the Brain. Springer*. DOI: 10.1007/978-1-0716-4019-7_15.

* Shared first authorship.

The author and CL contributed equally as the primary contributors to the above publication, and this chapter comprises excerpts from it. MD and SJA served as supervisors for the project. MD conceived the project and all authors collectively provided ideas for the publication. The author and CL organized the proposed ideas, defined subtopics, and assigned groups of authors to conduct the literature review and draft texts for specific subtopics.
The author collaborated with JSt and AM on the subtopic of plasticity. Specifically, the author conducted the literature review and drafted the section on structural plasticity, while JSt performed the same tasks for the section on functional plasticity. The author, JSt, and AM subsequently revised the entire section on plasticity. Figure 2.1, Figure 2.2, Figure 2.3, and Figure 2.4 are adaptations of existing figures from other publications. The author contributed to the adaptations of Figure 2.1, Figure 2.2, and Figure 2.4. Additionally, the author wrote the draft for the introduction, CL drafted the discussion, and the author and CL jointly drafted the notes section. The main revision work was carried out by the author and CL, with all authors contributing to the review process.
Furthermore, Dennis Terhorst and Jessica Mitchell assisted with the technical aspects of the NEST documentation links, Angela Fischer gave advice on the figure layout, and Rainer Waser provided valuable feedback on the manuscript.

2.1 INTRODUCTION

Creating mathematical models from experimental neurophysiological data has grown into an established and essential method for investigating the brain. Based on these mathematical models and exploiting the upswing of affordable and powerful computing architectures over the last few decades, a new sub-field concerned with the computational modeling of neurobiological systems has emerged. The discipline using mathematical modeling and analysis methods to understand principles of brain organization, dynamics, and function is called *computational neuroscience*. This discipline is also sometimes referred to as theoretical or mathematical neuroscience, each term having its own slightly different emphasis. One of the most challenging subjects of this comparatively young domain is the synaptic organization of the brain. This chapter reviews the status quo of synaptic modeling approaches. It aims to provide insight into ways data can be used to build mathematical or computational models.

The methods described in Lübke et al. [2] reveal diverse dynamical processes and heterogeneous components involved in synaptic signaling at various spatial and temporal scales. This variability is amplified by the size and complexity of neurobiological systems: both the density and the total number of synapses in mammalian brains are impressive, the former being on the order of 10^9 per cubic millimeter in the cerebral cortex [4] and the latter being estimated as roughly 5×10^{14} in the human brain [5]. Each cubic millimeter of the human brain contains on the order of $10^4 - 10^5$ neurons adding up to about 10^{11} neurons in the brain as a whole [6].

How can we model such a heterogeneous, complex, and dense large-scale system? The process of modeling and simulation can be understood as a cycle, as depicted in Figure 2.1. First, the experimental results recorded from the system of interest (here, the synaptic organization of the brain) are analyzed, and a mathematical model is formulated. Then, the mathematical model is translated into computer language, i.e., into an executable model that implements the mathematical operations needed to simulate the model. Finally, the model of the system is executed, whereby this simulation is the numerical analog to an experiment. This process yields results that can be compared with the experimental results. In turn, this comparison may deliver outcomes that can be used to improve the mathematical and computational models. Comparisons between the system of interest, the mathematical model, and the executable model ensure quality control. In general, three types of checks for correctness can be distinguished [7]: *Confirmation* ensures that the mathematical formulation applies to the system of interest, *verification* that the executable model sufficiently represents the mathematical model, and *validation* that the simulation outcome is consistent with and predictive of the system of interest. This review focuses on the inner triangle of arrows: the practical methods to formulate mathematical models in an informed way, to translate them into manageable and correct executable models, to run simulations, and to inform further modeling choices using the obtained data.

The challenge of computational neuroscience is to analyze the rich dynamics of neuronal systems and abstract their complexity into mathematical models that still capture essential characteristics of the experimental findings. At the same time, these models should be simple enough to be tractable and generalizable and thus reveal possible laws that govern the dynamical system. A fundamental question in this endeavor is what processes and variables are of interest and best

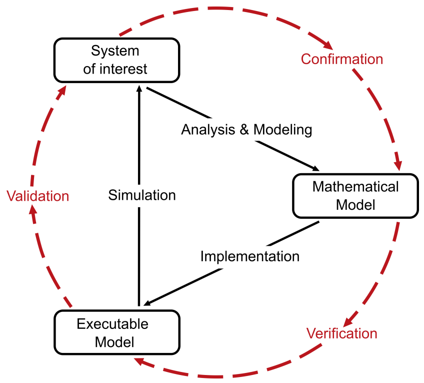


Figure 2.1: **Cycle of modeling and simulation.** The empirical data from the brain structure under study (the "system of interest") is first mathematically modeled and then implemented in software. Closing the loop, simulation results can be compared to experimental recordings. Reproduced with permission from Fig. 1 in Trensch et al. [7].

describe the data. In synaptic organization, candidates include the connection strengths, synaptic time constants, delays, vesicle release characteristics, synaptic plasticity, and neuromodulation.

Developing this thought further, a modeler needs to decide on the scale the model represents and how to parameterize it. It is advisable to constrain the number of model parameters to a minimal set that answers a specific research question. Limiting the parameter space increases the tractability, mechanistic interpretability, and robustness of the model and reduces the risk of overfitting. However, capturing biological detail and enhancing the direct link between parameters and their biological counterparts can usually only be done with a large set of parameters. Heterogeneity can be represented by introducing parameter value distributions, leading to additional parameters characterizing the dispersion and possibly higher-order properties of the corresponding distributions. Overall, a suitable parameterization involves a tradeoff between the model's controllability and biological plausibility.

These decisions on which aspects of the system to express as variables and the choice of the corresponding model equations are abstraction steps: they formalize a hypothesis on which features are germane to the question at hand and which mathematical descriptions are appropriate for capturing the phenomena of interest (see Section 2.3.8). In general, this abstraction can be approached from two different directions. The *bottom-up* approach starts from the low-level properties of the neurons and synapses making up the system and models the complexity step by step in the hope of achieving realistic dynamical and functional properties. However, one major point of modeling is to improve our understanding of a system. Given that the starting point is a poor understanding, this approach suffers from the fundamental problem that essential features might be abstracted away or obfuscated by an abundance of less relevant details. Another point can be to provide accurate predictions, even if we do not understand the model. The opposite approach, *top-down* modeling, starts from the high-level dynamical, functional, or behavioral properties one would like to capture and then proposes concrete implementations. The drawback of this approach is that a model created in this way might not fully conform to biology, so it is difficult to draw conclusions about the brain. One solution is to use different degrees of abstraction at different scales to arrive at an understanding of the system, which is the motivation behind *multi-scale modeling*. Ideally, biological realism is incrementally enhanced through cycles of data comparison and refinement (see Figure 2.1 and Section 2.3.1).

Some neurophysiological observations can be modeled with analytically solvable equations, i.e., in an exact way and usually with pen and paper. However, various simplifying assumptions generally flow into such abstractions, and deriving an analytical solution to a model's equations becomes less feasible as its complexity increases. For such cases, numerical solutions can provide a useful alternative. This computational approach tends to be slower, but it can validate the analytical approach by requiring fewer simplifying assumptions and it may even provide novel theoretical insight.

Simple small network models frequently consist of equations that can be solved analytically or calculated numerically with few computational resources. However, both numerical and analytical approaches reach certain limits when attempting to replicate realistic neuron numbers in the volume of the brain region under consideration. As the number of neurons increases, the number of connections grows quadratically in networks without spatial dependence and linearly for distant neurons in models incorporating spatial dependence since most connections are local. To approximate *natural density*, analytical techniques like mean-field theory sacrifice biological specificity. With sufficient computing power, numerical methods may solve model equations at natural density. A limitation is that, to date, this is only possible for small brains or small portions of larger brains.

We restrict the scope of this review to phenomenological models, which represent the empirical relationship between phenomena without explaining the reason for the interaction. We neglect the molecular level or ultrastructure, i.e., structures visible at magnifications higher than that provided by standard optical light microscopy. Furthermore, we address spiking neuron models, mainly so-called point or few-compartment neurons, which neglect the precise morphology of the neuron, as the effective dynamics of a morphologically complex neuron can often already be meaningfully captured by such models.

The models are usually translated to be executable by a computer, i.e., implemented in one of the various computer languages. One or an interconnected set of dynamical model components representing neurobiological entities such as neurons or synapses is *simulated*, i.e., evolved in time, for a specific duration with a set of parameters, initial conditions, and stimuli.

The generic framework that can numerically solve the dynamical equations of various models is called a *simulator*. It solves complex interactions with many coupled differential equations typically incrementally in discrete time steps. Spiking neuronal network simulators also communicate spike events from senders to receivers. The event-based nature of synaptic interactions in the form of action potentials is advantageous for the efficiency of the simulation. Furthermore, a simulator can record dynamical state variables in the network or other observables like connectivity and apply stimuli during simulation.

Since the simulator needs to organize and maintain the appropriate data structures in computer memory, a substantial amount of RAM may also be required, depending on the simulated neurobiological system. Biologically realistic network models are often large-scale to reflect or approach the natural density of neurons and synapses, have additional computational overhead due to heterogeneity, and are typically simulated for a long time, e.g., to gather statistics or study behavioral timescales. Thus, a simulator should be efficient and scale to high-performance architectures in terms of processing and memory usage.

In addition to these performance aspects, criteria for a good simulator include functional completeness, numerical accuracy, and reproducibility of results. Furthermore, a simulator increases its value for the community if the available models are relevant to many members and the documentation is comprehensive and easy to understand. Developing a simulator that fulfills all these criteria and supports diverse models is complex and time-intensive. Consequently, simulators with peer-reviewed collections of implemented models are continuously developed as a community effort and shared as software packages to benefit the field of computational neuroscience.

From the range of existing simulators for biologically inspired neuronal networks, this review focuses on NEST [8], an open-source software tool designed to simulate anywhere from small to large-scale networks of diverse spiking neuron models, and its associated domain-specific modeling language NESTML [9] that facilitates the creation of new neuron and synapse models. Other simulators with various scientific foci and special areas of application include NEURON [10], Brian [11], Nengo [12], Arbor [13], and ANNarchy [14]. Some common (simulator-agnostic) interfaces are provided by PyNN [15] and the modeling language NeuroML [16].

This chapter is structured as follows. Each section from Section 2.2.1 to Section 2.2.5 presents a two-step recipe to go from experimental data on a specific feature or mechanism of the synaptic organization to simulations: first, how to mathematically model experimental data, and second, how to simulate this model. Section 2.2.1 starts with the most simplified view of brain circuitry, namely binary connections between neurons. This view is advanced in Section 2.2.2 to the notion of weighted connections. Then, dynamics is introduced to the existence of connections in Section 2.2.3 and the weights of those connections in Section 2.2.4. Finally, Section 2.2.5 discusses how to model the additional heterogeneity in all these features. Throughout the text, links to Section 2.3 highlight aspects that are particularly challenging or contain pitfalls. Furthermore, links to provide URLs to NEST code snippets that help the reader develop an intuition on how the usage of the discussed models would look in code. The chapter ends with some concluding remarks in Section 2.4.

2.2 ASPECTS OF SYNAPTIC ORGANIZATION

2.2.1 *Connectivity*

From empirical data to mathematical models

Anatomical and physiological experiments are yielding ever richer data sets on brain connectivity. Integration of these data into dynamical models can help gain insight into their implications for brain activity and function, besides identifying gaps in the data which can inform future experiments. The data cover diverse scales, ranging from electron microscopy at the sub-micron scale of synapses to light microscopy for neuronal morphology, paired recordings identifying fractions of connected neuron pairs, glutamate uncaging at the scale of tens to hundreds of microns, axonal tracing for long-range connectivity, and diffusion imaging for the whole-brain scale [17].

Despite the richness of the available data, none of these experimental approaches can specify full connectomes at the single-neuron resolution, especially in organisms with complex brain structures such as mammals. Therefore, we need to make predictions in order to complete the detailed connectivity data. One strategy is to find statistical regularities in the existing data and use these to extrapolate to missing data points. Of course, models do not need to be fully data-driven; various abstractions may be used to explore the influence of specific aspects of the connectivity. We illustrate the data-driven approach using the example of the cerebral cortex. In view of the incompleteness of the known cortical connectivity for any individual, we describe the connectivity in a probabilistic manner. A different strategy for generating the connectivity may be to grow connections according to developmental or other plasticity rules (for structural synaptic plasticity, see Section 2.2.3).

The cerebral cortex contains different types of excitatory and inhibitory neurons, distinguished by their morphology, electrophysiology, connectivity, and molecular make-up [18, 19]. We refer to the set of neurons of the same type in a given cortical area and layer as a *population* (see Figure 2.2). Connection probabilities are specific to both source and target populations. Both within and between areas, connectivity is also layer-specific. Furthermore, connection probability decays with the distance between neurons, both locally within areas and at longer ranges between areas [20–22]. A further organizing principle is that excitatory connectivity tends to form patches [23, 24], meaning that neurons establish additional synapses onto other nearby neurons resulting in spatial clusters (see Figure 2.2).

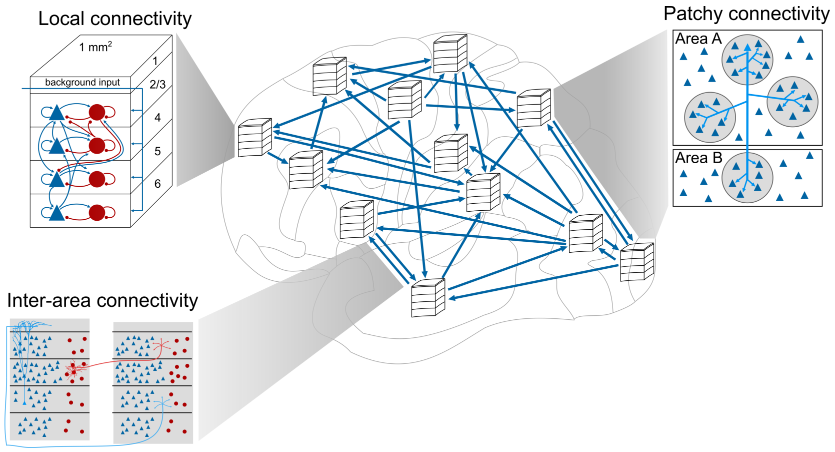


Figure 2.2: **Substructures of the cortex on different scales.** The neurons in the cortex are organized into areas on the macroscale (middle). In each area, neurons are organized into layers with distinct connectivity (here, shown as a microcircuit under 1 mm^2 of cortical surface) and into populations of neurons with similar properties within each layer (upper left). Blue triangles and red circles represent excitatory and inhibitory neurons, respectively. Long-range projections connect the areas (lower left). Moreover, on an intermediate scale of millimeters, both intra-area and inter-area excitatory connections cluster into "patches" (right, only outgoing connections for one neuron shown). Adapted with permission from Fig. 1 in Schmidt et al. [25] under license CC BY 4.0 originally from Fig. 1 in Potjans et al. [26] and Fig. 1 in Kunkel et al. [27].

When formalizing these properties into models, a number of subtleties are involved [28]. First, the term *connection probability* needs to be defined carefully. This could, for instance, refer to either the total number of synapses divided by the product of the source and target population sizes or the probability for any neuron pair to be connected via at least one synapse. The two definitions diverge in the case of *multapses*, multiple synapses between a given source and target neuron pair, often observed in reconstruction data [29]. Further, models can either allow self-connections, also called *autapses*, or prohibit them. Moreover, beyond a certain model size, the spatial decay of the connection probability becomes important. To capture this, simulated neurons are assigned spatial coordinates, and additional specifications are necessary, including boundary conditions and the choice of connectivity profile. Common choices for the local profile are Gaussian and exponential functions, where the latter generally appears to be a better approximation to experimental data [20, 21].

Figure 2.3a illustrates the local decay of connectivity with distance. Choosing a symmetric exponential as a model, Figure 2.3b shows that fitting to the experimental data can reveal fundamental constants such as the characteristic length λ.

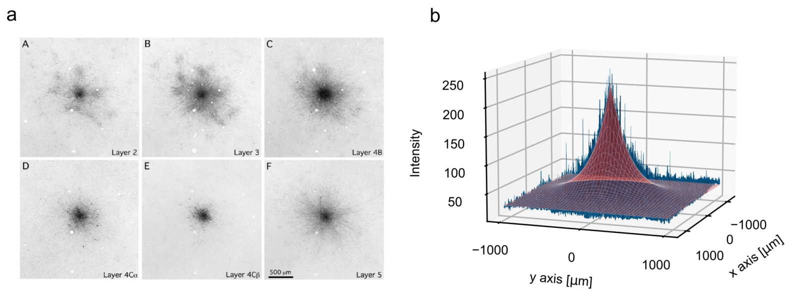


Figure 2.3: **Fitting a model of connectivity to observed data. (a)** Layer-resolved axonal tracing data from V1 of New World monkeys. (Adapted from Sincich et al. [30], Fig. 3, Copyright 2001 Society for Neuroscience) Images show staining of cortical layers after an injection of biocytin into layer 3, anterogradely staining axons. **(b)** Fit of a 2D symmetric exponential function $f(r) = ae^{-\lambda/r} + b$ (red) to the axonal density distribution (blue) of layer 5. The model does not fully capture the connectivity profile for layers 2, 3, and 4B **(b)** (A, B, C), which display patchy connectivity.

When including *patchy connectivity*, the spatial position of the patches can be specified via a radial distance from a cell body and an angle [24]. Further possible parameters are the number of patches, the size of each patch, and the degree of overlap between patches. Layer-specific axonal tracing data, such as fractions of supragranular labeled neurons from retrograde tracing experiments [31], can inform the laminar inter-area patterns of cortical models. Here one should pay attention to the fact that, on the target side, axonal tracing tells us about axonal or synaptic locations but not about the locations of the target cell bodies. To a reasonable approximation, one can statistically estimate which synapses are established on which target neurons using morphological reconstructions, a method that assigns the number of synapses proportionally to the total length of dendritic elements in the vicinity of the synapses [32, 25].

This is only a tiny selection of data and features that can be included in neuronal network models. One can go into greater complexity and, for example, consider the higher-level organization of networks, such as hierarchical modularity or small-world properties. For a further discussion on model detail in general, see Section 2.4.

From mathematical models to simulation

To simulate how the dynamics of a neuronal network model evolve, the mathematical model description needs to be translated into an executable one. This is preferably done using a dedicated simulator to avoid mistakes in the implementation and to enhance comparability and reproducibility of results (for the precise definition of the different forms of *reproducibility*, see Section 2.3.2.) Executing a neuronal network simulation typically involves two successive phases: during the *build phase*, the network is set up on the machine by instantiating objects and data structures for neurons and synapses. The subsequent *simulation phase* propagates the network state for a specified biological model time. How fast a simulation runs, i.e., how the biological model time relates to the wall-clock time, crucially depends not only on the machine specifications but also on the representation of the network model on the machine. Parallel computing combines the computational power of many separate compute cores or nodes to enable large-scale simulations; to this end, NEST uses a hybrid approach with the Message Passing Interface (MPI) and Open Multi-Processing (OpenMP). The former enables parallel computing on multiple processors with distributed memory, while the latter enables parallel computing even on single processors with shared memory, referred to as *threading*. The total number of so-called *virtual processes* is determined as the product of the number of MPI processes and the number of OpenMP threads per process. A direct mapping between network structure and hardware is in general difficult to realize. Therefore, NEST uniformly distributes the neurons of each population across the available processors to balance the compute load (Section 2.3.3). The neurons are connected via synapses, which are assigned specific *weights* and *delays* reflecting conduction times. Synapse models are stored and updated on the same compute nodes that hold their postsynaptic partner neurons. Maintaining the complete network connectivity in computer memory enables the use of plasticity mechanisms that can modify synaptic strengths at runtime (for functional synaptic plasticity, see Section 2.2.4). The alternative *procedural connectivity* approach generates the required routing information on the fly and thereby requires fewer memory resources [33, 34].

Establishing synapses in a computational network model requires defining which neurons are connected. For specific data-driven models, the network structure can be loaded from a file, but simulators also provide built-in routines for generating connectivity. These routines[1] range from a primitive that just connects individual source and target neurons, to high-level connection rules acting on the neuron population level [28]. For example, the deterministic rule *all-to-all* connects each neuron of a source population to each neuron of a target population. Probabilistic rules account for the often statistically described sparse connectivity in biological neuronal networks. *Random, fixed in-degree* connectivity, for instance, specifies only the number of incoming connections per neuron but not which individual ones are selected as sources. If the connectivity is described as *pairwise Bernoulli*, each pair of neurons is connected with a given probability.

1 https://nest-simulator.readthedocs.io/en/latest/tutorials/pynest_tutorial/part_3_connecting_networks_with_synapses.html

The fixed in-degree rule needs to be combined with the specification of whether multapses are allowed, whereas the pairwise Bernoulli rule excludes them by definition as each pair of neurons is considered only once. High-level connection rules enable efficient low-level implementations such as parallelization of the network construction.

Pseudo-random number generators (pRNGs) are used for drawing connections according to a probabilistic rule and optionally also for setting neuron and synapse parameters (see Section 2.2.5). The resulting network realization will be identical if the same sequence of random numbers is sampled; this is achieved by fixing the pRNG seed. Random distributions sometimes have to be constrained in order to restrict the sign of a weight, e.g, according to Dale's law [35], or to enforce connection delays to be larger than the simulation time step, for which a typical value is 0.1 ms. A longer minimum delay, for instance, 1 ms, can furthermore be used to limit the necessary frequency of communication between virtual processes.

Large-scale neuronal network models require high-performance computing. Employing several compute nodes in parallel not only distributes the workload but also gives access to sufficient memory for storing the network connectivity. Storing a single synaptic weight costs 8 bytes in NEST [36] as it sums up the effects of a set of vesicles that may differ in size, as well as of potentially different amounts of receptors. Moreover, a typical neuron in the mammalian brain has on the order of 10^4 synapses [37]. This leads to a substantial amount of resources required for large models. Networks with reduced neuron and synapse numbers can preserve some characteristics (e.g., firing rates) of full-scale networks if the downscaling is compensated for with informed parameter adjustments [38]. The pairwise correlation structure of the neuronal activity, however, cannot be preserved simultaneously, rendering neuroscientific simulations at natural density a necessity where correlation structure is relevant. This may be the case, for instance, to ensure the correct network state: correlation changes may even shift a network between linearly stable and unstable regimes. Data structures that keep the memory usage per MPI process constant regardless of the total number of MPI processes used in the simulation [39] provide a potential solution, paving the way toward brain-size networks with realistic connectivity.

2.2.2 *Synaptic transmission*

From empirical data to mathematical models

Electrochemical signaling between neurons is mediated by various receptor types, expressed post- and presynaptically. Different receptor types trigger different physiological responses. Ligand- or voltage-gated ion channels influence ionic flows through the membrane, with distinctive kinetics for each receptor type (examples include AMPA and NMDA). Metabotropic receptors trigger intracellular biochemical signaling cascades with downstream actions that are typically not instantaneously noticeable but mediate physiological adaptation processes. *Electrical synapses* (*gap junctions*) are transmembrane channels that form a direct electrical and biochemical coupling between the cytosol of two adjacent cells. Compared to *chemical synapses*, they provide increased speed as the signal does not need to be converted from electrical to chemical and back across a synaptic cleft. The composition of receptor types on a neuron's synapses, their spatial distribution on the dendritic tree and cell body, and their individual, instantaneous efficacy and response kinetics determine how and at which timescales the neuron filters and integrates its many presynaptic inputs. We will focus here on chemical synapses.

The amplitude of the postsynaptic response is proportional to the synapse's strength or *weight*, which depends on the amount and types of both neurotransmitters and receptors as well as the state of the postsynaptic neuron. Synaptic weights can be estimated from paired recordings, usually *in vitro*, to avoid background activity that confounds the measurements. However, here it should be taken into account that the synaptic weight obtained from paired-cell recordings is determined by a combination of biophysical properties, e.g., postsynaptic receptor density, amount of released neurotransmitters, reuptake kinetics, or existence of more than one connection between a pair of cells (multapses). Hence, the terms *strength* and *weight* refer to an effective, phenomenological quantity. Mapping the corresponding parameters to the *in vivo* condition is nontrivial because experimental conditions like temperature and extracellular fluid may differ, as well as a high-conductance network state affecting the measured quantities such as time constants [40, 41]. Change of synaptic strengths over time is discussed in Section 2.2.4.

When a presynaptic neuron has emitted an action potential, the signal that arrives at the postsynaptic neuron can be observed as a postsynaptic potential (PSP): the deflection in the somatic membrane voltage caused by the incoming spike. Alternatively, synaptic currents (PSCs) may be measured at different holding potentials using voltage-clamp recordings. From a PSC or PSP, the weight of the synapse can be derived. Synapses are often modeled as injecting a current into the postsynaptic cell or acting as a conductance, the current of which is proportional to the difference between the membrane potential and a synapse- or receptor-specific reversal potential. In the simplest approximation of the postsynaptic response kinetics, the time course of this current or conductance may be modeled as a Dirac delta function causing a step increase in the membrane potential or current, respectively. With increasing levels of complexity, the time course of the PSC (or postsynaptic conductance, PSG) can be approximated by an instantaneous rise followed by an exponential decay or by a double exponential with separate time constants for the rising and the decaying phase [42].

Beside spatially precise communication via synapses, the spatially more diffuse process of *neuromodulation* can alter the excitability of neurons and affect synaptic plasticity (see Section 2.2.4). Neuromodulation is achieved by the release of a neurotransmitter with less detail in the connectivity patterns than in typical synaptic (for instance, glutamatergic) neurotransmission. The neuromodulator, such as dopamine or serotonin, is typically released from a neuron whose cell body lies in a small, circumscribed nucleus in the brain but which projects broadly and affects many downstream targets simultaneously. The precise spatiotemporal profile of neurotransmitter concentration is often approximated in models by assuming the neuromodulator diffuses through extracellular space, referred to as *volume transmission*. Simulating the diffusion process entails solving the Laplace equation—an equation that involves the spatial gradient and divergence operators, requiring a different type of solver than those that solve the neuronal network system dynamics. Instead of a detailed representation of the geometry of extracellular space (for instance, based on the finite-element method), the medium may be assumed to be spatially homogeneous, and diffusion can even be assumed to occur instantaneously, considerably simplifying the model and its computational requirements [43].

Neurons have a spatial extent, and their dendrites often exhibit intricate branching patterns. Consequently, the spatial collocation of synapses on the dendrites has important consequences for the neuron's response to input. Dendritic responses are often nonlinear, as dendrites are studded with a high density of voltage-gated channels, which, combined with intracellular responses like calcium signaling, can cause a nonlinear interaction between nearby synaptic inputs. In addition, the dendrite itself can exhibit action potentials distinct from a somatic action potential, for instance, involving a local, intracellular calcium transient [see, e.g., 44]. The (local) change in membrane potential and conductance, in turn, affects the integration at adjacent synapses in the branch. The triggering of dendritic action potentials by co-activated and co-located synapses and their effects on the somatic dynamics can be accounted for in simple point neuron models by including nonlinearities in synaptic input currents [45, 46]. For a more fine-grained analysis, multicompartment models are commonly used. In these models, each neuron consists of dozens or hundreds of compartments, each equipped with a distinct type of dynamics and parameterization and coupled to neighboring compartments according to Ohm's law [42]. Multicompartment models permit integrating experimental data at a highly detailed level of description but are computationally and conceptually much more complex and demanding. On the other hand, some biophysical details like synaptic adaptation can be adequately modeled without the need to address the microscopic biophysics of synaptic vesicles but can be treated phenomenologically by adding one or a few extra continuous state variables to the model [e.g., 47, 48].

From mathematical models to simulation

NEST integrates equations for neurons with linear subthreshold dynamics exactly [49] and uses standard numerical solvers for nonlinear neuron models. For synapses, an efficient approach is to specify the characteristic time evolution of some postsynaptic quantity, such as current or conductance, as a linear system of equations. If responses sum linearly across a neuron's synapses, they can be lumped together into a single or a few state variables and do not have to be stored and updated for each synapse separately. For this reason, the postsynaptic response is typically specified as part of the (postsynaptic) neuron model. As described in Section 2.2.2, the postsynaptic kernel could be, for example, a Dirac delta function (causing an instantaneous jump in the postsynaptic membrane potential), an instantaneous rise followed by an exponential decay, or a double-exponential function with a finite rise time. Furthermore, because they are linear, solving these equations does not require a numerical solver but only multiplication with a constant at each time step [50]. The reduction to a simple multiplication generally makes the solution much more precise: the computed values are closer to the mathematically "true" solution and more efficient to compute. Thus, simulations of networks with many synapses become feasible. Multiple types of synapses can be easily incorporated into this scheme by grouping them according to their kinetics, for instance, into a separate AMPA and NMDA group[2].

In simulations of large networks, the layout of data structures in memory and communication can become bottlenecks. Conceptual modeling decisions can interact with data layouts; for example, the synaptic delay can be chosen as a property of the synapses or the pre- or postsynaptic neurons. In the point-neuron framework, the delay is assigned to either a neuron's axonal or dendritic side, implying different biophysical interpretations and simulation outcomes. The biophysical object of a synapse is not necessarily represented in code by a specific software object but distributed into a presynaptic and a postsynaptic component. In the instantiation of a particular model, these components may not even live on the same compute node. As noted in Figure 2.2.1, NEST stores synapses on the process containing the postsynaptic neuron.

In simulations using parallel computing, spike events and potentially other quantities such as synaptic weights have to be communicated between threads, processes, or across a computer network (Section 2.3.4). Parallel computing presents a set of unique design requirements because the evolution of the dynamical model needs to occur synchronously, lest the model's state becomes internally inconsistent when some parts of it have become desynchronized in time. This requirement can be addressed by instituting a minimum, nonzero transmission delay for each synaptic connection in the model. A delay between the presynaptic spike and the resulting postsynaptic response effectively decouples neurons for this time window so that events can be transmitted across the computer network in a regular cadence at the end of each window (Section 2.3.5). This decoupling allows simulations to scale to many compute nodes [51].

From a mathematical modeling point of view, gap junctions are much simpler than chemical synapses; their delay is negligible, and they do not filter the input.

2 https://github.com/nest/nestml/blob/master/models/neurons/wb_cond_multisyn.nestml

However, modeling gap junctions numerically can be challenging because they entail an instantaneous coupling between compartments. The *waveform relaxation* technique helps retain simulation efficiency when combining gap junctions with a numerical simulation method that takes advantage of a minimum, nonzero synaptic delay. Each neuron is considered a separate subsystem in this technique, and the gap junction coupling terms (current flowing from one neuron into the other) are solved iteratively. This solution requires exchanging data (in particular, membrane potentials) between the gap-junction coupled neurons only at the end of each minimum delay step, thus limiting the necessary communication frequency. In exchange, it requires only a modest increase in computation and the size of the communicated packets since solving the forward dynamics of each separately considered neuron needs to be repeated only once per iteration of the waveform relaxation algorithm [52, 53].

2.2.3 *Structural plasticity*

The models introduced earlier in this chapter had static connectivity (see Section 2.2.1). However, macroscopic observations of the brain have revealed that the connections in cortical networks continually change as new synapses form and others dwindle and disappear [for a review, see 54]. This rewiring is a lifelong process to encode experiences but happens extensively during development and recovery from lesions in the brain tissue [for a review, see 55]. The underlying mechanisms introducing dynamics into the connectivity are summarized as *structural synaptic plasticity*.

From empirical data to mathematical models

Including structural plasticity mechanisms into a synapse model can increase its biological plausibility, e.g., regarding learning, development, reformation after lesions [56] or topographic map formation [57]. Structural plasticity might also be the basis for associative connections [58] and metaplasticity [59]. However, plasticity mechanisms capable of generating network connectivity in a principled fashion can also be helpful in other ways. First, they can help reduce dependence on cumbersome and expensive connectivity recordings in animals (see Section 2.2.1). Second, they can serve a range of functional purposes. For example, they can enhance learning performance [60] or increase the storage efficiency of long-term memories and, by that, prevent catastrophic forgetting [61]. Plasticity mechanisms also frequently serve a homeostatic function. In general, the term "homeostasis" refers to a range of vital physiological processes that assist organisms in maintaining internal states (such as body temperature, blood sugar levels, and heart rate) at optimal levels. Likewise, homeostatic plasticity maintains quantities such as spiking activity or numbers of connections at an energetically or computationally favorable set-point [62]. Efficient pruning of the connectivity and preserving sparse connectivity [63] can help save energy and optimize the usage of limited synaptic resources, which is particularly important in neuromorphic computing [60, 64, 65].

To understand the process of developing a comprehensive and accurate mathematical model of structural plasticity, the following paragraphs sketch the steps involved in creating the model suggested by Butz et al. [56] as an example. This model is based on the observation that the creation and deletion of synapses can bring the postsynaptic neuron's firing rate into a certain physiological range. The authors consider synaptic elements, namely axonal boutons on the presynaptic side and dendritic spines on the postsynaptic side. When two such elements are combined, a synapse is created. The dynamics of the number of synaptic elements for each neuron depends on a readiness variable c (associated with the calcium concentration), which indicates the propensity of a neuron to grow synapses. The readiness variable is a low-pass filtered version of the spiking activity and thus approximates the neuron's instantaneous rate up to a scalar multiplier.

The algorithm comprises four steps, which are repeated until the connectivity converges. First, it continuously updates the spiking activity of the neurons since each neuron's mean firing rate influences the creation of synaptic elements. Second, it updates the readiness c for each neuron:

$$\frac{\mathrm{d}c}{\mathrm{d}t} = -\frac{c(t)}{\tau} + \beta\delta\left(t - t_j^{\mathrm{f}}\right) \ , \tag{2.1}$$

i.e., c decays exponentially with the time constant τ and increases by a fixed amount β whenever the neuron j spikes at t_j^{f}, where $\delta(\cdot)$ denotes the Dirac delta function and f stands for "firing". Third, a homeostatic rule drives the neuron to reach and maintain a target activity by deleting postsynaptic elements if the instantaneous activity is higher than the target activity or creating synaptic elements if the current activity is lower than the target activity. A growth curve defines the speed of these modifications toward a target calcium concentration c_{target} by means of a growth rate ν and can be expressed as a linear function

$$\frac{\mathrm{d}z}{\mathrm{d}t} = \nu\left(1 - \frac{c(t)}{c_{\mathrm{target}}}\right) \ . \tag{2.2}$$

Alternatively, a downward shifted Gaussian or other more complex function may be used, as long as it has a zero-crossing with a negative gradient that allows convergence. If the value of z increases or decreases by 1, the neuron grows or deletes a synaptic element, respectively. The algorithm creates new connections between randomly chosen synaptic elements from the available set in the fourth and last step. This set comprises synaptic elements generated in previous iterations that are not yet connected and the connection partners of deleted synaptic elements.

Beyond this specific example, there exists a range of different structural plasticity models: some have rules for deleting, some for forming synapses, and some for both [for a book, see 66]. Often the algorithm prunes synapses that do not have the chance to become active again [67], are the weakest according to a specific metric [68, 69], or have too little causal correlation between pre- and postsynaptic spikes [70]. Sometimes the mechanism's objective is to maintain a preset number of connections [60], preserve short-range connections [71], or prune until the connectivity has converged to the most efficient constellation [67]. In many algorithms, the determining factors for rewiring are the pre- and postsynaptic activity and the vicinity of other synapses. In general, one can divide structural plasticity mechanisms into two categories: Hebbian structural plasticity, which leads to an increase in the number of synapses during phases of high neuronal activity and, conversely, a decrease in phases of low neuronal activity; and homeostatic structural plasticity, which balances these changes by removing and adding synapses [72].

From mathematical models to simulation

The NEST implementation of the discussed particular structural plasticity mechanism [73] updates the network connectivity at time intervals that are long compared to the computational time step used to update the neurons, based on experimental observations[3]. This slow timescale makes the algorithm more efficient, as the available synaptic elements do not need to be calculated and communicated at every time step. However, when using structural plasticity to generate connections in a network, note that convergence is not guaranteed but determined by the growth rate, network connectivity, and network activity; thus, visual guidance is advised (see Nowke et al. [74] and Section 2.3.6).

2.2.4 *Functional plasticity*

The strength of a synapse is usually parameterized by a single static value, the synaptic efficacy or synaptic weight (see Section 2.2.2). However, existing synapses can grow stronger or weaker as an effect of a variety of biophysical mechanisms on both the pre- and postsynaptic side, phenomena collectively known as *functional synaptic plasticity*. These adjustments of synaptic efficacies are likely to form the basis of learning and memory processes in the brain. Thus, this section addresses the temporal evolution of synaptic efficacies and the underlying mechanisms.

From empirical data to mathematical models

Over 70 years ago, Hebb famously postulated that neurons that fire together wire together [75]. Since then, many phenomenological models of functional plasticity have been derived and developed. Introducing categories brings some order into the vast landscape of models, even if they do not have clear-cut boundaries and often overlap. Four categorizations are common. First, with

3 https://www.nest-simulator.org/py_sample/structural_plasticity/

respect to the timescale: while short-term plasticity models cover timescales from milliseconds to seconds, long-term plasticity models cover minutes to hours, and homeostatic plasticity models (e.g., synaptic scaling) even up to days [76, 77]. While structural plasticity can occur over the course of hours [78], timescales of functional plasticity are typically shorter. One needs structural plasticity to create new synapses (happening on a long timescale) which then grow stronger via functional plasticity (happening on a faster timescale). Vice versa, synapses that have grown weak are more likely to be pruned. To date, models usually include either functional or structural plasticity but not both together, and the effects of these mechanisms on synaptic learning are thus studied independently. A second categorization distinguishes functional plasticity according to the mechanisms involved: Unsupervised learning rules are based on unlabeled data, supervised learning rules involve a target signal, and reinforcement learning rules function via rewards [76, 77]. A third categorization considers the number of factors that constitute the update formula of the synaptic efficacy: Standard correlation-based rules usually involve two factors, the pre- and postsynaptic spiking, as opposed to three-factor models that involve an additional modulatory signal, e.g., neuromodulation. Fourth, based on activity dependence, plasticity mechanisms occur in two counteracting forms: in Hebbian-type mechanisms, higher activity levels of the pre- and postsynaptic neurons lead to strengthening of the (positive or negative) weights, whereas in homeostatic mechanisms, to their weakening [72]. Without further constraints, Hebbian-type plasticity may lead to a positive feedback loop and, consequently, substantial changes in synaptic weights and network activity. In contrast, homeostatic synaptic plasticity pushes the synaptic efficacy up if activities are low and down if neuronal activities are high, inducing a negative feedback loop and stabilizing the dynamics.

Experimental studies show that the efficacy of a synapse can change for a short time window of hundreds to thousands of milliseconds depending on the history of the presynaptic spikes [79–81]. This phenomenon is termed *short-term plasticity (STP)*, or more precisely, *short-term facilitation (STF)* if the efficacy is elevated, and *short-term depression (STD)* if the efficacy is decreased. The biophysical mechanism underlying STP is the dynamics of vesicle pools and spike-triggered exocytosis. On the one hand, after the generation of a spike, calcium accumulates in the presynaptic axon terminal, increasing the probability of neurotransmitter release, which enhances the synaptic efficacy and thus causes STF. On the other hand, repetitive firing leads to the depletion of vesicles and saturation of postsynaptic receptors, which decreases the efficacy and thus causes STD. The mechanisms for STF and STD are counteracting, and a combination of both can be present in the same synapse. Depending on the synapse or neuron type, one of them may be more pronounced. These phenomena form the basis for many STP models [for a review, see 82]. The following paragraph outlines one possible modeling approach by using the example of the model proposed by Tsodyks et al. [79].

The starting point is the view introduced in Section 2.2.1 and Section 2.2.2: a neuron k receives spikes from neuron j over a synapse with the weight w_{jk}. Now, to make the static synaptic weight a dynamical variable, w_{jk} is multiplied by a time-dependent scaling factor $f_{\mathrm{a}}(t)$, modeled by a set of three coupled differential equations:

$$\frac{\mathrm{d}f_{\mathrm{a}}}{\mathrm{d}t} = -\frac{f_{\mathrm{a}}}{\tau_{\mathrm{i}}} + u^{+} f_{\mathrm{r}}^{-} \delta\left(t - t_j^{\mathrm{f}}\right), \tag{2.3}$$

$$\frac{\mathrm{d}f_{\mathrm{r}}}{\mathrm{d}t} = \frac{f_{\mathrm{i}}}{\tau_{\mathrm{r}}} - u^{+} f_{\mathrm{r}}^{-} \delta\left(t - t_j^{\mathrm{f}}\right), \text{ and} \tag{2.4}$$

$$\frac{\mathrm{d}f_{\mathrm{i}}}{\mathrm{d}t} = \frac{f_{\mathrm{a}}}{\tau_{\mathrm{i}}} - \frac{f_{\mathrm{i}}}{\tau_{\mathrm{r}}}, \tag{2.5}$$

describing the utilization of synaptic resources by each presynaptic spike arriving at time t_j^{f}. Here, $f_{\mathrm{a}}(t)$, $f_{\mathrm{r}}(t)$, and $f_{\mathrm{i}}(t) = 1 - f_{\mathrm{r}}(t) - f_{\mathrm{a}}(t)$ denote the fractions of active, recovered, and inactive synaptic resources, respectively, and $\delta(\cdot)$ the Dirac delta function. The superscripts "$-$" and "$+$" refer to the values of the associated variables before and after their update. Each presynaptic spike increases the active synaptic resources, i.e., the synaptic weight, and simultaneously reduces the available (recovered) resources by an amount proportional to u^{+}. The utilization variable u^{+} represents the probability of vesicle release, controlled by the calcium concentration in the axon terminal. In the absence of STF, this utilization is constant. However, in facilitating synapses, it is dynamic and evolves according to

$$\frac{\mathrm{d}u}{\mathrm{d}t} = -\frac{u}{\tau_{\mathrm{fac}}} + U\left(1 - u^{-}\right)\delta\left(t - t_j^{\mathrm{f}}\right), \tag{2.6}$$

where the parameter U corresponds to the amplitude of the postsynaptic current (the synaptic weight) in response to a single isolated presynaptic spike. The time constants τ_{r}, τ_i, and τ_{fac} describe the recovery time from synaptic depression, the decay of the postsynaptic currents, and the decay of the utilization, respectively. Despite its simplicity, this phenomenological model approximates experimental findings well [79].

The modifications of the synaptic efficacy by STP occur only during presynaptic firing and last for a few hundred milliseconds. After presynaptic firing has stopped, the synaptic resource variables, and hence the synaptic weight, return to their resting states $f_{\mathrm{a}} = 0$, $f_{\mathrm{r}} = 1$, $f_{\mathrm{i}} = 0$, $u = 0$, and w_{jk}. In contrast, *spike-timing-dependent plasticity (STDP)* has a prolonged effect on synaptic efficacy and thus constitutes a form of long-term plasticity. This form of plasticity was discovered in several spike pairing experiments where a pre- and a postsynaptic neuron were repetitively stimulated to emit spikes at a predefined interval [83, 84]. Reviews of the experimental findings can be found in Caporale et al. [85] and Markram et al. [86] and Brzosko et al. [87]. Although the results of these studies vary across cell types and pairing protocols, they all find that the induced change of the synaptic efficacy depends on the precise time difference

$\Delta t = t_{\text{post}} - t_{\text{pre}}$ between a pair of pre- and postsynaptic spikes. The mapping between the sign and value of the weight changes and the time lags between pre- and postsynaptic spikes is described by the *STDP kernel*, which can take either a Hebbian or an anti-Hebbian form and is sometimes modeled as either symmetric or anti-symmetric [all forms illustrated by Fig. 2 in 88]. In nature, however, the size and shape of the potentiation and depression windows might differ, leading to an overall asymmetric window (see Figure 2.4a). Generally, a postsynaptic spike occurring slightly after the presynaptic spike ($\Delta t > 0$) induces *long-term potentiation (LTP)*, whereas a postsynaptic spike occurring slightly before the presynaptic spike ($\Delta t < 0$) induces *long-term depression (LTD)*. Thus, STDP can encode a causal relationship between the firing of the pre- and postsynaptic neuron and the synaptic weight change. Since this finding follows Hebb's principle, this type of STDP belongs to the class of Hebbian plasticity rules.

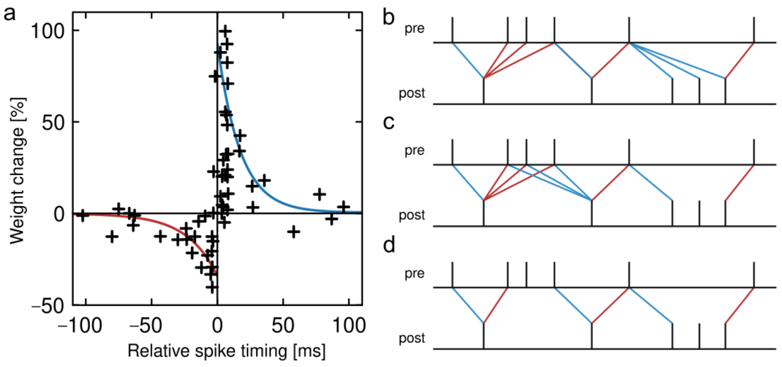


Figure 2.4: **Spike-timing-dependent plasticity (STDP). (a)** Weight change expressed as a function of relative pre- and postsynaptic spike timing for an example of STDP with an anti-symmetric (and slightly asymmetric), Hebbian learning window. Markers correspond to empirical data from Bi et al. [84]. Solid lines show exponential approximations used in the model (red indicates depression and blue indicates potentiation in all panels). An anti-Hebbian window would look similar but mirrored about the vertical axis. **(b-d)** Three nearest-neighbor spike pairing rule variants for STDP. **(b)** *Symmetric:* each presynaptic spike is paired with the last postsynaptic spike, and each postsynaptic spike is paired with the last presynaptic spike. **(c)** *Presynaptic centered:* each presynaptic spike is paired with the last postsynaptic spike and the next postsynaptic spike. **(d)** *Reduced symmetric:* as in panel **(c)**, but only for closest pairs. Adapted from Fig. 7 in Morrison et al. [77].

Formalizing this robust finding based on the above experimental data allows for mathematical treatment. Morrison et al. [77] developed a phenomenological model with only a few free parameters, which reproduces the experimental observations without referencing the underlying molecular mechanisms. They restricted the observables that can enter the plasticity rule to locally available ones because biologically plausible phenomenological models should only contain terms that are identifiable with mechanisms that exist in biology. In the case of STDP, the synaptic weight change depends on the pre- and postsynaptic spike times and potentially also on the current synaptic weight. Experiments demonstrating that action potentials back-propagating through the dendritic tree convey information about a postsynaptic spike support the fact that postsynaptic spikes can be available at the synapse [83].

The dependence of LTP and LTD on Δt is usually captured by exponential functions with decay times $\tau_\pm$. Morrison et al. [77] give a simple model of the weight change Δw for pair-based STDP:

$$\begin{aligned} \Delta w_+ &= F_+(w)\, e^{-\frac{|\Delta t|}{\tau_+}} \qquad \text{if } \Delta t > 0, \\ \Delta w_- &= -F_-(w)\, e^{-\frac{|\Delta t|}{\tau_-}} \qquad \text{if } \Delta t \leq 0, \end{aligned} \tag{2.7}$$

where the functions $F_\pm$ capture the dependence on the current weight w and have to be specified further by fitting them to experimental data [89, 90] (Figure 2.4a). A spike pair can be defined in different ways: for example, each presynaptic spike can be paired with the most recent preceding postsynaptic spike and vice versa, which is one variant of the class of *nearest-neighbor schemes* (see Figure 2.4b to Figure 2.4d), whereas in the *all-to-all scheme*, each presynaptic spike is paired with all preceding postsynaptic spikes [77, 91].

The STDP model described above can serve as a starting point for designing extended models to describe more nuanced experimental results, for example by including the postsynaptic membrane potential as an additional modulatory factor beyond the pre- and postsynaptic spikes. Along these lines, Clopath et al. [92] and Clopath et al. [93] account for the effects of voltage-dependent receptors and channels. Their approach is based, among other findings, on experiments showing that the same spike pairing protocol can, depending on the postsynaptic membrane potential V_m, induce no change in synaptic weights at all, LTD, or LTP [94]. While a V_m smaller than an experimentally determined threshold potential Θ_- induces neither LTD nor LTP, an intermediate membrane potential $\Theta_- < V_\mathrm{m} < \Theta_+$ triggers LTD, and a high $V_\mathrm{m} > \Theta_+$ enables LTP. To capture this behavior, the mathematical description of the plasticity rule contains terms for facilitation and depression that are active based on these conditions of the membrane voltage, formally expressed as Heaviside step functions. With this mechanism, Clopath et al. [92] were able to reproduce the complex frequency dependence of the synaptic weight changes in spike pairing experiments [95].

In Urbanczik et al. [96], the postsynaptic membrane potential is included as a modulating factor. This plasticity rule, in particular, applies to synapses that connect to the dendrite of a postsynaptic neuron. Experiments show that presynaptic spikes that do not cause postsynaptic spikes lead to a depression of synaptic weights whose strength increases with increasing dendritic voltage [97]. From this observation, Urbanczik et al. [96] conclude that the synaptic weights are adjusted such that the dendritic voltage assumes high values if and only if the soma of the postsynaptic neuron emits spikes. Therefore, in this rule, the difference between the dendritic voltage and the somatic activity drives the synaptic weight change.

Instead of the postsynaptic membrane potential, a third factor could also be a neuromodulator concentration, which is motivated by experimental studies [for a review, see 98] and by the fact that they provide a biologically plausible implementation of reward signals [99].

From mathematical models to simulation

The STP implementation in NEST[4][5] exploits several practical properties of the corresponding differential equations for their numerical integration. Since the synaptic resources are conserved (i.e., the fractions f_r, f_a, and f_i add up to 1), f_a can be eliminated from the system. Furthermore, thanks to its linear form, the system of coupled differential equations Equation 2.4, Equation 2.3 and Equation 2.5 can be integrated exactly between two consecutive presynaptic spikes [49]. Concretely, the joint state of u, f_r, and f_a can be iteratively evolved by multiplying the state at the previous presynaptic spike with a *propagator matrix*.

To simulate STDP, Equation 2.7 needs to be calculated efficiently [77]. Having restricted the model parameters to those locally available at the synapse facilitates the implementation in software[6]. These constraints also improve the model's performance, since network simulators running on distributed systems take advantage of a limited need for global access to variables to reduce memory consumption and high-latency communication between compute nodes [100, 101]. The all-to-all pairing scheme can be efficiently implemented using a specific update scheme of the synaptic traces. These traces represent a fading memory of past spikes at the synapse without explicit knowledge of all past spike times [89, 77]. If a pre- or postsynaptic spike occurs, the corresponding trace and synaptic weight are updated, while no actions need to be performed in the periods in between. Defining the exact order of updates in a plasticity model, particularly with regard to pre- and postsynaptic spike timing, indicated by the "+" and "-" in Equation 2.3, Equation 2.4, and Equation 2.5 is crucial and facilitated by the high-level language NESTML[7].

More complex learning scenarios, like reinforcement learning, are made possible by advanced plasticity rules, which, for example, depend on the postsynaptic membrane potential or neuromodulators [102]. However, these rules typically make it more difficult to discover an effective implementation. For example,

4 https://nest-simulator.readthedocs.io/en/latest/auto_examples/tsodyks_depressing.html

5 https://nest-simulator.readthedocs.io/en/latest/auto_examples/tsodyks_facilitating.html

6 https://nestml.readthedocs.io/en/latest/models_library/stdp.html

7 https://nestml.readthedocs.io/en/latest/nestml_language/synapses_in_nestml.html

neuromodulators (e.g., dopaminergic signals) affect several nearby synapses through *volume transmission* requiring a notion of physical 3D space (see Section 2.2.2). Moreover, the presence of time-continuous signals in some of these advanced plasticity rules necessitates the storage of the signal history if one wishes to keep the efficient *event-driven* scheme of updating the synaptic weights only at presynaptic spike times [100]. Depending on how the data structures are laid out in a simulator, accessing the continuous third-factor variables can be difficult or computationally costly because they need to be queried for every spike at every synapse. Generally, rules that require only spike times are more efficient in memory and compute time than rules that depend on the entire history of variables, like the membrane potential trace.

Given access to the synaptic weights, another form of functional plasticity, *weight normalization*, can be realized. It entails keeping the total sum, or a norm of all incoming synaptic strengths of a neuron constant by re-normalizing all its synaptic weights[8]. Since, in the brain, these weight changes happen on timescales of several hundreds of milliseconds, the iterative re-normalization takes place on a coarse time grid, increasing the operation's efficiency.

Other advanced plasticity rules include, for example, a third-factor postsynaptic dendritic current [96] or inhibitory plasticity [103]. The discovery of new plasticity rules can be, to a certain degree, even automated [104]. Furthermore, state models with *synaptic tagging and capture* (STC), described, e.g., in Barrett et al. [105], incorporate even plasticity effects beyond synapse-specific ones. Ultimately, state-of-the-art computational plasticity models transcend the simple STP and STDP models [see, e.g., 106]. Algorithmically, however, these complicated models often use a combination of plasticity mechanisms and thus can be synthesized from such a base stack of simpler models.

2.2.5 *Heterogeneity*

Complexity and heterogeneity are ubiquitous and well-established design principles in neurobiological systems [107], covering a multitude of components and mechanisms at various spatial and temporal scales. From an information processing perspective, such variability is a fundamental component of the system, as it determines the types of computations a given circuit can perform and constrains the representational expressivity of its dynamics [108].

From empirical data to mathematical models

Biological synaptic connectivity is highly diverse in most of its constituent properties, including the type of neurotransmitter used, the composition of presynaptic vesicles and docking proteins (affecting release probability), the postsynaptic receptor composition (affecting efficacy and kinetics of the elicited

8 https://nest-simulator.readthedocs.io/en/latest/guides/weight_normalization.html

response), transmitter re-uptake and re-use, and the involvement of gliotransmission [see, e.g., 109], but also properties characterizing signal propagation such as axon diameter and conductance velocity [110–112]. These various types of diversity translate to a high degree of heterogeneity in phenomenological parameters characterizing mathematical models of the synaptic connectivity and dynamics, such as the synaptic weight [113–117], synaptic time constants [118, 119], response latencies (synaptic delays; Brunel et al. [120] and Roxin et al. [119]), and parameters specifying the plasticity dynamics [121]. In addition, biological neuronal networks exhibit a high degree of heterogeneity in the anatomical connectivity structure, such as the total number of inputs and outputs per neuron (in/out-degrees; Markram et al. [83], Feldmeyer et al. [122–124], Stepanyants et al. [125], and Roxin [126]), and the composition of presynaptic source and postsynaptic target neuron populations.

Previous theoretical work on recurrent neuronal networks shows that heterogeneity in single-neuron properties or connectivity broadens the distribution of firing rates [127, 119] and affects the stability of asynchronous or oscillatory states as well as the level of synchrony [128, 129, 120, 130, 131, 126, 132, 133]. A large number of theoretical and experimental studies point at the benefit of heterogeneity for the information processing capabilities of neuronal networks [134–140, 132, 141–143, 108]. Therefore, modeling studies aiming at understanding the dynamical and functional principles of biological neuronal networks need to account for the synaptic (and other types of) heterogeneity.

Depending on the type of synaptic heterogeneity, its implementation in mathematical models may follow different strategies. Synaptic heterogeneity is expressed on local scales, such as in the connections between neurons in a given layer of a cortical column, and on large scales, such as in cortical inter-area connections. One form of this heterogeneity results from cell-type, layer, or area specificity. It reflects the anatomical and electrophysiological diversity of neurons in different brain regions and, in addition, emerges from specific interactions with other components of the nervous system or with the environment during brain development and learning. In mathematical models, this specificity is usually accounted for by subdividing the network into several populations representing different cell types or brain regions and applying distinct connectivity, synapse, and plasticity parameters for each pair of populations (Figure 2.5a).

Another form of synaptic heterogeneity appears in an unspecific, quasi-random manner. It refers to variations in the synaptic characteristics across an ensemble of neuron pairs of seemingly identical type, for example, connections between a group of neurons with similar morphological and electrophysiological characteristics located in the same layer of a given cortical column (Figure 2.5b). Similarly to the cell-type-, layer-, or area-specific diversity described above, the unspecific forms of heterogeneity are partly caused by synaptic plasticity, i.e., by adapting synaptic parameters during learning and development. In this respect, unspecific heterogeneity is not truly unspecific; it is, on the contrary, the result of fine-tuning, optimization, or specialization. Without knowing the details of

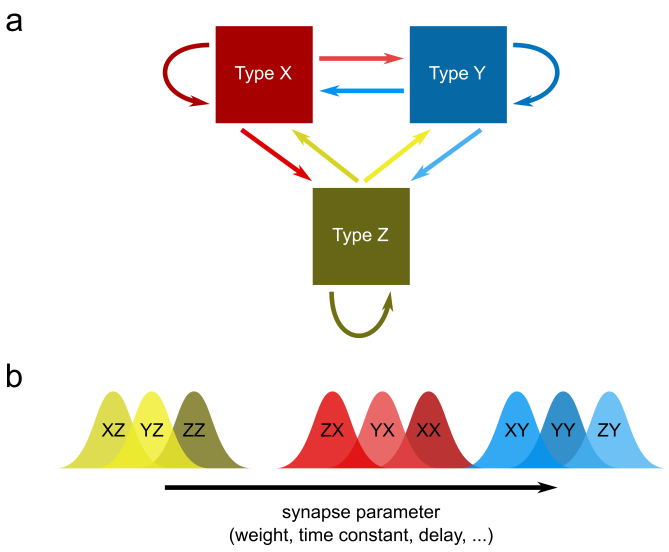


Figure 2.5: **Specific and unspecific heterogeneity in synaptic connectivity. (a)** Sketch of a neuronal network comprising three populations of neurons of type X, Y, and Z (boxes). The properties of the different projections (arrows), such as the number of synapses, the synaptic weights, synaptic time constants, or synaptic delays, depend on the types of pre- and postsynaptic neurons. We refer to the resulting synapse-type specific diversity as *specific heterogeneity*. **(b)** For each type of projection $\{PQ\}$ from population Q to population P ($P, Q \in \{X, Y, Z\}$), the synaptic parameters are distributed (illustrated here with bell-shaped curves). We refer to this form of variability as *unspecific heterogeneity*. The parameters characterizing each distribution, such as the mean (horizontal position of each curve) or the variance, are usually synapse-type specific.

these processes, the resulting diversity appears random or unspecific. Moreover, the distinction between type-specific and unspecific forms of heterogeneity relies on the assumption that different neuron types are distinguishable [144]. Without knowing the characteristics separating two neuronal phenotypes, these cell classes are treated as one type, and the observed diversity in neuron and synapse parameters appears unspecific.

To some extent, synaptic heterogeneity may also result from variations in experimental protocols and in unobserved variables affecting the synapse characteristics. Synaptic weights, for example, are often assessed in voltage-clamp experiments as the amplitudes of somatic postsynaptic currents evoked by presynaptic action potentials. The resulting synaptic weights are then determined not just by the properties of the pre- and postsynaptic cells or by the synapse type and position but also by the holding potential or the electrical characteristics of the electrode-cell contact. Even in the absence of variations in the experimental protocol, the amplitude of the postsynaptic response is affected by fluctuations in the postsynaptic membrane potential and by the pre- and postsynaptic spike history. Hidden variables such as the spike history or the synapse position are often not monitored in experimental studies. From the modeler's perspective, it is therefore not straightforward to decide what forms of reported heterogeneity should be accounted for in a given model and what forms are perhaps already represented indirectly by other model features (for example, the voltage dependence of synaptic currents, short-term plasticity, or dendritic filtering in multicompartment models).

In mathematical models, unspecific heterogeneity is typically accounted for in a probabilistic manner. Here, the parameters characterizing synaptic connectivity, such as synaptic weights, time constants, delays, in-degrees, etc., are randomly drawn from certain distributions. In particular, in the brain, many properties follow long-tailed distributions, often approximating the lognormal distribution [145–147]. These distributions, or the parameters characterizing them, such as the mean or the standard deviation, are extracted from experimental data. The rationale underlying this probabilistic modeling approach is twofold. First, it acknowledges that the synaptic parameters are typically not known for every single synapse in a given network. The majority of experimental studies provide data for small subsets of synapses, often pooled across different recording sessions or animals. Second, the probabilistic approach greatly simplifies the models, as the total number of parameters is substantially reduced. In probabilistic modeling approaches, the "model" is not defined by a single instantiation of a network and all its parameters but by the ensemble of many independent realizations generated from a given set of parameter distributions. Observations or findings obtained from a single network realization are meaningless unless they appear generically, i.e., frequently, for many different model realizations.

As described above, synaptic heterogeneity is often the result of an adaptation, development, or fine-tuning process. As demonstrated in a number of studies (see, e.g., Prinz et al. [148], Achard et al. [149], and Bahuguna et al. [150]), such processes typically lead to dependencies between parameters. Certain plasticity processes, for example, lead to a competition (anti-correlation) between synapses, such that the strengthening of one synapse results in the weakening of another synapse [151, 152]. In a comprehensive probabilistic model, the set of parameters $\{\xi_1, \xi_2, \ldots, \xi_n\}$ for a specific network realization is generated according to a joint probability density function (pdf) $p_{1,\ldots,n}(x_1, x_2, \ldots, x_n)$, describing the probability of observing $\xi_1 = x_1$, $\xi_2 = x_2$, ..., and $\xi_n = x_n$. This joint pdf captures all parameter dependencies. Many modeling studies neglect parameter dependencies and assume that the joint pdf $p_{1,\ldots,n}(x_1, x_2, \ldots, x_n) = p_1(x_1)p_2(x_2)\ldots p_n(x_n)$ factorizes. In these studies, each parameter ξ_i is drawn from its respective marginal distribution $p_i(\cdot)$, independently of all other parameters. As before, this simplifying assumption typically reflects a lack of knowledge, as the available experimental data generally do not capture parameter dependencies. Theoretical studies show that this choice can have detrimental consequences for the dynamical and functional properties of the resulting system. Bahuguna et al. [150], for example, demonstrate that when the parameter dependencies are unknown, replacing all parameters by their respective mean (and thereby ignoring diversity) can be a better choice than drawing them from their marginal distributions.

A more direct approach toward modeling synaptic heterogeneity is the explicit account of known plasticity, learning, or developmental processes that dynamically lead to the observed diversity in synaptic parameters, including the dependencies described above [89, 152]. Similarly, multivariate parameter distributions may arise from optimization procedures or supervised learning methods fitting the model to some desired target dynamics or behavior [153, 150, 154]. While these top-down approaches are promising and commonly used in state-of-the-art computational neuroscience, they bear the risk that the underlying data or targets do not sufficiently constrain the model of the actual biological system and hence lead to a multitude of solutions that may not be realized in nature. A combination of bottom-up and top-down constraints appears to be the most reliable method to reduce this form of uncertainty.

From mathematical models to simulation

Investigating the role of heterogeneity in synaptic connectivity by means of analytical mathematical methods is challenging [120, 119]. Therefore, theoretical studies often neglect heterogeneity to simplify the mathematical treatment and provide intuitive insight. A common strategy underlying many mathematical approaches is to reduce the dimensionality of the neuronal network dynamics by assuming that the network can be decomposed into homogeneous subpopulations, each of which comprises neurons with identical neuronal and synaptic parameters. While this approach can account for the specific heterogeneity described above to some extent, it can hardly describe the effects of unspecific heterogeneity. Simulation enables us to test whether the insights obtained under these homogeneity assumptions remain valid if heterogeneity in synaptic parameters is considered.

Even in simulation studies, however, accounting for synaptic heterogeneity is challenging. Acknowledging that every synapse is unique requires representing each synapse with the full individual set of parameters. In a homogeneous network where all synapses have identical properties, the connectivity is fully described by the adjacency matrix (which neuron is connected to which, and how often) and a small set of parameters describing the synapse characteristics, such as the synaptic weight w, the delay d, or the synaptic time constant τ. In a heterogeneous network where each synapse $\{j \rightarrow i\}$ is unique, in contrast, the individual weights w_{ij}, delays d_{ij}, and time constants τ_{ij} need to be stored for each connection. Therefore, representing the heterogeneous connectivity in simulations of neuronal networks at natural density imposes high memory demands for the underlying computing architecture (see Section 2.3.7).

In models of neuronal networks with heterogeneous synaptic connectivity, the heterogeneity is either implemented by drawing synapse parameters from predefined distributions or by a self-organization process driven by some plasticity or learning dynamics (see Section 2.2.5). Simulations based on the first, the probabilistic approach, require efficient methods of drawing random numbers from specified distributions during the network generation phase. The NEST simulator, for example, permits the high-level specification of probabilistic connection rules by the user (see Section 2.2.1), including distributions of synaptic weights, synaptic delays, or plasticity parameters. The task of generating a specific connectivity realization by drawing random numbers from these distributions is then delegated to fast low-level (C++) routines. The second approach relies on simulating plastic networks or on numerical optimization methods. Strategies for simulating different forms of synaptic plasticity are described in Sections Section 2.2.3 and Section 2.2.4. Simulating plastic networks with natural connection density is still a major challenge in computational neuroscience. Slow biological processes such as learning and development on timescales of hours, days, and years are presently inaccessible to simulation (or restricted to small and highly simplified models) because of the required wall-clock time. In this respect, dedicated neuromorphic computing architectures are particularly interesting as simulation platforms for neuroscience, as they offer the potential for faster-than-real-time simulations and hence, for an understanding of plasticity mechanisms on long timescales [155, 156].

2.3 INSIGHTS IN MODELING AND SIMULATION PRACTICE

In this section, we highlight some of the challenges and pitfalls that may be encountered during modeling and simulation.

2.3.1 *Biologically plausible model refinements*

Developing a mathematical model that exhibits a dynamic behavior close to empirical biological data involves iterative optimization procedures, e.g., fitting the experimental data or calibrating the model parameters. While this optimization improves the model in some aspects, it may, at the same time, alter it in ways no longer motivated by biological observations [149]. For example, fitting is inherently biased in that, by definition, it improves the validation of certain model features at the cost of those not included in the fitting procedure. Thus, each optimization step should be conducted cautiously and checked for biological plausibility.

2.3.2 *Reproducible simulations*

For small-scale simulations, sometimes custom simulation kernels are written. However, besides possibly duplicating published and established routines, these self-made frameworks are likely to contain bugs and lack documentation, for instance, on edge-case behavior. Therefore, even for small networks, it helps to use standardized simulators. In particular, these simulators offer the benefit of being well characterized under different operating conditions using a diverse array of automated tests, being updated on a regular release cycle, and benefiting from the open-source model of iterative refinement [157]. All these factors increase the likelihood of long-term reproducible results.

An individual simulator should exhibit *replicable* behavior: repeated simulations of the same model should yield bitwise identical results, regardless of the number of threads or processing nodes used, due to the use of deterministic pseudo-random number generators. However, simulating the same model on a different platform or using a different numerical solver or time step size for ordinary differential equations (ODEs) may alter the results, especially in network models exhibiting chaotic and unstable dynamics. Nonetheless, results and conclusions should be *reproducible*, obtaining the same overall quantitative and qualitative conclusions [for a commentary on this terminology, see 158]. Reproducibility of results requires the original software to be available (including libraries and other dependencies) and, where applicable, the original (raw or pre-processed) dataset/s and relevant metadata.

Similar to the model descriptions, it increases the reproducibility of methods and results [159] to use and contribute to existing simulation frameworks by reporting bugs, improving implemented methods, and developing and publishing custom modules of the respective framework, e.g., in NEST, in the form of *extension modules*[9].

9 https://nest-extension-module.readthedocs.io/en/latest/extension_modules.html

2.3.3 *Distribution of compute workload*

It is beneficial to distribute the workload evenly across compute nodes, even for networks with complex connectivity and heterogeneous population properties. One way to achieve this is a *round-robin* distribution of neurons across compute nodes, i.e., in the case of M compute nodes, assigning neuron n to node ($n \bmod M$).

In general, the computing system's size affects the workload distribution. On small machines, the number of synapses per neuron is larger than the number of compute nodes the simulation runs on. Hence, each neuron typically has many targets on every compute node. However, with growing network size and the emergence of new supercomputer architectures over the last decades, the ratio between the numbers of synapses per neuron and compute nodes is in some cases reversed. On these new-generation supercomputers, the distribution of neurons over many nodes decreases the chance that a neuron shares a node with a connected partner, especially considering the sparsity of biological neuronal networks. Mitigating this issue, even more modern compute nodes follow the opposite trend: they possess more memory and cores per processor and thus more processing power, which reduces the number of nodes required for the simulation and brings the neurons and their targets closer together.

One of the main computational challenges remains the connectivity of neuronal networks. For example, representing each of the estimated 10^{14} synaptic connections in the brain individually by two double-precision numbers requires about 1.6 PB of main memory. Furthermore, neurons form connections with nerve cells not only in their vicinity but also in various remote areas. This feature distinguishes neuronal simulations from simulations of classical physical systems, that use, for example, finite-element methods exploiting the locality of physical interactions. However, memory and communication bandwidth, as well as cache efficiency, are more critical than floating-point performance of spike communication to local and distant targets [160].

Future work should address how computer network connectivity and the simulation distribution across compute nodes can follow (simulated) biological network organization, such as a modular organization on both small and large scales in the brain.

2.3.4 *Scalability in theory and practice*

Recording the simulation time under varying network or computing system sizes characterizes an implementation's *scalability*, which is essential to judging its efficiency [39]. The scenario of increasing the computing resources while keeping the network size fixed is called *strong scaling*, and the scenario of increasing the network size and computing resources proportionally is called *weak scaling*.

Ideally, when in strong scaling the hardware becomes twice as powerful, the simulation time is divided by two. However, in practice, the gains are usually less due to communication overhead and other bottlenecks in the system. Sometimes, a parallelization can even exhibit a super-linear speed-up [161], but this behavior might only become apparent at very large computing system scales.

Scaling up standard network models to test weak scaling can induce unrealistic activity patterns, for example, regarding the regularity and synchrony of spiking. Since synapses tend to vastly outnumber neurons, the number of synapses is an important determinant of the necessary computing resources. Thus, a reasonable approach is to keep the in-degrees constant when increasing the model size and thereby reduce the overall connection probability [see, e.g., 39]. This method tends to lessen activity correlations between neurons and hence diminish synchrony. In the case of ideal weak scaling, a network of twice the size should run for twice the time, but in practice, the performance is worse due to the same reasons as for strong scaling. Moreover, there is a complex dependence of the scaling behavior on network properties, such as the connectivity's modularity [160].

2.3.5 *Precise spike times in discrete-time simulation*

A typical simulation of a continuous-time dynamical process runs in discrete steps of time Δt. However, exchanging events (spikes) on a grid can cause synchronization in the network as a pure simulation artifact. This effect disappears in the limit of $\Delta t \to 0$, but decreasing the time step increases the time necessary for the simulation to complete, so a tradeoff has to be made. A more computationally efficient solution is to store an extra offset value in spike events, which, in combination with a minimum synaptic delay and an algorithm that finds the precise time of spiking, decouples the simulation time step from the temporal precision with which spikes are exchanged [162].

2.3.6 *Simulating until convergence*

Simulating neuroplasticity for too short of a period, especially longer-timescale processes like normalization and neuromodulation, is a common pitfall. Several of these dynamic processes have the potential to cause an abrupt bifurcation in the system late in the simulation. Additionally, some plasticity rules produce a long-tailed distribution of synaptic strengths [163], whereby the distribution reaches equilibrium again only after an extended simulation period. A limited measurement duration can also be a problem in empirical neuroscience, but simulations can, in principle, run as long as desired. The only limitations are the computing resources available (as these need to be shared with other users on high-performance computing systems) and the amount of time the simulation takes to complete, which depends on the simulator's efficiency. Simulating until the measure of learning performance has adequately converged can circumvent this pitfall to some degree.

2.3.7 *Limited synaptic weight resolution*

Limiting the numerical resolution of synaptic parameters, such as the synaptic weight, appears to be an obvious strategy to reduce memory and compute load. To some extent, nature itself copes well with quantized synaptic weights: Transmission in chemical synapses is quantized due to the release of neurotransmitters in discrete packages from vesicles in the presynaptic axon terminals. The analysis of spontaneous (miniature) postsynaptic currents, i.e., postsynaptic responses to the release of neurotransmitters from single presynaptic vesicles, reveals that the resolution of synaptic weights is indeed finite for chemical synapses. As shown by Malkin et al. [164], the amplitudes of spontaneous excitatory postsynaptic currents recorded from different types of excitatory and inhibitory cortical neurons follow a unimodal distribution with a peak at about 20 pA and a lower bound at about 10 pA. Such a cut-off is present despite several factors that may wash out the discreteness of the synaptic transmission, such as variability in vesicle sizes, variability in the position of vesicle fusion zones, quasi-randomness in neurotransmitter diffusion across the synaptic cleft, and variability in postsynaptic receptor densities. However, the discreteness of synaptic strengths is obscured for evoked synaptic responses involving neurotransmitter release from many presynaptic vesicles and for superpositions of inputs from many synapses, and thus unlikely to play a particular role in the dynamics of the neuronal network as a whole.

Inspired by these observations, Dasbach et al. [165] systematically investigated the effects of a limited synaptic weight resolution on the dynamics of recurrent spiking neuronal networks resembling local cortical circuits. They show that a naive quantization of synaptic weights generally leads to a distortion of the firing statistics. However, in the example of one network type, they could demonstrate that the firing statistics remain unaffected under a weight discretization that preserves the mean and variance of the total synaptic input currents. In networks with sufficiently heterogeneous in-degrees, the firing statistics stay constant, even when replacing all synaptic weights with the mean of the weight distribution, i.e., entirely neglecting the unspecific form of heterogeneity in synaptic weights. Applying this finding in simulations reduces the memory demands substantially. The effect of discretized synaptic weights in networks undergoing different forms of synaptic plasticity has rarely been investigated [166] and remains a subject for future study.

2.3.8 *Precise specification of the model*

An incomplete and ambiguous description of a model without following conventions can impede understanding by the reader and thus the reproducibility of the respective study. The first step to avoiding this pitfall is to gain a clear picture of the requirements a model description must fulfill. For this, Nordlie et al. [167] propose that a neuronal network model description must contain a complete and detailed account of its architecture and the dynamics of its parts. In the formalization of model descriptions, computational neuroscience is less mature than other fields of science. Nonetheless, best practice guidelines are emerging that suggest the use of standardized tables of the model characteristics that cover the network architecture and connectivity, all neuron and synapse models used, the applied input stimuli, and the recorded data, described by a combination of text, equations, figures, subtables, and pseudocode [167]. Formalized sketches of the network and unified connectivity concepts [28] help computational neuroscientists to unequivocally convey their models and, consequently, readers to understand them.

It is advisable to use one of the numerous formal languages that facilitate such specifications and make them consistent, e.g., NeuroML [16], NESTML [9], or PyNN [15]. Most of such languages adhere to the class of either declarative or procedural languages. While a declarative language specifies the model's features, a procedural language specifies the series of commands or instructions needed for constructing the model. Best practices for either of these approaches include formally defined syntax and semantics or an API specification, both uniquely identified by version numbers.

It is good practice to keep the specification and implementation of a model separate. For example, implementation details such as the time resolution and the spike threshold detection method are essential for the reproducibility of the results but are not part of the model itself. Like the model description, the implementation specification should be complete and sufficient, as it can be challenging to reverse-engineer implementations and test the robustness of the results to implementation alterations [167]. However, the model should be robust to different choices in such implementation details.

Generating an executable representation of the model can be fully automated. For example, tools like NESTML specialize in processing the model descriptions of different complexity levels and in verbose formats like XML, creating executable implementations and visualizations, and facilitating debugging [168].

To summarize, it is advisable to follow standardized model descriptions, implementations, and generation procedures wherever possible for a project and ideally share the models with the community in dedicated databases for computational neuroscience models like ModelDB [169] or Open Source Brain [170].

2.4 DISCUSSION

Large-scale neuronal network simulations are a key tool for understanding brain processes. They make the complex nonlinear dynamics of neuronal activity, which are out of reach with analytic methods, accessible to inquiry. Moreover, continuous interaction between computational modeling and advances in empirical understanding have iteratively refined simulation approaches throughout the history of computational neuroscience. We hence outline a few promising directions for empirical and modeling approaches to work together.

New empirical data can bolster simulation studies in many ways. For example, long-term tracking of synapses *in vivo* will elucidate the relationship between synaptic plasticity and function [171]. Besides, recordings of complete connectomes at single-neuron resolution are becoming feasible for ever-larger brains and may soon be available on the scale of a mouse brain [172]. Data-driven models based on such detailed and specific connectivity are complemented by models whose connectivity is generated with statistical approaches. The latter models also profit from more available data constraining their parameters; higher-level network organization can, for instance, be informed by data on hierarchical modularity and small-world properties. Sometimes, the conditions of data retrieval are inconsistent between experiments, or experiments only cover a small subset of the model system, and modeling could benefit from additional studies to fill the gaps.

How the available data are integrated into models depends on the particular research question and the level of abstraction appropriate for it. The modeler needs to decide (or find out) which features and phenomena of the natural system need to be represented in detail and which ones can be approximated. For instance, a biophysically detailed model with discrete vesicle release dynamics would be suitable when investigating how a compound influencing vesicle fusion to the membrane affects synaptic transmission. In contrast, investigating the compound's effects on large-scale network dynamics could necessitate approximation of the vesicle release by a simplified set of continuous quantities and differential equations. In essence, a good computational model should represent the relevant attributes of the studied biological structure, have explanatory power and simultaneously not necessitate extensive simulation time.

There are several ways in which the infrastructure of computer simulations itself can innovate, indicated by current trends and feature requests from the community. Simulation efficiency is still a bottleneck, especially in simulations involving plasticity, as they need to run for a comparatively long time. Furthermore, large-scale networks require powerful, high-performance compute clusters, which provide large amounts of RAM, and enable running the simulation in parallel and distributed across many CPUs or GPUs which are interconnected through a low-latency network. More advanced synaptic plasticity rules, for example involving tripartite synapses influenced by astrocytes or neuromodulators like dopamine, still lack software support for efficient simulations on a larger scale. Finally, simulations at cellular resolution could be extended toward multiphysics modeling by incorporating other physical phenomena. Such models could account for the volume diffusion of neuromodulators or the electric field in the neuropil to simulate ephaptic coupling for instance.

In general, what should one strive for in a model? The statistician George Box famously said "All models are wrong, but some are useful". Although it helps to remind ourselves of the difficulties of modeling, to say that all models are wrong is not doing them justice, as put by Sir David R. Cox in a comment on Chatfield [173]: "The very word *model* implies simplification and idealization. The idea that complex physical, biological or sociological systems can be exactly described by a few formulae is patently absurd. The construction of idealized representations that capture important stable aspects of such systems is, however, a vital part of general scientific analysis and statistical models [. . .]". A model with unlimited parameters can fit the data perfectly but at the cost of generalizability and explanatory power. Ockham's razor, or the law of parsimony, provides a helpful heuristic in this context, guiding us to prefer the theory with fewer parameters between two competing theories. This objective is also formalized in Bayesian information criteria, which penalize models with larger numbers of parameters. In other words, the aim to reduce complexity should guide modeling choices to address a given research question.

However, condensing the biophysical details and terminology to arrive at a parsimonious, phenomenological formulation could impede testing such a minimal model experimentally. Moreover, the rigorous application of Ockham's razor leads to models optimized for single phenomena (e.g., connectivity, synaptic transmission, structural or functional plasticity) that are thus hard to combine. To solve this problem, after reducing the phenomena to their essential variables, we should express each model in a way that allows for their combination. Biophysical details could provide contact points between model concepts from different subdomains. Progress in integrating heterogeneous phenomena in large-scale models requires that models act as platforms that can be modified and extended over time. For example, after a minimal model has achieved a satisfactory performance, modelers could test whether the same results still hold for models of greater biophysical detail.

As a foreseeable trend, more and more computational neuroscientists will adopt procedures ensuring the reproducibility of their methods and results over the following years. This approach includes, where possible, data, model, and code sharing in open-access online repositories, adherence to open standards for model formats and software tools, active management of metadata, containerized distribution of dependencies, unit testing, and continuous integration instead of creating new in-house toolchains from the ground up. Last but not least, active contribution to an existing model database or open-source software, be it as small as a feature request, is a low-threshold action conducive to reproducible research everyone can take. Ultimately, the community's research interests in the form of these requests and contributions shape the landscape of available models in an open-access simulator.

3 BIOLOGICALLY INSPIRED ENHANCEMENTS TO A SYNAPTIC PLASTICITY MODEL

Author Contributions

Event-driven eligibility propagation in large sparse networks: efficiency shaped by biological realism. A. Korcsak-Gorzo, J. A. Espinoza Valverde, J. Stapmanns, H. E. Plesser, D. Dahmen, M. Bolten, S. J. van Albada†, M. Diesmann† (2025) *arXiv preprint*. DOI: 10.48550/arXiv.2511.21674. Submitted to *Nature Computational Science*.

† Shared last authorship.

All parts of this chapter are excerpts from the above manuscript in preparation for submission. The author was the primary contributor, actively participating in all aspects of its development. MD and SJA supervised the author's work, MB supervised JAEV's work, and HEP provided technical oversight for the implementation.
The idea to adapt e-prop for large sparse networks arose from discussions between SJA and the developers of the original time-driven e-prop. The author, in collaboration with JS, created the first event-driven online e-prop. The author and JAEV jointly developed the biological extension of the model. DD contributed to the theoretical development of both models. The author and JAEV refined, tested, and finalized both models for integration into NEST, with HEP providing technical review and refinement. The N-MNIST task implementation was carried out by JAEV and refined by the author, who also collaborated with JAEV on parameter refinement. The author and JAEV collaboratively developed the simulation framework and jointly conceived and conducted the simulations. The author visualized the experiments with contributions from JAEV. JAEV designed the initial schematics, which the author recreated using an alternative program and further refined. The storyline was conceptualized by the author, MD, SJA, and JAEV, with input from HEP and DD. The author drafted the text, with JAEV contributing to the results and methods sections. The manuscript underwent collaborative review and editing by all authors.
Jakob Jordan and Alexander van Meegen implemented an early event-driven offline e-prop. Figure 3.3 is the result of joint work by the author and Charl Linssen, inspired by activities and feedback at the CapoCaccia Workshop toward Neuromorphic Intelligence 2023. The project also benefited from discussions with Franz Scherr on the original e-prop model, review comments

on the NEST models from Charl Linssen, feedback on the documentation from Jessica Mitchell, and technical assistance from Dennis Terhorst.

3.1 INTRODUCTION

Artificial intelligence has permeated and is revolutionizing many aspects of our lives. Examples include large language models in chatbots, computer vision in security cameras, and generative models for content creation. However, as the complexity and number of underlying neural networks and machine learning algorithms grow, so does their energy demand. This presents a challenge for computational science: optimizing these algorithms and networks for energy efficiency. The goal is to strike a balance between computational efficiency and accuracy. One promising approach to addressing this challenge is to draw inspiration from the human brain, which operates with remarkable energy efficiency. Emerging brain-inspired learning models not only deepen our understanding of neural processes but also offer new opportunities to develop more sustainable AI solutions [1].

One class of these learning models, supported by ample experimental evidence [174], are the so-called three-factor models. These models combine classical Hebbian learning, characterized by pre- and post-synaptic terms, with an additional modulatory signal [for a review, see 175, 76]. An example of this class is eligibility propagation (e-prop) [154], a biologically plausible plasticity rule for recurrent spiking neural networks (RSNNs) that approaches the performance of backpropagation through time (BPTT) [176]. The local, online-computable e-prop weight update rule was derived by calculating the gradients from the loss at the output layer and introducing approximations to address the biologically implausible aspects of BPTT, such as time blocking and symmetric feedback weights. This algorithm was originally implemented in TensorFlow [177] with time-driven weight updates, meaning the weight updates are calculated synchronously at each time step.

Current time-driven weight update methods face limitations, as substantial computational resources are often wasted due to the sparse nature of spiking activity. In biological brain networks, connectivity is sparse and neurons spike infrequently with inter-spike intervals much larger than the time scale of neuronal dynamics. An event-driven algorithm, where synaptic weight updates are computed asynchronously and only when an event occurs at the corresponding synapse, offers computational advantages by leveraging this sparsity. The reduced computations in this sparse setting, combined with event-driven updates, would enhance the scalability of the e-prop algorithm. This improvement could leverage the recent advances in exascale computing, enabling the simulation of large-scale networks. Such brain-scale simulations are of great interest to computational neuroscience for understanding learning processes and to machine learning for tackling tasks of higher complexity that require larger networks.

Drawing inspiration from nature and making these algorithms increasingly biologically plausible offers dual benefits. From a neuroscience perspective, it aids in understanding human learning. From a machine learning perspective, it provides insights that could lead to the development of more energy-efficient algorithms.

We present a computationally accurate, efficient, and scalable version of e-prop with event-driven weight updates and additional biologically inspired features, and benchmark its performance. The results are structured as follows. In Section 3.2.1, we develop an event-driven algorithm for synaptic weight updates in e-prop. In Section 3.2.2, we first reproduce a regression and a classification proof-of-concept task from Bellec et al. [154] to demonstrate that the event-driven scheme can replicate the results of the time-driven scheme. Then, we implement the neuromorphic MNIST (N-MNIST) task using our event-driven algorithm, showcasing its performance on a widely recognized benchmark. In Section 3.2.3, we introduce additional features to enhance the model's biological plausibility and demonstrate the computational accuracy of these enhancements on the N-MNIST task. Finally, in Section 3.2.5, we evaluate the computational efficiency of the event-driven algorithm in terms of runtime, comparing versions with and without additional biological features. Across these sections, we present the mathematical and algorithmic formulations, as well as simulation results, that underpin our approach and support our findings.

3.2 RESULTS

3.2.1 *From time-driven to event-driven e-prop*

In artificial neural networks (ANNs), all elements, including neurons and synapses, are updated synchronously at every computational time step. Matrix multiplications are well-suited for this algorithm, as implemented in widely used frameworks like TensorFlow [177] and PyTorch [178]. However, spiking neural networks (SNNs) require algorithms better tailored to their distinct temporal dynamics.

3.2.1.1 *The case for event-driven synapse updates*

To accurately integrate the differential equations governing neuron dynamics, a typical computational time step of 0.1 ms is required, defining the smallest relevant timescale. Neurons communicate via spikes, which lack magnitude and encode information solely through their timing. The propagation time of a spike, known as the synaptic delay, varies between synapses and is typically an order of magnitude larger than the computation time step. Given that the typical spike frequency is 1 spikes s^{-1}, two spike events at a given synapse are separated by approximately 10 000 computational time steps.

Biological neural networks exhibit sparsity not only in time but also in space. Even within a cubic millimeter of tissue — where a neuron forms most of its connections — the probability of two neurons sharing a common synapse is only about 0.1 Despite this spatial sparsity, each neuron still receives roughly 10 000 synapses and projects to as many target neurons. Consequently, with the parameters mentioned above, a neuron processes an input spike roughly every 0.1 ms, corresponding to the computation time step required for the neuron's internal dynamics. Thus, spikes are rare events from the perspective of a synapse, but not from the neuron's point of view. Given the widely different time scales and object counts, separating neurons and synapses into distinct entities proves computationally advantageous. Some algorithms for simulating SNNs therefore adopt a hybrid approach, combining event-driven and time-driven strategies: spike communication and synaptic processing are handled in an event-driven manner, while the neurons' state updates are time-driven. In this framework, a spike is multiplied by its corresponding (potentially plastic) synaptic weight when it arrives at the post-synaptic neuron, without the pre-synaptic neuron needing to be aware of the weight. Over two decades of research on such a hybrid algorithm [101, 89, 77] have demonstrated its suitability for parallel and distributed computing [39, 161].

3.2.1.2 *Event-driven weight updates in e-prop*

The e-prop learning rule, designed specifically for SNNs, relies solely on information locally available at the synapse. The primary bottlenecks of the e-prop algorithm, ranked in descending order based on the number of objects, are the computations in the synapses, recurrent neurons, and output neurons. This makes it an ideal candidate for transitioning from the strictly time-driven algorithms used in ANNs to a hybrid approach that leverages the spatiotemporal sparsity of large-scale neural networks.

Our event-driven algorithm for the e-prop model stores presynaptic spikes z_i^{t-1} from each neuron i within the synapse. Following the framework described by Stapmanns et al. [100], the time-driven neurons archive the histories of other quantities required for weight updates: surrogate gradients ψ_j^t and learning signals L^t (see Figure 3.1a, Equation 3.41, and Bellec et al. [154]). The weight update is calculated over the history corresponding to the update interval which is defined as the sample length T (typically 1 s to 2 s) multiplied by the batch size n. At the first spike occurring after each sample length, the algorithm computes and accumulates a gradient (Figure 3.1b). During this process, the synapse retrieves the histories from the postsynaptic neuron to compute the weight update associated with the update interval. Then, at the first spike occurring after each complete update interval, the accumulated gradients are averaged, and the weights are updated.

Especially in sparse spiking scenarios the first spike after a complete sample length may occur several multiples of the sample length or even of the update

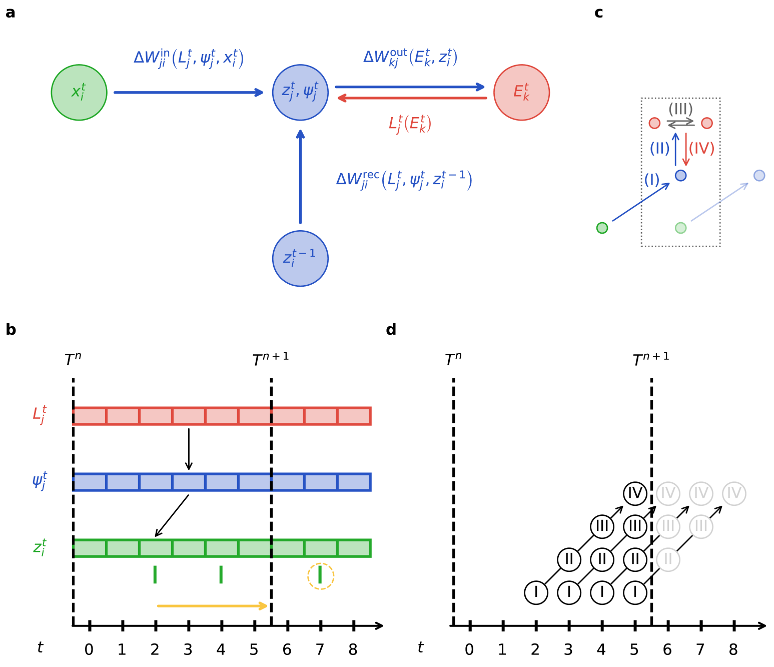


Figure 3.1: **Mathematical basis and technical implementation of e-prop with event-driven weight updates. (a)** Weight update rules for input, recurrent, and output synapses. **(b)** The first spike of the update interval T^{n+1} triggers the retrieval of the archived history for T^n and the computation of the corresponding weight update. Arrows indicate the surrogate gradient entries ψ_j^t and presynaptic spikes z_i^{t-1} associated with a learning signal entry L_j^t. **(c)** Instantaneous propagation of spikes and signals within a single time step (dotted box). Over e-prop synapses (blue arrows): (I) Transmit spikes from input neurons (green) to recurrent neurons (blue), (II) transmit spikes to output neurons (red). (III) Transmit signals between output neurons (gray arrows) to compute the softmax. (IV) Send the learning signal (red arrow) from the output layer to the recurrent layer. **(d)** Representation of transmission I-IV as a pipeline across four time steps. The number of incomplete operations (gray) at the boundary (dashed line) increases with pipeline depth. In this case, three learning signals are missing, corresponding to the pipeline depth minus one.

interval later. In this case, the synapse retrieves the histories from the postsynaptic neuron to compute the weight update associated with the sample during which the last spike traversed the synapse. All later samples during which no incoming spike activated the synapse can be neglected since the presynaptic factor is zero and since it enters multiplicatively in the weight update, these sections of the history do not change the weight.

For each time step t of that update interval, the loss can be incrementally calculated from the corresponding history values z_i^{t-1}, ψ_j^t, and L^t:

$$\mathcal{L}(W_{ji}) = \sum_{t=1}^{T} l^t \left(W_{ji}\right), \tag{3.1}$$

allowing to accumulate the gradient over the entire update interval:

$$\left.\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}W_{ji}}\right|_{W_{ji}=W_{ji}^0} = \sum_{t=1}^{T} \left.\frac{\mathrm{d}l^t \left(W_{ji}\right)}{\mathrm{d}W_{ji}}\right|_{W_{ji}=W_{ji}^0}. \tag{3.2}$$

This final gradient can be optimized with gradient descent or Adam (Equation 3.54), yielding the weight update. The weight updates are accumulated over a batch of n update intervals, averaged, and then applied to the synaptic weights. To free up memory, we clear the histories of all update intervals where synapses either retrieved data or lacked presynaptic spikes.

3.2.1.3 *Transmission delays*

In the main derivation of the original e-prop model by Bellec et al. [154], for simplicity, delays are modeled only for recurrent connections and with a fixed value of one time step. In the supplementary materials, they generalize the formulas to account for non-uniform input and recurrent delays with arbitrary values. However, transmissions from recurrent to output neurons, output to recurrent neurons, and within the output layer are still modeled as instantaneous (see Figure 3.1c).

Instantaneous transmission from recurrent to the output layer is indicated by the update equation for the output layer, where the output signal $\mathbf{y}^t$ and the spike state $\mathbf{z}^t$ share the same temporal index:

$$\mathbf{y}^t = h\left(\mathbf{y}^{t-1}, \mathbf{z}^t\right). \tag{3.3}$$

Here, $\mathbf{y}^t$ represents a vector of continuous signals from the output neurons, while $\mathbf{z}^t$ denotes the spike state variables of the recurrent neurons, both at time t.

Instantaneous transmission between output neurons is indicated in the softmax function in Equation 3.28, where the numerator and denominator share the same temporal index.

Instantaneous transmission from the output to the recurrent layer is indicated by the same temporal index in both the learning signal, L_j^t, and the surrogate gradient, ψ_j^t in Equation 3.30. However, the learning signal is computed remotely from the synapse and neuron and is derived from the neuron's state, represented by the surrogate gradient. Consequently, having the same temporal index for the learning signal and the surrogate gradient introduces a causality issue.

These assumptions of instantaneous transmissions conflict with empirical evidence showing delays in neurobiological processes [179] and their critical role in the processing and representation of information in the brain [180]. Furthermore, instantaneous transmission presents a computational challenge when integrating this framework into neural simulators, which typically separate spike and signal transmission by at least one time step. To address this, our implementation compensates for transmission delays by synchronizing the histories of the factors involved when calculating the weight updates. Moreover, we decouple the connection delay from the simulation time step, allowing for different, particularly shorter, time steps.

All transmission processes can be conceptualized as a pipeline [181], which allows the distribution of these instantaneous transmissions across multiple time steps. However, the pipeline reveals that three learning signals are missing at the end of each update interval due to the assumption of instantaneous transmission (Figure 3.1d).

One way to address the absence of the third learning signal is by integrating over a semi-open range, which is open at the beginning and closed at the end. In the following sections, we will present methods to recover the missing learning signals and incorporate the necessary delays. But first we will introduce different tasks which are used for validation of our implementation and the assessment of further changes to the algorithm.

3.2.2 *Supervised benchmark tasks with event-driven e-prop*

3.2.2.1 *Pattern generation*

As a first proof of concept, we reproduced a supervised regression task from Bellec et al. [154] with the event-driven strategy visualized in Figure 3.2a. In this task, the network learns to generate a signal composed of the summation of four sinusoids, each with randomly assigned phases and amplitudes. The frozen spike input pattern functions as a temporal backbone for the signal. The network output is projected on one output neuron whose membrane voltage y_k fluctuates around zero before training and follows the entrained signal after training.

Figure 3.2b shows the time courses of the dynamic variables before and after training. The before vs. after training weight distributions (Figure 3.2c) show that this task can be solved if only the output synapses are plastic. The loss time course recorded in a simulation with the event-driven implementation

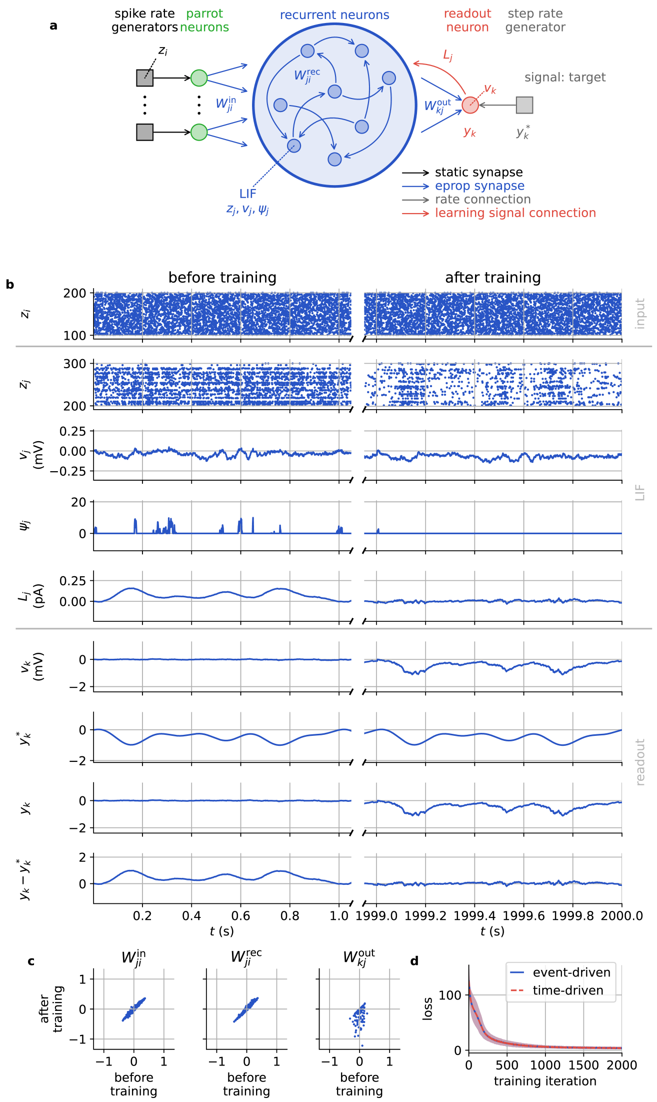


Figure 3.2: **Pattern generation as a regression task with event-driven e-prop. (a)** Network architecture. **(b)** Time traces of dynamic variables recorded before and after training. **(c)** Distributions of input, recurrent, and output weights. **(d)** Time course of the loss (mean squared error), compared between the event-driven and the time-driven implementation, with mean and standard deviation values averaged over 10 random seeds.

accurately matches the loss time course recorded in a simulation with the time-driven implementation (Figure 3.2d). This task uses a mean-squared error as loss (Equation 3.26), and the gradient is optimized via gradient descent (Equation 3.48).

To demonstrate potential applications, we implemented two tasks requiring a second output neuron inspired by tasks introduced in Laje et al. [182]. The two output neurons encode the horizontal and vertical coordinates of a signal, enabling the creation of patterns such as a lemniscate (Figure 3.3a) or handwriting (Figure 3.3b).

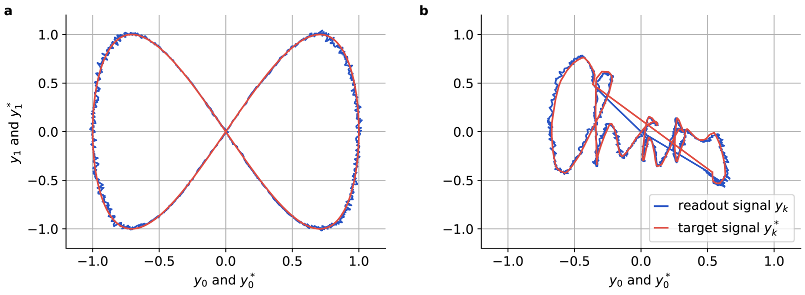


Figure 3.3: **Pattern generation with two output neurons.** A lemniscate pattern **(a)** and the handwritten word “chaos” **(b)**, with two output neurons encoding the horizontal and vertical coordinates, trained for 4000 iterations.

3.2.2.2 *Evidence accumulation*

As second proof of concept, we reproduce a classification task from Bellec et al. [154] that is more challenging to learn than the regression task described in the previous section. This evidence accumulation task, illustrated in Figure 3.4a, is inspired by a behavioral task in which a mouse runs on a trail and gets cues on the left and right. At the end of the tunnel, it has to decide if it turns left or right, whereby the correct decision would be according to the underlying rationale to turn to the side with more cues. In the spiking network, two input populations provide Poisson spike trains that represent the cues. A third input population provides background input throughout the task, and a fourth is only active at the end of each update interval, indicating the phase when the network must decide. The plasticity is turned on only in this last period, so we refer to it as a learning window. A long intermediate phase between the presentation of the cues and the onset of the recall phase with solely background adds an extra challenge to this task since the network needs to keep the cues in memory. This memory can be enabled by spike threshold adaptation with a slow decay, which introduces the required long time constants into the dynamics. Thus, the recurrent network for this task consists of adaptive LIF (ALIF) neurons (Equation 3.21).

To solve a classification task, each output neuron represents a single class; in this case, two output neurons correspond to the two possible options. Since this is a classification task, a cross-entropy loss (Equation 3.29) is employed, and the membrane voltages of the output neurons are converted into probabilities using a softmax function (Equation 3.28). The softmax function computes probabilities by dividing the exponential of each output neuron's membrane voltage by the sum of the exponentials of all output neuron membrane voltages. Consequently, the softmax function requires access to the membrane voltages of all output neurons, introducing additional connections between the output neurons and necessitating an extra time step in the algorithm.

The weight updates for this task are optimized using the Adam algorithm (Equation 3.54). Our experiments further indicate that this task can only be solved using batch learning. Batch learning in the original model is implemented by running multiple network copies, equal to the batch size, in parallel with identical initialization but different task examples. After each iteration, the weights are averaged, and the averaged weights are applied across all networks. To achieve biologically realistic batch learning, we process the batch sequentially rather than in parallel, applying the averaged weights only after completing the specified number of iterations equal to the batch size.

Comparisons of the time courses of the dynamic variables before and after training are presented in Figure 3.4b, while comparisons of the weight distributions are shown in Figure 3.4c. As the loss shows, the event-driven implementation reproduces the time-driven implementation accurately (Figure 3.4d). The deviations are due to floating point arithmetics. Minor deviations in the optimized weights might slightly change a neuron's membrane voltage time course. A tiny deviation in the membrane voltage can lead to the membrane voltage in one implementation be slightly above the threshold whilst in the other slightly below. This is already enough to trigger one spike in one case that is missing in the other. To investigate this impact, we conducted perturbation experiments, introducing one extra spike and comparing the network spikes to a simulation without this extra spike (Figure 3.5a). These experiments show that this extra spike introduces a cascade of deviations in spike times, which accumulate to a significant deviation in loss (Figure 3.5b).

3.2.2.3 *Neuromorphic MNIST*

We implement a classification task using the N-MNIST dataset [183], an adaptation of the traditional MNIST dataset for handwritten digits, specifically designed for neuromorphic computing. The N-MNIST dataset captures changes in pixel intensity using a dynamic vision sensor, converting static images into sequences of binary events. These binary events can be interpreted as spike trains,

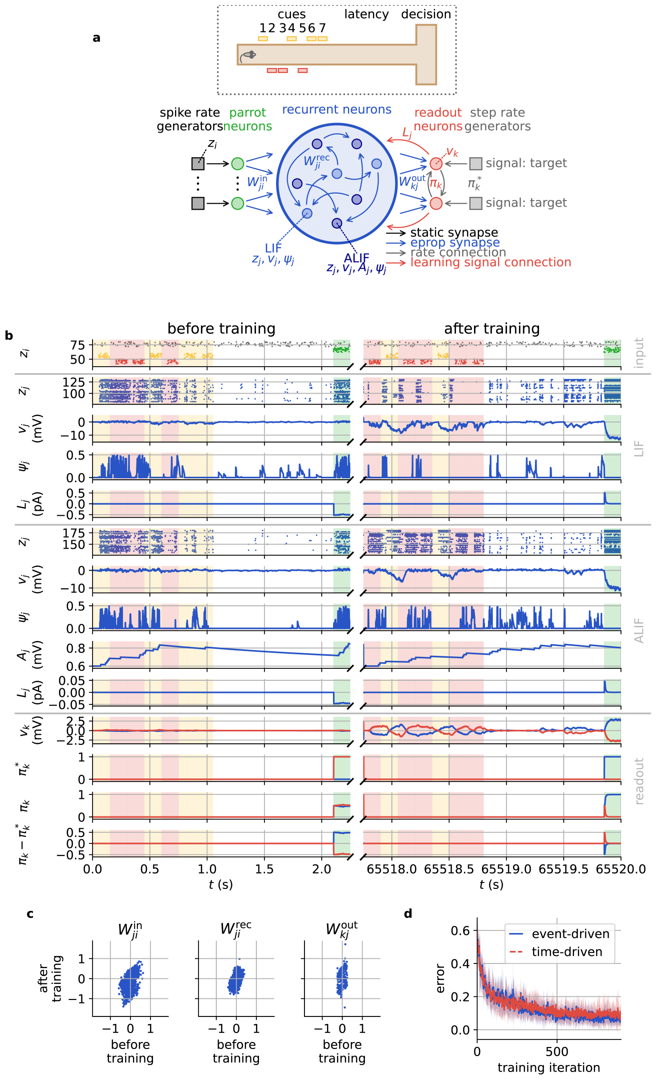


Figure 3.4: **Evidence accumulation as a classification task with event-driven e-prop. (a)** Schematic of a behavioral experiment where a mouse receives cues on the left and right sides of a track, followed by a decision after a latency, along with the corresponding network architecture. **(b)** Time traces of dynamic variables recorded before and after training. **(c)** Distributions of input, recurrent, and output weights. **(d)** Time course of the prediction error, compared between the event-driven and the time-driven implementation, with mean and standard deviation values averaged over 10 random seeds.

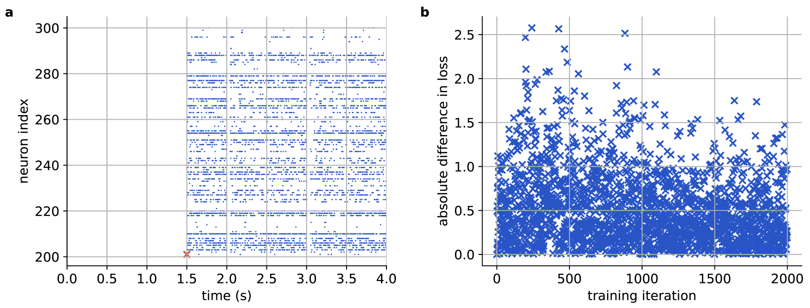


Figure 3.5: **Impact of a perturbation spike on learning dynamics during the regression task. (a)** Spikes that differ between simulations with and without a forced spike emission from the neuron indexed at 201 at 1.5 s (red cross). **(b)** Absolute difference in loss between the perturbed and non-perturbed simulations.

mimicking biological neural processing, making this task particularly suitable for spiking neural networks (SNNs) equipped with the e-prop algorithm.

Each image in the dataset consists of a 34×34 pixel grid and a channel encoding pixel intensity. Pixel intensity changes are encoded as binary events: a value of 1 for an ON event (corresponding to an increase in intensity) and a value of 0 for an OFF event (corresponding to a decrease in intensity). Increases in intensity indicate brightening or the appearance of new features in the visual field, while decreases signify dimming or the disappearance of features.

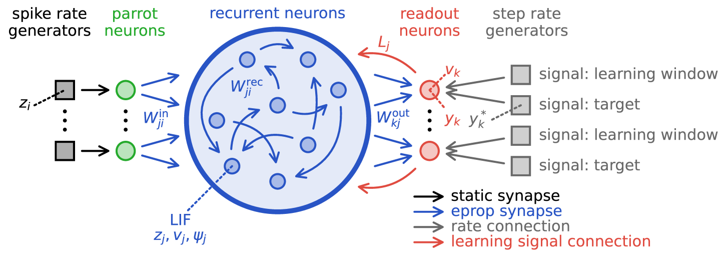
To enhance computational efficiency, we exclude pixels that generate no or very few events. We represent each remaining pixel by a spike generator, which emits a spike for every ON event registered in that pixel. Each spike generator sends these spikes to a corresponding input neuron. The input neurons project onto a recurrent network, which is further connected to 10 output neurons — one for each digit class. Each output neuron compares the network signal to the teacher signal, generated by a rate generator representing the correct digit class.

The setup is visualized in Figure 3.6a. The training error is computed using the cross-entropy loss, and optimization is performed using gradient descent with a batch size of 1. The time courses of the dynamic variables are shown in Figure 3.7a, the weight distributions in Figure 3.7b — both comparing before and after training — and the prediction error time course in Figure 3.7c.

3.2.3 *Event-driven e-prop with additional biological features*

In this section, we omit several machine learning components from the original model and incorporate additional biological features into our event-driven e-prop model.

a



b

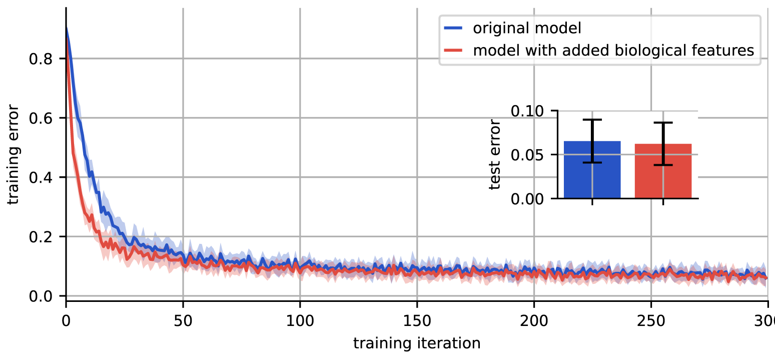


Figure 3.6: **Computational accuracy and efficiency of event-driven e-prop with biological features. (a)** Implementation of network learning for the N-MNIST task using e-prop plasticity with event-driven weight updates. In the biologically enhanced model, each readout neuron receives an additional learning window signal. **(b)** Training error time courses for the N-MNIST task for the original model and a biologically enhanced model, where the membrane voltage resets to a fixed value following each spike. The inset presents test errors averaged over 10 iterations. All error values for both training and testing are expressed as the mean ± standard deviation, computed across 10 random seeds.

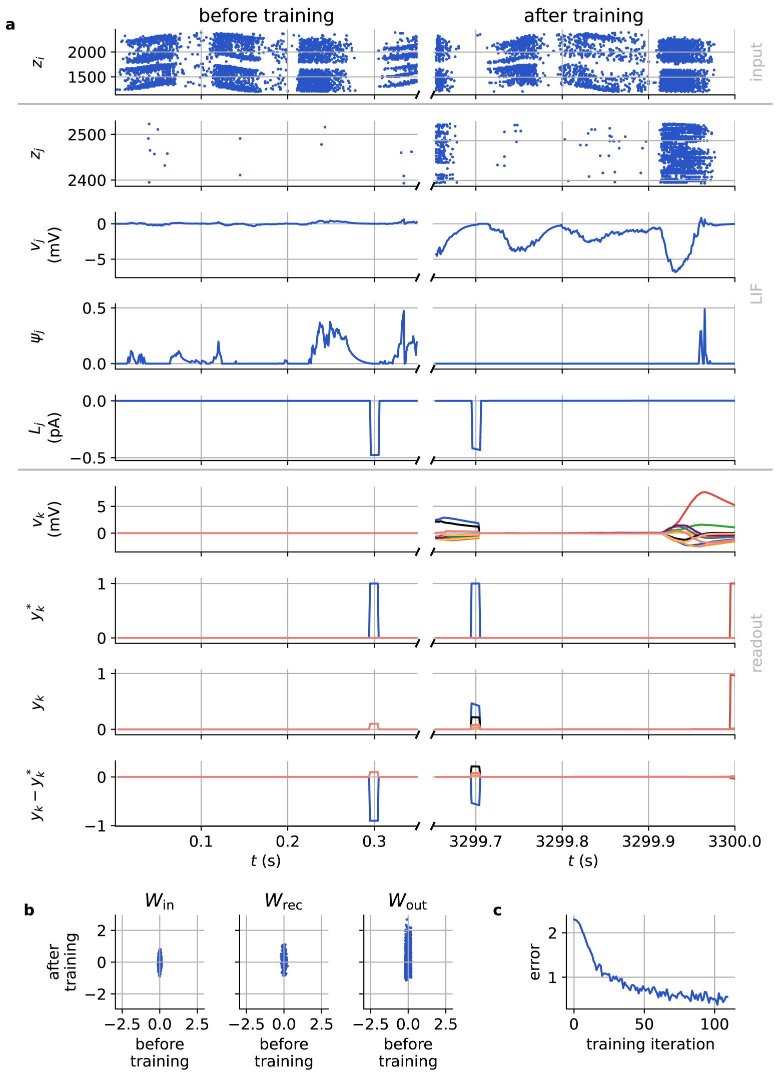


Figure 3.7: **Dynamics and weight distributions before and after training N-MNIST using event-driven e-prop. (a)** Time traces of dynamic variables recorded before and after training. **(b)** Distributions of input, recurrent, and output weights. **(c)** Time course of the prediction error.

3.2.3.1 *Dynamic firing rate regularization*

In the original implementation by Bellec et al. [154], another feature dependent on the update interval is the regularization of the firing rate (Equation 3.46), which we successfully reproduce in our event-driven implementation. Additionally, Bellec et al. [154] propose a moving average variant of firing rate regularization that accumulates spikes from the start of the iteration up to the current time step and normalizes them by the current time step. We adopt this dynamic firing rate regularization as it eliminates the dependency on the update interval, thereby enabling inter-spike integration.

The corresponding loss is defined as

$$\mathcal{L}_{\text{reg}} = c_{\text{reg}} \frac{1}{2} \sum_t \sum_j \left(\bar{f}_j^{(t)} - f^{\text{target}} \right)^2, \tag{3.4}$$

with the dynamic firing rate

$$\bar{f}_j^{(t)} = \frac{1}{\Delta t} \frac{1}{t} \sum_{t'=0}^{t} z_j^{(t')} = \beta^{(t)} \bar{f}_j^{(t-1)} + \left(1 - \beta^{(t)}\right) \frac{z_j^{(t)}}{\Delta t}, \tag{3.5}$$

where $\beta^t = \frac{t}{t+1}$ and c_{reg} is the regularization coefficient. We propose using an exponential moving average by replacing β^t with a constant β, effectively converting the operation into a low-pass filter. This approach offers two key advantages over the standard moving average. First, it eliminates explicit time dependence, both as a relative time measure and as an index, and does not rely on a fixed time origin. Second, while the standard moving average assigns equal importance to all spikes independent of the temporal occurrence, the exponential moving average decay prioritizes recent spikes, making it more responsive to dynamic changes in the neural activity.

In case of the exponential moving average, Equation 3.5 can be further expressed as:

$$\begin{aligned} \bar{f}_j^t &= \mathcal{F}_\beta \left(z_j^t \right) && (3.6) \\ &= \beta \bar{f}_j^{t-1} + (1 - \beta)\, z_j^t && (3.7) \\ &= (1 - \beta) \sum_{t'=0}^{t} \beta^{t-t'} z_j^{t'}. && (3.8) \end{aligned}$$

The gradient of the dynamic firing rate regularization loss is given by

$$\frac{\mathrm{d}\mathcal{L}_{\text{reg}}^t}{\mathrm{d}W_{ji}} = c_{\text{reg}} \left(\bar{f}_j^t - f^{\text{target}} \right) \frac{\mathrm{d}\bar{f}_j^t}{\mathrm{d}W_{ji}}, \tag{3.9}$$

Applying the core idea of e-prop [154], which considers only local interactions, and using the definition of the eligibility trace, gives:

$$\frac{\mathrm{d}\bar{f}_j^t}{\mathrm{d}W_{ji}} = \frac{\mathrm{d}\mathcal{F}_\beta\left(z_j^t\right)}{\mathrm{d}W_{ji}} \tag{3.10}$$

$$= (1-\beta)\sum_{t'=0}^{t}\beta^{t-t'}\frac{\mathrm{d}z_j^{t'}}{\mathrm{d}W_{ji}} \tag{3.11}$$

$$= \mathcal{F}_\beta\left(\frac{\mathrm{d}z_j^t}{\mathrm{d}W_{ji}}\right) \tag{3.12}$$

$$\approx \mathcal{F}_\beta\left(\left[\frac{\mathrm{d}z_j^t}{\mathrm{d}W_{ji}}\right]_{\text{local}}\right) \tag{3.13}$$

$$= \mathcal{F}_\beta\left(e_{ji}^t\right). \tag{3.14}$$

With this expression Equation 3.9 becomes:

$$\frac{\mathrm{d}\mathcal{L}_{\text{reg}}^t}{\mathrm{d}W_{ji}} \approx c_{\text{reg}}\left(\bar{f}_j^t - f^{\text{target}}\right)\mathcal{F}_\beta\left(e_{ji}^t\right), \tag{3.15}$$

which yields the weight update for the dynamic firing rate regularization for synapses with recurrent neurons on the postsynaptic site:

$$\Delta W_{ji}^{\text{reg}} = \eta c_{\text{reg}}\left(\bar{f}_j^t - f^{\text{target}}\right)\mathcal{F}_\beta\left(e_{ji}^t\right). \tag{3.16}$$

3.2.3.2 *Inter-spike integration*

In the time-driven implementation, all synaptic weights are updated on a fixed time grid. In contrast, the event-driven implementation updates a synaptic weight after the passage of the update interval plus the time when the first spike is transmitted. This update scheme ensures that the first spike requiring the new weight utilizes the correct value.

Thus, in the time-driven implementation, all synapses are updated synchronously, whereas, in the event-driven implementation, synaptic updates are clustered around the beginning of each update interval.

However, this update scheme presents two problems. First, all training samples must have the same length. Second, it remains unclear what biological mechanism corresponds to such a synchronous update, perhaps a central clock, as all components of the system (i.e., synapses and neurons across all layers) require knowledge of this global update interval.

Some algorithms update weights more frequently than once per update interval. For instance, Truncated Real-Time Recurrent Learning (TRTRL) [184], an adaptation of Truncated Backpropagation Through Time (TBPTT) [185], updates weights midway through an update interval. Truncated algorithms offer the advantage of adapting more quickly to changes in the input sequence, making them more responsive to new information and improving the learning of short-term patterns. However, they have a reduced ability to capture long-term dependencies and are less accurate, as they only approximate the exact gradients that would otherwise be computed over the full sequence. In other words, the gradients are calculated using outdated parameters, which, in this case, are the synaptic weights.

This trade-off between dynamic adaptation and gradient accuracy [186] must be carefully balanced based on the specific characteristics of the task. Forward Propagation Through Time (FPTT) [187] solves this issue by updating weights at every time step while minimizing a dynamically regularized risk function for the loss [for SNNs, see 188].

To mitigate the biological implausibility of a synchronous update mechanism, we draw on the central idea behind truncated algorithms and propose an inter-spike integration scheme. In this approach, each spike transmitted by a synapse triggers the calculation of a weight update based on the time elapsed since the last spike, ensuring that the current spike utilizes the newly calculated weight. This scheme aligns with established approaches to plasticity [e.g., 189, 89].

Given two consecutive spikes in a synapse, $t_{\text{prev spike}}$ and t_{spike}, the weight update corresponding to the most recent learning signal, L^t, is computed using the history within the interval $[t_{\text{prev spike}} + \Delta t, t_{\text{spike}}]$, where Δt represents one time step. This scheme produces loss values that deviate from the original approach, where weights remain constant over the entire update interval, as each transmitted spike uses a distinct weight.

3.2.3.3 *Continuous dynamics*

In the original implementation by Bellec et al. [154], the neuronal dynamic variables and the traces of the filtered e-prop variables are reset to zero after each update interval. This reset aims to mitigate sample interference by preventing residual activity from one sample from affecting the weight updates of the next. We conduct experiments without resetting the neuron dynamics and the e-prop traces.

3.2.3.4 *Learning window signal generator*

The learning window in the classification task is defined as the last few hundred milliseconds of each update interval. To support the new implementation, we introduce a learning window signal generator that indicates with the value one that the learning window is open and plasticity is on, and zero if the learning window is closed and plasticity is off. This generator makes the learning window independent of the update interval and ensures maximal flexibility.

3.2.3.5 *Variable-length update intervals*

In the original implementation, the weights are updated at fixed time intervals determined by the batch size and the sample length. The features introduced in Section 3.2.3.2 through Section 3.2.3.1 eliminate this dependency on fixed update intervals. This allows the removal of fixed update intervals and enables the processing of samples with variable lengths.

3.2.3.6 *Classification via mean-squared error*

For the classification tasks, the original implementation uses a cross-entropy loss (Equation 3.29) which requires the output signal to be a softmax (Equation 3.28), but this is problematic for two reasons. First, it introduces an extra time step to communicate the signals between the output neurons to calculate the denominator. Second, exchanging these output signals and the transformation into a probability might not be biologically realistic. A potential alternative to classification learning with cross-entropy loss is introduced by Hui et al. [190], who show that squared error loss performs comparably to cross-entropy loss across several prominent neural network architectures and benchmark classification tasks. Considering the temporal dynamics of our training regime, we naturally extend this concept to mean-squared error and replace the temporal cross-entropy loss in Equation 3.29 with a temporal mean squared error:

$$\mathcal{L} = \frac{1}{K} \sum_{t=1}^{T} \sum_{k=1}^{K} {E_k^t}^2 = \frac{1}{K} \sum_{t=1}^{T} \sum_{k=1}^{K} \left(y_k^{*,t} - y_k^t \right)^2, \tag{3.17}$$

where K is the number of output labels, k the index of the output neuron, and the error signal E_k^t is given by Equation 3.25. The teacher signal $y^{*,t} \in \mathbb{R}^K$ is a one-hot encoded vector representing the correct label when t is within the learning window and a vector of zeros otherwise.

This reformulation eliminates the need for inter-neuron communication at each time step, thereby reducing the number of missing learning signals by one.

3.2.3.7 *Eligibility trace filter decoupled from output time constant*

In the rigorous derivation of the weight update rule in Bellec et al. [154], the output neuron's time constant in Equation 3.24 emerges as the time constant of the eligibility trace filter in Equation 3.39. Since the weight update is computed at the synapses, this implies that the synapses must be aware of the output neuron's time constant, which is biologically implausible and violates the locality of the learning rule.

To address this issue, we conducted experiments to assess how learning performance depends on the filter having a time constant identical to that of the output neuron.

3.2.3.8 *Smooth surrogate gradient function*

The original model employs a piecewise linear surrogate gradient as defined in Equation 3.34. We first tune the two scale factors. Additionally, the literature offers a plethora of surrogate gradients to explore (for an overview, see Neftci et al. [191]).

Inspired by Shrestha et al. [192], we employ an exponential surrogate gradient function:

$$\psi(v^t) = \gamma \exp\left(-\beta \left|v^t - v^t_{\text{th}}\right|\right), \tag{3.18}$$

as it is smoother than the piecewise linear function and therefore more biologically plausible.

We also investigate further smooth surrogate gradients: a function corresponding to the derivative of a fast sigmoid as used in Zenke et al. [193]

$$\psi(v^t) = \gamma\left(1 + \beta\left|v^t - v^t_{\text{th}}\right|\right)^2, \tag{3.19}$$

and an arctan function as used in Fang et al. [194]

$$\psi(v^t) = \frac{\gamma}{\pi} \frac{1}{1 + \left(\beta\pi\left(v^t - v^t_{\text{th}}\right)\right)^2}. \tag{3.20}$$

We find that the different surrogate gradients suggested in the literature can be made similar in shape by adequately adjusting the height and width parameters (Figure 3.8a).

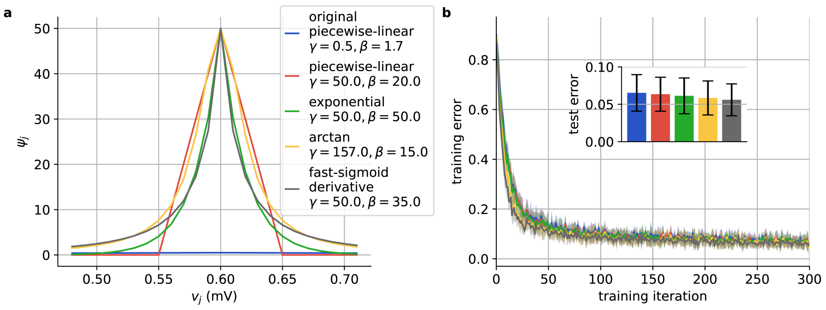


Figure 3.8: **Accuracy comparison between different surrogate gradients. (a)** Profiles of different surrogate gradient functions as a function of voltage, with a threshold voltage set at 0.6 mV. **(b)** Training error time courses for the N-MNIST task, comparing models with surrogate gradient shown in **(a)**. The inset shows test errors averaged over 10 iterations. All error values, for both training and testing, are reported as the mean ± standard deviation, calculated across 10 random seeds.

3.2.3.9 *Full membrane voltage reset*

In the original model, the membrane voltage undergoes a partial reset by decrementing it by a specific value after each spike emission. In contrast, we adopt a neuron model with a full reset mechanism, where the membrane voltage is reset to a fixed value after each spike.

3.2.3.10 *Sparse connectivity and weight constraints*

In artificial neural networks, weights can typically assume both positive and negative values. In contrast, biological neurons release only a single type of neurotransmitter, meaning that the outgoing synapses of a given neuron are either exclusively excitatory (positive) or exclusively inhibitory (negative) — a principle known as Dale's law. Although the type of synapse remains constant over time, its strength can change (functional plasticity), and synapses can vanish while new ones form (structural plasticity). Additionally, biological neural networks usually consist of approximately 80 % excitatory neurons and 20 % inhibitory neurons, with a connection probability of about 10 %.

Bellec et al. [154] construct a network fulfilling all these constraints and train it using e-prop in combination with the stochastic rewiring algorithm DEEP R [195], which mimics structural plasticity. The learning performance of this setup exceeds that of the standard network and e-prop without rewiring after 700 iterations on the evidence accumulation task [see 154, Fig. 3c].

Simulating the N-MNIST task with a fully connected network is impractical due to the excessive computational load. Furthermore, the training error remains high over many iterations, suggesting that the dense network fails to effectively learn the task. These difficulties are overcome by reducing the connection probability to 25 % for the input and 1 % for the recurrent connections, while keeping the output connections dense. Consequently, we use sparse networks for all experiments presented in Table 3.1.

We further explore biological weight constraints using an e-prop model that incorporates all relevant biological features along with a full membrane voltage reset. Specifically, we enforce that the weights do not change signs, applied separately to input, recurrent, and output weights, as well as to all weights combined. Additionally, we implement Dale's law with an excitatory-to-inhibitory ratio of 4:1.

3.2.4 *Computational Accuracy*

We evaluate the computational accuracy of each feature on the N-MNIST task in isolation. Computational accuracy, defined as the learning performance, measures how effectively the network learns the task over 300 training iterations. It is quantified by the classification error, averaged over 10 test iterations. A smaller classification error corresponds to better learning performance. The

Table 3.1: **Computational accuracy and efficiency of additional biological features.** The results were obtained using the N-MNIST task, simulated for 300 training iterations and 10 test iterations, averaged over 10 random seeds. Absolute test errors were rounded to four decimal places, and runtimes to three decimal places. Relative values were calculated as deviations from the baseline model (without features) using unrounded values and subsequently rounded to the nearest whole number.

biological feature	test error		runtime	
	absolute	relative	absolute	relative
		%	core–h	%
without features	0.0653 ± 0.0064	+0	368.532 ± 2.986	+0
continuous neuron dynamics	0.0654 ± 0.0061	+0	368.373 ± 1.307	+0
continuous e-prop trace dynamics	0.0674 ± 0.0061	+3	383.024 ± 2.156	+4
mean squared error classification	0.0557 ± 0.0058	**-15**	367.437 ± 2.874	+0
eligibility trace filter decoupled from output	0.0692 ± 0.0061	+6	370.026 ± 4.478	+0
scaled linear surrogate gradient	0.0635 ± 0.0056	**-3**	426.177 ± 8.718	+16
scaled exponential surrogate gradient	0.0613 ± 0.0062	**-6**	419.797 ± 8.904	+14
all biological features combined	0.0659 ± 0.0120	+1	402.528 ± 3.871	+9
all biological features combined, full voltage reset	0.0622 ± 0.0083	**-5**	391.628 ± 2.644	+6
fixed sign input weights	0.0784 ± 0.0077	+20	425.519 ± 3.564	+15
fixed sign recurrent weights	0.0630 ± 0.0040	**-4**	406.037 ± 1.184	+10
fixed sign output weights	0.0726 ± 0.0059	+11	394.436 ± 3.936	+7
fixed sign all weights	0.0957 ± 0.0124	+47	427.141 ± 3.311	+16
Dale's law input weights	0.0941 ± 0.0107	+44	412.022 ± 3.422	+12
Dale's law recurrent weights	0.0626 ± 0.0067	**-4**	405.430 ± 3.023	+10
Dale's law output weights	0.0752 ± 0.0067	+15	394.125 ± 2.922	+7
Dale's law all weights	0.1129 ± 0.0132	+73	412.774 ± 3.558	+12

results for each experiment are presented in Table 3.1, both as absolute values and as relative differences from the learning performance of the event-driven model without features.

Experiments without resetting the neuron dynamics or the e-prop trace dynamics show 0 % and 3 % lower learning performance compared to implementations with resets, respectively. These findings suggest that such resets are not critical for maintaining learning performance, indicating minimal impact from sample interference.

Using mean-squared error for classification improves learning performance by 15 % compared to cross-entropy loss.

In an experiment where the filter time constant is decoupled from the output time constant (reduced from 100 ms to 30 ms), the learning performance decreases by only 6 %, showing that learning remains effective. In a regression task, the filter can be removed entirely, resulting in improved learning performance (see Figure 3.9). This suggests that the filter's time constant can be set independently of the output neuron's time constant, enabling synapses to function without requiring knowledge of the output neuron's parameters.

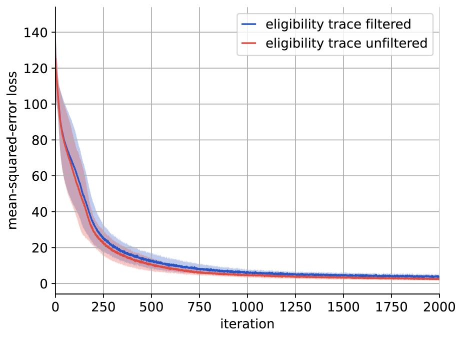


Figure 3.9: **Accuracy comparison between models with filtered and unfiltered eligibility traces.** Loss time courses for the pattern generation task simulated for 2000 iterations and averaged over 10 random seeds.

We find that tuning the two scale factors of the surrogate gradient function improves learning performance by 3 %, while using a scaled exponential surrogate gradient further enhances performance by 6 %. Surrogate gradient functions, when parametrized to have similar shapes, yield comparable learning performance (see Figure 3.8b). These findings align with previous studies, confirming that learning is robust to variations in the shapes of surrogate gradient functions [196]. Consequently, biological realism can be enhanced by using a smoother exponential surrogate gradient instead of a piecewise linear function, without compromising learning performance.

The learning performance of a model that integrates all biological features is less than 0 % worse than the feature-less model, while a model with an additional full membrane voltage reset achieves a 5 % improvement. For the model with all additional biological features and the full membrane voltage reset, the time courses of the dynamic variables are shown in Figure 3.10a, the weight distributions in Figure 3.10b — both comparing before and after training — and the prediction error time course in Figure 3.10c. The error time course in comparison to the original model is shown in Figure 3.6b.

To further investigate biological weight constraints, we utilize an e-prop model combining all biological features with a neuron model featuring a full membrane voltage reset, as described in Section 3.2.3.9. A slight decrease in learning performance is observed when prohibiting weight sign changes, with reductions of 20 % for input weights, 11 % for output weights, and 47 % for all weights combined, including recurrent weights.

A more pronounced decrease in learning performance occurs when applying Dale's law with an excitatory-to-inhibitory ratio of 4:1. This reduction is 44 % for input weights, 15 % for output weights, and a striking 73 % when applied to all weights, including recurrent weights.

Conversely, constraining only the recurrent weights, both by prohibiting sign changes and by implementing Dale's law with an excitatory-to-inhibitory ratio, leads to a 4 % improvement in learning performance. These findings suggest that constraining recurrent weights facilitates learning, whereas constraining input or output weights has the opposite effect. However, in these latter cases, the learning performance may eventually converge with that of the unconstrained network, albeit at a slower rate, indicating the network's ability to adapt and function effectively within these constraints.

3.2.5 *Computational Efficiency*

We evaluate the computational efficiency of each feature on the N-MNIST task in isolation. Computational efficiency is measured by the time required to simulate 300 training iterations and 10 test iterations, corresponding to a biological time of 2 h 35 min. The results for each experiment are presented in Table 3.1 as absolute values and as relative deviations from the runtime of the event-driven model without features.

Features such as continuous neuron dynamics, mean-squared error classification, and eligibility traces independent of the readout time constant result in a 0 % change in runtime. However, computational overheads are introduced by continuous e-prop trace dynamics (4 %), as well as by the scaled piecewise linear (16 %) and exponential surrogate gradient (14 %). Simulating models that incorporate all additional biological features requires 9 % more runtime, while including a full membrane voltage reset increases runtime by 6 % compared to the feature-less model.

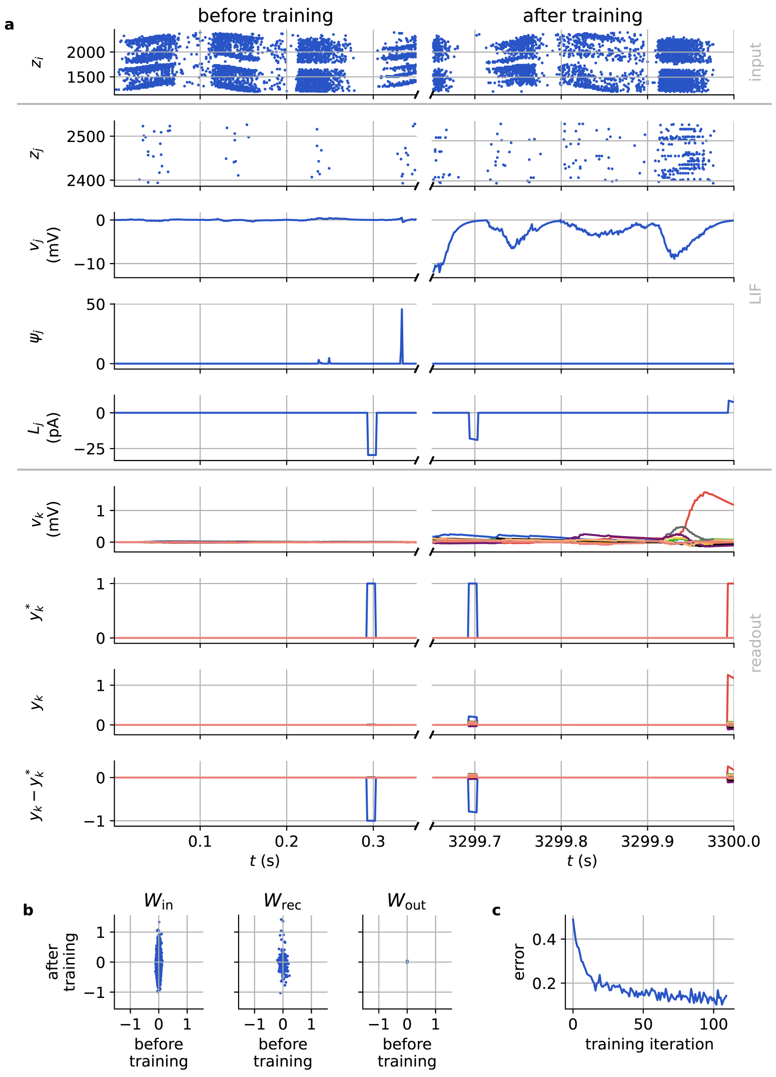


Figure 3.10: **Dynamics and weight distribution before and after training N-MNIST using biologically enhanced e-prop. (a)** Time traces of dynamic variables observed before and after training. **(b)** Distributions of input, recurrent, and output weights. **(c)** Time course of the prediction error.

In networks using the model with additional biological features and a full membrane voltage reset, small runtime overheads are introduced when constraining the sign of the input (15 %), recurrent (10 %), output (7 %), or all weights combined (16 %). Similarly, applying Dale's law with an excitatory-to-inhibitory ratio of 4 in the same networks increases simulation times by 12 % for input, 10 % for recurrent, 7 % for output, and 12 % for all weights combined.

These runtime overheads are within an acceptable range, rendering simulations with biological features feasible.

3.3 METHODS

In this section, we provide background on the original time-driven e-prop setup as outlined in Bellec et al. [154], along with details of their network architecture, neuron models, and optimization schemes to reproduce their experiments using our implementation as a proof of concept. Minor adjustments to the equations are occasionally made for consistency within our framework.

3.3.1 *Network architecture*

This work focuses on recurrent spiking neural networks (RSNNs), which consist of an input layer, a hidden layer with recurrent connections, and an output layer.

The primary function of the hidden layer is to process input sequences over time, extract temporal patterns within the data, and generate corresponding spike sequences. Recurrence plays a crucial role by providing the network with memory, enabling it to retain information from previous time steps. This allows the network to operate across multiple timescales, facilitating the effective modeling and processing of sequential or temporal data.

3.3.2 *Recurrent neuron models*

The neurons in the hidden layer are modeled as leaky integrate-and-fire (LIF) neurons in some cases with additional adaptation (ALIF). The following equations are presented for the case with adaptation but can be simplified for standard LIF neurons by setting the dynamic spike threshold voltage to a constant $v_{\text{th}}^t = v_{\text{th}}$ and omitting the equations for updating the spike threshold.

The dynamics of these neurons are described by the following update equation:

$$v_j^t = \alpha v_j^{t-1} + \sum_{i \neq j} W_{ji}^{\text{rec}} z_i^{t-1} + \sum_i W_{ji}^{\text{in}} x_i^t - z_j^{t-1} v_{\text{th},j}^t, \tag{3.21}$$

where j and i are neuron indices, W^{rec} and W^{in} are the recurrent and input synaptic strengths, and $\alpha = \exp\left(-\frac{\Delta t}{\tau_{\text{m}}}\right)$ represents the decay factor of the membrane voltage over time, with τ_{m} denoting the membrane time constant.

The binary spike state variable, $z_j^t = H\left(v_j^t - v_{\text{th}}^t\right)$, is set to 1 if the neuron spikes at time t, and 0 otherwise. The spike threshold voltage is updated according to:

$$v_{\text{th},j}^t = v_{\text{th}} + \beta_{\text{a}} a_j^t, \tag{3.22}$$

$$a_j^t = \rho a_j^{t-1} + z_j^{t-1}, \tag{3.23}$$

where $\rho = \exp\left(-\frac{\Delta t}{\tau_{\text{a}}}\right)$, β_{a} is the prefactor of the threshold adaptation, and τ_{a} is the adaptation time constant.

3.3.3 *Output neuron model*

All spikes from the recurrent neurons are integrated by the output neurons, which are modeled as leaky integrators:

$$y_k^t = \kappa y_k^{t-1} + \sum_j W_{kj}^{\text{out}} z_j^t + b_k^{\text{out}}, \tag{3.24}$$

where $\kappa = \exp\left(-\frac{\Delta t}{\tau_{\text{m,out}}}\right)$. The output layer's role is to integrate and filter the recurrent activity, convert it into a continuous output signal y_k^t, and compare it to the target signal $y_k^{*,t}$. The continuous target or teacher signal minus the output signal is the error signal

$$E_t^k = y_k^{*,t} - y_k^t. \tag{3.25}$$

3.3.4 *Loss function*

In regression tasks, the output signal is expected to approximate the target signal. The corresponding loss function is defined as:

$$\mathcal{L} = \frac{1}{2} \sum_t \sum_k E_k^{t\,2}. \tag{3.26}$$

The gradient of the loss with respect to the output of the recurrent network is given by:

$$\frac{\partial \mathcal{L}}{\partial y_k^t} = E_k^t. \tag{3.27}$$

In classification tasks, the network outputs a probability distribution over labels, calculated using the softmax function:

$$\pi_k^t = \text{softmax}\left(y_k^t\right) = \frac{\exp\left(y_k^t\right)}{\sum_{k'} \exp\left(y_{k'}^t\right)}. \tag{3.28}$$

This output signal is then used in conjunction with a cross-entropy loss function:

$$\mathcal{L} = -\sum_t \sum_k \pi_k^{*,t} \log \pi_k^t. \tag{3.29}$$

3.3.5 *Gradients*

The e-prop plasticity rule for supervised learning consists of a different set of weight update equations for the input, recurrent, and output synapses. The derivative of the loss w.r.t. to the recurrent weights is given by:

$$\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}W_{ji}^{\mathrm{rec}}} = \sum_t L_j^t \mathcal{F}_\kappa \left(e_{ji}^t\right), \tag{3.30}$$

whereby the filter $\mathcal{F}$ is defined in general defined as

$$\mathcal{F}_\kappa \left(e_{ji}^t\right) = \kappa \mathcal{F}_\kappa \left(e_{ji}^{t-1}\right) + e_{ji}^t. \tag{3.31}$$

One component of this plasticity rule is the eligibility trace, which accounts only for local information:

$$e_{ji}^t = \left[\frac{\mathrm{d}z_j^t}{\mathrm{d}W_{ji}^{\mathrm{rec}}}\right]_{\mathrm{local}} = \frac{\partial z_j^t}{\partial \mathbf{h}_j^t} \frac{\partial \mathbf{h}_j^t}{\partial W_{ji}} = \psi_j^t \mathcal{F}_\alpha \left(z_i^{t-1}\right), \tag{3.32}$$

where $\mathbf{h}_j^t$ denotes the vector of state variables for the recurrent neuron. The first factor of the eligibility trace represents the postsynaptic information:

$$\frac{\partial z_j^t}{\partial \mathbf{h}_j^t} = \frac{\partial z_j^t}{\partial v_j^t} \leftarrow \psi_j^t. \tag{3.33}$$

This surrogate gradient or pseudo-derivative function captures the relationship between the discrete postsynaptic spike state variable, which is not differentiable, and the postsynaptic membrane voltage. In the original model, a piecewise-linear surrogate gradient ψ_j^t is used:

$$\psi\left(v^t\right) = \gamma \max\left(0, 1 - \beta\left|v^t - v_{\mathrm{th}}^t\right|\right), \tag{3.34}$$

which has a peak of the size of the prefactor γ at the spike time, that is, when the membrane voltage crosses the spike threshold and linearly falls off to zero in the positive and negative directions. In comparison to the original definition of the surrogate gradient in Bellec et al. [154], we redefine the prefactor as $\gamma = \frac{\gamma_{\mathrm{original}}}{v_{\mathrm{th}}}$ and introduce a scaling factor, β, which incorporates the $\frac{1}{v_{\mathrm{th}}}$ term from the original definition.

The second factor of the eligibility trace represents the presynaptic information in the form of the filtered presynaptic spike train

$$\left[\frac{\partial \mathbf{h}_j^t}{\partial W_{ji}}\right]_{\mathrm{local}} = \mathcal{F}_\alpha\left(z_i^t\right) = \alpha \mathcal{F}_\alpha\left(z_i^{t-1}\right) + z_i^t, \tag{3.35}$$

where z_i^t denotes the spike state variable at time t. The surrogate gradient factor multiplied by the filtered spike train forms the eligibility trace

$$e_{ji}^t = \psi_j^t \mathcal{F}\left(z_i^{t-1}\right), \tag{3.36}$$

indicating if the synapse is eligible for change according to the Hebbian principle. For ALIF neurons, the eligibility trace is given by:

$$e_{ji}^{t} = \psi_j^t \left(\mathcal{F}_\alpha \left(z_i^{t-1} \right) - \beta_{\mathrm{a}} \epsilon_{ji,\mathrm{a}}^{t-1} \right), \tag{3.37}$$

with the adaptive component of the eligibility vector given as:

$$\epsilon_{ji,\mathrm{a}}^{t} = \rho \epsilon_{ji}^{t-1} + e_{ji}^{t}. \tag{3.38}$$

Moreover, the entire eligibility trace is filtered:

$$\mathcal{F}_\kappa \left(e_{ji}^{t} \right) = \kappa \mathcal{F}_\kappa \left(e_{ji}^{t-1} \right) + e_{ji}^{t}, \tag{3.39}$$

where $\kappa = \exp\left(-\frac{\Delta t}{\tau_{\mathrm{out}}}\right)$ represents the decay factor of the output neuron's membrane voltage over time, with τ_{out} denoting the output neuron's membrane time constant. The error signal multiplied by the feedback weights is the learning signal, and it provides the third factor; thus, the plasticity rule belongs to the class of three-factor learning rules. Here, L_j^t represents the learning signal:

$$L_j^t = \sum_k B_{kj} E_k^t. \tag{3.40}$$

The static random feedback weight matrix B_{kj} embodies the principle of feedback alignment [197], wherein the feedback weights are not symmetric to the output weights — a symmetry that would be biologically implausible. Instead, the plastic output weights adapt to align with these random feedback weights during training. When error signals are propagated directly from the output layer to each hidden layer, with each layer receiving its own distinct error signal, this process corresponds to direct feedback alignment [198]. Conversely, when the same error signal is broadcast to all neurons and modulated only by the random feedback weights, it is referred to as broadcast alignment [199]. In networks with a single layer (as used in all the experiments in this work), the concepts of feedback alignment and broadcast alignment are equivalent.

The factors are multiplied for each time step and summed for all time steps within an update interval typically around one to two seconds long, where each time step is one millisecond long. The gradient for each recurrent synapse is given as

$$g_{ji}^{t} = L_j^t \mathcal{F}_\kappa \left(\psi_j^t \mathcal{F}_\alpha \left(z_i^{t-1} \right) \right), \tag{3.41}$$

and for each input synapse as

$$g_{ji}^{t} = L_j^t \mathcal{F}_\kappa \left(\psi_j^t \mathcal{F}_\alpha \left(x_i^{t} \right) \right), \tag{3.42}$$

The plasticity rule for synapses projecting onto output neurons lacks the surrogate gradient part since the output neurons are leaky integrators without a spike mechanism:

$$g_{kj}^{t} = E_k^t. \tag{3.43}$$

3.3.6 *Firing rate regularization*

Firing rate regularization introduces a penalty when the firing rate of a recurrent neuron deviates from a target firing rate:

$$\mathcal{L}_{\text{reg}} = c_{\text{reg}} \frac{1}{2} \sum_j \left(\bar{f}_j - f^{\text{target}} \right)^2, \tag{3.44}$$

where $\bar{f}_j = \frac{1}{T} \sum_t z_j^t$ is the average firing rate of neuron j that is the sum of postsynaptic spike emissions averaged over an update interval T, and f^{target} is the target firing rate. This mechanism ensures that throughout the optimization, the firing rate of each recurrent neuron stays close to a desired target firing rate despite the weight updates. It is realized by adding a regularization term to the weight update. This regularization term computes the deviation of the average firing rate from the target firing rate averaged over an update interval:

$$g_{ji}^{\text{reg}} = c_{\text{reg}} \frac{1}{T n_{\text{trial}}} \left(f^{\text{target}} - \bar{f}_j \right) e_{ji}^{(t)}, \tag{3.45}$$

$$\bar{f}_j = \frac{1}{T n_{\text{trial}}} \sum_t z_j^{(t)}. \tag{3.46}$$

It is negative if the average firing rate is larger than the target firing rate, thus decreasing the weight, and positive if it is smaller, thus increasing the weight. While this deviation is specific to the postsynaptic neuron, each synapse multiplies the deviation with its eligibility trace history and averages it over an update interval. The full gradient is given as the sum of the e-prop gradient and the regularization gradient.

3.3.7 *Locality and causality*

The local and causal e-prop plasticity rules can be viewed as an approximation of the non-local, non-causal Backpropagation Through Time (BPTT) and non-local Real-Time Recurrent Learning (RTRL), both of which compute exact gradients [for a detailed comparison of these algorithms, see 200]. RTRL is computationally more intensive because it maintains eligibility traces that are updated recursively at each time step to ensure causality [200]. Here, non-causal refers to requiring knowledge of future activity, while non-local implies that the error must be propagated across all neurons and synapses [200]. Unlike BPTT, e-prop and RTRL are classified as "online gradient computation algorithms", as their computations for a given time step are causal and can be performed during the forward pass [201].

In the typical definition of BPTT, weights remain constant over T time steps and are only updated after the full sequence of T time steps. Under this condition, BPTT is equivalent to RTRL. An "online weight update algorithm", in contrast, would involve updating the weights at every time step, which produces results distinct from those of these three algorithms [201]. More precisely, in e-prop, weights are updated every nT time steps, where n is the batch size.

3.3.8 *Optimization*

Let the function $f(\cdot)$ represent an optimization mechanism, such as gradient descent. Once the new weights are computed, the time step is reset to zero, $t \leftarrow 0$, and the weights for the next interval are initialized to the updated values, $W_{ji}^{0} \leftarrow W_{ji}^{T+1}$, within the learning algorithm, and applied to the synapses:

$$W_{ji}^{T+1} = W_{ji}^{0} + f\left(\left. \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}W_{ji}} \right|_{W_{ji}=W_{ji}^{0}} \right). \tag{3.47}$$

In simple gradient descent, the weight update is calculated as

$$\Delta W_{ji} = -\eta \sum_{t} g_{ji}^{t}. \tag{3.48}$$

The Adam algorithm cannot be applied to the sum of gradients as in gradient descent since it has two internal variables that depend linearly and quadratically on the gradient in each time step, respectively, and thus have to be evolved in each time step. Here, we omit the indices ji for clarity, but the following equations are synapse-specific:

$$m^{(0)} = 0, v^{(0)} = 0, t = 1, \tag{3.49}$$

$$m^{(t)} = \beta_1 m^{(t-1)} + (1-\beta_1)\, g^{(t)}, \tag{3.50}$$

$$v^{(t)} = \beta_2 v^{(t-1)} + (1-\beta_2)\, {g^{(t)}}^{2}, \tag{3.51}$$

$$\hat{m}^{(t)} = \frac{m^{(t)}}{1-\beta_1^{t}}, \tag{3.52}$$

$$\hat{v}^{(t)} = \frac{v^{(t)}}{1-\beta_2^{t}}, \tag{3.53}$$

$$\Delta W = -\eta \frac{\hat{m}^{(t)}}{\sqrt{\hat{v}^{(t)}} + \varepsilon}. \tag{3.54}$$

The exponential decay rate of for the first and second moment estimate typically have a value of $\beta_1 = 0.9$ and $\beta_2 = 0.999$, and the small numerical stabilization constant a value of $\varepsilon = 10^{-8}$.

3.3.9 *Weight update rules*

For the case of gradient descent the full weight update rules are given for input neurons as recurrent neurons as

$$\Delta W_{ji} = -\eta \sum_{t} L_j^{t} \mathcal{F}_{\kappa}\left(\psi_j^{t} \mathcal{F}_{\alpha}\left(z_i^{t-1} \right) \right), \tag{3.55}$$

for recurrent neurons as

$$\Delta W_{ji} = -\eta \left(\sum_t L_j^t \mathcal{F}_\kappa \left(\psi_j^t \mathcal{F}_\alpha \left(x_i^t \right) \right) + c_{\text{reg}} \frac{1}{T n_{\text{trial}}} \left(f^{\text{target}} - \bar{f}_j \right) e_{ji}^{(t)} \right), \tag{3.56}$$

and for output neurons as

$$\Delta W_{kj} = -\eta \sum_t E_k^t. \tag{3.57}$$

3.4 DISCUSSION

In this study, we present an extension to e-prop, a biologically plausible and powerful spike-based three-factor learning rule. As a reference implementation, we embed the new algorithm in the open-source spiking neural network simulator NEST, which is optimized for the distributed simulation of large-scale spiking neural networks. All quantitative data reported in this study were obtained using this implementation. The code[1] is partially available as part of NEST release 3.7 [202], while additional functionality is currently under review[2]. To ensure accessibility for the wider community, we provide comprehensive tutorials[3]. This conceptual and algorithmic work builds upon prior efforts to implement three-factor learning rules in NEST [43, 100] and forms part of a long-term collaborative project aimed at advancing neural systems simulation technology [8].

By adapting e-prop's original synchronous, time-driven weight updates to an asynchronous, event-driven framework, we successfully reproduced two supervised tasks from the original publication. While minor numerical differences between frameworks occasionally trigger an extra spike in one implementation — resulting in a cascade of downstream spikes and cumulative differences in loss — these variations did not affect overall learning success.

Our model generalized well to additional tasks, including the widely used N-MNIST benchmark, and shows potential for extension to other tasks. Porting the algorithm from TensorFlow to NEST and adapting it to NEST's biologically grounded constraints provided valuable insights, such as the necessity of incorporating previously absent connection delays and ensuring that neuron interactions adhered to both locality and causality. The model serves as a foundation for implementing reward-based e-prop [154] and generally three-factor learning rules in NEST. Future development could be streamlined by porting the algorithm to NESTML [9]. Other potential learning rules could include algorithms similar to RTRL, which are practical for neuromorphic hardware [203] and suitable for recurrent networks. One example is, EventProp [204] which has

1 https://github.com/nest/nest-simulator/pull/2867
2 https://github.com/nest/nest-simulator/pull/3207
3 https://nest-simulator.readthedocs.io/en/stable/auto_examples/eprop_plasticity/index.html

been recently implemented in an event-driven manner on the neuromorphic hardware BrainScaleS [205] [BrainScaleS 2, see 206] and SpiNNaker 2 [207] [SpiNNaker 2, see 208]. Additional examples include a spiking BP variant [209], UORO [210], KeRNL [211], KF-RTRL [212], RFLO [213], SnAP [186], DECOLLE [214], OSTL [215], FP [216], ETLP [217], as well as STD-ED and MP-ED [218].

To enhance biological plausibility, we replaced several machine learning components with biologically inspired mechanisms. While batch learning is supported for the event-driven model, this functionality could be extended to the biologically enhanced model for tasks requiring batch processing in the future. Using the N-MNIST task, we optimized learning performance and evaluated individual features for their impact. Despite variations in learning speed, all test errors remained below 0.12 , demonstrating robust learning. Notably, continuous neuron dynamics did not affect performance, and continuous e-prop dynamics had only a marginal effect, suggesting that residual activity from previous samples does not impair learning. Decoupling the eligibility trace filter from the output time constant slightly reduced performance in this task, though complete filter removal improved performance in others. This indicates that the reduction in performance likely stems from the lowered filter factor rather than the decoupling itself. Up-scaling and smoothing surrogate gradient functions, along with using mean-squared error as a biologically plausible alternative to cross-entropy loss, enhanced learning. When combining all these features, updating weights with each spike, and incorporating dynamic firing rate regularization, learning performance was comparable to the baseline model without additional features.

Simulating the N-MNIST task for 300 training iterations and 10 test iterations, corresponding to 2 h 35 min of biological time, requires approximately 3.2 core−h without incorporating biological features. Runtime overheads increased by up to 15 %, with some features having little to no impact. These modest variations in runtime are likely partly due to changes in network dynamics introduced by the additional features, leading to increased firing rates and, consequently, longer simulation times. Such effects can be mitigated by making appropriate parameter adjustments.

The computational accuracy and efficiency results presented here are specific to the N-MNIST task. However, future work could extend validation to encompass a broader range of network architectures, neuron parameters, and learning hyperparameters. Achieving a fair comparison of computational efficiency and accuracy across different frameworks would require fully optimizing the respective frameworks, an area that remains a promising direction for further investigation. The inherent energy efficiency of spike-based computations holds the potential for considerable computational savings. When combined with biologically inspired mechanisms, these features have the potential to inspire advancements in machine learning technologies.

This work contributes to recent efforts to port e-prop to various frameworks and computational substrates. Knight et al. [219] implemented e-prop in mlGeNN [220], a spike-based machine learning library optimized for GPU-based sparse data structures and algorithms provided by the pyGeNN simulator [221]. Their implementation demonstrates the functionality of e-prop on both CPU and GPU platforms. E-prop has also been adapted for neuromorphic hardware. For example, Perrett et al. [222] ported e-prop to SpiNNaker 1 [223], while Rostami et al. [224] implemented it on SpiNNaker 2 [208], and Frenkel et al. [225] realized it in ReckOn. Notably, the SpiNNaker implementations incorporate event-driven weight updates [222, 224].

In contrast to our study, which focuses on faithfully reproducing and extending the original e-prop implementation, these efforts primarily emphasize reproducing the core mechanism and, in some cases, simplifying the algorithm to accommodate hardware constraints [225]. However, none of these studies demonstrate an exact reproduction of the original implementation, and only one provides an explicit accuracy comparison [see 224, Fig. 5]. These works demonstrate e-prop across a variety of tasks. For instance, the pattern generation task has only been reproduced by Perrett et al. [222], while evidence accumulation tasks are addressed by both Perrett et al. [222] and Frenkel et al. [225]. The N-MNIST task has not been demonstrated by others, but Knight et al. [219] trained on a similar dataset, sequential MNIST. Additional tasks include Google Speech Commands [224], DVS gesture recognition [219], spiking Heidelberg digits [220, 225], and a synthetic behavioral dataset [225]. Most of these studies investigate network sizes ranging from several hundred to a few thousand neurons. Only one study explicitly examines scaling properties in terms of processing time (measured in clock cycles) as a function of network size [see 224, Fig. S1].

Our study stands apart in its emphasis on improving the biological plausibility of e-prop while systematically evaluating these enhancements in terms of computational efficiency and accuracy. In contrast, other works primarily focus on optimizing functional performance, energy efficiency, and memory usage, often at the expense of biological plausibility.

This implementation, now integrated into a widely used neural simulator optimized for large-scale networks, equips the research community with a powerful tool to study neuroscientifically relevant tasks, simulate behavioral experiments, and explore learning hypotheses. It has the potential to provide valuable insights into how learning underpins diverse behaviors and neurological disorders, while also driving advancements in machine learning that enable real-world applications.

4 SPIKE-BASED SAMPLING FACILITATED BY OSCILLATORY BACKGROUND ACTIVITY

Author Contributions

Cortical oscillations support sampling-based computations in spiking neural networks. A. Korcsak-Gorzo[*], M. G. Müller[*], A. Baumbach, L. Leng, O. J. Breitwieser, S. J. van Albada, W. Senn, K. Meier, R. Legenstein, M. A. Petrovici (2022) *PLOS Computational Biology*. DOI: 10.1371/journal.pcbi.1009753.

* Shared first authorship.

The author and MGM contributed equally as the joint first authors of the above publication. The publication is included in MGM's doctoral thesis. The results presented in the publication consist of two parts: the first part with current-based spiking networks, contributed by the author, AB, MAP, LL, OJB, SJA, WS, and KM, is included as an excerpt in the results section of this chapter. The second part with conductance-based spiking networks, contributed by MGM and RL, is not included in this thesis.

For the portion of the work included in this thesis, MAP, SJA, and KM served as supervisors. MAP conceived, designed, and conceptualized the project. The software, primarily written by the author with contributions from AB, is an extension of frameworks developed by OJB, LL, and MAP, who also supported the author in using these frameworks. The simulations performed with the software were primarily implemented by the author, with contributions from AB. The methodology was mainly developed by the author, AB, and MAP, with additional contributions from LL and OJB. The formal analysis and investigation were conducted by the author, AB, and MAP. Visualization was performed by the author, AB, and MAP, with MGM contributing to Figure 4.5b and Figure 4.2a. Validation of the work was carried out by the author, AB, LL, OJB, and MAP. Data curation for upload to Zenodo was performed by the author and AB. The original draft of this part of the publication was written by the author, AB, and MAP.

The introduction and discussion sections of this chapter are based on the corresponding sections of the publication, which were jointly written by the author, AB, MAP, MGM, and RL. The entire publication was collaboratively reviewed and edited by the author, MGM, AB, SJA, WS, RL, and MAP.

Early explorations on spike-based tempering are reported in the author's master thesis and AB's doctoral thesis.

The ability to build an internal, predictive model of reality endows an agent with a clear evolutionary benefit. How the mammalian brain accomplishes this feat remains a subject of debate, but the representation of uncertainty certainly plays a role, considering the probabilistic nature of sensory data and uncertainty about past and future events. A good representation of an uncertain reality must allow efficient access to a large variety of plausible beliefs about the environmental state.

Distributions over sensory data, characteristic for natural scenes, are complex in the sense that the coexisting beliefs about the data manifest as numerous deep, dissimilar modes of the state space — one of the many facets of the curse of dimensionality. In probabilistic models of such complex data, exact inference becomes intractable, but the distribution can be approximated by sampling. Rapid convergence towards the target distribution requires the sampler to switch (or mix) between these modes frequently. However, due to their dissimilarity, this switching is notoriously difficult for most sampling methods, an issue which is known as the "mixing problem".

In this manuscript, we put forward a hypothesis for how the brain can efficiently overcome this challenge. In doing so, we unify two aspects of cortical dynamics under a common normative framework: spike-based probabilistic inference and cortical oscillations. Both of these phenomena have been well-studied but have not been explicitly linked in the context of spiking neural networks. In particular, we consider the interpretation of spiking activity in the cortex as probabilistic inference via sampling, which has gained ample experimental [226–228] and theoretical [229–233] support over the last decade. Mathematically, these models are closely related to Gibbs sampling, which tends to get stuck in single states of high probability that act as local attractors.

We propose that this problem of sampling-based representations can be overcome by firing rate oscillations. Firing rate oscillations over multiple frequency bands are a naturally emerging phenomenon in spiking networks [234–237] and have been extensively studied in the mammalian brain [238, 239]. Notably, they appear to play an important role both during awake perception [240–242] and during sleep [243, 244], suggesting a fundamental role in cognition and learning. In previous modeling studies, oscillating changes of neuronal excitabilities have been shown to be beneficial for mixing [245–249], but how such changes might arise on the cellular level within networks of spiking neurons has thus far remained unclear.

We propose that the background firing rate of cortical neurons can be interpreted as a (computational) temperature and can accordingly modify the probability landscape sampled by cortical circuits. If the background activity is oscillatory, the network temperature changes periodically and phase-dependent stationary distributions emerge. By cyclically alternating between "hot" and "cold" periods, cortical networks can effectively instantiate a tempering schedule, with hot phases corresponding to flat probability distributions in which the network can move freely and cold phases representing the multimodal target distribution. This schedule allows networks to escape from local minima and efficiently sample from challenging distributions characterized by multiple high-probability modes separated by large low-probability volumes of the state space.

In this work, we provide an analytical treatment of tempering in spiking networks induced by cortical background oscillations and demonstrate the benefits of this phenomenon in simulations. We explicitly consider current-based synaptic interactions as well as different network architectures. These observations establish a novel connection between multiple observed cortical phenomena, as well as between these experimental findings and normative theoretical models of brain computation.

4.2 RESULTS

To understand how cortical oscillations affect computation at the network scale, we study the behavior of single spiking neurons and networks of spiking neurons under variable levels of background activity. We consider current-based leaky integrate-and-fire (LIF) neurons, for which we can derive analytical expressions for the neuronal response. We show how the level of background input affects the input-output relationship of individual neurons (Section 4.2.1). We then discuss the effect of the background activity on entire networks (Section 4.2.2), where we show that this local increase of stochasticity at the single-neuron level gives rise to corresponding changes of the probability landscape at the network level. In particular, we find that these changes can be parametrized by a Boltzmann temperature parameter. Moving to recurrent networks as models of computation in the sensory cortex, we establish a rigorous interpretation of cortical oscillations as a tempering algorithm (Section 4.2.2). We then demonstrate the functional advantages of such oscillation-induced tempering for generative models of the visual hierarchy trained on two different visual datasets (Section 4.2.3).

4.2.1 *Single-neuron statistics*

Cortical neurons are embedded in a noisy environment (Figure 4.1a). In addition to functional input I_{in}, their many presynaptic partners provide them with an effectively stochastic background [250, 233]. This background activity leads to stochastic single-neuron behavior [251]. To understand this behavior, we consider a simple LIF neuron model with current-based input synapses (see Section 4.3.1). The neuron receives a large number of background inputs, with firing rates ν_i and synaptic efficacies w_i. In line with standard literature, we model this stochastic background input as uncorrelated Poisson spike trains [42]. We first consider the free membrane potential u_{free} of this neuron, that is, the membrane potential in the hypothetical case that there is no firing threshold. The steady-state distribution $p(u_\mathrm{free})$ is well-described by a Gaussian (Figure 4.1b) with moments

$$\mu_u := \mathbb{E}\left[u_\mathrm{free}\right] = E_\mathrm{l} + \frac{I_\mathrm{in} + \sum_i w_i \nu_i \tau_{\mathrm{s},i}}{g_\mathrm{l}}, \tag{4.1}$$

$$\sigma_u^2 := \mathrm{Var}\left[u_\mathrm{free}\right] = \sum_i \nu_i w_i^2 \frac{\tau_{\mathrm{s},i}^2}{2 g_\mathrm{l}^2 \left(\tau_\mathrm{m} + \tau_{\mathrm{s},i}\right)}, \tag{4.2}$$

where E_l and g_l are the leak potential and conductance, and τ_m and τ_s are the membrane and synaptic time constants [see Section 4.3 in 252]. Here, $\sum_i$ runs over all background presynaptic partners. Note that excitatory and inhibitory inputs (defined by the sign of the synaptic weight w_i) can cancel each other out in the mean but always add up towards the variance of the free membrane potential distribution.

Upon introducing a firing threshold, some portion of the free membrane potential probability density will lie above it, causing the neuron to spike stochastically. The shape of the neuronal response function, i.e., the firing rate in response to a constant input current I_in, depends strongly on the characteristic time constant of the neuronal membrane. Cortical neurons under strong presynaptic bombardment have been shown to operate in a high-conductance regime [40], which greatly reduces the effective membrane time constant τ_m. Under such conditions, the neuronal response function (Figure 4.1c) can be well approximated by a logistic function [232]:

$$\nu_\mathrm{out}(I_\mathrm{in}) = \frac{1}{1 + \exp\left[-\beta\left(I_\mathrm{in} - I_0\right)\right]}. \tag{4.3}$$

Hence, the neuron's stochastic response is characterized by two parameters, the offset I_0 and the slope β of the sigmoid. Both of these depend on the background activity. The response function can be intuitively understood as the area under the free membrane potential distribution that lies above the firing threshold. Thus, its shape is similar to the integral of $p(u_\mathrm{free})$, its offset I_0 has a similar linear dependence on μ_u, and its slope parameter β will decrease for increasing σ_u. Their exact dependence on the background rates is shown in Figure 4.1d and Figure 4.1e. In particular, the relationship between the slope of the response function and the standard deviation of the free membrane potential distribution is well-approximated by a linear function, which allows us, in turn, to establish the relationship between the slope parameter β and the total (i.e., summing over all background presynaptic partners) excitatory and inhibitory background firing rates ν_exc and ν_inh and the corresponding weights w_exc and w_inh using Equation 4.2:

$$\frac{1}{\beta} \propto \sigma_u \propto \sqrt{w_\mathrm{exc}^2 \nu_\mathrm{exc} + w_\mathrm{inh}^2 \nu_\mathrm{inh}}. \tag{4.4}$$

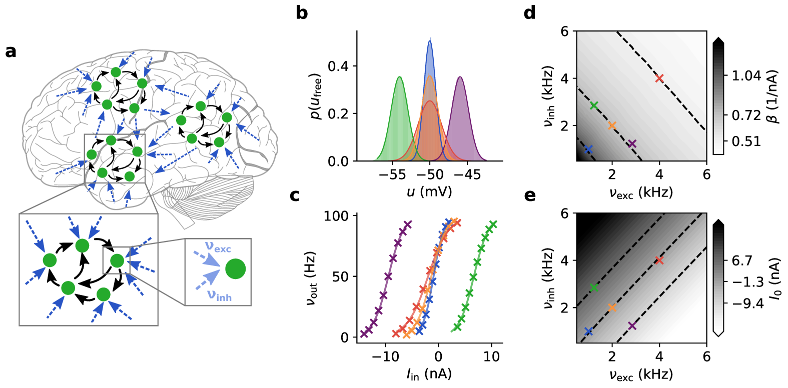


Figure 4.1: **Response functions of neurons in an ensemble. (a)** Cortical ensemble of networks. The spike input received by a neuron can be partitioned into functional (solid black arrows) and background (dashed dark blue arrows) input. The background can be partitioned into an excitatory and an inhibitory subset (dashed light blue arrows). In the following panels, we consider one such neuron under five different illustrative background regimes, each of which is assigned a specific color. **(b)** Steady-state free membrane potential distributions. Shaded areas: numerical simulation; solid lines: analytical approximation using Equation 4.1 and Equation 4.2. Purple, orange, green: same σ_u, different μ_u; blue, orange, red: same μ_u, different σ_u. **(c)** Corresponding neuronal response functions. Crosses: numerical simulation; solid lines: logistic fit with Equation 4.3. **(d)** Slope parameter β and **(e)** offset I_0 of response functions under various background regimes defined by their respective pairs of excitatory and inhibitory input rates $(\nu_{\mathrm{exc}}, \nu_{\mathrm{inh}})$. Dashed isolines indicate configurations of constant slope (cf. Equation 4.4) or offset, with specific values given as colorbar ticks. Note the approximate linearity of the contour lines.

To summarize, we have established how the stochastic response of individual LIF neurons depends on the level of background input. In particular, the background input determines the slope β of the (logistic) neuronal response function.

4.2.2 *Temperature in spiking networks*

As discussed in Section 4.2.1, under Poisson background activity, individual neurons react to their input stimulus in a well-defined stochastic manner. Based on this result, we show here how the level of background activity influences the stochastic properties of a recurrently connected network of LIF neurons (Figure 4.2a, for the network setup see Section 4.3.5).

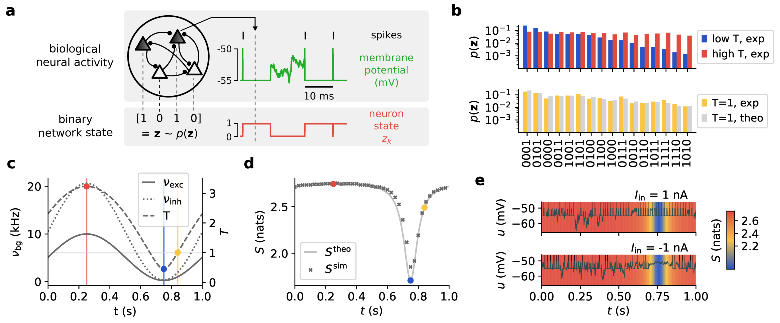


Figure 4.2: **Effects of oscillatory background activity on sampled distributions. (a)** Network dynamics. Each neuron encodes a binary random variable according to its refractoriness. When the membrane potential (green) is clamped to the reset value, the neuron state (red) is considered to be $z = 1$ ($z = 0$ otherwise). The collection of the resulting network states $\mathbf{z}$ forms an estimate for the implemented probability distribution $p(\mathbf{z})$. **(b)** Distributions sampled by a 4-neuron network at the three temperatures marked in **(c)**. States are ordered according to their respective probabilities at the low temperature to emphasize the effect of tempering visually. **(c)** Time course of excitatory and inhibitory background rates (dashed and dotted lines, Equation 4.10), along with the associated temperature (solid line, Equation 4.4). Note that ν_{exc} is scaled by 0.5 and w_{exc} by the square root of the inverse scaling factor to demonstrate that balance is independent of such a rescaling. **(d)** Simulated (crosses) vs. calculated (solid line) entropy course $S(t)$. The slight lag is due to the finite relaxation time constants $\tau_{\text{s}}, \tau_{\text{ref}}$ of the network (Equation 4.26 only holds strictly for quasi-static temperature changes). **(e)** Effect of tempering on individual membrane potentials and spiking activity. The background color represents the corresponding entropy.

In a spiking network, the information conveyed by a neuron at any point in time can be described as binary: the neuron either spikes or it does not. A spike has a twofold effect: it initiates a refractory period and elicits postsynaptic potentials (PSPs) in postsynaptic partner neurons. We can therefore view the binary state z of a neuron being refractory ($z = 1$) or not ($z = 0$) following a spike as corresponding to the state communicated to its downstream partners [229, 232, see Figure 4.2a]. Thus, each neuron can be interpreted as sampling from the conditional distribution $p(z_k = 1|\mathbf{z}_{\setminus k})$, i.e., the probability of the k^{th} neuron to be in the state "1" given the states of all other neurons $\mathbf{z}_{\setminus k}$.

In general, the joint distribution sampled by the network cannot be given in a closed form. To allow an analytical approach, we begin with a set of assumptions about the neuron and network parameters (see Section 4.3.1 and Section 4.3.4). For parameters emulating a high-conductance state [40] the activity of an LIF network can be interpreted as sampling from a joint Boltzmann distribution [232]

$$p_T(\mathbf{z}) \propto \exp\left[-E(\mathbf{z})/(k_\mathrm{B}T)\right], \tag{4.5}$$

where $E(\mathbf{z}) = -\frac{1}{2}\sum_{kj} W_{kj}z_k z_j - \sum_k B_k z_k$ represents the energy of a particular joint state $\mathbf{z}$, with W_{kj} denoting effective recurrent synaptic weights and B_k effective individual neuron biases. Here, k_B is the Boltzmann constant and T is the ensemble (Boltzmann) temperature. For such a network state distribution, the state probability of each neuron k is given by

$$p(z_k = 1|\mathbf{z}_{\setminus k}) = \frac{1}{1 + \exp\left(-\frac{\sum_{i\neq k} W_{ki}z_i + B_k}{k_\mathrm{B}T}\right)}, \tag{4.6}$$

as expressed in Petrovici [252] and Buesing et al. [229]. Note that this equation has the same form as the neuronal response function in Equation 4.3.

Since weights and biases can both be interpreted as movements along the I_in-axis of the neuronal response function, their simultaneous multiplicative scaling by T is equivalent to a horizontal stretching of the response function. This similarity allows us to identify

$$\beta = 1/(k_\mathrm{B}T), \tag{4.7}$$

again analogous to statistical physics, for a Boltzmann constant k_B that relates the (unitless) reference temperature $T = 1$ to a chosen set of neuron and background parameters via the resulting response function (here, the unit of k_B is nA). Note that the Boltzmann parametrization with unitless weights $\mathbf{W}$ and biases $\mathbf{B}$ in Equation 4.6 is different from the synaptic weights and biasing effects in Equation 4.3 induced by leak, threshold potentials, unbalanced input etc. in the LIF domain, but they can be linearly mapped such that the sampling distributions match (see Equation 4.24 and Equation 4.25). Equation 4.4 and Equation 4.7 thus establish an exact relationship between the ensemble temperature and the background firing rates:

$$T \propto \sqrt{w_\mathrm{exc}^2 \nu_\mathrm{exc} + w_\mathrm{inh}^2 \nu_\mathrm{inh}}. \tag{4.8}$$

In order to study the effect of pure temperature variations without affecting neuronal offsets, excitatory and inhibitory background rates need to be balanced. Such a balance is also well-documented in vivo [253, 254]. Note that this is not simply achieved by setting $w_{\text{exc}} \nu_{\text{exc}} \tau_{\text{exc}} = w_{\text{inh}} \nu_{\text{inh}} \tau_{\text{inh}}$ and thus effectively equalizing the effects of excitation and inhibition; while this would leave μ_u unchanged, it would still affect I_0 (cf Figure 4.1b and Figure 4.1c). A balanced regime can be achieved by a linear dependence between firing rates (Section 4.3.2), following one of the isolines in Figure 4.1e, which are well approximated by

$$\nu_{\text{inh}} = \nu_0 + m \nu_{\text{exc}}. \tag{4.9}$$

The exact parameters ν_0 and m that are necessary for balance depend on the synaptic time constants and background input weights (see Section 4.3.3). Following such an isoline then results in a constant I_0 and a $\sqrt{\nu}$ dependence of the (inverse) slope parameter $1/\beta$ (see Section 4.3.2). While this approach enables a strict realization of a Boltzmann temperature, the achieved effect does not rely strongly on such a balance. Following Equation 4.4 we can maintain the balance if we rescale the ν_{exc} and multiply w_{exc} by the square root of the scaling factor, which we apply in Figure 4.2.

With this definition of temperature, we now turn to its effects on the distribution. In Equation 4.6, the ensemble (Boltzmann) temperature T scales the effective weights and biases multiplicatively, identically to its effect in statistical physics: as the temperature of an ensemble rises, particle interactions (here: synaptic weights) and external fields (here: neuronal biases) become increasingly inconsequential.

We can observe a similar effect on the sampled distribution when modulating the temperature implemented by the background input (see Figure 4.2b): at high temperatures, the distribution becomes flat, while at low temperatures, the high-probability maxima become even more pronounced. Cyclic heating and cooling — enabled here by oscillatory background — can thus alternate between hot phases with equalized state probabilities and cold phases for reading out the most relevant samples of the correct distribution, where the sampled distribution approximates the target distribution most closely in the $T = 1$ crossings (see Figure 4.3 for the divergence during one cycle). Such a cycle is often referred to as tempering. We consider a simple sinusoidal oscillation as a basis function for modeling cortical oscillations:

$$\nu_{\text{exc}}(t) = \frac{\nu_{\max} - \nu_{\min}}{2} \sin\left(2\pi f_{\text{osc}} t\right) + \frac{\nu_{\max} + \nu_{\min}}{2}, \tag{4.10}$$

with minimum rate $\nu_{\min}$, maximum rate $\nu_{\max}$, and oscillation frequency f_{osc}. This time course implicitly also defines $\nu_{\text{inh}}(t)$ through Equation 4.9, such that in this setup, excitation and inhibition vary synchronously (see Figure 4.2c), as observed in vivo [see, e.g., 255]. Note that the network activity follows the instantaneous level of balanced background input in the label layer (Figure 4.4a), visible layer (Figure 4.4b), and hidden layer (Figure 4.4c), which is also reflected in the summed activity across all layers (**(d)**).

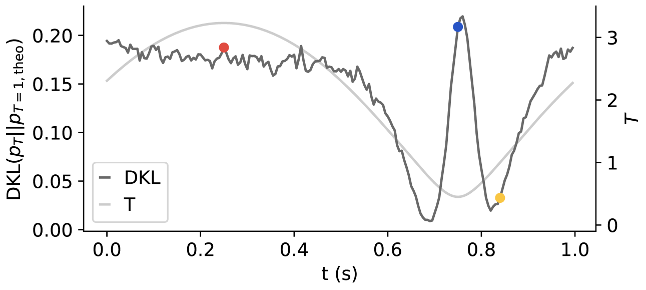


Figure 4.3: **Divergence from the target distribution during one oscillation period.** Time course of the DKL to the target distribution together with the time course of the temperature demonstrated at the network in Figure 4.2. The DKL is high for both high temperatures (red dot, quasi-uniform distribution) and low temperatures (blue dot, quasi-single state distribution), indicating that the distributions at these temperatures differ. The divergence is small at the two crossings of $T = 1$, indicating high fidelity representations. The yellow dot indicates the time of the readout.

The resulting temperature thus also varies periodically, with the square root of a sine (see Figure 4.2c and Equation 4.8). Moreover, the ensemble temperature controls the entropy of the sampled distribution, which effectively describes the "disorder" of the network and corresponds to the uniformity of the sampled distribution. For higher temperatures, as the sampled distribution becomes more uniform, the entropy increases (Figure 4.2d). In high-temperature/high-entropy states, membrane potentials are extremely noisy, causing neurons to fire randomly and independently. In contrast, in low-temperature/low-entropy states, membrane potentials are nearly constant, and neurons are "frozen" in certain states, firing either persistently or not at all (Figure 4.2e).

4.2.3 *Mixing in high-dimensional multimodal data spaces*

In the following, we discuss the computational role of background oscillations for spiking networks trained to represent complex distributions over high-dimensional visual data. Here, we have chosen two commonly used visual datasets to serve as examples, but our conclusions hold for arbitrary distributions. As a simplified model of cortical visual hierarchy, we consider recurrent layered spiking networks consisting of LIF neurons, which we train as simultaneous generative and discriminative models (Figure 4.5a). These two forms of computation happen concurrently and bi-directionally: the label neurons classify the state of the visible layer, while the visible neurons adapt their states

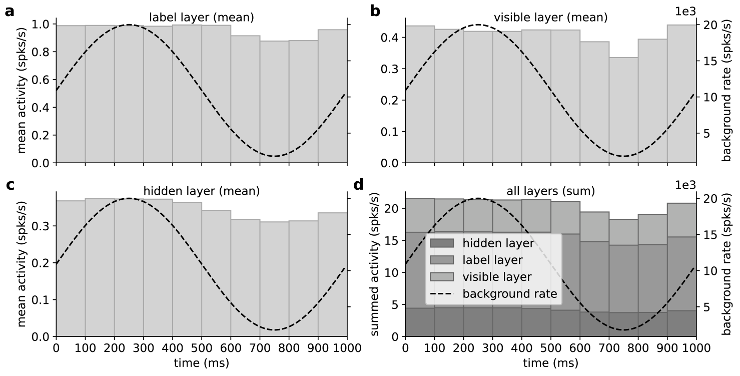


Figure 4.4: **Layerwise spike activity.** Spike activity in the label layer (**(a)**), visible layer (**(b)**), hidden layer (**(c)**), and summed across all layers (**(d)**) of the NORB network in Figure 4.5 is shown as a function of the phase of the background oscillation. The mean firing rate per neuron oscillates in all three layers, synchronized with the phase of the background oscillation.

to produce images that are compatible with the class represented by the label layer. For each class, during the preceding training, probability mass was built up in the corresponding region of the probability landscape, forming the modes of the network.

High-dimensional but well-recognizable visual data confronts such networks with two contradictory challenges. On the one hand, they need to produce good samples, i.e., clean images corresponding to particular sharp high-probability modes separated by large vanishing-probability volumes of the state space that correspond to out-of-distribution samples. On the other hand, they need to be able to switch between different modes in order to sample from the target distribution fully; this is at fundamental odds with the probability landscape described above. This so-called mixing problem is well-known and quasi-ubiquitous for sampling models.

One solution to this problem was proposed by Marinari et al. [256] in the context of Markov-chain Monte Carlo sampling for Ising models, which is intimately related to our form of spike-based sampling in both dynamics and sampled distribution [232]. This simulated tempering method describes a cyclic heating and cooling schedule reminiscent of the periodic temperature variation induced by cortical oscillations discussed above (Equation 4.10). In-between readouts at the reference temperature, a temporary rise in temperature flattens the probability landscape, allowing the network to escape from local attractors. Thus, Equation 4.5, Equation 4.4, and Equation 4.10 establish a rigorous analogy

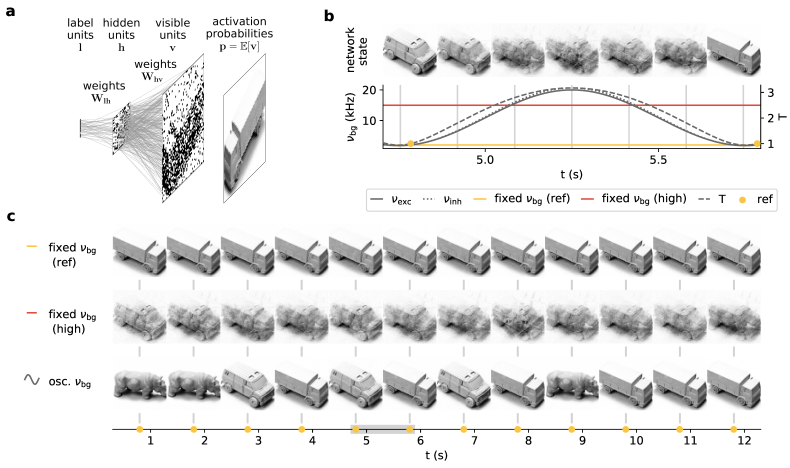


Figure 4.5: **Background oscillations improve generative properties of spiking sampling networks. (a)** Architecture of a hierarchical 3-layer (visible $\mathbf{v}$, hidden $\mathbf{h}$ and label $\mathbf{l}$) network of LIF neurons and example layerwise activity. For a better representation of the visible layer statistics, we consider neuronal activation probabilities $p(\mathbf{v}|\mathbf{h})$ rather than samples thereof, to speed up the calculation of averages over (conditional) visible layer states. Here, we show a network trained on images from the NORB dataset. **(b)** Evolution of the activation probabilities of the visible layer (top) over one period of the background oscillation (bottom). **(c)** Evolution of the visible layer over multiple periods of the oscillation compared to a network with constant background input at the reference rate (2 kHz, top) and at a high rate (10 kHz, middle), cf. also yellow and red lines in **(b)**. The activation probabilities are shown whenever the reference rate (see panel **(b)**) is reached. The gray bar denotes the period shown in **(b)**.

between simulated tempering and cortical oscillations, which thereby take on the computational role of enabling mixing in challenging real-world scenarios.

To evaluate these effects, we considered two example scenarios based on well-studied visual datasets: NORB [257] and MNIST [258]. Network training was done using a variant of wake-sleep learning [259], a contrastive Hebbian scheme inspired by biological phenomenology and widely used for sampling models (see in particular Leng et al. [260]). A background rate of $\nu_{\text{exc}} = \nu_{\text{inh}} = 2\,\text{kHz}$ was chosen as reference, implicitly defining the reference temperature $T = 1$.

For visual datasets, the weakened correlations at higher temperatures correspond to blurred images. For the network trained on NORB (Section 4.3.6), this is particularly well observable (cf. Figure 4.5b). The network produces sharp images at low background rates, whereas the images become blurred under increased background activity. Note especially how the network enters a superposition of several "clean" states at higher background rates. Constant background stimulus cannot reproduce the ease of switching between different image classes (modes). The network is either stuck in one mode while producing sharp images ($T = 1$ upper row in Figure 4.5c) or only able to produce blurred images ($T = 2.5$ middle row in Figure 4.5c). Tempering through background oscillations effectively combines these two regimes, allowing a better sampling of the target distribution at phases where the reference temperature is reached (lower row in Figure 4.5c).

The effectiveness of this tempering schedule depends on the parameters of the background oscillations: ν_{min}, ν_{max}, and f_{osc}. In particular, the frequency f_{osc} plays a critical role, as it represents a tradeoff between exploration and exploitation of the network's state space. Low frequencies guarantee that the network has time to relax towards its momentary stationary distribution p_T, with $f_{\text{osc}} \to 0$ representing the quasi-static limit, i.e., constant background. This enables accurate sampling from the target distribution at $T = 1$, as the network loses memory of previous states occupied at higher temperatures. However, lower oscillation frequencies come at the cost of slower sampling, as they increase the time between consecutive readouts. Furthermore, frequencies significantly lower than 0.1 Hz are rarely observed in vivo [239]. In the following, we study the behavior of spiking sampling networks under different background oscillation regimes for a network trained on handwritten digits from the MNIST dataset (for the network setup see Section 4.3.7).

Two essential quality criteria for any sampling network are its mixing speed and sample fidelity. In principle, Equation 4.5 allows an analytical evaluation of these properties, but in practice, this is unfeasible for high-dimensional distributions. We, therefore, use a sample-based measure, the indirect sampling likelihood (ISL, see Breuleux et al. [261] and Section 4.3.9). The ISL accumulates fidelity values for all generated samples, assigning high values if they are similar to images in the test set and low values otherwise. Additionally, the rate at which the ISL increases over time implicitly represents a measure of the mixing speed. We use the distribution of times between label switches as a more explicit measure of mixing times for different image categories.

Our MNIST-trained network allows a quantitative evaluation of the benefits of oscillation-induced tempering. In each tempering cycle, around $T = 1$, one digit stabilizes in the visible layer for a wide time window (see Figure 4.6). The corresponding network mode is defined by the label neuron with the highest probability inferred from the hidden layer activity. With oscillatory background (Figure 4.8a), the sampled digits and labels change more frequently as compared to constant background (Figure 4.8b). Consequently, the average mode duration (see Section 4.3.10), as defined by the time interval between two mode switches, is shorter for oscillatory background (compare Figure 4.8c and Figure 4.8d). Since frequent mode switches are essential to efficiently cover the target distribution, the Kullback-Leibler divergence (DKL, see Section 4.3.8) between the target and sampled distribution also decreases more rapidly with oscillatory background (Figure 4.8e). Furthermore, the ISL converges to higher values compared to the constant background (Figure 4.8f), which indicates an overall better tradeoff between generating clear examples of the imprinted classes and good mixing between these classes (also see Figure 4.10). Note that tempering can likewise improve the network's performance in inference tasks (Figure 4.7).

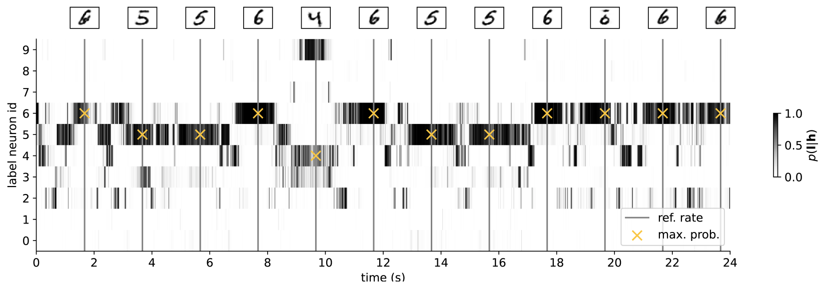


Figure 4.6: **Probability of the label layer.** Exemplary time course of the inferred activity per label neuron over time (lower plot) and the associated state of the visible layer (top bar) of the MNIST network in Figure 4.8. Spike probability is high and unique during the low activity phases (around the $T = 1$ readout, vertical lines) and lower and distributed over several labels during the high activity phases. The network is typically in a stable response state for a certain time window around the readout. The length of this time window depends on the depth of the modes.

Next, we studied tempering under a range of biologically plausible regimes, with background rates (per neuron) varying between 0.5 and 30 kHz and oscillation frequencies ranging from the alpha range to the first slow-wave band [238]. In the landscapes over the mode durations (Figure 4.8g) and the ISLs (Figure 4.8h), we find that the most important prerequisite for effective tempering is the maximum background rate, as the temperature between readouts has to be high enough for frequent mode switches. For our networks, this required

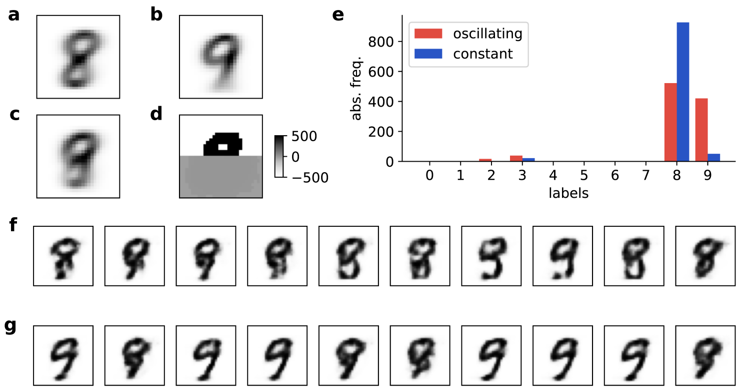


Figure 4.7: **Inference task with ambiguous input.** Superposition of the first 5421 images of class 8 **(a)** and class 9 **(b)** of the MNIST training data set. **(c)** Superposition of images in **(a)** and **(b)**. **(d)** Biases of the network to clamp visible layer to the upper part of the image in **(c)** and emulate an ambiguous input. **(e)** Distribution over the inferred labels of the MNIST network from Figure 4.8 in a 100-cycles run averaged over ten random seeds. The imprinted labels 8 and 9 dominate the distribution — the posterior distribution — illustrating the uncertainty of the input. With oscillating background input, the distribution is more balanced. Thus, oscillations can help in inference tasks. Note that the network simultaneously completes the lower part of the ambiguous input image in the visible layer — shown as the inferred visible layer activity for constant background in **(f)** and oscillating background in **(g)**.

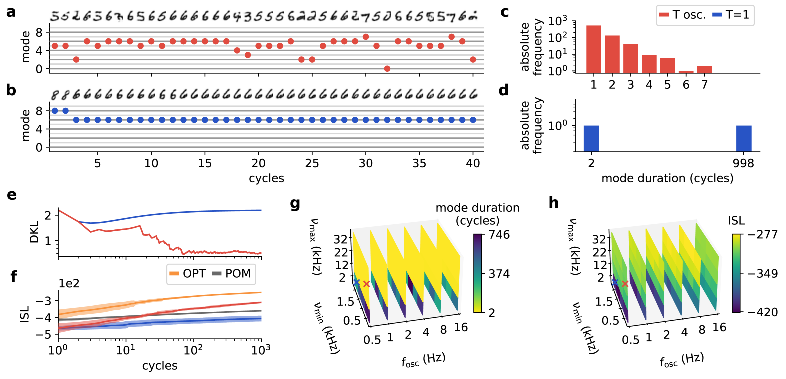


Figure 4.8: **Parameter dependence of tempering effectiveness. (a, c)** Visible and label layer activity of an LIF network trained on the MNIST dataset, with **(b, d)** showing the corresponding mode duration distributions (the active mode corresponds to the image class and is determined by the most active label neuron). The network with oscillatory background (red) moves quickly between modes, with correspondingly short mode durations, whereas the network with constant background activity (blue) switches to the 6 mode after two samples and remains there until the last of the 10^3 collected samples. **(e)** Kullback-Leibler divergence (DKL) between the distribution of sampled modes and the uniform distribution. The sampled distribution quickly becomes significantly more uniform for the oscillatory (red) compared to the constant (blue) background. **(f)** Indirect sampling likelihood (ISL) as a measure of image quality and diversity for the two background settings and, for orientation, for the optimal sampling (OPT, orange) and the product of marginals (POM, gray). Under this measure, the averaged MNIST images described by the POM are more similar to the entire dataset than the near-unimodal distribution generated under constant background at $T = 1$. Similarly, the network with oscillatory background needs several samples to produce a distribution that is diverse enough to overtake the POM. The mean (solid lines) and standard deviation (shades) over 10 runs of 10^3 samples are plotted. **(g)** Average mode duration for different oscillation parameters: The peak background rate $\nu_{\max}$ represents the most critical parameter and needs to be high enough to enable good mixing. The minimum background rate $\nu_{\min}$ and the oscillation frequency f_{osc} are less important. **(h)** Same as **(g)** for the ISL values. The image quality remains consistently high across a wide range of parameter configurations. The data used for **(a-f)** corresponds to the simulations marked by the red and blue crosses, respectively. Values represent averages over 10 runs of 10^4 samples.

input rates above 10 kHz (Figure 4.8g and Figure 4.8h). On the other hand, the minimum background rates in the cold phases have a much smaller influence. In general, effective tempering is achieved over a wide range of oscillation parameters (yellow and light green areas in Figure 4.8g and Figure 4.8h) covering all studied frequency bands. Overall, the best performance was achieved in the slow-wave regime.

4.3 METHODS

4.3.1 *Neuron model*

The membrane potential u of a current-based leaky integrate-and-fire (LIF) neuron evolves according to

$$C_\mathrm{m}\frac{\mathrm{d}u}{\mathrm{d}t} = g_\mathrm{l}(E_\mathrm{l} - u) + I(t), \tag{4.11}$$

with membrane capacitance C_m, leak potential E_l and leak conductance g_l. The resulting membrane time constant is $\tau_\mathrm{m} = C_\mathrm{m}/g_\mathrm{l}$. When the membrane voltage u reaches a threshold value v_th from below, a spike is emitted and the membrane potential is fixed to a reset value $v_\mathrm{reset} \leq v_\mathrm{th}$ for the refractory time τ_ref (see Table 4.1). The input current $I(t)$ is a sum of synaptic currents

$$I(t) = I_\mathrm{rec}(t) + I_\mathrm{in}(t) + I_\mathrm{bg}(t), \tag{4.12}$$

where we distinguish between functional input I_rec, synaptic background input I_bg and any other form of bias input I_in (see Figure 4.1a). Assuming exponential synaptic kernels, the input current obeys

$$\frac{\mathrm{d}I}{\mathrm{d}t} = \frac{I_\mathrm{in} - I}{\tau_\mathrm{s}} + \sum_j w_j S_j(t), \tag{4.13}$$

where w_j and τ_s respectively denote the synaptic weight and time constant. The sum goes over all presynaptic spike sources j, including both background and recurrent input, with the corresponding spike trains $S_j(t) = \sum_f \delta\left(t - t_j^{(f)}\right)$, where $t_j^{(f)}$ denotes the f^{th} spike time of spike source j.

Table 4.1: **Neuron parameters.**

C_m	g_l	E_l	τ_exc	τ_inh	v_th	v_reset	τ_ref
pF	nS	mV	ms	ms	mV	mV	ms
200	2000	-50	10	10	-50	-55.1	10

Without loss of generality, we endow each neuron with a single excitatory and a single inhibitory Poisson source characterized by rates ν_exc and ν_inh and corresponding connection strengths w_exc and w_inh.

The resulting distribution of the free membrane potential u_{free} (no spiking, $\vartheta_{\text{th}} \to \infty$) is well described by a Gaussian with moments given by Equation 4.1 and Equation 4.2 [for more details see Section 4.3 in 252]. In general, more background input, originating from either larger weights $w_{\text{exc}}, |w_{\text{inh}}|$ or higher frequencies $\nu_{\text{exc}}, \nu_{\text{inh}}$, increases the variance. The resulting neuronal response function can be calculated from this distribution using a recursive approach [232]. In the high-conductance state [40], the membrane time constant becomes small, leading to a more symmetric response function, which is well-approximated by a logistic function (Equation 4.3). In the interpretation of spiking neurons as binary random variables, the neuronal response becomes an expression for the conditional probability of a neuron to be in state "1" given the states of its presynaptic partners $p(z_k = 1|\mathbf{z}_{\setminus k})$. Neuron parameters are given in Table 4.1.

4.3.2 *Spike response of sampling neurons*

In the experiments underlying Figure 4.1, we connect a current-based sampling neuron with one excitatory and one inhibitory Poisson source with weights w_{exc} and w_{inh}, where $w_{\text{exc}} = -w_{\text{inh}}$, and vary the corresponding firing rates ν_{exc} and ν_{inh}. Background input parameters are listed in Table 4.2. We can freely choose the mapping of background rates to the Boltzmann temperature. For simplicity, we chose $T = 1$ in the lower range of physiological values, such that the readout can happen at low points in the oscillation cycle:

$$T = 1 \iff \nu_{\text{exc}} = \nu_{\text{inh}} = 2\,\text{kHz}, \tag{4.14}$$

which results in a slope of $\beta = 1.39\,\text{nA}^{-1}$ and an offset of $I_0 = 1.34\,\text{nA}$ (see Figure 4.1d and Figure 4.1e). Since shifting the offset implies a change of the neuronal bias, we only have one degree of freedom when changing the temperature T. We, again arbitrarily, choose the excitatory rate ν_{exc}. The relationship $\nu_{\text{inh}} = h(\nu_{\text{exc}})$ is then found by interpolating the measured response functions from Figure 4.1e. In practice, this function can be approximated with the linear fit in Equation 4.9 with $\nu_0 = -0.13\,\text{kHz}$, $m = 1.04$ and the coefficient of determination $r^2 = 0.999\,98$ (Figure 4.9a). Choosing inhibitory and excitatory rate combinations along this line, keeps the offset current constant and varies solely the temperature (Figure 4.9b), which results into response functions with constant inflection point and varying slope (Figure 4.9c).

The five explicitly marked background configurations shown in Figure 4.1b to Figure 4.1e are given in Table 4.3.

4.3.3 *Temperature as a function of background rates*

The relationship between our temperature definition and the background rates can be approximated by linking the probability density function of the membrane

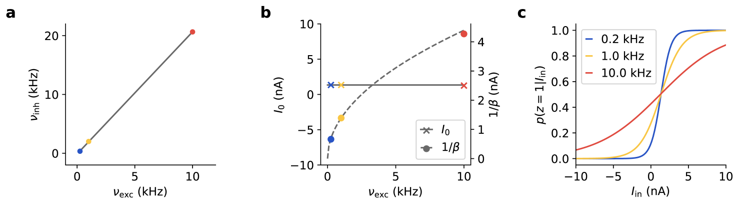


Figure 4.9: **Changing the temperature of the system.** The linear relationship $\nu_{\mathrm{inh}}(\nu_{\mathrm{exc}})$ in **(a)** can keep the offset of the response function constant (solid line in **(b)**), while only changing the slope of activation functions and thereby the temperature of a network (dashed line line in **(b)**). This relationship also reflects the relative strengths of afferent excitatory and inhibitory weights. **(c)** Three example activation functions with constant offset for different background rates. Colors correspond to those in panel **(a)**. Here, we emphasize the binary-state interpretation by plotting $p(z = 1|I_{\mathrm{in}}) = \nu_{\mathrm{out}}\tau_{\mathrm{ref}}$ (cf. Figure 4.1c).

Table 4.2: **Hierarchical network parameters.** For deep networks, the number of neurons, N_{nrns}, is specified separately for the visible, hidden, and label layers.

symbol	unit	response function	entropy	NORB	MNIST
		Figure 4.1	Figure 4.2	Figure 4.5	Figure 4.8
N_{nrns}		1	4	3600, 500, 10	784, 400, 10
$\nu_{\mathrm{exc,min}}$	kHz	0.5–30	0.25	0.5	0.5
$\nu_{\mathrm{exc,max}}$	kHz	0.5–30	10	20	22
$\nu_{\mathrm{inh,min}}$	kHz	0.4–31	0.1	0.4	0.4
$\nu_{\mathrm{inh,max}}$	kHz	0.4–31	10.3	20.6	22.7
f_{osc}	Hz	const.	1	1	2
w_{exc}	nA	0.5	1.0	0.5	0.5
w_{inh}	nA	-0.5	-0.5	-0.5	-0.5

Table 4.3: **Background parameters for the colored response functions and membrane potential distributions in Figure 4.1.**

color	ν_{exc} kHz	ν_{inh} kHz
blue	1.000	1.000
orange	2.000	2.000
red	4.000	4.000
green	2.848	1.232
purple	1.232	2.848

potential to the derivative of the logistic response function. In the diffusion approximation, the free membrane potential distribution is Gaussian:

$$f\left(u; \mu_u, \sigma_u\right) = \frac{1}{\sqrt{2\pi}\sigma_u} \exp\left(-\frac{\left(u-\mu_u\right)^2}{2\sigma_u^2}\right). \tag{4.15}$$

In the high-conductance state, the cumulative distribution function (CDF) has a very similar shape to the (logistic) response function (Equation 4.3). In particular, they have approximately the same derivative at their inflection point [for details, see 232]. With the parameter transformation $u_{\text{in}} = I_{\text{in}}/g_{\text{l}}$ and $\beta = \beta_u g_{\text{l}}$, where β_u is the slope in the potential domain, the response function reads:

$$\nu_{\text{out}}(u_{\text{in}}) = \frac{1}{1+\exp\left(-\beta u_{\text{in}}/g_{\text{l}}\right)}. \tag{4.16}$$

The slope of the CDF at its inflection point is

$$\partial_u \mathrm{F}|_{u=0} = f|_{u=0} = \frac{1}{\sqrt{2\pi}\sigma_u}, \tag{4.17}$$

whereas for the activation function it is

$$\partial_{u_{\text{in}}} \nu_{\text{out}}|_{u_{\text{in}}=0} = \left.\frac{\beta \exp\left(-\beta u_{\text{in}}/g_{\text{l}}\right)}{g_{\text{l}}(1+\exp\left(-\beta u_{\text{in}}/g_{\text{l}}\right))^2}\right|_{u_{\text{in}}=0} = \frac{\beta}{4 g_{\text{l}}}. \tag{4.18}$$

Equating the two creates a direct correspondence between the inverse temperature β and the width σ_u of the free membrane potential distribution:

$$\beta\left(\sigma_u\right) \approx \frac{4 g_{\text{l}}}{\sqrt{2\pi}\sigma_u}. \tag{4.19}$$

In our case, with $w_{\text{exc}} = w_{\text{inh}}$, $\tau_{\text{exc}} = \tau_{\text{inh}}$ and $1/k_{\text{B}} = \beta_{\text{ref}}$, plugging in the expression for σ_u from Equation 4.2, the more precise expression for the temperature in Equation 4.7 is given by

$$T = \frac{\beta_{\text{ref}}}{\beta} = \sqrt{\frac{\nu_{\text{exc}} + \nu_{\text{inh}}}{\nu_{\text{exc,ref}} + \nu_{\text{inh,ref}}}}, \tag{4.20}$$

where $\nu_{\mathrm{exc,ref}}$ and $\nu_{\mathrm{inh,ref}}$ are the excitatory and inhibitory reference rate corresponding to $T = 1$.

4.3.4 *Entropy of spiking sampling networks*

Networks of current-based LIF neurons can sample, to a very good approximation, from binary Boltzmann distributions

$$p(\mathbf{z}) = \frac{1}{Z} \exp\left(\frac{-E(\mathbf{z})}{k_{\mathrm{B}}T}\right), \tag{4.21}$$

with energy function

$$E(\mathbf{z}) = -\frac{1}{2}\sum_{k,j} W_{kj} z_k z_j - \sum_k B_k z_k. \tag{4.22}$$

where $\mathbf{W}$ is a symmetric zero-diagonal matrix and $\mathbf{B}$ a bias vector [232]. The associated neuronal response function represents a conditional state probability and reads

$$p(z_k = 1|\mathbf{z}_{\backslash k}) = \frac{1}{1 + \exp\left(-\sum_j W_{kj} z_j - B_k\right)}. \tag{4.23}$$

The synaptic strength w_{kj} and input current $I_{\mathrm{in},k}$ in the equivalent LIF network can be related to the Boltzmann parameters W_{kj} and B_k via the slope of the response function β (cf. Equation 4.3):

$$w_{kj} = \frac{W_{kj}}{\beta} \frac{g_{\mathrm{l}}(\tau_{\mathrm{s}} - \tau_{\mathrm{m}})}{\tau_{\mathrm{s}}(1 - \exp(-1)) - \tau_{\mathrm{m}}\left(1 - \exp\left(\frac{\tau_{\mathrm{ref}}}{\tau_{\mathrm{m}}}\right)\right)}, \tag{4.24}$$

$$I_{\mathrm{in},k} = \frac{B_k}{\beta} + I_{\mathrm{o}}. \tag{4.25}$$

Biases are implemented via a shift of the leak potential E_{l}. The entropy is given by

$$S(p_T) = \sum_{\mathbf{z}} -p_T(\mathbf{z}) \log p_T(\mathbf{z}) . \tag{4.26}$$

Depending on the base of the logarithm, the unit of S is either nats or bits.

4.3.5 *Parameters of the Boltzmann distribution*

For the entropy scaling in Figure 4.2, we use a 4-neuron network (for the layer sizes, see Table 4.2) with random weights and biases distributed according to

$$\hat{W}_{kj} \propto \mathcal{N}(0.0, 0.5), \tag{4.27}$$

$$W_{kj} = \frac{\hat{W}_{kj} + \hat{W}_{jk}}{2}, \tag{4.28}$$

$$B_k \propto \mathcal{N}(0.0, 0.5), \tag{4.29}$$

$$\tag{4.30}$$

where $\mathcal{N}(\mu, \sigma)$ is the normal distribution with mean μ and standard deviation σ. The third and fourth neuron's bias is set to ± 1 to ensure one leak-over-threshold and one leak-below-threshold neuron for Figure 4.2e.

4.3.6 *NORB*

The layer sizes of our hierarchical NORB networks is given in Table 4.2. In order to reduce the pixelation in Figure 4.5a we do not plot the visible state $\mathbf{v} \in \{0,1\}^{3600}$ directly but instead show the activation probability $p(\mathbf{v})$ that is imprinted by the instantaneous state of the hidden layer:

$$p_{T=1}(\mathbf{v}|\mathbf{h}) = \frac{1}{1 + \exp\left(-\mathbf{W_{vh}}\mathbf{h} - \mathbf{B_v}\right)} \, . \tag{4.31}$$

The temperature schedule of the oscillating background case can be found in Table 4.2. For the static background input we use the reference configuration (Equation 4.14) and retrieve samples every $1/f_{\mathrm{osc}} = 1\,\mathrm{s}$ in order to get an equal-time comparison.

4.3.7 *MNIST*

The layer sizes of our hierarchical MNIST network is given in Table 4.2. In Figure 4.8 we use a similar network structure to the one in Figure 4.5, with parameters from Leng et al. [260]. Background configurations are varied according to Equation 4.9 as before and sine parameters are given in Table 4.2.

4.3.8 *Kullback-Leibler divergence*

The Kullback-Leibler divergence is a standard measure of the discrepancy between two probability distributions. Intuitively, it measures how many bits are wasted when encoding a distribution Q according to the optimal encoding for distribution P. For a discrete probability distribution P with respect to another Q, this divergence is defined as:

$$D_{\mathrm{KL}}\left(P||Q\right) = \sum_i P(i) \log\left(\frac{P(i)}{Q(i)}\right) . \tag{4.32}$$

Note that Q must be strictly positive, whereas P may have states with zero probabilities associated with it.

4.3.9 *Indirect sampling likelihood*

We quantitatively evaluate how well the samples generated by our networks reflect the target distribution by calculating the indirect sampling likelihood

(ISL) described in Breuleux et al. [261]. The ISL measures the similarity between the generated samples and samples from the dataset that were not shown during training (test set). Each test sample $\mathbf{y}_j$ and generated sample $\mathbf{x}_i$ is a d-dimensional binary vector, whereby each $\mathbf{x}_i$ is given by the instantaneous visible layer activity $\mathbf{v} \in \{0, 1\}^d$.

For retrieving the ISL, a density model $\mathcal{P}$ is trained on N generated samples, and the likelihood of each test sample under $\mathcal{P}$ is calculated. For d-dimensional binary vectors, a non-parametric kernel density estimator is suitable:

$$\mathcal{P}(\mathbf{y}) = \frac{1}{N} \sum_{i=1}^{N} \prod_{j=1}^{d} \gamma^{1_{\mathbf{y}_j = \mathbf{x}_{ij}}} (1-\gamma)^{1_{\mathbf{y}_j \neq \mathbf{x}_{ij}}}, \tag{4.33}$$

which is essentially a mixture model representing the $\mathbf{x}_i$. The hyperparameter $\gamma \in [0.5, 1[$ determines how much the empirical distribution over $\mathbf{x}_i$ is smoothed out (we use $\gamma = 0.95$).

The two exponents denote identity functions that compare an individual test to a generated sample and count the identical and different pixels, respectively. Intuitively, the ISL penalizes each test sample far from any generated sample.

In Figure 4.8f, we plot $\log \mathcal{P}(\mathbf{y})$ averaged over all test samples versus the number of samples. This time course reveals how many main modes of the target distribution are well covered and how fast. Note that the ISL does not necessarily evaluate how diverse the network output is, but rather how well the test set is covered — repetitive samples would yield a high ISL compared to a not very diverse test set.

For orientation, we show the ISL curves for the optimal sampler (OPT) and the product of marginals (POM) (see Figure 4.8f). The optimal sampler draws randomly, without replacement, from a pool of 10^5 images that were generated with Adaptive Simulated Tempering (AST) [259], a complex algorithm that is constructed for optimal mixing properties. The POM sampler generates examples by independently sampling each vector component from its respective intensity distribution over the training set. Hence, the marginal probability distribution for each component is preserved, and correlations between components, i.e., the overall structure, are discarded. Note that since these off-class samples overlap significantly with all image classes, they can be associated with higher ISL values than a series of samples from a single mode.

One known drawback of the ISL is that it does not represent an accurate reflection of a human's perceptual judgment of image quality [262]. Therefore, we additionally checked the sampling quality by eye and evaluated the activation probability of the visible layer as shown in Figure 4.8a and Figure 4.8b. Based on this, we picked a point on the $f_{\text{osc}} = 0.5\,\text{Hz}$ plane with an intermediate ISL value for display in Figure 4.8a, Figure 4.8b, Figure 4.8e and Figure 4.8f.

4.3.10 *Mode duration as a measure of mixing speed*

We calculate the mode duration as the average time between two mode switches, where the current mode is defined as the most active label unit, as measured by its probability inferred from the hidden layer activity. The label layer reflects the network's interpretation of its own current visible state **v** and as such requires the network to be self-consistent. In practice, we did not find significant deviations (see Figure 4.8a and Figure 4.8b) from this assumption. Due to computational constraints, we only simulated 1000 s in a single run and averaged over multiple simulations for improved statistics. Note that conventionally, mixing speed is measured by the area under the autocorrelograms of the network neurons' activity, where a smaller area corresponds to faster mixing. For comparison, we also recorded this measure from the inferred spike probabilities of the label neurons, which confirmed the speed-up in mixing with oscillations (see Figure 4.10). However, when classes are discrete, like in the MNIST data set, mode durations are a sufficient and intuitive measure of mixing speed.

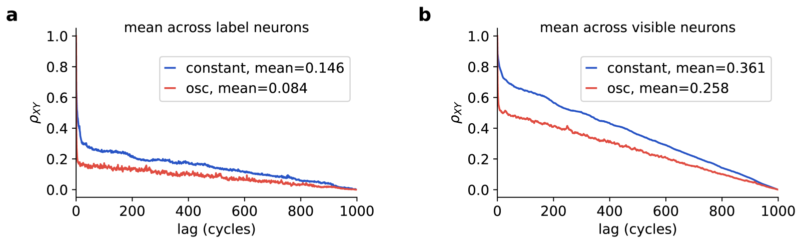


Figure 4.10: **Layerwise autocorrelograms indicate improved mixing. (a)** Mean Pearson autocorrelation coefficient calculated from the inferred spike probability of the label layer neurons of the MNIST network in Figure 4.8 — for oscillating background (red) and constant background at $T = 1$ (blue). **(b)** Same as **(a)**, for the visible neurons. Autocorrelation is reduced more quickly for the oscillating setup, leading to a smaller area under the curve, indicating faster mixing.

4.3.11 *Simulation details*

The simulations of sampling experiments with current-based neurons were performed with sbs [263] version 1.8.2 with slight modifications. This framework was executed with PyNN [15] version 0.9.1 and NEST [264] version 2.14.0 with a time resolution of $\Delta t = 0.1\,\text{ms}$.

4.4 DISCUSSION

Oscillatory activity is a naturally emerging phenomenon in spiking neuronal networks. As it is well-known that background input increases the variability of neuronal firing, oscillatory background implies oscillatory variability. In the context of ensemble theory, this creates a direct link to the notion of temperature. We have shown that the level of background input determines the sampling temperature in networks of LIF neurons and demonstrated that this effect leads to functional advantages in sampling networks when oscillatory background input is present. This finding holds in the case of current-based synaptic interactions, for which we have presented an analytical treatment of cortical oscillations as tempering. We have furthermore shown that oscillations improve sampling from the distribution represented by the network (i.e., a prior distribution, see Figure 4.5 and Figure 4.8) as well as for dealing with uncertainty evoked by input (i.e., posterior distributions, see Figure 4.7.) Our results suggest that the ubiquity of oscillations in human and animal brains provides a clear benefit for behaviorally relevant computations, which is elucidated by considering the analogy to simulated tempering.

RELATED THEORETICAL WORK Our considerations rest on the assumption that for fixed parameters, spiking networks sample from a stationary distribution. This assumption has been shown to hold under only mild constraints for a large class of neuron and network models in Habenschuss et al. [265]. They also showed that in the presence of periodic input, a phase-specific stationary distribution exists, influenced by the network parameters and the properties of the inputs. The existence of such a distribution naturally leads to questions about its specific nature, given specific ensemble dynamics such as those arising in networks of connected LIF neurons and its functional properties for cortical computation. In this work, we have shown that the phase-dependent component is a temperature scaling of a Boltzmann distribution, with periodic background alternating between its exploration and exploitation.

An alternative way of promoting mixing was proposed by Leng et al. [260]. There, short-term synaptic plasticity was shown to weaken local attractors. This mechanism has a similar effect but is different from a change in temperature. Since this form of plasticity only affects active synapses, it only suppresses active local modes rather than flattening the entire distribution. These dynamics ensure that local modes can be abandoned quickly, as synapses can be weakened significantly by only a few spikes, but they come at the cost of changing the sampled distribution. In contrast, cortical oscillations induce a well-defined temporal structure that promotes an undistorted readout. For mathematical tractability, we considered LIF neurons with current-based synaptic interactions and network structures that are easily amenable to contrastive Hebbian training. This suggests that the computational role we propose for cortical oscillations is generalizable to a diverse set of cortical structures and their associated functions. Indeed, it has already been observed that oscillations appear to have a similar function throughout the cortex [266].

A similar function of brain rhythms related to slower oscillations was proposed by Sohal et al. [246], who suggested on theoretical grounds that during the hippocampal theta cycle, modulation of GABA_B synapses performs a process similar to simulated annealing in a model of population dynamics. Such a mechanism was shown to be advantageous for sequence disambiguation [245]. In this work, we propose that temperature control takes place on the level of individual neurons via input regardless of the synapse type. Thus, the mechanism we propose for incorporating such annealing in neural networks has a much more general scope. Savin et al. [248] showed the benefits of rhythmic changes of neuron excitability in a model of probabilistic memory recall, resulting in a similar kind of annealing as in our model. Our work shows how such a schedule of excitability changes arises in spiking neural networks via background input, thus suggesting an implementation of this mechanism on the cellular level.

One key aspect of previous models is their reliance on excitability modulation of a limited subset of inputs (e.g., recurrent vs. feedforward inputs [248]). While distinct modulations might arise in biological neurons when inputs target different neuronal compartments, our model shows that this constraint is not necessary to leverage the computational benefits of oscillations as global changes of neuronal input-output behavior suffice. Our model also does not rely on a specific synapse or receptor type, and the proposed mechanism can play out across different oscillatory frequency bands, thus giving our results a very general scope.

The results in this work suggest that oscillations of the background input promote mixing. Previous theoretical work has shown that other sampling methods such as Langevin [267] and Hamiltonian Monte Carlo [249] sampling can also serve this purpose. These studies use rate-based models to sample from continuous-valued probability distributions such as multivariate Gaussians. Our models differ from this approach in two important ways. First, the sampling models based on firing rates [267, 249] require specifically tuned network weights to accomplish rapid sampling. We have shown that oscillating background input can speed up mixing without requiring specifically tuned weights, thus providing our proposed mechanism with a broader scope. Second, our model is based on more complex network state distributions, defined over binary-valued random vectors instead of continuous values. Importantly, these values relate directly to spiking activity. Langevin sampling and Hamiltonian Monte Carlo are not directly applicable to this case. However, it could still be the case that these mechanisms complement each other in the cortex, potentially acting on different timescales.

RELATED EXPERIMENTAL WORK AND MODEL PREDICTIONS Across the entire spectrum of cortical rhythms, individual components of these oscillations are characterized by their frequency and amplitude. We have shown that an effective tempering schedule can be achieved for sinusoidal waves across a wide range of frequencies and amplitudes, roughly corresponding to the range lying between slow and alpha waves [238]. For higher modulatory frequencies, the sampling quality quickly deteriorates as the internal network dynamics cannot react quickly enough to the changes in temperature. However, the soft upper-frequency limit is not fixed and depends on model parameters and the network distribution. In particular, the speed at which the network can change its state depends on the ratio of the dominant time constants of individual neurons and synapses (here on the order of 10 ms) to the duration of a cycle. For faster dynamics, as often observed in vivo (e.g., membrane time constants [268]; synaptic time constants [269, 270]; refractory periods [271, 272]), correspondingly faster oscillations can be accommodated. For example, oscillations in the gamma band could be employed by ensembles with synaptic time constants and refractory times in the order of a few milliseconds, as discussed in recent sampling-based modeling approaches [249, 273]. Thus, this form of tempering can be exploited both for inference in the awake state, where oscillations are typically fast, and during sleep, for functions such as memory retrieval and consolidation [274, 242–244].

Concerning experimental neuroscience, the suggested computational mechanisms relate to various physiological and psychophysical phenomena, ranging from single-neuron activity to behavior. The tempering in our model modulates the gain of the neuronal transfer functions, similar to the stochastic sampling of a scene through an oscillatory modulation of attentional gain [242, 275], particularly through top-down input [276, 277]. The stochasticity in our model by which stored memories are selectively recalled is mirrored in the randomness of hippocampal replay during sleep that goes beyond the more typical behavior [278], or in free memory recall in humans [279]. The oscillatory recall that supports cognitive computation in our model can also be related to creative thinking [280], to midbrain oscillatory activity during stimulus disambiguation [281], to mind wandering [282] and to local sleep [283].

The oscillation frequency, and thus the rate of temperature change, carries another subtle effect. For slow waves, the effect of a single transition from maximum to minimum temperature is similar to simulated annealing [284]. As the network effectively has more time to relax towards its corresponding thermodynamic equilibrium, it will, at least statistically, tend towards the global minimum energy state. On the other hand, faster oscillations are more akin to tempered transitions [285, 286]. Indeed, the extreme scenario of quenching (extremely rapid cooling) could be implemented by switches between synchronized cortical up and down states [236, 254, 253]. Thus, different oscillatory phenomena in the brain can shift the focus from finding a small set of maximum probability modes to finding a larger range of relevant modes. Similarly, the oscillation amplitude can also control the effective breadth of the exploration space, with larger maximum rates promoting larger jumps between more dissimilar network states.

The benefits of cortical oscillations also extend to other facets of Bayesian inference. For example, when the state distribution is constrained by partial observations, tempering helps explore the conditional distribution and find multiple ways to solve this pattern completion problem. Similarly, this can help find multiple solutions to a given problem, such as assigning multiple categories to particular input patterns. Importantly, this also highlights the potential benefits of background oscillations during learning (see also Capone et al. [287]), where exploration plays an essential role.

Our results demonstrate that oscillations provide an additional benefit to improved mixing: they serve as a reference for reading out computational results, reducing the amount of data requiring processing, and facilitating the temporal organization of neural computations. Furthermore, they can also serve as a means of input filtering, increasing susceptibility to coherent stimuli [288]. In general, it is well known that information encoding via a background oscillation can be found in the brain, for example, in the hippocampus [289], where place cells convey information by firing earlier or later relative to the theta rhythm. A similar form of coding takes place in our models, as the network distribution changes during each cycle of the background input.

Furthermore, cyclic background input results in the network generating a stream of candidate solutions, with one such state arriving in each cycle. This leads to a form of computing in discrete steps, as computations are structured into episodes defined by background oscillations. A similar type of structured computation has been suggested to take place in monkey and human visual brains during the processing of visual inputs [290]. These experiments showed that shifts in attention were aligned to beta-band oscillations, and every shift took place within a single cycle. In our model, we find similar shifts of the state taking place within each cycle as the temperature decreases. Temperature changes from oscillations predict that the time course of the network state variability is coupled to the oscillation phase, as we have shown in our model. This suggests that a similar coupling could be found in sampling-based computations in the brain. Jezek et al. [291] have given a hint that this can indeed be the case in hippocampal circuits by showing that ambiguous interpretations of the network input are more likely in the first half of the theta cycle. However, this data is rather coarse, thus, more detailed experimental data are required to constrain sampling models based on background oscillations adequately.

In particular, experimental data could elucidate whether cortical networks are tuned to an unbiased sampling regime. In our model, achieving unbiased sampling requires tuning of neuron parameters and the background oscillation time course. A model more closely matching biological networks would help to either corroborate this finding or provide more insight into how such a tuning might be achieved in brain networks. The closer matching could be achieved, for example, by incorporating additional features such as short-term plasticity, neuronal adaptation, and more specific inhibition (see [292] for an example). In general, the balance between excitation and inhibition is of renewed interest in this context, as it connects directly to experimental data. Individual neurons or neuron populations can, for example, use unbalanced rates to implement biases for their associated random variables. Moreover, we expect that different networks tune their background inputs to different balances, depending on which biases are beneficial for their respective tasks.

The experiments of Jezek et al. [291] provide evidence for a sampling-based representation of spatial beliefs in the hippocampus, with one sample drawn in each theta cycle. Another important account of place cell activity states that the activity within different parts of each theta cycle corresponds to different places of the animal within its movement trajectory (e.g., O'Keefe et al. [289] and Pfeiffer et al. [293]). This view is consistent with a sampling strategy that samples trajectories (temporal sequences) instead of static values, where one trajectory sample is drawn per cycle. While the analysis of trajectory sampling in spiking neural networks is beyond the scope of this work, we note that the general sampling framework can be extended to temporal sequences [265] in which a phase-dependent probability distribution arises from external, phase-dependent input. A model of such a form of sequence sampling could be used for both modeling the phase-dependent activity of neurons encoding previously visited locations [289] as well as for sampling possible future trajectories [293]. For the latter case of sampling diverse sequences within one theta cycle, faster background oscillations (e.g., alpha-band), superimposed on the theta activity, could provide rapid sequential annealing to the individual sequence elements.

In general, our model relates to simple experimental observations at multiple levels. For example, with respect to the activity of single neurons or small populations, the strength and frequency of cortical oscillations should directly influence the decorrelation of neuronal activity (see also Figure 4.10). At a more behavioral level, oscillatory changes in background activity would influence the frequency of perceptual switches. For example, for multi-stable or incomplete images (such as those in Figure 4.7), perceptual switches should happen in phases of high activity (i.e., during cortical up-states), as measured, for example, by EEG data. We would thus predict a monotonic relationship between the frequency of switches between up and down states and the frequency of perceptual switches.

In this work, we have used sinusoidal modulations of the background rates. This represents a natural choice, as any other periodic waveform can be described via Fourier synthesis over such elementary waveforms. Particular time courses of the background input would influence and possibly even benefit computations in the network, depending on the circumstances and nature of the task that needs to be solved. For example, prolonging the low-temperature phase could allow valid samples to be read out over a longer period of time. In contrast, more frequent high-temperature phases would prevent the network from clinging to a possibly wrong belief. The background rates could even take on only two distinct values and alternate between high background activity (resulting in a high temperature, allowing the network to traverse the state space) and low background activity (where the network converges onto a single mode). This provides a link to experiments that study the computational role of cortical on/off states. For example, Engel et al. [294] report that monkeys are more likely to correctly recognize subtle visual cues if they happen during on-states. This aligns with our proposed computational role of cortical background activity, as networks need a stronger background to be able to change their current belief and react to small changes in their input. Note also that these different phases need not be strictly cyclic but might underlie external control, allowing external circuitry to flexibly guide computations in cortical networks according to momentary cognitive demands.

APPLICATIONS Recent years have seen an increasing interest in using spike-based computation on specialized hardware to perform energy-efficient computations [1]. This has spurred efforts to develop models which allow efficient learning and inference with spiking neural networks. Some of these platforms explicitly exploit the stochasticity of their components for computation [295, 296]. By offering a mechanism for modulating neuronal stochasticity, the oscillatory background can enhance computation in stochastic neuromorphic networks, for example, in generative spiking models [233, 297].

Periods of faithful matching between the sampled and target distribution mark the implicit time windows in which computational results can be read out and manifest as constrained intervals of the entire cycle (also see Figure 4.3 and Figure 4.6). However, it is important to note that the length of these time windows depends on the underlying distribution, the time course of the background modulation, and the time constants in the network. The time window suggests that such oscillations may also improve the performance of networks used for constraint satisfaction problems [298–300]. These are solved by shaping the stationary distribution of the network so that solution states have a high probability. However, it is not clear at any given point in time whether the current state is a solution candidate or a transitional state. In contrast, in an oscillation-driven tempering schedule, it is known that solutions are likely at low-temperature phases.

Overall, the parallels with a variety of empirical phenomena and the advantages for spike-based sampling demonstrated here make neuronal oscillations not only a likely mechanism for supporting stochastic computations in the brain but also a useful tool for fulfilling this same function in biologically inspired neural networks.

5 CONCLUSIONS

"If the figures obtained from the satellite were simply the product of my deranged mind, they could not possibly coincide with the second series. My brain might be unhinged, but it could not conceivably compete with the Station's giant computer and secretly perform calculations requiring several months' work. Therefore if the figures corresponded, it would follow that the Station's computer really existed, that I had really used it, and that I was not delirious."

Stanislaw Lem in *Solaris*, 1961

This thesis explores spike-based neural networks, with a particular focus on network effects and emergent functions. Situated at the intersection of neuroscience and machine learning, this work has the potential to drive innovation in both fields [1, 301]. The research was guided by three overarching questions, which were addressed through the development of novel models and computational approaches. By approaching these themes indirectly through specific research problems, this thesis enhances the understanding of complex issues in this interdisciplinary field from multiple perspectives.

Chapter 3: biologically inspired enhancements to a synaptic plasticity model

How can machine learning principles be expressed through biological mechanisms and incorporated into biologically plausible models?

We integrated eligibility propagation (e-prop) [154] into NEST [8], a spiking neural network simulator specialized for large-scale network simulations. E-prop, a biologically plausible synaptic plasticity model for spiking recurrent neural networks, approximates the performance of backpropagation through time (BPTT) [176]. It already provides biologically plausible solutions for machine learning features such as gradient-based optimization, regularization, weight optimization, credit assignment, and loss functions. This learning rule belongs to the class of three-factor models, supported by substantial experimental evidence [174], which combine classical Hebbian learning (pre- and post-synaptic terms) with an additional modulatory signal [for a review, see 175, 76]. E-prop further employs surrogate gradient methods to enable gradient-based training of spiking neural networks, approximating gradients despite the non-differentiable nature of spike-induced membrane voltage resets [for a review, see 191].

Porting the algorithm from the machine learning library TensorFlow [177] to NEST involved translating its synchronous, time-driven weight update scheme into an asynchronous, event-driven scheme better suited to NEST's architecture. This transition accommodates the sparse, spike-driven nature of biologically plausible networks and promises improvements in computational efficiency. Using two proof-of-concept tasks from [154], we showed that our event-driven implementation precisely reproduces the results of the original time-driven model. Adapting the plasticity rule to the simulator's built-in biologically inspired constraints revealed certain shortcomings in the original model's biological realism. For instance, the original model assumes instantaneous signal transmission on some connections, neglecting delays. To address this, we introduced connection delays and restored the original model's behavior by shifting the histories used for weight updates. This modification facilitates future experiments without the compensatory shift and with varying connection delays. Moreover, the original model employs a batch learning paradigm from classical machine learning, processing multiple samples in parallel and averaging the resulting weight updates. To replicate the original model's behavior while enhancing biological plausibility, we replaced this approach with the sequential processing of batch samples, summing updates to compute the final average.

We extended the synaptic plasticity model by replacing machine learning-inspired features with biologically plausible mechanisms and introducing new realistic components. By removing the reset of neuron dynamics and the eligibility trace filter, we enabled continuous dynamics. The addition of a learning signal generator and dynamic firing rate regularization allowed for weight updates with every spike, replacing fixed updates after each sample and eliminating the need for a biologically unrealistic central clock mechanism.

The biological mechanism of connection delays revealed that certain loss functions, such as cross-entropy loss, necessitate additional communication between network components. This insight prompted us to explore alternatives, ultimately leading to the adoption of mean squared error for classification tasks. The simulator's biologically inspired constraint, which mandates that all information must be locally available unless transmitted through continuous signals or spikes, exposed a biologically unrealistic aspect of the original model: synapses depend on the output neuron's membrane voltage time constant to compute the eligibility trace filter. To address this, we decoupled the eligibility trace filter from this time constant. Additionally, we introduced smoother and more fine-tuned surrogate gradients, which are biologically more realistic compared to the original piecewise linear surrogate gradient.

Finally, we combined all the above features and adopted a neuron model with a full reset of the membrane voltage. We conducted experiments to constrain the weights — separately for input, recurrent, and output layers as well as for all weights collectively. These constraints included fixed weight signs and Dale's law together with a biologically realistic excitatory-to-inhibitory ratio.

How does function emerge from biological and artificial mechanisms?

In our biologically enhanced model, the function of "learning" emerges from the interplay of multiple biologically inspired mechanisms. We evaluated the contribution of each biological feature to learning performance using the widely recognized neuromorphic MNIST task [183], a benchmark adapted from classical machine learning for spiking neural networks. Learning performance was assessed by averaging the error across iterations during the test phase following training.

Several features contributed to improved learning performance, including mean squared error classification, scaled linear surrogate gradients, scaled exponential surrogate gradients, and the combination of all features with full membrane voltage reset. Additionally, fixed weight signs and Dale's law combined with a biologically realistic excitatory-to-inhibitory ratio applied to the recurrent weights, further enhanced learning performance. Continuous neuron dynamics did not influence learning performance, while the other features led to slight reductions. Nevertheless, the overall test error consistently remained below 0.12, demonstrating effective learning.

Continuous dynamics eliminate the need for additional operations to reset neuron and synapse dynamics, simplifying the model. Dynamic firing rate regularization enables the network to respond more flexibly to changes in spiking activity and adapt sensitively to recent data. Introducing a dedicated signal allows for the flexible opening and closing of learning windows. Together, these three features — along with inter-spike integration — enable the processing of variable-sized samples, moving the model closer to real-world data processing capabilities.

How can principles underlying biological mechanisms be leveraged to advance machine learning algorithms?

Transforming biological phenomena into innovations for machine learning requires several key steps. As discussed in Chapter 2, biological phenomena can be described using mathematical models. While mathematical derivations provide exact solutions, they often rely on significant simplifications, limiting their capacity to capture the complex interactions observed in biological systems. Simulations bridge this gap, enabling the empirical exploration of such complexities. Computational models and simulations, as emphasized in Chapter 2, are particularly valuable for studying neural processes that analytical methods alone cannot fully address. To effectively answer research questions, modelers must carefully balance biological realism with the computational simplicity necessary for large-scale simulations.

The simulations presented in Chapter 3 allowed us to identify several biological features that enhance learning performance, making them strong candidates for improving machine learning algorithms. In machine learning systems, where performance is critical, computational efficiency is as important as accuracy. To evaluate the computational efficiency of each feature, we measured the runtime of each simulation across the training and test iterations. Our results showed that biologically realistic sparsity in network connections significantly improved computational efficiency. While most features — such as continuous neuron dynamics, mean squared error classification, and decoupling the eligibility trace filter from the output — did not increase runtime, the other features led to runtime increases of up to 16 %. These minor increases could partly be attributed to altered network activity, which may prolong simulation times. Nevertheless, our biologically realistic plasticity model remained computationally feasible.

The algorithmic formulation of the biological mechanisms presented here enables seamless integration of these underlying principles into machine learning algorithms. This allows the emergent functions of the biological features identified to provide tangible benefits to machine learning systems.

Chapter 4: spike-based sampling facilitated by oscillatory background activity

How can machine learning principles be expressed through biological mechanisms and incorporated into biologically plausible models?

We consider the interpretation of spiking activity in the cortex as probabilistic inference via sampling, a perspective that has gained substantial experimental [226–228] and theoretical [229–233] support over the last decade. Mathematically, these models are closely related to Gibbs sampling, which often struggles to transition between states of high probability. This challenge, referred to as the “mixing problem”, arises particularly when networks are trained on high-dimensional data, where strong attractor states dominate the representation.

An algorithm known as “simulated tempering” introduced by Marinari et al. [256] in the context of Markov-chain Monte Carlo sampling for Ising models, addresses the mixing problem. Ising models are closely related to spike-based sampling, both in their dynamics and in the distributions they sample [232]. We demonstrate how the rate of background spike activity functions as an effective temperature, with oscillations in this rate being interpretable as a form of simulated tempering. Firing rate oscillations across multiple frequency bands and amplitudes are a naturally emerging phenomenon in spiking networks [234–237] and have been extensively studied in the mammalian brain [238, 239]. Notably, these oscillations appear to play a critical role both during awake perception [240–242] and during sleep [243, 244], highlighting their fundamental importance in cognition and learning.

We establish a rigorous mathematical link between spike-based probabilistic inference and cortical oscillations, two phenomena that have not been explicitly connected in the context of spiking neural networks. Through simulations, we provide a proof of concept showing that oscillations robustly realize the same functionality as tempering across different frequency bands. We demonstrate this in various networks and tasks, including a Boltzmann network with a few neurons, the NORB dataset [257], and the MNIST dataset [258]. To evaluate the quality of the sampled distributions, we use machine learning metrics such as Kullback-Leibler divergence and indirect sampling likelihood [261], and measure the duration the network remains in a high-probability state, referred to as a "mode".

How does function emerge from biological and artificial mechanisms?

Our results demonstrate that cortical oscillations can fulfill a functional role similar to simulated tempering: enabling mixing by helping the circuit escape strong attractors and transition efficiently between different high-probability network states. Unlike short-term plasticity [260], oscillations maintain an undistorted distribution. Brain waves rhythmically modulate exploration and exploitation in the state space of learned representations by periodically flattening the energy landscape through an increased effective temperature. High-frequency oscillations facilitate fine-grained sampling of the representation, while larger, high-amplitude oscillations enable coarser sampling. Thus, these oscillations accelerate sampling, enhance the network's generative properties, and support efficient stochastic computations.

How can principles underlying biological mechanisms be leveraged to advance machine learning algorithms?

The presented analogies between oscillations and simulated tempering, combined with the extensive literature on the role of oscillations in various cognitive functions, offer valuable inspiration for innovative machine learning algorithms. For instance, oscillations can improve the handling of input uncertainty in the context of sequence disambiguation [245] by enabling rapid switching between interpretations of ambiguous data, which is essential for efficient learning and sampling-based probabilistic inference. Experimental evidence strongly links oscillations to memory formation and retrieval [274, 242–244] and attention [242, 275–277]. Furthermore, oscillations can serve as temporal references for organizing inputs and structuring them into discrete episodes [290]. Our mathematical and algorithmic formulation of brain oscillations allows for their straightforward integration into machine learning algorithms.

REPRODUCIBILITY

Chapter 2 highlights the growing importance of reproducibility in computational neuroscience, emphasizing the value of open-access data, models, and code. Adhering to open standards, contributing actively to community resources, and employing practices such as containerized distribution and version-controlled software enhance transparency and facilitate collaboration. These collective efforts are fostering the development of a robust ecosystem of simulation tools, enabling the creation of integrative computational models in neuroscience.

In the research projects presented in this thesis, we adhered to these standards. In the project on the event-driven implementation of e-prop (Chapter 3), we first reproduced the original results exactly using a different computational scheme before extending the model further. Both implementations are provided as open-source code within the NEST simulator, accompanied by comprehensive tutorials on their usage. The project on cortical oscillations (Chapter 4) has been published as an open-access codebase on Zenodo.

LIMITATIONS

Although the algorithms presented in this work are broadly applicable, they were demonstrated on a limited set of benchmark tasks, serving primarily as proofs of concept. Each benchmark comes with its own inherent biases, which may influence the generalizability of the results. Furthermore, the algorithms were implemented with only a limited number of neuron and network models, which do not fully capture the diversity of biological systems or potential use cases.

Our findings suggest that the dynamics — and consequently the learning performance — are sensitive to variations in neuron and network parameters, as well as the hyperparameters of the algorithms. Testing the robustness of these algorithms against parameter changes and fine-tuning the parameters for optimal performance remains an intriguing direction for future research.

Additionally, the rapid advancements in the field of machine learning have introduced more sophisticated metrics for evaluating performance, which could further refine the evaluation of these algorithms.

OUTLOOK

Advancements in AI demonstrate that large-scale networks exhibit qualitatively different performance compared to small ones, underscoring the need to study biologically realistic models at scale. Such studies are essential for understanding system-level learning. Future research should therefore explore the scalability of these models under both strong and weak scaling scenarios and identify tasks that challenge large-scale networks. A key open challenge is finding a scalable, real-world task of neuroscientific relevance or one inspired by behavioral studies. Video frame prediction, for example, is a promising candidate that could leverage the models' strengths in temporal processing.

The abstract models developed in this work could be refined to incorporate brain-area-specific architectures, enabling a deeper evaluation of how structural constraints influence functional outcomes. Constraining parameters and network architectures with experimental data would enhance both the biological relevance and predictive accuracy of these models. Additionally, the oscillatory dynamics could be improved to align more closely with biological plausibility, guided by experimental data such as EEG recordings, which provide accessible measurements of brain wave activity for constructing realistic oscillatory patterns.

Our implementations serve as benchmarks for other coding platforms and provide a foundation for deployment on neuromorphic hardware, particularly if their computational efficiency surpasses that of conventional systems. These models can assist neuromorphic hardware development by serving as a reference for verification, ensuring systems are built correctly, and validation, confirming their performance matches or exceeds traditional computing standards. Although designed with hardware-agnostic principles, adapting and optimizing the algorithms for neuromorphic hardware remains a critical area for future exploration.

Finally, measuring energy consumption on conventional computing architectures is an important direction for assessing the models' efficiency and sustainability, as well as providing a basis for comparison with alternative platforms.

CONCLUDING STATEMENT

This dissertation demonstrates the implementation of biologically inspired algorithms in a widely used neural simulation platform, enhancing the learning and sampling functions of spiking neural networks for diverse applications. Balancing biological plausibility and machine learning performance while leveraging sparsity and locality, the algorithms are primed for future implementation on neuromorphic hardware. The findings highlight the computational benefits of biologically grounded constraints, enabling scalable and efficient, brain-inspired AI systems. This work bridges machine learning and neuroscience, deepening our understanding of processes underlying cognitive functions.

BIBLIOGRAPHY

[1] K. Roy, A. Jaiswal, and P. Panda. “Towards spike-based machine intelligence with neuromorphic computing.” In: *Nature* 575.7784 (2019), pp. 607–617. DOI: 10.1038/s41586-019-1677-2.

[2] J. H. R. Lübke and A. Rollenhagen. *New Aspects in Analyzing the Synaptic Organization of the Brain*. Ed. by J. H. R. Lübke and A. Rollenhagen. New York, NY: Springer US, 2024, pp. 277–321. ISBN: 9781071640197. DOI: 10.1007/978-1-0716-4019-7.

[3] G. T. Einevoll, A. Destexhe, M. Diesmann, S. Grün, V. Jirsa, M. de Kamps, M. Migliore, T. V. Ness, H. E. Plesser, and F. Schürmann. “The Scientific Case for Brain Simulations.” In: *Neuron* 102.4 (2019), pp. 735–744. DOI: 10.1016/j.neuron.2019.03.027.

[4] L. Alonso-Nanclares, J. Gonzalez-Soriano, J. Rodriguez, and J. DeFelipe. “Gender differences in human cortical synaptic density.” In: *Proceedings of the National Academy of Sciences* 105.38 (2008), pp. 14615–14619. DOI: 10.1073/pnas.0803652105.

[5] D. J. Linden. “Our Human Brain was Not Designed All at Once by a Genius Inventor on a Blank Sheet of Paper.” In: *Think Tank: Forty Neuroscientists Explore the Biological Roots of Human Experience*. Yale University Press, 2018, pp. 1–8. DOI: 10.12987/9780300235470-002.

[6] F. A. Azevedo, L. R. Carvalho, L. T. Grinberg, J. M. Farfel, R. E. Ferretti, R. E. Leite, W. J. Filho, R. Lent, and S. Herculano-Houzel. “Equal numbers of neuronal and nonneuronal cells make the human brain an isometrically scaled-up primate brain.” In: *Journal of Comparative Neurology* 513.5 (2009), pp. 532–541. DOI: 10.1002/cne.21974.

[7] G. Trensch, R. Gutzen, I. Blundell, M. Denker, and A. Morrison. “Rigorous Neural Network Simulations: A Model Substantiation Methodology for Increasing the Correctness of Simulation Results in the Absence of Experimental Validation Data.” In: *Frontiers in Neuroinformatics* 12 (2018). DOI: 10.3389/fninf.2018.00081.

[8] M.-O. Gewaltig and M. Diesmann. “NEST (NEural Simulation Tool).” In: *Scholarpedia* 2.4 (2007), p. 1430. DOI: http://dx.doi.org/10.4249/scholarpedia.1430.

[9] D. Plotnikov, B. Rumpe, I. Blundell, T. Ippen, J. M. Eppler, and A. Morrison. “NESTML: a modeling language for spiking neurons.” In: *arXiv* (2016). DOI: 10.48550/arXiv.1606.02882. eprint: 1606.02882v1.

[10] M. L. Hines and N. T. Carnevale. "NEURON: a tool for neuroscientists." In: *The Neuroscientist* 7.2 (2001), pp. 123–135. DOI: 10.1177/107385840100700207.

[11] M. Stimberg, R. Brette, and D. F. Goodman. "Brian 2, an intuitive and efficient neural simulator." In: *eLife* 8 (2019), e47314. DOI: 10.7554/eLife.47314.

[12] T. Bekolay, J. Bergstra, E. Hunsberger, T. DeWolf, T. C. Stewart, D. Rasmussen, X. Choo, A. Voelker, and C. Eliasmith. "Nengo: a Python tool for building large-scale functional brain models." In: *Frontiers in Neuroinformatics* 7 (2014), p. 48. DOI: 10.3389/fninf.2013.00048.

[13] N. Abi Akar, B. Cumming, V. Karakasis, A. Küsters, W. Klijn, A. Peyser, and S. Yates. "Arbor — A Morphologically-Detailed Neural Network Simulation Library for Contemporary High-Performance Computing Architectures." In: *27th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*. 2019, pp. 274–282. DOI: 10.1109/EMPDP.2019.8671560.

[14] J. Vitay, H. ü. Dinkelbach, and F. H. Hamker. "ANNarchy: a code generation approach to neural simulations on parallel hardware." In: *Frontiers in Neuroinformatics* 9 (2015), p. 19. DOI: 10.3389/fninf.2015.00019.

[15] A. P. Davison, D. Brüderle, J. M. Eppler, J. Kremkow, E. Muller, D. Pecevski, L. Perrinet, and P. Yger. "PyNN: a common interface for neuronal network simulators." In: *Frontiers in Neuroinformatics* 2 (2009), p. 11. DOI: 10.3389/neuro.11.011.2008.

[16] P. Gleeson et al. "NeuroML: A Language for Describing Data Driven Models of Neurons and Networks with a High Degree of Biological Detail." In: *PLOS Computational Biology* 6 (6 2010). DOI: 10.1371/journal.pcbi.1000815.

[17] S. J. van Albada, A. Morales-Gregorio, T. Dickscheid, A. Goulas, R. Bakker, S. Bludau, G. Palm, C.-C. Hilgetag, and M. Diesmann. "Bringing anatomical information into neuronal network models." In: *Computational Modelling of the Brain*. Ed. by M. Giugliano, M. Negrello, and D. Linaro. Vol. 1359. Advances in Experimental Medicine and Biology. Springer, 2021, pp. 201–234. DOI: 10.1007/978-3-030-89439-9_9.

[18] N. W. Gouwens, S. A. Sorensen, J. Berg, C. Lee, T. Jarsky, J. Ting, S. M. Sunkin, D. Feng, C. A. Anastassiou, E. Barkan, et al. "Classification of electrophysiological and morphological neuron types in the mouse visual cortex." In: *Nature Neuroscience* 22.7 (2019), pp. 1182–1195. DOI: 10.1038/s41593-019-0417-0.

[19] R. D. Hodge, T. E. Bakken, J. A. Miller, K. A. Smith, E. R. Barkan, L. T. Graybuck, J. L. Close, B. Long, N. Johansen, O. Penn, et al. "Conserved cell types with divergent features in human versus mouse cortex." In: *Nature* 573.7772 (2019), pp. 61–68. DOI: 10.1038/s41586-019-1506-7.

[20] A. M. Packer and R. Yuste. "Dense, unspecific connectivity of neocortical parvalbumin-positive interneurons: a canonical microcircuit for inhibition?" In: *Journal of Neuroscience* 31.37 (2011), pp. 13260–13271. DOI: 10.1523/JNEUROSCI.3131-11.2011.

[21] R. Perin, T. K. Berger, and H. Markram. "A synaptic organizing principle for cortical neuronal groups." In: *Proceedings of the National Academy of Sciences* 108.13 (2011), pp. 5419–5424. DOI: 10.1073/pnas.1016051108.

[22] M. Ercsey-Ravasz, N. T. Markov, C. Lamy, D. C. Van Essen, K. Knoblauch, Z. Toroczkai, and H. Kennedy. "A predictive network model of cerebral cortical connectivity based on a distance rule." In: *Neuron* 80.1 (2013), pp. 184–197. DOI: 10.1016/j.neuron.2013.07.036.

[23] D. J. Felleman and D. C. Van Essen. "Distributed hierarchical processing in the primate cerebral cortex." In: *Cerebral Cortex* 1.1 (1991), pp. 1–47. DOI: 10.1093/cercor/1.1.1-a.

[24] N. Voges, A. Schüz, A. Aertsen, and S. Rotter. "A modeler's view on the spatial structure of intrinsic horizontal connectivity in the neocortex." In: *Progress in Neurobiology* 92.3 (2010), pp. 277–292. DOI: 10.1016/j.pneurobio.2010.05.001.

[25] M. Schmidt, R. Bakker, C. C. Hilgetag, M. Diesmann, and S. J. van Albada. "Multi-scale account of the network structure of macaque visual cortex." In: *Brain Structure and Function* 223.3 (2018), pp. 1409–1435. DOI: 10.1007/s00429-017-1554-4.

[26] T. C. Potjans and M. Diesmann. "The cell-type specific cortical microcircuit: relating structure and activity in a full-scale spiking network model." In: *Cerebral Cortex* 24.3 (2014), pp. 785–806. DOI: 10.1093/cercor/bhs358.

[27] S. Kunkel, T. C. Potjans, A. Morrison, and M. Diesmann. "Simulating macroscale brain circuits with microscale resolution." In: *Frontiers in Neuroinformatics*. Proceedings of the 2nd INCF congress. 2009. DOI: 10.3389/conf.neuro.

[28] J. Senk, B. Kriener, M. Djurfeldt, N. Voges, H.-J. Jiang, L. Schüttler, G. Gramelsberger, M. Diesmann, H. E. Plesser, and S. J. van Albada. "Connectivity concepts in neuronal network modeling." In: *PLOS Computational Biology* 18.9 (2022), e1010086. DOI: 10.1371/journal.pcbi.1010086.

[29] N. Kasthuri, K. J. Hayworth, D. R. Berger, R. L. Schalek, J. A. Conchello, S. Knowles-Barley, D. Lee, A. Vázquez-Reina, V. Kaynig, T. R. Jones, et al. "Saturated reconstruction of a volume of neocortex." In: *Cell* 162.3 (2015), pp. 648–661. DOI: 10.1016/j.cell.2015.06.054.

[30] L. C. Sincich and G. G. Blasdel. “Oriented Axon Projections in Primary Visual Cortex of the Monkey.” In: *Journal of Neuroscience* 21.12 (2001), pp. 4416–4426. DOI: 10.1523/JNEUROSCI.21-12-04416.2001.

[31] N. T. Markov, J. Vezoli, P. Chameau, A. Falchier, R. Quilodran, C. Huissoud, C. Lamy, P. Misery, P. Giroud, S. Ullman, et al. “Anatomy of hierarchy: feedforward and feedback pathways in macaque visual cortex.” In: *Journal of Comparative Neurology* 522.1 (2013), pp. 225–259. DOI: 10.1002/cne.23458.

[32] C. L. Rees, K. Moradi, and G. A. Ascoli. “Weighing the evidence in Peters’ rule: does neuronal morphology predict connectivity?” In: *Trends in Neurosciences* 40.2 (2017), pp. 63–71. DOI: 10.1016/j.tins.2016.11.007.

[33] U. Roth, F. Eckardt, A. Jahnke, and H. Klar. “Efficient On-Line Computation of Connectivity: Architecture of the Connection Unit of NESPINN.” In: *Proceedings of the MicroNeuro*. Dresden, 1997, pp. 31–39.

[34] J. C. Knight and T. Nowotny. “Larger GPU-accelerated brain simulations with procedural connectivity.” In: *Nature Computational Science* 1.2 (2021), pp. 136–142. DOI: 10.1038/s43588-020-00022-7.

[35] P. Strata and R. Harvey. “Dale’s principle.” In: *Brain Research Bulletin* 50.5-6 (1999), pp. 349–350. DOI: 10.1016/S0361-9230(99)00100-8.

[36] S. Kunkel, M. Schmidt, J. M. Eppler, H. E. Plesser, G. Masumoto, J. Igarashi, S. Ishii, T. Fukai, A. Morrison, M. Diesmann, et al. “Spiking network simulation code for petascale computers.” In: *Frontiers in Neuroinformatics* 8 (2014), p. 78. DOI: 10.3389/fninf.2014.00078.

[37] E. R. Kandel, J. H. Schwartz, T. M. Jessell, S. A. Siegelbaum, and A. J. Hudspeth. “Principles of neural science.” In: 5th. McGraw-Hill, 2012, p. 1472.

[38] S. J. van Albada, M. Helias, and M. Diesmann. “Scalability of Asynchronous Networks Is Limited by One-to-One Mapping between Effective Connectivity and Correlations.” In: *PLOS Computational Biology* 11.9 (2015), e1004490. DOI: 10.1371/journal.pcbi.1004490.

[39] J. Jordan, T. Ippen, M. Helias, I. Kitayama, M. Sato, J. Igarashi, M. Diesmann, and S. Kunkel. “Extremely Scalable Spiking Neuronal Network Simulation Code: From Laptops to Exascale Computers.” In: *Frontiers in Neuroinformatics* 12 (2018), p. 2. DOI: 10.3389/fninf.2018.00002.

[40] A. Destexhe, M. Rudolph, and D. Paré. “The high-conductance state of neocortical neurons in vivo.” In: *Nature Reviews Neuroscience* 4.9 (2003), pp. 739–751. DOI: 10.1038/nrn1198.

[41] A. Maksimov, M. Diesmann, and S. J. van Albada. “Criteria on balance, stability, and excitability in cortical networks for constraining computational models.” In: *Frontiers in Computational Neuroscience* 12 (2018), p. 44. DOI: 10.3389/fncom.2018.00044.

[42] W. Gerstner, W. M. Kistler, R. Naud, and L. Paninski. *Neuronal dynamics: From single neurons to networks and models of cognition*. Cambridge University Press, 2014. DOI: 10.1017/CBO9781107447615.

[43] W. Potjans, A. Morrison, and M. Diesmann. "Enabling functional neural circuit simulations with distributed computing of neuromodulated plasticity." In: *Frontiers in Computational Neuroscience* 4 (2010), p. 141. DOI: 10.3389/fncom.2010.00141.

[44] M. E. Larkum, J. Wu, S. A. Duverdin, and A. Gidon. "The Guide to Dendritic Spikes of the Mammalian Cortex In Vitro and In Vivo." In: *Neuroscience* 489 (2022), pp. 15–33. DOI: 10.1016/j.neuroscience.2022.02.009.

[45] S. Jahnke, M. Timme, and R.-M. Memmesheimer. "Guiding Synchrony through Random Networks." In: *Physical Review X* 2.4 (2012), p. 041016. DOI: 10.1103/physrevx.2.041016.

[46] Y. Bouhadjar, D. J. Wouters, M. Diesmann, and T. Tetzlaff. "Sequence learning, prediction, and replay in networks of spiking neurons." In: *PLOS Computational Biology* 18.6 (2022), e1010233. DOI: 10.1371/journal.pcbi.1010233.

[47] M. Tsodyks, K. Pawelzik, and H. Markram. "Neural networks with dynamic synapses." In: *Neural Computation* 10.4 (1998), pp. 821–835. DOI: 10.1162/089976698300017502.

[48] R. Brette and W. Gerstner. "Adaptive exponential integrate-and-fire model as an effective description of neuronal activity." In: *Journal of Neurophysiology* 94.5 (2005), pp. 3637–3642. DOI: 10.1152/jn.00686.2005.

[49] S. Rotter and M. Diesmann. "Exact digital simulation of time-invariant linear systems with applications to neuronal modeling." In: *Biological Cybernetics* 81.5-6 (1999), pp. 381–402. DOI: 10.1007/s004220050570.

[50] A. Morrison, S. Straube, H. E. Plesser, and M. Diesmann. "Exact Subthreshold Integration with Continuous Spike Times in Discrete-Time Neural Network Simulations." In: *Neural Computation* 19.1 (2007), pp. 47–79. DOI: 10.1162/neco.2007.19.1.47.

[51] A. Morrison and M. Diesmann. "Maintaining Causality in Discrete Time Neuronal Network Simulations." In: *Lectures in Supercomputational Neurosciences: Dynamics in Complex Brain Networks*. Springer, 2007, pp. 267–278. DOI: 10.1007/978-3-540-73159-7_10.

[52] J. Hahne, M. Helias, S. Kunkel, J. Igarashi, M. Bolten, A. Frommer, and M. Diesmann. "A unified framework for spiking and gap-junction interactions in distributed neuronal network simulations." In: *Frontiers in Neuroinformatics* 9 (2015), p. 22. DOI: 10.3389/fninf.2015.00022.

[53] J. Jordan, M. Helias, M. Diesmann, and S. Kunkel. "Efficient Communication in Distributed Simulations of Spiking Neuronal Networks With Gap Junctions." In: *Frontiers in Neuroinformatics* 14 (2020), p. 12. DOI: 10.3389/fninf.2020.00012.

[54] D. D. Stettler, H. Yamahachi, W. Li, W. Denk, and C. D. Gilbert. "Axons and synaptic boutons are highly dynamic in adult visual cortex." In: *Neuron* 49.6 (2006), pp. 877–887. DOI: 10.1016/j.neuron.2006.02.018.

[55] M. Butz, F. Wörgötter, and A. van Ooyen. "Activity-dependent structural plasticity." In: *Brain Research Reviews* 60.2 (2009), pp. 287–305. DOI: 10.1016/j.brainresrev.2008.12.023.

[56] M. Butz and A. van Ooyen. "A simple rule for dendritic spine and axonal bouton formation can account for cortical reorganization after focal retinal lesions." In: *PLOS Computational Biology* 9.10 (2013), e1003259. DOI: 10.1371/journal.pcbi.1003259.

[57] S. A. Bamford, A. F. Murray, and D. J. Willshaw. "Synaptic rewiring for topographic mapping and receptive field development." In: *Neural Networks* 23.4 (2010), pp. 517–527. DOI: 10.1016/j.neunet.2010.01.005.

[58] J. V. Gallinaro and S. Rotter. "Associative properties of structural plasticity based on firing rate homeostasis in recurrent neuronal networks." In: *Scientific Reports* 8.1 (2018), pp. 1–13. DOI: 10.1038/s41598-018-22077-3.

[59] G. Kalantzis and H. Z. Shouval. "Structural plasticity can produce metaplasticity." In: *PLOS One* 4.11 (2009), e8062. DOI: 10.1371/journal.pone.0008062.

[60] G. Bellec, D. Kappel, W. Maass, and R. Legenstein. "Deep rewiring: Training very sparse deep networks." In: *arXiv* 1711.05136 (2017). DOI: 10.48550/arXiv.1711.05136.

[61] A. Knoblauch. "Impact of structural plasticity on memory formation and decline." In: *The rewiring brain*. Elsevier, 2017, pp. 361–386. DOI: 10.1016/B978-0-12-803784-3.00017-2.

[62] G. Turrigiano. "Homeostatic Synaptic Plasticity: Local and Global Mechanisms for Stabilizing Neuronal Function." In: *Cold Spring Harbor Perspectives in Biology* 4.1 (2012), a005736–a005736. DOI: 10.1101/cshperspect.a005736.

[63] D. Kappel, S. Habenschuss, R. Legenstein, and W. Maass. "Network plasticity as Bayesian inference." In: *PLOS Computational Biology* 11.11 (2015), e1004485. DOI: 10.1371/journal.pcbi.1004485.

[64] S. Billaudelle, B. Cramer, M. A. Petrovici, K. Schreiber, D. Kappel, J. Schemmel, and K. Meier. "Structural plasticity on an accelerated analog neuromorphic hardware system." In: *Neural Networks* 133 (2021), pp. 11–20. DOI: 10.1016/j.neunet.2020.09.024.

[65] R. George, G. Indiveri, and S. Vassanelli. “Activity dependent structural plasticity in neuromorphic systems.” In: *IEEE Biomedical Circuits and Systems Conference (BioCAS)*. IEEE. 2017, pp. 1–4. DOI: 10.1109/BIOCAS.2017.8325074.

[66] A. van Ooyen and M. Butz-Ostendorf. *The rewiring brain: a computational approach to structural plasticity in the adult brain*. Academic Press, 2017. DOI: 10.1016/B978-0-12-803784-3.00033-0.

[67] J. Iglesias, J. Eriksson, F. Grize, M. Tomassini, and A. E. Villa. “Dynamics of pruning in simulated large-scale spiking neural networks.” In: *Biosystems* 79.1-3 (2005), pp. 11–20. DOI: 10.1016/j.biosystems.2004.09.016.

[68] J. Hawkins and S. Ahmad. “Why neurons have thousands of synapses, a theory of sequence memory in neocortex.” In: *Frontiers in Neural Circuits* (2016), p. 23. DOI: 10.3389/fncir.2016.00023.

[69] S. Roy, A. Banerjee, and A. Basu. “Liquid state machine with dendritically enhanced readout for low-power, neuromorphic VLSI implementations.” In: *IEEE Transactions on Biomedical Circuits and Systems* 8.5 (2014), pp. 681–695. DOI: 10.1109/TBCAS.2014.2362969.

[70] M. A. Bourjaily and P. Miller. “Synaptic plasticity and connectivity requirements to produce stimulus-pair specific responses in recurrent networks of spiking neurons.” In: *PLOS Computational Biology* 7.2 (2011), e1001091. DOI: 10.1371/journal.pcbi.1001091.

[71] M. Butz, I. D. Steenbuck, and A. van Ooyen. “Homeostatic structural plasticity increases the efficiency of small-world networks.” In: *Frontiers in Synaptic Neuroscience* 6 (2014), p. 7. DOI: 10.3389/fnsyn.2014.00007.

[72] M. Fauth and C. Tetzlaff. “Opposing Effects of Neuronal Activity on Structural Plasticity.” In: *Frontiers in Neuroanatomy* 10 (2016). DOI: 10.3389/fnana.2016.00075.

[73] S. Diaz-Pier, M. Naveau, M. Butz-Ostendorf, and A. Morrison. “Automatic generation of connectivity for large-scale neuronal network models through structural plasticity.” In: *Frontiers in Neuroanatomy* 10 (2016), p. 57. DOI: 10.3389/fnana.2016.00057.

[74] C. Nowke, S. Diaz-Pier, B. Weyers, B. Hentschel, A. Morrison, T. W. Kuhlen, and A. Peyser. “Toward rigorous parameterization of underconstrained neural network models through interactive visualization and steering of connectivity generation.” In: *Frontiers in Neuroinformatics* 12 (2018), p. 32. DOI: 10.3389/fninf.2018.00032.

[75] D. O. Hebb. *The Organisation of Behaviour: A Neuropsychological Theory*. Wiley, 1949. DOI: 10.4324/9781410612403.

[76] J. C. Magee and C. Grienberger. “Synaptic Plasticity Forms and Functions.” In: *Annual Review of Neuroscience* 43 (2020), pp. 95–117. DOI: 10.1146/annurev-neuro-090919-022842.

[77] A. Morrison, M. Diesmann, and W. Gerstner. "Phenomenological models of synaptic plasticity based on spike timing." In: *Biological Cybernetics* 98 (2008), pp. 459–478. DOI: 10.1007/s00422-008-0233-1.

[78] S. Okabe, H.-D. Kim, A. Miwa, T. Kuriu, and H. Okado. "Continual remodeling of postsynaptic density and its regulation by synaptic activity." In: *Nature Neuroscience* 2.9 (1999), pp. 804–811. DOI: 10.1038/12175.

[79] M. V. Tsodyks and H. Markram. "The neural code between neocortical pyramidal neurons depends on neurotransmitter release probability." In: *Proceedings of the National Academy of Sciences* 94 (1997), pp. 719–723. DOI: 10.1073/pnas.94.2.719.

[80] H. Markram, Y. Wang, and M. Tsodyks. "Differential signaling via the same axon of neocortical pyramidal neurons." In: *Proceedings of the National Academy of Sciences* 95.9 (1998), pp. 5323–5328. DOI: 10.1073/pnas.95.9.5323.

[81] A. Gupta, Y. Wang, and H. Markram. "Organizing principles for a diversity of GABAergic interneurons and synapses in the neocortex." In: *Science* 287.5451 (2000), pp. 273–278. DOI: 10.1126/science.287.5451.273.

[82] R. S. Zucker and W. G. Regehr. "Short-Term Synaptic Plasticity." In: *Annual Review of Physiology* 64.1 (2002), pp. 355–405. DOI: 10.1146/annurev.physiol.64.092501.114547.

[83] H. Markram, J. Lübke, M. Frotscher, A. Roth, and B. Sakmann. "Physiology and anatomy of synaptic connections between thick tufted pyramidal neurones in the developing rat neocortex." In: *Journal of Physiology* 500.2 (1997), pp. 409–440. DOI: 10.1113/jphysiol.1997.sp022031.

[84] G.-q. Bi and M.-m. Poo. "Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type." In: *Journal of Neuroscience* 18.24 (1998), pp. 10464–10472. DOI: 10.1523/JNEUROSCI.18-24-10464.1998.

[85] N. Caporale and Y. Dan. "Spike timing–dependent plasticity: a Hebbian learning rule." In: *Annual Reviews Neuroscience* 31 (2008), pp. 25–46. DOI: 10.1146/annurev.neuro.31.060407.125639.

[86] H. Markram, W. Gerstner, and P. J. Sjöström. "Spike-timing-dependent plasticity: a comprehensive overview." In: *Frontiers in Synaptic Neuroscience* 4 (2012), p. 2. DOI: 10.3389/fnsyn.2012.00002.

[87] Z. Brzosko, S. B. Mierau, and O. Paulsen. "Neuromodulation of spike-timing-dependent plasticity: past, present, and future." In: *Neuron* 103.4 (2019), pp. 563–581. DOI: 10.1016/j.neuron.2019.05.041.

[88] Y. Li, Y. Zhong, J. Zhang, L. Xu, Q. Wang, H. Sun, H. Tong, X. Cheng, and X. Miao. “Activity-dependent synaptic plasticity of a chalcogenide electronic synapse for neuromorphic systems.” In: *Scientific Reports* 4.1 (2014), p. 4906. DOI: 10.1038/srep04906.

[89] A. Morrison, A. Aertsen, and M. Diesmann. “Spike-Timing-Dependent Plasticity in Balanced Random Networks.” In: *Neural Computation* 19.6 (2007), pp. 1437–1467. DOI: 10.1162/neco.2007.19.6.1437.

[90] M. C. Van Rossum, G. Q. Bi, and G. G. Turrigiano. “Stable Hebbian learning from spike timing-dependent plasticity.” In: *Journal of Neuroscience* 20.23 (2000), pp. 8812–8821. DOI: 10.1523/JNEUROSCI.20-23-08812.2000.

[91] A. N. Burkitt, H. Meffin, and D. B. Grayden. “Spike-timing-dependent plasticity: the relationship to rate-based learning for models with weight dynamics determined by a stable fixed point.” In: *Neural Computation* 16.5 (2004), pp. 885–940. DOI: 10.1162/089976604773135041.

[92] C. Clopath and W. Gerstner. “Voltage and spike timing interact in STDP–a unified model.” In: *Frontiers in Synaptic Neuroscience* 2 (2010), p. 25. DOI: 10.3389/fnsyn.2010.00025.

[93] C. Clopath, L. Büsing, E. Vasilaki, and W. Gerstner. “Connectivity reflects coding: a model of voltage-based STDP with homeostasis.” In: *Nature Neuroscience* 13 (2010), pp. 344–352. DOI: 10.1038/nn.2479.

[94] A. Ngezahayo, M. Schachner, and A. Artola. “Synaptic Activity Modulates the Induction of Bidirectional Synaptic Changes in Adult Mouse Hippocampus.” In: *Journal of Neuroscience* 20.7 (2000), pp. 2451–2458. DOI: 10.1523/JNEUROSCI.20-07-02451.2000.

[95] P. J. Sjöström, G. G. Turrigiano, and S. B. Nelson. “Rate, timing, and cooperativity jointly determine cortical synaptic plasticity.” In: *Neuron* 32.6 (2001), pp. 1149–1164. DOI: 10.1016/S0896-6273(01)00542-6.

[96] R. Urbanczik and W. Senn. “Learning by the Dendritic Prediction of Somatic Spiking.” In: *Neuron* 81.3 (2014), pp. 521–528. DOI: 10.1016/j.neuron.2013.11.030.

[97] A. Artola, S. Bröcher, and W. Singer. “Different voltage dependent thresholds for inducing long-term depression and long-term potentiation in slices of rat visual cortex.” In: *Nature* 347 (1990), pp. 69–72. DOI: 10.1038/347069a0.

[98] V. Pawlak, J. R. Wickens, A. Kirkwood, and J. N. Kerr. “Timing is not everything: neuromodulation opens the STDP gate.” In: *Frontiers in Synaptic Neuroscience* 2 (2010), p. 146. DOI: 10.3389/fnsyn.2010.00146.

[99] F. Wörgötter and B. Porr. “Temporal sequence learning, prediction, and control: a review of different models and their relation to biological mechanisms.” In: *Neural Computation* 17.2 (2005), pp. 245–319. DOI: 10.1162/0899766053011555.

[100] J. Stapmanns, J. Hahne, M. Helias, M. Bolten, M. Diesmann, and D. Dahmen. "Event-based Update of Synapses in Voltage-Based Learning Rules." In: *Frontiers in Neuroinformatics* 15 (2021), p. 609147. DOI: 10.3389/fninf.2021.609147.

[101] A. Morrison, C. Mehring, T. Geisel, A. Aertsen, and M. Diesmann. "Advancing the Boundaries of High-Connectivity Network Simulation with Distributed Computing." In: *Neural Computation* 17.8 (2005), pp. 1776–1801. DOI: 10.1162/0899766054026648.

[102] P. Weidel, R. Duarte, and A. Morrison. "Unsupervised Learning and Clustered Connectivity Enhance Reinforcement Learning in Spiking Neural Networks." In: *Frontiers in Computational Neuroscience* 15 (2021), p. 543872. DOI: 10.3389/fncom.2021.543872.

[103] T. P. Vogels, H. Sprekeler, F. Zenke, C. Clopath, and W. Gerstner. "Inhibitory plasticity balances excitation and inhibition in sensory pathways and memory networks." In: *Science* 334.6062 (2011), pp. 1569–1573. DOI: 10.1126/science.1211095.

[104] J. Jordan, M. Schmidt, W. Senn, and M. A. Petrovici. "Evolving interpretable plasticity for spiking networks." In: *eLife* 10 (2021), e66273. DOI: 10.7554/eLife.66273.

[105] A. B. Barrett, G. O. Billings, R. G. M. Morris, and M. C. W. van Rossum. "State Based Model of Long-Term Potentiation and Synaptic Tagging and Capture." In: *PLOS Computational Biology* 5.1 (2009). Ed. by L. J. Graham, e1000259. DOI: 10.1371/journal.pcbi.1000259.

[106] G. Mongillo, O. Barak, and M. Tsodyks. "Synaptic Theory of Working Memory." In: *Science* 319.5869 (2008), pp. 1543–1546. DOI: 10.1126/science.1150769.

[107] C. Koch and G. Laurent. "Complexity and the nervous system." In: *Science* 284.5411 (1999), pp. 96–98. DOI: 10.1126/science.284.5411.96.

[108] R. Duarte and A. Morrison. "Leveraging heterogeneity for neural computation with fading memory in layer 2/3 cortical microcircuits." In: *PLOS Computational Biology* 15.4 (2019), e1006781. DOI: 10.1371/journal.pcbi.1006781.

[109] V. Parpura and R. Zorec. "Gliotransmission: exocytotic release from astrocytes." In: *Brain Research Reviews* 63.1-2 (2010), pp. 83–92. DOI: 10.1016/j.brainresrev.2009.11.008.

[110] P. Girard, J. M. Hupé, and J. Bullier. "Feedforward and Feedback Connections Between Areas V1 and V2 of the Monkey Have Similar Rapid Conduction Velocities." In: *Journal of Neurophysiology* 85.3 (2001), pp. 1328–1331. DOI: 10.1152/jn.2001.85.3.1328.

[111] D. Liewald, R. Miller, N. Logothetis, H.-J. Wagner, and A. Schüz. "Distribution of axon diameters in cortical white matter: an electron-microscopic study on three human brains and a macaque." In: *Biological Cybernetics* 108.5 (2014), pp. 541–557. DOI: 10.1007/s00422-014-0626-2.

[112] L. Muller, F. Chavane, J. Reynolds, and T. J. Sejnowski. "Cortical travelling waves: mechanisms and computational principles." In: *Nature Reviews Neuroscience* 19.5 (2018), pp. 255–268. DOI: 10.1038/nrn.2018.20.

[113] S. Song, P. Sjöström, M. Reigl, S. Nelson, and D. Chklovskii. "Highly nonrandom features of synaptic connectivity in local cortical circuits." In: *PLOS Computational Biology* 3.3 (2005), e68. DOI: 10.1371/journal.pbio.0030068.

[114] S. Lefort, C. Tomm, J.-C. F. Sarria, and C. C. H. Petersen. "The Excitatory Neuronal Network of the C2 Barrel Column in Mouse Primary Somatosensory Cortex." In: *Neuron* 61 (2009), pp. 301–316. DOI: 10.1016/j.neuron.2008.12.020.

[115] A. A. Koulakov, T. Hromadka, and A. M. Zador. "Correlated Connectivity and the Distribution of Firing Rates in the Neocortex." In: *Journal of Neuroscience* 29.12 (2009), pp. 3685–3694. DOI: 10.1523/JNEUROSCI.4500-08.2009.

[116] M. Avermann, C. Tomm, C. Mateo, W. Gerstner, and C. C. Petersen. "Microcircuits of excitatory and inhibitory neurons in layer 2/3 of mouse barrel cortex." In: *Journal of Neurophysiology* 107.11 (2012), pp. 3116–3134. DOI: 10.1152/jn.00917.2011.

[117] Y. Ikegaya, T. Sasaki, D. Ishikawa, N. Honma, K. Tao, N. Takahashi, G. Minamisawa, S. Ujita, and N. Matsuki. "Interpyramid Spike Transmission Stabilizes the Sparseness of Recurrent Network Activity." In: *Cerebral Cortex* 23.2 (2013), pp. 293–304. DOI: 10.1093/cercor/bhs006.

[118] A. Kuhn, A. Aertsen, and S. Rotter. "Neuronal integration of synaptic input in the fluctuation-driven regime." In: *Journal of Neuroscience* 24.10 (2004), pp. 2345–2356. DOI: 10.1523/JNEUROSCI.3349-03.2004.

[119] A. Roxin, N. Brunel, D. Hansel, G. Mongillo, and C. van Vreeswijk. "On the distribution of firing rates in networks of cortical neurons." In: *Journal of Neuroscience* 31.45 (2011), pp. 16217–16226. DOI: 10.1523/jneurosci.1677-11.2011.

[120] N. Brunel and V. Hakim. "Fast Global Oscillations in Networks of Integrate-and-Fire Neurons with Low Firing Rates." In: *Neural Computation* 11.7 (1999), pp. 1621–1671. DOI: 10.1162/089976699300016179.

[121] B. M. Kampa, J. J. Letzkus, and G. J. Stuart. "Dendritic mechanisms controlling spike-timing-dependent synaptic plasticity." In: *Trends in Neuroscience* 30.9 (2007), pp. 456–463. DOI: 10.1016/j.tins.2007.06.010.

[122] D. Feldmeyer, V. Egger, J. Lübke, and B. Sakmann. “Reliable synaptic connections between pairs of excitatory layer 4 neurones within a single ‘barrel’ of developing rat somatosensory cortex.” In: *Journal of Physiology* 521.1 (1999), pp. 169–190. DOI: 10.1111/j.1469-7793.1999.00169.x.

[123] D. Feldmeyer, J. Lübke, R. A. Silver, and B. Sakmann. “Synaptic connections between layer 4 spiny neurone-layer 2/3 pyramidal cell pairs in juvenile rat barrel cortex: physiology and anatomy of interlaminar signalling within a cortical column.” In: *Journal of Physiology* 538.3 (2002), pp. 803–822. DOI: 10.1113/jphysiol.2001.012959.

[124] D. Feldmeyer, J. Lübke, and B. Sakmann. “Efficacy and connectivity of intracolumnar pairs of layer 2/3 pyramidal cells in the barrel cortex of juvenile rats.” In: *Journal of Physiology* 575 (2006), pp. 583–602. DOI: 10.1113/jphysiol.2006.105106.

[125] A. Stepanyants, J. Hirsch, L. M. Martinez, Z. F. Kisvarday, A. S. Ferecsko, and D. B. Chklovskii. “Local potential connectivity in cat primary visual cortex.” In: *Cerebral Cortex* 18.1 (2008), pp. 13–28. DOI: 10.1093/cercor/bhm027.

[126] A. Roxin. “The role of degree distribution in shaping the dynamics in networks of sparsely connected spiking neurons.” In: *Frontiers in Computational Neuroscience* 5.8 (2011). DOI: 10.3389/fncom.2011.00008.

[127] C. van Vreeswijk and H. Sompolinsky. “Chaotic balanced state in a model of cortical circuits.” In: *Neural Computation* 10.6 (1998), pp. 1321–1371. DOI: 10.1162/089976698300017214.

[128] M. Tsodyks, I. Mitkov, and H. Sompolinsky. “Pattern of synchrony in inhomogeneous networks of oscillators with pulse interactions.” In: *Physical Review Letters* 71.8 (1993). DOI: 10.1103/PhysRevLett.71.1280.

[129] D. Golomb and J. Rinzel. “Dynamics of globally coupled inhibitory neurons with heterogeneity.” In: *Physical Review Letters* 48.6 (1993), pp. 4810–4814. DOI: 10.1103/PhysRevE.48.4810.

[130] L. Neltner, D. Hansel, G. Mato, and C. Meunier. “Synchrony in Heterogeneous Networks of Spiking Neurons.” In: *Neural Computation* 12.7 (2000), pp. 1607–1641. DOI: 10.1162/089976600300015286.

[131] M. Denker, M. Timme, M. Diesmann, F. Wolf, and T. Geisel. “Breaking Synchrony by Heterogeneity in Complex Networks.” In: *Physical Review Letters* 92.7 (2004), pp. 074103-1–074103-4. DOI: 10.1103/PhysRevLett.92.074103.

[132] J. F. Mejias and A. Longtin. “Optimal Heterogeneity for Coding in Spiking Neural Networks.” In: *Physical Review Letters* 108 (2012). DOI: 10.1103/PhysRevLett.108.228102.

[133] T. Pfeil, J. Jordan, T. Tetzlaff, A. Grübl, J. Schemmel, M. Diesmann, and K. Meier. "Effect of Heterogeneity on Decorrelation Mechanisms in Spiking Neural Networks: A Neuromorphic-Hardware Study." In: *Physical Review X* 6 (2 2016), p. 021023. DOI: 10.1103/PhysRevX.6.021023.

[134] N. G. Stocks. "Suprathreshold stochastic resonance in multilevel threshold systems." In: *Physical Review Letters* 84.11 (2000), p. 2310. DOI: 10.1103/PhysRevLett.84.2310.

[135] M. Shamir and H. Sompolinsky. "Implications of neuronal diversity on population coding." In: *Neural Computation* 18.8 (2006), pp. 1951–1986. DOI: 10.1162/neco.2006.18.8.1951.

[136] M. I. Chelaru and V. Dragoi. "Efficient coding in heterogeneous neuronal populations." In: *Proceedings of the National Academy of Sciences* 105.42 (2008). DOI: 10.1073/pnas.0807744105.

[137] L. C. Osborne, S. E. Palmer, S. G. Lisberger, and W. Bialek. "The neural basis for combinatorial coding in a cortical population response." In: *Journal of Neuroscience* 28.50 (2008), pp. 13522–13531. DOI: 10.1523/JNEUROSCI.4390-08.2008.

[138] K. Padmanabhan and N. N. Urban. "Intrinsic biophysical diversity decorrelates neuronal firing while increasing information content." In: *Nature Neuroscience* 13.10 (2010), pp. 1276–1282. DOI: 10.1038/nn.2630.

[139] G. Marsat and L. Maler. "Neural Heterogeneity and Efficient Population Codes for Communication Signals." In: *Journal of Neurophysiology* 104.5 (2010), pp. 2543–2555. DOI: 10.1152/jn.00256.2010.

[140] L. A. Holmstrom, L. B. Eeuwes, P. D. Roberts, and C. V. Portfors. "Efficient encoding of vocalizations in the auditory midbrain." In: *Journal of Neuroscience* 30.3 (2010), pp. 802–819. DOI: 10.1523/JNEUROSCI.1964-09.2010.

[141] M. Y. Yim, A. Aertsen, and S. Rotter. "Impact of intrinsic biophysical diversity on the activity of spiking neurons." In: *Physical Review E* 87 (2013). DOI: 10.1103/PhysRevE.87.032710.

[142] J. Lengler, F. Jug, and A. Steger. "Reliable Neuronal Systems: The Importance of Heterogeneity." In: *PLOS One* (2013). DOI: 10.1371/journal.pone.0080694.

[143] J. F. Mejias and A. Longtin. "Differential effects of excitatory and inhibitory heterogeneity on the gain and asynchronous state of sparse cortical networks." In: *Frontiers in Computational Neuroscience* 8 (2014). DOI: 10.3389/fncom.2014.00107.

[144] D. Battaglia, A. Karagiannis, T. Gallopin, H. W. Gutch, and B. Cauli. "Beyond the frontiers of neuronal types." In: *Frontiers in Neural Circuits* 7 (2013), p. 13. DOI: 10.3389/fncir.2013.00013.

[145] G. Buzsáki and K. Mizuseki. "The log-dynamic brain: how skewed distributions affect network operations." In: *Nature Reviews Neuroscience* 15.4 (2014), pp. 264–278. DOI: 10.1038/nrn3687.

[146] P. Robinson, X. Gao, and Y. Han. "Relationships between lognormal distributions of neural properties, activity, criticality, and connectivity." In: *Biological Cybernetics* 115.2 (2021), pp. 121–130. DOI: 10.1007/s00422-021-00871-z.

[147] A. Morales-Gregorio, A. van Meegen, and S. J. van Albada. "Ubiquitous lognormal distribution of neuron densities across mammalian cerebral cortex." In: *bioRxiv* (2022). DOI: 10.1101/2022.03.17.480842.

[148] A. A. Prinz, D. Bucher, and E. Marder. "Similar network activity from disparate circuit parameters." In: *Nature Neuroscience* 7 (2004), pp. 1345–1352. DOI: 10.1038/nn1352.

[149] P. Achard and E. De Schutter. "Complex parameter landscape for a complex neuron model." In: *PLOS Computational Biology* 2.7 (2006), e94. DOI: 10.1371/journal.pcbi.0020094.

[150] J. Bahuguna, T. Tetzlaff, A. Kumar, J. H. Kotaleski, and A. Morrison. "Homologous basal ganglia network models in physiological and Parkinsonian conditions." In: *Frontiers in Computational Neuroscience* 11 (2017), p. 79. DOI: 10.3389/fncom.2017.00079.

[151] L. F. Abbott and S. B. Nelson. "Synaptic plasticity: taming the beast." In: *Nature Neuroscience* 3 (2000), pp. 1178–1183. DOI: 10.1038/81453.

[152] C. Tetzlaff, C. Kolodziejski, M. Timme, and F. Wörgötter. "Synaptic scaling in combination with many generic plasticity mechanisms stabilizes circuit connectivity." In: *Frontiers in Computational Neuroscience* 5 (2011), p. 47. DOI: 10.3389/fncom.2011.00047.

[153] C. Eliasmith and C. H. Anderson. *Neural engineering: Computation, representation, and dynamics in neurobiological systems*. MIT press, 2002. URL: https://mitpress.mit.edu/9780262050715/.

[154] G. Bellec, F. Scherr, A. Subramoney, E. Hajek, D. Salaj, R. Legenstein, and W. Maass. "A solution to the learning dilemma for recurrent networks of spiking neurons." In: *Nature Communications* 11.1 (2020), p. 3625. DOI: 10.1038/s41467-020-17236-y.

[155] S. Furber. "Large-scale neuromorphic computing systems." In: *Journal of Neural Engineering* 13.5 (2016), p. 051001. DOI: 10.1088/1741-2560/13/5/051001.

[156] T. Wunderlich et al. "Demonstrating Advantages of Neuromorphic Computation: A Pilot Study." In: *Frontiers in Neuroscience* 13 (2019). DOI: 10.3389/fnins.2019.00260.

[157] Y. V. Zaytsev and A. Morrison. “Increasing quality and managing complexity in neuroinformatics software development with continuous integration.” In: *Frontiers in Neuroinformatics* 6 (2013), p. 31. DOI: 10.3389/fninf.2012.00031.

[158] H. E. Plesser. “Reproducibility vs. replicability: a brief history of a confused terminology.” In: *Frontiers in Neuroinformatics* 11 (2018), p. 76. DOI: 10.3389/fninf.2017.00076.

[159] S. N. Goodman, D. Fanelli, and J. P. A. Ioannidis. “What does research reproducibility mean?” In: *Science Translational Medicine* 8.341 (2016). DOI: 10.1126/scitranslmed.aaf5027.

[160] S. J. van Albada, S. Kunkel, A. Morrison, and M. Diesmann. “Integrating Brain Structure and Dynamics on Supercomputers.” In: *Brain-Inspired Computing*. 2014, pp. 22–32. DOI: 10.1007/978-3-319-12084-3_3.

[161] A. C. Kurth, J. Senk, D. Terhorst, J. Finnerty, and M. Diesmann. “Subrealtime simulation of a neuronal network of natural density.” In: *Neuromorphic Computing and Engineering* 2.2 (2022), p. 021001. DOI: 10.1088/2634-4386/ac55fc.

[162] A. Hanuschkin, S. Kunkel, M. Helias, A. Morrison, and M. Diesmann. “A General and Efficient Method for Incorporating Precise Spike Times in Globally Time-Driven Simulations.” In: *Frontiers in Neuroinformatics* 4 (2010). DOI: 10.3389/fninf.2010.00113.

[163] J. N. Teramae, Y. Tsubo, and T. Fukai. “Optimal spike-based communication in excitable networks with strong-sparse and weak-dense links.” In: *Scientific Reports* 2 (2012), p. 485. DOI: 10.1038/srep00485.

[164] S. Malkin, K. K. Kim, D. Tikhonov, and A. Zaitsev. “Properties of spontaneous and miniature excitatory postsynaptic currents in neurons of the rat prefrontal cortex.” In: *Journal of Evolutionary Biochemistry and Physiology* 50.6 (2014), pp. 506–514. DOI: 10.1134/S0022093014060052.

[165] S. Dasbach, T. Tetzlaff, M. Diesmann, and J. Senk. “Dynamical Characteristics of Recurrent Neuronal Networks Are Robust Against Low Synaptic Weight Resolution.” In: *Fontiers in Neuroscience* 15 (2021). DOI: 10.3389/fnins.2021.757790.

[166] T. Pfeil, T. C. Potjans, S. Schrader, W. Potjans, J. Schemmel, M. Diesmann, and K. Meier. “Is a 4-bit synaptic weight resolution enough?–constraints on enabling spike-timing dependent plasticity in neuromorphic hardware.” In: *Frontiers in Neuroscience* 6 (2012), p. 90. DOI: 10.3389/fnins.2012.00090.

[167] E. Nordlie, M.-O. Gewaltig, and H. E. Plesser. “Towards Reproducible Descriptions of Neuronal Network Models.” In: *PLOS Computational Biology* 5.8 (2009), pp. 1–18. DOI: 10.1371/journal.pcbi.1000456.

[168] I. Blundell, R. Brette, T. A. Cleland, T. G. Close, D. Coca, A. P. Davison, S. Diaz-Pier, C. Fernandez Musoles, P. Gleeson, D. F. Goodman, et al. "Code generation in computational neuroscience: a review of tools and techniques." In: *Frontiers in Neuroinformatics* 12 (2018), p. 68. DOI: 10.3389/fninf.2018.00068.

[169] R. A. McDougal, T. M. Morse, T. Carnevale, L. Marenco, R. Wang, M. Migliore, P. L. Miller, G. M. Shepherd, and M. L. Hines. "Twenty years of ModelDB and beyond: building essential modeling tools for the future of neuroscience." In: *Journal of Computational Neuroscience* 42.1 (2016), pp. 1–10. DOI: 10.1007/s10827-016-0623-7.

[170] P. Gleeson, M. Cantarelli, B. Marin, A. Quintana, M. Earnshaw, S. Sadeh, E. Piasini, J. Birgiolas, R. C. Cannon, N. A. Cayco-Gajic, et al. "Open Source Brain: a collaborative resource for visualizing, analyzing, simulating, and developing standardized models of neurons and circuits." In: *Neuron* 103.3 (2019), pp. 395–411. DOI: 10.1016/j.neuron.2019.05.019.

[171] A. Holtmaat, T. Bonhoeffer, D. K. Chow, J. Chuckowree, V. De Paola, S. B. Hofer, M. Hübener, T. Keck, G. Knott, W.-C. A. Lee, et al. "Long-term, high-resolution imaging in the mouse neocortex through a chronic cranial window." In: *Nature Protocols* 4.8 (2009), pp. 1128–1144. DOI: 10.1038/nprot.2009.89.

[172] S. Foxley, V. Sampathkumar, V. De Andrade, S. Trinkle, A. Sorokina, K. Norwood, P. La Riviere, and N. Kasthuri. "Multi-modal imaging of a single mouse brain over five orders of magnitude of resolution." In: *NeuroImage* 238 (2021), p. 118250. DOI: 10.1016/j.neuroimage.2021.118250.

[173] C. Chatfield. "Model Uncertainty, Data Mining and Statistical Inference." In: *Journal of the Royal Statistical Society. Series A (Statistics in Society)* 158.3 (1995), p. 419. DOI: 10.2307/2983440.

[174] W. Gerstner, M. Lehmann, V. Liakoni, D. Corneil, and J. Brea. "Eligibility Traces and Plasticity on Behavioral Time Scales: Experimental Support of NeoHebbian Three-Factor Learning Rules." In: *Frontiers in Neural Circuits* 12 (2018), p. 53. DOI: 10.3389/fncir.2018.00053.

[175] N. Frémaux and W. Gerstner. "Neuromodulated Spike-Timing-Dependent Plasticity, and Theory of Three-factor Learning Rules." In: *Frontiers in Neural Circuits* 9 (2016), p. 85. DOI: 10.3389/fncir.2015.00085.

[176] P. J. Werbos. "Backpropagation through time: what it does and how to do it." In: *Proceedings of the IEEE* 78.10 (1990), pp. 1550–1560. DOI: 10.1109/5.58337.

[177] M. Abadi et al. "TensorFlow: Large-scale Machine Learning on Heterogeneous Distributed Systems." In: *arXiv* (2016). DOI: 10.48550/arXiv.1603.04467. eprint: 1603.04467v2.

[178] J. Ansel et al. “PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation.” In: *29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2 (ASPLOS ’24)*. ACM, 2024. DOI: 10.1145/3620665.3640366.

[179] B. Sabatini and W. Regehr. “Timing of Synaptic Transmission.” In: *Annual Review of Physiology* 61 (1999), pp. 521–542. DOI: 10.1146/annurev.physiol.61.1.521.

[180] V. K. Jirsa. “Connectivity and dynamics of neural information processing.” In: *Neuroinformatics* 2 (2004), pp. 183–204. DOI: 10.1385/NI:2:2:183.

[181] V. H. Allan, R. B. Jones, R. M. Lee, and S. J. Allan. “Software pipelining.” In: *ACM Computing Surveys (CSUR)* 27.3 (1995), pp. 367–432. DOI: 10.1145/212094.212131.

[182] R. Laje and D. V. Buonomano. “Robust timing and motor patterns by taming chaos in recurrent neural networks.” In: *Nature Neuroscience* 16.7 (2013), pp. 925–933. DOI: 10.1038/nn.3405.

[183] G. Orchard, A. Jayawant, G. K. Cohen, and N. Thakor. “Converting Static Image Datasets to Spiking Neuromorphic Datasets Using Saccades.” In: *Frontiers in Neuroscience* 9 (2015), p. 437. DOI: 10.3389/fnins.2015.00437.

[184] D. Budik and I. Elhanany. “TRTRL: A localized resource-efficient learning algorithm for recurrent neural netowrks.” In: *49th IEEE International Midwest Symposium on Circuits and Systems*. Vol. 1. IEEE, 2006, pp. 371–374. DOI: 10.1109/MWSCAS.2006.382075.

[185] R. J. Williams and J. Peng. “An Efficient Gradient-Based Algorithm for On-Line Training of Recurrent Network Trajectories.” In: *Neural Computation* 2.4 (1990), pp. 490–501. DOI: 10.1162/neco.1990.2.4.490.

[186] J. Menick, E. Elsen, U. Evci, S. Osindero, K. Simonyan, and A. Graves. “A Practical Sparse Approximation for Real Time Recurrent Learning.” In: *arXiv* (2020). DOI: 10.48550/arXiv.2006.07232. eprint: 2006.07232v1.

[187] A. Kag and V. Saligrama. “Training Recurrent Neural Networks via Forward Propagation Through Time.” In: *Proceedings of the 38th International Conference on Machine Learning*. Vol. 139. PMLR, 2021, pp. 5189–5200. URL: https://proceedings.mlr.press/v139/kag21a.html.

[188] B. Yin, F. Corradi, and S. M. Bohté. “Accurate online training of dynamical spiking neural networks through Forward Propagation Through Time.” In: *Nature Machine Intelligence* 5.5 (2023), pp. 518–527. DOI: 10.1038/s42256-023-00650-4.

[189] M. Tsodyks, A. Uziel, and H. Markram. “Synchrony Generation in Recurrent Networks with Frequency-Dependent Synapses.” In: *The Journal of Neuroscience* 20.1 (2000). DOI: 10.1523/JNEUROSCI.20-01-j0003.2000.

[190] L. Hui and M. Belkin. “Evaluation of neural architectures trained with square loss vs cross-entropy in classification tasks.” In: *arXiv* (2021). DOI: 10.48550/arXiv.2006.07322. eprint: 2006.07322v5.

[191] E. O. Neftci, H. Mostafa, and F. Zenke. “Surrogate Gradient Learning in Spiking Neural Networks: Bringing the Power of Gradient-Based Optimization to Spiking Neural Networks.” In: *IEEE Signal Processing Magazine* 36.6 (2019), pp. 51–63. DOI: 10.1109/MSP.2019.2931595.

[192] S. B. Shrestha and G. Orchard. “SLAYER: Spike Layer Error Reassignment in Time.” In: *Advances in Neural Information Processing Systems (NeurIPS)* 31 (2018). URL: https://proceedings.neurips.cc/paper_files/paper/2018/hash/82f2b308c3b01637c607ce05f52a2fed-Abstract.html.

[193] F. Zenke and S. Ganguli. “SuperSpike: Supervised Learning in Multilayer Spiking Neural Networks.” In: *Neural Computation* 30.6 (2018), pp. 1514–1541. DOI: 10.1162/neco_a_01086.

[194] W. Fang, Z. Yu, Y. Chen, T. Huang, T. Masquelier, and Y. Tian. “Deep Residual Learning in Spiking Neural Networks.” In: *Advances in Neural Information Processing Systems (NeurIPS)* 34 (2021), pp. 21056–21069. URL: https://proceedings.neurips.cc/paper/2021/hash/afe434653a898da20044041262b3ac74-Abstract.html.

[195] G. Bellec, D. Kappel, W. Maass, and R. Legenstein. “Deep Rewiring: Training very sparse deep networks.” In: *International Conference on Learning Representations (ICLR)*. 2018. URL: https://openreview.net/forum?id=BJ_wN01C-.

[196] F. Zenke and T. P. Vogels. “The Remarkable Robustness of Surrogate Gradient Learning for Instilling Complex Function in Spiking Neural Networks.” In: *Neural Computation* 33.4 (2021), pp. 899–925. DOI: 10.1162/neco_a_01367.

[197] T. P. Lillicrap, D. Cownden, D. B. Tweed, and C. J. Akerman. “Random synaptic feedback weights support error backpropagation for deep learning.” In: *Nature Communications* 7.1 (2016), p. 13276. DOI: 10.1038/ncomms13276.

[198] A. Nøkland. “Direct Feedback Alignment Provides Learning in Deep Neural Networks.” In: *Advances in Neural Information Processing Systems (NeurIPS)* 29 (2016). URL: https://proceedings.neurips.cc/paper/2016/hash/d490d7b4576290fa60eb31b5fc917ad1-Abstract.html.

[199] A. Samadi, T. P. Lillicrap, and D. B. Tweed. “Deep Learning with Dynamic Spiking Neurons and Fixed Feedback Weights.” In: *Neural Computation* 29.3 (2017), pp. 578–602. DOI: 10.1162/NECO_a_00929.

[200] G. Martín-Sánchez, S. Bohté, and S. Otte. “A Taxonomy of Recurrent Learning Rules.” In: *International Conference on Artificial Neural Networks*. Springer, 2022, pp. 478–490. DOI: 10.1007/978-3-031-15919-0_40.

[201] G. Martín-Sánchez, S. Bohté, and S. Otte. “A Taxonomy of Recurrent Learning Rules.” In: *arXiv* (2022). DOI: 10.48550/arxiv.2207.11439. eprint: 2207.11439v2.

[202] J. A. Espinoza Valverde et al. *NEST*. Version 3.7. 2024. DOI: 10.5281/zenodo.10834751.

[203] F. Zenke and E. O. Neftci. “Brain-Inspired Learning on Neuromorphic Substrates.” In: *Proceedings of the IEEE* 109.5 (2021), pp. 935–950. DOI: 10.1109/JPROC.2020.3045625.

[204] T. C. Wunderlich and C. Pehle. “Event-based backpropagation can compute exact gradients for spiking neural networks.” In: *Scientific Reports* 11.1 (2021), p. 12829. DOI: 10.1038/s41598-021-91786-z.

[205] C. Pehle, L. Blessing, E. Arnold, E. Müller, and J. Schemmel. “Event-based Backpropagation for Analog Neuromorphic Hardware.” In: *arXiv* (2023). DOI: 10.48550/arXiv.2302.07141. eprint: 2302.07141v1.

[206] S. Billaudelle et al. “Versatile Emulation of Spiking Neural Networks on an Accelerated Neuromorphic Substrate.” In: *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2020, pp. 1–5. DOI: 10.1109/ISCAS45731.2020.9180741.

[207] G. Béna, T. Wunderlich, M. Akl, B. Vogginger, C. Mayr, and H. A. Gonzalez. “Event-based backpropagation on the neuromorphic platform SpiNNaker2.” In: *NeurIPS 2024 Workshop Machine Learning with new Compute Paradigms*. Curran Associates, Inc. URL: https://openreview.net/forum?id=hpfJ4Xkjzr.

[208] H. A. Gonzalez et al. “SpiNNaker2: A Large-Scale Neuromorphic System for Event-Based and Asynchronous Machine Learning.” In: *arXiv* (2024). DOI: 10.48550/arXiv.2401.04491. eprint: 2401.04491v1.

[209] J. H. Lee, T. Delbruck, and M. Pfeiffer. “Training Deep Spiking Neural Networks Using Backpropagation.” In: *Frontiers in Neuroscience* 10 (2016), p. 508. DOI: 10.3389/fnins.2016.00508.

[210] C. Tallec and Y. Ollivier. “Unbiased Online Recurrent Optimization.” In: *arXiv* (2017). DOI: 10.48550/arXiv.1702.05043. eprint: 1702.05043v3.

[211] C. Roth, I. Kanitscheider, and I. Fiete. “Kernel RNN Learning (KeRNL).” In: *International Conference on Learning Representations (ICLR)*. 2018. URL: https://openreview.net/forum?id=ryGfnoC5KQ.

[212] A. Mujika, F. Meier, and A. Steger. “Approximating Real-Time Recurrent Learning with Random Kronecker Factors.” In: *Advances in Neural Information Processing Systems (NeurIPS)* 31 (2018). URL: https://proceedings.neurips.cc/paper/2018/hash/dba132f6ab6a3e3d17a8d59e82105f4c-Abstract.html.

[213] J. M. Murray. “Local online learning in recurrent networks with random feedback.” In: *eLife* 8 (2019), e43299. DOI: 10.7554/eLife.43299.

[214] J. Kaiser, H. Mostafa, and E. Neftci. "Synaptic Plasticity Dynamics for Deep Continuous Local Learning (DECOLLE)." In: *Frontiers in Neuroscience* 14 (2020), p. 424. DOI: 10.3389/fnins.2020.00424.

[215] T. Bohnstingl, S. Woźniak, A. Pantazi, and E. Eleftheriou. "Online Spatio-Temporal Learning in Deep Neural Networks." In: *IEEE Transactions on Neural Networks and Learning Systems* 34.11 (2022), pp. 8894–8908. DOI: 10.1109/TNNLS.2022.3153985.

[216] J. Lee, S. Haghighatshoar, and A. Karbasi. "Exact Gradient Computation for Spiking Neural Networks." In: *OPT Optimization for Machine Learning (NeurIPS Workshop)*. Curran Associates, Inc., 2022. URL: https://openreview.net/forum?id=UC_gA3cyFNu.

[217] F. M. Quintana, F. Perez-Peña, P. L. Galindo, E. O. Neftci, E. Chicca, and L. Khacef. "ETLP: Event-based three-factor local plasticity for online learning with neuromorphic hardware." In: *Neuromorphic Computing and Engineering* 4.3 (2024), p. 034006. DOI: 10.1088/2634-4386/ad6733.

[218] W. Wei, M. Zhang, J. Zhang, A. Belatreche, J. Wu, Z. Xu, X. Qiu, H. Chen, Y. Yang, and H. Li. "Event-Driven Learning for Spiking Neural Networks." In: *arXiv* (2024). DOI: 10.48550/arXiv.2403.00270. eprint: 2403.00270v1.

[219] J. C. Knight and T. Nowotny. "Efficient GPU training of LSNNs using eProp." In: *Proceedings of the Annual Neuro-Inspired Computational Elements Conference (NICE)*. Association for Computing Machinery, 2022, pp. 8–10. DOI: 10.1145/3517343.3517346.

[220] J. C. Knight and T. Nowotny. "Easy and efficient spike-based Machine Learning with mlGeNN." In: *Proceedings of the Annual Neuro-Inspired Computational Elements Conference (NICE)*. Association for Computing Machinery, 2023, pp. 115–120. DOI: 10.1145/3584954.3585001.

[221] J. C. Knight, A. Komissarov, and T. Nowotny. "PyGeNN: A Python Library for GPU-Enhanced Neural Networks." In: *Frontiers in Neuroinformatics* 15 (2021), p. 659005. DOI: 10.3389/fninf.2021.659005.

[222] A. Perrett, S. Summerton, A. Gait, and O. Rhodes. "Online learning in SNNs with e-prop and Neuromorphic Hardware." In: *Proceedings of the Annual Neuro-Inspired Computational Elements Conference (NICE)*. Association for Computing Machinery, 2022, pp. 32–39. DOI: 10.1145/3517343.3517352.

[223] O. Rhodes, L. Peres, A. G. Rowley, A. Gait, L. A. Plana, C. Brenninkmeijer, and S. B. Furber. "Real-time cortical simulation on neuromorphic hardware." In: *Philosophical Transactions of the Royal Society A* 378.2164 (2019), p. 20190160. DOI: 10.1098/rsta.2019.0160.

[224] A. Rostami, B. Vogginger, Y. Yan, and C. G. Mayr. "E-prop on SpiNNaker 2: Exploring online learning in spiking RNNs on neuromorphic hardware." In: *Frontiers in Neuroscience* 16 (2022), p. 1018006. DOI: 10.3389/fnins.2022.1018006.

[225] C. Frenkel and G. Indiveri. "ReckOn: A 28nm Sub-mm2 Task-Agnostic Spiking Recurrent Neural Network Processor Enabling On-Chip Learning over Second-Long Timescales." In: *2022 IEEE International Solid-State Circuits Conference (ISSCC)*. Vol. 65. IEEE, 2022, pp. 1–3. DOI: 10.1109/ISSCC42614.2022.9731734.

[226] J. Fiser, P. Berkes, G. Orbán, and M. Lengyel. "Statistically optimal perception and learning: from behavior to neural representations." In: *Trends in Cognitive Sciences* 14.3 (2010), pp. 119–130. DOI: 10.1016/j.tics.2010.01.003.

[227] P. Berkes, G. Orbán, M. Lengyel, and J. Fiser. "Spontaneous cortical activity reveals hallmarks of an optimal internal model of the environment." In: *Science* 331.6013 (2011), pp. 83–87. DOI: 10.1126/science.1195870.

[228] G. Orbán, P. Berkes, J. Fiser, and M. Lengyel. "Neural variability and sampling-based probabilistic representations in the visual cortex." In: *Neuron* 92.2 (2016), pp. 530–543. DOI: 10.1016/j.neuron.2016.09.038.

[229] L. Buesing, J. Bill, B. Nessler, and W. Maass. "Neural dynamics as sampling: a model for stochastic computation in recurrent networks of spiking neurons." In: *PLOS Computational Biology* 7.11 (2011), e1002211. DOI: 10.1371/journal.pcbi.1002211.

[230] D. Pecevski, L. Buesing, and W. Maass. "Probabilistic inference in general graphical models through sampling in stochastic networks of spiking neurons." In: *PLOS Computational Biology* 7.12 (2011), e1002294. DOI: 10.1371/journal.pcbi.1002294.

[231] D. Probst, M. A. Petrovici, I. Bytschok, J. Bill, D. Pecevski, J. Schemmel, and K. Meier. "Probabilistic inference in discrete spaces can be implemented into networks of LIF neurons." In: *Frontiers in Computational Neuroscience* 9 (2015), p. 13. DOI: 10.3389/fncom.2015.00013.

[232] M. A. Petrovici, J. Bill, I. Bytschok, J. Schemmel, and K. Meier. "Stochastic inference with spiking neurons in the high-conductance state." In: *Physical Review E* 94.4 (2016), p. 042312. DOI: 10.1103/physreve.94.042312.

[233] D. Dold, I. Bytschok, A. F. Kungl, A. Baumbach, O. Breitwieser, W. Senn, J. Schemmel, K. Meier, and M. A. Petrovici. "Stochasticity from function—why the Bayesian brain may need no noise." In: *Neural Networks* 119 (2019), pp. 200–213. DOI: 10.1016/j.neunet.2019.08.002.

[234] N. Brunel. "Dynamics of sparsely connected networks of excitatory and inhibitory spiking neurons." In: *Journal of Computational Neuroscience* 8.3 (2000), pp. 183–208. DOI: 10.1023/A:1008925309027.

[235] E. M. Izhikevich. “Polychronization: computation with spikes.” In: *Neural Computation* 18.2 (2006), pp. 245–282. DOI: 10.1162/089976606775093882.

[236] A. Destexhe. “Self-sustained asynchronous irregular states and up–down states in thalamic, cortical and thalamocortical networks of nonlinear integrate-and-fire neurons.” In: *Journal of Computational Neuroscience* 27.3 (2009), pp. 493–506. DOI: 10.1007/s10827-009-0164-4.

[237] L. Muller and A. Destexhe. “Propagating waves in thalamus, cortex and the thalamocortical system: experiments and models.” In: *Journal of Physiology-Paris* 106.5-6 (2012), pp. 222–238. DOI: 10.1016/j.jphysparis.2012.06.005.

[238] G. Buzsáki and A. Draguhn. “Neuronal oscillations in cortical networks.” In: *Science* 304.5679 (2004), pp. 1926–1929. DOI: 10.1126/science.1099745.

[239] G. Buzsáki. *Rhythms of the Brain*. Oxford University Press, USA, 2006. DOI: 10.1093/acprof:oso/9780195301069.001.0001.

[240] E. Başar, C. Başar-Eroglu, S. Karakaş, and M. Schürmann. “Gamma, alpha, delta, and theta oscillations govern cognitive processes.” In: *International Journal of Psychophysiology* 39.2-3 (2001), pp. 241–248. DOI: 10.1016/s0167-8760(00)00145-8.

[241] X.-J. Wang. “Neurophysiological and computational principles of cortical rhythms in cognition.” In: *Physiological Reviews* 90.3 (2010), pp. 1195–1268. DOI: 10.1152/physrev.00035.2008.

[242] W. Klimesch. “Alpha-band oscillations, attention, and controlled access to stored information.” In: *Trends in Cognitive Sciences* 16.12 (2012), pp. 606–617. DOI: 10.1016/j.tics.2012.10.007.

[243] J. G. Klinzing, N. Niethard, and J. Born. “Mechanisms of systems memory consolidation during sleep.” In: *Nature Neuroscience* 22.10 (2019), pp. 1598–1610. DOI: 10.1038/s41593-019-0467-3.

[244] A. R. Adamantidis, C. Gutierrez Herrera, and T. C. Gent. “Oscillating circuitries in the sleeping brain.” In: *Nature Reviews Neuroscience* 20.12 (2019), pp. 746–762. DOI: 10.1038/s41583-019-0223-4.

[245] V. S. Sohal and M. E. Hasselmo. “Changes in GABAB modulation during a theta cycle may be analogous to the fall of temperature during annealing.” In: *Neural Computation* 10.4 (1998), pp. 869–882. DOI: 10.1162/089976698300017539.

[246] V. S. Sohal and M. E. Hasselmo. “GABAB modulation improves sequence disambiguation in computational models of hippocampal region CA3.” In: *Hippocampus* 8.2 (1998), pp. 171–193. DOI: 10/czggd8.

[247] P. Merolla, T. Ursell, and J. Arthur. “The thermodynamic temperature of a rhythmic spiking network.” In: *ArXiv* (2010). arXiv:1009.5473.

[248] C. Savin, P. Dayan, and M. Lengyel. "Optimal recall from bounded metaplastic synapses: predicting functional adaptations in hippocampal area CA3." In: *PLOS Computational Biology* 10.2 (2014), e1003489. DOI: 10.1371/journal.pcbi.1003489.

[249] L. Aitchison and M. Lengyel. "The Hamiltonian brain: efficient probabilistic inference with excitatory-inhibitory neural circuit dynamics." In: *PLOS Computational Biology* 12.12 (2016), e1005186. DOI: 10.1371/journal.pcbi.1005186.

[250] A. S. Ecker, P. Berens, G. A. Keliris, M. Bethge, N. K. Logothetis, and A. S. Tolias. "Decorrelated neuronal firing in cortical microcircuits." In: *Science* 327.5965 (2010), pp. 584–587. DOI: 10.1126/science.1179867.

[251] M. N. Shadlen and W. T. Newsome. "The variable discharge of cortical neurons: implications for connectivity, computation, and information coding." In: *Journal of Neuroscience* 18.10 (1998), pp. 3870–3896. DOI: 10.1523/jneurosci.18-10-03870.1998.

[252] M. A. Petrovici. *Form versus function: theory and models for neuronal substrates*. 1st ed. Springer International Publishing, 2016. DOI: 10.1007/978-3-319-39552-4.

[253] Y. Shu, A. Hasenstaub, and D. A. McCormick. "Turning on and off recurrent balanced cortical activity." In: *Nature* 423.6937 (2003), pp. 288–293. DOI: 10.1038/nature01616.

[254] B. Haider, A. Duque, A. R. Hasenstaub, and D. A. McCormick. "Neocortical network activity in vivo is generated through a dynamic balance of excitation and inhibition." In: *Journal of Neuroscience* 26.17 (2006), pp. 4535–4545. DOI: 10.1523/jneurosci.5297-05.2006.

[255] N. Dehghani, A. Peyrache, B. Telenczuk, M. Le Van Quyen, E. Halgren, S. S. Cash, N. G. Hatsopoulos, and A. Destexhe. "Dynamic balance of excitation and inhibition in human and monkey neocortex." In: *Scientific reports* 6.1 (2016), pp. 1–12. DOI: 10.1038/srep23176.

[256] E. Marinari and G. Parisi. "Simulated tempering a new Monte Carlo scheme." In: *Europhysics Letters EPL* 19.6 (1992), pp. 451–458. DOI: 10.1209/0295-5075/19/6/002.

[257] Y. LeCun, F. J. Huang, and L. Bottou. "Learning methods for generic object recognition with invariance to pose and lighting." In: *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. Vol. 2. IEEE. 2004, pp. II–104. DOI: 10.1109/CVPR.2004.1315150.

[258] Y. LeCun, C. Cortes, and C. J. C. Burges. "MNIST handwritten digit database." In: *ATT Labs* (2010). URL: http://yann.lecun.com/exdb/mnist/.

[259] R. Salakhutdinov. “Learning Deep Boltzmann Machines using adaptive MCMC.” In: *ICML 2010 - Proceedings, 27th International Conference on Machine Learning*. 2010, pp. 943–950. URL: https://icml.cc/Conferences/2010/papers/441.pdf.

[260] L. Leng, R. Martel, O. Breitwieser, I. Bytschok, W. Senn, J. Schemmel, K. Meier, and M. A. Petrovici. “Spiking neurons with short-term synaptic plasticity form superior generative networks.” In: *Scientific Reports* 8.1 (2018), p. 10651. DOI: 10.1038/s41598-018-28999-2.

[261] O. Breuleux, Y. Bengio, and P. Vincent. *Unlearning for better mixing*. Tech. rep. Universite de Montreal/DIRO, 2010. URL: http://www.iro.umontreal.ca/~lisa/publications/unlearning_for_better_mixing.pdf.

[262] A. Borji. “Pros and cons of gan evaluation measures.” In: *Computer Vision and Image Understanding* 179 (2019), pp. 41–65. DOI: 10.1016/j.cviu.2018.10.009.

[263] O. Breitwieser, A. Baumbach, A. Korcsak-Gorzo, J. Klähn, M. Brixner, and M. Petrovici. “sbs: Spike-based Sampling (v1.8.2).” Version v1.8.2. In: *Zenodo* (2020). DOI: 10.5281/zenodo.3686015.

[264] A. Peyser et al. “NEST 2.14.0.” In: *Zenodo* (2017). DOI: 10.5281/zenodo.882971.

[265] S. Habenschuss, Z. Jonke, and W. Maass. “Stochastic computations in cortical microcircuit models.” In: *PLOS Computational Biology* 9.11 (2013). DOI: 10.1371/journal.pcbi.1003311.

[266] M. Lundqvist, A. M. Bastos, and E. K. Miller. “Preservation and Changes in Oscillatory Dynamics across the Cortical Hierarchy.” In: *Journal of Cognitive Neuroscience* 32.10 (2020), pp. 2024–2035. DOI: 10/gg8q3f.

[267] G. Hennequin, L. Aitchison, and M. Lengyel. “Fast Sampling-Based Inference in Balanced Neuronal Networks.” In: *Advances in Neural Information Processing Systems*. Vol. 27. 2014, pp. 2240–2248. URL: https://papers.neurips.cc/paper/2014/file/a7d8ae4569120b5bec12e7b6e9648b86-Paper.pdf.

[268] C. Koch, M. Rapp, and I. Segev. “A brief history of time (constants).” In: *Cerebral cortex* 6.2 (1996), pp. 93–101. DOI: 10.1093/cercor/6.2.93.

[269] M. Häusser and A. Roth. “Estimating the time course of the excitatory synaptic conductance in neocortical pyramidal cells using a novel voltage jump method.” In: *Journal of Neuroscience* 17.20 (1997), pp. 7606–7625. DOI: 10.1523/jneurosci.17-20-07606.1997.

[270] M. Bartos, I. Vida, M. Frotscher, J. R. Geiger, and P. Jonas. “Rapid signaling at inhibitory synapses in a dentate gyrus interneuron network.” In: *Journal of Neuroscience* 21.8 (2001), pp. 2687–2698. DOI: 10.1523/JNEUROSCI.21-08-02687.2001.

[271] D. A. McCormick, B. W. Connors, J. W. Lighthall, and D. A. Prince. "Comparative electrophysiology of pyramidal and sparsely spiny stellate neurons of the neocortex." In: *Journal of neurophysiology* 54.4 (1985), pp. 782–806. DOI: 10.1152/jn.1985.54.4.782.

[272] M. Steriade, I. Timofeev, N. Dürmüller, and F. Grenier. "Dynamic properties of corticothalamic neurons and local cortical interneurons generating fast rhythmic (30–40 Hz) spike bursts." In: *Journal of Neurophysiology* 79.1 (1998), pp. 483–490. DOI: 10.1152/jn.1998.79.1.483.

[273] R. Echeveste, L. Aitchison, G. Hennequin, and M. Lengyel. "Cortical-like dynamics in recurrent circuits optimized for sampling-based probabilistic inference." In: *Nature Neuroscience* 23.9 (2020), pp. 1138–1149. DOI: 10.1038/s41593-020-0671-1.

[274] S. Diekelmann and J. Born. "Slow-wave sleep takes the leading role in memory reorganization." In: *Nature Reviews Neuroscience* 11.3 (2010), pp. 218–218. DOI: 10.1038/nrn2762-c2.

[275] A. Marzecová, A. Schettino, A. Widmann, I. SanMiguel, S. A. Kotz, and E. Schröger. "Attentional gain is modulated by probabilistic feature expectations in a spatial cueing task: ERP evidence." In: *Scientific Reports* 8.1 (2018), pp. 1–14. DOI: 10.1038/s41598-017-18347-1.

[276] J. H. Reynolds and L. Chelazzi. "Attentional modulation of visual processing." In: *Annual Review of Neuroscience* 27 (2004), pp. 611–647. DOI: 10.1146/annurev.neuro.26.041002.131039.

[277] M. E. Larkum, W. Senn, and H. Lüscher. "Top-down Dendritic Input Increases the Gain of Layer 5 Pyramidal Neurons." In: *Cerebral Cortex* 14.10 (2004), pp. 1059–1070. DOI: 10.1093/cercor/bhh065.

[278] F. Stella, P. Baracskay, J. O'Neill, and J. Csicsvari. "Hippocampal Reactivation of Random Trajectories Resembling Brownian Diffusion." In: *Neuron* 102.2 (2019), 450–461.e7. DOI: 10.1016/j.neuron.2019.01.052.

[279] M. Katkov, S. Romani, and M. Tsodyks. "Memory Retrieval from First Principles." In: *Neuron* 94.5 (2017), pp. 1027–1032. DOI: 10.1016/j.neuron.2017.03.048.

[280] C. Lustenberger, M. R. Boyle, A. A. Foulser, J. M. Mellin, and F. Fröhlich. "Functional role of frontal alpha oscillations in creativity." In: *Cortex* 67 (2015), pp. 74–82. DOI: 10.1016/j.cortex.2015.03.012.

[281] J. D. Pettigrew. "Searching for the Switch: Neural Bases for Perceptual Rivalry Alternations." In: *Brain and Mind* 2 (2001), pp. 85–118. DOI: 10.1023/A:1017929617197.

[282] R. J. Compton, D. Gearinger, and H. Wild. "The wandering mind oscillates: EEG alpha power is enhanced during moments of mind-wandering." In: *Cognitive, Affective and Behavioral Neuroscience* 19.5 (2019), pp. 1184–1191. DOI: 10.3758/s13415-019-00745-9.

[283] L. Stålesen Ramfjord, E. Hertenstein, K. Fehér, C. Mikutta, C. L. Schneider, C. Nissen, and J. G. Maier. “Local sleep and wakefulness—the concept and its potential for the understanding and treatment of insomnia disorder.” In: *Somnologie* 24.2 (2020), pp. 116–120. DOI: 10.1007/s11818-020-00245-w.

[284] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. “Optimization by simulated annealing.” In: *Science (New York, N.Y.)* 220.4598 (1983), pp. 671–80. DOI: 10.1126/science.220.4598.671.

[285] R. M. Neal. “Sampling from multimodal distributions using tempered transitions.” In: *Statistics and computing* 6.4 (1996), pp. 353–366. DOI: 10.1007/bf00143556.

[286] R. Salakhutdinov. “Learning in Markov Random Fields Using Tempered Transitions.” In: *Proceedings of the 22nd International Conference on Neural Information Processing Systems*. 2009, pp. 1598–1606. URL: https://proceedings.neurips.cc/paper/2009/hash/b7ee6f5f9aa5cd17ca1aea43ce848496-Abstract.html.

[287] C. Capone, E. Pastorelli, B. Golosio, and P. S. Paolucci. “Sleep-like slow oscillations improve visual classification through synaptic homeostasis and memory association in a thalamo-cortical model.” In: *Scientific reports* 9.1 (2019), pp. 1–11. DOI: 10.1038/s41598-019-45525-0.

[288] I. Grothe, S. D. Neitzel, S. Mandon, and A. K. Kreiter. “Switching neuronal inputs by differential modulations of gamma-band phase-coherence.” In: *Journal of Neuroscience* 32.46 (2012), pp. 16172–16180. DOI: 10.1523/jneurosci.0890-12.2012.

[289] J. O’Keefe and M. L. Recce. “Phase relationship between hippocampal place units and the EEG theta rhythm.” In: *Hippocampus* 3.3 (1993), pp. 317–330. DOI: 10.1002/hipo.450030307.

[290] T. J. Buschman and E. K. Miller. “Shifting the spotlight of attention: evidence for discrete computations in cognition.” In: *Frontiers in human neuroscience* 4 (2010), p. 194. DOI: 10.3389/fnhum.2010.00194.

[291] K. Jezek, E. J. Henriksen, A. Treves, E. I. Moser, and M.-B. Moser. “Theta-paced flickering between place-cell maps in the hippocampus.” In: *Nature* 478.7368 (2011), pp. 246–249. DOI: 10.1002/hipo.22743.

[292] S. Mark, S. Romani, K. Jezek, and M. Tsodyks. “Theta-paced flickering between place-cell maps in the hippocampus: A model based on short-term synaptic plasticity.” In: *Hippocampus* 27.9 (2017), pp. 959–970. DOI: 10.1002/hipo.22743.

[293] B. E. Pfeiffer and D. J. Foster. “Hippocampal place-cell sequences depict future paths to remembered goals.” In: *Nature* 497.7447 (2013), pp. 74–79. DOI: 10.1038/nature12112.

[294] T. A. Engel, N. A. Steinmetz, M. A. Gieselmann, A. Thiele, T. Moore, and K. Boahen. “Selective modulation of cortical state during spatial attention.” In: *Science* 354.6316 (2016), pp. 1140–1144. DOI: 10.1126/science.aag1420.

[295] M. Al-Shedivat, R. Naous, G. Cauwenberghs, and K. N. Salama. “Memristors empower spiking neurons with stochasticity.” In: *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 5.2 (2015), pp. 242–253. DOI: 10.1109/jetcas.2015.2435512.

[296] A. Sengupta, P. Panda, P. Wijesinghe, Y. Kim, and K. Roy. “Magnetic tunnel junction mimics stochastic cortical spiking neurons.” In: *Scientific Reports* 6.1 (2016), pp. 1–8. DOI: 10.1038/srep30039.

[297] A. F. Kungl, S. Schmitt, J. Klähn, P. Müller, A. Baumbach, D. Dold, A. Kugele, E. Müller, C. Koke, M. Kleider, et al. “Accelerated physical emulation of Bayesian inference in spiking neural networks.” In: *Frontiers in Neuroscience* 13 (2019), p. 1201. DOI: 10.3389/fnins.2019.01201.

[298] Z. Jonke, S. Habenschuss, and W. Maass. “Solving constraint satisfaction problems with networks of spiking neurons.” In: *Frontiers in neuroscience* 10 (2016), p. 118. DOI: 10.3389/fnins.2016.00118.

[299] J. Binas, G. Indiveri, and M. Pfeiffer. “Spiking analog VLSI neuron assemblies as constraint satisfaction problem solvers.” In: *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE. IEEE, 2016, pp. 2094–2097. DOI: 10.1109/ISCAS.2016.7538992.

[300] G. A. Fonseca Guerra and S. B. Furber. “Using stochastic spiking neural networks on spinnaker to solve constraint satisfaction problems.” In: *Frontiers in Neuroscience* 11 (2017), p. 714. DOI: 10.3389/fnins.2017.00714.

[301] B. A. Richards, T. P. Lillicrap, P. Beaudoin, Y. Bengio, R. Bogacz, A. Christensen, C. Clopath, R. P. Costa, A. de Berker, S. Ganguli, et al. “A deep learning framework for neuroscience.” In: *Nature Neuroscience* 22.11 (2019), pp. 1761–1770. DOI: 10.1038/s41593-019-0520-2.

ACKNOWLEDGMENTS

I want to thank Prof. Markus Diesmann and Prof. Andreas Offenhäusser for taking the time to examine this thesis.

I am profoundly grateful to several people who supported and believed in me throughout this journey: Prof. Markus Diesmann, for his wise guidance and positive attitude, which always lifted my spirits; Prof. Sacha J. van Albada, who has been a role model to me and whose advice has been invaluable; and Dr. Mihai A. Petrovici, whose attention to detail and clarity taught me so much about good science.

A big thanks to my collaborators, who made the work both productive and enjoyable and from whom I learned a lot. In particular, for many hours of pair programming fun: Alexander van Meegen (in a project preceding a study of this thesis), Andreas Baumbach, Charl Linssen, Jesús A. Espinoza Valverde, and Jonas Stapmanns. Special thanks to Prof. Hans Ekkehard Plesser, whose thorough reviews helped me better understand how to write clean and effective code; Dennis Terhorst for generously sharing his extensive technical expertise; and Tom Tetzlaff for reviewing part of this thesis.

I would like to express my gratitude to Prof. Markus Diesmann and Prof. Sonja Grün for creating such a welcoming and brain-friendly environment at the institute. It has been an ideal place for the work on this thesis. I am also thankful for all my colleagues (too many to name here) who made it such a lively place. From exhilarating scientific discussions, lunch breaks, social events, to retreats and trips — I will carry many fond memories of this time.

Outside work, I am thankful to those people who helped me keep my balance: To Philipp, for countless fun hours on the tennis court. To Anna, Christian, Hendrik, Philipp, Sarah, and Sven, for all the good times, whether through pub quizzes on video calls, electro swing parties, or other gatherings. Special thanks to Anna — I always treasure our time together. To my dear Honeybadgers — Andrej, Boris, David, Jan Philipp, and Michael — for the laughter and warmth throughout our video calls and meetups. Michael and Andrej, in particular, for their generosity and all the joyous times we spent together. To Regi, for all the kind support and for teaching me how humor can ease the most challenging moments. To my brother Peter, for cheering me up with his humor and wisdom, and to my sister Kati, for all the encouragement and for inspiring me in so many ways. Finally, my heartfelt thanks go to Mama and Tata for their boundless love and support.

With any of these pieces missing this work would not have been possible and by far not as enjoyable.

Thank you.

FUNDING

This thesis contributed to the following projects:

- European Union's Horizon 2020 Framework Programme for Research and Innovation under Specific Grant Agreement No. 720270 (Human Brain Project SGA1), No. 785907 (Human Brain Project SGA2), and No. 945539 (Human Brain Project SGA3),
- Helmholtz Association Initiative and Networking Fund under project number SO-092 (Advanced Computing Architectures, ACA),
- Joint Lab "Supercomputing and Modeling for the Human Brain",
- HiRSE_PS, the Helmholtz Platform for Research Software Engineering — Preparatory Study, an innovation pool project of the Helmholtz Association,
- NeuroSys as part of the initiative "Clusters4Future" funded by the Federal Ministry of Education and Research BMBF (03ZU1106CB),
- Jülich-Aachen Research Alliance (JARA),
- computing time granted by the JARA Vergabegremium and provided on the JARA Partition of the supercomputer JURECA at Forschungszentrum Jülich (computation grant JINB33),
- state of Baden-Württemberg through bwHPC and the German Research Foundation (DFG) through grant no INST 39/963–1 FUGG (bwForCluster NEMO).

The funders had no role in the design of the study, collection, analysis, or interpretation of data, the decision to publish, or the preparation of this thesis.

COLOPHON

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede and Ivo Pletikosić. The style was inspired by Robert Bringhurst's seminal book on typography *"The Elements of Typographic Style"*. `classicthesis` is available for both LaTeX and LyX:

`https://bitbucket.org/amiede/classicthesis/`

Information
Band / Volume 119
ISBN 978-3-95806-876-6