Stochastic Computations and Learning in Networks of Spiking Neurons: Simulation framework, Analysis and Theory

by

Zeno Jonke

DISSERTATION

submitted for the degree of

Doctor Technicae

Institute for Theoretical Computer Science
Graz University of Technology

Thesis Advisor:
O.Univ.-Prof. Dipl.-Ing. Dr.rer.nat. Wolfgang Maass

defended on January 8th, 2014

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommene Stellen als solche kenntlich gemacht habe.

Statutory Declaration

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

Graz, December 2013 ....

(signature)

Abstract

Noise in the brain – friend or foe? Noise is omnipresent at almost every temporal and spatial scale in the brain, introducing seemingly undesired variability at virtually every step of neural computation. In this thesis, rather than studying how noise could be suppressed and its impact on circuit function could be minimized, the opposite route is taken: noise is considered as a potential computational resource.

As a basis for investigating various aspects of noise, a software framework for the simulation and analysis of stochastic networks of spiking neurons is developed. Using this framework, three key research questions are addressed through a combined approach of simulation, computational analysis and theory:

First, based on the observation that neurons are inherently stochastic, it is investigated how probability distributions could be stored in the brain through stochastic networks of neurons that are able to generate samples from those stored distributions. Computation times for typical probabilistic inference tasks on these internally stored distributions are investigated through computer simulations and the convergence speed of a standard cortical data-based microcircuit model is examined, showing that convergence occurs quite fast in the range of 100ms of biological time, which is sufficient for common probabilistic inference tasks.

Second, it is investigated how these internally stored distributions of a network with noise could be programmed to achieve desired computations. To address this question, generic design principles for constructing stochastic spiking networks, which autonomously generate heuristic solutions to difficult computational problems, are developed. By encoding in a stochastic network a distribution which assigns highest probability to correct solutions, and implementing specific problem constraints by introducing controlled interactions among neurons, it is shown that one can solve many different problems from the class of Constraint Satisfaction Problems. The resulting computational capabilities of networks of spiking neurons to solve well-known NP-hard problems such as 3-SAT and the Traveling Salesman Problem are demonstrated in computer simulations.

Third, it is studied how stochastic microcircuits in the cortex can learn to isolate input component features and autonomously form a suitable representation of high-dimensional spike input streams. A biologically motivated stochastic winner-take-all (WTA) circuit motif, the so called Sparse WTA microcircuit motif with soft lateral inhibition, is proposed, where several neurons can spike simultaneously to explain the current input. By theoretical analysis and simulations, the circuit is shown to acquire through spike-timing dependent plasticity (STDP) the capability to extract and represent multiple salient features from complex inputs and to become temporally selective at the same time. This suggests that STDP installs in ubiquitous microcircuit motifs with noise a powerful operation that could explain some of the remarkable computational capabilities of a generic cortical column.

Altogether, the results of this thesis represent important contributions to the understanding of basic principles of stochastic computations and learning in networks of spiking neurons.

Zusammenfassung

Rauschquellen im Gehirn – Freund oder Feind? Rauschen ist ein allgegenwärtiges Phänomen, welches auf fast jeder zeitlichen und räumlichen Skala im Gehirn beobachtet werden kann, und scheinbar unerwünschte Variabilität in praktisch jeder Stufe neuronaler Berechnung erzeugt. Anstatt zu untersuchen wie Rauschen unterdrückt und seine Auswirkungen auf die Funktion eines Netzwerks minimiert werden könnten, wird in dieser Arbeit der entgegengesetzte Weg beschritten: Rauschen wird als potentielle Ressource für Berechnungen in neuronalen Netzwerken betrachtet.

Als Grundlage für die Untersuchung verschiedener Aspekte wird ein Software-Framework für die Simulation und Analyse von stochastischen Netzwerken von Spiking Neuronen entwickelt. Darauf basierend werden drei wichtige Forschungsfragen durch einen kombinierten Ansatz aus Simulation, Computeranalyse und Theorie behandelt.

Zuerst wird untersucht, wie Wahrscheinlichkeitsverteilungen im Gehirn durch stochastische Netzwerke von Neuronen gespeichert werden können, in einer Art und Weise die es Netzwerken ermöglicht, effizient Stichproben (samples) aus gespeicherten Verteilungen zu erzeugen. Berechnungszeiten für typische Operationen auf diesen intern gespeicherten Verteilungen werden durch Computersimulationen untersucht, und die Konvergenzgeschwindigkeit eines datenbasierten Standardmodells einer kortikalen Kolumne analysiert. Es wird gezeigt, dass die Konvergenz zu einer stationären Verteilung innerhalb von wenigen 100 ms biologischer Zeit erfolgt, sodass typische probabilistische Inferenzoperationen auf sehr kurzen Zeitskalen ausgeführt werden können.

Zweitens wird untersucht, wie diese intern gespeicherten Verteilungen eines Netzwerks mit Rauschen programmiert werden könnten, um gewünschte Berechnungen zu ermöglichen. Um diese Frage zu beantworten, werden generische Design-Prinzipien für die Konstruktion stochastischer Spiking Netze, welche autonom heuristische Lösungen für schwierige Rechenprobleme erzeugen können, entwickelt. Die Programmierung einer Verteilung erfolgt so, dass korrekten Lösungen eines Problems die höchste Wahrscheinlichkeit zugeordnet wird, und konkrete Randbedingungen eines Problems durch gezielte Interaktionen zwischen Neuronen implementiert werden. Es wird gezeigt, dass auf diese Weise viele Probleme aus der Klasse der Constraint-Satisfaction-Probleme mit Neuronalen Netzen lösbar sind. Die daraus resultierenden rechnerischen Fähigkeiten von Spiking Neuronalen Netzen hinsichtlich des Lösens bekannter NP-schwerer Probleme wie 3-SAT oder dem Traveling Salesman Problem werden in Computersimulationen gezeigt.

Drittens wird die Frage gestellt, wie stochastische Netze im Gehirn lernen können, verschiedene Komponenten eines Eingangssignals zu isolieren und autonom eine geeignete Darstellung von hochdimensionalen Spike Trains zu bilden. Ein biologisch motiviertes stochastisches Netzwerkmotiv wird vorgeschlagen: das so genannte Sparse Winner-Take-All (SWTA) Motiv mit schwacher lateraler Inhibition, in dem mehrere Neuronen gleichzeitig feuern können, um gemeinsam ein Eingangssignal zu erklären. Mittels theoretischer Analyse und Netzwerksimulationen wird gezeigt, dass solche Netze durch Spike-Timing Dependent Plasticity (STDP) die Fähigkeit erwerben, mehrere Aspekte eines hochdimensionalen Eingangssignals zugleich zu extrahieren und zu repräsentieren, sowie zeitliche Selektivität für verschiedene Phasen eines Eingangsmusters zu entwickeln. Diese Ergebnisse deuten darauf hin, dass STDP in einem allgegenwärtigen Netzwerkmotiv des Gehirns eine leistungsfähige Rechenoperation installiert, die einige der bemerkenswerten Rechenfähigkeiten einer generischen kortikalen Kolumne erklären konnte.

Insgesamt stellen die Ergebnisse dieser Arbeit wichtige Beiträge zum Verständnis stochastischer Berechnungen und Lernvorgänge in Spiking Neuronalen Netzwerken dar.

Acknowledgments

I am thankful to the many people who made my PhD studies an unforgettable experience. First of all I would like to thank Prof. Wolfgang Maass, my supervisor, for giving me the opportunity to work on very exciting and challenging topics, but also for all his useful ideas, enthusiasm and support throughout my PhD studies. Thanks to him I had the chance to work in various exciting research programs of the European Union, including SECO, Brain-i-Nets, BrainScaleS and the Human Brain Project, whose financial support made my PhD work possible. I am also very grateful to Prof. Robert Legenstein for the fruitful collaboration, for being a member of my PhD committee, and for everything I learned from him, including that going to a conference with a professor can be a lot of fun. I would also like to thank Prof. Helmut Pockberger for kindly accepting to be the external referee of my work.

This thesis would not have been possible without Stefan Habenschuss, with whom I had a fruitful collaboration during my whole PhD studies, and Dejan Pecevski, who developed PCSIM and NEVESIM, the backbones for most of the simulations of this thesis. With them, and together with Johannes Bill and David Kappel, I had countless useful, interesting and insightful discussions in the office, and maybe even more inspiring and important discussions outside the office.

My life in Graz and at the institute would not have been the same without the ”IGI family”, especially Daniela Potzinger, Regina Heidinger and Oliver Friedl who did an amazing job making me feel at home in Graz (and in the office). I am thankful to all the current and previous members of the ”IGI family” who have contributed immensely to my personal and professional time in Graz, and who I had the pleasure to work with and share everyday life at and outside the institute, especially Stefan Habenschuss, Dejan Pecevski, Johannes Bill, David Kappel, Elmar Rückert, Stefan Häusler, Bernhard Nessler, Octave Etard, Martin Ratajczak and many others.

I am grateful to all my friends who bridged over the distance from Zagreb to Graz with their frequent visits and made me feel as if I still live in my hometown, Zagreb. Thank you for enriching my life, for the nice memories and for reminding me of other sides of life. I especially thank those whose ambition and inspiring projects (such as book writing) kept me motivated to finish my thesis.

Lastly, I would like to thank my family who always believed in me. I especially thank my uncle Ivo, for all his support throughout my education, my brother Alen who always supported and encouraged me, and finally and most importantly my parents, Marija and Ivan, for their endless love and encouragement. You made all this possible. Thank you!

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Quantum mechanics at its fundamental level states that nature is inherently random and therefore unpredictable. Due to random processes governing nature, noise is an inevitable phenomenon influencing everything around us: from the interaction of physical particles to the decisions we make. The brain, arguably the most sophisticated product of evolution, enable us not only to handle such inherently noisy environment with ease, but it also enables us to solve problems, make art, experience emotions and make decisions, although not always the right ones. Ironically, to deal with noise and uncertainty this magnificent and most complex information processing system known in the universe is made of billions of noisy and unreliable components - neurons (Faisal et al., 2008). Remarkably, neurons self-organize in networks and use spikes (action potentials) for asynchronous communication through trillions of equally noisy and unreliable synapses in order to perform complex information processing functions that lie at the heart of human intelligence. Still, a theory that explains how exactly neurons are connected and how they self-organize in order for these complex functions to arise is missing.

Learning a new song, reminding yourself of all the nice places you have visited last summer, or simply planning the shortest and the safest path while crossing the street, are examples of simple and typical situations we deal with during our lives. Although people take those actions of learning, remembering and inference for granted, the basic understanding of how the brain actually stores, recalls or processes information, and how it combines it in real time with sensory input in a manner that is consistent with the noisy nature of its computing elements is not known. One hypothesis supported by numerous recent studies is that many of these complex brain functions involve probabilistic inference through sampling (Griffiths and Tenenbaum, 2006; Vul et al., 2009; Gershman et al., 2009), which suggests that networks of neurons process information in the form of probability distributions. This is an interesting perspective as it does not view noise as a source of undesired variability in neural responses but rather as a potential resource for stochastic computations in a network of neurons that stores probability distributions. On the other hand it raises the question how complex distributions could be represented and stored in large and diverse networks of neurons such that they could be later used for knowledge extraction and inference.

Typically a large enough brain network, such as a cortical microcircuit, which potentially gives rise to some useful computational function contains a large number of neurons of different types connected in some functional units. These neurons might form some small motifs and the network might be seen as composed of many different frequently occurring small network motifs. For example it is known that winner-take-all (WTA) circuits constitute a ubiquitous motif of cortical microcircuits (Douglas and Martin, 2004). If one tries to understand such networks in terms of the probability distributions they embody, interesting questions arise: How does a specific network motif shape the complex distribution embodied by the large network? Is there some principle for constructing networks with certain functional properties such as problem solving capabilities? Answering these questions would obviously significantly advance our understanding of how the brain can store complex knowledge and perform inference with it.

Probably the most fascinating and important property of the brain is its plasticity (Fregnac, 2003), due to which we can learn or acquire new knowledge and skills throughout our life, but through which we can also forget. For example, information memorized in a short time, if of crucial importance, could be remembered for a life time, while some existing knowledge if not repeated on a regular basis could easily get partially or completely lost. When learning a new skill or acquiring some new piece of information, networks in the brain are self-organizing in order to maintain useful representations and computational functions. Consequently, during learning networks are constantly adapting and, as a consequence, internally stored distributions are changing. A fundamental question is what the mechanisms governing these network changes are. More precisely, what are the mechanisms that are activated during acquisition of new skills to shape the internally stored distributions such that useful new computational operations can emerge?

The brain computes in a massively parallel and asynchronous manner, in stark contrast to a traditional computer architecture. Furthermore, it does so without a global clock and memory and by using only "simple" computational units. The power of such parallel cognitive architecture is unrivaled by the most powerful supercomputer. Therefore, from the perspective of Computer Science, providing answers to the above questions in the form of general principles for understanding computations in such a highly parallel architecture could open the door to very efficient, clever and fast implementations of computational algorithms for various problems that could bring human level intelligence or general problem solving capabilities to desktop computers.

In this thesis, with the above goals in mind, principles underlying the computational function of networks of spiking neurons are investigated from the perspective that noise is a potential computational resource, rather than a nuisance. The key contributions made by this thesis are: a) A new software framework for the simulation and analysis of stochastic networks of spiking neurons that was developed as a foundation for addressing various research questions in the scope of the thesis. b) An analysis of stochastic computations in a standard model of a cortical column. This analysis revealed, among other important results, that a cortical column model is capable of storing probability distributions in a manner that enables very fast probabilistic inference operations on these distributions. c) A demonstration of how probability distributions can be systematically manipulated in networks of spiking neurons with noise to endow them with powerful problem solving capabilities. d) A new theory for how representations could self-organize in stereotypical WTA networks of spiking neurons with soft lateral inhibition through synaptic plasticity. Taken together, these results constitute important contributions to the understanding of general principles underlying the computational functions of networks of stochastic spiking neurons.

1.1 Organization of the Thesis

The results presented in all but last chapter of this thesis are based on three paper manuscripts written during my PhD studies in collaboration with my supervisor prof. Wolfgang Maass, prof. Robert Legenstein and my colleague Stefan Habenschuss. In each of these chapters only the main findings are presented, while additional supporting information, such as derivations and simulation details, can be found in the Appendix. The last chapter contains description of the simulation framework developed and used within the scope of this thesis, where the section on NEVESIM is based on the paper manuscript written in collaboration with my colleagues Dejan Pecevski and David Kappel.

In Chapter 2, based on the observation that neurons are inherently stochastic, it is investigated how probability distributions could be stored in the brain through stochastic networks of neurons that are able to generate samples from those stored distributions. It is proposed that the stochastic dynamics of biological spiking networks with virtually arbitrarily complex dynamics (including short term plasticity, nonlinear dendritic computations, etc.) can be viewed as Markov Chain Monte Carlo sampling. These computation times were assessed by adapting the Gelman-Rubin diagnostics, a method originally developed to assess convergence speed of Markov Chains in statistics and Machine Learning, for monitoring convergence in cortical microcircuits. An analysis of the convergence speed of a standard cortical data-based microcircuit model showed that convergence occurs quite fast in the range of 100ms of biological time, which is sufficient for common probabilistic inference tasks. Finally, as a first step towards understanding how internally stored distribution of a network could be programmed to achieve desired computations, design principles for constructing networks which autonomously generate heuristic solutions to a Sudoku puzzle were designed and tested through simulations.

In Chapter 3 it is investigated how these internally stored distributions of a network with noise could be programmed to achieve desired computations. To address this question, generic design principles for constructing stochastic spiking networks, which autonomously generate heuristic solutions to difficult computational problems, are developed. By encoding in a stochastic network a probability distribution that assigns highest probability to correct solutions, and implementing specific problem constraints by introducing controlled interactions among neurons, it is shown that one can solve various problems from the class of Constraint Satisfaction Problems. The underlying theoretical principle is that one can construct a network such that in the stationary distribution of network states $p(\mathbf{x})$, the network visits states $\mathbf{x}$ in proportion to $\exp(-C * \#\text{violated constraints})$ with some positive constant C. Hence, network states $\mathbf{x}$ which correspond to correct solutions occur particularly often. The resulting computational capabilities of networks of spiking neurons to solve well-known NP-complete problems such as 3SAT and the Traveling Salesman Problem are demonstrated in computer simulations.

Chapter 4 addresses the question how stochastic microcircuits in the cortex can learn to isolate input component features and autonomously form a suitable representation of high-dimensional spike input streams. A biologically motivated stochastic WTA circuit motif, the so called Sparse WTA microcircuit motif, with soft lateral inhibition where several neurons can spike simultaneously, is proposed. By theoretical analysis and network simulations, the circuit motif is shown to acquire through spike-timing dependent plasticity (STDP) the capability to simultaneously extract and represent multiple salient features from complex inputs and to become temporally selective at the same time. This suggests that STDP installs in ubiquitous microcircuit motifs with noise a very powerful computational operation that could explain some of the remarkable computational capabilities of a generic cortical column.

Chapter 5 provides an overview of a software framework for the simulation and analysis of stochastic networks of spiking neurons which consists of software tools I developed or extended for the purpose of the thesis. ZLIB, a library for parallelization and optimization, and CSP2SNN, the framework for automatic porting of Constraint Satisfaction Problems to Spiking Neural Networks, were developed by myself in order to facilitate analysis and large scale simulations. PCSIM, a time step based simulator developed by Dejan Pecevski and Thomas Natschlager, and NEVESIM, an event based simulator developed by Dejan Pecevski, were used as a backbone for all simulations and were adapted in order to support specific simulation requirements of the thesis.

Chapter 2

Stochastic Computations in Cortical Microcircuit Models

Contents

Experimental data from neuroscience suggest that a substantial amount of knowledge is stored in the brain in the form of probability distributions over network states and trajectories of network states. We provide a theoretical foundation for this hypothesis by showing that even very detailed models for cortical microcircuits, with data-based diverse nonlinear neurons and synapses, have a stationary distribution of network states and trajectories of network states to which they converge exponentially fast from any initial state. We demonstrate that this convergence holds in spite of the non-reversibility of the stochastic dynamics of cortical microcircuits. We further show that, in the presence of background network oscillations, separate stationary distributions emerge for different phases of the oscillation, in accordance with experimentally reported phase-specific codes. We complement these theoretical results by computer simulations that investigate resulting computation times for typical probabilistic inference tasks on these internally stored distributions, such as marginalization or marginal maximum-a-posteriori estimation. Furthermore, we show that the inherent stochastic dynamics of generic cortical microcircuits enables them to quickly generate approximate solutions to difficult constraint satisfaction

problems, where stored knowledge and current inputs jointly constrain possible solutions. This provides a powerful new computing paradigm for networks of spiking neurons, that also throws new light on the way how networks of neurons in the brain could carry out complex computational tasks such as prediction, imagination, memory recall and problem solving.

2.1 Introduction

The question whether brain computations are inherently deterministic or inherently stochastic is obviously of fundamental importance. Numerous experimental data highlight inherently stochastic aspects of neurons, synapses and networks of neurons on virtually all spatial and temporal scales that have been examined (Allen and Stevens, 1994; Faisal et al., 2008; Borst, 2010; Yarom and Hounsgaard, 2011; Clarke, 2012). A clearly visible stochastic feature of brain activity is the trial-to-trial variability of neuronal responses, which also appears on virtually every spatial and temporal scale that has been examined (Faisal et al., 2008). This variability has often been interpreted as side-effect of an implementation of inherently deterministic computing paradigms with noisy elements, and it has been attempted to show that the observed noise can be eliminated through spatial or temporal averaging. However, more recent experimental methods, which make it possible to record simultaneously from many neurons (or from many voxels in fMRI), have shown that the underlying probability distributions of network states during spontaneous activity are highly structured and multimodal, with distinct modes that resemble those encountered during active processing. This has been shown through recordings with voltage-sensitive dyes starting with (Tsodyks et al., 1999; Kenet et al., 2003), multi-electrode arrays (Luczak et al., 2009b), and fMRI (Raichle, 2010; Lewis et al., 2009). It was also shown that the intrinsic trial-to-trial variability of brain systems is intimately related to the observed trial-to-trial variability in behavior (see e.g. (Fox et al., 2007)). Furthermore, in (Kelemen and Fenton, 2010) it was shown that during navigation in a complex environment where simultaneously two spatial frames of reference were relevant, the firing of neurons in area *CA1* represented both frames in alternation, so that coactive neurons tended to relate to a common frame of reference. In addition it has been shown that in a situation where sensory stimuli are ambiguous, large brain networks switch stochastically between alternative interpretations or percepts, see (Leopold and Logothetis, 1996, 1999; Kim and Blake, 2005). Furthermore, an increase in the volatility of network states has been shown to accompany episodes of behavioral uncertainty (Karlsson et al., 2012). All these experimental data point to inherently stochastic aspects in the organization of brain computations, and more specifically to an important computational role of spontaneously varying network states of smaller and larger networks of neurons in the brain. However, one should realize that the approach to stochastic computation that we examine in this article does not postulate that all brain activity is stochastic or unreliable, since reliable neural responses can be represented by probabilities close to 1.

The goal of this article is to provide a theoretical foundation for understanding stochastic computations in networks of neurons in the brain, in particular also for the generation of structured spontaneous activity. To this end, we prove here that even biologically realistic models C for networks of neurons in the brain have – for a suitable definition of network state – a unique stationary distribution p_C of network states. Previous work had focused in this context on neuronal models with linear sub-threshold dynamics (Brémaud and Massoulié, 1996; Borovkov et al., 2012) and constant external input (e.g. constant input firing rates). However, we show here that this holds even for quite realistic models that reflect, for example, data on nonlinear dendritic integration (dendritic spikes), synapses with data-based short term dynamics (i.e., individual mixtures of depression and facilitation), and different types of neurons on specific laminae. We also show that these results are not restricted to the case of constant external input, but rather can be extended to periodically changing input, and to input generated by arbitrary ergodic stochastic processes.

Our theoretical results imply that virtually any data-based model C, for networks of neurons featuring realistic neuronal noise sources (e.g. stochastic synaptic vesicle release) implements a Markov process through its stochastic dynamics. This can be interpreted – in spite of its non-reversibility – as a form of sampling from a unique stationary distribution p_C. One interpretation of p_C, which is in principle consistent with our findings, is that it represents the posterior distribution of a Bayesian inference operation (Hoyer and Hyvärinen, 2003; Berkes et al., 2011; Buesing et al., 2011; Pecevski et al., 2011), in which the current input (evidence) is combined with prior knowledge encoded in network parameters such as synaptic weights or intrinsic excitabilities of neurons (see (Friston, 2010; Vilares and Kording, 2011; Fiser et al., 2010; Doya et al., 2007) for an introduction to the "Bayesian brain"). This interpretation of neural dynamics as sampling from a posterior distribution is intriguing, as it implies that various results of probabilistic inference could then be easily obtained by a simple readout mechanism: For example, posterior marginal probabilities can be estimated (approximately) by observing the number of spikes of specific neurons within some time window (see related data from parietal cortex (Huk and Shadlen, 2005)). Furthermore, an approximate maximal a posteriori (MAP) inference can be carried out by observing which network states occur more often, and/or are more persistent.

A crucial issue which arises is whether reliable readouts from p_C in realistic cortical microcircuit models can be obtained quickly enough to support, e.g., fast decision making in downstream areas. This critically depends on the speed of convergence of the distribution of network states (or distribution of trajectories of network states) from typical initial network states to the stationary distribution. Since the initial network state of a cortical microcircuit C depends on past activity, it may often be already quite "close" to the stationary distribution when a new input arrives (since past inputs are likely related to the new input). But it is also reasonable to assume that the initial state of the network is frequently unrelated to the stationary distribution p_C, for example after drastic input changes. In this case the time required for readouts depends on the expected convergence speed to p_C from – more or less – *arbitrary* initial states. We show that one can prove exponential upper bounds for this convergence speed. But even that does not guarantee fast convergence for a concrete system, because of constant factors in the theoretical upper bound. Therefore we complement this theoretical analysis of the convergence speed by extensive computer simulations for cortical microcircuit models.

The notion of a cortical microcircuit arose from the observation that "it seems likely that there is a basically uniform microcircuit pattern throughout the neocortex upon which certain specializations unique to this or that cortical area are superimposed" (Mountcastle, 1998). This notion is not precisely defined, but rather a term of convenience: It refers to network models that are sufficiently large to contain examples of the main types of experimentally observed neurons on specific laminae, and the main types of experimentally observed synaptic connections between different types of neurons on different laminae, ideally in statistically representative numbers (Douglas and Martin, 2004). Computer simulations of cortical microcircuit models are practically constrained both by a lack of sufficiently many consistent data from a single preparation and a single cortical area, and by the available computer time. In the computer simulations for this article we have focused on a relatively simple standard model for a cortical microcircuit in the somatosensory cortex (Haeusler and Maass, 2007) that has already been examined in some variations in previous studies from various perspectives (Haeusler et al., 2009; Rasch et al., 2011; Potjans and Diesmann, 2012; Bastos et al., 2012).

We show that for this standard model of a cortical microcircuit marginal probabilities for single random variables (neurons) can be estimated through sampling even for fairly large instances with 5000 neurons within a few 100 ms of simulated biological time, hence well within the range of experimentally observed computation times of biological organisms. The same holds for probabilities of network states for small sub-networks. Furthermore, we show that at least for sizes up to 5000 neurons these "computation times" are virtually independent of the size of the microcircuit model.

We also address the question to which extent our theoretical framework can be applied in the context of periodic input, for example in the presence of background theta oscillations (Dragoi and Buzsaki, 2006). In contrast to the stationary input case, we show that the presence of periodic input leads to the emergence of unique *phase-specific* stationary distributions, i.e., a separate unique stationary distribution for each phase of the periodic input. We discuss basic implications of this result and relate our findings to experimental data on theta-paced path sequences (Dragoi and Buzsaki, 2006; Gupta et al., 2012) and bi-stable activity (Jezek et al., 2011) in hippocampus.

Finally, our theoretically founded framework for stochastic computations in networks of spiking neurons also throws new light on the question how complex constraint satisfaction problems could be solved by cortical microcircuits (Hinton et al., 1984; Davenport et al., 1994). We demonstrate this in a toy example for the popular puzzle game Sudoku. We show that the constraints of this problem can be easily encoded by synaptic connections between excitatory and inhibitory neurons in such a way that the stationary distribution p_C assigns particularly high probability to those network states which encode correct (or good approximate) solutions to the problem. The resulting network dynamics can also be understood as parallel stochastic search with anytime computing properties: Early network states provide very fast heuristic solutions, while later network states are distributed according to the stationary distribution p_C, therefore visiting with highest probability those solutions which violate only a few or zero constraints.

In order to make the results of this article accessible to non-theoreticians we present in the subsequent Results section our main findings in a less technical formulation that emphasizes relationships to experimental data. Rigorous mathematical definitions and proofs can be found in the Methods section, which has been structured in the same way as the Results section in order to facilitate simultaneous access on different levels of detail.

2.2 Network states and distributions of network states

A simple notion of network state at time t simply indicates which neurons in the network fired within some short time window before t. For example, in (Berkes et al., 2011) a window size of 2 ms was selected. However, the full network state could not be analyzed there experimentally, only its projection onto 16 electrodes in area V1 from which recordings were made. An important methodological innovation of (Berkes et al., 2011) was to analyze under various conditions the probability distribution of the recorded fragments of network states, i.e., of the resulting bit vectors of length 16 (with a "1" at position i if a spike was recorded during the preceding 2 ms at electrode i). In particular, it was shown that during development the distribution over these 2^{16} network states during spontaneous activity in darkness approximates the distribution recorded during natural vision. Apart from its functional interpretation, this result also raises the even more fundamental question how a network of neurons in the brain can represent and generate a complex distribution of network states. This question is addressed here in the context of data-based models C for cortical microcircuits. We consider notions of network states y similar to (Berkes et al., 2011) (see the simple state $y_S(t)$ in Figure 2.1C) and provide a rigorous proof that under some mild assumptions any such model C represents and generates for different external inputs x associated different internal distributions $p_C(y|x)$ of network states y. More precisely, we will show that for any specific input x there exists a unique stationary distribution $p_C(y|x)$ of network states y to which the network converges exponentially fast from any initial state.

This result can be derived within the theory of Markov processes on general state spaces, an extension of the more familiar theory of Markov chains on finite state spaces to continuous time and infinitely many network states. Another important difference to typical Markov chains (e.g. the dynamics of Gibbs sampling in Boltzmann machines) is that the Markov processes describing the stochastic dynamics of cortical microcircuit models are non-reversible. This is a well-known difference between simple neural network models and networks of spiking neurons in the brain, where a spike of a neuron causes postsynaptic potentials in other neurons - but not vice versa. In addition, experimental results show that brain networks tend to have a non-reversible dynamics also on longer time scales (e.g., stereotypical trajectories of network states (Abeles et al., 1995; Luczak et al., 2007; Buzsáki, 2010; Luczak and MacLean, 2012)).

In order to prove results on the existence of stationary distributions $p_C(y|x)$ of network states y, one first needs to consider a more complex notion of network state $y_M(t)$ at time t, which records the history of all spikes in the network C since time $t-\Theta$ (see Figure 1C). The window length Θ has to be chosen sufficiently large so that the influence of spikes before time $t-\Theta$ on the dynamics of the network after time t can be neglected. This more complex notion of network state then fulfills the *Markov property*, such that the future network evolution depends on the past only through the current Markov state. The existence of a window length Θ with the Markov property is a basic assumption of the subsequent theoretical results. For standard models of networks of spiking neurons a value of Θ around 100 ms provides already a good approximation of the Markov property, since this is a typical time during which a post-synaptic potential has a non-negligible effect at the soma of a post-synaptic neuron. For more complex models of networks of spiking neurons a larger value of Θ in the range of seconds is more adequate, in order to accommodate for dendritic spikes or the activation of $GABA_B$ receptors that may last 100 ms or longer, and the short term dynamics of synapses with time constants of several hundred milliseconds. Fortunately, once the existence of a stationary distribution is proved for such more complex notion of network state, it also holds for any simpler notion of network state (even if these simpler network states do not fulfill the Markov property), that results when one ignores details of the more complex network states. For example, one can ignore all spikes before time $t-2$ ms, the exact firing times within the window from $t-2$ ms to t, and whether a neuron fired one or several spikes. In this way one arrives back at the simple notion of network state from (Berkes et al., 2011).

Theorem 1 (Exponentially fast convergence to a stationary distribution) *Let C be an arbitrary model for a network of spiking neurons with stochastic synaptic release or some other mechanism for stochastic firing. C may consist of complex multi-compartment neuron models with nonlinear dendritic integration (including dendritic spikes) and heterogeneous synapses with differential short term dynamics. We assume that this network C receives external inputs from a set of input neurons $i = 1 \ldots N$ which fire according to Poisson processes at different rates $x_i(t)$. The vector $x(t)$ of input rates can be either constant over time ($x(t) \equiv x$), or generated by any external Markov process that converges exponentially fast to a stationary distribution.*

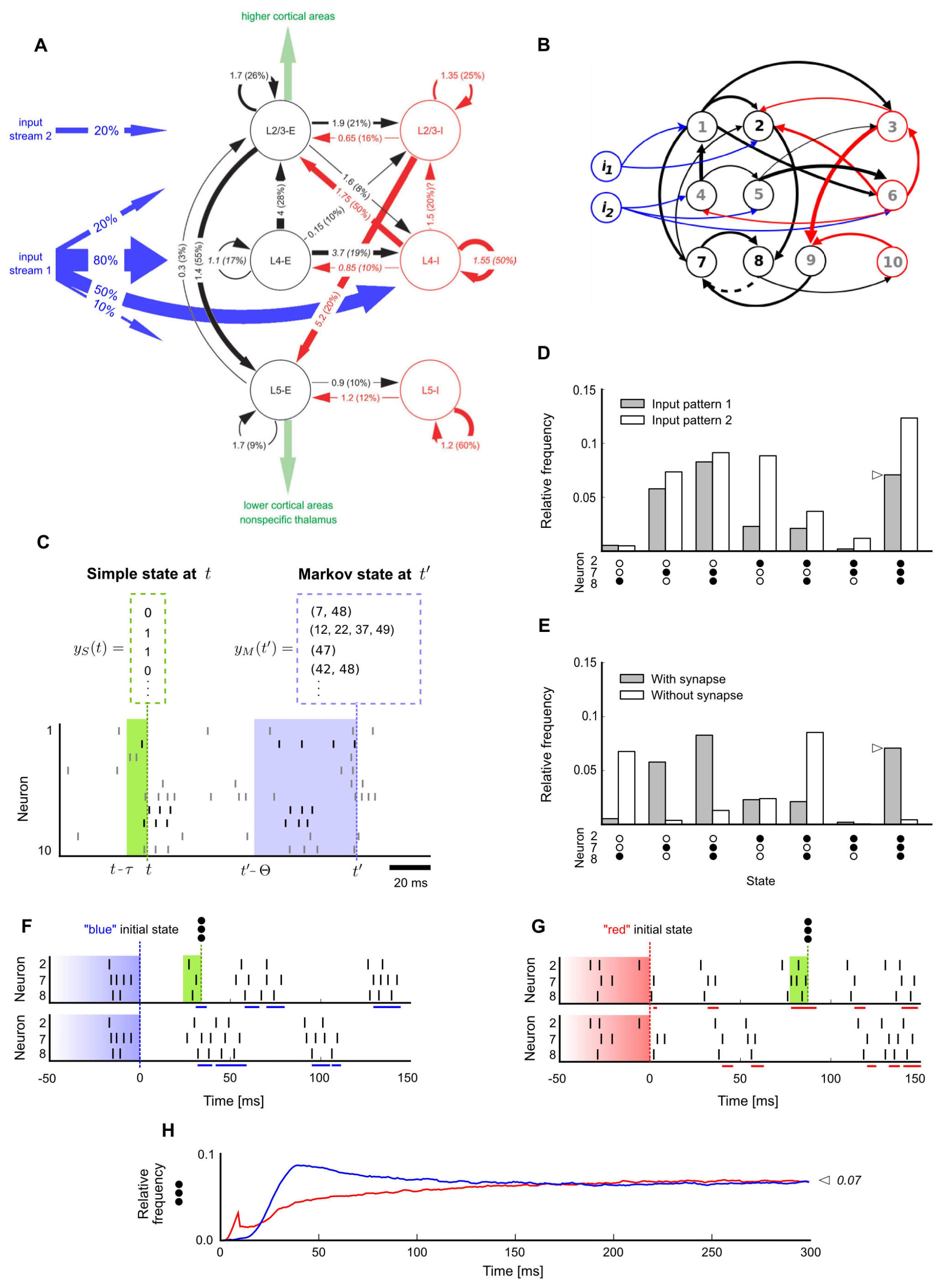

Figure 2.1: (see next page for Figure caption)

Figure 2.1: Network states and stationary distributions of network states in a cortical microcircuit model. **A**. Data-based cortical microcircuit template from (Haeusler and Maass, 2007); © 2007 by Oxford University Press, reprinted by permission of the authors and Oxford University Press. **B**. A small instantiation of this model consisting of 10 network neurons $1, \ldots, 10$ and 2 additional input neurons i_1, i_2. Neurons are colored by type (blue:input, black:excitatory, red:inhibitory). Line width represents synaptic efficacy. The synapse from neuron 8 to 7 is removed for the simulation described in E. **C**. Notions of network state considered in this article. Markov states are defined by the exact timing of all recent spikes within some time window Θ, shown here for $\Theta = 50$ ms. Simple states only record which neurons fired recently (0=no spike, 1=at least one spike within a short window τ, with $\tau = 10$ ms throughout this figure). **D**. Empirically measured stationary distribution of simple network states. Shown is the marginal distribution $p_C(\tilde{y}|x)$ for a subset of three neurons 2,7,8 (their spikes are shown in C in black), under two different input conditions (input pattern 1: i_1 firing at 10 Hz and i_2 at 50 Hz, input pattern 2: i_1 at 50 Hz and i_2 at 10 Hz). The distribution for each input condition was obtained by measuring the relative time spent in each of the simple states (0,0,0), ..., (1,1,1) in a single long trial (100 s). The zero state (0,0,0) is not shown. **E**. Effect of removing one synapse, from neuron 8 to neuron 7, on the stationary distribution of network states (input pattern 1 was presented). **F**. Illustration of trial-to-trial variability in the small cortical microcircuit (input pattern 1). Two trials starting from identical initial network states $y_M(0)$ are shown. Blue bars at the bottom of each trial mark periods where the subnetwork of neurons 2,7,8 was in simple state (1,1,1) at this time t. Note that the "blue" initial Markov state is shown only partially: it is actually longer and comprises all neurons in the network (as in panel C, but with $\Theta = 1s$). **G**. Two trials starting from a different ("red") initial network state. Red bars denote periods of state (1,1,1) for "red" trials. **H**. Convergence to the stationary distribution p_C in this small cortical microcircuit is fast and independent of the initial state: This is illustrated for the relative frequency of simple state (1,1,1) within the first 300 ms after input onset. The blue/red line shows the relative frequency of simple state (1,1,1) at each time t estimated from many (10^5) "blue"/"red" trials. The relative frequency of simple state (1,1,1) rapidly converges to its stationary value denoted by the symbol $\triangleleft$ (marked also in panels D and E). The relative frequency converges to the same value regardless of the initial state (blue/red).

Then there exists a stationary distribution $p_C(y|x)$ of network states y, to which the stochastic dynamics of C converges from any initial state of the network exponentially fast. Accordingly, the distribution of subnetwork states $\tilde{y}$ of any subset of neurons converges exponentially fast to the marginal distribution $p_C(\tilde{y}|x)$ of this subnetwork.

Note that Theorem 1 states that the network embodies not only the joint distribution $p_C(y|x)$ over all neurons, but simultaneously all marginal distributions $p_C(\tilde{y}|x)$ over all possible subsets of neurons. This property follows naturally from the fact that $p_C(y|x)$ is represented in a sample-based manner (Fiser et al., 2010). As a consequence, if one is interested in estimating the marginal distribution of some subset of neurons rather than the full joint distribution, it suffices to observe the activity of the particular subnetwork of interest (while ignoring the remaining network). This is remarkable insofar, as the exact computation of marginal probabilities is in general known to be quite difficult (even NP-complete (Koller and Friedman, 2009)).

Theorem 1 requires that neurons fire stochastically. More precisely, a basic assumption required for Theorem 1 is that the network behaves sufficiently stochastic at *any point in time*, in the sense that the probability that a neuron fires in an interval $[t, t + \delta t)$ must be smaller than 1 for any t. This is indeed fulfilled by any stochastic neuron model as long as instantaneous firing rates remain bounded. It is also fulfilled by any deterministic neuron model if synaptic transmission is modeled via stochastic vesicle release with bounded release rates. Another assumption is that long-term plasticity and other long-term memory effects have a negligible impact on the network dynamics on shorter timescales which are the focus of this article (milliseconds to a few seconds). Precise mathematical definitions of all assumptions and notions involved in Theorem 1 as well as proofs can be found in Methods (see Lemma 2 and 3).

An illustration for Theorem 1 is given in Figure 2.1. We use as our running example for a cortical microcircuit model C the model of (Haeusler and Maass, 2007) shown in Figure 2.1A, which consists of three populations of excitatory and three populations of inhibitory neurons on specific laminae. Average strength of synaptic connections (measured as mean amplitude of postsynaptic potentials at the soma in mV, and indicated by the numbers at the arrows in Figure 2.1A) as well as the connection probability (indicated in parentheses at each arrow as % in Figure 2.1A) are based in this model on intracellular recordings from 998 pairs of identified neurons from the Thomson Lab (Thomson et al., 2002). The thickness of arrows in Figure 2.1A reflects the products of those two numbers for each connection. The nonlinear short-term dynamics of each type of synaptic connection was modeled according to data from the Markram Lab (Gupta et al., 2000; Markram et al., 1998). Neuronal integration and spike generation was modeled by a conductance-based leaky-integrate-and-fire model, with a stochastic spiking mechanism based on (Jolivet et al., 2006). See Methods for details.

The external input x consists in a cortical microcircuit of inputs from higher cortical areas that primarily target neurons in superficial layers, and bottom-up inputs that arrive primarily in layer 4, but also on other layers (details tend to depend on the cortical area and the species). We model two input streams in a qualitative manner as in (Haeusler and Maass, 2007). Also background synaptic input is modeled according to (Haeusler and Maass, 2007).

Figure 2.1B shows a small instantiation of this microcircuit template consisting of 10 neurons (we had to manually tune a few connections in this circuit to facilitate visual clarity of subsequent panels). The impact of different external inputs x and of a single synaptic connection from neuron 8 to neuron 7 on the stationary distribution is shown in Figure 2.1D and E, respectively (shown is the marginal distribution $p_C(\tilde{y}|x)$ of a subset of three neurons 2,7 and 8). This illustrates that the structure and dynamics of a circuit C are intimately linked to properties of its stationary distribution $p_C(y|x)$. In fact, we argue that the stationary distribution $p_C(y|x)$ (more precisely: the stationary distribution $p_C(y|x)$ for all relevant external inputs x) can be viewed as a mathematical model for the most salient aspects of stochastic computations in a circuit C.

The influence of the initial network state on the first 150 ms of network response is shown in Figure 2.1F and G for representative trials starting from two different initial Markov states (blue/red, two trials shown for each). Variability among trials arises from the inherent stochasticity of neurons and the presence of background synaptic input. Figure 2.1H is a concrete illustration of Theorem 1: it shows that the relative frequency of a specific network state (1,1,1) in a subset of the three neurons 2,7 and 8 converges quickly to its stationary value. Furthermore, it converges to this (same) value regardless of the initial network state (blue/red).

2.3 Stationary distributions of trajectories of network states

Theorem 1 also applies to networks which generate stereotypical trajectories of network activity (Luczak et al., 2007). For such networks it may be of interest to consider not only the distribution of network states in a short window (e.g. simple states with $\tau = 10$ ms, or $\Theta = 50$ ms), but also the distribution of longer trajectories produced by the network. Indeed, since Theorem 1 holds for Markov states y_M with any fixed window length Θ, it also holds for values of Θ that are in the range of experimentally observed trajectories of network states (Mazor and Laurent, 2005; Luczak et al., 2007; Harvey et al., 2012). Hence, a generic neural circuit C automatically has a unique stationary distribution over *trajectories* of (simple) network states for any fixed trajectory length Θ. Note that this implies that a neural circuit C has simultaneously stationary distributions of trajectories of (simple) network states of various lengths for arbitrarily large Θ, and a stationary distribution of simple network states. This fact is not surprising if one takes into consideration that if a circuit C has a stationary distribution over simple network states this does *not* imply that subsequent simple network states represent independent drawings from this stationary distribution. Hence the circuit C may very well produce stereotypical trajectories of simple network states. This feature becomes even more prominent if the underlying dynamics (the Markov process) of the neural circuit is non-reversible on several time scales.

2.4 Extracting knowledge from internally stored distributions

We address two basic types of knowledge extraction from a stationary distribution p_C of a network C: the computation of *marginal probabilities* and *maximal a posteriori (MAP) assignments*. Both computations constitute basic inference problems commonly appearing in real-world applications (Wainwright and Jordan, 2008), which are in general difficult to solve as they involve large sums, integrals, or maximization steps over a state space which grows exponentially in the number of random variables. However, already (Fiser et al., 2010; Buesing et al., 2011) noted that the *estimation* of marginal probabilities would become straightforward if distributions were represented in the brain in a sample-based manner (such that each network state at time t represents one sample from the distribution). Theorem 1 provides a theoretical foundation for how such a representation could emerge in realistic data-based microcircuit models on the implementation level: Once the network C has converged to its stationary distribution, the network state at any time t represents a sample from $p_C(y|x)$ (although subsequent samples are generally not independent). Simultaneously, the subnetwork state $\tilde{y}(t)$ of any subset of neurons represents a sample from the marginal distribution $p_C(\tilde{y}|x)$. This is particularly relevant if one interprets $p_C(y|x)$ in a given cortical microcircuit C as the posterior distribution of an implicit generative model, as suggested for example by (Berkes et al., 2011) or (Buesing et al., 2011; Pecevski et al., 2011).

In order to place the estimation of marginals into a biologically relevant context, assume that a particular component y_1 of the network state $y = (y_1, \ldots, y_K)$ has a behavioral relevance. This variable y_1, represented by some neuron n_1, could represent for example the perception of a particular visual object (if neuron n_1 is located in inferior temporal cortex (Zhang et al., 2011)), or the intention to make a saccade into a specific part of the visual field (if neuron n_1 is located in area LIP (Shadlen and Newsome, 2001)). Then the computation of the marginal

$$p_C(y_1 = 1|x) = \sum_{v_2 \in \{0,1\}, \ldots, v_K \in \{0,1\}} p_C(y_1 = 1, y_2 = v_2, \ldots, y_K = v_K|x) \qquad (2.1)$$

would be of behavioral significance. Note that this computation integrates information from the internally stored knowledge p_C with evidence about a current situation x. In general this computation is demanding as it involves a sum with exponentially many terms in the network size K.

But according to Theorem 1, the correct marginal distribution $p_C(y_1|x)$ is automatically embodied by the activity of neuron n_1. Hence the marginal probability $y_1 = 1$ can be estimated by simply observing what fraction of time the neuron spends in the state $y_1 = 1$, while ignoring the activity of the remaining network (Buesing et al., 2011). In principle, a downstream neuron could gather this information by integrating the spike output of n_1 over time.

Marginal probabilities of subpopulations, for example $p_C(y_1 = 1, y_2 = 0, y_3 = 1|x)$, can be estimated in a similar manner by keeping track of how much time the subnetwork spends in the state (1,0,1), while ignoring the activity of the remaining neurons. A downstream network could gather this information, for example, by integrating over the output of a readout neuron which is tuned to detect the desired target pattern (1,0,1).

Notably, the estimation of marginals sketched above is guaranteed by ergodic theory to converge to the correct probability as observation time increases (due to Theorem 1 which ensures that the network is an ergodic Markov process, see

Methods). In particular, this holds true even for networks with prominent sequential dynamics featuring, for example, stereotypical trajectories. However, note that the observation time required to obtain an accurate estimate may be longer when trajectories are present since subsequent samples gathered from such a network will likely exhibit stronger dependencies than in networks lacking sequential activity patterns. In a practical readout implementation where recent events might be weighed preferentially this could result in more noisy estimates.

Approximate maximal a posteriori (MAP) assignments to small subsets of variables $y_1, \ldots, y_m$ can also be obtained in a quite straightforward manner. For given external inputs x, the marginal MAP assignment to the subset of variables $y_1, \ldots, y_m$ (with some $m \leq K$) is defined as the set of values $\hat{v}_1, \ldots, \hat{v}_m$ that maximize

$$\sum_{v_{m+1} \in \{0,1\}, \ldots, v_K \in \{0,1\}} p_C(y_1 = \hat{v}_1, \ldots, y_m = \hat{v}_m, \;\; y_{m+1} = v_{m+1}, \ldots, y_K = v_K | x) \;\; . \tag{2.2}$$

A sample-based approximation of this operation can be implemented by keeping track of which network states in the subnetwork $n_1, \ldots, n_m$ occur most often. This could, for example, be realized by a readout network in a two stage process: first the marginal probabilities $p_C(y_1 = \hat{v}_1, y_2 = \hat{v}_2, y_3 = \hat{v}_3 | x)$ of all $2^3 = 8$ subnetwork states $(0,0,0), \ldots, (1,1,1)$ are estimated (by 8 readout neurons dedicated to that purpose), followed by the selection of the neuron with maximal probability. The selection of the maximum could be achieved in a neural network, for example, through competitive inhibition. Such competitive inhibition would ideally lead to a winner-take-all function such that the neuron with the strongest stimulation (representing the variable assignment with the largest probability) dominates and suppresses all other readout neurons.

2.5 Estimates of the required computation time

Whereas many types of computations (for example probabilistic inference via the junction tree algorithm (Wainwright and Jordan, 2008)) require a certain computation time, probabilistic inference via sampling from an embodied distribution p_C belongs to the class of *anytime computing* methods, where rough estimates of the result of a computation become almost immediately available, and are automatically improved when there is more time for a decision. A main component of the convergence time to a reliable result arises from the time which the distribution of network states needs to become independent of its initial state y_0. It is well known that both, network states of neurons in the cortex (Arieli et al., 1996) and quick decisions of an organism, are influenced for a short time by this initial state y_0 (and this temporary dependence on the initial state y_0 may in fact have some behavioral advantage, since y_0 may contain information about preceding network inputs, expectations, etc.). But it has remained unknown, what range of convergence speeds for inference from p_C is produced by common models for cortical microcircuits C.

We address this question by analyzing the convergence speed of stochastic computations in the cortical microcircuit model of (Haeusler and Maass, 2007). A typical network response of an instance of the cortical microcircuit model comprising 560 neurons as in (Haeusler and Maass, 2007) is shown in Figure 2.2A. We first checked how fast marginal probabilities for single neurons converge to stationary values from different initial network Markov states. We applied the same analysis as in Figure 2.1H to the simple state ($\tau = 10$ ms) of a single representative neuron from layer 5. Figure 2.2B shows quite fast convergence of the "on"-state probability of the neuron to its stationary value from two different initial states. Note that this straightforward method of checking convergence is rather inefficient, as it requires the repetition of a large number of trials for each initial state. In addition it is not suitable for analyzing convergence to marginals for subpopulations of neurons (see Figure 2.2G).

Various more efficient *convergence diagnostics* have been proposed in the context of discrete-time Markov Chain Monte Carlo theory (Gelman and Rubin, 1992; Cowles and Carlin, 1996; Brooks et al., 2010; Gjoka et al., 2010). In the following, we have adopted the Gelman and Rubin diagnostic, one of the standard methods in applications of MCMC sampling (Gelman and Rubin, 1992). The Gelman Rubin convergence diagnostic is based on the comparison of many runs of a Markov chain when started from different randomly drawn initial states. In particular, one compares the typical variance of state distributions during the time interval $[t, 2t]$ within a single run (within-variance) to the variance during the interval $[t, 2t]$ between different runs (between-variance). When the ratio $\hat{R}$ of between- and within-variance approaches 1 this is indicative of convergence. A comparison of panels B and C of Figure 2.2 shows that in the case of marginals for single neurons this interpretation fits very well to the empirically observed convergence speed for two different initial conditions. Various values between 1.02 (Gjoka et al., 2010) and 1.2 (Kass et al., 1998; Gelman et al., 2004; Brooks et al., 2010) have been proposed in the literature as thresholds below which the ratio $\hat{R}$ signals that convergence has taken place. The shaded region in Figure 2.2C-G corresponds to $\hat{R}$ values below a threshold of 1.1. An obvious advantage of the Gelman-Rubin diagnostic, compared with a straightforward empirical evaluation of convergence properties as in Figure 2.2B, is its substantially larger computational efficiency and the larger number of initial states that it takes into account. For the case of multivariate marginals (see Figure 2.2G), a straightforward empirical evaluation of convergence is not even feasible, since relative frequencies of 2^{30} states would have to be analyzed.

Using the Gelman-Rubin diagnostic, we estimated convergence speed for marginals of single neurons (see Figure 2.2C, mean/worst in Figure 2.2E), and for the product of the simple states of two neurons (i.e., pairwise spike coincidences) in Figure 2.2D. We found that in all cases the Gelman-Rubin value drops close to 1 within just a few 100 ms. More precisely, for a typical threshold of 1.2 convergence

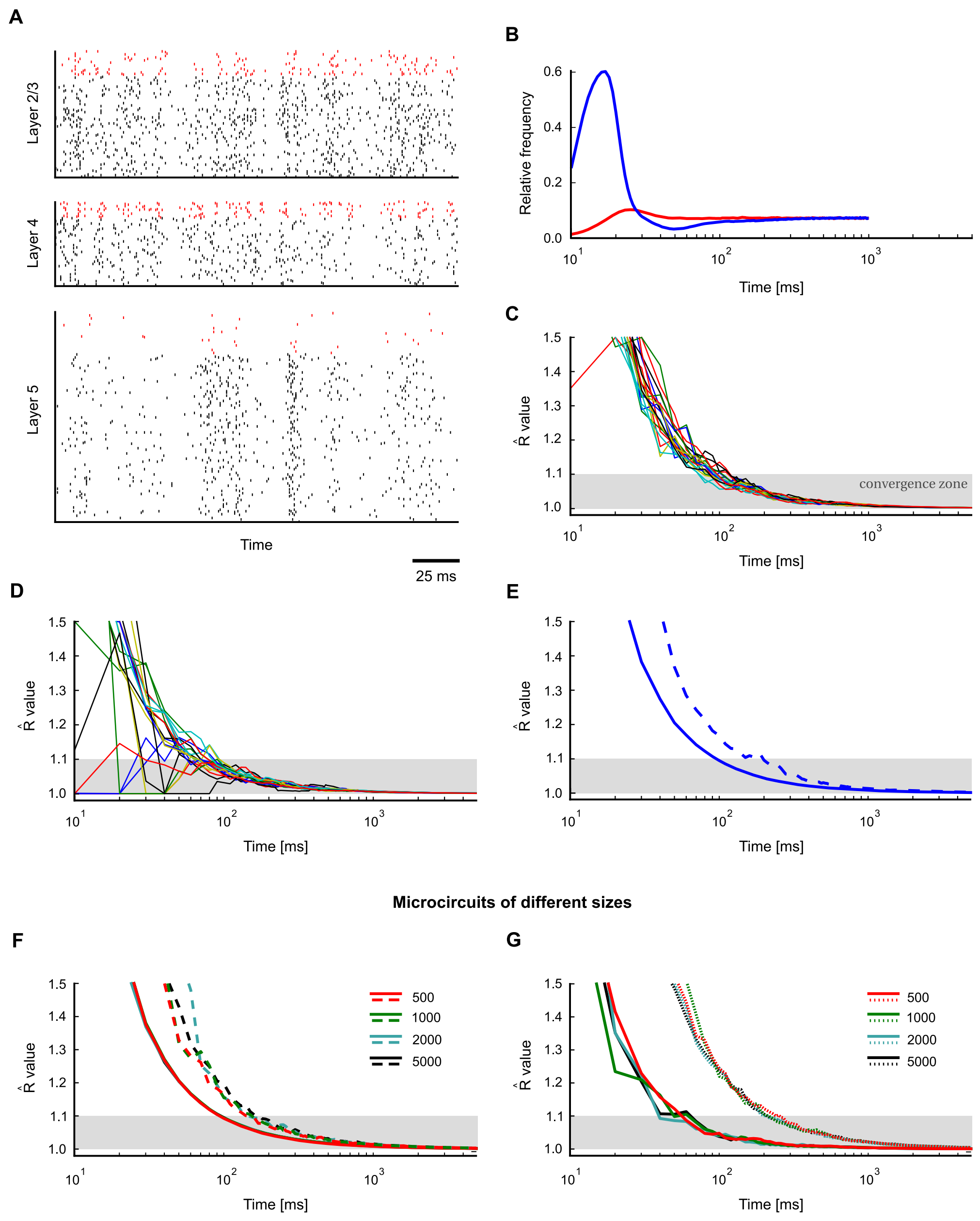

Figure 2.2: (see next page for Figure caption)

times are slightly below 100 ms in Figure 2.2C-E. A very conservative threshold of 1.02 yields convergence times close to 600 ms.

The above simulations were performed in a circuit of 560 neurons, but eventually one is interested in the properties of much larger circuits. Hence, a crucial

Figure 2.2: Fast convergence of marginals of single neurons and more complex quantities in a cortical microcircuit model. **A**. Typical spike response of the microcircuit model based on (Haeusler and Maass, 2007) comprising 560 stochastic point neurons. Spikes of inhibitory neurons are indicated in red. **B**. Fast convergence of a marginal for a representative layer 5 neuron (frequency of "on"-state, with $\tau = 10$ ms) to its stationary value, shown for two different initial Markov states (blue/red). Statistics were obtained for each initial state from 10^5 trials. **C**. Gelman-Rubin convergence diagnostic was applied to the marginals of all single neurons (simple states, $\tau = 10$ ms). In all neurons the Gelman-Rubin value $\hat{R}$ drops to a value close to 1 within a few 100 ms, suggesting generally fast convergence of single neuron marginals (shown are 20 randomly chosen neurons; see panel E for a summary of all neurons). The shaded area below 1.1 indicates a range where one commonly assumes that convergence has taken place. **D**. Convergence speed of pairwise spike coincidences (simple states (1,1) of two neurons, 20 randomly chosen pairs of neurons) is comparable to marginal convergence. **E**. Summary of marginal convergence analysis for single neurons in C: Mean (solid) and worst (dashed line) marginal convergence of all 560 neurons. Mean/-worst convergence is reached after a few 100 ms. **F**. Convergence analysis was applied to networks of different sizes (500-5000 neurons). Mean and worst marginal convergence of single neurons are hardly affected by network size. **G**. Convergence properties of populations of neurons. Dotted: multivariate Gelman-Rubin analysis was applied to a subpopulation of 30 neurons (5 neurons were chosen randomly from each pool). Solid: convergence of a "random readout" neuron which receives spike inputs from 500 randomly chosen neurons in the microcircuit. It turns out that the convergence speed of such a generic readout neuron is even slightly faster than for neurons within the microcircuit (compare with panel E). A remarkable finding is that in all these cases the network size does not affect convergence speed.

question is how the convergence properties scale with the network size. To this end, we compared convergence in the cortical microcircuit model of (Haeusler and Maass, 2007) for four different sizes (500, 1000, 2000 and 5000). To ensure that overall activity characteristics are maintained across different sizes, we adopted the approach of (Haeusler and Maass, 2007) and scaled recurrent postsynaptic potential (PSP) amplitudes inversely proportional to network size. A comparison of mean (solid line) and worst (dashed line) marginal convergence for networks of different sizes is shown in Figure 2.2F. Notably we find that the network size has virtually no effect on convergence speed. This suggests that, at least within the scope of the laminar microcircuit model of (Haeusler and Maass, 2007), even very large cortical networks may support fast extraction of knowledge (in particular marginals) from their stationary distributions $p_C(y|x)$.

In order to estimate the required computation time associated with the estimation of marginal probabilities and MAP solutions on small *subpopulations* $n_1, \ldots, n_m$, one needs to know how fast the marginal probabilities of *vector-valued* states $(y_1, \ldots, y_m)$ of subnetworks of C become independent from the initial state of the network. To estimate convergence speed in small subnetworks, we applied a multivariate version of the Gelman-Rubin method to vector-valued simple states of subnetworks (Figure 2.2G, dotted lines, evaluated for varying circuit sizes from 500 to 5000 neurons). We find that multivariate convergence of state frequencies for a population of $m = 30$ neurons is only slightly slower than for uni-variate marginals. To complement this analysis, we also investigated convergence properties of a "random readout" neuron which integrates inputs from many neurons in a subnetwork. It is interesting to note that the convergence speed of such a readout neuron, which receives randomized connections from a randomly chosen subset of 500 neurons, is comparable to that of single marginals (Figure 2.2F, solid lines), and in fact slightly faster.

2.6 Impact of different dynamic regimes on the convergence time

An interesting research question is which dynamic or structural properties of a cortical microcircuit model C have a strong impact on its convergence speed to the stationary distribution p_C. Unfortunately, a comprehensive treatment of this question is beyond the scope of this paper, since virtually any aspect of circuit dynamics could be investigated in this context. Even if one focuses on a single aspect, the impact of one circuit feature is likely to depend on the presence of other features (and probably also on the properties of the input). Nonetheless, to lay a foundation for further investigation, first empirical results are given in Figure 2.3.

As a reference point, Figure 2.3A shows a typical activity pattern and convergence speed of single marginals in the small cortical microcircuit model from Figure 2.1. To test whether the overall activity of a network has an obvious impact on convergence speed, we constructed a small network of 20 neurons (10 excitatory, 10 inhibitory) and tuned connection weights to achieve sparse overall activity (Figure 2.3B). A comparison of panels A and B suggests that overall network activity has no significant impact on convergence speed. To test whether the presence of stereotypical trajectories of network states (similar to (Luczak et al., 2007)) has a noticeable influence on convergence, we constructed a small network exhibiting strong sequential activity patterns (see Figure 2.3C). We find that convergence speed is hardly affected, except for the first 200 ms (see Figure 2.3C). Within the scope of this first empirical investigation, we were only able to produce a significant slow-down of the convergence speed by building a network that alternated between two attractors (Figure 2.3D).

2.7 Distributions of network states in the presence of periodic input

In Theorem 1 we had already addressed one important case where the network C receives dynamic external inputs: the case when external input is generated by some Markov process. But many networks of neurons in the brain are also subject to more or less pronounced periodic inputs ("brain rhythms" (Engel et al., 2001;

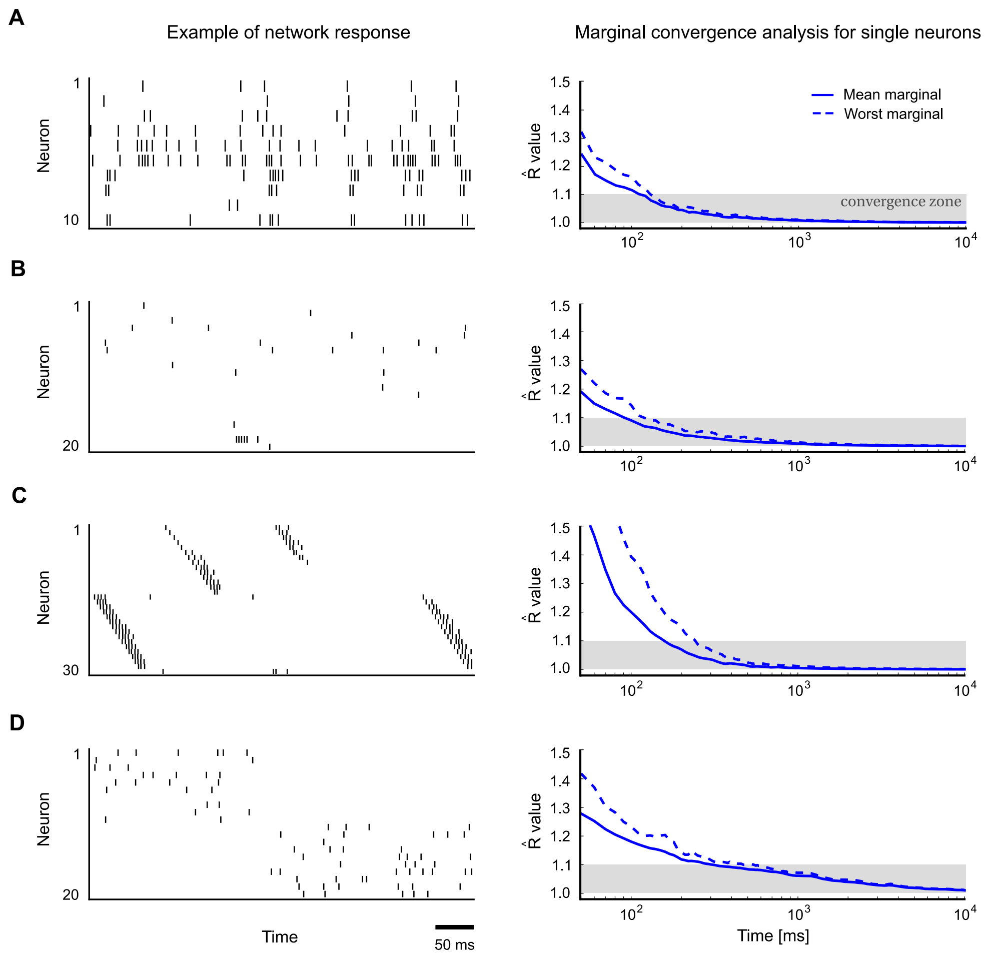

Figure 2.3: Impact of network architecture and network dynamics on convergence speed. Convergence properties for single neurons (as in Figure 2.2C) in different network architectures were assessed using univariate Gelman-Rubin analysis. Typical network activity is shown on the left, convergence speed on the right (solid: mean marginal, dashed: worst marginal). **A**. Small cortical column model from Figure 2.1 (input neurons not shown). **B**. Network with sparse activity (20 neurons). **C**. Network with stereotypical trajectories (50 neurons, inhibitory neurons not shown). Despite strongly irreversible dynamics, convergence is only slightly slower. **D**. Network with bistable dynamics (two competing populations, each comprising 10 neurons). Convergence is slower in this circuit due to low-frequency switching dynamics between two attractors.

Buzsaki, 2009; Wang, 2010)), and it is known that these interact with knowledge represented in distributions of network states in specific ways. For instance, it had been shown in (Dragoi and Buzsaki, 2006) that the phase of the firing of place cells in the hippocampus of rats relative to an underlying theta-rhythm is related to the expected time when the corresponding location will be reached. Inhibitory neurons in hippocampus have also been reported to fire preferentially at specific phases of the theta cycle (see e.g. Figure S5 in (Kelemen and Fenton, 2010)). Moreover it was shown that different items that are held in working memory are preferentially encoded by neurons that fire at different phases of an underlying gamma-oscillation in the monkey prefrontal cortex (Siegel et al., 2009) (see (Pipa et al., 2009) for further evidence that such oscillations are behaviorally relevant). Phase coding was also reported in superior temporal sulcus during category representation (Turesson et al., 2012). The following result provides a theoretical foundation for such phase-specific encoding of knowledge within a framework of stochastic computation in networks of spiking neurons.

Theorem 2 (Phase-specific distributions of network states) *Let C be an arbitrary model for a network of stochastic spiking neurons as in Theorem 1. Assume now that the vector of input rates $x(t)$ has in addition to fixed components also some components that are periodic with a period L (such that each input neuron i emits a Poisson spike train with an L-periodically varying firing rate $x_i(t)$). Then the distribution of network states y converges for every phase l ($0 \leq l < L$) exponentially fast to a unique stationary distribution of network states $p_{C,l}(y|x)$ at this phase l of the periodic network input x.*

Hence, a circuit C can potentially store in each clearly separable phase l of an (externally) imposed oscillation a different, phase-specific, stationary distribution $p_{C,l}(y|x)$. Below we will address basic implications of this result in the context of two experimentally observed phenomena: stereotypical trajectories of network states and bi-stable (or multi-stable) network activity.

Figure 2.4A-D demonstrates the emergence of phase-specific distributions in a small circuit (the same as in Figure 2.3C but with only one chain) with a built-in stereotypical trajectory similar to a spatial path sequence generated by hippocampal place cell assemblies (Dragoi and Buzsaki, 2006; Gupta et al., 2012). Figure 2.4A shows a typical spike pattern in response to rhythmic background stimulation (spikes from inhibitory neurons in red). The background oscillation was implemented here for simplicity via direct rhythmic modulation of the spiking threshold of all neurons. Note that the trajectory becomes particularly often initiated at a specific phase of the rhythm (when neuronal thresholds are lowest), like in experimental data (Dragoi and Buzsaki, 2006; Gupta et al., 2012). As a result, different phases within a cycle of the rhythm become automatically associated with distinct segments of the trajectory. One can measure and visualize this effect by comparing the frequency of network states which occur at two different phases, i.e., by comparing the stationary distributions $p_{C,l}(y|x)$ for these two phases. Figure 2.4B shows a comparison of phase-specific marginal distributions on a small subnetwork of 3 neurons, demonstrating that phase-specific stationary distributions may indeed vary considerably across different phases. Convergence to the phase-specific stationary distributions $p_{C,l}(y|x)$ can be understood as the convergence of the probability of any given state to a periodic limit cycle as a function of the phase

l (illustrated in Figure 2.4C). An application of the Gelman-Rubin multivariate diagnostic suggests that this convergence takes places within a few cycles of the theta oscillation (Figure 2.4D).

Theta-paced spatial path sequences in hippocampus constitute a particularly well-studied example of phase-specific network activity(Dragoi and Buzsaki, 2006). Our theoretical framework suggests a novel interpretation of these patterns as samples from a Markov chain with a phase-dependent stationary distribution of network states induced by the theta-rhythm. A basic prediction of this interpretation is that two trajectories in successive theta cycles should exhibit significantly stronger similarities than two trajectories from randomly chosen cycles (due to inherent temporal dependencies of the Markov chain). Two trajectories from distant cycles, on the other hand, should relate to each other similarly as randomly chosen pairs of trajectories. Evidence for such an effect has been reported recently by (Gupta et al., 2012), where it was found that "sequences separated by 20 cycles approach random chance, whereas sequences separated by only a single theta cycle are more likely to be similar to each other."

The previously described theoretical framework also provides an interesting new perspective on multi-stability, a wide-spread phenomenon which has been observed in various sensory domains (Blake and Logothetis, 2002; Sterzer et al., 2009). Different authors have noted that multi-stability, both on the neuronal and perceptual level, could be understood as a side effect of sampling from a multi-modal distribution (Hoyer and Hyvärinen, 2003; Buesing et al., 2011; Gershman et al., 2012). Recent data from hippocampus suggest that oscillations, which had previously received little attention in this context, may play an important role here: (Jezek et al., 2011) found that switching between different attractors (= modes of the stationary distribution in our terminology) occurs preferentially at a specific phase during the theta cycle, whereas activity patterns within each cycle preferentially stayed in one attractor. Hence, the precise timing of switching between modes was found to be strongly tied to the theta rhythm. Such chunking of information in separate packages (theta cycles) has been proposed as an important constituent of neural syntax (Buzsáki, 2010).

In Figure 2.4E we reproduce phase-dependent switching in a simple network model of bi-stable dynamics (the same network as in Figure 2.3D) in the presence of a 6 Hz background oscillation. Indeed, we find that switching occurs preferentially at a specific phase of the oscillation (see Figure 2.4F) when the total firing rate of the network is lowest. Note that this is consistent with (Jezek et al., 2011) who found that the separation between representations in different cycles was strongest at the point of the lowest average firing rate in the population (see Figure 1b in (Jezek et al., 2011)). This phenomenon can be explained in our model by noting that the attractors are deeper during periods of high network activity. Conversely, attractors are more shallow when the population firing rate is lower, leading to an increased transition probability between attractors. If one takes a closer look at Proposition 1 and Lemma 1 in Methods one sees that this is also consistent with our theoretical framework: A lower population firing rate $\hat{\rho}$ translates into a smaller

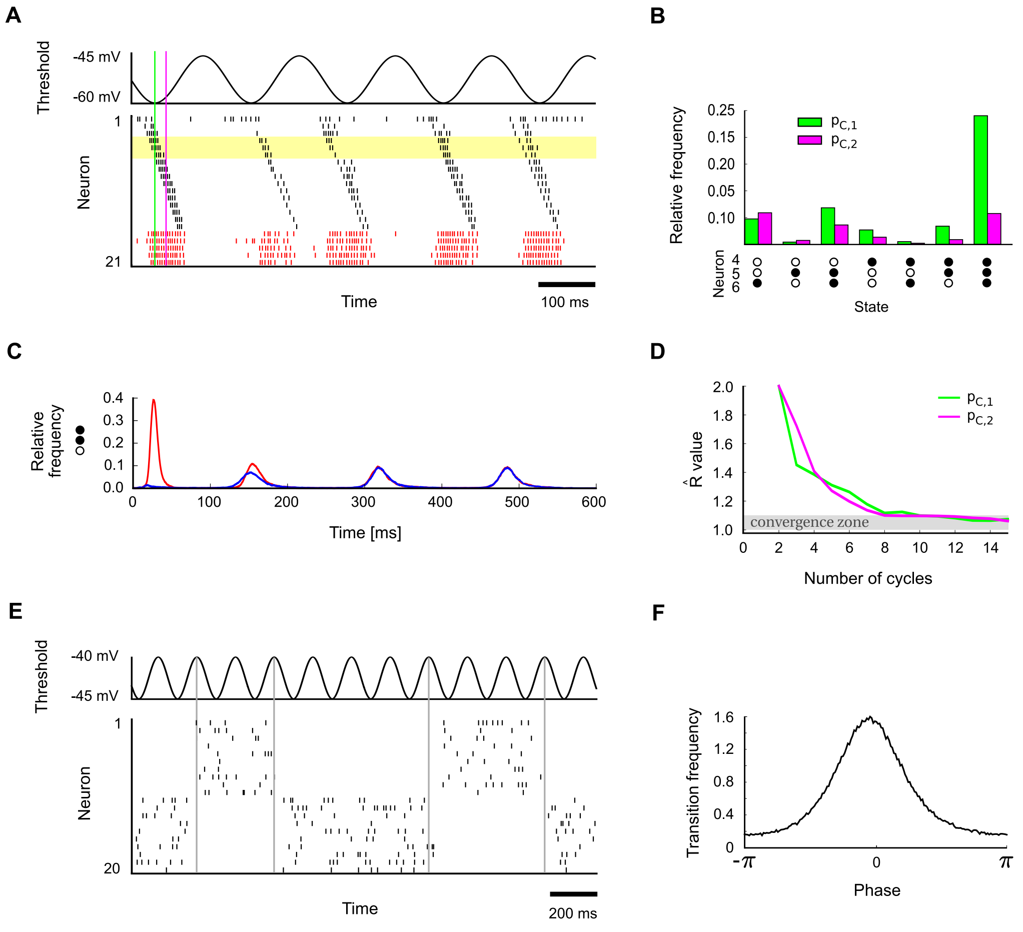

Figure 2.4: (see next page for Figure caption)

contraction factor $(1 - \epsilon^{\Theta})$, implying a tighter bound on the contraction speed of state distributions and thus higher transition probabilities to radically different states from the current (initial) network state.

Altogether, one sees that the presence of background oscillations has relevant functional implications on multi-stability. In particular, the presence of background oscillations in multi-stable networks facilitates both exploitation within a cycle and exploration across cycles: Within a cycle high firing rates force the network into one of the attractors, thereby avoiding interference with other attractors and facilitating the readout of a consistent network state. At the end of a cycle low firing rates allow the network to switch to different attractors, thereby promoting fast convergence to the stationary distribution. The rhythmic deepening and flattening of attractors and the resulting phase-specific attractor dynamics could be particularly useful for the extraction of information from the circuit if downstream networks are phase-

Figure 2.4: Emergence of phase-specific stationary distributions of network states in the presence of periodic network input. **A**. A network with a built-in stereotypical trajectory is stimulated with a 6 Hz background oscillation. The oscillation (top) is imposed on the neuronal thresholds of all neurons. The trajectories produced by the network (bottom) become automatically synchronized to the background rhythm. The yellow shading marks the three neurons for which the analysis in panels B and C was carried out. The two indicated time points (green and purple lines) mark the two phases for which the phase-specific stationary distributions are considered in panels B and D (83 ms and 103 ms into the cycle, with phase-specific distributions $p_{C,1}$ and $p_{C,2}$, respectively). **B**. The empirically measured distributions of network states are observed to differ significantly at two different phases of the oscillation (phases marked in panel A). Shown is for each phase the phase-specific marginal distribution over 3 neurons (4, 5 and 6), using simple states with $\tau =$ 10 ms. The zero state (0,0,0) is not shown. The empirical distribution for each phase ϕ was obtained from a single long run, by taking into account the network states at times $\phi, \phi + T, \phi + 2T$, etc., with cycle length $T = \frac{1}{6}s$. **C**. Illustration of convergence to phase-specific stationary distributions. Shown is the relative frequency of subnetwork state (1,1,0) on the subset of neurons 4,5 and 6 over time, when the network is started from two different initial states (red/blue). In each case, the state frequency quickly approaches a periodic limit cycle. **D**. Convergence to phase-specific stationary distributions takes place within a few cycles of the underlying oscillation. Shown is the multivariate Gelman-Rubin convergence analysis to the phase-specific stationary distribution for two different phases. **E**. Bi-stable network under the influence of a 6 Hz background oscillation. **F**. In response to the periodic stimulation, transitions between the two attractors (modes) become concentrated around a specific phase of the distribution.

locked to the same rhythm, as reported, for example, for the interactions between neurons in hippocampus and prefrontal cortex (Siapas et al., 2005).

2.8 Generation of heuristic solutions to constraint satisfaction problems

Whenever an inhibitory neuron fires, it reduces for a short while the probability of firing for its postsynaptic targets. In fact, new experimental data (Haider et al., 2013) show that inhibitory neurons impose quite powerful constraints on pyramidal cells. But also how pyramidal cells are embedded into their network environment imposes constraints on local network activity. From this perspective, the resulting firing patterns of a cortical microcircuit can be viewed as stochastically generated solutions of an immensely complex constraint satisfaction problem, that is defined both by external inputs x to the circuit and by the way each excitatory and inhibitory neuron is embedded into its circuit environment. Constraint satisfaction problems are from the computational perspective a particularly interesting class of problems, because many tasks that a brain has to solve, from the generation of a percept from unreliable and ambiguous sources to higher level tasks such as memory recall, prediction, planning, problem solving, and imagination, can be formulated as constraint satisfaction problems (Kumar, 1992). However, numerous constraint satisfaction problems are known to be NP-hard, thereby limiting the applicability of exact solution strategies. Instead, approximate or heuristic algorithms are commonly used in practice (for example evolutionary algorithms (Craenen et al., 2003)). Here we propose that networks C of spiking neurons with noise have an inherent capability to solve constraint satisfaction problems in an approximate (heuristic) manner through their stochastic dynamics. The key principle is that those network states y, which satisfy the largest number of local constraints, have the highest probability under the distribution $p_C(y|x)$. These constraints are imposed by the way each neuron of C is embedded into the circuit, and the current external input x which can selectively activate or deactivate specific constraints.

We have selected a specific constraint satisfaction problem for demonstrating the capability of networks of spiking neurons to generate rapidly approximate solutions to constraint satisfaction problems through their inherent stochastic dynamics: solving Sudoku puzzles (see Figure 2.5A). Sudoku is a well-suited example because it is complex enough to be representative for many problem solving tasks, and lends itself well to visual interpretation and presentation (but note that we do not aim to model here how humans solve Sudoku puzzles). The rules of the Sudoku game can be easily embedded into common models for cortical microcircuits as recurrent networks of Winner-Take-All (WTA) microcircuit motifs (Douglas and Martin, 2004). Each WTA motif is an ensemble of pyramidal cells (on layers 2/3 or 5/6) that are subject to lateral inhibition (see Figure 2.5B). Each pyramidal cell can in fact be part of several interlocking WTA motifs (Figure 2.5B, right).

This architecture makes it easy to impose the interlocking constraints of Sudoku (and of many other constraint satisfaction problems). Each pyramidal cell (or each local group of pyramidal cells) votes for placing a particular digit into an empty field of the grid, that is not dictated by the external input x. But this pyramidal cell is subject to the constraints that only one digit can be placed into this field, and that each digit $1, \ldots, 9$ occurs only once in each column, in each row, and in each 3x3 sub-grid. Hence each pyramidal cell is simultaneously part of four inhibitory subnetworks (WTA motifs).

A specific puzzle can be entered by providing strong input x to those neurons which represent the given numbers in a Sudoku (Figure 2.5A, left). This initiates a quite intuitive dynamics: "Clamped“ neurons start firing strongly, and as a consequence, neurons which code for conflicting digits in the same Sudoku field, the same row, column or 3x3 sub-grid, become strongly inhibited through di-synaptic inhibition. In many Sudoku fields this will lead to the inhibition of a large number of otherwise freely competing neurons, thereby greatly reducing the space of configurations generated by the network. In some cases, inhibition will immediately quieten all neurons except those associated with a single remaining digit (the only choice consistent with the givens). In the absence of competition, these uninhibited neurons will start firing along with the givens, thereby further constraining neighboring neurons. This form of inhibitory interaction therefore implicitly implements a standard strategy for solving easy Sudokus: checking for fields in which only one possibility remains. In harder Sudokus, however, this simple strategy alone would

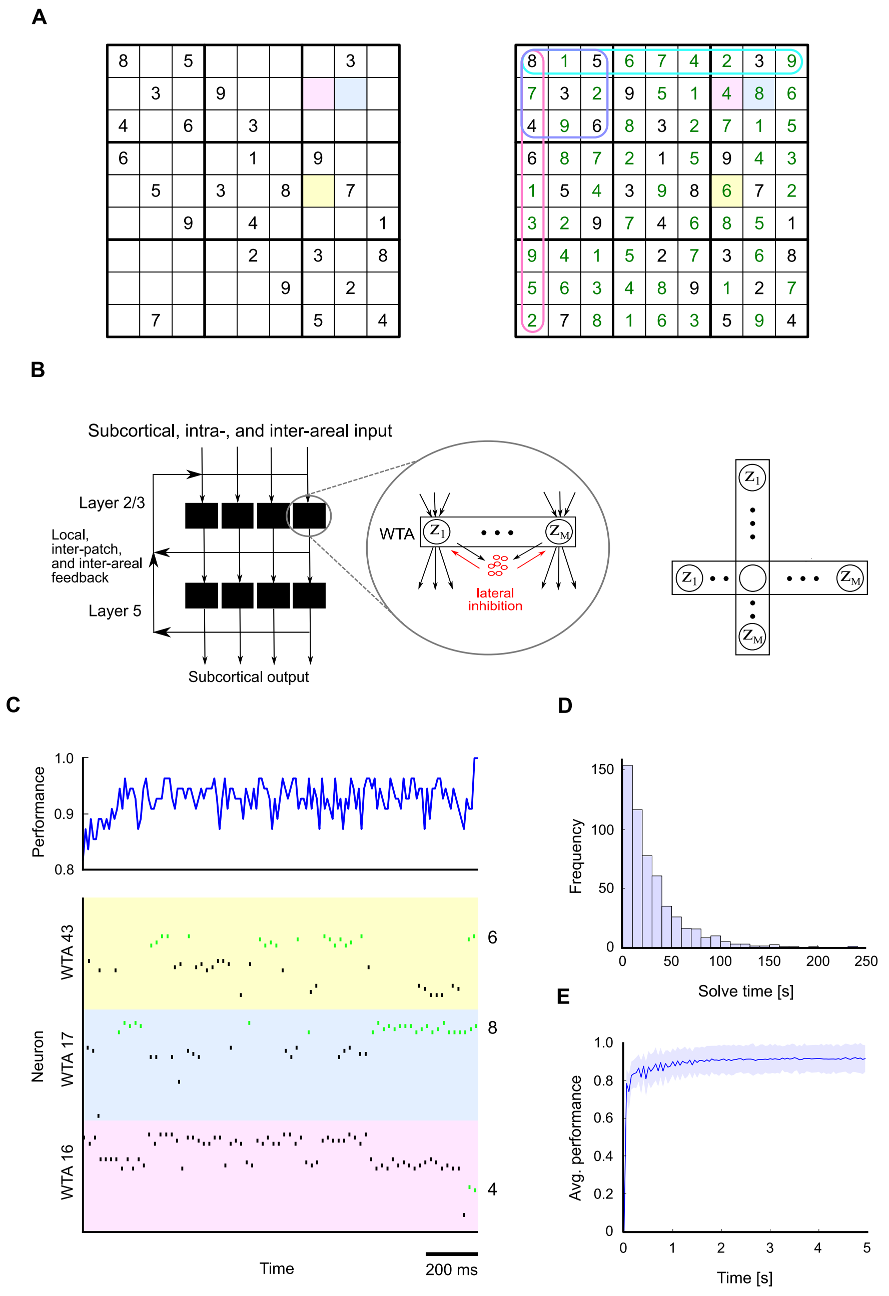

Figure 2.5: (see next page for Figure caption)

be typically insufficient, for example when several possibilities remain in all fields. In such cases, where inhibition leaves more than one possible digit open, a tentative digit will be automatically picked randomly by those neurons which happen to fire

Figure 2.5: Solving Sudoku, a constraint satisfaction problem, through structured interactions between stochastically firing excitatory and inhibitory neurons. **A**. A "hard" Sudoku puzzle with 26 given numbers (left). The solution (right) is defined uniquely by the set of givens and the additional constraints that each digit must appear only once in each row, column and 3x3 subgrid. **B**. An implementation of the constraints of the Sudoku game in a spiking neural network C consists of overlapping WTA circuits. WTA circuits are ubiquitous connection motifs in cortical circuits (Douglas and Martin, 2004). A WTA circuit can be modeled by a set of M stochastically spiking output neurons z_k that are subject to lateral inhibition (left). The same pyramidal cell can be part of several such WTA motifs (right). In the Sudoku example, each digit in a Sudoku field is associated with four pyramidal cells which vote for this digit when they emit a spike. Each such pyramidal cell participates in four WTA motifs, corresponding to the constraints that only one digit can be active in each Sudoku field, and that a digit can appear only once in each row, column and 3x3 subgrid. **C**. A typical network run is shown during the last 1500 ms before the correct solution was found to the Sudoku from panel A (the total solve time was approximately $3s$ in this run, see panel D for statistics of solve times). The network performance (fraction of cells with correct values) over time is shown at the top. The spiking activity is shown for 3 (out of the 81) WTA motifs associated with the 3 colored Sudoku fields in A and B. In each of these WTA motifs there are 36 pyramidal cells (9 digits and 4 pyramidal cells for each digit). Spikes are colored green for those neurons which code for the correct digit in each Sudoku field (6, 8 and 4 in the example). **D**. Histogram of solve times (the first time the correct solution was found) for the Sudoku from panel A. Statistics were obtained from 1000 independent runs. The sample mean is 29 s. **E**. Average network performance for this Sudoku converges quickly during the first five seconds to a value of 0.9, corresponding to 90% correctly found digits (average taken over 1000 runs; shaded area: ± 2 standard deviations). Thereafter, from all possible 9^{81} configurations the network spends most time in good approximate solutions. The correct solution occurs particularly often, on average approximately 2% of the time (not shown).

first among its competitors. This ensures that, instead of getting stuck, the network automatically explores potential configurations in situations where multiple possibilities remain. Altogether, through this combination of constraint enforcement and random exploration, those network states which violate few constraints (good approximate solutions) are visited with much higher probability than states with conflicting configurations. Hence, most time is spent in good approximate solutions. Furthermore, from all 9^{81} Sudoku configurations the solving configuration is visited in this process especially often.

Figure 2.5C shows a typical network run during the last 1.5 seconds (out of a total simulation time of approximately 3 s) before the correct solution was found to the Sudoku puzzle from Figure 2.5A. For this simulation we modeled lateral inhibition in each WTA motif by reciprocally connecting each neuron in the subnetwork to a single inhibitory neuron. For each of the 9 digits in a Sudoku field, we created an associated local group of four pyramidal cells. This can be seen in Figure 2.5C, where spike responses of pyramidal cells associated with three different Sudoku fields are shown (the three colored fields in Figure 2.5A and B). Each field has 9 possible digits, and each digit has four associated neurons. Hence, for each of the three Sudoku fields (WTA motifs), $9 \cdot 4 = 36$ neurons are shown. Spikes are colored black for those neurons which code for a wrong digit, and green for the four neurons which code for the correct digit in a Sudoku field (the correct digits in Figure 2.5C are 6, 8 and 4). The overall performance of the network (fraction of correctly solved fields) during the last 1.5 seconds before the solution is found is shown in Figure 2.5C above.

In our simulations we found that the solve time (the time until the correct solution is found for the first time) generally depends on the hardness of the Sudoku, in particular on the number of givens. For the "hard" Sudoku with 26 givens from Figure 2.5A, solve times are approximately exponentially distributed at an average of 29 seconds (Figure 2.5D). The average performance during the first five seconds of a run (obtained from 1000 independent runs) is shown in Figure 2.5E. The plot shows quick convergence to a (stationary) average performance of approximately 0.9. This demonstrates that the network spends on average most time in approximate solutions with high performance. Among these high-performance solutions, the correct solution occurs especially often (on average 2% of the time).

2.9 Discussion

A theoretical foundation for memory-based stochastic computation in cortical microcircuits

We have shown that for common noise models in cortical microcircuits, even circuits C with very detailed and diverse non-linear neurons and synapses converge exponentially fast to a stationary distribution $p_C(y|x)$ of network states y. This holds both for external inputs x that consist of Poisson spike trains of a fixed rate, and for the case where x is periodic, or generated by some Markov process with a stationary distribution. The same mathematical framework also guarantees exponentially fast convergence to a stationary distribution of *trajectories* of network states (of any fixed time length), thereby providing a theoretical foundation for understanding stochastic computations with experimentally observed stereotypical trajectories of network states. These results extend and generalize previous work in (Brémaud and Massoulié, 1996) and (Borovkov et al., 2012) in two ways. First, previous convergence proofs had been given only for networks of simplified neurons in which the (sub-threshold) neuronal integration of pre-synaptic spikes was assumed a linear process, thereby excluding the potential effects of dendritic non-linearities or synaptic short-term dynamics. Second, previous work had focused only on the case where input is provided by neurons with fixed firing rates (a special case of Theorem 1). In addition we show that these convergence proofs can be derived from a fundamental property of stochastic spiking networks, that we have formulated as the Contraction Lemma (Lemma 1 in Methods).

The stationary distribution p_C provides an attractive target for investigating the stochastic computing capabilities of data-based models C for local circuits or larger networks of neurons in the brain. In contrast to the much simpler case of Boltzmann machines with non-spiking linear neurons and symmetric synaptic connections, it is unlikely that one can attain for cortical microcircuit models C a simple analytical description of p_C. But our computer simulations have shown that this is not necessarily an obstacle for encoding salient constraints for problem solving in p_C, and for merging knowledge that is encoded in p_C with online information from external inputs x in quite fast stochastic computations. In fact, the resulting paradigm for computations in cortical microcircuits supports anytime computing, where one has no fixed computation time. Instead, first estimates of computational results can be produced almost immediately, and can be rapidly communicated to other circuits. In this way, no processor (circuit) has to idle until other processors have completed their subcomputations, thereby avoiding the arguably most critical general bottleneck of massively parallel computing systems. Instead, each microcircuit C can contribute continuously to an iterative refinement of a global computation.

Estimates for the computation time of stochastic computations

Our computer simulations for a standard cortical microcircuit model C suggest that convergence to p_C is fast enough to support knowledge extraction from this distribution p_C within a few 100 ms, i.e. within the typical computation time of higher-level brain computations. These first estimates need to be corroborated by further theoretical work and computer simulations. In particular, the relationship between the structure and dynamics of cortical microcircuits and their convergence speed merits further investigation. Furthermore, in the case where p_C is a multi-modal distribution there exists an obvious tradeoff between the convergence speed to p_C and the typical duration of staying in an "attractor" (i.e., a region of the state space which has high probability under p_C). Staying longer in an attractor obviously facilitates the readout of the result of a computation by downstream networks. A number of experimental data suggest that neuromodulators can move neural circuits (at least in the prefrontal cortex) to different points on this tradeoff curve. For example it is argued in (Durstewitz, 2006, 2009) that the activation of D_1 receptors through dopamine deepens all basins of attraction, making it harder for the network state to leave an attractor. Additional molecular mechanisms that shift the tradeoff between fast sampling (exploration) and the temporal stability of found solutions are reviewed in (Arnsten et al., 2012). Another interesting perspective on convergence speed is that slow convergence may be beneficial for certain computations in specific brain areas (especially early sensory areas). Slow convergence enlarges the time span during which the network can integrate information from non-stationary external inputs (Maass et al., 2002; Nikolic et al., 2009; Klampfl et al., 2012). In addition the initial state y_0 of a network may contain information about preceding events that are computationally useful. Those considerations suggest that there exist systematic differences between the convergence speed to p_C in different neural systems C, and that it can be modulated in at least some systems C dependent on the type of computational task that needs to be solved.

Another important issue is the tradeoff between sampling time and sampling accuracy. In high-level cognitive tasks, for example, it has been argued that "approximate and quick" sample-based decisions are often better than "accurate but slow" decisions (Vul et al., 2009; Lieder et al., 2013). Of particular interest in this context is the analysis of (Lieder et al., 2013) who studied the time-accuracy tradeoff during decision making, under the assumption that the mind performs inference akin to MCMC sampling. Due to the nature of MCMC sampling, early samples before convergence (during the burn-in period) are biased towards the initial state of the system. In the absence of time pressure, the optimal strategy is therefore to wait and collect samples for a long period of time (in theory indefinitely). In the presence of even moderate time costs, however, the optimal sampling time can be shown to be finite, a result which can provide a rational explanation of the anchoring effect in cognitive science (Lieder et al., 2013) (under time pressure people's decisions are influenced by their "initial state"). Notably, the analysis of (Lieder et al., 2013) was based on the assumption that the MCMC algorithm exhibits geometric convergence, the discrete-time equivalent to the exponential convergence speed proved in this paper for stochastic spiking networks. Applying a similar analysis to study optimal time-accuracy tradeoff points in cortical microcircuits therefore presents a promising avenue for future research.

Which probability distributions can be encoded as a stationary distribution of some neural circuit?

It had been shown in (Buesing et al., 2011) and (Pecevski et al., 2011) that, under certain assumptions on the neuron models and circuit structure, in principle every joint distribution p over discrete-valued random variables can be represented as a stationary distribution p_C of some network C of spiking neurons. Forthcoming unpublished results suggest that such internal representations of a given distribution p can even be learned from examples drawn from p. This will provide a first step towards understanding how the stationary distribution p_C of a microcircuit can be adapted through various plasticity processes to encode salient constraints, successful solution strategies (rules), and other types of knowledge. This research direction promises to become especially interesting if one takes into account that knowledge can not only be encoded in the stationary distribution of network states, but also in the simultaneously existing stationary distribution of trajectories of network states.

Relationship to attractor networks and transients between attractors

Attractor neural networks (Hopfield, 1982) were originally deterministic computational models, where gradient descent leads the network from some given initial state y_0 (the input for the computation) to the lowest point of the attractor (the output of the computation) in whose basis of attraction y_0 lies. The computational capability of an attractor neural network is substantially larger if its attractor landscape can be reconfigured on the fly by external input x, as in (Hopfield and Tank,

1986) and in the Sudoku example of this article. This usually requires that the attractors are not programmed directly into the network parameters, but emerge from some more general computational principles (e.g. constraint satisfaction). Attractor neural networks gain additional computational capability if there is some noise in the system (Rolls and Deco, 2010). This enables the network to leave after a while suboptimal solutions (Durstewitz and Deco, 2008). Alternative modeling frameworks for the transient dynamics of neural systems are provided by the liquid computing model (Maass et al., 2002), and on a more abstract level by sequences of metastable states in dynamical systems (Rabinovich et al., 2008). Here we propose to view both transient and attractor dynamics of complex data-based circuits C from the perspective of probabilistic inference, in particular as neural sampling (Buesing et al., 2011) (or more abstractly: as MCMC sampling) from their inherent probability distribution p_C over network states (or trajectories of network states), that serves as the knowledge base of these neural systems.

A new computational framework for analyzing brain activity

We had focused in our computer simulations on the investigation of the stationary distribution p_C for models C of cortical microcircuits. But the results of Theorem 1 and Theorem 2 are of course much more general, and in principle apply to models C for networks of neurons in the whole brain (Sporns, 2011). This perspective suggests understanding spontaneous brain activity (see (Raichle, 2010)) as sampling from this global distribution in the absence of external input, and brain computations with external inputs x as sampling of brain states from conditional distribution $p_C(y|x)$, thereby merging the knowledge base p_C of the brain with incoming new information x. This computational framework could in principle explain how the brain can merge both types of information in such seemingly effortless manner, a capability that can only partially be reproduced in artificial devices with current technology. Large-scale computer simulations will be needed to test the viability of this hypothesis, in particular the relationship between the known global structure of the brain network C and properties of its stationary distribution p_C, and the convergence speed to p_C. Possibly the brain uses an important trick to speed up convergence during brain-wide sampling, for example by sampling during any concrete brain computation only from a subnetwork C' of C: those brain areas that control variables that are relevant for this computation. Functional connectivity would be explained from this perspective as opening of communication channels that support sampling from the (marginal) joint distribution of those variables that are stored within the functionally connected brain areas. Structured spontaneous brain activity (Raichle, 2010) would then receive a functional interpretation in terms of updating these marginal joint distributions on the basis of newly acquired knowledge.

Stochastic solutions of constraint satisfaction problems as a paradigm for higher level brain computation

A surprisingly large number of computational tasks that the brain has to solve, from the formation of a percept from multi-modal ambiguous sensory cues, to prediction, imagination, motor planning, rule learning, problem solving, and memory recall, have the form of constraint satisfaction problems: A global solution is needed that satisfies all or most of a set of soft or hard constraints. However, this characterization per se does not help us to understand how the brain can solve these tasks, because many constraint satisfaction problems are computationally very demanding (in fact, often NP-hard (Garey and Johnson, 1979)), even for a fast digital computer. In the Sudoku example we have shown that the inherent stochastic dynamics of cortical microcircuits provides a surprisingly simple method for generating *heuristic* solutions to constraint satisfaction problems. This is insofar remarkable, as this computational organization does not require that specific algorithms are programmed into the network for solving specific types of such problems (as it is for example needed for solving Sudoku puzzles according to the ACT-R approach (Qin et al., 2012)). Rather, it suffices that salient constraints are encoded into the network (e.g. through learning) in such a way that they make certain firing patterns of a subset of neurons more or less likely.

Future work will need to investigate whether and how this approach can be scaled up to larger instances of NP-complete constraint satisfaction problems. For example, it will be interesting to see whether stochastic networks of spiking neurons can also efficiently generate heuristic solutions to energy minimization problems (Boykov et al., 2001) arising in visual processing.

Furthermore, additional research is needed to address suitable readout mechanisms that stabilize and evaluate promising candidate solutions (see (Arnsten et al., 2012) for an experimentally supported mechanism that might contribute to this function). This is an important issue since, in its current form, the network will simply continue the stochastic exploration of heuristic solutions even after it has found the optimal solution. Therefore, in the absence of additional mechanisms the network is not able to hold on to (or store) previously found (near-)optimal solutions. To solve this issue one could consider, for example, one or several networks $C_1, \ldots, C_i$ which generate in parallel heuristic solutions to a given problem. The output of these networks could then be further processed and integrated by a readout network C_{i+1} which attempts to extract a MAP solution, for example by adopting a solution from some C_j only if it has higher value than the currently stored state. Hence, the sampling networks $C_1, \ldots, C_i$ would have stationary distributions $p_{C_j}(y|x)$ which encourage exploration and broadly assign probability to many different heuristic solutions, whereas the readout network would ideally exhibit a sharply peaked stationary distribution at the global optimum of the constraint satisfaction problem. Studying the feasibility of this approach requires further research.

Relationship to models for probabilistic inference in cognitive science

A substantial number of behavioral studies in cognitive science (see e.g. (Griffiths and Tenenbaum, 2006; Vul and Pashler, 2008; Denison et al., 2009; Gershman et al., 2012; Tenenbaum et al., 2011)) have arrived at the conclusion that several of the previously discussed higher level mental operations are implemented through probabilistic inference. Some of the underlying data also suggest that probabilistic inference is implemented in the brain through some form of sampling (rather than through arithmetical approaches such as belief propagation (Koller and Friedman, 2009)). But according to (Tenenbaum et al., 2011): "The key research questions are as follows: What approximate algorithms does the mind use, how do they relate to engineering approximations in probabilistic AI, and how are they implemented in neural circuits?" This article contributes to these fascinating questions by providing a rigorous theoretical foundation for the hypothesis that neural circuits in the brain represent complex probability distributions $p_C(y|x)$ through sampling. In addition, we have provided evidence that this form of sampling in cortical microcircuits may be fast enough to facilitate the approximate estimation of marginals or marginal MAP assignments, which commonly appear in real-world inference tasks, within a few 100 ms. A major challenge for future work will be to understand also neuronal plasticity on the implementation level from this perspective. For example, how can prior knowledge be acquired and integrated into the stationary distribution $p_C(y|x)$ of a realistic circuit C (featuring short-term plasticity, dendritic processing, etc.) in an autonomous fashion, and in a manner consistent with statistically optimal learning (Fiser et al., 2010)?

Long-term plasticity and other slower features of network dynamics

In biological networks it is reasonable to assume that the network dynamics unfolds on a continuum of time scales from milliseconds to days. Our goal in this article was to focus on stochastic computations on shorter time scales, between a few milliseconds to seconds. To this end we assumed that there exists a clear separation of time scales between fast and slow dynamical network features, thus allowing us to exclude the effect of slower dynamical processes such as long-term plasticity of synaptic weights during these shorter time scales. In network models and experimental setups where slower processes significantly influence (or interfere with) the dynamics on shorter time scales, it would make sense to extend the concept of a stationary distribution to include, for example, also the synaptic parameters as random variables. A first step in this direction has been made for neurons with linear sub-threshold dynamics and discretized synapses in (Borovkov et al., 2012).

Deterministic network models and chaos

Deterministic network models such as leaky integrate-and-fire neurons without noise (no external background noise, no synaptic vesicle noise and no channel noise) violate the assumptions of Theorem 1 and 2. Furthermore, although realistic neurons are known to possess various noise sources, the theoretical assumptions could in principle still fail if the network is not *sufficiently* stochastic: this would happen, for example, if there exists some strong input (within the limits of typical input activity) which entirely overrules the noise, leading to a firing probability 1 in some time interval $[t, t + \delta t)$ during the network simulation. Such deterministic behavior would correspond to the instantaneous firing rate of a stochastic neuron becoming infinite at some point during that interval (in violation of assumption A2, see Methods: Scope of theoretical results). From an empirical perspective, a simple necessary condition for sufficient stochasticity is the presence of trial-to-trial variability for each single spike produced by a network. Consider, for example, the spike times generated by a specific neuron in a network simulation, in response to some fixed input spike train. If there exists a spike which always occurs at the exact same time during multiple repetitions of this experiment starting from identical initial states, then the assumptions of Theorem 1 and 2 are obviously violated.

For deterministic (or insufficiently stochastic) networks the question arises whether convergence to a unique stationary distribution may still occur under appropriate conditions, perhaps in some modified sense. Notably, it has been recently observed that deterministic networks may indeed lead to apparently stochastic spiking activity (Churchland and Abbott, 2012; Litwin-Kumar and Doiron, 2012). This apparent stochasticity was linked to chaotic spiking dynamics. This suggests that chaos may act as a substitute for "real" noise in deterministic networks (similar to pseudo random-number generators emulating true randomness): Chaotic systems are sensitive to small perturbances in initial conditions, and may thus exponentially amplify otherwise insignificant noise sources such as ubiquitous thermal noise (Clarke, 2012). Thus, chaos could play an important role in both emulating and amplifying stochasticity on the network level.

(Litwin-Kumar and Doiron, 2012) focused their analysis of stochasticity on firing rate fluctuations and spiking irregularity, and it remains unclear whether these networks would still appear stochastic if one takes into account full network states (as in this article). The Gelman-Rubin convergence analysis of population activity proposed in this paper could be applied to provide some insight into this question. A more thorough investigation of chaos in the context of our results would also call for a rigorous theoretical analysis of ergodic properties of chaotic spiking networks.

Further experimentally testable predictions

Our theoretical results demonstrate that every neural system C has a stationary distribution $p_C(y|x)$ of network states y. This can be tested experimentally, for various behavioral regimes and external inputs x. A first step in this direction has already been carried out in (Berkes et al., 2011) (see also the discussion in (Okun et al., 2012)). The hypothesis that p_C serves (for "neutral" external inputs x) as a prior for probabilistic inference through sampling suggests that p_C is constantly modified through prior experience (see (Zhang et al., 2012; Xu et al., 2012) for first results) and learning (see (Lewis et al., 2009) for fMRI data).

Our Theorem 2 suggests in addition that neural systems C that have a prominent rhythm (such as for example the theta oscillation in the hippocampus) are able to store *several* stationary distributions $p_{C,l}$ of network states, one for each clearly separable phase l of this rhythm. It has already been shown in a qualitative manner that in some behavioral situations certain states y appear with substantially high probability at specific phases l of the rhythm (see e.g. (Harris et al., 2003; Buzsaki, 2009; Siegel et al., 2009; Gupta et al., 2012; Turesson et al., 2012)). But a systematic experimental analysis of phase-dependent distributions of network states in the style of (Berkes et al., 2011) is missing.

Our Theorem 1 predicts in addition that a generic neural circuit C also has a stationary distribution over *trajectories* of network states. The existence of stereotypical trajectories of network states in the awake brain has been frequently reported (see e.g. (Abeles et al., 1995; Jones et al., 2007; Luczak et al., 2007; Zhang et al., 2012)). But a statistical analysis of the distribution of such trajectories, especially also during spontaneous activity, is missing. Of particular interest is the relationship between the distribution of trajectories and the stationary distribution of (simple) network states. Do some network states y typically have a high probability because they occur in some high probability trajectory? And how does the distribution of trajectories change during learning?

The model for problem solving that we have presented in Figure 2.5 suggests that external constraints have a significant and characteristic impact on the structure of the stationary distribution p_C, by reducing the probability of network states which are inconsistent with the current constraints x. In principle, this could be analyzed experimentally. In addition, this model suggests that there may be special mechanisms that prolong the time span during which a neural system C stays in a network state y with high probability under $p_C(y|x)$, in order to support a readout of y by downstream networks. These mechanisms need to be revealed through experiments.

New ideas for neuromorphic computation

The Sudoku example has shown that networks of spiking neurons with noise are in principle able to carry out quite complex computations. The constraints of many other demanding constraint satisfaction problems, in fact even of many NP-complete problems, can be encoded quite easily into circuit motifs composed of excitatory and inhibitory spiking neurons, and can be solved through the inherent stochastic dynamics of the network. This provides new computational paradigms and applications for various energy-efficient implementations of networks of spiking neurons in neuromorphic hardware, provided they can be equipped with sufficient amounts of noise. In particular, our results suggest that attractive computational properties of Boltzmann machines can be ported into spike-based hardware. These novel stochastic computing paradigms may also become of interest for other types of innovative computer hardware: Computer technology is approaching during the coming decade the molecular scale, where noise is abundantly available (whether one wants it or not) and it becomes inefficient to push through traditional deterministic computing paradigms.

Conclusion

The results of this article show that stochastic computation provides an attractive framework for the investigation of computational properties of cortical microcircuits, and of networks of microcircuits that form larger neural systems. In particular it provides a new perspective for relating the structure and dynamics of neural circuits to their computational properties. In addition, it suggests a new way of understanding the organization of brain computations, and how they are modified through learning.

2.10 Acknowledgments

This chapter is based on a joint work with Stefan Habenschuss (SH) and Wolfgang Maass (WM) that was published 2013 in PLOS Computational Biology ("Stochastic Computations in Cortical Microcircuit Models", I am joint first author with SH). The experimental setup and organization of figures was developed in close collaboration of all authors, by ZJ, SH and WM. The theoretical analysis of the paper was provided by SH. I developed the software and simulation framework required for large-scale simulation of the stochastic cortical column model, as well as the analysis tools for studying the convergence properties of networks of spiking neurons (the Gelman-Rubin analysis of one-dimensional and high-dimensional spike trains). The simulation and computational convergence analysis of the cortical column model and other microcircuits for Figure 2.2-2.4 as well as the simulations and analysis for Figure 2.5 was done by me. The main text was written by WM and SH, and the Methods were written by ZJ and SH. The figures were designed by ZJ, SH and WM.

Chapter 3

Networks of Spiking Neurons with Noise can Solve Hard Computational Problems

Contents

Networks of neurons in the brain compute and communicate very differently from transistors in digital computers: with unsynchronized short pulses, called action potentials or spikes. But it has remained unknown how difficult computational problems could be solved in this way. We present here new principles of spike-based computation with noise that enable networks of spiking neurons to carry out a very efficient stochastic search in high-dimensional spaces, thereby producing fast approximate solutions to hard computational problems such as logical inference (SATISFIABILITY) and planning (TRAVELING SALESMAN PROBLEM). The underlying computational theory that we present also suggests new methods for organizing massively parallel computations in novel energy-efficient but noisy computing hardware.

3.1 Introduction

Despite the astonishing advancements of digital computing in the past decades, the human brain is still considered the most powerful, versatile and "intelligent" computing device. Most of the remarkable mental faculties of humans, from imagination, prediction, and creative problem solving to abstract thought, are unrivaled by the most powerful supercomputers. This is achieved by the brain with only ~ 25 Watt of energy consumption (Kandel et al., 1991) (several order of magnitude less than its most powerful digital counterparts), without the need for a global clock through inherently asynchronous communication (Gerstner and Kistler, 2002), and in spite of powerful noise sources introducing random variability at virtually every step of neural computation, from spike generation, to action potential propagation, to synaptic transmission (Faisal et al., 2008). Despite decades of research, however, it is still largely unknown how complex computations, beyond mere sensory processing, could be implemented in neural circuits on the basis of such noisy asynchronous computing units.

In this article we present a theoretical framework and four new principles for circuit design with spiking neurons that demonstrate how the inherent stochasticity and asynchronous dynamics of neural circuits can be systematically exploited to solve hard computational problems. We report that the application of this new theoretical framework leads to a qualitative jump in the computational capabilities of networks of spiking neurons.

3.2 New design principles for spike-based computation

A spiking neuron responds to stimulation by emitting short pulses, called action potentials or spikes. Spikes occur asynchronously (in continuous time) and are communicated to other neurons via inhibitory or excitatory synaptic connections (Figure 3.1A, top). Biological spiking neurons are inherently noisy (Faisal et al., 2008). We model the stochastic spiking behavior of a neuron k via an instantaneous firing probability (or firing rate), $\rho_k(t)$,

$$\rho_k(t) = \frac{1}{\tau}\ \exp(u_k(t)) \quad , \tag{3.1}$$

the magnitude of which depends on the current *membrane potential* $u_k(t)$ of the neuron. The membrane potential is defined as the weighted sum of the neuron's inputs,

$$u_k(t) = b_k + \sum_l w_{kl}\, x_k(t) \quad . \tag{3.2}$$

The additional bias term b_k represents the intrinsic excitability of neuron k. After each emitted spike, neuron k enters a refractory period of length τ before it can re-spike. A spike by neuron k is transmitted via synaptic connections to all post-synaptic neurons receiving input from neuron k. The effect of a spike on a post-synaptic neuron l, the so-called post-synaptic potential (PSP), is short-lived and can be either inhibitory or excitatory, depending on the sign of the synaptic weight w_{lk}. In general, PSPs can assume complex shapes and the effective duration of a PSP may depend on various dynamically changing factors. Here we assume for mathematical tractability a rectangular shape with a fixed length $\tau = 10$ms as shown in Figure 3.1A, such that $x_k(t) = 1$ if a spike occurred within $(t - \tau, t]$, and $x_k(t) = 0$ otherwise.

The *network state* at time t is defined as the vector of neural states $\mathbf{x}(t) = (x_1(t), x_2(t), \ldots, x_N(t))$, i.e. only those neurons are set to 1 which fired recently

(Figure 3.1A, middle). Due to network connectivity some network states $\mathbf{x}(t)$ will naturally occur more often on average than others. Consider the distribution of network states that can be measured empirically by observing network activity over a long period (Figure 3.1A, bottom). In general networks of spiking neurons, the resulting long-term distribution will depend on the initial state of the network at the beginning of the experiment. If the network is sufficiently stochastic, however, the distribution over network states becomes independent of the initial state. In networks composed of stochastic neurons of the type (3.1), a sufficient condition for this to occur is that all excitatory weights in the network are bounded. Note that when such a *unique stationary distribution* $p(\mathbf{x})$ exists, it reflects which network states can be most likely observed after the vanishing of transients (after convergence to equilibrium). In analogy with statistical physics, we define the energy function of a sufficiently stochastic network as $E(\mathbf{x}) = -\log p(\mathbf{x}) + \text{const}$. According to this definition low energy states correspond to likely network states after convergence to equilibrium.

We present a set of four new principles of circuit design with spiking neurons. *Principle 1* – the foundation of our framework – states that one should add sufficient stochasticity to a (possibly otherwise deterministic) network with spiking neurons so that the network has a *unique stationary distribution* $p(\mathbf{x})$ of network states (Figure 3.1A). For stochastic neurons (3.1) adding further noise is obviously not required. In order to use such a stochastic network to solve a given computational task, the circuit should then be constructed in such a manner that the stationary distribution of network states $p(\mathbf{x})$ assumes especially high values for circuit states that encode good solutions to the computational task. Equivalently, the energy function $E(\mathbf{x}) = -\log p(\mathbf{x}) + \text{const}$ of a circuit should be particularly low for states $\mathbf{x}$ representing solutions to the problem.

Principle 2 states that the energy function $E(\mathbf{x})$ over a set of *principal neurons* can be systematically shaped through the use of a few auxiliary circuit motifs (Figure 3.1B). In particular we present two circuit motifs, the winner-take-all (WTA) and the OR motif, that can be used to impose powerful higher-order constraints on the activity of principal neurons in order to encode a variety of hard computational problems. The WTA circuit motif, applied to some set of principal neurons, increases the energy (decreases the probability) of all network states where not exactly (i.e. not more and not less than) one principal neuron in the WTA circuit is active. The OR motif increases the energy of all states where none of the involved principal neurons is active. A third way of systematically shaping energies is to add bi-directional and symmetric synaptic connections between two neurons k and l which either increase or decrease the energy of network states where both involved neurons are active, $x_k = 1, x_l = 1$ (via inhibitory or excitatory connections, respectively).

One important use of the WTA motif is to represent discrete random variables (RV): Consider a set of K principal neurons, where each neuron codes for one out of K possible values of a discrete RV. Then, by applying the WTA motif to these neurons one can ensure that most of the time only one of the principal neurons

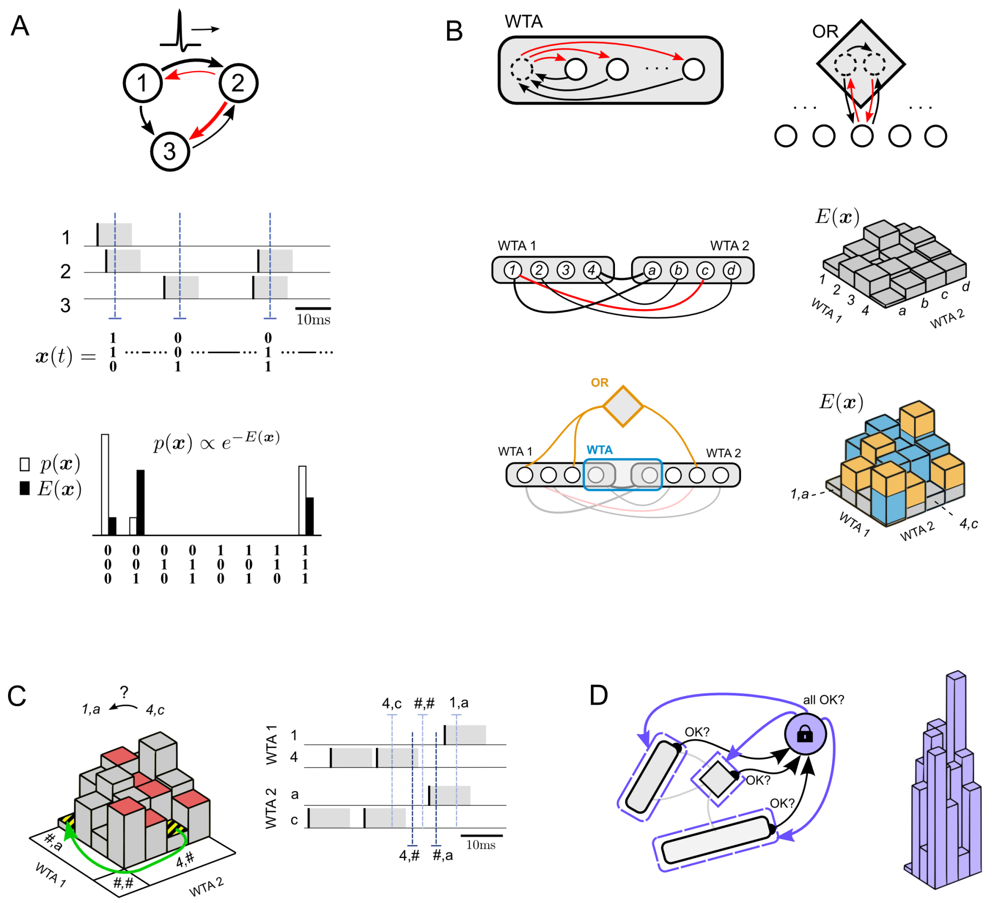

Figure 3.1: (see next page for Figure caption)

is active and, as a result, most of the time the RV has a well-defined value which can be derived from the current network state. Nevertheless, it may happen that for some short period of time none (or more than one) of the K neurons is active. During that period, the value of the represented RV is then considered undefined. The WTA motif ensures that such periods are very brief when they occur. Various computational problems, including 3-SAT and TSP, can be represented in terms of a number of discrete RVs with a finite state space. Hence, the representation of discrete RVs by WTA circuits is the foundation for the encoding of many problems. Specific constraints of a problem can be implemented by adding symmetric synaptic connections among principal neurons, as well as connecting additional WTA and OR circuits to different subsets of principal neurons (Figure 3.1B).

The systematic design of complex energy landscapes composed of large numbers

Figure 3.1: Four new principles of circuit design with spiking neurons. **A**. Unlike transistors in a digital computer, spiking neurons communicate via brief asynchronous pulses called spikes (inhibitory/excitatory synaptic connections shown in red/black, connection line width represents synaptic strength). To study the collective behavior of spiking networks, we consider joint network states $\mathbf{x}(t)$, a binary vector where all neurons which fired recently (within the last 10ms) are assigned 1, over time t. The stationary distribution of network states, $p(\mathbf{x})$, reflects which network states can be likely observed after transients have vanished. We propose that $p(\mathbf{x})$, or equivalently the energy function $E(\mathbf{x}) = -\log p(\mathbf{x}) + \text{const}$, represents the computational output of a spiking network with noise (Principle 1). **B**. The energy function $p(\mathbf{x})$ over a set of *principal neurons* (white) can be shaped in a systematic manner by adding circuit motifs (shown in gray). Top row: Two motifs winner-take-all (WTA) and OR. These circuit motifs constrain activity patterns such that most of the time exactly one (WTA) or at least one (OR) in the set of connected principal neurons is active. Middle/bottom row: Example network consisting of two WTA circuits, each representing a discrete random variable (RV) with four possible values (middle left). Symmetric synaptic connections (only some shown) can be used to modulate the energies of different joint assignments to these two variables (middle right). Additional auxiliary circuits can be added (bottom left) to shape the energy landscape in more complex ways (bottom right). Those network states which violate the fewest of the imposed constraints have lowest energy (such as the highlighted states $1, a$ and $4, c$). The energy contributions of different auxiliary circuits sum up linearly. This facilitates the construction of complex energy landscapes through repeated use of simple circuit motifs (Principle 2). **C**. Stochastic search for low energy states is facilitated by the asymmetry of spike-based signaling. Direct transitions between the two low-energy states $1, a$ and $4, c$ are blocked due to high energy intermediate states (marked in red). An alternative route goes over a series of states where the value of one or both discrete RVs is briefly undefined (marked by #). Such "exploratory" periods of undefined RVs occur particularly frequently (and briefly) in a spiked-based communication scheme (Principle 3). **D**. In contrast to traditional stochastic search algorithms, the search process in a physical implementation cannot be "stopped" when a satisfactory solution has been found. Instead, internal temperature control is proposed as a principled alternative for high-speed computing systems: each circuit motif detects and reports to a global *lock-in* neuron whether its constraint is currently met (*OK* signals). As soon as all (or most) constraints are met the global *lock-in* neuron activates a set of additional circuit motifs which sharpen the existing energy landscape (right). This leads to a global reduction of the *temperature* of the circuit, thereby reducing exploration and forcing the network to lock into the locally best solution (Principle 4).

of circuit elements calls for an understanding of how circuit elements interact with each other. For example, what is the joint effect of two auxiliary circuit motifs which are operating on an overlapping set of principal neurons? Notably, one can show theoretically that under certain idealized conditions the energy contributions of circuit motifs sum up linearly. This occurs in particular when a) integration of synaptic inputs is linear as in (3.2) and b) the total instantaneous synaptic drive $\Delta u_{k,i}$ onto a principal neuron k due to the presence of a circuit motif C_i is given by,

$$\Delta u_{k,i}(t) = \Delta E_i(\{x_k = 0, \mathbf{x}_{\backslash k}(t)\}) - \Delta E_i(\{x_k = 1, \mathbf{x}_{\backslash k}(t)\}) \qquad (3.3)$$

at any point in time during circuit operation. Since the design of large-scale circuits is greatly facilitated by linear compositionality of individual elements, (3.3) can be seen as the idealized reference functionality of a circuit motif with energy contribution $\Delta E_i(\mathbf{x})$. In practice, such reference can be used to guide circuit design; the WTA and OR circuit motifs shown in Figure 3.1B were specifically designed to approximate (3.3). A basic consequence is that the relative energy contribution of any given circuit motif is practically independent of the presence of other circuit motifs. This allows one to apply all three types of circuit motifs on different subsets of principal neurons in a combinatorial fashion to generate a rich diversity of energy landscapes in a highly controlled fashion. As a result, energy landscapes of important computational problems can be constructed in a relatively straightforward manner.

Principles 1 and 2 lay the foundation for implementing *massively parallel local search* for low energy states in complex energy landscapes through the intrinsic dynamics of spiking networks with noise.[1] *Principle 3* states that this search process is facilitated by the inherent asymmetry of spike-based signaling (Figure 3.1C). This is because asymmetric signaling, where a spike is followed by a fixed period of *on*-time whereas *off*-times are subject to random variation, alleviates one of the practical issues of stochastic local search: the presence of deep local minima in which the search process gets stuck. The benefits of spike-based signaling are particularly visible in conjunction with the WTA circuit motif: suppose that some WTA circuit represents a discrete RV in a computational problem, and each principal neuron in the WTA circuit represents one possible value of that variable, as described above. Then the WTA circuit motif permits that sometimes (randomly and briefly) none of the principal neurons in the WTA is active, and hence the value of the RV is temporarily undefined (Figure 3.1C, right). When this occurs, most of the time it has no lasting effect because the previously defined state is quickly restored. But when the transition to an undefined RV state occurs in two or more WTA circuits at approximately the same time, the principal neurons in these circuits are momentarily given the opportunity to reconfigure their states from scratch and explore radically different state configurations. Although such transitions to brief periods of undefined RVs occur rarely, their frequency is greatly enhanced through the asymmetry of spike-based signaling because spikes have a fixed *on* time and neurons are therefore bound to switch to an *off* state on a much more regular basis (but more briefly) than expected from the energy landscape alone. Indeed one can show theoretically that, compared with a symmetrized system which samples from

[1]The search is *local* in the sense that when the network moves from state $\mathbf{x}$ to some other state $\mathbf{x}'$ this always occurs through a series of small changes (a series of individual neurons turning on and off). In addition, the search process is parallel because state changes occur in a highly distributed manner across the network, thus supporting the efficient exploitation of independent substructures in the energy landscape.

the same stationary distribution $p(\mathbf{x})$[2], the stochastic dynamics of noisy spiking networks is considerably more explorative due to an increased frequency of state transitions which gap large energy barriers.

Finally, *Principle 4* proposes internal, rather than the traditional external, temperature control for regulating stochastic search as part of the spike-based computing architecture (Figure 3.1D). Temperature control, i.e. the strategic modulation of the energy landscape according to $E_T(\mathbf{x}) = E(\mathbf{x})/T$ with some temperature T, is an essential ingredient of many stochastic search algorithms (Kirkpatrick et al., 1983; Michalewicz and Fogel, 2000). High temperatures T generally lead to a flattening of the energy landscape and increased exploration, whereas low temperatures T correspond to a sharpening of the landscape and increased exploitation and drive towards (local) energy minima. We propose an internal temperature control mechanism capable of a) automatically detecting *in-situ* when an acceptable solution has been reached and b) reducing temperature once such a solution has been found, leading to decreased exploration and a quasi lock-in effect. The key advantage of such internal temperature control is that solutions are automatically detected and stabilized which facilitates readout. In particular, in the absence of stabilization the network may visit solution states arbitrarily briefly and transiently, and thus solutions may be easily missed by a sloppy readout. In the presence of a lock-in mechanism, on the other hand, good solutions are maintained and it therefore suffices to check for solutions at irregular intervals. As a practical consequence, the readout logic may run on a much slower timescale than the spiking dynamics, which may be particularly beneficial in the context of high-speed neuromorphic simulation of spiking networks.

3.3 Solving 3-SAT problems

To demonstrate these principles we applied the proposed framework to hard logical inference problems. As problem instances, hard random 3-SAT problems with a clauses-to-variables ratio 4.3 near the phase transition (Biere, 2009) are considered (Figure 3.2). Each clause (constraint) of a 3-SAT formula consists of three literals, where a literal is either a variable X_i or its negation $\overline{X_i}$ (Figure 3.2A). A clause is fulfilled if at least one out of the three literals is true. The goal is to find an assignment to the variables which satisfies all clauses.[3] For hard problems this typically means finding one out of a handful of solutions in an astronomically large search space of $2^{\#vars}$ possible assignments. Using Principles 1 and 2, any 3-SAT problem can be encoded in a straightforward manner in a spiking network by representing each boolean variable by two neurons ($X_i/\overline{X_i}$) and a WTA circuit, and adding for each clause an OR circuit which is linked to the three literals of the

[2]A symmetrized non-spike-based system which samples from the same stationary distribution $p(\mathbf{x})$ but in which transitions from *on* to *off* occur in the same stochastic manner as transitions from *off* to *on*: a continuous-time variant of Gibbs sampling.

[3]This is the search variant of the satisfiability problem (Biere, 2009).

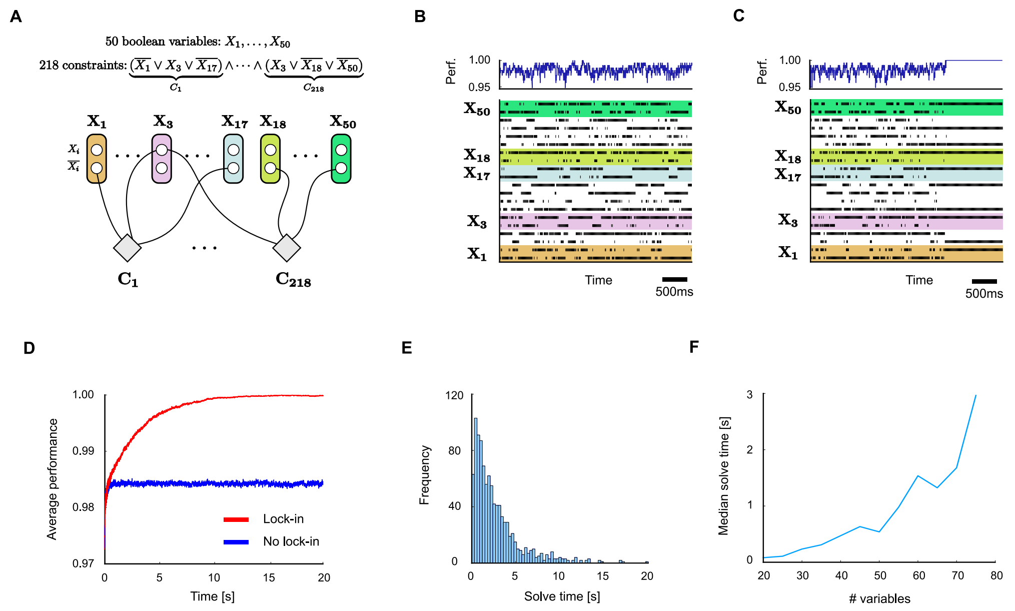

Figure 3.2: Solving logical inference (SATISFIABILITY) problems. **A**. A 3-SAT problem consisting of 50 boolean variables and 218 constraints (*clauses*). Each clause is fulfilled if at least one out of the three literals is true. For the sketched problem only one assignment to the variables exists which fulfills all clauses – the goal is to find this solution. A network implementation based on Principles 1 and 2 is shown below. **B.** Example network run during the first 4 seconds of network time (bottom: spike trains of neurons of selected WTA circuits, top: performance of current network solution over time). **C.** Example run in the presence of an additional *lock-in* circuit: when a solution is found it is automatically detected and stabilized. **D**. Average network performance over time in the absence and presence of *lock-in*. **E**. Distribution of wait times, i.e. times until the solution is found for the first time. **F.** As expected for hard NP-complete problem instances, simulation results suggest an exponential increase in wait times for hard random 3-SAT problems with increasing problem size.

clause (Figure 3.2A). Note that the network states representing correct solutions to the problem violate the fewest circuit constraints and are therefore assigned particularly low energies (and high probability) in the energy landscape. In simulations it is observed that networks constructed in this manner quickly generate good approximate solutions to the encoded 3-SAT problem, i.e. assignments which meet many but not all constraints (Figure 3.2B). For hard problems with 50 variables a correct solution which meets all constraints is usually found for the first time after a few 100ms to a few seconds of network time (Figure 3.2B,E). Without a *lock-in* mechanism the network then continues to search for other potential solutions to the problem (Figure 3.2B). When a *lock-in* circuit is added, solutions are automatically maintained and stabilized (Figure 3.2C). As a result, the average performance of the network is considerably enhanced with *lock-in*. Regarding scalability on hard random 3-SAT problems, simulations suggest that typical wait times (the time until a solution is found by the network for the first time) scale exponentially with problem size (Figure 3.2E), as expected for hard NP-complete problem instances.

3.4 Generating solutions to Traveling Salesman Problems

We further applied the proposed framework to planning problems (Figure 3.3), in particular instances of the Traveling Salesman Problem (TSP) (Gutin and Punnen, 2002). Given a list of cities and the traveling costs for going from any city i to any other city j, the goal is to find the least costly (the "shortest") round-trip route that visits each city exactly once. The problem can be encoded in a spiking network by representing each step s in the trajectory by a WTA circuit with N_{cities} neurons, one for each city (Figure 3.3A). To encourage short routes in the energy landscape, synaptic weights between two successive steps are chosen inversely proportional to movement costs, such that low costs map onto strong excitatory synaptic connections, whereas high costs are represented by low excitatory (or inhibitory) connections. The constraint that each city must be visited only once is enforced by inhibitory connections among neurons coding for the same city at different time steps. Furthermore, to facilitate the search process, N_{resting} additional "resting" steps are introduced which allow the salesman to "rest" in a city for one time step before moving on (Figure 3.3A). Note that in the TSP optimization problem the optimality of solutions cannot be easily verified, and hence, in contrast to the 3-SAT application, the objective in this case is not necessarily to recover an *optimal* solution, but to find *good approximate* solutions. We tested the performance of the network architecture with respect to this objective on a planar 38-city problem instance ($\approx 10^{43}$ unique tours). We find that the network quickly generates good approximate solutions to the TSP problem: The average performance of generated network solutions converges within a few seconds to approximately 0.75 (where 1 corresponds to the optimal solution). Furthermore, due to fluctuations around this stationary value, performances up 0.99 are typically reached within a few 10s. We also tested Principle 3 (the advantage of asymmetric spike-based communication) in the described setup. The results are shown in Figure 3.3C-E: Brief periods of partially undefined network states occur with increased frequency in the asymmetric spike-based architecture compared with a comparable symmetrized sampler which samples from the same stationary distribution $p(\mathbf{x})$ (Figure 3.3C). Moreover, convergence to high-performance solutions (short trajectories) is considerably faster in the spike-based architecture compared to the symmetrized system (Figure 3.3D). Similar results are obtained for an asymmetric 39-city TSP problem instance (Figure 3.3E).

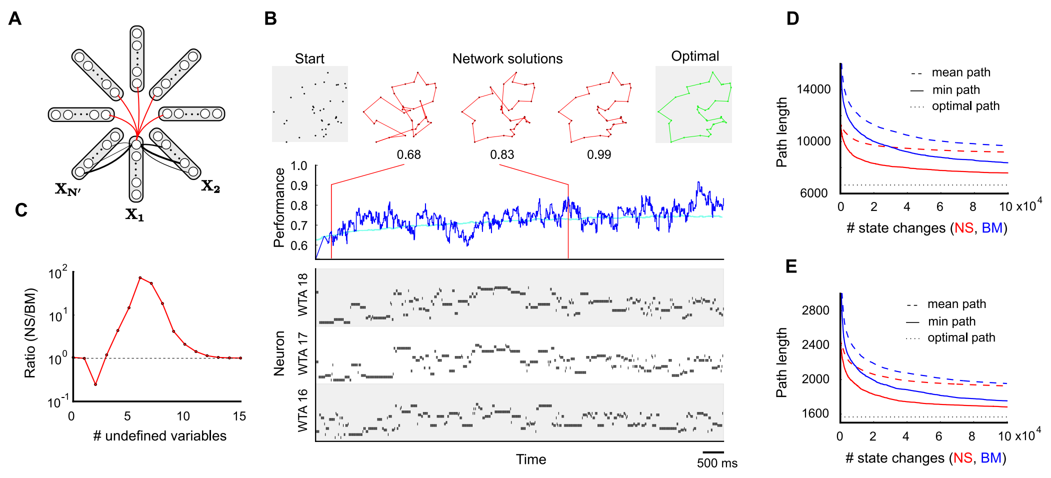

Figure 3.3: Generating approximate solutions to traveling salesman problems (TSP). **A**. Network architecture for solving TSP problems. Each step i in the trajectory of the traveling salesman is represented by a WTA circuit $\mathbf{X_i}$ with N_{cities} neurons (one for each city). The synaptic weights between two steps are chosen to reflect movement costs between each pair of cities. The constraint that each city must be visited exactly once is enforced by inhibitory connections among neurons coding for the same city at different time steps. **B**. Example application to a planar 38-city TSP instance. Top: 38-city problem; solution trajectories generated by the network; optimal solution. Bottom: spike trains of neurons in selected WTA circuits during the first few seconds of a typical run. Middle: network performance over time (dark blue: single trial, cyan: average over 100 trials). The network quickly generates good approximate solutions to the TSP problem. **C**. Advantage of asymmetric spike-based signaling (Principle 3). The number of (very brief) exploratory transitions to network states with partially undefined RVs (where no neuron is active in several WTAs i such that the value of the corresponding RVs $\mathbf{X}_i$ is undefined) is compared between the asymmetric spike-based system (NS=Neural Sampling) and an otherwise equivalent but symmetrized non-spiking system (BM=Boltzmann Machine). Intermittent transitions to exploratory states with more than 3 undefined variables $\mathbf{X}_i$ are found to occur up to 80 times more frequently in the spike-based system. **D**. Convergence is considerably faster in the spike-based system than in the symmetrized system (shown is for each system the cumulative mean/max performance as a function of the number of state changes). **E**. Similar results are found when the same analysis is repeated for an asymmetric TSP problem with 39 cities.

3.5 Discussion

In summary, we have presented a set of four new principles of circuit design with spiking neurons which enable the systematic construction of networks of spiking neurons for solving hard computational problems. In simulations, we have demonstrated this for two well-known problems, 3-SAT and the Traveling Salesman Problem. The proposed architecture for solving 3-SAT (and k-SAT) has a particularly wide range of potential applications. On the one hand, this is because every decision problem in NP can be reduced to the satisfiability problem (Cook, 1971). On the other hand, efficient solvers for satisfiability problems are in high demand in many practical applications, such as model checking and software verification (Biere, 2009) or haplotyping in genomics (Lynce and Marques-Silva, 2006). Apart from boolean k-SAT and TSP, a direct application of the proposed framework to many other important problems such as MAX-CUT, non-boolean k-SAT, the Hamiltonian path problem and graph-coloring (Karp, 1972), should be straightforward to realize and could be examined in simulations in future work. A more ambitious goal for the future is to examine to what extent the proposed framework can be realized in practice on current or future neuromorphic hardware. If successful, this would represent a major breakthrough in the pursuit of a long-standing goal: the demonstration of powerful problem solving capabilities emerging in brain-inspired computing hardware.

3.6 Acknowledgments

This chapter is based on a collaboration with Stefan Habenschuss (SH) and Wolfgang Maass (WM) that was initiated in direct succession of the collaboration on the paper "Stochastic Computations in Cortical Microcircuit Models" described in the last chapter. The paper idea, as well as experimental setup and organization of figures were conceived in discussions of all three authors. The theoretical analysis of the paper was provided by SH. I developed the software tools for the generic and automated construction of large-scale networks from the specification of hard computational problems, as well as the software framework for the large-scale simulation and performance evaluation of the resulting stochastic spiking network architectures. I conducted the computer experiments and analyzed the simulations results. The main text was written by SH and WM, and the Methods were written by ZJ and SH. The figures were designed by ZJ, SH and WM.

Chapter 4

Emergent Coding of Features and Bayesian Computation in Ensembles of Pyramidal Cells with Relaxed Lateral Inhibition

Contents

Neurons in the cortex are not randomly connected. Instead, they form networks of stereotypical microcircuit motifs. We study here arguably the most frequently occurring microcircuit motif: ensembles of pyramidal cells with lateral inhibition. It was shown previously in winner-take-all (WTA) models of this motif that powerful Bayesian computations can emerge through synaptic plasticity if one assumes that at most one pyramidal cell of the ensemble can fire at any moment in time. But new experimental data show that the lateral inhibition between pyramidal cells in layer 2/3 and layer 5 is not consistent with this model, since several pyramidal cells have to fire together in order to engage lateral inhibition. Here we propose a modified computational model for these microcircuit motifs: Sparse WTA (SWTA), where a certain number of pyramidal cells can fire simultaneously, and which is more consistent with these data. We show that this SWTA model has arguably more powerful computational capabilities than the traditional WTA model. Whereas pyramidal cells in a WTA circuit only become selective through synaptic plasticity for specific input patterns that are separately presented, the pyramidal cells in a SWTA circuit are able to become selective for input patterns even if these never occur in isolation, but are superimposed by other patterns and by noise. Furthermore the neurons in a SWTA circuit automatically solve the binding problem, since neurons for different input features fire simultaneously when these features co-occur in the input stream. Thereby this microcircuit motif acquires the capability to extract and represent multiple salient features from complex inputs. A remarkable fact is that these codes for salient features emerge through local synaptic plasticity without any teacher or rewards. Altogether our theoretical analysis and network simulations suggest that synaptic plasticity installs in these ubiquitous microcircuit motifs a very powerful computational operation, that could explain some of the remarkable computational capabilities of a generic cortical column.

4.1 Introduction

WTA circuits constitute a ubiquitous motif of cortical microcircuits (Douglas and Martin, 2004). Recently it was shown that spike-timing-dependent plasticity (STDP) supports the emergence of Bayesian computation in such winner-take-all (WTA) circuits (Nessler et al., 2013; Habenschuss et al., 2013b; Klampfl and Maass, 2013). But these models assumed that the input to a WTA is explained at any point in time by a single neuron, and that strong lateral inhibition among pyramidal cells ensures a basically fixed total output rate of the WTA. These assumptions, however, may not be suitable in the context of more realistic input streams on the circuit level. In particular, neurons in any cortical area simultaneously receive synaptic inputs from a large number of neurons in many other brain areas, and synaptic inputs from each of these other areas might together encode one or several particular features of a currently experienced (or mentally considered) scenario. Indeed, if the underlying circuits are not able to extract and separate those component features, they face the "combinatorial explosion" problem: for the simplest case of n binary features this already gives rise to 2^n possible combinations of feature values. Therefore, from a purely computational perspective this makes it very unlikely that neurons in some brain area specialize in a strict WTA sense, where each possible combination of salient features of external stimuli is represented by a separate neuron. Instead, to avoid the combinatorial explosion problem, each neuron likely specializes on some partial feature of a different component of the input. Since representations in cortex form in a strongly experience dependent manner, this raises the question how such component features could emerge through learning. In order for this to happen each feature should ideally be represented separately in a training sequence where only this particular feature varies and all other features remain fixed. For example, in order to learn representations of orientations of line segments in a particular part of a visual scene, one should ideally train the visual system with a sequence of images with an edge in all possible orientations at this particular location in a visual scene. Obviously, if one does not create such special training sequences, an autonomously learning system has to learn on its own from presentations of combinations of features (e.g. edge orientations at many different positions in a visual scene, possibly combined in addition with varying local colors, textures, and movement directions) what the component features are, and at the same time it needs to create neural codes for each of these component features. Altogether this poses a serious challenge for most models for learning in biological networks of spiking neurons and therefore the apparent open question is how neurons that typically receive synaptic input from a large number of afferent neurons (possibly from several brain areas), which might simultaneously encode several input features from multiple modalities, can learn to isolate these component features and form a suitable representation of their high-dimensional spike input stream.

On a more abstract level, a possible solution to this problem had been proposed by (Földiak, 1990). There it was shown that even in a scenario where combinations of features, such as the position of a vertical bar and simultaneously the position of a horizontal bar that crosses the vertical bar, occur simultaneously in the visual input (in the form of a cross, consisting of two bars), a network of artificial (non-spiking) neurons could in principle self-organize and form one family of neurons in which each neuron represents one possible position of a vertical bar in the scene, and another family of neurons where each neuron represents one possible position of the horizontal bar. Note that once such neural codes for each single feature in these more complex visual scenes (crosses of two bars) have been created, behaviorally relevant combinations of the two features can then be encoded by downstream neurons that learn to respond only to a particular combination (an AND-like computational operation) of two neurons, i.e., to a particular form of a cross. (Földiak, 1990) had proposed to solve this problem through competitive Hebbian learning with a relaxed lateral inhibition, that not only allows a single neuron within a competitive group to provide a high output value in response to a complex stimulus, but allows a larger number of neurons to respond with a large output value, and thereby to update their weights according to some Hebbian rule.

But it turned out to be rather difficult to port this learning principle to models for networks of spiking neurons, with a spike-based learning rule. One interesting model was proposed by (Lücke and Eggert, 2010), but the resulting spike-based learning rule turned out to be very complicated, and it is hard to relate this learning rule to experimental data on synaptic plasticity. In an independent stream of recent work, a different model for spike-based learning (using a very simple STDP-like rule, which is consistent with experimental data on synaptic plasticity) was successfully used in a WTA network of spiking neurons, where only one neuron was allowed to fire (and update its weights via STDP) at any point in time and explains the whole input alone (Nessler et al., 2013; Habenschuss et al., 2013a). However, a generalization to a WTA model where several neurons are allowed to spike simultaneously turned out to be difficult, because the underlying theory (fitting an implicit generative model in the form of a mixture of distributions to the actual distribution of spike inputs) does not support such generalization, as the computational requirements for learning derived from this theory turned out to be inconsistent with spiking neural networks.

Here we propose a solution to this problem in the form of a novel theory-driven model for autonomous learning in spiking WTA networks with soft lateral inhibition which allows for multiple neurons to respond simultaneously in order to jointly explain the input. The resulting neuronal response, which in general depends on the complexity of the input, is sparse and therefore we refer to this model as Sparse WTA (SWTA). The neural responses in this SWTA network are tuned via simple STDP-like learning rule in order to maintain an internal representation of its spike input, which is capable of simultaneously extracting multiple features from the high dimensional input streams. Our approach relies on statistical first-principles which guarantee the emergence of useful internal representations for the extraction of hidden causes of high-dimensional input streams. Although the mathematical proofs apply only to an idealized implementation, we show that this simple but biologically realistic SWTA network implementation, approximates this ideal behavior in simulations well and that indeed it can enhance the computational properties of WTA circuits. Moreover, we demonstrate that neurons in an SWTA circuit can specialize on different time segments of their preferred stimuli, endowing the emergent population code with additional temporal selectivity to the fine structure of spatio-temporal spike patterns. Similar effects have been recently reported by (Luczak et al., 2009a) in rat auditory cortex, where neuronal responses to natural sounds were shown to exhibit considerable temporal selectivity at the population level.

In addition to its substantially increased learning and computing capability, the resulting SWTA model has also a dynamic feature that is more consistent with recent data on the firing response of such populations of pyramidal cells in vivo. When presented with complex input composed of several feature components, the resulting population response will consist of activations of several groups of neurons where each one of these groups code for one of features presented in the input. Therefore the neurons in SWTA which code for the same or similar feature components will tend to fire together and consequently their activity will be correlated (and the number of such correlated neurons depends on the strength of the soft lateral inhibition). This is in agreement with (Sakata and Harris, 2009) who found positive correlation for such pyramidal cells in vivo, especially for close-by pyramidal cells in layers 2/3. Additionally, according to (Avermann et al., 2012) simultaneous activation of several pyramidal cells in layer 2/3 or 4 is required in order to activate lateral inhibition, which is consistent with intrinsic property of SWTA that several groups of neurons spike simultaneously in response to the complex input. Altogether, one may propose that such SWTA motif with soft lateral inhibition provides a better model for the emergent computational operation of populations of pyramidal cells (which could be located for example in layer 2/3, or in layer 5) in a cortical microcircuit which is more consistent with experimental data.

4.2 A model for pyramidal cells with relaxed lateral inhibition

The brain needs to operate in a noisy and constantly changing environment, but in addition it also has to deal with complex input consisting of possibly many component features each of which could be noisy and individually provide only partial information about some percept. Furthermore, individual component features may be superimposed in a nontrivial manner (e.g. combined nonlinearly) and they do not need to be temporally aligned. Nevertheless, neurons exposed to such complex input have to work jointly in order to extract relevant information and to create sparse and efficient codes for representing the most likely components ("hidden causes") of the input. It has been shown in (Nessler et al., 2013; Habenschuss et al., 2013a) that WTA circuits with lateral inhibition can learn through STDP to extract the most likely hidden causes from the input in a strict WTA sense (where the input is explained by only one neuron at a time). Here we show, theoretically and through simulation, that in a WTA motif with *soft lateral inhibition* (set according to theoretical considerations), an STDP-like learning rule enables neurons to learn to simultaneously extract multiple hidden causes from complex input streams, in this way increasing significantly its computational power.

We consider a microcircuit WTA motif that produces sparse codes due to soft lateral inhibition. We refer to this model as Sparse WTA (SWTA). The SWTA motif consists of M stochastic output spiking neurons (Figure 4.1), which we model in a similar way as in (Jolivet et al., 2006): the instantaneous spiking probability of a neuron j depends on its current membrane potential u_j,

$$p(j \text{ spikes in } (t, t+\delta t]) = \delta t \cdot \frac{1}{\tau} \exp(\gamma \cdot u_j(t)) \quad , \tag{4.1}$$

for $\delta t \to 0$. After emitting a spike, a neuron enters a refractory period of duration τ (while in the refractory state, the instantaneous spiking probability of a neuron is zero). Each of these output neurons receives feedforward synaptic inputs $y_1(t), .., y_N(t)$ whose contribution to the membrane potential of a neuron j at time t depends on the synaptic efficiency w_{ij} between the input neuron i and the output neuron j. In addition, each output neuron inhibits all other output neurons within the SWTA circuit. In biological networks, such lateral inhibition occurs di-synaptically via intermediate inhibitory neurons. Here we model this for simplicity by direct inhibitory connections among output neurons (we show later that the functionality of our model can be implemented also with a pool of inhibitory neurons). The total inhibitory drive to an output neuron due to lateral inhibition depends on the number of currently active output neurons and the timing of their activations. The precise impact of an output neuron k on other output neurons j is modeled in two different ways. In the theoretically ideal model, whenever neuron k spikes the output signal $z_k(t)$ of neuron k is set to 1 for a period of duration τ, with $\tau = 10$ms unless otherwise stated (this corresponds to a rectangular post-synaptic potential (PSP)). In the more biologically plausible model, the PSP shape is modeled in a more realistic fashion by a double-exponential function, i.e. the time course of the output signal $z_k(t)$ has a double-exponential shape.

The membrane potential of an output neuron j consists of its intrinsic excitability (α), inhibitory signals from other output neurons, and the sum of weighted inputs,

$$u_j(t) = -\alpha - \sum_{k \neq j} z_k(t)\beta + \sum_i y_i(t) w_{ij} \quad , \tag{4.2}$$

where α, β and γ are constants which can be set based on theoretical considerations and further fine-tuned in order to ensure an optimal working regime of the network.

Although the inhibitory signal is common to all neurons in the circuit, contrary to the inhibition modeled in (Nessler et al., 2013) it does not normalize the firing rates of neurons and therefore the total firing rate of the circuit is variable and depends on the input strength. This has several interesting manifestations. Each time one of the output neurons spikes, it decreases the membrane potential of all other neurons with weight β. This results in a reduced probability of spiking for all other neurons. The extent to which another neuron's firing activity is affected by this reduction depends on the current neuron's membrane potential: generally speaking, the closer the current neuron's membrane potential is to a balanced state (i.e. a membrane potential close to 0), the stronger is the effective influence of inhibition on its activity. If the inhibition is soft (weak), the decrease will be small and therefore some other neuron with the same or a similar feature preference will be very likely to spike despite inhibition. Overall, this enables several neurons (but still a small fraction of all neurons) to respond to the same input, and thereby results in the creation of a sparse code. Interestingly, in a natural input scenario where the number of feature components may vary over time, there will be natural fluctuations of the number of active output neurons, and consequently also a fluctuation in inhibition strength. In particular, if one uses smooth PSPs (e.g. with a double exponential shape), inhibition strength will change smoothly over time. We will see further below that this facilitates desynchronization among neurons. Altogether, we will show that this allows the SWTA motif to robustly handle inputs of different complexity, large variances in input strength and significant delays in inhibitory signal transmission. It should be noted, however, that the circuit has a feedforward architecture and, since there is no fixed normalization of firing rates, the inhibition strength must be tailored according to the expected input strength. In particular, α and β must be set such that they support a certain dynamic range of the input, which is determined by the maximum synaptic strengths and complexity of the input.

In order to learn in an experience-dependent manner and to extract hidden causes from its input stream, the network uses the following theoretically motivated simple STDP-like learning rule. The learning rule is triggered at post-synaptic spike

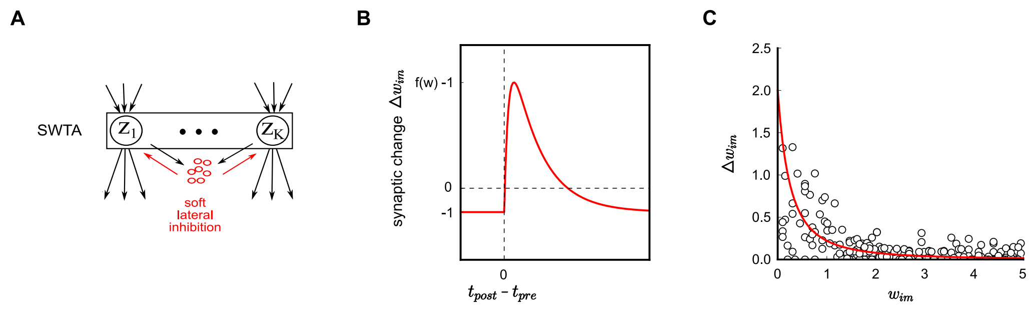

Figure 4.1: The network model and learning curves. **A**. SWTA architecture. Spiking input neurons are feed into a Sparse winner-take-all (SWTA) circuit, where spiking output neurons of the SWTA circuit compete via soft lateral inhibition in order to jointly explain the input. The synaptic weights between input and output neurons are updated through STDP. **B**. STDP-like learning rule. Learning rule is triggered at every post-synaptic spike of output neuron, where potentiation occurs only if the presynaptic spike occurs shortly before the postsynaptic spike and otherwise constant depression takes place. The shape of learning curve depends on the shape of PSPs at synapses(here α shape PSP was used). **C**. Weight potentiation under STDP. The weight update of synapse depends on its current value (red line is ideal dependence) and parameter θ (here set to 2). However the SWTA model can successfully learn even when the weight dependence is very noisy (circuits represent samples from ideal curve with 100% noise).

times t of network neurons j, and changes the synaptic weight w_{ij} between the input neuron y_i and the output neuron z_j according to,

$$\Delta w_{ij}(t) = \eta \left(y_i(t) f(w_{ij}) - 1 \right) \quad , \tag{4.3}$$

where η is a constant learning rate set small enough such that learning takes place on a longer time scale than typical input presentation, and $f(w_{ij}) = 1 + (\sqrt{2}w_{ij} + \theta)^{-2}$ is a weight-dependent term. In addition, the function f depends also on a constant θ which limits the maximum weight update. According to this learning rule, when the presynaptic neuron y_i spikes shortly before the postsynaptic neuron this results in long-term potentiation (LTP) which follows the shape of the PSPs at synapses and is weight dependent. When a post-synaptic spike by j is not preceded by a presynaptic spike i (e.g. when the pre-synaptic spike comes after the post-synaptic spike), this results in long term depression (LTD) which is weight-independent. The learning rule (4.3) can be understood as a local approximation to the optimal theoretical learning rule (see Methods) which can be derived from statistical first-principles by optimizing an implicit generative model of the input statistics. (4.3) approximates the ideal rule by using only locally available information at the synapse. Nevertheless, it emulates the behavior of the optimal rule reasonably well in simulations. Moreover, this learning rule has proven to be quite robust as one can use a noisy version of this learning rule (Figure 4.1C) and still successfully learn to extract multiple hidden causes.

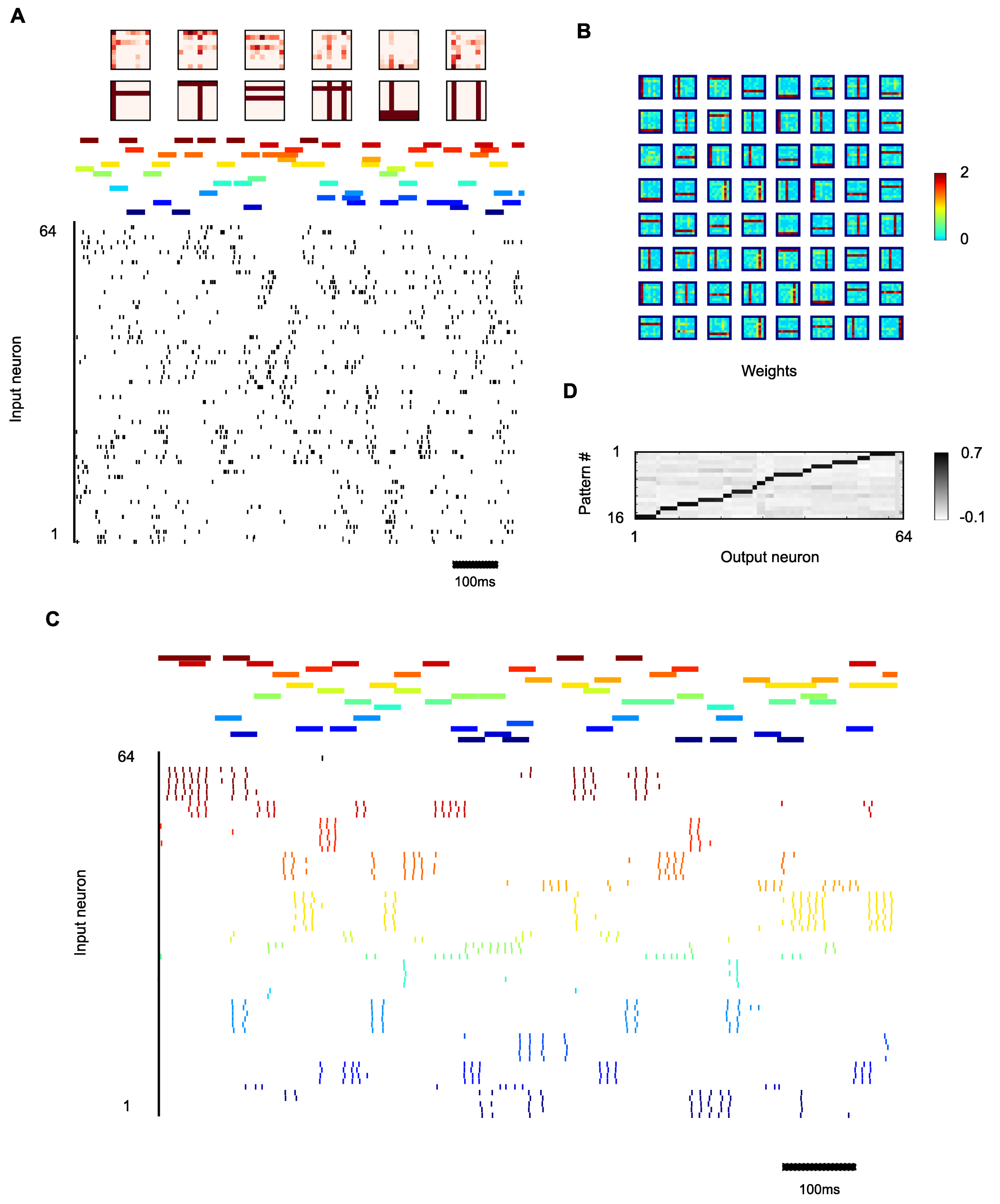

Figure 4.2: (Figure caption on the next page)

To illustrate the performance of the described network architecture at extracting multiple hidden causes we run computer simulations in which we exposed the SWTA circuit to input composed of multiple rate bar patterns which were superimposed in a nonlinear fashion (Figure 4.2). Such input composed of overlapping bars constitutes a well known benchmark for checking whether a model is capable

Figure 4.2: Extraction of hidden causes. **A**. Input spike train made of nonlinear superposition of at most 3 out of 16 different bar rate patterns of 30ms in length which were repeated at irregular time intervals, where lines of different colors at the top of raster plot denote different bar patterns shown at particular time. Examples of overlapped bars and corresponding input that the neurons actually see at different times can be seen at the top. **B**. Weights of output neurons in topological representation after SWTA model was trained on complex input from **A**. Each neuron specialized on one of bar rate patterns. **C**. Response of a SWTA circuit made of 64 output neurons, which receives the input from 64 input neurons, with color bars on top of the raster times of bar pattern presentation are shown as in **A**. Spike responses of output neurons are colored and grouped according to their preferred pattern. The neurons in SWTA are using learned internal representation to successfully extract patterns from the complex input. **D**. Correlation between activity of output neurons and patterns. Each pattern was learned by at least one neuron, and there are several neurons that specialize on a particular bar pattern. On average activity of a neuron and corresponding pattern is highly correlated, with an average of 0.78.

of extracting feature components (Földiak, 1990). Here we used a slightly modified setup where bars are not synchronized (shown at the same time as usual), but are rather shown at random times, such that at any point of time there is a variable number of bars which are superimposed nonlinearly and shown to the network. This is indeed a slightly harder problem as a variable number of component features are presented at various times. This challenges the network dynamics as it requires that inputs of dynamically changing strengths need to handled, but also because bars which partially overlap with other bars represent a substantial source of noise. Figure 4.2A displays a short segment of spiking input that was shown to the SWTA network (colored bars on top of spikes denote the time of presentation of specific bar pattern). There are in total 16 different rate bar patterns (8 vertical and 8 horizontal bars from 8x8 grid) of 30ms duration, where each one of them consists of 64 channels. 8 channels are assigned a 50Hz rate (those corresponding to the bar) and all others are silent (0Hz). The input was created by randomly superimposing (through nonlinear combination) at most 3 out of these 16 bar patterns. The considered SWTA network, consisting of 64 input and output neurons, whose synaptic weights were initialized randomly, was exposed to such input for 120s. During learning each pattern was seen in different combinations with other bars. Nevertheless output neurons started to specialize on individual bar patterns. After learning, most of the output neurons developed a specific preference for one of the patterns (see weights in Figure 4.2C), but also each pattern was picked up by at least one neuron (see Figure 4.2D). After learning the network was able to successfully extract bars (hidden causes) when presented with the input composed of superimposed bars (see Figure 4.2C), which confirms that STDP in fact tuned the internal representation in SWTA to extract feature components.

In addition to the simulation results shown in Figure 4.2, we report that we successfully trained the SWTA model in several variations of the described setup: extracting bar patterns even when those never appeared in isolation, when they were synchronized, superimposed with noise, and when there was a higher number of overlapping patterns (6 and more), as well as when we used frozen instead of rate patterns and when we were using a noisy version of the learning rule (see Figure 4.1).

4.3 Theoretical framework for simultaneous extraction of multiple features from high dimensional input streams

We have demonstrated in the experiment with overlapping bars (see Figure 4.2) that the SWTA model can learn to extract and separate hidden causes through STDP. Here we show that this computational capability that arises from learning and the use of soft inhibition can be theoretically motivated and supported. We base our theoretical analysis on the notion of a generative model that is implicitly implemented in the SWTA circuit. We remark that this approach appears particularly well-suited because generative models constitute one of the most powerful paradigms for unsupervised learning and a well known statistical tool for the extraction of hidden causes from high dimensional data. To tackle the problem of separation of multiple hidden causes we assume that some binary hidden variables $\boldsymbol{z}$ jointly generate observations represented with observed variables $\boldsymbol{y}$. The joint probabilistic model of this graphical model is then given by

$$p(\boldsymbol{y}, \boldsymbol{z}|\boldsymbol{W}) = p(\boldsymbol{z})p(\boldsymbol{y}|\boldsymbol{z}, \boldsymbol{W}), \tag{4.4}$$

where the likelihood of data under the assumption that the individual visible variables are independent given the hidden variables is given as follows

$$p(\boldsymbol{y}|\boldsymbol{z}, \boldsymbol{W}) = \prod_{i=1}^{N} p(y_i|\boldsymbol{z}, \boldsymbol{W}) \tag{4.5}$$

Here the non-negative parameters $\boldsymbol{W} = [w_{ij}]_{N\times M}$ encode the coupling strength between observed and hidden variables (indeed, in the network implementation w_{ij} is the synaptic weight between input neuron i and network neuron j). For the prior distribution on $\boldsymbol{z}$ we assume a form of truncated Gaussian distribution, which for small values of the prior parameter μ favors sparse activation patterns of the hidden variables.

$$p(\boldsymbol{z}) = \frac{1}{Z} \exp\left\{ -(\sum_{j=1}^{M} z_j - \mu)^2 \,/\, 2\sigma^2 \right\} \tag{4.6}$$

Interestingly, in order to end up with a model which can handle biological constraints such as delays of inhibitory signals, the parameter μ needs to be set quite low (indeed it should be negative), as this favors very sparse activation and ensures that neurons can be activated only by strong enough external input, thereby leading to increased noise robustness.

Given some input data with an empirical data distribution $p^*(\boldsymbol{y})$, we would like to adapt parameters $\boldsymbol{W}$ of this model such that it can learn an efficient sparse code for the representation of high-dimensional input streams. More formally, we seek to adapt parameters $\boldsymbol{W}$ to match the generative model $p(\boldsymbol{y}|\boldsymbol{W})$ to the empirical data distribution $p^*(\boldsymbol{y})$, or in another words we would like to minimize the Kullback-Leibler (KL) divergence $\mathrm{KL}\left(p^*(\boldsymbol{y})\,||\,p(\boldsymbol{y}|\boldsymbol{W})\right)$. This optimization of parameters $\boldsymbol{W}$ to minimize KL can be done via Stochastic Online Expectation Maximization (EM) algorithm (Sato, 1999; Bishop, 2006). Indeed, EM is one of the most powerful algorithms in machine learning for fitting generative models to high-dimensional inputs. The optimization via the EM algorithm is done in two steps: the E-step and M-step. In the E-Step given a new input pattern $\boldsymbol{y}^t$ at time t one draws a sample $\boldsymbol{z}^t \sim p(\boldsymbol{z}|\boldsymbol{y}^t, \boldsymbol{W})$. In the M-Step a parameter update is performed according to $\Delta\boldsymbol{W} \propto \partial_{\boldsymbol{W}} \log p(\boldsymbol{y}^t, \boldsymbol{z}^t|\boldsymbol{W})$. Note that the expected update $\langle \Delta W(\boldsymbol{y}, \boldsymbol{z}|\boldsymbol{W})\rangle_{p(\boldsymbol{z}|\boldsymbol{y},\boldsymbol{W})p^*(\boldsymbol{y})}$is guaranteed to decrease the KL-divergence between data and model (if a sufficiently small learning rate is used).

In the following we consider that a spiking neural network may perform an E-by step automatically sampling through its internal dynamics from $p(\boldsymbol{z}|\boldsymbol{y}^t, \boldsymbol{W})$. Additionally, by performing the weight update the network can perform an online M step. In other words, an application of STDP within this motif can be understood as stochastic online EM algorithm.

According to the recently developed neural sampling theory (Buesing et al., 2011), sampling a from desired distribution is carried out automatically by the network of recurrently connected stochastic spiking neurons if the membrane potential dynamics of all neurons satisfy the neural computability condition (NCC). While it is unclear whether and how spiking neurons could compute or sample from the complex posterior distribution $p(\boldsymbol{z}|\boldsymbol{y}, \boldsymbol{W})$ in an exact manner, here we consider an approximation (see Methods for details of this approximation) $\tilde{p}(\boldsymbol{z}|\boldsymbol{y}, \boldsymbol{W}) \approx p(\boldsymbol{z}|\boldsymbol{y}, \boldsymbol{W})$ of the form,

$$\tilde{p}(\boldsymbol{z}|\boldsymbol{y}, \boldsymbol{W}) = \exp\left\{\gamma \cdot (\boldsymbol{z}^T \boldsymbol{R} \boldsymbol{z} + \boldsymbol{y}^\mathsf{T} \boldsymbol{W} \boldsymbol{z})\right\} / \mathrm{norm}. \tag{4.7}$$

where $\boldsymbol{R} = [r_{jk}]_{M\times M}$ with $r_{jj} = -\alpha$, and $r_{jk} = -\beta/2$ for $j \neq k$. An application of the NCC condition to this distribution yields the required membrane potentials, which correspond by construction to the previously defined (4.2), where $-\alpha$ can be interpreted the as intrinsic excitability of neuron, w_{ij} as feed-forward weights and β as recurrent inhibitory weights.

Note that this network will perform only approximate inference as it samples from the approximate distribution $\tilde{p}(\boldsymbol{z}|\boldsymbol{y}, \boldsymbol{W})$. Also, neural sampling guarantees correct sampling from $\tilde{p}(\boldsymbol{z}|\boldsymbol{y}, \boldsymbol{W})$ only under relatively unrealistic constraints, in particular constancy of inputs $\boldsymbol{y}$ for a sufficient amount of time, rectangular PSPs and instantaneous synaptic transmission (zero delay), absolute refractoriness for the duration of a PSP and direct inhibition between pyramidal cells (no inhibitory neurons). Despite all these theoretical constraints, however, in simulations we demonstrate that the network remains nevertheless functional and that it can successfully learn to extract hidden causes also with more realistic neural dynamics: variable input, double exponential PSPs, significant delays, short absolute refractoriness, usage of non-theoretical STDP rules and usage of inhibitory neurons.

Regarding the M-step, the theoretically optimal learning rule is given by $\Delta \boldsymbol{W} \propto \partial_{\boldsymbol{W}} \log p(\boldsymbol{y}^t, \boldsymbol{z}^t | \boldsymbol{W})$, but it contains non-local terms which are hard to interpret from a biological point of view. Therefore by using a local approximation of the optimal learning rule we get the following simple learning rule that emulates the behavior of the optimal rule reasonably well in simulations:

$$\Delta w_{ij} = \eta z_j^t \left(y_i^t f(w_{ij}) - 1 \right) \quad , \tag{4.8}$$

which is in the network implemented by the STDP rule given in (4.3).

4.4 Emergent extraction of time-varying rate patterns from superpositions in complex input streams

So far we have demonstrated that the SWTA circuit is capable of learning to extract individual features when exposed to overlapping superpositions of components. Although rate patterns can look drastically different on each trial (temporally but not spatially) due to the spike variability within different channels, on average they will have the same number of spikes per channel during some time window. Therefore, there is no additional information encoded in each channel other than the rate. Here we challenge the SWTA model with more biological input composed of overlapped spatio-temporal patterns where each channel has a time varying rate and therefore contains additional information about pattern timing. For this purpose we create 2 different spatio-temporal patterns (Figure 4.3A, second column) where each pattern of duration 150ms consists of 64 channels with time varying rates. For each channel we create random rates with an independent Ornstein-Uhlenbeck process, and limit them to at most 50Hz (Figure 4.3A, first column). Two instantiations of the same pattern have huge trial to trial variability, both spatially and temporally (Figure 4.3A, last 2 columns). These two patterns are finally superimposed nonlinearly (in the same way as previously rate bar patterns) at random times to form a complex input stream that will be presented to a SWTA circuit consisting of 100 neurons (see Figure 4.3B where colored bars present times of different pattern presentations). After exposing the network for 100s to such input the output neurons in a SWTA circuit specialize on a specific pattern but also on a particular time segment within this pattern (see Figure 4.3C where neurons are grouped according to pattern preference and ordered according to the mean spiking time within the preferred pattern). In other words, the SWTA circuit is capable of extracting also temporal information and producing a population code which reflects the feature components composing the complex input but also the current temporal position within those components. This is indeed a very important and desirable circuit feature as this provides, for example, a way how to encode time and on-set of a certain stimulus, but also a way how to compute with sequences of activations, while all this information can be inferred from population activity. Similar effects have been recently reported by (Luczak et al., 2009a) in rat auditory cortex, where neuronal responses to natural sounds were shown to exhibit considerable temporal selectivity at the population level. The simplest explanation for such temporal selectivity is that spatio-temporal pattern can be seen as a sequence of very short rate patterns (sub-patterns) and neurons in the SWTA circuit are picking up those sub-patterns which are most discriminative and strongest (in terms of the sum of total rates). It is significant here that SWTA circuit is capable of dealing with a huge number of those sub-patterns, which can differ substantially in strength. An interesting property of such a more biological setup is that there will be significant positive correlation in the co-activation of groups of neurons, where borders between groups are smooth (Figure 4.3D), in agreement with recent data from (Sakata and Harris, 2009) where positive correlation between pyramidal cells in vivo in layer 2/3 was found. Note that although in this experiment individual patterns (actually their segments) do appear in isolation, for demonstration purposes we neglect this as we focus here on the temporal aspect of the model and we have already shown before that the network is capable of extracting patterns which never appear in isolation.

4.5 Computational properties of microcircuit motif with relaxed lateral inhibition

Although the SWTA circuit only approximates the ideal theoretical model for extraction of multiple hidden causes, it is nevertheless capable of performing quite demanding computational tasks (see Figure 4.2 and Figure 4.3). Indeed, it proves to be very robust to different biological constraints such as noise, PSP shapes and inhibition delays. Here, we study in more detail properties of the SWTA circuit while learning 16 different rate bar patters. More precisely, we look at the speed of learning, how precisely neurons are assigned to various discovered patterns, and the influence of the number of output neurons within the circuit as well as the inhibition delay on the quality of pattern extraction. In order to quantify the performance of the circuit we measure the average correlation between occurrence of patterns and activation of neurons which specialized on those patterns (see Methods).

Once the SWTA circuit is exposed to complex input its neurons very quickly start to specialize on some of the patterns and as a result the average correlation (i.e. the performance) increases (Figure 4.4A). During the learning process the circuit recruits more and more available neurons (each specializing on some patterns). After some time this process converges to an apparently stable regime, where the average correlation has also converged and the network has learned most of patterns (on average 15.6). Interestingly, the average correlation reaches almost top performance already after 30s of learning, when each pattern was presented on av-

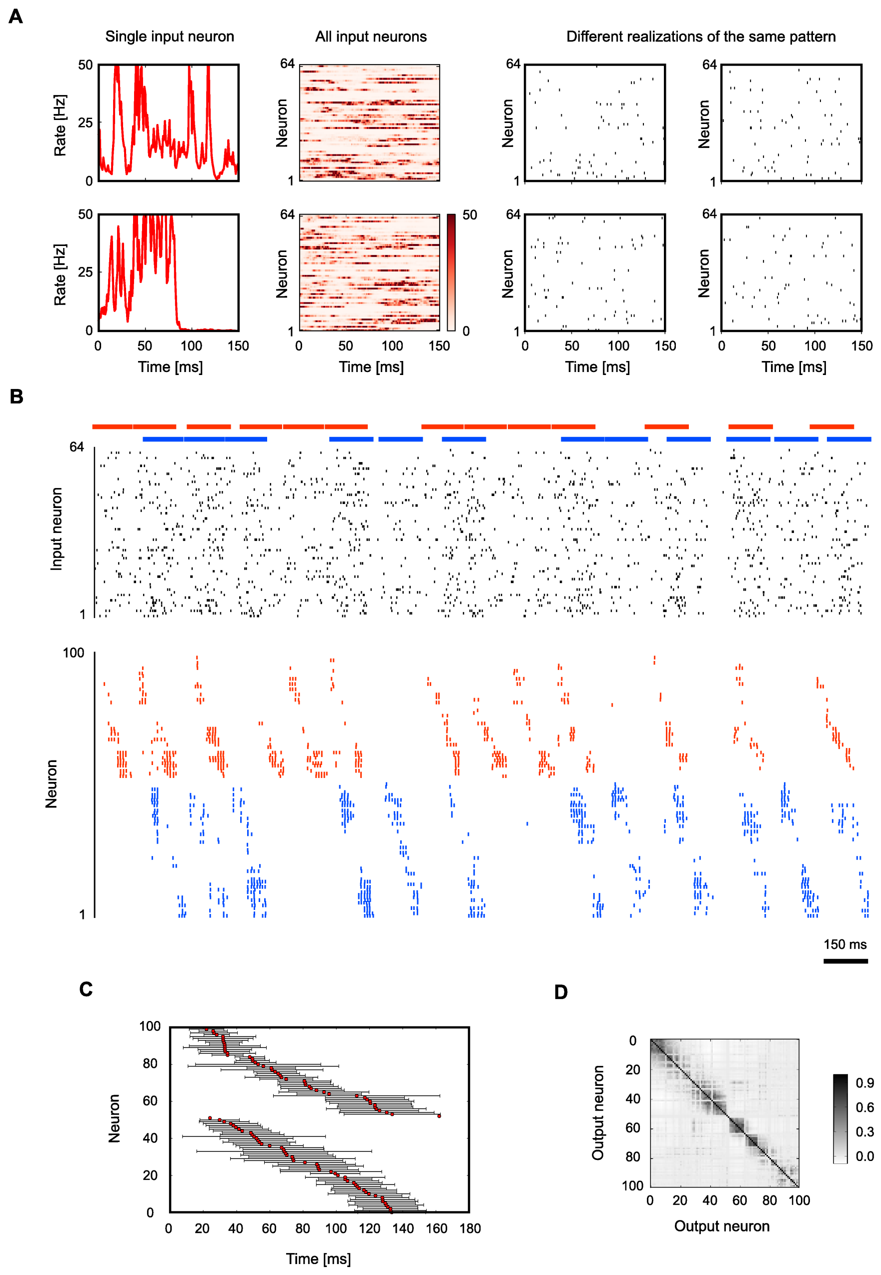

Figure 4.3: (Figure caption on the next page)

erage 100 times, indicating fast learning. On the other hand, there is a minimum number of output neurons needed within the circuit to successfully learn to extract all the patterns. Due to the soft lateral inhibition, more neurons will specialize

Figure 4.3: Emergence of population code through STDP. **A**. Spatio-temporal patterns. Each spatio-temporal pattern of 150ms length consists of 64 input channels (neurons). The evolution of the membrane potential for a particular input neuron is generated with an independent Ornstein-Uhlenbeck process, while the rates for that neuron are calculated as exp(membrane potential). Example of patterns generated from rates that were presented to network are shown on the right. Notice the huge trial to trial variability for different instantiation of same pattern. **B**. The complex input and network response after learning. Two previously defined input spatio-temporal patterns of 150ms in length were repeated and superimposed non-linearly with each other at irregular time intervals (red/blue lines at top). After extended exposure to these superimposed patterns, output neurons in the SWTA circuit developed a specific preference for one pattern (red/blue, spike responses of output neurons are colored according to their preferred pattern and ordered by their mean spiking time relative to the start of their preferred pattern). In particular, each neuron specialized on a specific pattern and on a particular time segment within this pattern. **C**. The mean spike time with standard deviation for each neuron relative to the beginning of the preferred pattern. The neurons are grouped according to the preferred pattern and within the group ordered according to the mean spike time. **D**. Correlation between activity of output neurons in SWTA circuit. Neurons are ordered based on pattern preference and mean spike time within preferred pattern.

on the same pattern and therefore SWTA will need more output neurons than the number of patterns in order to learn (discover) them all. For the case of 16 bar patterns, almost top average performance is already achieved with 35 neurons (Figure 4.4B). Having more neurons does not significantly improve the quality of pattern extraction but it does recruit more neurons (on average the number of neurons that specialize on the same pattern increases). Indeed, approximately a third to half of all neurons will be used by the network to extract patterns while others will remain "free" (Figure 4.4B). This means that not all neurons are used to encode information, but only as many of them as needed with respect to the complex input and inhibition properties (strength, delay). This is generally a desirable property as it leaves space for learning of new patterns which have not been seen before. To check whether this is really the case, we first train the SWTA circuit with complex input consisting of 8 different bar rate patterns, for which the circuit recruits some number of output neurons. After 10s we present a more complex input consisting of additional 8 bar rate patterns that were never seen before (so in total all 16 of them are present in the input after 10s). The free neurons in the network specialize on these new patterns very quickly (Figure 4.4C). As noted before, inhibition plays an important role in determining how many neurons are recruited during the specialization process. Although the ideal theoretical model assumes ideal inhibition (no delays), we show here that the SWTA circuit motif is capable of handling significant inhibitory delays (due to its property that multiple neurons may specialize on the same pattern). Indeed from the results in Figure 4.4C one sees that SWTA is using free neurons to overcome inhibition delays: the bigger the delays the more neurons are recruited. This is a direct consequence of the fact that if inhibition is delayed it cannot immediately decrease the spiking probability of other neurons and therefore more neurons are likely to spike and specialize on the same pattern.

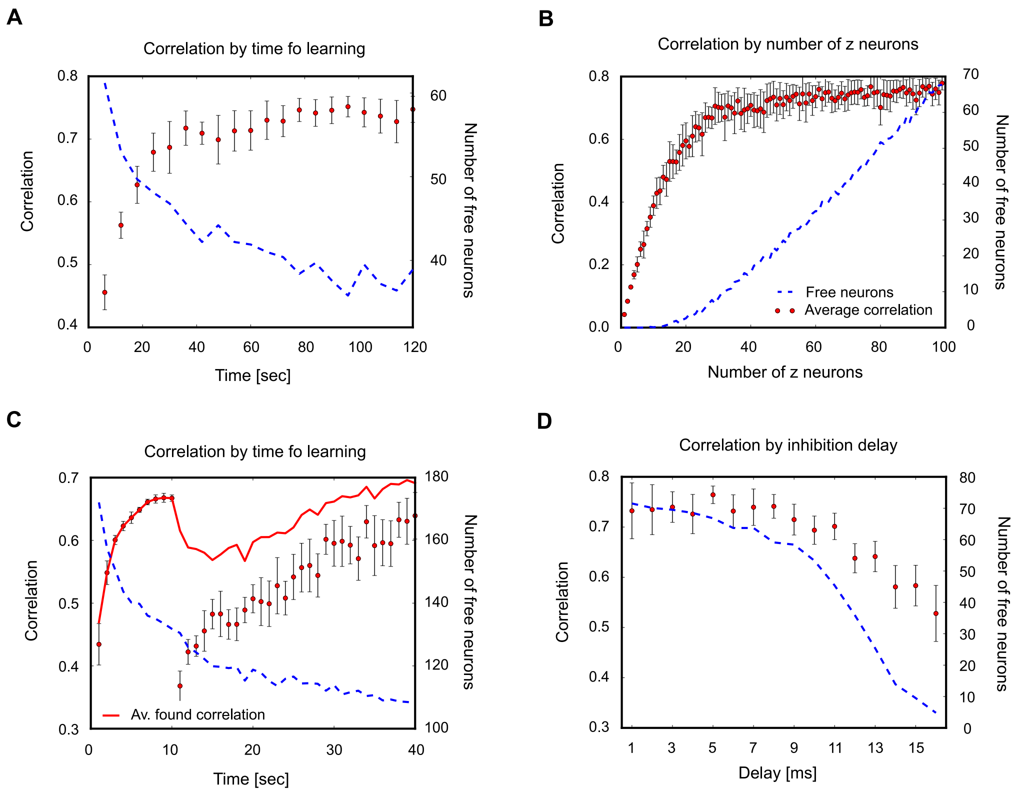

Figure 4.4: Properties of SWTA circuit. **A**. Correlation dependence of learning time. Average correlation between patterns presentations and activation of corresponding neurons calculated based on 10 runs during 120s of learning complex input made by superimposing at most 3 out of 16 bar rate patterns are denoted with red dots (corresponding line denote standard deviation). Dashed blue line shows the number of neurons within SWTA circuit which did not specialize on any of patterns (free neurons), where the SWTA circuit consists of 100 neurons. **B**. Correlation dependence of number of neurons in SWTA circuit. Same as in **A**: the average correlation is denoted by red dots and free neurons by dashed blue line, for different sizes of SWTA circuit. **C**. Learning additional patterns. After 10s of learning SWTA circuit with input where at most 3 out 8 bar rate patterns were superimposed, additional (rest) 8 bar rate patterns were added to the input, so that the new input is made by superimposing 3 out of 16 patterns. The average correlation and free neurons are denoted as in **A**. The red line denote average correlation of found patterns. **D**. Correlation dependence of inhibition delay. The average correlation and free neurons are denoted as in **A**.

4.6 Discussion

We have presented a theoretical analysis of STDP in biologically realistic network motifs with relaxed inhibition. Theory and simulation results suggest that this biologically more realistic SWTA circuits with soft inhibition exhibits in conjunction with STDP substantially enhanced computational power. In particular, it enables neurons to develop additional temporal selectivity and to solve difficult tasks such as the extraction and separation of multiple features from high-dimensional input streams. We have demonstrated this in computer experiments where the microcircuit motif extracted single features from asynchronous superpositions of rate and spatio-temporal spike patterns, the latter arguably representing a quite realistic input structure for a cortical column. We showed that learning in this motif is quite fast and enables neurons within this motif to acquire receptive fields for the extraction of individual feature components. Interestingly, all this is achieved without the need for complex learning rules adjusting neural excitabilities. Furthermore, the spike-based learning rule uses only locally available information. Moreover, we have also shown that while this microcircuit motif only approximates an underlying theoretical model for extraction of multiple hidden causes, it is nevertheless very powerful and robust in spite of deviations from the theoretical dynamics due to various biological constraints. More precisely, we showed that microcircuit motif can handle significant inhibition delays and biologically plausible PSP shapes. Since the SWTA circuit can handle significant inhibition delays this allows to consider more realistic implementations of lateral inhibition via an inhibitory population, rather than through direct inhibitory connections among output neurons. Given the robustness to inhibition delays observed in simulations, such more realistic implementation, in which lateral inhibition occurs via propagation of a spike from an output neuron to the inhibitory population, the activation of inhibitory neurons and the propagation of a spike back from the inhibitory population to output neurons, could in principle still work. However there are several potential obstacles. If one simply connects a population of excitatory neurons (output neurons in SWTA circuit) and an inhibitory population with all-to-all connections, where the inhibitory population consists simply of one inhibitory neuron, this inhibitory neuron will inhibit every output neuron in SWTA circuit while in theory it should inhibit all except the one which activated it. In addition, this single inhibitory neuron should be infinitely fast (with no refractory period) in order to respond to each spike from output neurons. The later problem can be somewhat relaxed if one considers inhibitory populations consisting of multiple neurons, because in that case there is potentially always some inhibitory neuron which is not refractory and therefore can be triggered by an incoming spike from an output neuron. On the other side, the strength of ideal inhibition should be proportional to the number of active output neurons. This condition is easily violated since inhibitory neurons are also stochastic and therefore the net inhibition strength seen by other output neurons has a significant variance and depends nonlinearly on the number of active output neurons. Note that in order for any such approach to work one has to set the excitatory connections (from output neurons to inhibitory neurons) strong enough such that one spike coming from an output neurons can trigger one inhibitory neuron (but it should not trigger them all). The inhibitory connections (from inhibitory neurons to excitatory neurons) have to be set according to the theory (depending on the prior $p(\boldsymbol{z})$). Interestingly, most of these problems can be circumvented very simply by choosing the excitatory connections such that inhibitory neuron can be triggered only by at least G incoming spikes (or activation of at least G output neurons), while the inhibitory connections should be set G times stronger than theoretically proposed, where G is the number of neurons which should specialize on the same pattern. Indeed, preliminary simulations with a setup in which inhibitory population consists of several inhibitory neurons where each one of them needs several incoming spikes to be triggered showed that such SWTA circuit with quite realistic implementation of inhibition can successfully learn to solve the problem almost equally well. We also investigated in preliminary simulations how additional biological constraints, such as the use of a more common and biologically plausible STDP rule, influence the capability of an SWTA circuit to extract hidden causes. For this we considered an STDP rule which is triggered at every pre- and post-synaptic spike (the theoretically optimal learning rule is triggered only at each post-synaptic spike), where the potentiation part of STDP is time- (pairing difference) and weight dependent (both are exponentially dependent), while the depression part is only time dependent. Despite the differences between this more common and the theoretically optimal STDP rule, we found that, with a proper choice of STDP parameters, the SWTA circuit can nevertheless successfully learn to extract hidden causes. Clearly, although these preliminary results, which we report here only verbally, are encouraging, more simulations and a quantitative analysis are required to corroborate these findings in future work.

In addition to the more abstract models (Földiak, 1990; Lücke and Eggert, 2010), related work includes a recent model for Independent Component Analysis with spiking neurons (Savin et al., 2010). In this work the authors derived theoretical rules for intrinsic plasticity which, when combined with input normalization, weight scaling and STDP, enable a neuron to extract one of a set of independent components of inputs. While closely related in terms of the pursued computational goal, two key differences to the present work can be identified. First, in contrast to the present work, no theoretical explanation for the computational role of lateral inhibition could be provided by (Savin et al., 2010), because theoretical derivations were limited to the single neuron case. Second, in contrast to (Savin et al., 2010), the model presented in this article achieves automatic separation of input features with only two basic circuit components, soft lateral inhibition and STDP, without the need for weight normalization and intrinsic plasticity.

The proposed learning rule (4.3) is structurally similar to those reported in (Nessler et al., 2013; Habenschuss et al., 2013a) who studied emergence of feature representations in a strict WTA setting. This is insofar significant as it raises the question why the application of almost the same learning rule in one motif leads to learning and extraction of a single hidden cause and in another to the extraction of multiple causes. The answer most likely lies in the interplay between "prior knowledge" in the model (e.g. in the form of the intrinsic excitability of neurons), the learning rule and inhibition strength: As there are multiple neurons in the SWTA model which can spike in response to the same input, each one of them can adapt its synaptic weights to increase the likelihood of spiking again whenever the same or a similar input pattern is presented in the future, possibly in conjunction with other different input components. This is manifested through increased total input strength to those neurons when the pattern is seen again. But this results also in increased total inhibition to all other neurons, thereby effectively limiting the number of winners. As there is no fixed normalization of firing rates (probabilities), as soon as the input strength caused by a single feature component is strong enough to trigger the spike in some neuron, the neuron will respond to each pattern which consists of that particular feature. On average this will force neurons to specialize on a single feature component. Therefore, after learning, each spike can be interpreted as indication of a particular feature component.

Altogether, given its increased computational power and robustness to many biological constraints and noise, as well as its consistency with biological data, the presented microcircuit model provides a substantial contribution to the understanding of emergent computational operations in stereotypical cortical microcircuits.

4.7 Acknowledgments

This chapter is based on a collaboration with Robert Legenstein (RL), Stefan Habenschuss (SH) and Wolfgang Maass (WM). The idea of this work was originally conceived by WM and RL. Based on an initial report by RL, I developed the theory of the paper, with useful contributions from SH. The computer experiments were designed by ZJ, RL, SH and WM. I wrote the software framework, conducted the experiments, and analyzed the results. The manuscript was written by ZJ, RL, SH and WM.

Chapter 5

Software framework for simulations of spiking neuronal networks

Contents

The research done in this thesis required extensive and complex simulations of large data-driven cortical microcircuits made of complex synapse and neuron models (Chapter 2), as well as construction of large and complex neural networks consisting of thousands of neurons that embody distributions with specific properties (Chapter 3). Thus, construction and simulation together with analysis of those networks, as well as adaptation of them in order to gain certain properties, required very specialized software tools. To support those software demands I have developed new software packages, ZLIB and CSP2SNN, which allow for parallelization, optimization and automatic creation of large neural networks with specific properties. Additionally, I have extended the existing neural network simulators, PCSIM and NEVESIM, with required neuron and synapse models. These tools together made the simulation framework which was used throughout the research of this thesis and which enabled for simple and fast simulation, exploration and analysis of the underlying large neural networks.

As a basis for simulations of neural networks within this thesis the PCSIM and the NEVESIM simulators were used. PCSIM is a clock-based neural simulation environment intended for simulation of spiking and analog neural networks with a support for distributed simulation of large-scale neural networks on multiple machines. NEVESIM is a software package for event-driven simulation of networks of spiking neurons which supports simulation of heterogeneous networks with different types of neurons and synapses.

As a basis for parallel execution, optimization and analysis of simulated neural networks, ZLIB and CSP2SNN were used. ZLIB is a parallelization and optimization library written in Python with an intuitive and simple user interface which enables the end user to harvest the power of all connected machines in order to execute a given set of tasks. CSP2SNN is a software package written in Python on top of the NEVESIM simulator for porting Constraint Satisfaction Problems (CSP) to the implementation in Spiking Neural Networks (SNN). CSP2SNN allows for an automatic mapping of CSP constraints to SNN architecture as well as for a manual implementation of additional circuit mechanisms.

By combining these tools such as in Chapter 3 for solving CSPs, where CSP2SNN was used on top of the EVESIM simulator to port required CSP problem to SNN architecture, and ZLIB was used to explore the dependencies and optimize the SNN architecture such that it can solve required CSP problem in an optimal manner, one can get a powerful simulation framework for automatic creation, simulation, optimization and analysis of spiking neural networks.

The following sections contain an overview and description of the simulation tools developed for the purpose of this thesis, ZLIB and CSP2SNN, and a short demonstration of the NEVESIM simulator's capability to simulate exactly and efficiently networks of stochastic spiking neurons from the theoretical framework of neural sampling, which was the basis for most of the work done in this thesis. Note that additional features and details of the NEVESIM architecture can be found in the paper.

5.1 CSP2SNN: a framework for implementing Constraint Satisfaction Problems in Spiking Neural Networks

Constructing manually large networks consisting of thousands of neurons with a specific architecture, which embody distributions with specific properties, capable of solving certain class of problems such as Constraint Satisfaction Problems can be a tedious and quite difficult process. For this reason, within the scope of this thesis CSP2SNN, the framework for porting Constraint Satisfaction Problems to the Spiking Neural Networks, was developed. CSP2SNN is written in Python on top of the NEVESIM simulator which is abstracted away so that the end user has to deal only with a CSP notation and abstract definitions of neurons and synapses, therefore no knowledge of NEVESIM is required (note that due to the Python flexibility all the NEVESIM power is still accessible through this framework).

The main feature of CSP2SNN is an automatic mapping of CSP constraints, both hard and soft, to the SNN architecture, which assumes WTA motifs of neuron populations that code for a value of associated variables. The automatic mapping of the constraints is done based on a library of known constraints implementations, e.g. implementation of *unique value* or *unique var* constraint will result with a WTA motif of underlying neuron populations as in Chapter 2. Therefore the automatic mapping feature can be used to convert only CSPs whose constraints are already known to the library, or one has to extend the library with new constraint implementation in order to support new constraint. In addition to the automatic mapping it is possible to manually override or extend automatically created architecture which allows for the manipulation of the whole architecture on the neural population level (creation of a neuron population and connections between them, where the population can consists of only one neuron and therefore one can gain control at the level of a single neuron as well). The manual mapping is important for the implementation of additional circuit mechanisms which extend the default network architecture not defined with constraints, such as the lock-in mechanism described in Chapter 3.

Architecture

The CSP2SNN framework is divided into four layers: problem definition, constraints, architecture and core (see Figure 5.1). This separation makes the CSP2SN architecture very flexible as it allows for a implementation of many different constraints and architectures which can be additionally manually fine-tuned or set, but also reused. Therefore the real power of the architecture is in its reusability as many different CSP problems, which are similar in nature in terms of types of constraints they use, can be basically implemented identically. Indeed, in order to implement different CSP problem which uses the same set of constraint types as some existing one, it would be enough to change only the problem database interface for problem loading and constraints definition (definition of the variables and values they take).

Problem layer

The problem layer consists of classes for database interface and readout mechanism definition. The database interface allows for loading of a CSP problem from the database written in different file formats such as CNF notation for 3-SAT problems (see Chapter 3). In addition the CSP problem can be loaded from database via Python script which automatically creates an instance of a certain CSP problem. For example, it is possible to generate automatically problem for 3-SAT with specific properties(number of clauses and variables, and theirs ratio), which can be checked automatically for satisfiability with some other 3-SAT solvers such as zchaff software (Fu et al., 2004)(note that SNN 3-SAT solver presented in Chapter 3 can not make decision about the satisfiability, but can rather provide close to the optimal solution or solution if there is such). The class for the readout mechanism allows for a definition of how exactly the activity of a neuron population is mapped to the variable values. In this way the user can simply choose whether it wants to use one to one representation (the population coding), to use multiple neurons to represent the same value, to use population rate coding or some other mapping.

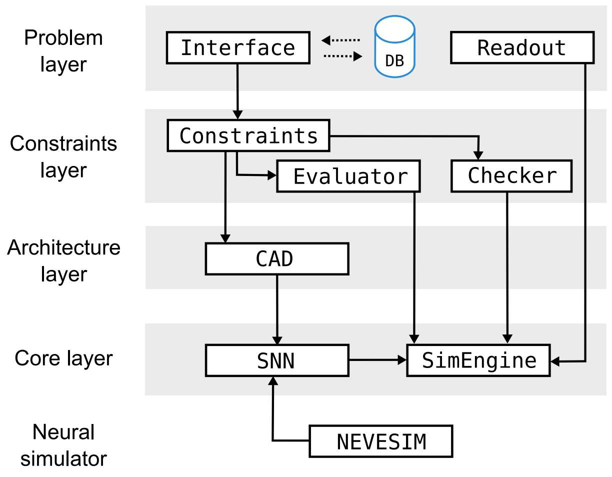

Figure 5.1: CSP2SNN architecture. The CSP2SNN framework is build on top of the NEVESIM simulator and consists of four layers: problem, constraints, architecture and core. The problem layer provides an access to the database containing definition of problems and it gives possibility to define how the neural activity is interpreted and read out. All CSP problems defined based on the same type of constraints can be implemented by working only on this layer. The constraint layer provides an access to the definition of the constraints and how they are checked and evaluated, which is needed when implementing a CSP problem which contains a new constraint (note that this is done once for each new constraint). The architecture layer provides a way to manually define the architecture, which is useful for implementation of additional mechanisms not covered by the constraints. Finally, the core layer contains the simulation engine and interface to NEVESIM.

Constraints layer

The constraints layer contains classes for CSP problem definition, which maintains the list of constraints defining the CSP that will be used for construction of the corresponding SNN, classes for checking the consistency of all constraints and variables, and classes for evaluation of the proposed solution (the ratio of satisfied constraints or the ratio of variables not breaking any of constraints). Each constraint consists of a type which describes the relation between the variables taking part in the constraint, variables' IDs which take part in the constraint and theirs values.

Architecture layer

The mechanisms for mapping of CSP to SNN are provided in the architecture layer and those mechanisms are used to convert a list of constraints defining the CSP to a neural implementation. This can be done automatically via the library of known constraints implementations or manually by defining neuron population names (IDs) and connections between them. Automatic mapping is done by applying a corresponding constraint implementation for a given constraint from the list of constraints. For example, if a constraint is of type *unique var* the corresponding constraint implementation will take all the neuron populations which code for different values of considered variable and arrange them into a WTA circuit (an additional population of inhibitory neurons will be automatically created, to which each one of these populations will be connected bidirectionally).

Core layer

Finally, the core layer provides the functionality for implementation and simulation of spiking neural network based on specifications provided by the architecture layer, through *SNN* and *SimEngine* classes. The *SNN* class, which constructs and simulates the network for a certain time, takes definitions of neural populations types (the size, neuron type and properties), connections types between populations (synapse properties and connection probability) as well as other network or NEVESIM specific properties such as network temperature control parameters used for temperature scheduling (oscillations, simulated annealing). The final network constructed by SNN is based on these definition and constraints implementation architectures. The SimEngine unifies all the classes and theirs functionality by providing a common place where the constructed network is simulated in a specified way, the readout of neuron populations is constantly performed, and the proposed solutions are evaluated and checked for the consistency.

5.2 ZLIB: a library for parallelization and optimization

Large computing resources in a form of cluster or locally connected computers, nowadays more and more common, are ideal for farming tasks or executing multiple instances of the same task simultaneously on different machines, e.g. for simulation of large neural networks or gathering statistics. Moreover, such approach allows for parameter exploration or search for the optimal setup in a task. In order to support such simulations, which were essential to this thesis, I have created ZLIB, the parallelization and optimization library, which essentially provides a command creation procedure for the execution of scripts via the *ssh* protocol at all connected machines sharing the same file system (e.g. RAID storage system). In other words, ZLIB provides a simple and intuitive way for running multiple copies of a certain script or putting it in a loop in order to optimize for a target objective function.

ZLIB is written in Python and it abstracts away parallelism and optimization, such that the end user can focus on solving their problem. It harvests the power of all connected machines to execute a given set of tasks and at the same time it takes into account the current and the maximum load of each machine while trying to equalize the usage of available machines between the users. While getting an insight in a small subset of parameters dependencies is possible through trying out all the possible combinations of parameters (with a brute force approach), this fails for larger subsets of parameters. Therefore the real power of the library is in its simple usage and combination with optimization which facilitates a clever parameter search.

While other libraries for parallelization exists, such as e.g. iPython parallel (Perez and Granger, 2007), ZLIB offers a unique environment which combines the functionality for parallel execution, optimization, and automatic organization and storage of results and their analysis. Furthermore, it allows for control over machines being used and their loads, and provides an intuitive and simple user interface which enables rapid usage of the whole framework.

Architecture

The architecture of ZLIB is based on the *Command creator* class for creation of all commands that will be executed, a hub for distribution of these commands to the remote machines, procedures for saving and loading results, and an optimization module (see Figure 5.2). All these individual modules are put together in templates in order to provide the desired functionality and a simple interface for the users.

Templates

The end user of ZLIB is exposed to the two templates, *main.py* and *test.py*, which interact in order to implement the desired functionality of parallelization or optimization. While *main.py* contains only a high level implementation of a loop for the exploration of a parameter state space and the creation of corresponding commands, as well as the communication with the remote computers and saving/loading of results, its main purpose is to provide a central place for the definition of parameters and their ranges which are to be explored. The *main.py* will indeed create a central hub for parallel execution of the other template, *test.py*, to which it will pass a specific combination of parameters, where the *test.py* basically contains the desired code one wants to execute multiple times, a header which parses parameters, and a footer which saves the results of execution. The parsing of parameters from command line is done with Python's *commandparser* class, where defining the support for a command is as simple as naming the parameter and providing its type. Obviously, all the parameters listed in *main.py* must be parsed in the header of *test.py* and the code must be modified to accept those parameters. Finally, execution results are saved at the end by using the *shelve* class from Python, which saves the results at the location specified by *main.py* in the form of a dictionary so that they can be later easily retrieved for analysis.

As there are two main functionalities that ZLIB provides, parallelization and optimization, there are also two different *main.py* templates: one for a parallel execution of desired code and the other for optimization, which is executed in parallel as well. The *main.py* for optimization contains, in addition to the definition of parameters, also the definition of an objective function which is calculated based on saved results from *test.py* - the user in principle has to specify only the key (name) of the results to be loaded.

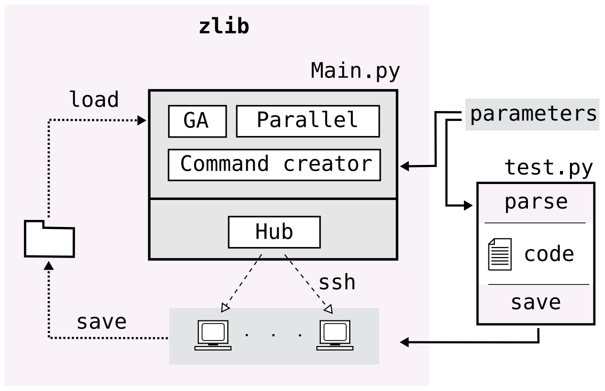

Figure 5.2: ZLIB architecture. The architecture of ZLIB consists of several modules encapsulated in two templates, the *main.py* and the *test.py*, which interact in order to implement the desired functionality. These two templates are connected via the definition of parameters, which define the state space to be searched. The *main.py* template contains an interface to the *Command creator* class for the creation of all commands that will be executed, a hub for the distribution of these commands to remote machines via the *ssh* protocol, procedures for loading results, and an optimization module (GA). The *test.py* template contains procedures for parameter parsing and saving results, and the code that is to be parallelized (or executed multiple times).

Parameters

The main connection between the *main.py* and the *test.py* templates are parameters. The same set of parameters that one wants to explore within the original code should be parsed at the beginning of the *test.py* while in the *main.py* ranges of those parameters should be defined. Each parameter in *main.py* can be assigned either of two types, *fix* or *param*, where the type *fix* assumes a parameter which is not changed during simulations and is defined by the name and value of the parameter, while the *param* type assumes a real parameter defined by the name and a set of values (defined by the range and resolution) it can take during the simulation. Discretization of parameter values allows for having a finite number of values and therefore a finite number of parameter combinations which simplifies the automatic creation of all possible combinations when applicable. Note that by defining a high resolution of parameters one indeed approximates a continuous space.

Command creator

The main class for creation of all commands is *Command creator*, which creates for each considered combination of parameter values a command that will execute the target script at a target machine with specific values of parameters. In addition to the user defined parameters also the following ZLIB parameters are added automatically by the *Command creator*: ID, outfolder and results. The Outfolder and results parameters organize script results in a specific structure that is easy to explore and analyze during or after simulations, while the ID parameter is used for repetition of exactly the same test - using the same parameter values, which can be useful for simulations of stochastic models in order to obtain e.g. an average performance.

Hub

A distribution of tasks to the remote machines is done via the hub in an asynchronous manner. The hub keeps track of the load of each machine and accordingly sends them new tasks as soon as there are free resources at some of them according to certain criteria. Communication with remote machines is done via *ssh* protocol, through which new tasks are started and through which also each connected machine is regularly checked (in intervals of several seconds) for the current load in order to determine the number of new processes that could be run on it. Due to the use of the *ssh* protocol the ZLIB is not intended for execution of very short tasks or for communication between different processes to implement real parallel execution of some problem, but rather the library is aimed for simulations which take from minutes to hours to execute.

There are two modes the hub can operate in: equal balance, in which it balances the load of all connected machines, and priority queue, in which it tries to fill up machine by machine according to some predefined priority list and depending on the current load of machines. These two modes allow for multiple users (or a single user with multiple tests) to simultaneously perform simulations - to share equally available computational resources. In addition the hub enables the user to define the maximum number of processes which will run at the same time at all machines as well as the maximum load per individual machine. Together, asynchronous execution of tasks and load-limit per machine allow to connect machines of different power (different number of CPUs or amount of memory) into one functional computational unit which will be used in an optimal way for the execution of given tasks.

Optimization

As parallel execution is the default behavior of ZLIB, for a given list of parameters the *Command creator* automatically creates all possible combinations of *param* type parameters. This behavior is useful for exploration of few parameters within some range (with finite resolution), but obviously the state space can be easily blown up and therefore more clever ways of state space exploration are required. For this reason, on top of this parallel functionality an optimization module is added which loads and analyzes results during the execution in order to direct the search in the parameter space in a clever way. While in general the choice of optimization technique depends on the type and nature of problem, for the purpose of simulations and optimization within scope of this thesis, a Genetic Algorithm

(GA) (Whitley, 1994) was chosen as GAs are known to be able to escape local minima and generate useful solutions to optimization problems. Note that ZLIB can be extended with any other optimization algorithm as it has a simple and modular structure which allows for simple extensions. a GA is a heuristic algorithm which mimics natural selection by using inheritance, mutation, selection, and crossover in order to generate solutions to optimization problems. While there are many different variants of GAs, ZLIB contains an implementation of a GA with roulette based selection and injection of new individuals, where a population consists of individuals whose chromosome codes for parameter values, where crossover is done between parameters and mutation is done within each parameter by choosing a new value of a parameter which is within some range from its the current value.

5.3 Simulation of Neural sampling networks with NEVESIM

As already pointed out, one marked feature of NEVESIM is its capability to efficiently simulate, through an exact event-driven simulation, neural models derived from a novel theoretical framework for computation with network of spiking neurons called neural sampling (Buesing et al., 2011; Pecevski et al., 2011; Habenschuss et al., 2013a). The neural sampling theory gives a new perspective of how biological networks of neurons can perform probabilistic inference computations, by showing that, given certain assumptions, their stochastic dynamics can be interpreted as MCMC sampling. One of the values of this theoretical result is that it creates a link between many existing probabilistic computational models on a behavioral and cognitive level, and models of networks of spiking neurons which model brain computations on a neural level. Furthermore, neural sampling provides a bridge for porting a large body of useful results from the field of Machine Learning on MCMC sampling and stochastic computations with artificial neural networks, to the modeling research that uses networks of spiking neurons as more detailed, lower level models of brain computation. Thus, the capability to simulate neural sampling models brings an additional value to NEVESIM as a simulation tool. Indeed, it can be very useful for users that want to explore further the potential of the neural sampling theory and neural sampling models for elucidating various aspects of the organization of computation in networks of spiking neurons in the brain.

In order to demonstrate this typical usage of NEVESIM for simulating networks of spiking neurons that perform neural sampling, we will present in the following an example simulation demo of such networks. As a first example, we will construct a neural sampling network $\mathcal{N}$ that in its stationary dynamics samples from a probability distribution with second-order interactions in the following form

$$p(\mathbf{z}) = \frac{1}{Z_N} \exp\left(\sum_{i<j} w_{ij} z_i z_j + \sum_i b_i \right) \tag{5.1}$$

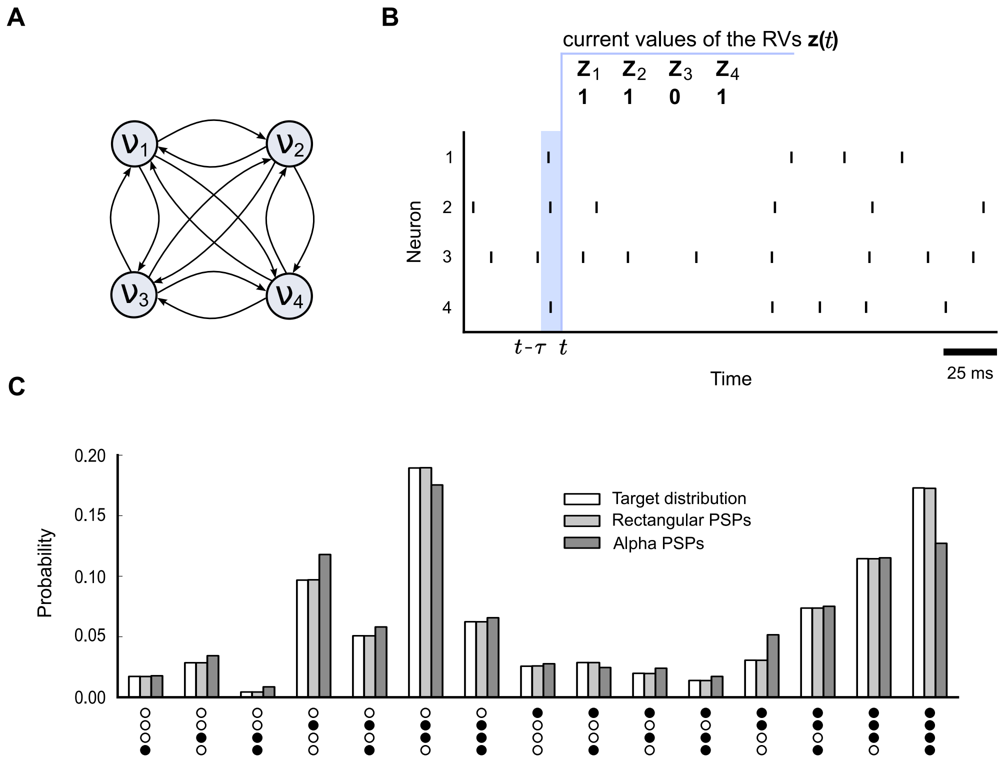

Figure 5.3: NEVESIM simulation example with a simple neural sampling network. **A**. A graph of the example neural network with 4 neurons and all-to-all connectivity. As the neural sampling network corresponds to a probability distribution with second-order dependencies, the synaptic connections and their weights are symmetric. **B**. Spiking activity of the neural network and depiction of the definition of vector of the current values of the random variables $\mathbf{z}(t)$ that represents a sample from the distribution $p(\mathbf{z})$. In continues time a variable z_k is set to 1 if neuron ν_k spiked in last τ period and otherwise is set to 0. We used $\tau = 20ms$. **C**. Comparison between the target probability distribution $p(\mathbf{z})$ (white bars) and the estimated probability distributions from the simulation of the neural network with rectangular shaped PSPs (light gray bars) and the neural network with alpha shaped PSPs (dark gray bars). The bars show the probabilities of all possible assignment of values to the random variables z_1, z_2, z_3, z_4, except the zero assignment (0,0,0,0).

where $\mathbf{z} = (z_1, z_2, \ldots, z_K)$ is a vector of binary random variables, w_{ij} and b_i are the parameters of the distribution and Z_N is the normalization constant. The constructed network $\mathcal{N}$ consists of K spiking neurons $\nu_1, ..., \nu_K$, one for each random variable in the distribution. Each neuron ν_k is a stochastic firing neuron with a instantaneous firing probability equal to $\rho_k(u_k) = \frac{1}{\tau}\exp(u_k)$ where u_k denotes the current value of the membrane potential of the neuron. After it spikes, the neuron has an absolute refractory period of duration τ during which it is silent. According to the neural sampling theory, a sufficient condition for the network to sample from $p(\mathbf{z})$ is that the membrane potential of neuron ν_k at time t is equal to $u_k(t) = b_k + \sum_{i=1}^{K} w_{ki} z_i(t)$. Here b_k is the bias of the neuron, w_{ki} is the strength of the synaptic connection from neuron ν_i to ν_k, and $z_i(t)$ is the post-synaptic potential caused by a firing of the neuron ν_i that has value 1 during the time interval of duration τ after a spike of ν_i, and otherwise value 0. If the network $\mathcal{N}$ satisfies the sufficient condition, then its firing activity in the stationary regime generates, at any point in continuous time, a correct random sample from the distribution $p(\mathbf{z})$. The samples are defined by the spikes of the neurons, by setting $z_k(t) = 1$ if and only if the neuron ν_k has fired within the preceding time interval $(t - \tau, t]$ of length τ, and otherwise setting $z_k(t) = 0$ (see Fig. 5.3B). A convenient property of neural sampling, similar to other stochastic systems in MCMC sampling in general, is that the same network $\mathcal{N}$ that samples from $p(\mathbf{z})$ can also estimate marginal posterior distributions derived from $p(\mathbf{z})$. Marginal posterior distributions are calculated in probabilistic inference tasks when we have concrete evidence about some of the random variables, and we want to estimate the probability of some of the unknown random variables given the evidence.

It is easy to see from their definition that it is possible to simulate the theoretically ideal neural sampling models exactly with an event-driven simulation algorithm. What makes them in particular suitable for very fast implementation is the rectangular shape of the PSPs. Namely, with a rectangular shape the state of the network should be updated only at the time of a spike, and at the time when an EPSP ends, resulting in at most two network state updates per spike, which is quite efficient.

For implementation in NEVESIM we considered a simple network composed of 4 neurons with absolute refractory period ($\tau = 20$ms). The values for the w and b parameters were drawn from a normal distribution with mean equal to $\mu = 0$ and standard deviation $\sigma = 1$. The resulting network according to the neural sampling theory consists of neurons with intrinsic excitability given by b_i that are interconnected by symmetric synapses given by w_{ij}, see Fig. 5.3A. The spiking activity of this network is shown in Fig. 5.3B. At any point in time the current values of the variables $\mathbf{z}(t)$ can be read out by looking for each neuron ν_k whether it fired a spike in last τ ms (Figure 5.3B) and accordingly assigning a value to the associated variable z_k. According to the neural sampling theory, the empirically estimated distribution obtained by counting the produced samples from the simulated network should converge to the target distribution $p(\mathbf{z})$. A comparison between the target distribution and the estimated distribution calculated from the generated samples during 1000s of network simulation confirms that the network really samples from the target distribution (see Figure 5.3C). In addition to using the theoretically ideal rectangular shapes of the PSPs elicited by the input spikes, we also performed simulations with a network that has piecewise constant spike responses that approximate an alpha shape PSP. It can be observed that even when using alpha-shaped PSPs as an approximation of the ideal rectangular PSPs, the estimated distribution by the

```
1 net = NeurSamplingNetwork()
2 nrn_ids= [ net.create(ExpPoissonNeuron(1.0/tau, b[i], tau)) for i in
      range(K)]
3 for src_id in nrn_ids:
4     for dest_id in nrn_ids:
5         if src_id != dest_id:
6             syn_factory = CompositeSynapse(BasicActiveSynapse(w[i,j]),
7                                            ResetRectSpikeResponse(tau))
8             net.connect(src_id, dest_id, syn_factory)
9 net.simulate(T)
```

Table 5.1: A code snippet for generic implementation of a neural sampling network in NEVESIM.

network is very similar to the target distribution (Figure 5.3C), with minor errors for some assignment of values.

The demonstrated neural sampling network can be implemented in NEVESIM with a few lines of code given in Code Block 5.1. In the code one assumes that the network consists of `K` neurons with refractory period of duration `tau`. We additionally assume that the biases of all neurons are given in one dimensional array `b` and the synaptic weights in a two dimensional array `w`. The network is simulated for `T` seconds. At the beginning, in the first line of the code, we create a NEVESIM network object `net` which is an instance of the `NeurSamplingNetwork` class, which will hold all the created network elements, such as the neurons and synapses, and the connections between them. In the next line we create K neurons via the `create` method of the `net` object. To simplify the code we use here *list comprehension* construct in Python which is a convenient way to create a Python list with a for loop. The return value of the statement in the second line is a Python list with the IDs of all created neurons. The used neuron type `ExpPoissonNeuron` to create the neurons implements a stochastic neuron with an instantaneous firing probability that is an exponential function of the membrane potential $\rho = C\exp(u)$, exactly as defined in the neural sampling theory. Its constructor accepts three arguments, where the first argument is the coefficient `C` in the firing rate function, in this case set to `C = 1/tau`, the second argument is the bias of the neuron set to `b[i]`, and the third argument is the duration of the refractory period set to `tau`. After the second line we have two nested for loops which create the synaptic connections between the neurons. The for loops iterate through the IDs of the created neurons and invoke the `connect` method of the `net` object to create a synaptic connection from the neuron with ID `src_id` to the neuron with ID `dest_id`, by utilizing a so-called synapse factory object `syn_factory` (line 8). The utility of the synapse factory object is for the user to specify all traits of the synaptic connection that is to be created between the two neurons. In the particular case we specify that we want to have a composite synapse, which means that the synaptic connection will be composed of two network elements, the synapse network element and a separate network element for the spike response. For the synapse network element we use the type `BasicActiveSynapse` which is appropriate for neural sampling, and for the spike response we use the type `ResetRectSpikeResponse` which implements a rectangular shaped PSP. Note that the `connect` method of the `NeurSamplingNetwork` class differs from the basic connect method of the `EvSimNetwork` class, in that it implements a more complex functionality of connecting two neurons with a synapse, rather than just connecting two network elements with an event connection. Other than the rectangular shaped PSP, NEVESIM also has spike response classes that implement piecewise constant and piecewise linear shapes of responses, which can be used for approximating other arbitrary shapes of PSPs, like for example an alpha shape which we also used in our simulations. In this case, however, having higher precision of PSP shapes requires more segments in the spike response which comes at a price since it reduces the efficiency of the simulation.

5.4 Acknowledgments

CSP2SNN and ZLIB were developed by me. The last section on NEVESIM is based on my contribution to a joint work with Dejan Pecevski (DP) and David Kappel (DK). In the course of this joint work of writing the manuscript, which presents the architecture of NEVESIM developed by DP and demonstrates its usability, DP wrote the main text, while DK and ZJ conducted simulations for the manuscript and wrote the corresponding parts of manuscript describing them. The simulator website, `http://sim.igi.tugraz.at/`, was authored by me.

Appendix A

List of Publications

1. Jonke, Z., Silic, A., & Dalbelo B. B. (2009) Exploring String and Word Kernels on Croatian-English Parallel Corpus, Intelligent systems MIPRO.
2. Jonke, Z. (2009) Text classification applying kernel methods over sequences of haraters and words, Diploma thesis, University of Zagreb.
3. Jonke, Z., Habenschuss, S., Legenstein, R., & Maass, W. (2012) Improved feature extraction by pyramidal cells through relaxed lateral inhibition. In *42nd Annual Conference of the Society for Neuroscience*.
4. Habenschuss, S., Jonke, Z., & Maass, W. (2013) Networks of spiking neurons with noise have an inherent capability to generate heuristic solutions to complex constraint satisfaction problems. In *43rd Annual Conference of the Society for Neuroscience*.
5. Maass, W., Habenschuss, S., & Jonke, Z. (2013) Neural circuits naturally encode probability distributions over network states, and over trajectories of network states. In *43rd Annual Conference of the Society for Neuroscience*.
6. Habenschuss, S.*, Jonke, Z.*, & Maass, W. (2013) Stochastic Computations in Cortical Microcircuit Models. *PLoS Computational Biology* 9(11):e1003311.
7. Jonke, Z.*, Habenschuss, S.*, & Maass, W. (in preparation) Networks of spiking neurons with noise can solve hard computational problems.
8. Jonke, Z., Legenstein, R., Habenschuss, S., & Maass, W. (in preparation) Emergent coding of features and Bayesian computation in ensembles of pyramidal cells with relaxed lateral inhibition.
9. Pecevski, D., Kappel, D., & Jonke Z. (submitted) NEVESIM: Event-based Neural Simulation Framework with a Python Interface.

* both authors contributed equally

Comments and Contributions to Papers

Publications 1. and 2. were written during my Master studies and are not included in this thesis.

The paper *Stochastic Computations in Cortical Microcircuit Models* was written in collaboration with Stefan Habenschuss (SH) and Wolfgang Maass (WM). The experimental setup and organization of figures were developed in close collaboration of all authors, by ZJ, SH and WM. The theoretical analysis of the paper was provided by SH. I developed the software and simulation framework required for large-scale simulation of the stochastic cortical column model, as well as the analysis tools for studying the convergence properties of networks of spiking neurons (the Gelman-Rubin analysis of one-dimensional and high-dimensional spike trains). The simulation and computational convergence analysis of the cortical column model and other microcircuits for Figure 2.2-2.4 as well as the simulations and analysis for Figure 2.5 was done by me. The main text was written by WM and SH, and the Methods were written by ZJ and SH. The figures were designed by ZJ, SH and WM.

The manuscript *Networks of spiking neurons with noise can solve hard computational problems* was written in collaboration with Stefan Habenschuss (SH) and Wolfgang Maass (WM). The paper idea, as well as experimental setup and organization of figures were conceived in discussions of all three authors. The theoretical analysis of the paper was provided by SH. I developed the software tools for the generic and automated construction of large-scale networks from the specification of hard computational problems, as well as the software framework for the large-scale simulation and performance evaluation of the resulting stochastic spiking network architectures. I conducted the computer experiments and analyzed the simulations results. The main text was written by SH and WM, and the Methods were written by ZJ and SH. The figures were designed by ZJ, SH and WM.

The manuscript *Emergent coding of features and Bayesian computation in ensembles of pyramidal cells with relaxed lateral inhibition* was written in collaboration with Robert Legenstein (RL), Stefan Habenschuss (SH) and Wolfgang Maass (WM). The idea of this work was originally conceived by WM and RL. Based on an initial report by RL, I developed the theory of the paper, with useful contributions from SH. The computer experiments were designed by ZJ, RL, SH and WM. I wrote the software framework, conducted the experiments, and analyzed the results. The manuscript was written by ZJ, RL, SH and WM.

The manuscript *NEVESIM: Event-based Neural Simulation Framework with a Python Interface* was written in collaboration with Dejan Pecevski (DP) and David Kappel (DK). In the course of this joint work of writing the manuscript, which presents the architecture of NEVESIM developed by DP and demonstrates its usability, DP wrote the main text, while DK and ZJ conducted simulations for the manuscript and wrote the corresponding parts of manuscript describing them. The simulator website, `http://sim.igi.tugraz.at/`, was authored by me.

Appendix B

Appendix to Chapter 2: Stochastic Computations in Cortical Microcircuit Models

Contents

B.1 Network states and distributions of network states

Markov states

The Markov state $y_M(t)$ (or more explicitly, $y_{M:\Theta}(t)$) of a network at time t is defined here as the recent history of spike times of all neurons in the network within the period $(t-\Theta, t]$. The term “Markov” refers to the fact that, under mild conditions and for a sufficiently long window Θ, the network dynamics of a neural circuit after time t becomes independent of the network activity at times $\leq t-\Theta$, given the Markov state $y_M(t)$ and the external input x. Hence, the network dynamics has the Markov property with respect to this state definition.

For each neuron $k \in 1 \dots K$ in a neural circuit a spike history of length Θ is defined as the list of spike times emitted by neuron k within the window $(t-\Theta, t]$.

Spike times are counted relative to the beginning of the window at $t - \Theta$. If m is the number of spikes within $(t - \Theta, t]$ for neuron k, then the list takes the form,

$$y_M^k(t) \equiv (y^{k,1}(t), \dots, y^{k,m}(t)) \in \mathbb{R}^m \quad , \tag{B.1}$$

where $0 < y^{k,1}(t) < \cdots < y^{k,m}(t) \leq \Theta$.

We denote the space of all possible network states of length Θ by S_Θ or, when unambiguous, simply by S. Note that this definition is equivalent to the state definition in (Borovkov et al., 2012), to which the interested reader is referred for further formal details (e.g. the associated σ-algebra $\mathcal{S}$ of the state space S).

Scope of theoretical results: Required properties of the network and neuronal noise models

We study general theoretical properties of stochastic spiking circuit models, driven by some external, possibly vector-valued, input $x(t)$, which could represent for example input rates in a set of input neurons or injected input currents. Formally, the input sequence can assume values from any state space Q; a concrete example is vector-valued input with $Q = \mathbb{R}^N$, where N is the number of input dimensions.

We consider in this article two different noise models for a neuron: In noise model I, the spike generation is directly modeled as a stochastic process. All network dynamics, including axonal delays, synaptic transmission, short-term synaptic dynamics, dendritic interactions, integration of input at the soma, etc. can be modeled by a function which maps the Markov state (which includes the recent spike history of the neuron itself) onto an instantaneous spiking probability. This model is highly flexible and may account for various types of neuronal noise. In the more specific noise model II, the firing mechanism of the neuron is assumed to be deterministic, and noise enters its dynamics through stochastic vesicle release at afferent synaptic inputs. Also combinations of noise models I and II in the same neuron and circuit can be assumed for our theoretical results, for example neurons with a generic stochastic spiking mechanism which possess in addition stochastic synapses, or mixtures of neurons from model I and II in the same circuit.

In noise model I, the instantaneous spiking probability of neuron k at time t is given by,

$$\lim_{\delta t \to 0} \frac{1}{\delta t} \cdot p(\text{neuron } k \text{ fires in } (t, t + \delta t)) = \rho_k(t) \quad . \tag{B.2}$$

This instantaneous firing rate $\rho_k(t) = f(y_M(t))$ at time t is assumed to be bounded and completely determined by the network's current Markov state $y_{M:\Theta}(t)$, for some sufficiently large Θ. More precisely, the following four assumptions are made for noise model I:

$A1$ **Spikes are individual events:** We assume that,

$$\lim_{\delta t \to 0} \frac{1}{\delta t} \cdot p(\text{more than one neuron fires in } (t, t + \delta t)) = 0 \quad , \tag{B.3}$$

which is, for example, fulfilled if each neuron has some independent source of stochasticity.

$A2$ **Bounded rates:** The instantaneous firing rates are bounded from above: $0 \leq \rho_k(t) \leq \hat{\rho}_k$ for some $\hat{\rho}_k < \infty$. The ensuing upper bound on the total network firing rate is denoted by $\hat{\rho}$, i.e. $0 \leq \sum_{k=1}^{K} \rho_k(t) \leq \hat{\rho}$. It is assumed that instantaneous rates are bounded at any time, and in the presence of any input $x(t)$.

$A3$ **Bounded memory:** The firing rates $\rho_k(t)$ at time t depend on the network's past activity only through the history of recent spikes in a finite window $(t - \underline{\Theta}, t]$ of length $\underline{\Theta}$. Hence, the *direct* effect of a spike at time t on future firing rates of all neurons is limited to a bounded "memory period", $[t, t + \underline{\Theta})$. This bounded memory period $\underline{\Theta}$ can be understood as a lower bound for Θ during the subsequent convergence proofs (since smaller Θ would violate the Markov property). In addition to this bounded-memory dependence on network spikes, $\rho_k(t)$ may depend on the current input $x(t)$ in any manner consistent with $A2$.

$A4$ **Time-homogeneity:** The functional mapping from recent spikes and/or input signals $x(t)$ to instantaneous firing rates $\rho_k(t)$ does not change over time. In particular, we do not consider long-term plasticity of synaptic weights and/or excitabilities in this work.

Assumptions $A2-A4$ can be summarized as follows: Let $x(t) \in Q$ and $y_{M:\underline{\Theta}}(t) \in S$ be the trajectories of input and network states as defined above. Then there exists a memory constant $\underline{\Theta}$ and rate bounds $0 \leq \hat{\rho}_k < \infty$, such that for each neuron k there exists a function $f_k : Q \times S \rightarrow [0, \hat{\rho}_k]$, where $\rho_k(t) = f_k(x(t), y_{M:\underline{\Theta}}(t))$ for all t. The function f_k is time-invariant but otherwise unconstrained, and can capture complex dynamical effects such as non-linear dendritic interactions between synaptic inputs or short-term plasticity of synapses.

The input signal $x(t)$ can formally represent any variable which exerts some arbitrary influence on the instantaneous network dynamics (the neuronal firing functions f_k). In the simplest case, $x(t)$ could be a vector of firing rates controlling the spiking behavior of a set of N input neurons i, such that $f_i(x(t), y_{M:\underline{\Theta}}(t)) = x_i(t)$ in these neurons. In this case (which we focused on in the main text), input neurons are formally considered part of the circuit C. Note that in principle, $x(t)$ could also represent the strength of currents which are injected into a subset of neurons in the network C, or the recent spiking history of a set of external input neurons ("input Markov states"). If the input comprises rates or currents, these can be either fixed (e.g. fixed input firing rates) or dynamically changing (in particular rates which are either subject to stochastic ergodic dynamics, or periodically changing rates). Below convergence proofs will be provided for both fixed and dynamic input conditions. If the input is defined in terms of input Markov states, the dynamic input analysis is applicable under conditions described further below.

In noise model II the basic stochastic event is a synaptic vesicle release (in noise model I it is a spike). Accordingly, the Markov state $y_M(t)$ of a network in noise model II is defined as the list of vesicle release times for each synaptic release site in the network (instead of spike timings for each neuron). We assume here that each synaptic release site releases at a given instance t at most one vesicle filled with neurotransmitters. But a synaptic connection between two neurons may consist of multiple synaptic release sites (see (Lisman et al., 2007; Branco and Staras, 2009) and (Borst, 2010) for reviews). Instead of expressing the network dynamics through an instantaneous firing probability function for each neuron k, $\rho_k(t) = f_k(x(t), y_{M:\underline{\Theta}}(t))$ (noise model I), for noise model II the network dynamics is expressed in terms of instantaneous release probabilities for each synapse k: $\psi_k(t) = g_k(x(t), y_{M:\underline{\Theta}}(t))$. Similar to noise model I, it is assumed that there exists a window length $\underline{\Theta}$, such that the dynamics of vesicle release at time t is fully determined by the timing of previous vesicle releases within $(t - \underline{\Theta}, t]$, and hence can be expressed in terms of a corresponding variation of the definition of a Markov state $y_{M:\underline{\Theta}}(t)$. The same framework of assumptions applies as in noise model I: vesicle releases are individual events, and the functions g_k are assumed to be bounded from above by rate constants $\hat{\psi}_k < \infty$.

Combinations of noise model I and II are also possible. In this case, the Markov state $y_M(t)$ may contain both spike times and vesicle release times. The assumptions of noise model I/II described above apply to the corresponding stochastic neurons and vesicle releases, respectively. Altogether, note that all three types of networks (based on model I, II and mixtures of the two) are based on a common framework of definitions and assumptions: in all cases the dynamics is described in terms of stochastic components (neurons, synapses) which generate point events (spikes/vesicle releases) according to instantaneous probabilities which depend on the recent event history of the network.

Convergence of state distributions

Below, proofs for the existence and uniqueness of stationary distributions of network states for the considered network models are given. Furthermore, bounds on the convergence speed to this stationary distribution are provided. To obtain a comprehensive picture, convergence is studied under three different input conditions: constant, stochastic, and periodic input. All proofs are described in detail for noise model I. The results transfer in a straightforward manner to noise model II and mixtures of these two models, since the same framework of assumptions applies to all cases.

Network dynamics as a Markov process

We view the simulation of a cortical microcircuit model, under a given input condition and starting from a given initial network state, as a random experiment. Formally, we denote the set of all possible outcomes in this random experiment by

Ω, the set of all considered *events* by $\mathcal{F}$ (i.e. a σ-algebra on Ω), and the probability measure which assigns a probability to each event in $\mathcal{F}$ by $\mathbb{P}$. An outcome is the result of a single run of the network. An outcome is associated with an assignment of particular values to all defined random variables. An event is a set of outcomes, for example the set of all outcomes in which neuron 7 spikes within the first 200 milliseconds of the experiment. Suppose X is a random variable with some state space $(R, \mathcal{R})$, i.e. X assumes values in R, and $\mathcal{R}$ is a set of events on the space R. Formally, such a random variable X is defined as a map $X : \Omega \to R$, which assigns a value $x \in R$ to every possible outcome $\omega \in \Omega$. To denote the probability that the random variable X assumes some value in the set $B \in \mathcal{R}$, we define the short-hand $\mathbb{P}_X(B) := \mathbb{P}(X \in B)$. Furthermore, if Y is another random variable we use the notation $\mathbb{P}_{X|Y=y}(B) := \mathbb{P}(X \in B|Y = y)$ for conditional probabilities, and write even shorter, when unambiguous, $\mathbb{P}_{X|y}(B)$. The base probability space $(\Omega, \mathcal{F}, \mathbb{P})$ is assumed to be rich enough such that all random variables which are needed in the following exist.

We define the index set of time $T = \{t \in \mathbb{R} : t \geq 0\}$, and the stochastic process $(Y_t,\ t \in T)$, as a description of the stochastic evolution of Markov states of a network C for $t \geq 0$. For each time $t \in T$ we define a random variable Y_t (also written $Y(t)$) representing the Markov state of the network at time t. Y_t takes values on the state space S of all possible Markov states of some fixed duration Θ. We denote by $\mathcal{S}$ the σ-algebra associated with S. The assumptions on the network described in the previous section imply that the process has the Markov property for Markov states of any length $\Theta \geq \underline{\Theta}$, since the future evolution of the process is then entirely independent of the past, given the current Markov state. For the subsequent proofs, we therefore assume some $\Theta \geq \underline{\Theta}$. We also define a random variable Y of *entire sample paths* on the measurable space $(S^T, \mathcal{S}^T)$, i.e. a map $Y : \Omega \to S^T$. Realizations of Y are sample paths (or trajectories), i.e. functions $y_M(t),\ \ t \in T$, taking values in S. Since realizations of Y are functions, Y can be thought of as a random function.

For subsequent proofs the following definition of a *transition probability kernel* is essential: A transition probability kernel $\mathcal{P}$ on a measurable state space $(S, \mathcal{S})$ is a function $\mathcal{P} : S \times \mathcal{S} \to [0, 1]$, which assigns a probability to the transition from any point $x \in S$ to any set $B \in \mathcal{S}$. More precisely, if one fixes a particular "initial state" $x \in S$, then $\mathcal{P}(x, B) \equiv \mathcal{P}_x(B)$ is a probability measure in its target argument B, corresponding to the result of applying the transition kernel $\mathcal{P}$ to x (in addition, for each event $B \in \mathcal{S}$ in the target space, $\mathcal{P}(x, B)$ is $\mathcal{S}$-measurable in its source argument). Stochastic transition matrices of Markov chains are, e.g., transition probability kernels.

Here we write $\mathcal{P}^{s:t}$ for the transition probability kernel corresponding to progression of the state of the network C from time s to $s + t$, i.e.,

$$\mathcal{P}^{s:t}(y_0, B) := \mathbb{P}(Y(s + t) \in B \mid Y(s) = y_0)\ . \tag{B.4}$$

We further define the shorthand $\mathcal{P}^t = \mathcal{P}^{0:t}$ for the progression of duration t starting from initial time $s = 0$. Transition kernels can also be applied to probability measures ϕ of initial states y_0 (as opposed to single initial states y_0). We will write $\mathcal{P}^{s:t}\phi$ to denote the result of applying the kernel $\mathcal{P}^{s:t}$ to an initial probability measure ϕ. The result $\mathcal{P}^{s:t}\phi$ is again a probability measure, assigning a probability to any event $B \in \mathcal{S}$ on the state space according to:

$$(\mathcal{P}^{s:t}\phi)(B) := \int_S \mathcal{P}^{s:t}(y_0, B)\, d\phi(y_0) \quad , \tag{B.5}$$

Since $\mathcal{P}^{s:t}\phi$ is again a probability measure on the state space $(S, \mathcal{S})$, transition kernels can be applied sequentially. Note that due to the Markov property one has, $\mathcal{P}^{r:(t_1+t_2)}\,\phi = \mathcal{P}^{(r+t_1):t_2}\,\mathcal{P}^{r:t_1}\,\phi$ for $s \geq 0, \;\; t_1, t_2 > 0$.

Stochastic network dynamics is contracting

Before studying specific input conditions, a few basic key properties of the network dynamics Y are developed. Let $\mathcal{P}^{s:t}$ be the transition probability kernel corresponding to progression of the network C from time s to $s + t$. For the proofs below, transitions to the *resting state*, $Y(s + t) = \mathbf{0}$, will be of particular importance. The resting state $\mathbf{0}$ is defined as the "empty" Markov state in which no spikes occurred within the last Θ time units. The first key observation is the following Proposition:

Proposition 1 *Consider the probability $\mathcal{P}^{s:\Theta}(y_0, \mathbf{0})$, that the process Y will be in the resting state $\mathbf{0}$ at time $s + \Theta$, starting from some initial state $y_0 \in S$ at time s. This "return probability" to the resting state is bounded from below by,*

$$\mathcal{P}^{s:\Theta}(y_0, \mathbf{0}) \geq \epsilon^{\Theta} \quad , \tag{B.6}$$

where $\epsilon := e^{-\hat{\rho}}$. This holds regardless of the input trajectory $x(t)$ driving the network.

The proposition follows directly from the fact that $\hat{\rho}$ bounds the sum of all instantaneous firing rates in the network. Hence with at least probability $e^{-\hat{\rho}\Theta} = \epsilon^{\Theta}$ no neuron fires within Θ time units (cf. (Borovkov et al., 2012)). In technical terms, this implies that the stochastic kernel corresponding to a duration of length Θ fulfills the Doeblin condition (Doeblin, 1937) – a property which is highly useful for proving convergence and ergodicity results.

Proposition 1 entails a central contraction property of stochastic networks of spiking neurons C, which holds in the presence of any input trajectory $x(t)$, and forms the basis for several subsequent proofs. The following definitions are essential: We will measure below the difference between any two probability distributions ϕ_1 and ϕ_2 in terms of the total variation $\|\cdot\|$ of the signed measure $\mu = \phi_1 - \phi_2$. Any such signed measure μ can be expressed in terms of its non-negative and non-positive components, $\mu = \mu^+ - \mu^-$, where μ^+ and μ^- are both non-negative measures (but in general no probability measures). The total variation of a signed measure μ on a measurable space $(X, \mathcal{X})$ is defined as $\|\mu\| = \mu^+(X) + \mu^-(X)$, i.e. the total mass of its positive and negative components. According to this definition, $\|\mu\| = \|\mu^+\| + \|\mu^-\|$.

Lemma 1 (Contraction Lemma) *The following strict contraction property holds for the Markov process Y, for any $\Theta \geq \underline{\Theta}$, and for any initial probability measures ϕ_1 and ϕ_2 at any time $s \geq 0$:*

$$\|\mathcal{P}^{s:\Theta}\phi_1 - \mathcal{P}^{s:\Theta}\phi_2\| \leq (1-\epsilon^{\Theta}) \cdot \|\phi_1 - \phi_2\| \ . \tag{B.7}$$

In words: applying the dynamics of the network C for Θ time units is guaranteed to reduce the distance between any two initial distributions ϕ_1 and ϕ_2 of network states by a factor $1-\epsilon^{\Theta}$.

Proof: Define the auxiliary measure ν_0 as zero everywhere outside $\mathbf{0}$, and $\nu_0(\mathbf{0}) = \epsilon^{\Theta}$. Rewrite $\phi_1 - \phi_2 = \mu = \mu^+ - \mu^-$ in terms of the non-negative measures μ^+ and μ^-, such that

$$\|\phi_1 - \phi_2\| = \|\mu^+\| + \|\mu^-\| \ , \tag{B.8}$$

and note that $\|\phi_1\| = \|\phi_2\| = 1$ implies that $\|\mu^+\| = \|\mu^-\|$. Then

$$\begin{aligned}
\|\mathcal{P}^{s:\Theta}\phi_1 - \mathcal{P}^{s:\Theta}\phi_2\| &= \|\mathcal{P}^{s:\Theta}\mu^+ - \mathcal{P}^{s:\Theta}\mu^-\| && \text{(B.9)}\\
&= \|(\mathcal{P}^{s:\Theta}\mu^+ - \|\mu^+\|\cdot\nu_0) - (\mathcal{P}^{s:\Theta}\mu^- - \|\mu^-\|\cdot\nu_0)\| && \text{(B.10)}\\
&\leq \|\underbrace{\mathcal{P}^{s:\Theta}\mu^+ - \|\mu^+\|\cdot\nu_0}_{\geq 0 \text{ for all events } B\in\mathcal{S}}\| + \|\underbrace{\mathcal{P}^{s:\Theta}\mu^- - \|\mu^-\|\cdot\nu_0}_{\geq 0 \text{ for all events } B\in\mathcal{S}}\| && \text{(B.11)}\\
&= \|\mathcal{P}^{s:\Theta}\mu^+\| - \|\mu^+\|\cdot\|\nu_0\| \ + \ \|\mathcal{P}^{s:\Theta}\mu^-\| - \|\mu^-\|\cdot\|\nu_0\| && \text{(B.12)}\\
&= (1-\|\nu_0\|)\cdot\|\mu^+\| \ + \ (1-\|\nu_0\|)\cdot\|\mu^-\| && \text{(B.13)}\\
&= (1-\epsilon^{\Theta})\cdot\left(\|\mu^+\| + \|\mu^-\|\right) && \text{(B.14)}\\
&= (1-\epsilon^{\Theta})\cdot\|\phi_1 - \phi_2\| \ . && \text{(B.15)}
\end{aligned}$$

The equality in (B.9) follows from linearity of transition probability kernels. The transition to (B.11) is an application of the triangle inequality. The transition to (B.12) uses the fact that both $\mathcal{P}^{s:\Theta}\mu^+ - \|\mu^+\|\cdot\nu_0$ and $\mathcal{P}^{s:\Theta}\mu^- - \|\mu^-\|\cdot\nu_0$ are non-negative: this follows from Proposition 1, which ensures that the measure $\mathcal{P}^{s:\Theta}\mu^+$ has at least mass $\|\mu^+\|\cdot\epsilon^{\Theta}$ at the resting state $\mathbf{0}$ and, hence, for any (non-negative) measure ν,

$$\mathcal{P}^{s:\Theta}\nu \geq \|\nu\|\cdot\nu_0 \ . \tag{B.16}$$

Finally, note that (B.13) uses a general property of transition probability kernels $\mathcal{P}$, which ensures that $\|\mathcal{P}\nu\| = \|\nu\|$, for any non-negative measure ν. □

Note that the above Contraction Lemma which holds for spiking neural networks has some similarities to Lemma 1 in (Maass and Sontag, 1999) who analyzed artificial analog neural networks in discrete time.

B.2 Proof of Theorem 1

We divided the precise formulation of Theorem 1 into two Lemmata: Lemma 2 is a precise formulation for the case where inputs are fixed (e.g. fixed input rates). Lemma 3 in the next section corresponds to the case where input rates are controlled by a Markov process. The precise assumptions on the network model required for both Lemmata are described above (see “Scope of theoretical results”).

Proof of Theorem 1 for fixed input rates

Here we assume that the vector of inputs $x(t)$ provided to the network is kept fixed during a trial. Concretely, this is for example the case if there is a set of input neurons whose rates are fixed. In this case, $x(t)$ is a vector of input rates, which remains constant over time. The input neurons are formally considered part of the network in this case. Alternatively, a constant x could correspond to constant currents which are injected into a subset of neurons.

Under constant input conditions, $x(t) \equiv x$, the dynamics of the process is time-homogeneous: the transition probability kernels are invariant to time-shifts, i.e.

$$\mathcal{P}^{s_1:t}\phi = \mathcal{P}^{s_2:t}\phi, \quad s_1, s_2 \geq 0,\ t > 0 \ . \tag{B.17}$$

Lemma 2 *Let $x(t) \equiv x$. Then the Markov process Y has a unique stationary distribution π, to which it converges exponentially fast,*

$$\|\mathcal{P}^t(y_0, \cdot) - \pi\| \leq 2 \cdot (1 - \epsilon^{\Theta})^{t-1} \ , \quad t \geq 0 \ , \tag{B.18}$$

from any initial Markov state $y_0 \in S$.

Proof: Y is clearly non-explosive, aperiodic and stochastically continuous (cf. (Borovkov et al., 2012)). To prove exponential ergodicity it thus suffices to show that some skeleton chain is geometrically ergodic (see for example Theorem 18.1 in (Borovkov, 1998)). The skeleton chain $Y_{\Theta n},\ n \in \mathbb{N}$, with transition probability kernel $\mathcal{P}^{\Theta}$ is aperiodic and irreducible and hence has a unique stationary distribution π. Then, through recursive application of Lemma 1 with $\phi_2 = \pi$,

$$\|\mathcal{P}^{\Theta n}\phi_1 - \mathcal{P}^{\Theta n}\pi\| \leq (1 - \epsilon^{\Theta})^n \cdot \|\phi_1 - \pi\| \ , \tag{B.19}$$

$$\|\mathcal{P}^{\Theta n}\phi_1 - \pi\| \leq 2 \cdot (1 - \epsilon^{\Theta})^n \ , \tag{B.20}$$

proving geometric ergodicity of the skeleton chain, and thus exponential ergodicity of Y. The quantitative convergence bound follows from (B.20) by choosing a singleton y_0 as initial distribution, and using the general fact that for any transition probability kernel $\mathcal{P}$ and distributions ϕ_1 and ϕ_2,

$$\|\mathcal{P}\phi_1 - \mathcal{P}\phi_2\| \leq \|\phi_1 - \phi_2\| \ , \tag{B.21}$$

thus guaranteeing that the total variation distance does not (temporarily) grow between Θn and $\Theta(n+1)$. $\square$

Lemma 2 provides a general ergodicity result for the considered class of stochastic spiking networks in the presence of fixed input rates x. The proof relies on two key properties of stochastic spiking networks: aperiodicity and irreducibility. These properties can be understood intuitively in the context of Figure 2.1H. If the intrinsic network dynamics was not aperiodic, for example, then one might be able to observe oscillating pattern frequencies over time (as in Figure 2.4C). Lemma 2 proves that this cannot occur in stochastic spiking networks as long as input rates are fixed. Oscillating pattern frequencies can indeed only emerge when input rates are themselves periodically changing (see Theorem 2 and Figure 2.4). If the network dynamics was not irreducible on the other hand, i.e. if there were network states which are unreachable from some other network states, then pattern frequencies could potentially be observed to converge to different fixed points for different initial states (e.g. the two lines in Figure 2.1H settling at different values). This cannot occur in stochastic spiking networks due to Proposition 1 which guarantees that the state space is connected through the resting state $\mathbf{0}$.

Note that, although aperiodicity and irreducibility are well known necessary and sufficient conditions for ergodicity in discrete time Markov chains on finite state spaces, they are not sufficient for exponential ergodicity in continuous time Markov processes on general state spaces (see (Down et al., 1995) for precise definitions of ϕ-irreducibility and aperiodicity for such processes). Additional conditions in this more complex case which ensure exponential ergodicity, such as nonexplosivity, stochastic continuity and geometric ergodicity of a skeleton chain, have also been taken into account in the proof for Lemma 2 (i.e. stochastic spiking networks also meet these additional criteria).

Lemma 2 constitutes a proof for Theorem 1 for fixed input rates x. In the main text we refer to the stationary distribution of the circuit C under fixed input x as $p_C(y|x)$. The proof above guarantees a stationary distribution for both Markov and simple states. In the main text y refers to the simple network state y_S if not stated otherwise.

Proof of Theorem 1 for input rates controlled by a Markov process

Fixed input assumptions may often hold for the external input $x(t)$, driving a stochastic computation in a neural system C, only approximately. Stochastic fluctuations on various spatial and temporal scales may be present in the input. In addition, inputs may have their own short-term stochastic dynamics: Imagine, for example, a visual scene of randomly moving dots. Despite the presence of such short-term dynamical features in the input, in many cases one may still suspect that network state distributions converge. Indeed, below we generalize the convergence results from the constant case to the quite large class of stochastic (and stochastically changing) inputs which are generated by a uniformly ergodic Markov process. Uniform ergodicity is defined as exponential ergodicity (exponentially fast convergence to a unique stationary distribution) with convergence constants which apply uniformly to all initial states (Down et al., 1995) (this holds for example for the convergence constants in Lemma 2).

Let X be a time-homogeneous *input Markov process*, in the sense that the input trajectory $x(t)$ provided to the network C is itself generated randomly from a Markov process X. Let $(Q, \mathcal{Q})$ be the (measurable) state space of X. Then define a joint input/network Markov process Z on the state space $(Q \times S,\ \ \sigma(\mathcal{Q} \times \mathcal{S}))$, where $\sigma(\cdot)$ denotes the σ-algebra generated by $\cdot$. Further definitions for Z are analogous to those introduced for Y.

Lemma 3 *If the input process X is uniformly ergodic, then the joint Markov process Z has a unique stationary distribution $\hat{\pi}$ on the joint input/network state space, to which convergence occurs exponentially fast, i.e. there exist constants $C < \infty, \rho < 1$, such that*

$$\|\mathcal{P}^t(z_0, \cdot) - \hat{\pi}\| \leq C \cdot \rho^t \ , t \geq 0 \ , \tag{B.22}$$

for any initial state z_0 of the joint Markov process Z.

Proof: If X and Y were entirely independent processes (if X did not influence Y) then the joint process Z would automatically be exponentially ergodic if both X and Y are. Although in the present case Y is not independent of X, a weaker version of independence applies: the return probability to the resting state $Y(t) = \mathbf{0}$ during $(t - \Theta, t]$ is at least ϵ^{Θ} *regardless* of the input trajectory of X during that time. This property can be exploited to show that the distribution of hitting times to a joint resting state has an exponential bound. It follows that the joint process is exponentially ergodic. A detailed proof is given in the next section. □

The second part of Theorem 1 (exponentially fast convergence for the case of external input generated by an ergodic Markov process) follows from Lemma 3. Note that in the main text we slightly abuse the notation $p_C(y|x)$ for the dynamic case to indicate the stationary distribution over network states y, where x denotes a specific Markov process controlling the inputs.

Detailed proof of Lemma 3

We have split the proof of Lemma 3 into proofs of four auxiliary claims (Propositions 2-5). Consider the following variations of Proposition 1, which hold for the Markov process Z describing the joint dynamics of input and network states. Let $\{x(t)\}$ denote a particular input sequence defined for $t \geq 0$ (a realization of the input process X) and $y_0 \in S$ an initial network Markov state (with $\Theta \geq \underline{\Theta}$) at time $s \geq 0$. Then

$$\mathbb{P}(Y(s + \Theta) = \mathbf{0} \mid Y(s) = y_0, X = \{x(t)\}) \geq \epsilon^{\Theta} \ , \tag{B.23}$$

$$\mathbb{P}(Y(s + \Theta) = \mathbf{0} \mid X = \{x(t)\}) \geq \epsilon^{\Theta} \ . \tag{B.24}$$

It is easy to show that these properties, together with the fact that X is uniformly ergodic, ensure that Z is irreducible and aperiodic. Hence, to prove exponential ergodicity of Z it suffices to show that some skeleton chain is geometrically ergodic (Down et al., 1995). To that end, we will consider the skeleton chain $Z_{\Theta n}$, $n \in \mathbb{N}$ and prove geometric ergodicity by showing that the hitting time distribution $\mathbb{P}_{\tau_C}(\tau_C)$ to a *small set* C on the joint state space $Q \times S$ of input and network states admits an exponential bound.

The *hitting time* τ_D to some set D on the input state space Q is defined as

$$\tau_D = \min\left\{n \in \mathbb{N}^+ : X_{\Theta n} \in D\right\} \ . \tag{B.25}$$

For notational convenience we abbreviate in the following $\tau = \tau_D$. Due to uniform ergodicity of X (which implies Harris recurrence (Down et al., 1995)), there exists some set D to which the hitting time τ is finite ($< \infty$) from any initial state, with probability one (Meyn and Tweedie, 1993). Furthermore, there exists according to (Down et al., 1995) a *small set* D and constants $\kappa > 1$ and $1 \leq V < \infty$, such that

$$\forall x_0 \in Q : \quad \mathbb{E}\left[\kappa^\tau \mid X(0) = x_0\right] < V \ . \tag{B.26}$$

This implies that there exists a small set D on the input state space Q which can not only be reached in finite time from any initial input state x_0, but for which the hitting time distribution to D has also finite mean and variance (and finite higher-order moments). At least one pair of constants κ and V which fulfills (B.26) is guaranteed to exist, but in fact the following Proposition shows that one can specify a particular desired bound on the right-hand side (for reasons which will become clear later), and find a matching λ on the left-hand side.

Proposition 2 *There exists a $\lambda > 1$, such that*

$$\forall x_0 \in Q : \quad \mathbb{E}\left[\lambda^\tau \mid X(0) = x_0\right] < (1 - \epsilon^\Theta)^{-1/2}. \tag{B.27}$$

Proof: Define $v(\lambda) := \mathbb{E}\left[\lambda^\tau \mid X(0) = x_0\right]$. Let κ and V be any valid pair of constants which fulfills (B.26). The trivial case is $v(\kappa) < (1 - \epsilon^\Theta)^{-1/2}$. In the remainder of the proof it is assumed that κ is "too large", such that $v(\kappa) \geq (1 - \epsilon^\Theta)^{-1/2}$. By definition of the exponential function, for any $\lambda > 0$,

$$v(\lambda) = \mathbb{E}\left[\lambda^\tau | x_0\right] = \mathbb{E}\left[\sum_{n=0}^{\infty} \frac{(\log\lambda)^n \tau^n}{n!} | x_0\right] \tag{B.28}$$

$$= \sum_{\tau=0}^{\infty}\sum_{n=0}^{\infty} \mathbb{P}_{\tau|x_0}(\tau)\frac{(\log\lambda)^n \tau^n}{n!} \ . \tag{B.29}$$

By Tonelli's theorem, since all summands are non-negative, the order of the double sum can be exchanged:

$$v(\lambda) = \sum_{n=0}^{\infty}\sum_{\tau=0}^{\infty}\mathbb{P}_{\tau|x_0}(\tau)\frac{(\log\lambda)^n\tau^n}{n!} \tag{B.30}$$

$$= \sum_{n=0}^{\infty}\frac{(\log\lambda)^n}{n!}\mathbb{E}[\tau^n|x_0] \quad . \tag{B.31}$$

Note that $\mathbb{E}[\tau^n|x_0]$ are the moments of the distribution $\mathbb{P}_{\tau|x_0}[\tau]$. By uniform ergodicity of X, all moments must exist, and in addition there exists a $\kappa > 1$ such that $v(\kappa) < \infty$. It is straightforward to see that the series then converges for all $1 \le \lambda \le \kappa$, such that $v(\lambda)$ is continuous on $[1,\kappa]$. Finally, since $v(1) = 1$ and $v(\kappa) \ge (1-\epsilon^{\Theta})^{-1/2}$, by the intermediate value theorem there exists some $1 < \lambda < \kappa$ such that $v(\lambda) = (1 + (1-\epsilon^{\Theta})^{-1/2})/2$. □

Denote by $\tau^{(m)}$ the time at which the skeleton chain $X_{\Theta n}$ visits the small set D for the m-th time:

$$\tau^{(m)} = \min\{n \in \mathbb{N}^+ : \exists\, n_1 < n_2 < \cdots < n_m \le n \in \mathbb{N} : X_{\Theta\cdot n_k} \in D, k \in \{1,\ldots,m\}\}. \tag{B.32}$$

Furthermore, denote by $\delta^{(m)}$ the time between the $(m-1)$-th and m-th visit:

$$\delta^{(1)} := \tau^{(1)}, \tag{B.33}$$

$$\delta^{(m)} := \tau^{(m)} - \tau^{(m-1)}, \quad m > 1. \tag{B.34}$$

According to this definition, one can express the hitting time of degree m as $\tau^{(m)} = \sum_{k=1}^{m}\delta^{(k)}$. The following Proposition extends the exponential bound on the first hitting time to hitting times of higher degrees.

Proposition 3 *There exists a $\lambda > 1$, such that,*

$$\forall x_0 \in Q: \quad \mathbb{E}\left[\lambda^{\tau^{(m)}} \mid X(0) = x_0\right] < (1-\epsilon^{\Theta})^{-m/2} \quad . \tag{B.35}$$

Proof:

$$\mathbb{E}\left[\lambda^{\tau^{(m)}}|y_0\right] = \int d\mathbb{P}_{\delta^{(1\ldots m)}|y_0}(\delta^{(1\ldots m)})\cdot\lambda^{\sum_{k=1}^{m}\delta^{(k)}} \tag{B.36}$$

$$= \int d\mathbb{P}_{\delta^{(1)}|y_0}(\delta^{(1)})\cdot\lambda^{\delta^{(1)}}\int d\mathbb{P}_{\delta^{(2)}|y_0,\delta^{(1)}}(\delta^{(2)})\cdot\lambda^{\delta^{(2)}}\cdots$$

$$\cdots\int d\mathbb{P}_{\delta^{(m)}|y_0,\delta^{(1\ldots m-1)}}(\delta^{(m)})\cdot\lambda^{\delta^{(m)}} \tag{B.37}$$

$$< \left[(1-\epsilon^{\Theta})^{-1/2}\right]^m \quad . \tag{B.38}$$

□

Let τ_C be the hitting time to the small set $C = D \times \mathbf{0}$ on the joint state space $Q \times S$ of input and network states,

$$\tau_C = \min\{n \in \mathbb{N}^+ : X_{\Theta n} \in D, Y_{\Theta n} = \mathbf{0}\} \ . \tag{B.39}$$

Furthermore, let R be the number of visits to the small set D prior to and including time τ_C,

$$R = \max\{m \in \mathbb{N}^+ : \exists\, n_1 < n_2 < \cdots < n_m \leq \tau_C \in \mathbb{N} : X_{\Theta n_k} \in D,\ k \in \{1, \ldots, m\}\}. \tag{B.40}$$

Proposition 4 *For any input trajectory $x(t)$ and any initial network state $y_0 \in S$,*

$$\mathbb{P}(R = m \mid Y(0) = y_0, X = \{x(t)\}) \leq (1 - \epsilon^{\Theta})^{m-1} \ . \tag{B.41}$$

This follows from (B.23) and (B.24) which ensure that whenever the input process visits the small set D, there is also a small probability that the network is in the resting state.

Proposition 5 *There exists a $\lambda > 1$ and a constant $W < \infty$ such that,*

$$\forall z_0 \in (Q \times S): \quad \mathbb{E}\left[\lambda^{\tau_C} \mid Z(0) = z_0\right] < W \ . \tag{B.42}$$

Proof: Let $\boldsymbol{\tau} = (\tau^{(m)},\ m \in \mathbb{N}^+)$. Choose some λ which fulfills Proposition 3.

$$\begin{aligned}
\mathbb{E}\left[\lambda^{\tau_C}|z_0\right] &= \int d\mathbb{P}_{\boldsymbol{\tau},R|z_0}(\boldsymbol{\tau}, m)\, \lambda^{\tau^{(m)}} && \text{(B.43)}\\
&= \int d\mathbb{P}_{\boldsymbol{\tau}|z_0}(\boldsymbol{\tau}) \int d\mathbb{P}_{X|z_0,\boldsymbol{\tau}}(\{x(t)\}) \sum_{m=1}^{\infty} \mathbb{P}(R = m|y_0, \{x(t)\})\, \lambda^{\tau^{(m)}} && \text{(B.44)}\\
&\leq \int d\mathbb{P}_{\boldsymbol{\tau}|z_0}(\boldsymbol{\tau}) \sum_{m=1}^{\infty} (1 - \epsilon^{\Theta})^{m-1} \lambda^{\tau^{(m)}} && \text{(B.45)}\\
&= \int d\mathbb{P}_{\boldsymbol{\tau}|x_0}(\boldsymbol{\tau}) \sum_{m=1}^{\infty} (1 - \epsilon^{\Theta})^{m-1} \lambda^{\tau^{(m)}} && \text{(B.46)}\\
&= \sum_{m=1}^{\infty} (1 - \epsilon^{\Theta})^{m-1} \int d\mathbb{P}_{\tau^{(m)}|x_0}(\tau^{(m)}) \cdot \lambda^{\tau^{(m)}} && \text{(B.47)}\\
&< \sum_{m=1}^{\infty} (1 - \epsilon^{\Theta})^{m-1} (1 - \epsilon^{\Theta})^{-m/2} && \text{(B.48)}\\
&= \sum_{m=1}^{\infty} (1 - \epsilon^{\Theta})^{(m/2)-1} =: W < \infty \ . && \text{(B.49)}
\end{aligned}$$

□

By Proposition 5, Z is exponentially ergodic (Down et al., 1995). This completes the proof of Lemma 3.

Distribution of trajectories of network states

The Markov states $y_{M:\Theta}(t)$ are segments of spiking trajectories of length Θ. Hence, all statements developed above apply to convergence of the distribution over these (short) spiking trajectories. If one is interested in the convergence of longer trajectories, the simplest option is to choose a larger Θ, since any finite $\Theta \geq \underline{\Theta}$ is admissible, and all convergence results readily extend to trajectories of any finite length. A limitation of this approach is that the quantitative convergence statements will suffer from making Θ too large, since convergence rates scale approximately with ϵ^{Θ} (and $\epsilon \ll 1$). Hence, in practice, empirical convergence tests are required to make statements about specific circuits.

B.3 Proof of Theorem 2

If the input sequence is periodic with period L, i.e. $x(t) = x(t+L)$ for all $t \geq 0$, then the Markov process Y will be time-periodic, in the sense that transition kernels are invariant to shifts which are multiples of the period L:

$$\mathcal{P}^{s:t}\phi = \mathcal{P}^{s+kL:t}\phi, \quad s \geq 0,\ t > 0,\ k \in \mathbb{N} \ . \tag{B.50}$$

This implies the following result, which is a more precise version of Theorem 2:

Lemma 4 *Under periodic input, i.e. $x(t) = x(t+L)$ for all $t \geq 0$ with some $L \geq \Theta$, the time-periodic Markov process Y with period L has a periodically stationary distribution $\tilde{\pi}_l$, to which convergence occurs exponentially fast from any initial state. In particular, for every $0 \leq l < L$ there exists a unique stationary distribution $\tilde{\pi}_l$ such that,*

$$\|\mathcal{P}^{l+Ln}(y_0, \cdot) - \tilde{\pi}_l\| \leq 2 \cdot (1 - \epsilon^{\Theta})^{\lfloor \frac{L}{\Theta} \rfloor \cdot n} \ , \quad n \in \mathbb{N} \ , \tag{B.51}$$

from any initial Markov state y_0.

Proof: For each $0 \leq l < L$ there exists a skeleton chain $Y_{l+Ln},\ n \in \mathbb{N}$, with transition probability kernel $\mathcal{P}^{l:L} = \mathcal{P}^{l+L:L} = \mathcal{P}^{l+2L:L} = \dots$, which is time-homogeneous, irreducible, and aperiodic and thus has a unique stationary distribution $\tilde{\pi}_l$. An application of $\mathcal{P}^{l:L}$, which corresponds to a full period, decreases the total variation distance to $\tilde{\pi}_l$ by at least $(1 - \epsilon^{\Theta})^{\lfloor \frac{L}{\Theta} \rfloor}$:

$$\|\mathcal{P}^{l:L}\phi_1 - \tilde{\pi}_l\| = \|\mathcal{P}^{l:L}\phi_1 - \mathcal{P}^{l:L}\tilde{\pi}_l\| \tag{B.52}$$

$$\leq \|\mathcal{P}^{l:\Theta\lfloor \frac{L}{\Theta} \rfloor}\phi_1 - \mathcal{P}^{l:\Theta\lfloor \frac{L}{\Theta} \rfloor}\tilde{\pi}_l\| \tag{B.53}$$

$$\leq (1 - \epsilon^{\Theta})^{\lfloor \frac{L}{\Theta} \rfloor} \cdot \|\phi_1 - \tilde{\pi}_l\| \ . \tag{B.54}$$

The first inequality follows from the fact that applying the remaining $\mathcal{P}^{l+\Theta\lfloor \frac{L}{\Theta} \rfloor : L - \Theta\lfloor \frac{L}{\Theta} \rfloor}$ can only further decrease the total variation distance between the two distributions, according to (B.21). The second inequality is due to Lemma 1.

Lemma 4 then follows from recursive application of (B.52)-(B.54) for multiple periods, and choosing a singleton y_0 as initial distribution. □

In the main text, we use the notation $p_{C,l}(y|x)$ for a phase-specific stationary distribution, where x denotes a specific periodic input sequence.

B.4 Relation to previous theoretical work

Previous work on the question whether states of spiking neural networks might converge to a unique stationary distribution had focused on the case where neuronal integration of incoming spikes occurs in a linear fashion, i.e., linear subthreshold dynamics followed by a single output non-linearity (Brémaud and Massoulié, 1996; Borovkov et al., 2012). In addition these earlier publications did not allow for the experimentally observed short term dynamics of synapses. The earlier publication (Brémaud and Massoulié, 1996) had studied this question as a special case of the mathematical framework of non-linear Hawkes processes, a class of mutually exciting point processes (see also (Massoulié, 1998)). The authors had arrived for the more restricted type of neurons which they considered at exponential convergence guarantees under a similar set of assumptions as in this article, in particular bounded memory and bounded instantaneous firing rates (and these results can thus be seen as a special case of Theorem 1, for the case of constant external input). (Brémaud and Massoulié, 1996) also derived convergence results for linearly integrating neurons with unbounded memory dynamics under a different set of assumptions, in particular Lipschitz conditions on the output non-linearity and constraints on the effective connectivity matrix of the network. Whether such alternative set of assumptions can be found also in the context of non-linear integration of incoming spikes (needed e.g. for synaptic short-time dynamics or dendritic non-linearities) remains an open question.

The recent publication (Borovkov et al., 2012) also focused on neurons with linear sub-threshold dynamics followed by an output non-linearity (termed there non-linear Poisson neurons) with static synapses, and extended the convergence results of (Brémaud and Massoulié, 1996) to networks with Hebbian learning mechanisms. In addition, an important methodological innovation by (Borovkov et al., 2012) was the introduction of spike history states (which are equivalent to the Markov states $y_M(t)$ in this article) which allowed them to study convergence in the framework of general Markov processes (in contrast to point processes in (Brémaud and Massoulié, 1996)). Theorem 1 in this article contains as a special case the convergence results of (Borovkov et al., 2012) for their Model I (non-linear Poisson neurons in the absence of Hebbian learning). We note that although (Borovkov et al., 2012) focused on neurons with linear sub-threshold dynamics (and required that firing rates are strictly greater than 0), their method of proof for Model I could be readily extended to cover also non-linear sub-threshold dynamics to yield the first part of our Theorem 1 (the case where inputs have constant firing rates).

We are not aware of previous work that studied convergence in spiking networks with dynamic synapses, or in the presence of stochastic or periodic inputs (see the second part of Theorem 1 concerning Markov processes as input, and Theorem 2). We further note that our method of proof builds on a new and rather intuitive intermediate result, Lemma 1 (Contraction Lemma), which may be useful in its own right for two reasons. On the one hand it provides more direct insight into the mechanisms responsible for convergence (the contraction between any two distributions). On the other hand, it holds regardless of the input trajectory $x(t)$, and hence has in fact an even larger scope of applicability than Theorem 1 and 2. Hence, Lemma 1 could be, for example, applied to study non-stationary evolutions of state distributions in response to arbitrary input trajectories.

B.5 Extracting knowledge from internally stored distributions

A key advantage of sample-based representations of probability distributions is that probabilities and expected values are in principle straightforward to estimate: To estimate the expected value $\mathbb{E}_{p(y)}[g(y)]$ of a function $g(y)$ under a distribution $p(y)$ from a number of samples $y^1, \ldots, y^T$, simply apply the function to each sample and compute the time average $\frac{1}{T}\sum_{t=1}^{T} g(y^t)$. As long as the samples y^t are distributed according to $p(y)$, either independently drawn, or as the result of an ergodic Markov chain/process with stationary distribution $p(y)$, this estimate is guaranteed to converge to the correct value as one increases the number of samples (Gray, 2009), i.e. $\lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} g(y^t) = \mathbb{E}_{p(y)}[g(y)]$. Estimates based on a finite observation window represent an approximation to this exact value.

Under the mild assumptions of Theorem 1 the dynamics of a stochastic spiking network in response to an input x are exponentially ergodic and there exists a unique stationary distribution $p_C(y|x)$, according to which network states $y(t)$ are distributed. Hence, the expected value $\mathbb{E}_{p_C(y|x)}[g(y)]$ of any function $g(y)$ under the stationary distribution $p_C(y|x)$ can be estimated by computing the sample-based time average

$$\frac{1}{T}\int_0^T g(y(t))\, dt \quad . \tag{B.55}$$

This approach can also be used to estimate marginal probabilities, since probabilities can be expressed as expected values, for example,

$$p_C(y_1 = 1|x) = \mathbb{E}_{p_C(y|x)}[\delta(y_1, 1)] \quad \text{, or} \tag{B.56}$$

$$p_C(y_1 = 1, y_2 = 0, y_3 = 1|x) = \mathbb{E}_{p_C(y|x)}[\delta(y_1, 1)\cdot\delta(y_2, 0)\cdot\delta(y_3, 1)] \quad , \tag{B.57}$$

where $\delta(a, b) = 1$ if $a = b$ and 0 otherwise. Hence, in order to estimate the probability $p_C(y_1 = 1|x)$ it suffices to measure the relative time the neuron spends in its active state, i.e. $\frac{1}{T}\int_0^T \delta(y_1, 1)\, dt$. Similarly, to estimate the probability $p_C(y_1 = 1, y_2 = 0, y_3 = 1|x)$ it is sufficient to keep track of the relative frequency of the pattern $(1, 0, 1)$, by computing $\frac{1}{T}\int_0^T \delta(y_1, 1)\cdot\delta(y_2, 0)\cdot\delta(y_3, 1)\, dt$.

B.6 Simulations of data-based cortical microcircuit models

All simulations of microcircuit models for Figures 2.1-2.4 were carried out in PCSIM (Pecevski et al., 2009). A time step of 1 ms was chosen throughout. Further analysis of spike trains was performed in Python (van Rossum and Drake, 2001).

Stochastic neuron model

A stochastic variation of the leaky integrate-and-fire model with conductance-based integration of synaptic inputs was used, for both excitatory and inhibitory neurons. Sub-threshold dynamics of the membrane potential $u(t)$ was defined according to a standard leaky integration model with conductance-based synapses (Gerstner and Kistler, 2002), using passive membrane parameters $R = 60\ \text{M}\Omega, C = 0.35$ nF and a resting potential $V_{\text{resting}} = -60$ mV. At simulation start, initial potentials were randomly chosen from $[-65, -55]$ mV. Reversal potentials for excitatory synapses and inhibitory synapses were set to 0 mV and -75 mV, respectively. Neuronal noise was modeled by a voltage-dependent instantaneous probability of firing (instead of a fixed threshold) (Jolivet et al., 2006),

$$\frac{p(\text{neuron spikes in } [t, t+\delta t))}{\delta t} = \frac{1}{\tau_s} e^{(u(t)-\vartheta)/\delta u}, \tag{B.58}$$

for $\delta t \to 0$, with parameters $\tau_s = 19$ ms, $\delta u = 4$ mV taken from (Jolivet et al., 2006). In contrast to (Jolivet et al., 2006) we used a non-adaptive threshold, $\vartheta = -45$ mV. After a spike, a neuron enters an absolute refractory period of 3 ms. Thereafter, the membrane is reset to the resting potential and leaky integration is continued. Altogether, the resulting neuronal spiking mechanism is consistent with the theoretical noise model I described in equation (B.2).

Note that Theorem 1 also holds for substantially more complex multi-compartment neuron models incorporating, for example, data on signal integration in the dendritic tuft of pyramidal cells (Larkum, 2013; Jiang et al., 2013), and data on Ca-spikes in pyramidal cells on layer 5 (Larkum, 2012), but we have not yet integrated these into the simulated microcircuit model because of a lack of coherent quantitative data for all the neuron types involved.

Synaptic short-term plasticity

The short-term dynamics of synapses in all data-based simulations was modeled according to (Maass and Markram, 2002; Markram et al., 1998). The model predicts that at a synapse with "weight" w the amplitude A_k of the k^{th} spike in a spike train with interspike intervals $\Delta_1, \Delta_2, .., \Delta_{k-1}$ is given by,

$$\begin{aligned} A_k &= w \cdot u_k \cdot R_k \quad , \\ u_k &= U + u_{k-1}(1-U)\exp^{-\Delta_{k-1}/F} \quad , \\ R_k &= 1 + (R_{k-1} - u_{k-1}R_{k-1} - 1)\exp^{-\Delta_{k-1}/D} \quad , \end{aligned} \tag{B.59}$$

where the hidden dynamic variables $u_k \in [0, 1]$ and $R_k \in [0, 1]$ are initialized for the first spike to $u_1 = U$ and $R_1 = 1$. The parameters U,D and F represent the utilization of the synaptic efficacy of the first spike after a resting state, the recovery and the facilitation time constants, respectively. These parameters were set based on experimental data on short-term plasticity in dependence of pre- and post-synaptic neuron (excitatory or inhibitory) as in (Haeusler and Maass, 2007) (see in particular Table 1 in this reference), by randomly drawing for each neuron values for U, D, and F from corresponding data-based Gaussian distributions.

Connectivity and synaptic parameters

Synaptic parameters and connectivity rules for the data-based cortical column model were taken from (Haeusler and Maass, 2007), see Figure 2.1A. In particular, we adopted from (Haeusler and Maass, 2007) the connection probabilities and transmission delays for each type of connection (EE, EI, IE, II) and each cortical layer ((Haeusler and Maass, 2007), Figure 1), as well as short-term plasticity parameters. Furthermore, synaptic efficacies of individual synapses were drawn from Gamma distributions with data-based means and variances for each type of connection (EE, EI, IE, II) taken from (Haeusler and Maass, 2007). Two input streams were connected to the microcircuit, each consisting of 40 input neurons. In contrast to (Haeusler and Maass, 2007) we used rate-based Poisson input neurons instead of injecting “frozen” spike patterns. Background synaptic inputs were emulated as in (Haeusler and Maass, 2007) via background input currents to each neuron, with conductances modeled according to (Destexhe et al., 2001). To adjust connectivity for cortical microcircuit models of different sizes, we also adopted the method proposed by (Haeusler and Maass, 2007), in which recurrent weights are scaled inversely proportional to network size.

We tested the validity of our cortical microcircuit model by comparing the average activity of different layers (see Figure 2.2A) under various conditions against the values reported by (Haeusler and Maass, 2007). We confirmed that all layers exhibited very similar average activity to (Haeusler and Maass, 2007) under all considered conditions.

B.7 Details to small microcircuit model in Figure 2.1

The small cortical microcircuit model of Figure 2.1B was constructed based on the cortical column template of (Haeusler and Maass, 2007): Synaptic connections between neurons and their weights were chosen to approximately reflect connection probabilities and mean synaptic strengths of the cortical column template (Haeusler and Maass, 2007). Due to the very small size of this network, the resulting dynamics was not immediately satisfactory (for example, the influence of inputs on Layer 5 neurons was too weak). To shift the circuit into a more responsive regime, we manually adjusted a few synaptic weights and neuronal excitabilities. In particular, we injected small constant currents into some of the neurons to modulate their intrinsic excitability. Furthermore, to increase activity and correlations between highlighted neurons 2, 7 and 8, we increased synaptic weights $8 \rightarrow 2$ and $8 \rightarrow 7$ by factors 5 and 10, respectively. To set the initial Markov state of the network, preparatory input was shown for 1 s before the actual start of the simulation. Two different preparatory inputs were injected to set the two initial states considered in Figure 2.1F-H (first: i_1 at 100 Hz, i_2 at 100 Hz, second: both i_1 and i_2 at 0 Hz). To reproduce the same initial Markov state in multiple trials (for example the two trials shown in Figure 2.1F), the same random seed was used during the preparatory phase for these trials. The random seed was then reinitialized at $t = 0$ to different values for each trial.

B.8 Estimates of required computation time

Gelman-Rubin univariate and multivariate analysis

Various methods have been developed for measuring convergence speed to a stationary distribution in the context of Markov chain Monte Carlo sampling (Cowles and Carlin, 1996; Brooks and Roberts, 1998; El Adlouni et al., 2006). The Gelman Rubin diagnostic, which we adopted in this article, is one of the most widely used methods (Gelman and Rubin, 1992; Brooks and Roberts, 1998; Brooks et al., 2010; Gjoka et al., 2010), besides other popular methods such as the diagnostics by Raftery and Lewis (Raftery et al., 1992) and by Geweke (Geweke, 1991). We remark that the consensus in the literature is that no single method is perfect in general. Some attractive properties of the Gelman Rubin method are general applicability to any MCMC system (some other methods only work, for example, in the context of Gibbs sampling), ease of use, ease of implementation, computational efficiency, and the fact that results are quantitative (in contrast to graphical diagnostics) (Cowles and Carlin, 1996; Brooks and Roberts, 1998).

The Gelman-Rubin convergence diagnostic (Gelman and Rubin, 1992) takes as input samples from m different runs (trials/chains/sequences) produced by the same system, started from different initial states. The method was originally developed for discrete-time systems in the context of Markov Chain Monte Carlo sampling. Our simulations use a time step of 1 ms, so we simply treat each simulation step as one discrete time step in a Markov chain. The Gelman-Rubin method produces as output the potential scale reduction factor $\hat{R}(t)$ as a function of time t. The scale reduction factor $\hat{R}(t)$ is an indicator for whether or not the system has converged at time t. High values $\gg 1$ indicate that more time is needed until convergence, while values close to 1 suggest that convergence has (almost) taken place.

For computing the scale reduction factor $\hat{R}(t)$ at time t, samples from the period $[t, 2t]$ from each run of the network are taken into account. In the univariate case one focuses on a particular single variable (such as the marginal simple state of a single neuron, or the simple state of a "random readout" neuron as in the solid lines of Figure 2.2G). Let n be the number of samples obtained from the period $[t, 2t]$

from each of the simulations. Then one defines

$$\hat{R}(t) = \frac{n-1}{n} + \frac{m+1}{mn}\frac{B(t)}{W(t)} \ , \tag{B.60}$$

where $B(t)$ and $W(t)$ are between and within-sequence variances, respectively, which can be computed as described in (Gelman and Rubin, 1992), based on samples taken from the time period $[t, 2t]$. In the rare event of $W = 0$, which happens for example if a neuron never fires and hence its state is constant across all runs, we set $\hat{R}$ to 1.

An unfortunate source of confusion is the fact that Gelman and Rubin (Gelman and Rubin, 1992) originally introduced $\hat{R}$ in its “variance” form equivalent to equation (B.60), but later in (Gelman et al., 2004; Brooks et al., 2010) altered this definition and defined $\hat{R}$ as the square root of (B.60). This issue is particularly critical when considering threshold values for $\hat{R}$: a threshold of 1.2 was suggested in the context of the original definition (Kass et al., 1998). Later, a typical threshold of 1.1 was suggested, but this lower threshold applied to the modified definition (Gelman et al., 2004; Brooks et al., 2010). Squaring this apparently lower threshold yields again a typical threshold of approximately 1.2.

In the multivariate case (e.g. when analyzing convergence of the vector-valued simple state of a small subset of neurons as in the dotted lines of Figure 2.2G) one takes vector-valued (d-dimensional) samples, and computes the multivariate potential scale reduction factor $\hat{R}^d(t)$ according to:

$$\hat{R}^d(t) = \frac{n-1}{n} + \frac{m+1}{m}\lambda_1(t), \tag{B.61}$$

where $\lambda_1(t)$ is the largest eigenvalue of $W(t)^{-1}B(t)/n$, and $W(t)$ and $B(t)$ denote within and between sequence covariance matrix estimates (see (Brooks and Gelman, 1998) for details).

Convergence analysis for cortical microcircuit models

Gelman-Rubin values were calculated based on 100 runs, where the duration of each run was 10 s of biological time. We tried also much longer simulations of 100 s but did not notice any sign of non-convergent behavior. A random initial state was set in each run by showing random input for 1 s before the start of the actual simulation. This initial random input was fed into the network via the two regular input streams (40 neurons each), by assigning to each input neuron a random rate drawn uniformly from a $0 - 40$ Hz range. Convergence analysis of marginals was performed by applying univariate analysis to single components of the simple state y_S, with $\tau = 10$ ms. From individual marginal convergence values, mean and worst marginal convergence (as in Figure 2.2E,F) were derived by taking at time t the mean/max over all individual $\hat{R}$-values at time t. For pairwise spike coincidences (see Figure 2.2D), we analyzed samples of the product of simple states of two neurons (the product equals 1 only if both neurons spiked within the last 10 ms).

Random readouts for Figure 2.2G were implemented by adding an additional excitatory observer neuron to the network which receives synaptic inputs from a random subset of 500 network neurons (we kept this number 500 fixed across simulations with different network sizes to allow a fair comparison). The number of randomly chosen neurons from each of the pools is given in Table B.1.

	E	I
L2/3	120	30
L4	80	20
L5	200	50

Table B.1: Number of randomly chosen neurons per pool for readout neuron in Figure 2.2G

Synapses onto the readout neuron were created in a similar manner as connections within the cortical column model: short-term plasticity parameters were set depending on the type of connection (EE or IE) according to (Haeusler and Maass, 2007). The weights for EE and IE connections were randomly chosen from a Gamma distribution with mean 2 nS and scale parameter 0.7, and mean 5 nS and scale parameter 0.7, respectively. Gelman-Rubin convergence of readouts was then computed as for the marginal case.

Convergence analysis of vector-valued simple states of subsets of neurons (see Figure 2.2G) was performed by applying multivariate analysis to randomly chosen subnetworks of the cortical column. In particular, we randomly drew 5 neurons from each of the 6 pools, yielding a subnetwork of 30 neurons, and calculated $\hat{R}^{30}(t)$.

B.9 Impact of different dynamic regimes on the convergence time

In Figure 2.3 we compared convergence times in four different neural circuits. The first circuit was identical to the *small cortical microcircuit* from Figure 2.1. For the remaining three circuits, the same stochastic point neurons and conductance-based dynamic synapses with delays were used as for the data-based cortical microcircuit model. Dynamic synaptic parameters were set to the corresponding mean values of parameters used in the cortical column model. Synaptic delays of 1 ms were used for all networks, except for the network with sequential structure (Figure 2.3C) where delays were 3 ms. To modulate the intrinsic excitability of neurons we injected small currents to each neuron. The strengths of injected currents and connections were tuned for each network until the desired network activity was achieved. Synaptic background inputs were injected as in the cortical microcircuit model. To set different initial states (needed for Gelman Rubin analysis), during a preparatory phase of 1 s we injected into each neuron a random current chosen from $[-2, 2]$ nA. These small random input currents were strong enough to yield sufficiently diverse initial states. Gelman-Rubin values were then calculated based on 100 runs, where the duration of each run (after the preparatory phase) was 20 s of biological time. Convergence analysis was performed on marginals (individual simple states with $\tau = 10$ ms). Mean and worst marginals were computed as described in the previous section.

Below are additional details to the circuits used for Figure 2.3B-D: *The sparsely active network* of Figure 2.3B comprises one excitatory (E) and one inhibitory (I) population (each 10 neurons). Connections between neurons were drawn randomly according to the following set of connection probabilities: EE=0.1, EI=0.1, II=0.9, IE=0.9. *The network with sequential structure* of Figure 2.3C consists of two interconnected subnetworks where each one of them produces a stereotypical trajectory. Each subnetwork consists of a trigger neuron, a subsequent chain of neurons, and a pool of inhibitory neurons. Shown in Figure 2.3C are only the excitatory chain neurons from each subnetwork (neurons 1-15: first subnetwork; neurons 16-30: second subnetwork). Each excitatory neuron in the chain projects to all other neurons in the same chain with synaptic strengths decreasing with distance according to $\exp(-\text{distance}/\tau_d)$ where $\tau_d = 0.01$ applies to the forward direction in the chain and $\tau_d = 0.1$ to the backward direction. The trigger neuron projects (forward) to the chain in the same fashion with $\tau_d = 1$. All neurons in the chain project to the inhibitory pool, and all neurons in the inhibitory pool project back to the trigger neuron and to the chain. Finally, the two subnetworks are combined such that the inhibitory pool of one subnetwork projects to the trigger neuron and the chain of the other subnetwork, and vice versa. This ensures that only one of the two subnetworks can be active at a time (competition between two trajectories). *The bistable network* of Figure 2.3D consists of two populations which strongly inhibit each other (each population comprising 10 neurons).

B.10 Phase-specific distributions in the presence of periodic inputs

The theoretical proof for Theorem 2 can be found after the proof of Theorem 1 above. For Figure 2.4F, a single long simulation (100.000 s) of the bi-stable network in Figure 2.4E was carried out. Each of the two pools was defined active at time t if more than two neurons from the pool had an active simple state at time t (with $\tau = 10$ ms). A transition was defined as the succession of a period in which one pool was active and the other pool inactive by a period in which the other became active and the first pool turned inactive. Between those two periods it typically occurs that either both pools are active or both are inactive for some short time. The exact time (and phase within the current cycle) of each transition was defined as the point in the middle of this intermediate period.

B.11 Generation of heuristic solutions to constraint satisfaction problems

Formulation of Sudoku as a constraint satisfaction problem

A constraint satisfaction problem consists of a set of variables defined on some domain and a set of constraints, which limit the space of admissible variable assignments. A solution to a problem consists of an assignment to each variable such that all constraints are met. To formulate Sudoku as a constraint satisfaction problem, we define for each of the 81 fields (from a standard 9x9 grid), which has to be filled with a digit from 1 to 9, a set of 9 binary variables (taking values in $\{0, 1\}$) (Ercsey-Ravasz and Toroczkai, 2012). Each of these binary variables votes for exactly one digit in a field. The rules of the Sudoku game impose constraints on groups of these variables, which can be classified into the following three types.

Given number constraints: The given numbers of a puzzle are fixed. Hence, the binary variables for the given fields are constrained to fixed values, for example, a given value 2 corresponds to fixed binary values $(0, 1, 0, \ldots, 0)$.

Unique field constraints: In a correct solution, there must be only one digit active in each field. Hence in each field, exactly one of the 9 associated binary variables must be 1, and all others must be 0 (equivalent to stating that the sum over these binary variables must equal 1).

Unique group constraints: There are three types of groups: rows, columns and 3x3 subgrids. There are 9 row groups, 9 column groups, and 9 subgrid groups. In any of these groups, each digit $1, \ldots, 9$ must appear only once. Hence, in each group, all binary variables voting for the same digit i must sum to 1.

Network architecture for solving Sudoku

Sudoku can be implemented in a spiking neural network by creating for each of the 9 binary variables in each Sudoku field a local group of n_{group} pyramidal cells. Whenever one of these pyramidal cells fires, the corresponding binary variable is set to 1 for a short period $\tau = 20$ ms. The binary variable is defined 0 only if no neuron in its associated group fired within the last $\tau = 20$ ms. This mapping allows one to readout the current (tentative) solution represented by the network at any time t. The tentative solution is correct only if all constraints are met. For all simulations we used $n_{\text{group}} = 4$, resulting in a total $81 * 9 * 4 = 2914$ pyramidal cells. Constraints among Sudoku variables can be implemented via di-synaptic inhibition between the groups of pyramidal cells as detailed below.

Given number constraints are implemented by providing strong positive input currents selectively to those neurons which code for the given numbers, and negative currents to neurons coding for wrong digits in a given field. *Unique field constraints* are implemented by forming a winner-take-all (WTA) circuit among all $9 * 4 = 36$ neurons associated with the same Sudoku field. A WTA circuit is modeled by a single inhibitory neuron which is reciprocally connected to all 36 pyramidal cells. To reduce the probability that no pyramidal cell fires (which would violate the unique field constraint), thresholds of pyramidal cells are set to low values (see next section for details). *Unique group constraints* are implemented by a WTA circuit in which all neurons in a group which code for the same digit participate. In summary, there are 81 unique field constraints and $27 * 9 = 243$ unique group constraints (in each group there is a constraint for each digit), yielding a total of 324 WTA circuits. These WTA circuits are partially overlapping, in the sense that each pyramidal cell participates in 4 of these WTA circuits (one for the unique value constraint in its field, and three for the unique group constraints in its row/column/subgrid).

Stochastic spike generation in both excitatory and inhibitory neurons is implemented consistent with the theoretical noise model I (see next section for details). The network thus fulfills all theoretical conditions for Theorem 1, and is guaranteed to have a unique stationary distribution $p_C(y|x)$ of network states, to which it converges exponentially fast. This landscape will have automatically peaks at those states of the network which fulfill most of the game constraints, since each of the WTA circuits ensures that invalid configurations with respect to that constraint are unlikely to occur. Any specific Sudoku problem can be set by providing input x to the network in the form of strong currents to those neurons which correspond to the given values. This automatically modifies the landscape of the stationary distribution $p_C(y|x)$ such that only (or predominantly) solutions consistent with the givens are generated. Finally, due to neuronal noise the network can quickly probe different peaks in the landscape (different promising solution candidates) and escape them equally fast. Importantly, this process may occur at different places in the Sudoku puzzle simultaneously. Hence, one can interpret the network dynamics also as a highly parallel stochastic search algorithm.

Details to implementation and simulations for Figure 2.5

Simulations for Figure 2.5 were performed in NEVESIM, an event-based simulator for networks of spiking neurons developed in C++ with a Python Interface(Pecevski, 2013). The puzzle in Figure 2.5A was generated and rated “hard” by “Sudoku Solutions” (Aire Technologies, 2013). Spike generation is modeled according to equation (B.58), with parameters $\delta u = 0.5$, $\tau_s = 20$ ms. The stochastic threshold ϑ was set to -1 and 10 for excitatory and inhibitory neurons, respectively. An absolute refractory period of 3 ms was chosen for pyramidal cells. To maximize the speed up of event-based simulations, PSPs were modeled in a simplified manner as current-based rectangular pulses of length 20 ms (in contrast to the more complex conductance based integration of synaptic inputs used for cortical microcircuit models).

WTA circuits were formed by reciprocally connecting a single inhibitory neuron to all participating pyramidal cells. The single inhibitory neuron was modeled to mimic the response of a population of inhibitory neurons (i.e. strong inhibition for a prolonged amount of time), using an absolute refractory period of 20 ms, and strong bidirectional connections from and to excitatory neurons (synaptic weights 100 and -100, respectively).

To set a particular puzzle, given numbers were fixed by providing strong input currents to the corresponding pyramidal cells. In particular, neurons coding for the given numbers in a Sudoku field received a constant positive input current (a constant input +9 on the membrane potential). Neurons coding for conflicting digits in given Sudoku fields received a constant negative input current of strength −11.

A final practical remark concerns the number of neurons coding for each binary variable, $n_{\text{group}} = 4$. We found that networks with $n_{\text{group}} > 1$ have a number of attractive properties compared to networks with single neuron coding. In particular firing rates of individual neurons can be lower (for $n_{\text{group}} = 1$ a pyramidal cell would need to constantly burst to indicate a steady active state). Also, synaptic efficacies among neurons can be made weaker, and overall spike response patterns appear more biologically plausible. In view of a potential implementation in analog neuromorphic hardware, population coded variable assignments are also less prone to single unit failures or device mismatch.

Appendix C

Appendix to Chapter 3: Solving Hard Computational Problems with Networks of Stochastic Spiking Neurons

Contents

C.1 Stochastic neuron model

Neurons are modeled as simple stochastic point neurons with absolute refractory period τ. When not in a refractory state, neuron k spikes at an instantaneous firing rate which depends exponentially on the membrane potential $u_k(t)$ (3.2), according to,

$$\lim_{\delta t \to 0} p(\text{neuron k fires within}(t, t + \delta t])/\delta t = \rho_k(t) = \frac{1}{\tau} \exp(u_k(t)) \quad , \tag{C.1}$$

with $\tau = 10$ms unless otherwise stated. An exponential dependence of a neuron's firing probability on the membrane potential has been suggested by (Jolivet et al., 2006) based on a fit to experimental data. Similar stochastic neuron models have been suggested by (Truccolo et al., 2005; Buesing et al., 2011).

C.2 Details to Principle 1: stationary distributions and energy functions

Network states

We distinguish between principal neurons and auxiliary neurons. Principal neurons directly represent the random variables (RV) of a problem (e.g. the boolean variables in a satisfiability problem as in Figure 3.2). The state of principal neurons therefore reflects the state of the RVs. Auxiliary neurons (i.e. all auxiliary neurons in circuit motifs and the *lock-in* neuron), on the other hand, do not represent random variables. Their only purpose is to modulate and shape the distribution (and energy function) over principal neurons.

The state $x_k(t)$ of a principal neuron k at time t is defined as,

$$x_k(t) = \begin{cases} 1, & \text{if neuron k fired within } (t-\tau, t] \ , \\ 0, & \text{otherwise } \ , \end{cases} \tag{C.2}$$

where τ is a brief time window corresponding to the duration of a PSP. The state $\xi_m(t)$ of an auxiliary neuron m is defined in an analogous manner. The *full network state*,

$$(\mathbf{x}(t), \boldsymbol{\xi}(t)) = (x_1(t), \ldots, x_N(t), \xi_1(t), \ldots, \xi_M(t)) \tag{C.3}$$

is defined as the vector of states of all principal neurons $k = 1, \ldots, N$ and all auxiliary neurons $m = 1, \ldots, M$ in the network. Similar notions of network state have been suggested by a number of experimental (Schneidman et al., 2006; Berkes et al., 2011) and theoretical (Buesing et al., 2011; Pecevski et al., 2011; Habenschuss et al., 2013a) papers. The *principal network state* refers only to the state vector $\mathbf{x}(t)$ of all principal neurons. Unless otherwise stated, the term *network state* refers to the principal network state.

Convergence to stationary distribution

Under mild conditions, activity in a general spiking network with noise can be theoretically guaranteed to converge exponentially fast to a unique stationary distribution $p(\mathbf{x}, \boldsymbol{\xi})$ of full network states (Habenschuss et al., 2013a), regardless of initial network conditions. In the context of the stochastic neuron model (3.1-3.2) it can be easily verified that the theoretical conditions for convergence are fulfilled if all weights w_{kl} are bounded from above, i.e. if there exists some $w_{\max}$ such that all $w_{kl} \leq w_{\max}$. Throughout the paper this condition is met. Exponentially fast convergence to a unique *marginal* distribution $p(\mathbf{x})$ over principal network states is a simple corollary that follows from the convergence to a unique joint distribution $p(\mathbf{x}, \boldsymbol{\xi})$.

Energy functions

In analogy with statistical physics (Plischke and Bergersen, 2006), we define the energy function $E(\mathbf{x})$ of a network of spiking neurons with unique stationary distribution $p(\mathbf{x})$ of principal network states $\mathbf{x}$ as

$$E(\mathbf{x}) = -\log p(\mathbf{x}) + C \quad , \tag{C.4}$$

with an arbitrary constant C. The stationary distribution $p(\mathbf{x})$ can then be expressed as,

$$p(\mathbf{x}) = \frac{e^{-E(\mathbf{x})}}{\sum_{\mathbf{x}'} e^{-E(\mathbf{x}')}} \quad . \tag{C.5}$$

Note that according to this definition, energies are defined only up to a constant (a global shift applied to all states). To indicate that two energy functions are identical except for a constant shift we use the notation $E_1(\mathbf{x}) \triangleq E_2(\mathbf{x})$, i.e.

$$E_1(\mathbf{x}) \triangleq E_2(\mathbf{x}) \quad \Leftrightarrow \quad \exists C{\in}\mathbb{R}\ \forall\mathbf{x}\ \left(E_1(\mathbf{x}) = E_2(\mathbf{x}) + C\right) \quad . \tag{C.6}$$

C.3 Details to Principle 2: circuit motifs shaping the energy function

A key theoretical question is how the energy function $E(\mathbf{x})$ (or equivalently $p(\mathbf{x})$) over principal network states $\mathbf{x}$ depends on the parameters of a network, in particular on synaptic weights w_{kl} and neuronal excitabilities b_k among principal neurons, as well as on auxiliary circuits connected to the principal neurons. Previous work had shown that pair-wise symmetric connections between neurons map onto second-order dependencies between variables (Buesing et al., 2011). (Pecevski et al., 2011) demonstrated in addition how more complex dependencies can be encoded through the use of pre-processing circuits in the context of probabilistic inference.

Here we consider how in addition to second-order dependencies, common higher-order constraints of hard computational problems can be encoded through the use of simple auxiliary circuit motifs, in a manner suitable for compositionality and large-scale circuit design.

Compositionality

To facilitate systematic design of complex energy landscapes, we would like to find a basic set of auxiliary circuit motifs which can be combined in arbitrarily rich ways with predictable outcomes. A particularly desirable feature to aim for is linear compositionality, such that the energy contribution to the energy landscape of each circuit motif is independent of the presence of other circuits. A precise definition follows.

Definition 1 (Compositionality). *Let $\mathcal{N}$ be a principal network of $k = 1, \ldots, N$ stochastic principal neurons (3.1)-(3.2), symmetric connections $w_{kl} = w_{lk}$ (but no self-connections, i.e. $w_{kk} = 0$) and biases b_k, with energy function $E_{\mathcal{N}}(\mathbf{x})$. Let $\mathcal{C} = \{C_1, \ldots, C_L\}$ be a set of L additional auxiliary circuits which can be reciprocally connected to the principal network $\mathcal{N}$ to modulate the behavior of principal neurons. Denote by $E_{\mathcal{N},\mathcal{I}}(\mathbf{x})$ the modulated energy function of the network in the presence of a subset $\mathcal{I} \subseteq \{1, \ldots, L\}$ of these auxiliary circuits, and define the change in the energy landscape due to the presence of this subset $\mathcal{I}$ as $\Delta E_{\mathcal{N},\mathcal{I}}(\mathbf{x}) \triangleq E_{\mathcal{N},\mathcal{I}}(\mathbf{x}) - E_{\mathcal{N}}(\mathbf{x})$. Then the set of auxiliary circuits $\mathcal{C}$ is said to be compositional with respect to network $\mathcal{N}$ if changes in energies sum up linearly for all possible combinations, i.e.*

$$\Delta E_{\mathcal{N},\mathcal{I}}(\mathbf{x}) \triangleq \sum_{i \in \mathcal{I}} \Delta E_{\mathcal{N},i}(\mathbf{x}) \quad , \tag{C.7}$$

for any subset $\mathcal{I} \subseteq \{1, \ldots, L\}$.

Note that due to linearity of membrane integration (3.2), the membrane potential of a principal neuron k in the presence of some subset $\mathcal{I}$ of arbitrarily complex auxiliary circuits can be written as,

$$u_{k,\mathcal{I}}(t) = b_k + \sum_l w_{kl}\, x_k(t) + \sum_{i \in \mathcal{I}} \Delta u_{k,i}(t) \quad , \tag{C.8}$$

where the current contribution of auxiliary circuit C_i to the membrane potential of principal neuron k is denoted by $\Delta u_{k,i}(t)$. Define $\mathbf{x}_{\setminus k}(t)$ as the state vector of all principal neurons except neuron k, and $\{x_k = \cdot, \mathbf{x}_{\setminus k}(t)\}$ as the state vector $\mathbf{x}(t)$ with the state of neuron k replaced by $\cdot$. Then, building on the analysis of (Buesing et al., 2011), the following theoretical result provides sufficient conditions on the auxiliary circuit contributions $\Delta u_{k,i}(\cdot)$ for compositionality.

Theorem 3 (Sufficient conditions for compositionality). *Let $\mathcal{N}$ be any network of principal neurons and $\mathcal{C}$ a set of auxiliary circuits as defined above. Suppose that for each auxiliary circuit C_i there exists an energy function $U_i(\mathbf{x})$ such that at any time t the following relation holds,*

$$\Delta u_{k,i}(t) = U_i\left(\{x_k = 0, \mathbf{x}_{\setminus k}(t)\}\right) - U_i\left(\{x_k = 1, \mathbf{x}_{\setminus k}(t)\}\right) \tag{C.9}$$

Then the set of auxiliary circuits $\mathcal{C}$ is compositional with respect to $\mathcal{N}$. Furthermore, the energy change due to each individual circuit is given by $\Delta E_{\mathcal{N},i}(\mathbf{x}) \triangleq U_i(\mathbf{x})$.

Theorem 3 suggests that auxiliary circuits should be constructed in a highly specific manner to support compositionality. In particular, (C.9) states that auxiliary circuit contributions to the membrane potential of a principal neuron k should be basically memoryless and reflect a specific function of the current state of the remaining network, $\mathbf{x}_{\setminus k}(t)$. Note that this function (the right-hand side of (C.9)) has a very intuitive interpretation: a circuit C_i should inform each principal neuron k about the currently expected drop in the energy function U_i that can be achieved by a spike of neuron k (i.e. a switch from $x_k = 0$ to $x_k = 1$).

Proof of Theorem 3: If (C.9) holds for all C_i then the membrane potential of a principal neuron k in the presence of some subset of auxiliary neurons $\mathcal{I}$ is given at time t by,

$$u_{k,\mathcal{I}}(t) = b_k + \sum_{l=1}^{N} w_{kl}\, x_k(t) + \sum_{i \in \mathcal{I}} \left[U_i\left(\{x_k = 0, \mathbf{x}_{\setminus k}(t)\}\right) - U_i\left(\{x_k = 1, \mathbf{x}_{\setminus k}(t)\}\right)\right] \tag{C.10}$$

This can also be expressed as,

$$u_{k,\mathcal{I}}(t) = U_{\mathcal{I}}\left(\{x_k = 0, \mathbf{x}_{\setminus k}(t)\}\right) - U_{\mathcal{I}}\left(\{x_k = 1, \mathbf{x}_{\setminus k}(t)\}\right) \quad , \tag{C.11}$$

with

$$U_{\mathcal{I}}(\mathbf{x}) = -\sum_{k=1}^{N} b_k x_k - \frac{1}{2}\sum_{k=1}^{N}\sum_{l=1}^{N} w_{kl} x_k x_l + \sum_{i \in \mathcal{I}} U_i(\mathbf{x}) \quad . \tag{C.12}$$

One can then verify that the neural computability condition (NCC) from (Buesing et al., 2011) is fulfilled by a network with membrane dynamics (C.11) with respect to stationary distribution $p(\mathbf{x}) \propto \exp(-U_{\mathcal{I}}(\mathbf{x}))$:

$$\begin{aligned}
\log \frac{p(x_k = 1|\mathbf{x}_{\setminus k})}{p(x_k = 0|\mathbf{x}_{\setminus k})} &= \log \frac{p(\{x_k = 1, \mathbf{x}_{\setminus k}\})}{p(\{x_k = 0, \mathbf{x}_{\setminus k}\})} && \text{(C.13)}\\
&= \log p(\{x_k = 1, \mathbf{x}_{\setminus k}\}) - \log p(\{x_k = 0, \mathbf{x}_{\setminus k}\}) && \text{(C.14)}\\
&= -U_{\mathcal{I}}(\{x_k = 1, \mathbf{x}_{\setminus k}\}) + U_{\mathcal{I}}(\{x_k = 0, \mathbf{x}_{\setminus k}\}) && \text{(C.15)}\\
&= b_k + \sum_{l=1}^{N} w_{kl} x_l + \sum_{i \in \mathcal{I}} [-U_i(\{x_k = 1, \mathbf{x}_{\setminus k}\}) + U_i(\{x_k = 0, \mathbf{x}_{\setminus k}\})] && \text{(C.16)}
\end{aligned}$$

Thus, a network with membrane dynamics (C.12) meets the NCC for $p(\mathbf{x}) \propto \exp(-U_{\mathcal{I}}(\mathbf{x}))$, and the energy function of the network is given by $E_{\mathcal{N},\mathcal{I}}(\mathbf{x}) \triangleq U_{\mathcal{I}}(\mathbf{x})$. Furthermore, from (C.12) it is obvious that energies due to combinations of auxiliary circuits sum up linearly and that the energy contribution due to each single C_i equals $\Delta E_{\mathcal{N},i}(\mathbf{x}) \triangleq U_i(\mathbf{x})$.
□

Note that, in contrast to neural sampling theory (Buesing et al., 2011), Theorem 3 is only concerned with the distribution over a subset of all neurons (the principal neurons $\mathbf{x}$), i.e. the marginal distribution $p(\mathbf{x})$ after integrating out all auxiliary variables $\boldsymbol{\xi}$.

WTA circuit motif

The WTA circuit motif consists of a single auxiliary neuron which is reciprocally connected to some subset $\mathcal{K} \subseteq \{1, \dots N\}$ of principal neurons (Figure 3.1B). The goal of the WTA motif is to achieve that most of the time *exactly* one neuron in $\mathcal{K}$ is active. The WTA motif should thus increase the energies of all network states except those states where exactly one neuron in $\mathcal{K}$ is active. This can be achieved in two steps. First, the energy of all network states where more than one neuron in $\mathcal{K}$ is active is increased. We found that this can be robustly achieved by a single inhibitory neuron which receives strong excitatory connections from $\mathcal{K}$, and sends strong inhibitory connections back to $\mathcal{K}$ (with some weight $-w_{\mathrm{WTA}} \ll 0$). The inhibitory neuron should have a low bias such that it only fires when one of the principal neurons is active. Second, the energy of states where no neuron in $\mathcal{K}$ is active is raised. This can be done most easily by raising the biases of all neurons in $\mathcal{K}$ by some constant b_{WTA} (not shown in Figure 3.1B) with $0 < b_{\mathrm{WTA}} < w_{\mathrm{WTA}}$. Alternatively, this could in principle also be achieved by an additional auxiliary neuron which is constantly active and makes excitatory connections to all neurons in $\mathcal{K}$.

The described implementation of the WTA circuit motif is intended to approximate the requirements of Theorem 3 for compositionality. This can be seen if one considers the energy function

$$U_{\mathrm{WTA}[\mathcal{K}]}(\mathbf{x}) = \begin{cases} b_{\mathrm{WTA}} \, , & \text{if } \sum_{k \in \mathcal{K}} x_k = 0 \, , \\ 0 \, , & \text{if } \sum_{k \in \mathcal{K}} x_k = 1 \, , \\ (w_{\mathrm{WTA}} - b_{\mathrm{WTA}}) \cdot (-1 + \sum_{k \in \mathcal{K}} x_k) \, , & \text{if } \sum_{k \in \mathcal{K}} x_k > 1 \, . \end{cases} \tag{C.17}$$

According to (C.9) the ideal $\Delta u_{k,\mathrm{WTA}[\mathcal{K}]}(t)$ for implementing this energy function in a compositional manner is given by,

$$\Delta u_{k,\mathrm{WTA}[\mathcal{K}]}(t) = \begin{cases} b_{\mathrm{WTA}} \, , & \sum_{l \in \mathcal{K} \setminus k} x_l(t) = 0 \, , \\ b_{\mathrm{WTA}} - w_{\mathrm{WTA}} \, , & \sum_{l \in \mathcal{K} \setminus k} x_l(t) > 0 \, . \end{cases} \tag{C.18}$$

This behavior is closely approximated by the described WTA circuit implementation: Regardless of the network state, there is a bias term b_{WTA}. As soon as one (or more) of the neurons fire, this triggers the auxiliary inhibitory neuron which then strongly inhibits all competitors with weight $-w_{\mathrm{WTA}}$. The nature of the approximation lies mainly in the delay between the onset of activity of a winner and the onset of inhibition at the remaining principal neurons.

OR circuit motif

The OR circuit motif consists of two auxiliary neurons reciprocally connected to some subset $\mathcal{K} \subseteq \{1, \dots N\}$ of principal neurons (Figure 3.1B). The purpose of the OR motif is to ensure that most of the time *at least one* neuron in $\mathcal{K}$ is active. Thus, the energies of all network states where no neuron in $\mathcal{K}$ is active should be increased. At the same time, however, the energies of all other network states should ideally remain unmodified, regardless of how many neurons ≥ 1 are active. The first part, i.e. increasing the energies of states where no neuron is active, can be done by adding an auxiliary neuron (Figure 3.1B, left auxiliary OR neuron) which excites all neurons in $\mathcal{K}$ with equal synaptic weight w_{OR}. The second part is slightly more tricky, as it requires that the OR circuit should suspend its influence on the network during periods where at least one neuron in $\mathcal{K}$ is active. This can be achieved by a) adding an inhibitory connection from $\mathcal{K}$ to the first auxiliary neuron such that the neuron is only activated when needed, and b) by adding a second auxiliary neuron (Figure 3.1B, right auxiliary OR neuron) which is triggered when a neuron in $\mathcal{K}$ fires in response to the first auxiliary neuron. The goal of the second auxiliary neuron is to immediately cancel any effect of the first auxiliary neuron on the remaining neurons in $\mathcal{K}$ due to sustained post-synaptic potentials. This is achieved through inhibitory connections $-w_{OR}$ to all neurons in $\mathcal{K}$.

Analogous to the WTA circuit, the described implementation of the OR circuit motif aims to approximate the requirements of Theorem 3 for compositionality. To see this, consider the energy function

$$U_{\mathrm{OR}[\mathcal{K}]}(\mathbf{x}) = \begin{cases} 0\ , & \text{if } \sum_{k\in\mathcal{K}} x_k \geq 1\ , \\ w_{\mathrm{OR}}\ , & \text{if } \sum_{k\in\mathcal{K}} x_k = 0\ . \end{cases} \tag{C.19}$$

By (C.9) the corresponding ideal $\Delta u_{k,\mathrm{OR}[\mathcal{K}]}(t)$ supporting compositionality is given by,

$$\Delta u_{k,\mathrm{OR}[\mathcal{K}]}(t) = \begin{cases} 0\ , & \sum_{l\in\mathcal{K}\setminus k} x_l(t) \geq 1\ , \\ w_{\mathrm{OR}}\ , & \sum_{l\in\mathcal{K}\setminus k} x_l(t) = 0\ . \end{cases} \tag{C.20}$$

The OR circuit approximates this behavior as described above through the combination of two auxiliary neurons. The nature of the approximation is three-fold. First, when all principal neurons in an OR circuit have just turned off (and thus the constraint is not met anymore), the additional bias w_{OR} should ideally be communicated instantly to all neurons. However, the first auxiliary neuron fires in general with some small delay, and therefore the additional bias w_{OR} is signaled to the principal neurons slightly later than ideally required. Second, when a principal neuron eventually fires in response to the first auxiliary neuron, there is a delay until the second auxiliary neuron turns on to cancel the bias w_{OR} that is still present due to lingering PSPs from the first auxiliary neuron. Third, there is an “undershoot” effect when the excitatory PSP of the first principal neuron has already vanished, but the inhibitory PSP of the second auxiliary neuron is still present. To minimize the error due to this effect, the overall biases of all principal neurons in an OR circuit should be kept high, in order to keep the typical delay between the activity onset of the first and the second auxiliary neuron as short as possible.

C.4 Details to Principle 3: benefits of asymmetric spike-based signaling

Principles 1 and 2 pave the way towards massively parallel realizations of stochastic search in networks of spiking neurons. A first application of these principles has provided compelling results in simulations, as demonstrated in Figure 3.2 and Figure 3.3. A key theoretical question which then arises is to what extent different components of the system contribute to the observed performance. There are various aspects that can be examined in this context, such as the asynchronicity of message transfer, stochasticity, and the asymmetry of spike-based communication (a spike marks the onset of a fixed-length *on* period, whereas *off* periods vary randomly - hence *on* and *off* states are handled fundamentally different by a spiking network). We focus our analysis here on the role of the asymmetry of spike-based signaling, because its implications are arguably least well understood.

Asymmetric vs. symmetric dynamics

In order to isolate the effect of asymmetric signaling we consider an artificial non-spike-based "symmetrized" system in which *on* and *off* transitions of units are not mediated in an asymmetric fashion via spikes of fixed length, but rather in a symmetric manner. Specifically, we aim to morph neural spiking dynamics into the dynamics of Gibbs sampling (Bishop, 2006), one of the standard methods in statistics and machine learning for sampling from complex probability distributions. By theoretically analyzing and comparing the behavior of the two systems one can then reason about the specific role of asymmetric signaling.

A canonical way of symmetrizing the dynamics of a given spiking network with noise is to make sure that all other components and aspects of the systems remain unchanged (event-based asynchronous signaling, stochasticity, synaptic weights and biases, definition of membrane potential u_k given the current *on*/*off* states of other neurons) and modify only the way the system handles transitions between *on* and *off* states. Importantly, to facilitate a comparison between asymmetric vs. symmetric dynamics, such modification should not alter the stationary distribution and energy function of the system.

For a stochastic spiking neuron embedded in some network, transitions occur from *off* to *on* states according to

$$\rho_{\mathrm{on}}(u_k) = \frac{1}{\tau}\exp(u_k) \quad , \tag{C.21}$$

whereas transitions from *on* to *off* occur deterministically after a period of τ time units has passed. Clearly, in a symmetric system transitions must occur stochastically in both directions (they cannot be both deterministic), with transition rates $\rho'_{\mathrm{on}}(u_k)$ and $\rho'_{\mathrm{off}}(u_k)$. Concrete symmetric expressions for $\rho'_{\mathrm{on}}(u_k)$ and $\rho'_{\mathrm{off}}(u_k)$ are obtained by using a continuous-time variant of Gibbs sampling (Bishop, 2006).

$$\rho'_{\text{on}}(u_k) = \rho_0 \cdot \sigma(u_k) \ , \tag{C.22}$$

$$\rho'_{\text{off}}(u_k) = \rho_0 \cdot \sigma(-u_k) \ , \tag{C.23}$$

where $\sigma(u) = (1 + \exp(-u))^{-1}$ denotes the standard sigmoid function. Such a continuous-time variant of Gibbs sampling has been proposed in the literature, for example, in the context of sampling from second-order Boltzmann machines (Yamanaka et al., 1997).

Asymmetry facilitates transitions across large energy barriers

A somewhat unexpected but striking difference which emerges from the comparative analysis between asymmetric and symmetric dynamics is that transitions across large energy barriers are much more likely and frequently to occur with asymmetric (spike-based) signaling. To see this, define the mean *on*-transition time $m_{\text{on}}(u)$ as the average time from the last *on*→ *off* transition until the next *off*→*on* transition, at a given membrane potential u. The mean *off*-transition time is defined in an analogous manner. In the stochastic spiking network these are given by,

$$m_{\text{on}}(u) = \frac{1}{r_{\text{on}}(u)} = \tau \cdot \exp(-u) \ , \tag{C.24}$$

$$m_{\text{off}}(u) = \tau \ . \tag{C.25}$$

In the symmetric system, on the other hand, mean transition times are given by,

$$m'_{\text{on}}(u) = \frac{1}{r'_{\text{on}}(u)} = \frac{1}{\rho_0} \cdot (1 + \exp(-u)) \ , \tag{C.26}$$

$$m'_{\text{off}}(u) = \frac{1}{r'_{\text{off}}(u)} = \frac{1}{\rho_0} \cdot (1 + \exp(u)) \ . \tag{C.27}$$

Notably, one can identify a single translation factor $F(u)$ between the two systems,

$$m_{\text{on}}(u) = m'_{\text{on}}(u) \cdot F(u), \tag{C.28}$$

$$m_{\text{off}}(u) = m'_{\text{off}}(u) \cdot F(u) \tag{C.29}$$

which is given by,

$$F(u) = \underbrace{\tau\rho_0}_{\text{const.}} \cdot (1 + \exp(u))^{-1} \tag{C.30}$$

Note that $F(u)$ is strictly positive and decreases monotonically with increasing membrane potential u. Furthermore, note that small values $F(u)$ signify that the asymmetric dynamics is fast in comparison with the symmetric dynamics (in both, *on* and *off* directions). Hence, (C.30) shows that the asymmetric dynamics of spiking neurons increases specifically the *on*- and *off*-transition rates of those neurons with high membrane potentials u (i.e. neurons with strong input and/or high biases). This makes sense since *off* transitions in the symmetric case can be arbitrarily slowed down for large u (C.27), whereas the spike-based system will necessarily fall back to an *off* state on a regular basis regardless of u.

Given that transitions are specifically enhanced in the presence of high membrane potentials, and taking into account that large u reflect large energy barriers (according to (C.12)), it follows that the spike-based system is much more inclined to make exploratory *on*→*off* transitions (crossing large energy barriers) on a regular basis. Despite the resulting increased frequency of *transitions* to high-energy states due to (C.30), however, it should be stressed that on average the asynchronous spike-based system does not spend more *time* in high-energy states (both systems sample from the same $p(\mathbf{x})$), because according to (C.28) also the transition back to the corresponding *on* state (i.e. the lower energy state) happens at an increased rate for large u. The critical observation is that the return to the identical previous state *can* be intercepted by other neurons which, while the neuron is *off*, are given the brief opportunity to spike before the previous state is restored, and may thereby, e.g., escape from a previously inhibited state. This is particularly obvious in the context of WTA circuits, where such brief periods of *off*-time of the current winner allow other neurons to take over. Altogether, as we demonstrated in Figure 3.3, it is observed that this enhanced utilization of exploratory moves leads to improved search for low energy states in the asymmetric spike-based system, by facilitating fast escape routes from deep local minima which are not available to such extent in a symmetric system.

Asymmetry facilitates goal-directed transitions

(C.30) states that spike-based transition frequency is enhanced in proportion to u. It was already noted above that this encourages exploratory *on*→*off* transitions which may facilitate the escape from local minima. But clearly also *off*→*on* transitions are affected by (C.30). In particular, consider a situation where a group of neurons in the *off* state is competing for emitting the next spike (e.g. in a WTA circuit). Those neurons with the highest membrane potentials are particularly eager to fire. Suppose, for example, that there are two neurons with $u_a = 3$ and $u_b = 5$, and all other neurons have considerably lower u. In the symmetrized non-spiking system, transition rates scale with $\sigma(u)$ and are therefore approximately equal for the two neurons a and b (due to saturation of the sigmoid function). In the spike-based system, however, instantaneous transition rates scale with $\exp(u)$ and thus the competition will be much easier to win by the neuron which is most eager to fire (i.e. neuron b in the example). Clearly, this makes a substantial difference in the dynamics and performance of the stochastic search, especially since u_k reflects the drop in energy that can be gained by turning on some neuron k. In particular, it means that a spike-based system is not only more exploratory in the "up-hill" direction (*on*→*off* transitions towards higher energy levels), but also more goal-directed in the "down-hill" direction.

Obviously, the enhanced agility with respect to some transitions must come at a price. Indeed, those transitions which bring about only small changes in the energy landscape (transitions with small u) are considerably disadvantaged by the spike-based dynamics. In terms of convergence properties, however, this seems to be a small price to pay, since stochastic search appears in practice more frequently impeded by the presence of large energy barriers. [1]

C.5 Details to Principle 4: internal temperature control

In order to realize an internal temperature control mechanism which allows network activity to "lock in" when a good solution has been found, the following functional components are required (Figure 3.1D): 1. The generation of *OK* signals in each circuit motif. 2. A *lock-in unit* that integrates individual *OK* signals into a global *all OK* message. 3. The activation of additional circuits which reduce temperature.

The following realizations of these elements have proved effective: For the WTA circuit motif, the activity of the inhibitory neuron can be directly used as an *OK* signal. This works because the probability that two neurons are active at the same time is vanishingly small as long as strong inhibitory connections are used in the WTA motif. Hence, in practice whenever the inhibitory neuron is active it means that exactly one principal neuron is active (and the WTA constraint is met). For the same reasons, one can also simply connect all neurons in a WTA circuit to the *lock-in* neuron. Since at most one neuron is active at a time, the joint impact of these neurons on the *lock-in* unit precisely reflects whether the WTA constraint is met. For the OR circuit motif, the most straightforward way of implementing an *OK* signal is to add another auxiliary neuron with low bias and excitatory connections from all involved principal neurons, such that the neuron fires as long as one of the principal neurons is active, and remains silent otherwise. In simulations, however, a slightly different implementation has proved more effective, which can be used when all principal neurons involved in the OR circuit are also part of some WTA circuit. Then, a *not OK* signal can be derived by adding an auxiliary neuron with low bias which receives connections from all other neurons in the WTA circuits of the involved principal neurons. This works because, whenever principal neuron k (which is involved in the OR circuit and in addition in some WTA circuit) is not active, some other neuron in the WTA circuit of neuron k must be active (most of the time). Hence, whenever the OR constraint is violated and all K principal

[1]Clearly, also transitions with negative u are disadvantaged by the spike-based dynamics. In general, this may have a negative effect on convergence, and the magnitude of such negative effects would need to be examined in relation to the previously described advantages. In the context of this paper, however, negative u practically only occur in neurons which are not supposed to fire at all, for example neurons which are currently inhibited in a WTA circuit. And in this case it is in fact desirable that such transitions occur with decreased frequency.

neurons involved in the OR circuit are inactive, the auxiliary neuron will see that in each of the involved WTA circuits some other neuron is active. A more detailed description of how this was implemented as part of the *lock-in* mechanism for 3-SAT problems is given in Section "Details to 3-SAT application".

The *lock-in neuron* can be implemented by choosing a low bias and connection strengths from *OK* neurons in each circuit in such a manner that the firing probability reaches non-negligible values only when all *OK* signals are active. When circuits send either *OK* or *not OK* signals, the connection strengths from *not OK* should be negative and can be chosen in such a manner that non-negligible firing rates are achieved only if all *OK* but none of the *not OK* signals are active.

Regarding the activation of additional circuitry to reduce temperature, the most straightforward way of achieving this is to duplicate all circuit motifs (as indicated in Figure 3.1D). The biases of auxiliary neurons in duplicated circuits should be much lower, such that these circuits remain inactive unless an additional excitatory drive is provided by the *lock-in* neuron. This is exactly how temperature reduction was implemented for the OR circuit motif. For WTA circuits, however, there exists an even a simpler way of reducing temperature which does not require duplication of circuitry but only excitatory connections from the *lock-in* neuron to all principal neurons in a WTA circuit. This works well because the WTA circuit motif consists of two components, a) excitatory drive (increased bias) to all involved principal neurons, and b) strong mutual inhibition. If inhibition strength is very strong, however, duplication of that second component is not necessary. Hence, a reduction of temperature can be achieved by mere activation of additional excitatory drive. For further implementation details see Section "Details to 3-SAT application".

C.6 Details to simulations

All simulations were performed in NEVESIM, an event-based neural simulator. Optimization of networks, as well as exploration of their properties were done with ZLIB, a library for parallelization and optimization, developed within the scope of this thesis. The analysis of simulation results was performed in Python and Matlab.

Details to 3-SAT application (Figure 3.2)

A general 3-SAT problem consisting of a set of binary variables and a set of clauses, each involving three variables, can be implemented in a spiking network by representing each binary random variable (RV) with two neurons forming a WTA circuit (biases of the principal neurons: b_{nrn}). In particular, the WTA circuit is implemented by adding a single inhibitory neuron with bias b_{inh} and connecting it to the two neurons with bidirectional connections w_{inh} and w_{exc} (to and from the inhibitory neuron, respectively). The w_{inh} should be set strong enough to shut down all principal neurons in the WTA circuit (to overcome their biases). w_{exc} should be strong enough such that it activates inhibition almost immediately in order to prevent other neuron(s) from spiking.

For the implementation of a 3-SAT clause one needs to form an OR circuit consisting of those neurons which take part in the clause. In particular, two auxiliary neurons are added, with biases of $0.5B$ and $-3.5B$ for the first and the second auxiliary neuron, respectively, where B is some constant. Both auxiliary neurons should connect to those neurons involved in the clause (in total to 3 neurons), with bidirectional connections w_{OR} and $-B$ (to and from the first auxiliary neuron, respectively), and $-w_{OR}$ and B (to and from the second auxiliary neuron, respectively). Finally, the first auxiliary neuron connects to the second one with strength $3B$.

Therefore, the total number of neurons needed to implement a general 3-SAT problem in a spiking neural network is $3\#$variables $+$ $2\#$clauses ($2+1$ per WTA circuit, and 2 per OR circuit), while the number of connections is $4\#$variables $+$ $13\#$clauses. Notably, both the number of neurons and the number of synapses depend linearly on the number of variables (the number of clauses linearly depends on the number of variables if problems with some fixed clauses-to-variables ratio are considered).

At any point in time the principal network state $\mathbf{x}$ is defined based on the activity of principal neurons within the last τ time units. If exactly one of the two neurons which code for a RV $\mathbf{X}_i$ is active at some time t, then the variable has a properly defined value. The WTA circuit for each RV ensures that this is the case most of the time for most problem variables. When this is the case, one can simply read off the current assignment of values to the RVs from the network state. Any clause is considered satisfied if at least one of the three neurons, which correspond to the three literals of the clause, is active.

To calculate the current performance of a solution at any point in time we use as a performance measure the ratio between the number of satisfied clauses and total number of clauses. If none of the variables which take part in a clause are properly defined then that clause is considered unsatisfied. As a result, this performance measure is well-defined at any point in time.

In order to implement the lock-in mechanism we use two working regimes which differ in the temperature of the network. While the first one is the normal regime during which the network normally explores possible solutions, the second one is the regime of decreased temperature during which the network locks into the current state (solution) and is very unlikely to escape from it. To implement the second regime we add for each clause two additional auxiliary neurons which are connected in the same way as the original auxiliary neurons (they target the same neurons) but with different weights: w_{OR2} and $-B$ (to and from neuron), and $-w_{OR2}$ and B(to and from neuron), for the first and second additional auxiliary neuron, respectively. In addition their biases are set to $-0.5B$ and $-6.5B$ (first and second aux. neuron).

These additional auxiliary neurons are activated (i.e. functional) only when a certain state (the solution) was detected, which is signaled by the global *lock-in* neuron with bias b_{glob} that is connected to both additional auxiliary neurons of all clauses with connection strengths B and $3B$ to the first and the second additional auxiliary neuron, respectively. Additionally, the global *lock-in* neuron is connected to every other principal neuron with connection strength w_{glob}. This global neuron is active by default due to the high bias, but is deactivated whenever one of the status neurons, which check if a certain clause is not satisfied, is active (*not OK* signals). There is one status neuron for each clause, with bias set to $-2.5B$. The status neuron receives excitatory connections from all neurons corresponding to inverted literals of the clause, with strength B. Therefore, if all RVs that participate in the clause are set to the wrong values, this triggers the status neuron which reports that the clause is not satisfied. This automatically shuts down the global neuron signaling that the current network state is not a valid solution.

To implement this lock-in mechanism one needs additional 3#clauses+1 neurons and 2#variables + 20#clauses synapses.

The architecture described above was used throughout with the following parameters: $\tau = 10e - 3$ and refractory period of 10ms for all neurons except for the global neuron which has $\tau = 9e - 3$ and refractory period of 9ms, $b_{nrn} = 2$, $b_{inh} = -10$, $b_{glob} = 10$, $B = 40$, $w_{inh} = -100$, $w_{exc} = 100$, , $w_{inh} = -100$, $w_{OR} = 2.5$, $w_{OR} = 10$, with rectangular PSPs of 10ms duration without transmission delays for all synapses except for the one from the global neuron to the additional auxiliary neurons where the duration is 11ms.

For the analysis in Figure 3.2F of problem size dependence we created 3-SAT problems of different sizes with clause-to-variable ratio of 4.3. To ensure that a solution exists, each of the created problems was checked for satisfiability with zhaff, a freely available 3-SAT solver (Fu et al., 2004).

Details to TSP application (Figure 3.3)

For finding the shortest route for a TSP problem consisting of N_{cities} cities and $N_{resting}$ additional resting steps one needs in total $N_{\text{cities}} + N_{resting}$ variables, where each RV codes for the city visited at a certain step s. To solve TSP problems one needs to consider three types of constraints: (a) each RV must be properly defined, i.e. exactly one city must be visited at each step s. In addition, (b) each value of a variable must appear at least once in all RVs. At the same time only neighboring variables (those coding for consecutive steps) can have the same values (this allows for "resting" steps). Finally, (c) the penalty (in terms of additional energy) that two consecutive variables appear in a given configuration, i.e. a particular transition from city i to some other city j, must reflect the traveling cost between the pair of cities.

Based on Principle 2 these constraints can be implemented in a spiking neural network by forming circuits and interactions between neurons, where each of $N_{\text{cities}}+ N_{resting}$ variables with N_{cities} different values can be represented by N_{cities} neurons with bias b_{nrn} each of which represents one city. To implement variable constraints (a) it is enough to form a WTA circuits from neurons that code for different values of the same variable. The WTA is formed in the same way as described for 3-SAT problem by taking corresponding neurons (here WTA circuits have N_{cities} principal neurons). To force the network to visit some desired city at a particular step it is sufficient to set the biases of those neurons which code for the desired city at in the WTA circuit of that step to different values. In particular, the desired value in the WTA circuit of that step is set to b_P and all others to b_N.

The implementation of constraints (b) requires that all variables have different values except if they are neighboring variables. In the spiking network implementation this can be realized by adding negative connections of strength w_{unique} from each neuron that codes for a certain value in a variable to all other neurons which code for the same value in other variables, except for the neighboring variables. This simply prevents, or decreases the chance, that two particular variables have the same value except if they are neighboring variables.

Finally, constraints (c) can be implemented by adding connections between all the neurons which code for two consecutive variables. This results in a network with a ring structure (as the last and the first variable are also connected). We chose to encode weights of these connections such that they reflect the relative distances between cities. To calculate the weights we normalize all the distances with respect to the maximum distance (this procedure applies also for asymmetric problems) and then we rescale and shift them according to $w = w_{offset} + (1 - w_N) * w_{scale}$, where w_N are normalized weights in $[0, 1]$ range.

Such architecture requires $(N_{\text{cities}} + 1) * N_{resting}$ neurons and $N(3N_{resting} - 1) + (N_{\text{cities}} - 1) * (N_{\text{cities}} - 1) * N_{resting}$ number of synapses.

Reading out the current assignment to a variable can be done based on the activity of the principal neurons which take part in the WTA circuit (same as for 3-SAT). Note that in this case each variable has N_{cities} values and therefore it multiple neurons within the same WTA could be active. When this happens, the value of the associated RV is briefly undefined. The performance of the network at some time is calculated as the ratio of the optimal path and the current path represented by the network. In order for the currently represented path to be valid all variables have to properly defined and each value (city) has to appear at least once. Although this is not *always* the case, exceptions occur rarely and therefore are not visible in performance plots.

For solving symmetric planar TSP experiments in Figure 3.3 we used the following setup: $\tau = 20e - 3$ and refractory period of 20ms for all neurons, $b_{nrn} = -0.3$, $b_P = 100$, $b_N = -100$, $b_{inh} = -10$, $w_{inh} = -100$, $w_{exc} = 100$, $w_{unique} = -14.2$, $w_{scale} = 20.8$, $w_{offset} = -6.6$, $N_{resting} = 3$, with rectangular PSPs of 20ms duration without transmission delays for all synapses. The value of the first variable (the first step) was fixed to the first city.

For solving asymmetric TSP problems we used the same architecture but slightly different parameters: $b_{nrn} = 1.3$, $w_{unique} = -14.1$, $w_{offset} = -7.9$, $N_{resting} = 8$.

For the comparative analysis between asymmetric and symmetric sampler (Neural Sampling (NS) vs. Boltzmann machine (BM)) we used exactly the same weights, biases and architecture as described above. The only difference here was that no inhibitory neurons were used, so that WTAs were implemented via direct inhibition connections between neurons taking part instead of bi-synaptic connections via inhibitory neurons.

The comparison of the number of state changes between BM and NS implementations was done based on 100 runs, each of which was simulated for 100.000 state changes. In each run and after every state changes we evaluated the current network state, and checked how many RVs were properly defined or not. Combining all runs in each case, we calculated how often transitions occurred in each sampler to states with different numbers $N_{undef} = 0, \ldots, N_{\text{cities}} + N_{resting}$ of undefined RVs. Based on this information we constructed corresponding histograms for BM and NS. To highlight the differences between the two implementations, we calculated the ratios between the normalized histogram values for NS and BM (Figure 3.3C). For the convergence speed comparison, in each run we calculated after each state change the cumulative minimum and mean performance during the whole time leading up to that state change. This was first done for each of the 100 network runs individually. The results were then averaged for each number of steps over all runs.

Appendix D

Appendix to Chapter 4: Emergent Coding of Features and Bayesian Computation in Ensembles of Pyramidal Cells with Relaxed Lateral Inhibition

Contents

D.1 Generative model for extraction of multiple hidden causes

The generative model for extraction of multiple hidden causes contains visible variables $y_1, \ldots, y_N \in \{0, 1\}$ and hidden variables $z_1, \ldots, z_M \in \{0, 1\}$. The probabilistic model is given as:

$$p(y_i = 0|\boldsymbol{z}, \boldsymbol{\lambda}) = \prod_{m=1}^{M} \lambda_{im}^{z_m}, \tag{D.1}$$

where λ_{im} is in the range $(0, 1]$ and gives the probability that the input y_i is not active given that z_m is active. By defining $w_{im} = -\log \lambda_{im}$, where w_{im} is in the range $[0, \infty)$, we get

$$p(y_i = 0|\boldsymbol{z}, \boldsymbol{W}) = \prod_{m=1}^{M} e^{-w_{im} z_m} = e^{-\bar{\boldsymbol{w}}_i^T \boldsymbol{z}}. \tag{D.2}$$

Individual visible variables are independent, given the hidden variables. Hence, we have

$$p(\boldsymbol{y}|\boldsymbol{z},\boldsymbol{W}) = \prod_{i=1}^{N} p(y_i=1|\boldsymbol{z},\boldsymbol{W})^{y_i} p(y_i=0|\boldsymbol{z},\boldsymbol{W})^{(1-y_i)} \tag{D.3}$$

$$= \prod_{i=1}^{N} e^{-\overline{\boldsymbol{w}}_i^T \boldsymbol{z}(1-y_i)}(1-e^{-\overline{\boldsymbol{w}}_i^T \boldsymbol{z}})^{y_i} \tag{D.4}$$

We approximate $1-e^{-\overline{\mathbf{w}}_i^T \mathbf{z}}$ with $1-e^{-\overline{\mathbf{w}}_i^T \mathbf{z}} \approx e^{-1/(2\overline{\mathbf{w}}_i^T \mathbf{z}+b)}$, which holds very good for small $\overline{\mathbf{w}}_i^T \mathbf{z}$. This gives us:

$$\tilde{p}(\mathbf{y}\,|\,\mathbf{z},\boldsymbol{W}) = \prod_{i=1}^{N} e^{-\overline{\mathbf{w}}_i^T \mathbf{z}+y_i(\overline{\mathbf{w}}_i^T \mathbf{z}-\frac{1}{2\overline{\mathbf{w}}_i^T \mathbf{z}+b})} \tag{D.5}$$

in the following we consider realizations of E and M steps of Stochastic Online EM algorithm. For E-step, given a new input pattern $\boldsymbol{y}^t$ at time t, one needs to draw a sample $\boldsymbol{z}^t \sim p(\boldsymbol{z}|\boldsymbol{y}^t,\boldsymbol{W})$. To do so we use NCC condition from neural sampling theory in order to calculate the membrane potential of neurons in recurrent spiking neural network required such that it samples from desired distribution.

The membrane potential of a neuron is calculated as a log-odd ratio (NCC condition):

$$\begin{aligned} u_k &= log\frac{p(z_k=1|\,\mathbf{z}_{\backslash k},\mathbf{y})}{p(z_k=0|\,\mathbf{z}_{\backslash k},\mathbf{y})} \\ &= \pi + J_{\backslash k}^T\,\mathbf{z}_{\backslash k} + \sum_i^N [(y_i-1)w_{ik} + y_i(\frac{1}{\sum_{m\neq k}^M w_{im}z_m + b} - \frac{1}{\sum_{m\neq k}^M w_{im}z_m + w_{ik} + b})] \end{aligned}$$

with $J_{kl}=J_{lk}=-\frac{1}{\sigma^2}$ for $k\neq l$ and $\pi=\frac{2K-1}{2\sigma^2}$.
Here we linearize the complex calculation of difference of fractions with γw_{ik}. This approximation is good for small w_{ik} but it heavily depends on b and current activation of circuit, and it results with

$$u_k \approx \pi + J_{\backslash k}^T\,\mathbf{z}_{\backslash k} + \sum_i^N (\gamma y_i - 1)w_{ik} \tag{D.6}$$

where γ is a free parameter.

If there is some noise in the input means $\lambda_{im} > \rho > 0$ and $w_{im} < -\log\rho$, which is equivalent to $p(y_i=0|\,\mathbf{z})>0$ and means that there is some maximum value of w_{im}. Under assumption that the relevant weights for discrimination of pattern will reach the maximum value ($-\log\rho$) and others 0, we make approximation $\sum_i^N w_{ik} \approx const = \delta_k$.

$$u_k = \pi + J_{\backslash k}^T\,\mathbf{z}_{\backslash k} + \sum_i^N \gamma y_i w_{ik} - \delta_k \tag{D.7}$$

The δ_k can be incorporated into the prior π resulting with $\pi_k = \pi - \delta_k$, which is equivalent to using neuron specific intrinsic excitability and inhibition strength. Finally the inference can be done as:

$$u_k = \pi_k + J_{\setminus k}^T \mathbf{z}_{\setminus k} + \gamma\, \mathbf{y}^T \mathbf{w}_k \tag{D.8}$$

If one assumes that all $\mathbf{w}_k$ are the same, which is the case when all patterns are of same complexity (sum of rates from all channel) as in the case of bar rate patterns, one ends up with eq. 4.2 for membrane potential. This means that all neurons will prefer to learn patterns of similar complexity. If we allow for different constants (β_k) this should force the neurons to learn patterns of different complexity.

In order to perform the E-step one needs to perform an update of parameters as $\Delta\boldsymbol{W} \propto \partial_{\boldsymbol{W}} \log p(\boldsymbol{y}^t, \boldsymbol{z}^t | \boldsymbol{W})$. As prior $p(\boldsymbol{z})$ does not depend on $\boldsymbol{W}$, this is the same as taking derivative of $\log p(\boldsymbol{y}^t | \boldsymbol{z}^t, \boldsymbol{W})$.

Maximization of the log-likelihood with respect to w_{im} gives

$$\begin{aligned}
\frac{\partial}{\partial w_{im}} \log \tilde{p}(\mathbf{y} \,|\, \mathbf{z}, \boldsymbol{W}) &= \frac{\partial}{\partial w_{im}} \log \prod_{i=1}^{N} e^{-\overline{\mathbf{w}}_i^T \mathbf{z} + y_i(\overline{\mathbf{w}}_i^T \mathbf{z} - \frac{1}{2\overline{\mathbf{w}}_i^T \mathbf{z} + b})} \\
&= \frac{\partial}{\partial w_{im}} \sum_{i=1}^{N} -\overline{\mathbf{w}}_i^T \mathbf{z} + y_i(\overline{\mathbf{w}}_i^T \mathbf{z} - \frac{1}{2\overline{\mathbf{w}}_i^T \mathbf{z} + b}) \\
&= -z_m + y_i(z_m + \frac{2z_m}{(2\overline{\mathbf{w}}_i^T \mathbf{z} + b)^2})
\end{aligned}$$

Now we can update the parameters of the model in order to increase log-likelihood for the given configuration. To update, we use gradient descant technique, where gradient of log-likelihood with respect to w_{im} gives us the learning rule

$$\Delta w_{im} = \eta z_m (y_i - 1 + \frac{y_i}{(\sqrt{2}\overline{\mathbf{w}}_i^T \mathbf{z} + \theta)^2}) \tag{D.9}$$

where $\theta = b/\sqrt{2}$. This learning rule is not local as it requires information about activation of all output neurons as well as values of all synapse weights originating from input neuron i. In order to make this biologically plausible we approximate this by using only locally available information at synapse:

$$\Delta w_{im} = \eta z_m (y_i - 1 + \frac{y_i}{(\sqrt{2}2 w_{im} + \theta)^2}) \tag{D.10}$$

where η is learning rate and b is bias which limits weights.

D.2 Creation of rate and spatio-temporal patterns

P denotes the set of patterns, where each one has the length l_i. Each pattern consists of N_c channels (input neurons) with $r_{i,1,t}, .., r_{i,N_c,t}$ firing rates, where i denotes pattern and $t \in [0, l_i]$ denotes time.

In the case of bar rate pattern of shape $n \times n = N_c$, all channels have constant firing rate during the whole length of pattern. Among all channels there is a subset S of channels, $|S| = n$, whose channels have high rate (R_h) while others have low rate (R_l), $r_{i,S,t} = R_h$, $r_{i,\setminus S,t} = R_l$. Subset S is such that all elements belong to a specific row or a column of $n \times n$ shaped pattern.

The spatio-temporal or time varying rate patterns consist of channels where each one of them has changing rate during the whole length of pattern. The firing rate for each channel is calculated as $r_{i,j,t} = \exp(x_t)$, where x_t is a membrane potential of a neuron, which is calculated based on independent Ornstein-Uhlenbeck process (OU). The OU process is defined as $dx_t = \Theta_{OU}(\mu_{OU} - x_t)dt + \sigma_{OU} dW_t$, where t denotes time, $\Theta_{OU} > 0$ is the changing rate (convergence speed), $\mu_{OU} > 0$ is the mean, $\sigma_{OU} > 0$ is the noise variance and W_t is the Wiener process.

D.3 Creation of complex input spike trains

The input spike train is made by superimposing a number of patterns, or more precisely their rates, from the set of patterns P, where the maximum number of patterns(C_M) that can be overlapped at any given point of time is constrained. Superposition of patterns can be done linear on nonlinear. In the case of linear superposition the final rate of a particular channel is the sum of rates of patterns that overlap at given time (r_i), while in the case of nonlinear superposition it is calculated as $f_L + f_H * \sigma(r_i, \mu_S, \omega_S)$, where f_L is the lowest rate, f_H the highest rate, $\sigma(x, \mu_S, \omega_S) = 1/(1 + exp(-(x - \mu_S)/\omega_S)))$, and $\mu_S = f_L + f_H/2$, $\omega_S = f_H/(2\alpha)$, with α being the width of sigmoid function.

Additionally, the patterns are allowed to have an arbitrary relative timing. In order to create such an input we introduce channels $ch_{1,t}, .., ch_{C_M,t}$, where $ch_{i,t} \in \{0, .., |P|\}$ and t denotes time. Each channel $ch_{i,t}$ has a certain probability of being active pa_i, where active means that one of the patterns is presented in that channel and therefore $ch_{i,t} > 0$ (value depends on the pattern's index). Probability of some channel ch_i being on is $pa_i = u/l$, where u is the total time being active and l the total duration of input (simulation time). The total time of being active can be approximated as $u \approx n_s d$, yielding $pa_i = n_s d/l$, where n_s is the number of times one of patterns were active during simulation time l in a particular channel and d is average length of all patterns in P, $d = \sum l_i/|P|$. Finally, the probability of activating one of patterns from $|P|$ in channel ch_i is $p_i = n_s/(l - n_s(d-1))$, which together with the previous expression gives $p_i = \sigma(\tilde{u}_i - \log \tilde{\tau})$, where $\tilde{u}_i = \log(pa_i/(1 - pa_i))$, $\tilde{\tau} = d$ and $\sigma(x)$ is a standard sigmoid function ($\sigma(x) = 1/(1 + exp(-x))$). Note that this is equivalent to the neural sampling with only difference that activity of channels (neurons) are not binary but rather consist of cause information (pattern's index), while $\tilde{\tau}$ is an average of all patterns length (pattern specific $\tilde{\tau}$). Note that the this could also be solved with number of channels (ch)= $|P|$, which would ensure that activity is binary, but additional mechanism that ensures maximum and minimum number of overlapping patterns would be needed.

D.4 Details to the performance measure

According to the neural sampling theory (Buesing et al., 2011) the neuron is active for period τ after it spikes. We refer to the binary description of this behavior in discrete time as neuron's activity $na_{i,t} \in \{0,1\}$, where i denotes neuron, t time and $na_{i,t} = 1$ during $[spike, spike{+}\tau]$ period (this is the same as looking for periods when auxiliary variable $\zeta_i > 0$, for details look in (Buesing et al., 2011)). Grouped activity is obtained by performing logical OR operation between individual neurons activity belonging to the same group of neurons. In similar way as neuron's activity we define pattern's activity, $pa_{i,t} \in \{0,1\}$, where i denotes pattern, t time and $pa_{i,t} = 1$ during $[start + epsp/2, start + l_i + epsp/2 + tau])$ period, where $start$ denotes time of start of pattern's representation, $epsp$ denotes length of used excitatory PSP (EPSP) shape (this is the time period where EPSP shape has significant value) and l_i is the length of the pattern.

Grouping of neurons is done according to the precision measure (van Rijsbergen, 1974) which is defined as

$$Precision = \frac{TP}{TP + FP} \tag{D.11}$$

TP (True-Positive) is the number of times prediction was correct, while FP (False-Positive) is the number of times prediction was wrong. The precision measure between pattern's activity $pa_{i,t}$ and neural activity $na_{j,t}$ is calculated as

$$Precision = \frac{\sum_t na_{i,t} pa_{j,t}}{\sum_t na_{i,t} pa_{j,t} + \sum_t [\overline{na}_{i,t} pa_{j,t}]}, \tag{D.12}$$

where line over $na_{i,t}$ denotes binary NOT operation.

For sufficiently long patterns (here we assume there is some temporal structure) we calculate the correlation measure (Phi coefficient) between groups of neurons activity (grouped activity) and patterns activity, where neurons are grouped according to the precision measure. For the case of short rate patterns (bars) assumption is that there is no temporal structure, therefore we do not group neurons, but we rather directly calculate correlation between neurons' and patterns' activity.

To measure the quality of the model we use the average correlation of all and the average correlation of all found patterns. For each neuron we find the pattern with whom it has the highest correlation. Note that it is possible that there is no such neuron if the model did not discover (learn) certain pattern. Then the average correlation of all patterns is given as the ratio between the sum of correlations for patterns which are learned and the number of all patterns, while the average correlation of found pattern is given as the ratio between the sum of correlations for patterns that were learned and the number of those learned patterns.

D.5 Details to simulations

All simulations of SWTA networks with direct implementation of inhibition, as well as analysis, were performed in Python. Simulations of SWTA networks which implemented biological realistic inhibition were done in PCSIM, a neural simulator written in C++ that provides a Python interface, which was extended in order to support simulation of SWTA model. Optimizations of SWTA networks, as well as exploration of theirs properties were done with ZLIB (Chapter 5), a library for parallelization and optimization, developed by me within the scope of this thesis.

Details to experiment with superposition of bar rate patterns

In this experiment we train SWTA network with 120s of input made by nonlinear superposition of at most 3 out of 16 bar rate patterns (30ms) with constant rates per channel and arbitrary relative timing. We use SWTA network with following setup: $N = 64$, $M = 64$, $\mu = -7.$, $\sigma = 0.2$, $\eta = 0.04$, $b = 1.$, $\gamma = 2.$, additive alpha EPSP of 30ms length and top value at 1.15, additive double exponential inhibitory PSP (IPSP) shape($t_{rise} = 1ms, t_{fall} = 10ms$), $\tau = 10ms$, delay of $4ms$, maximum weight = 2., the same β_k for all neurons incorporated into μ and thresholding for correlation at 0.3 (min correlation, otherwise the neuron is considered as a free neuron). Inhibition is implemented via direct connections between SWTA neurons. The patterns have arbitrary relative timing and at most 3 patterns are allowed to overlap at the same time. The patterns activity probabilities are $[0.6, 0.6, 0.7]$, which results in the following distribution of number of overlapping patterns in increasing order (from 0 to 3) $[0.05, 0.26, 0.44, , 0.25]$.

Details to emergent extraction of time-varying rate patterns from superpositions in complex input streams

Here we consider nonlinear superposition of two spatio-temporal patterns (150ms) with varying rates per channel and arbitrary relative timing shown for 100s to the SWTA network with following parameters: $N = 64$, $M = 100$, $\mu = -5.$, $\sigma = 0.5$, $\eta = 0.07$, $b = 0.8$, $\gamma = 2.$, additive alpha EPSP of 30ms length and top value at 1., additive double exponential IPSP shape($t_{rise} = 1ms, t_{fall} = 10ms$), $\tau = 10ms$, delay of $3ms$, maximum weight = 2., the same β_k for all neurons incorporated into μ and thresholding for grouping neurons at 0.4 (min precision). Each pattern consists of 64 channels and each channel has varying rates limited to 50Hz generated by independent Ornstein-Uhlenbeck process with params: $\mu = 0, \theta = 5., \sigma = 0.5$. The patterns have arbitrary relative timing and at most 2 patterns are allowed to overlap at the same time. The patterns activity probabilities are $[0.5, 0.5]$, which results in the following distribution of number of overlapping patterns in increasing order (from 0 to 2) $[0.25, 0.5, 0.25]$.

Details to computational properties of microcircuit motif with relaxed lateral inhibition

For all experiments we used the same SWTA network setup. The SWTA network was trained with 120s of input made by nonlinear superposition of at most 3 out of 16 bar rate patterns (30ms). All measured were calculated based on 10runs. SWTA network parameters: $N = 64$, $M = 100$,$\mu = -7$, $\sigma = 0.2$, $\eta = 0.04$, $b = 1.$, $\gamma = 2.$, additive alpha EPSP of 30ms length and top value at 1.15, additive double exponential IPSP shape($t_{rise} = 1ms, t_{fall} = 10ms$), inhibition delay 4ms, $\tau = 10ms$, maximum weight = 5., the same β_k for all neurons incorporated into μ and thresholding for correlation at 0.3

Bibliography

Abeles, M., Bergman, H., Gat, I., Meilijson, I., Seidemann, E., Tishby, N., and Vaadia, E. (1995). Cortical activity flips among quasi-stationary states. *Proc Natl Acad Sci U S A*, 92(19):8616–8620. 10, 36

Aire Technologies (2013). Sudoku solutions. http://www.sudoku-solutions.com. Accessed February 27, 2013. 110

Allen, C. and Stevens, C. F. (1994). An evaluation of causes for unreliability of synaptic transmission. *PNAS*, 91:10380–3. 6

Arieli, A., Sterkin, A., Grinvald, A., and Aertsen, A. (1996). Dynamics of ongoing activity: explanation of the large variability in evoked cortical responses. *Science*, 273:1868–1871. 16

Arnsten, A. F. T., Wang, M. J., and Paspalas, C. D. (2012). Neuromodulation of thought: flexibilities and vulnerabilities in prefrontal cortical network synapses. *Neuron*, 76:223–239. 30, 33

Avermann, M., Tomm, C., Mateo, C., Gerstner, W., and Petersen, C. (2012). Microcircuits of excitatory and inhibitory neurons in layer 2/3 of mouse barrel cortex. *Journal of neurophysiology*, 107(11):3116–3134. 54

Bastos, A. M., Usrey, W. M., Adams, R. A., Mangun, G. R., Fries, P., and Friston, K. J. (2012). Canonical microcircuits for predictive coding. *Neuron*, 76(4):695–711. 8

Berkes, P., Orban, G., Lengyel, M., and Fiser, J. (2011). Spontaneous cortical activity reveals hallmarks of an optimal internal model of the environment. *Science*, 331:83–87. 7, 9, 10, 15, 35, 36, 114

Biere, A. (2009). *Handbook of satisfiability*, volume 185. IOS Press. 45, 49

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer, New York. 61, 120, 121

Blake, R. and Logothetis, N. K. (2002). Visual competition. *Nature Reviews Neuroscience*, 3(1):13–21. 23

Borovkov, A. A. (1998). *Ergodicity and stability of stochastic processes*. Hoboken, NJ: Wiley. 94

Borovkov, K., Decrouez, G., and Gilson, M. (2012). On stationary distributions of stochastic neural networks. 7, 29, 34, 88, 92, 94, 101

Borst, J. G. (2010). The low synaptic release probability in vivo. *Trends in Neurosciences*, 33(6):259–266. 6, 90

Boykov, Y., Veksler, O., and Zabih, R. (2001). Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1222–1239. 33

Branco, T. and Staras, K. (2009). The probability of neurotransmitter release: variability and feedback control at single synapes. *Nature Reviews Neuroscience*, 10(5):373–383. 90

Brémaud, P. and Massoulié, L. (1996). Stability of nonlinear Hawkes processes. *The Annals of Probability*, 24(3):1563–1588. 7, 29, 101

Brooks, S., Gelman, A., Jones, G., and Meng, X.-L. (2010). *Handbook of Markov Chain Monte Carlo: Methods and Applications*. Chapman & Hall. 17, 105, 106

Brooks, S. P. and Gelman, A. (1998). General methods for monitoring convergence of iterative simulations. *Journal of Computational and Graphical Statistics*, 7(4):434–455. 106

Brooks, S. P. and Roberts, G. O. (1998). Assessing convergence of Markov chain Monte Carlo algorithms. *Statistics and Computing*, 8(4):319–335. 105

Buesing, L., Bill, J., Nessler, B., and Maass, W. (2011). Neural dynamics as sampling: A model for stochastic computation in recurrent networks of spiking neurons. *PLoS Computational Biology*, 7(11):e1002211. 7, 15, 23, 31, 32, 61, 79, 113, 114, 115, 116, 117, 133

Buzsaki, G. (2009). *Rhythms of the brain*. Oxford: Oxford University Press. 21, 36

Buzsáki, G. (2010). Neural syntax: cell assemblies, synapsembles, and readers. *Neuron*, 68(3):362–385. 10, 23

Churchland, M. M. and Abbott, L. (2012). Two layers of neural variability. *Nature neuroscience*, 15(11):1472–1474. 35

Clarke, P. G. (2012). The limits of brain determinacy. *Proc Biol Sci*, 279(1734):1665–1674. 6, 35

Cook, S. A. (1971). The complexity of theorem-proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, STOC '71, pages 151–158, New York, NY, USA. ACM. 49

Cowles, M. K. and Carlin, B. P. (1996). Markov chain Monte Carlo convergence diagnostics: A comparative review. *Journal of the American Statistical Association*, 91(434):883–904. 17, 105

Craenen, B., Eiben, A., and van Hemert, J. I. (2003). Comparing evolutionary algorithms on binary constraint satisfaction problems. *Evolutionary Computation, IEEE Transactions on*, 7(5):424–444. 26

Davenport, A., Tsang, E., Wang, C. J., and Zhu, K. (1994). GENET: a connectionist architecture for solving constraint satisfaction problems by iterative improvement. In *Proceedings of the National Conference on Artificial Intelligence*, pages 325–325. John Wiley & Sons Ltd. 9

Denison, S., Bonawitz, E., Gopnik, A., and Griffiths, T. L. (2009). Preschoolers sample from probability distributions. In *Proceedings of the 32nd Annual Conference of the Cognitive Science Society*, volume 29, pages 1–10. 34

Destexhe, A., Rudolph, M., Fellous, J.-M., and Sejnowski, T. J. (2001). Fluctuating synaptic conductances recreate *in vivo*-like activity in neocortical neurons. *Neuroscience*, 107(1):13–24. 104

Doeblin, W. (1937). Sur le propriétés asymtotiques de mouvement régis par certain types de chaînes simples. *Bull Math Soc Roumaine Sci*, 39:(1) 57–115, (2) 3–61. 92

Douglas, R. J. and Martin, K. A. (2004). Neuronal circuits of the neocortex. *Annual Reviews of Neuroscience*, 27:419–451. 2, 8, 26, 28, 52

Down, D., Meyn, S., and Tweedie, R. (1995). Exponential and uniform ergodicity of Markov processes. *The Annals of Probability*, 23(4):1671–1691. 95, 96, 97, 99

Doya, K., Ishii, S., Pouget, A., and Rao, R. P. N. (2007). *Bayesian Brain: Probabilistic Approaches to Neural Coding*. MIT-Press. 7

Dragoi, G. and Buzsaki, G. (2006). Temporal encoding of place sequences by hippocampal cell assemblies. *Neuron*, 50(1):145–157. 8, 21, 22, 23

Durstewitz, D. (2006). A few important points about dopamine's role in neural network dynamics. *Pharmacopsychiatry*, 39:572–575. 30

Durstewitz, D. (2009). Implications of synaptic biophysics for recurrent network dynamics and active memory. *Neural Networks*, 22:1189–1200. 30

Durstewitz, D. and Deco, G. (2008). Computational significance of transient dynamics in cortical networks. *European Journal of Neuroscience*, 27:217–227. 32

El Adlouni, S., Favre, A.-C., and Bobée, B. (2006). Comparison of methodologies to assess the convergence of Markov chain Monte Carlo methods. *Computational Statistics & Data Analysis*, 50(10):2685–2701. 105

Engel, A. K., Fries, P., and Singer, W. (2001). Dynamic predictions: oscillations and synchrony in top-down processing. *Nat Rev Neurosci*, 2(10):704–16. 20

Ercsey-Ravasz, M. and Toroczkai, Z. (2012). The chaos within Sudoku. *Scientific Reports*, 2. 109

Faisal, A. A., Selen, L. P. J., and Wolpert, D. M. (2008). Noise in the nervous system. *Nature Reviews Neuroscience*, 9(4):292–303. 1, 6, 40

Fiser, J., Berkes, P., Orban, G., and Lengyel, M. (2010). Statistically optimal perception and learning: from behavior to neural representation. *Trends in Cognitive Sciences*, 14(3):119–130. 7, 12, 15, 34

Földiak, P. (1990). Forming sparse representations by local anti-hebbian learning. *Biological cybernetics*, 64(2):165–170. 53, 59, 68

Fox, M. D., Snyder, A. Z., Vincent, J. L., and Raichle, M. (2007). Intrinsic fluctuations within cortical systems account for intertrial variability in human behavior. *Neuron*, 56(1):171–184. 6

Fregnac, Y. (2003). Hebbian synaptic plasticity. In Arbib, M. A., editor, *The Handbook of Brain Theory and Neural Networks*, pages 515–522. MIT Press, Cambridge, MA. 2

Friston, K. (2010). The free-energy principle: a unified brain theory? *Nature Reviews Neuroscience*, 11(2):127–138. 7

Fu, Z., Mahajan, Y., and Malik, S. (2004). New features of the sat04 versions of zchaff. *SAT Competition*. 73, 126

Garey, M. and Johnson, D. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Mathematical Sciences. Freeman, New York, NY. 33

Gelman, A., Carlin, J. B., Stern, H. S., and Rubin, D. B. (2004). *Bayesian Data Analysis, Second Edition (Chapman & Hall/CRC Texts in Statistical Science)*. Chapman and Hall/CRC, 2 edition. 17, 106

Gelman, A. and Rubin, D. B. (1992). Inference from iterative simulation using multiple sequences. *Statistical Science*, 7(4):457–472. 17, 105, 106

Gershman, S., Vul, E., and Tenenbaum, J. (2012). Multistability and perceptual inference. *Neural Computation*, 24(1):1–24. 23, 34

Gershman, S. J., Vul., E., and Tenenbaum, J. (2009). Perceptual multistability as Markov chain Monte Carlo inference. *Advances in Neural Information Processing Systems*, 22:611–619. 1

Gerstner, W. and Kistler, W. (2002). *Spiking Neuron Models*. Cambridge University Press, Cambridge. 40, 103

Geweke, J. (1991). Evaluating the accuracy of sampling-based approaches to the calculation of posterior moments. Staff Report 148, Federal Reserve Bank of Minneapolis. 105

Gjoka, M., Kurant, M., Butts, C. T., and Markopoulou, A. (2010). Walking in facebook: A case study of unbiased sampling of osns. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–9. IEEE. 17, 105

Gray, R. M. (2009). *Probability, random processes, and ergodic properties.* New York: Springer. 102

Griffiths, T. L. and Tenenbaum, J. B. (2006). Optimal predictions in everyday cognition. *Psychological Science*, 17(9):767–773. 1, 34

Gupta, A., der Meer, M. A. A., Touretzky, D. S., and Redish, A. D. (2012). Segmentation of spatial experience by hippocampal theta sequences. *Nature Neuroscience*, 15(7):1032–1039. 8, 22, 23, 36

Gupta, A., Wang, Y., and Markram, H. (2000). Organizing principles for a diversity of gabaergic interneurons and synapses in the neocortex. *Science*, 287(5451):273–278. 13

Gutin, G. and Punnen, A. (2002). *The traveling salesman problem and its variations*, volume 12. Springer. 47

Habenschuss, S., Jonke, Z., and Maass, W. (2013a). Stochastic computations in cortical microcircuit models. *PLoS Computational Biology*, 9(11):e1003311. 53, 55, 68, 79, 114

Habenschuss, S., Puhr, H., and Maass, W. (2013b). Emergence of optimal decoding of population codes through stdp. *Neural computation*, 25(6):1371–1407. 52

Haeusler, S. and Maass, W. (2007). A statistical analysis of information-processing properties of lamina-specific cortical microcircuit models. *Cerebral Cortex*, 17(1):149–162. 8, 12, 13, 17, 19, 104, 107

Haeusler, S., Schuch, K., and Maass, W. (2009). Motif distribution, dynamical properties, and computational performance of two data-based cortical microcircuit templates. *Journal of Physiology, Paris*, 103(1-2):73–87. 8

Haider, B., Häusser, M., and Carandini, M. (2013). Inhibition dominates sensory responses in the awake cortex. *Nature*, 493(7430):97–100. 25

Harris, K., Csicsvari, J., Hirase, H., Dragoi, G., and Buzsáki, G. (2003). Organization of cell assemblies in the hippocampus. *Nature*, 424(6948):552–556. 36

Harvey, C. D., Coen, P., and Tank, D. W. (2012). Choice-specific sequencis in parietal cortex during a virtual-navigation decision task. *Nature*, 484:62–68. 14

Hinton, G. E., Sejnowski, T. J., and Ackley, D. H. (1984). Boltzmann machines: constraint satisfaction networks that learn. Technical Report CMS-CS-84-119, CMU Computer Science Department. 8

Hopfield, J. and Tank, D. (1986). Computing with neural circuits: a model. *Science*, 233(4764):625–633. 31

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *PNAS*, 79:2554–2558. 31

Hoyer, P. O. and Hyvärinen, A. (2003). Interpreting neural response variability as Monte Carlo sampling of the posterior. *Advances in Neural Information Processing Systems 15*, pages 293–300. 7, 23

Huk, A. and Shadlen, M. (2005). Neural activity in macaque parietal cortex reflects temporal integration of visual motion signals during perceptual decision making. *The Journal of Neuroscience*, 25(45):10420–10436. 7

Jezek, K., Henriksen, E., Treves, A., Moser, E., and Moser, M. (2011). Theta-paced flickering between place-cell maps in the hippocampus. *Nature*, 478(7368):246–249. 8, 23

Jiang, X., Wang, G., Lee, A. J., Stornetta, R. L., and Zhu, J. J. (2013). The organization of two new cortical interneuronal circuits. *Nature Neuroscience*, 16(2):210–218. 103

Jolivet, R., Rauch, A., Lüscher, H., and Gerstner, W. (2006). Predicting spike timing of neocortical pyramidal neurons by simple threshold models. *Journal of Computational Neuroscience*, 21:35–49. 13, 55, 103, 113

Jones, B., Stekel, D., Rowe, J., and Fernando, C. (2007). Is there a liquid state machine in the bacterium escherichia coli? *Artificial Life*. ALIFE'07, IEEE Symposium. 36

Kandel, E. R., Schwartz, J. H., and Jessel, T. M. (1991). *Principles of Neural Science*. Prentice-Hall, third edition. 39

Karlsson, M., Tervo, D. G., and Karpova, A. Y. (2012). Network resets in medial prefrontal cortex mark the onset of behavioral uncertainty. *Science*, 338(6103):135–139. 6

Karp, R. (1972). *Reducibility among combinatorial problems*. Springer. 49

Kass, R. E., Carlin, B. P., Gelman, A., and Neal, R. M. (1998). Markov Chain Conte Carlo in practice: A roundtable discussion. *The American Statistician*, 52(2):93–100. 17, 106

Kelemen, E. and Fenton, A. A. (2010). Dynamic grouping of hippocampal neural activity during cognitive control of two spatial frames. *BLoS Biology*, 8(6):e1000403. 6, 22

Kenet, T., Bibitchkov, D., Tsodyks, M., Grinvald, A., and Arieli, A. (2003). Spontaneously emerging cortical representations of visual attributes. *Nature*, 425(6961):954–956. 6

Kim, C. Y. and Blake, R. (2005). Psychophysical magic: rendering the visible 'invisible'. *Trends in Cognitive Sciences*, 9(8):381–388. 6

Kirkpatrick, S., C.D., G., and Vecchi, M. (1983). Optimization by simmulated annealing. *science*, 220(4598):671–680. 45

Klampfl, S., David, S. V., Yin, P., Shamma, S. A., and Maass, W. (2012). A quantitative analysis of information about past and present stimuli encoded by spikes of A1 neurons. *Journal of Neurophysiology*, 108:1366–1380. 30

Klampfl, S. and Maass, W. (2013). Emergence of dynamic memory traces in cortical microcircuit models through stdp. *The Journal of Neuroscience*, 33(28):11515–29. 52

Koller, D. and Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques (Adaptive Computation and Machine Learning)*. MIT Press. 13, 34

Kumar, V. (1992). Algorithms for constraint-satisfaction problems: A survey. *AI magazine*, 13(1):32. 25

Larkum, M. (2012). A cellular mechanism for cortical associations: an organizing principle for the cerebral cortex. *Trends in Neurosciences*, 951:1–11. 103

Larkum, M. (2013). The yin and yang of cortical layer 1. *Nature Neuroscience*, 16(2):114–115. 103

Leopold, D. A. and Logothetis, N. K. (1996). Activity changes in early visual cortex reflect monkeys' percepts during binocular rivalry. *Nature*, 379(6565):549–553. 6

Leopold, D. A. and Logothetis, N. K. (1999). Multistable phenomena: changing views in perception. *Trends in Cognitive Sciences*, 3(7):254–264. 6

Lewis, C. M., Baldassarre, A., Committeri, G., Romani, G. L., and Corbetta, M. (2009). Learning sculpts the spontaneous activity of the resting human brain. *PNAS*, 106(41):17558–17563. 6, 36

Lieder, F., Griffiths, T., and Goodman, N. (2013). Burn-in, bias, and the rationality of anchoring. In *Proc. of NIPS 2012*, volume 25, pages 2690–2698. MIT Press. 31

Lisman, J. E., Raghavachari, S., and Tsien, R. W. (2007). The sequence of events that underlie quantal transmission at central glutamatergic synapses. *Nature Reviews Neuroscience*, 8(8):597–609. 90

Litwin-Kumar, A. and Doiron, B. (2012). Slow dynamics and high variability in balanced cortical networks with clustered connections. *Nature neuroscience*, 15(11):1498–1505. 35

Lücke, J. and Eggert, J. (2010). Expectation truncation and the benefits of preselection in training generative models. *The Journal of Machine Learning Research*, 9999:2855–2900. 53, 68

Luczak, A., Barthó, P., and Harris, K. (2009a). Spontaneous events outline the realm of possible sensory responses in neocortical populations. *Neuron*, 62(3):413–425. 54, 63

Luczak, A., Barthó, P., and Harris, K. D. (2009b). Spontaneous events outline the realm of possible sensory responses in neocortical populations. *Neuron*, 62:413–425. 6

Luczak, A., Barthó, P., Marguet, S. L., Buzsáki, G., and Harris, K. D. (2007). Sequential structure of neocortical spontaneous activity in vivo. *PNAS*, 104(1):347–352. 10, 14, 20, 36

Luczak, A. and MacLean, J. N. (2012). Default activity patterns at the neocortical microcircuit level. *Frontiers in Integrative Neuroscience*, 6(30):doi: 10.3389/fnint.2012.00030. 10

Lynce, I. and Marques-Silva, J. (2006). Efficient haplotype inference with boolean satisfiability. In *Proceedings of the National Conference on Artificial Intelligence*, volume 21, page 104. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999. 49

Maass, W. and Markram, H. (2002). Synapses as dynamic memory buffers. *Neural Networks*, 15:155–161. 103

Maass, W., Natschlaeger, T., and Markram, H. (2002). Real-time computing without stable states: a new framework for neural computation based on perturbations. *Neural Computation*, 14(11):2531–2560. 30, 32

Maass, W. and Sontag, E. (1999). Analog neural nets with Gaussian or other common noise distributions cannot recognize arbitrary regular languages. *Neural Computation*, 11:771–782. 93

Markram, H., Wang, Y., and Tsodyks, M. (1998). Differential signaling via the same axon of neocortical pyramidal neurons. *PNAS*, 95:5323–5328. 13, 103

Massoulié, L. (1998). Stability results for a general class of interacting point processes dynamics, and applications. *Stochastic processes and their applications*, 75(1):1–30. 101

Mazor, O. and Laurent, G. (2005). Transient dynamics versus fixed points in odor representations by locust antennal lobe projection neurons. *Neuron*, 48:661–673. 14

Meyn, S. P. and Tweedie, R. L. (1993). Stability of markovian processes ii: Continuous-time processes and sampled chains. *Advances in Applied Probability*, pages 487–517. 97

Michalewicz, Z. and Fogel, D. (2000). *Escaping Local Optima*. Springer. 45

Mountcastle, V. B. (1998). *Perceptual Neuroscience: The Cerebral Cortex*. Harvard University Press, Cambridge. 8

Nessler, B., Pfeiffer, M., Buesing, L., and Maass, W. (2013). Bayesian computation emerges in generic cortical microcircuits through spike-timing-dependent plasticity. *PLoS Computational Biology*, 9(4):e1003037. 52, 53, 55, 56, 68

Nikolic, D., Haeusler, S., Singer, W., and Maass, W. (2009). Distributed fading memory for stimulus properties in the primary visual cortex. *PLoS Biology*, 7(12):1–19. 30

Okun, M., Yger, P., Marguet, S. L., Gerard-Mercier, F., Benucci, A., Katzner, S., and Harris, K. D. (2012). Population rate dynamics and multineuron firing patterns in sensory cortex. *Journal of Neuroscience*, 32(48):17108–17119. 35

Pecevski, D. (2013). Nevesim – an event based simulator for networks of spiking neurons. http://sim.igi.tugraz.at/. 110

Pecevski, D., Buesing, L., and Maass, W. (2011). Probabilistic inference in general graphical models through sampling in stochastic networks of spiking neurons. *PLoS Computational Biology*, 7(12):e1002294. 7, 15, 31, 79, 114, 115

Pecevski, D., Natschlaeger, T., and Schuch, K. (2009). PCSIM: a parallel simulation environment for neural circuits fully integrated with python. *Frontiers in Neuroinformatics*, 3. doi:10.3389/neuro.11.011.2009. 103

Perez, F. and Granger, B. E. (2007). Ipython: a system for interactive scientific computing. *Computing in Science & Engineering*, 9(3):21–29. 76

Pipa, G., Staedtler, E. S., Rodriguez, E. F., Waltz, J. A., Muckli, L., Singer, W., Goebel, R., and Munk, M. H. J. (2009). Performance- and stimulus-dependent oscillations in monkey prefrontal cortex during short-term memory. *Frontiers in Integrative Neuroscience*, 3(25). 22

Plischke, M. and Bergersen, B. (2006). *Equilibrium statistical physics*. World Scientific. 115

Potjans, T. C. and Diesmann, M. (2012). The cell-type specific cortical microcircuit: relating structure and activity in a full-scale spiking network model. *Cerebral Cortex*, page doi:10.1093/cerror/bhs358. 8

Qin, Y., Xiang, J., Wang, R., Zhou, H., Li, K., and Zhong, N. (2012). Neural bases for basic processes in heuristic problem solving: take solving sudoku puzzles as an example. *PsyCh Journal*, 1(2):101–117. 33

Rabinovich, M., Huerta, R., and Laurent, G. (2008). Transient dynamics for neural processing. *Science*, 321:45–50. 32

Raftery, A. E., Lewis, S., et al. (1992). How many iterations in the Gibbs sampler. *Bayesian Statistics*, 4(2):763–773. 105

Raichle, M. E. (2010). Two views of brain function. *Trends in Cognitive Sciences*, 14(4):180–190. 6, 32

Rasch, M. J., Schuch, K., Logothetis, N. K., and Maass, W. (2011). Statistical comparision of spike responses to natural stimuli in monkey area V1 with simulated responses of a detailed laminar network model for a patch of V1. *J Neurophysiol*, 105:757–778. 8

Rolls, E. T. and Deco, G. (2010). *The Noisy Brain: Stochastic Dynamics as a Principle of Brain Function*. Oxford Univ. Press. 32

Sakata, S. and Harris, K. (2009). Laminar structure of spontaneous and sensory-evoked population activity in auditory cortex. *Neuron*, 64(3):404–418. 54, 63

Sato, M. (1999). Fast learning of on-line EM algorithm. Technical report, ATR Human Information Processing Research Laboratories, Kyoto, Japan. 61

Savin, C., Joshi, P., and Triesch, J. (2010). Independent component analysis in spiking neurons. *PLoS computational biology*, 6(4):e1000757. 68

Schneidman, E., Berry, M., Segev, R., and Bialek, W. (2006). Weak pairwise correlations imply strongly correlated network states in a neural population. *Nature*, 440(7087):1007–1012. 114

Shadlen, M. N. and Newsome, W. T. (2001). Neural basis of a perceptual decision in the parietal cortex (area lip) of the rhesus monkey. *Journal of Neurophysiology*, 86(4):1916–1936. 15

Siapas, A. G., Lubenov, E. V., and Wilson, M. A. (2005). Prefrontal phase locking to hippocampal theta oscillations. *Neuron*, 46(1):141–151. 25

Siegel, M., Warden, M. R., and Miller, E. K. (2009). Phase-dependent neuronal coding of objects in short-term memory. *PNAS*, 106(50):21341–21346. 22, 36

Sporns, O. (2011). The human connectome: a complex network. *Annals of the New York Academy of Sciences*, 1224(1):109–125. 32

Sterzer, P., Kleinschmidt, A., and Rees, G. (2009). The neural bases of multistable perception. *Trends in cognitive sciences*, 13(7):310–318. 23

Tenenbaum, J., Kemp, C., Griffiths, T., and Goodman, N. (2011). How to grow a mind: statistics, structure, and abstraction. *Science*, 331(6022):1279–1285. 34

Thomson, A. M., West, D. C., Wang, Y., and Bannister, A. P. (2002). Synaptic connections and small circuits involving excitatory and inhibitory neurons in layers 2 - 5 of adult rat and cat neocortex: triple intracellular recordings and biocytin labelling in vitro. *Cerebral Cortex*, 12(9):936–953. 13

Truccolo, W., Eden, U., Fellows, M., Donoghue, J., and Brown, E. (2005). A point process framework for relating neural spiking activity to spiking history, neural ensemble, and extrinsic covariate effects. *Journal of neurophysiology*, 93(2):1074–1089. 113

Tsodyks, M., Kenet, T., Grinvald, A., and Arieli, A. (1999). Linking spontaneous activity of single cortical neurons and the underlying functional architecture. *Science*, 286(5446):1943–1946. 6

Turesson, H. K., Logothetis, N. K., and Hoffman, K. L. (2012). Category-selective phase coding in the superior temporal sulcus. *PNAS*, 109(47):19438–19443. 22, 36

van Rijsbergen, C. J. (1974). Foundation of evaluation. *Journal of Documentation*, 30(4):365–373. 133

van Rossum, G. and Drake, F. L. (2001). Python reference manual. Pythonlabs, Virginia, USA, 2001. Available at `http://www.python.org`. 103

Vilares, I. and Kording, K. (2011). Bayesian models: the structure of the world, uncertainty, behavior, and the brain. *Annals of the New York Academy of Sciences*, 1224(1):22–39. 7

Vul, E., Goodman, N. D., Griffiths, T. L., and Tenenbaum, J. B. (2009). One and done? optimal decisions from very few samples. In *Proceedings of the 31st Annual Conference of the Cognitive Science Society*, volume 1, pages 66–72. 1, 31

Vul, E. and Pashler, H. (2008). Measuring the crowd within: probabilistic representations within individuals. *Psychological Science*, 19(7):645–647. 34

Wainwright, M. J. and Jordan, M. I. (2008). Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1–2):1–305. 14, 16

Wang, X. J. (2010). Neurophysiological and computational principles of cortical rhythms in cognition. *Physiological Reviews*, 90(3):1195–1268. 21

Whitley, D. (1994). A genetic algorithm tutorial. *Statistics and computing*, 4(2):65–85. 79

Xu, S., Jiang, W., Poo, M., and Dan, Y. (2012). Activity recall in a visual cortical ensemble. *Nature Neuroscience*, 15(3):449–456. 35

Yamanaka, K., Agu, M., and Miyajima, T. (1997). A continuous-time asynchronous boltzmann machine. *Neural networks*, 10(6):1103–1107. 121

Yarom, Y. and Hounsgaard, J. (2011). Voltage fluctuations in neurons: signal or noise? *Physiol Rev*, 91(3):917–929. 6

Zhang, Q. F., Wen, Y., Zhang, D., She, L., Wu, J. Y., Dan, Y., and Poo, M. M. (2012). Priming with real motion biases visual cortical response to bistable apparent motion. *PNAS*, 109(50):20691–20696. 35, 36

Zhang, Y., Meyers, E. M., Bichot, N. P., Serre, T., Poggio, T. A., and Desimone, R. (2011). Object decoding with attention in inferior temporal cortex. *Proceedings of the National Academy of Sciences*, 108(21):8850–8855. 15