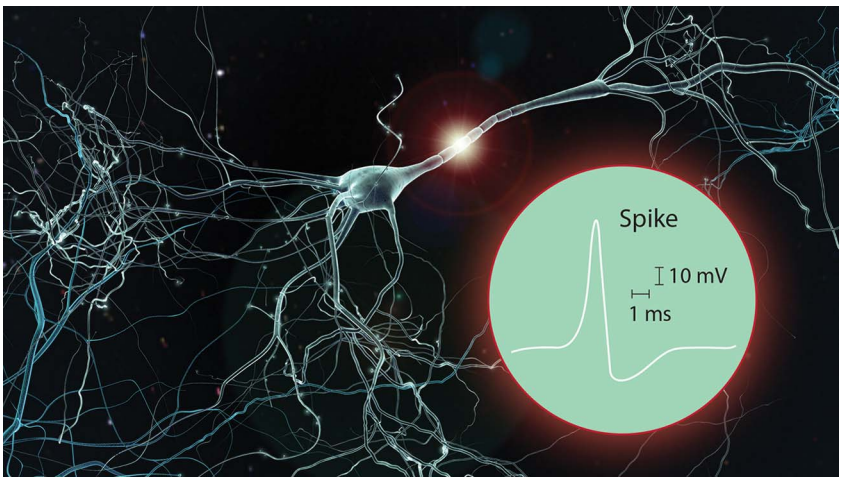
To Spike or Not to Spike: That Is the Question

By WOLFGANG MAASS
Institute for Theoretical Computer Science, Graz University of Technology, Graz 8010, Austria



I. INTRODUCTION

Both the brain and digital computers process information, but they do this in completely different ways. Neurons in the brain transmit information not through bits, but through spikes. Spikes are short voltage increases [see inset in the figure above] that are generated near the cell body of a neuron, with average spike rates below 10 Hz. These spikes are transmitted via fine axonal fibers and synapses to about 10 000 other neurons. Neurons also differ in another fundamental aspect from processors in a digital computer: they produce spikes according to stochastic rather than deterministic rules. This article discusses recent progress in understanding how complex computations can be carried out with such stochastically spiking neurons. Other recent developments suggest that spike-based neural networks can be emulated by neuromorphic hardware at a fraction of the energy consumed by current digital computing hardware [11]. Can both developments be merged to provide a blueprint for substantially more energy-efficient computing devices?

II. HOW CAN ONE COMPUTE WITH STOCHASTICALLY SPIKING NEURONS?

The human brain employs about 10^{11} spiking neurons, a number which is in the same range as the number of transistors in a modern supercomputer. But whereas the power consumption of supercomputers lies in the range of megawatts (for example, the K computer consumes almost as much energy as 10 000 suburban homes), the brain only consumes 30 W. Hence the brain provides an intriguing paradigm for energy-efficient computing. But in order to benefit from this paradigm, we need to understand how spike-based computations are organized in the brain.

A deterministic network of spiking neurons can be theoretically as powerful as a universal Turing machine [9]: It can employ the lengths of interspike intervals to encode and transmit information. But biological data suggest that brain computations are nondeterministic: If one tries to induce the brain to carry out the same computation again and again, the spiking activity varies substantially from trial to trial, as shown, for example, in Fig. 1(b). Stochastic signal transmission at synapses and

Digital Object Identifier: 10.1109/JPROC.2015.2496679

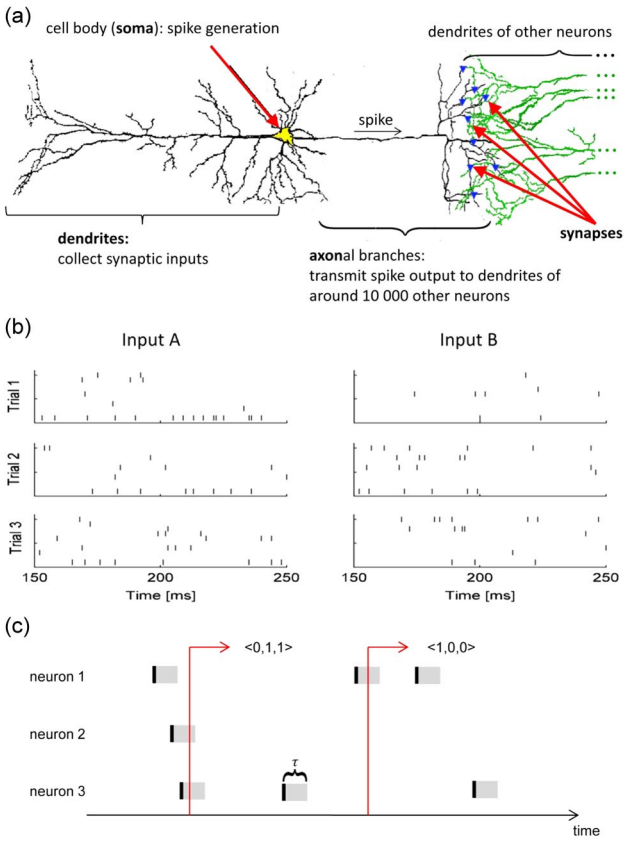


Fig. 1. *Spiking activity of biological neurons. (a) Neurons collect input from other neurons in their dendrites, generate spikes near their soma when the neuron "fires," and transmit these spikes via axonal branches and synapses of other neurons. Inset shows typical shape of a spike. (b) Generic trial-to-trial variability of spike responses in the visual cortex, on the left for three presentations of visual stimulus A, and on the right for three presentations of visual stimulus B (see [13] for details). Each row shows for one of the neurons the times (marked by vertical bars) when it fires. (c) The principle for encoding the state of a network of spiking neurons at any time t as a bit vector that records which neuron has fired within the preceding time window of length τ. The parameter τ is a time constant that is often chosen to be in the same range as the standard length of a postsynaptic potential (e.g., 10 ms).*

stochastic local amplifiers (voltage-dependent channels) in the dendrites of neurons are assumed to be the primary sources of this trial-to-trial variability of neural responses. Obviously, a digital computer would be judged as useless if it would exhibit such intrinsic variability. Hence, this trial-to-trial variability is a sign that brain computations with spiking neurons are organized differently than computations in our current generation of digital computers.

Recent experimental [3] and theoretical work [5] suggests to view networks of neurons in the brain as Markov chains, rather than Turing machines. Markov chains are finite state machines that change their state according to stochastic rules. For example, if we view a network of N spiking neurons as a Markov chain, the state of this Markov chain at time t can be defined as a binary vector of length N that has a 1 for each neuron that fires within some small time window of length τ around time t [see Fig. 1(c)]. But many other definitions of network states can also be used to model networks of stochastically spiking neurons as Markov chains with a unique stationary distribution p of network states [6]; see [10] for a review. This stationary distribution p over network states characterizes the long-term fraction of time that the Markov chain spends in each state (independently from its initial state). It provides the key for carrying out computations with a Markov chain, since its stationary distribution p can be viewed as an analog of a "program" for carrying out a particular type of computation. For example, a Markov chain can support probabilistic inference for p by generating samples from p, on which then some simple computational operations are performed. This approach is called Markov chain Monte Carlo (MCMC) sampling in machine learning. But Markov chains can also be used to generate heuristic solutions to difficult computational tasks. In particular, a realization of Markov chains through stochastic artificial neural networks, so-called Boltzmann machines, is frequently used in deep learning and for solving constraint satisfaction problems [1]. Boltzmann machines are networks of binary units ("artificial neurons") that are connected by synaptic connections with symmetric weights. Their units are allowed to change their state (one at a time) according to some global switching schedule. When a unit is allowed to change its state, it will assume state 1 with a probability that increases with the weighted sum of bits from other units to which it is synaptically connected. It will then stay in this new state until the next time when the global switching schedule allows it to change its state. One obvious difference between the commonly studied type of Boltzman machine and networks of neurons in the brain is that the latter are not subject to a global schedule that would allow them to spike only at specific times.

New theoretical results provide further insights into similarities and differences between Boltzmann machines and stochastically spiking neural networks, that I will sketch below. As concrete examples I will describe applications of spiking networks for

solving constraint satisfaction problems and probabilistic inference.

Box 1: A Simple Mathematical Model for a Network of Stochastically Spiking Neuron

A spike of neuron j causes—with some probability—at a synaptic connection to another neuron i a transient voltage change $w_{ij}\alpha_{ij}(t)$ (called postsynaptic potential) in a dendritic branch of neuron i, that is scaled by the strength or weight w_{ij} of this synaptic connection. The sign of the function $\alpha_{ij}(t)$ can be positive or negative, depending on the type (excitatory or inhibitory) of neuron j. It typically has a hill-like shape, whose exact form and duration depends on several factors, especially the location of the synapse on the dendritic tree of neuron i. The sum $u_i(t) = \sum_j w_{ij}\alpha_{ij}(t) + b_i$ of these postsynaptic potentials approximates in a simplified model the membrane potential of neuron i at time t, where b_i denotes the bias (excitability) of neuron i. An instantaneous firing probability of the form $\rho_i(t) = (1/a)\exp((u_i(t) - b)/c)$ works quite well for approximating—by fitting the parameters a, b, c—the experimentally observed stochastic spike generation of a biological neuron.

III. SOLVING CONSTRAINT SATISFACTION PROBLEMS

In order to illustrate how spiking networks can produce heuristic solutions even for difficult constraint satisfaction problems, I will sketch an application to the famous traveling salesman problem (TSP). An instance of the TSP consists in the simplest case of a set of N "cities" in a 2-D plane [see leftmost panel denoted "start" in Fig. 2(b)]. The task is to design a tour of minimal length that visits each city exactly once and returns to its origin. This task is inherently difficult; in fact, it is NP hard. In particular, it is not known how good local solutions could be efficiently expanded to provide good global solutions.

A neural network that produces approximate solutions for the TSP can be constructed as follows: One dedicates one winner-take-all (WTA) circuit or module X_s to each time step s of the tour. This WTA module consists of N neurons, one for each of the N cities that could possibly be visited at step s. Interconnections with strong negative weights between the N neurons in each WTA module ensure that usually at most one of its N neuron fires at any moment in time. The index i of the neuron in WTA module X_s that has most recently fired can be interpreted as its current proposition to visit city i at step s of the tour [using the same translation from spikes to bits as in Fig. 1(c)]. Thus, by recording for each WTA module which neuron has fired most recently, one can decode at any time t the firing activity of the whole network as a proposed TSP solution.

The black links between adjacent WTA modules in Fig. 2(a) (that select cities for adjacent steps of the tour) denote excitatory synaptic connections, whose strength (weight) encodes the distance matrix between the N cities. These weights are large for synaptic connections between neurons that encode cities i and j that have a small distance. This increases the probability that if one of them fires, the other one also fires, thereby proposing to go from city i directly to city j (or *vice versa*). Strongly negative weights on synaptic connections [red arcs in Fig. 2(a)] between neurons with the same index i in different WTA modules reduce the probability that a tour is proposed where city i is visited repeatedly.

Designing these weights in such a way that the best tours have the highest probability of being produced is nontrivial. But Buesing *et al.* [5] have provided for the case of networks of spiking neurons with symmetric weights a transparent relationship between the weights and biases of the neurons and the probability that a particular TSP tour, i.e., a particular network state $\langle x_1, \ldots, x_n \rangle$ [viewed as a bit vector like in Fig. 1(c)], is produced at an arbitrary time point t

$$p(\langle x_1, \ldots, x_m \rangle) = \frac{1}{z} e^{-\left(\sum_{i<j} w_{ij}x_i x_j - \sum_i b_i x_i\right)/T}.$$

This formula makes it easy to define weights w_{ij} and biases b_i in a spiking network in such a way that those TSP tours that satisfy all constraints and require the smallest total length have the lowest energy, and hence the highest probability under the stationary distribution of this Markov chain. The term $E(\langle x_1, \ldots, x_m \rangle) = \sum_{i<j} w_{ij}x_i x_j - \sum_i b_i x_i$ in the exponent is commonly referred to as the "energy" of this state. This terminology is motivated by the tendency of the network to move to states where this term is smaller (but it is not related to the physical energy consumed by an implementation of the network).

This probability $p(\langle x_1, \ldots, x_m \rangle)$ of a network state $\langle x_1, \ldots, x_m \rangle$ is the same as for a commonly used artificial (i.e., nonspiking) stochastic neural network, the Boltzmann machine. Their close relation to spiking networks is somewhat surprising since they are—in contrast to spiking networks—reversible Markov chains: Transitions between pairs of network states occur with equal probability in either direction. In contrast, spiking networks are nonreversible because a spike causes subsequent changes in the postsynaptic potential for some duration τ in other neurons, and this causal chain is nonreversable. But on the basis of our new theoretical understanding of stochastically spiking neurons one can now design a network of spiking neurons for solving a given constraint satisfaction problem through stochastic approximation as easily as a Boltzmann machine [1].

One would think that spiking neural networks and Boltzmann machines that have the same stationary distribution $p(\langle x_1, \ldots, x_m \rangle)$ solve

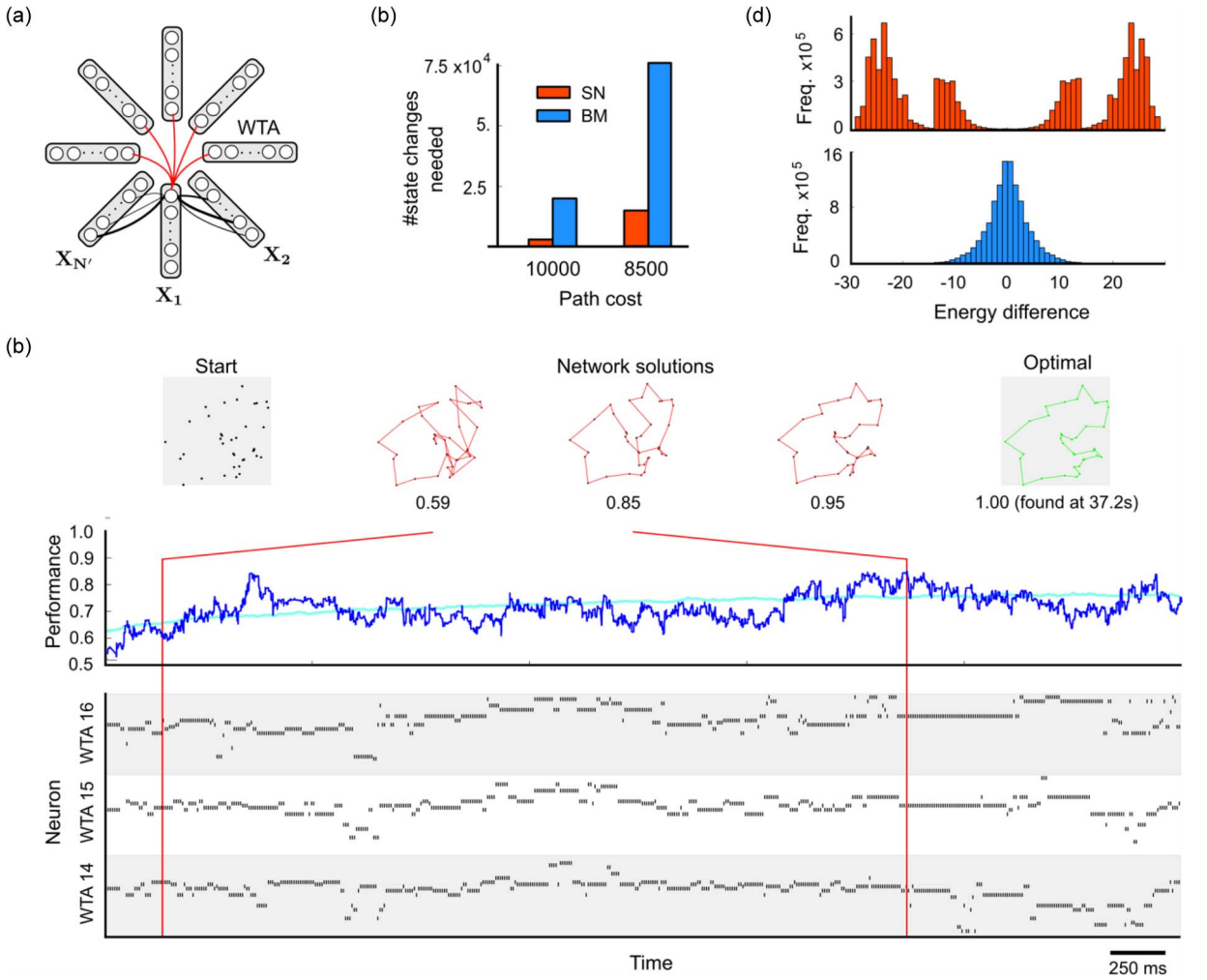


Fig. 2. *(a) Composition of the spiking network from WTA modules (gray oval boxes). (b) Top row: TSP instance on the far left and an optimal TSP tour on the far right. TSP tours shown in between result from interpreting network states [that are defined as in Fig. 1(c)] at some arbitrarily chosen time points t as proposed TSP tours. Middle row: Temporal evolution of the quality of TSP tours S generated by the network [performance measured by the quotient (optimal tour length)/(length of tour S)]. Dark blue: for a single trial run. Cyan: average over 100 trial runs. Bottom row: Spiking activity of neurons in three sample WTA modules of the network, where, e.g., WTA module 16 proposes which cities to visit at step 16 of the tour. (c) Comparison of the number of state changes needed until tours are found with path cost at most 10 000 (8500) by a spiking network (SN) and a Boltzmann machine (BM). Both are Markov chains that have exactly the same stationary distribution. (d) (top) Histograms of energy differences for state changes in the spiking network and (bottom) Boltzmann machine. These are markedly different, in spite of the fact that both Markov chains have the same stationary distribution, and can therefore solve the same constraint satisfaction problem. The spike-based implementation of the network is of additional interest because it is amenable to more energy-efficient hardware implementations.*

constraint satisfaction problems equally fast. But Fig. 2(c) and (d) shows that their stochastic search strategies differ. Since they have the same stationary distribution, both networks have to spend in the long run the same fraction of time in network states with high probability (that encode in this example good TSP solutions). But whereas a Boltzmann machine tends to stay on each visit for a longer duration in such a state, the spiking network visits such a state more often, each time for a shorter period. The source of this difference lies in the transient nature of state changes that are caused by a spike: A spike of neuron i sets a corresponding component x_i of the state vector for a duration τ to "1" [see Fig. 1(c)]. After that, the value of x_i is automatically set back to 0, no matter how large the energy difference is that is caused by this second switch [this explains the large fraction of state changes that cross large energy barriers in the upper panel of Fig. 2(d); the four distinct modes arise from the particular distribution of energies in this stationary distribution]. In contrast, a Boltzmann machine lets every state change depend on the energy difference between the two states. Therefore, it avoids switches that would increase the energy of the network state [lower panel of Fig. 2(d)]. Therefore, a Boltzmann machine crosses energy barriers (that typically separate a suboptimal TSP solution from a substantially better TSP

solution) more rarely. This feature explains why a spiking network needs substantially fewer state changes for finding a TSP solution of a given maximal cost [Fig. 2(c)]; see [8] for details.

With regard to the consumption of physical energy, these results suggest that the TSP can be solved (at least heuristically) in a very efficient manner by spike-based hardware, hence at a fraction of the energy consumption consumed by traditional digital hardware [11]. An additional energy savings may result from the faster stochastic search, compared with Boltzmann machines, that emerges in a spike-based approach.

Very recent design ideas for spike-based neuromorphic hardware that is able to solve constraint satisfaction problems can be found in [19] and [20].

IV. PROBABILISTIC INFERENCE

Another application range of spike-based stochastic networks is probabilistic inference through MCMC sampling. One creates here a network of spiking neurons whose stationary distribution of network states is a given distribution $p(z_1, \ldots, z_K)$ over many binary random variables z_k for which one wants to carry out probabilistic inference, e.g., estimate a posterior marginal

$$p(z_1|\mathbf{e}) = \sum_{v_2,\ldots,v_m} p(z_1, v_2, \ldots, v_m|\mathbf{e}). \quad (1)$$

The vector $\mathbf{e}$ represents here concrete values ("evidence") that are plugged in for the variables $z_{m+1}, \ldots, z_K$. The sum ranges over all possible values v_j of those random variables z_j that are not relevant for this inference, and are therefore marginalized out. Such probabilistic inference is in general NP-hard, but if p is realized ("embodied") as stationary distribution of a network of spiking neurons, one can estimate the value of (1) by observing the firing rate of the neuron that represents the variable z_1 (while forcing the neurons that represent the variables $z_{m+1}, \ldots, z_K$ to fire at a high rate or not to fire, in dependence of the evidence $\mathbf{e}$). This method is quite general, since theoretical results imply that any given distribution $p(z_1, \ldots, z_K)$ over discrete random variables can be embodied by some network of spiking neurons [14]. In fact, networks of spiking neurons can even learn through synaptic plasticity to approximate any such distribution p from examples that are produced by the target distribution p [15]. An interesting open question is whether the biologically inspired spike-based synaptic plasticity rule that is used here [it is called spike-timing-dependent plasticity (STDP)] supports further advantages of spiking networks. In contrast to covariance-based rules for plasticity in artificial neural networks such as the famous Hebb rule, STDP tracks and strengthens causal chains of spiking events, thereby emphasizing the nonreversible aspects of the underlying Markov chain of a spiking network.

As in the case of solving constraint satisfaction problems, an implementation of probabilistic inference in spike-based hardware could substantially reduce the energy that it consumes.

V. DEEP LEARNING

Another interesting application of STDP has recently been found in the context of spike-based implementations of deep learning [12], [16]. A large class of deep learning applications, especially those that require experience-based clever completion of partial input patterns, rather than just pattern classification, employ Boltzmann machines. The previously sketched link between Boltzmann machines and spiking neural networks has opened the door to potentially more energy-efficient hardware implementations of deep learning through spiking neurons. The work of [12] shows that STDP works very well in the resulting spike-based deep learning implementation.

These and other new results suggest that deep learning is moving closer to capturing the way how networks of neurons in the brain learn and represent complex patterns. But one fundamental difference becomes obvious when one compares the architecture of a typical deep learning network with that of neural networks in the brain. Both types of networks are hierarchical, where more abstract features are represented (and learned) on higher levels of the hierarchy. But whereas lateral connections between neurons on the same level of the hierarchy are forbidden in most types of deep learning models (because they disturb currently applied learning algorithms), biological networks of neurons have a very large number of synaptic connections within each level of a hierarchy of brain areas, say in the visual cortex. The previously sketched paradigm for solving constraint satisfaction problems in recurrent networks of spiking neurons suggests one possible use of these lateral connections in the brain: They can constrain resulting patterns of input representations on each level of the hierarchy in the light of hardwired ("innate") or learned prior knowledge. The use of such prior knowledge is likely to reduce the number of examples that are needed for learning.

VI. NEW HARDWARE IMPLEMENTATIONS OF STOCHASTICALLY SPIKING NEURAL NETWORKS

Numerous developments in the emergent area of neurally inspired stochastic electronics are reviewed in [7]. A key question is, of course, how the required stochasticity can be generated efficiently in dedicated hardware. A natural idea is to exploit and modify intrinsic noise at the microscale. Among other currently considered approaches are ring oscillators [18], chaotic neural networks [17], and stochastic memristors [2]. The primary source of stochasticity in networks

of neurons in the brain is closest to the approach via stochastic memristors: Biological synapses are stochastic, with random transmission failures ranging from less than 10% to more than 90%, depending on the brain area and species [4]. In addition, biological synapses trigger postsynaptic potentials also spontaneously, without a presynaptic spike.

VII. TO SPIKE OR NOT TO SPIKE

I have sketched new paradigms for spike-based computation that employ a stochastic neuron model: in order to decide whether to spike, the neuron tosses a coin whose bias depends on the current input to the neuron. Theoretical insight and clever hardware implementations are showing promise for energy-efficient solutions of important computing and learning tasks by networks of such neurons. But the question whether this approach will provide the key for substantially more energy-efficient computing hardware is still open at present. ■

Acknowledgment

The author would like to thank G. Cauwenberghs, V. Damle, and K. Meier for helpful comments.

REFERENCES

[1] E. Aarts and J. Korst, *Simulated Annealing and Boltzmann Machines: A Stochastic Approach to Combinatorial Optimizationand Neural Computing*. New York, NY, USA: Wiley, 1989.

[2] M. Al-Shedivat, R. Naous, E. Neftci, G. Cauwenberghs, and K. Salama, "Memristors empower spiking neurons with stochasticity," *IEEE J. Emerging Sel. Top. Circuits Syst.*, vol. 5, no. 2, pp. 242–253, Jun. 2015.

[3] P. Berkes, G. Orban, M. Lengyel, and J. Fiser, "Spontaneous cortical activity reveals hallmarks of an optimal internal model of the environment," *Science*, vol. 331, pp. 83–87, 2011.

[4] T. Branco and K. Staras, "The probability of neurotransmitter release: Variability and feedback control at single synapes," *Nature Rev. Neurosci.*, vol. 10, no. 5, pp. 373–383, 2009.

[5] L. Buesing, J. Bill, B. Nessler, and W. Maass, "Neural dynamics as sampling: A model for stochastic computation in recurrent networks of spiking neurons," *PLoS Comput. Biol.*, vol. 7, no. 11, 2011, Art. ID e1002211.

[6] S. Habenschuss, Z. Jonke, and W. Maass, "Stochastic computations in cortical microcircuit models," *PLoS Comput. Biol.*, vol. 9, no. 11, 2013, Art. ID e1003311.

[7] T. J. Hamilton, S. Afshar, A. van Schaik, and J. Tapson, "Stochastic electronics: A neuro-inspired design paradigm for integrated circuits," *Proc. IEEE*, vol. 102, no. 5, pp. 843–859, May 2014.

[8] Z. Jonke, S. Habenschuss, and W. Maass, "A theoretical basis for efficient computations with noisy spiking neurons," 2014. [Online]. Available: http://arxiv.org/abs/1412.5862

[9] W. Maass, "Lower bounds for the computational power of networks of spiking neurons," *Neural Comput.*, vol. 8, no. 1, pp. 1–40, 1996.

[10] W. Maass, "Noise as a resource for computation and learning in networks of spiking neurons," *Proc. IEEE*, vol. 102, Special Issue on Engineering Intelligent Electronic Systems Based on Computational Neuroscience, no. 5, pp. 860–880, May 2014.

[11] P. A. Merolla *et al.* "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, pp. 668–673, 2014.

[12] E. Neftci, S. Das, B. Pedroni, K. Kreutz-Delgado, and G. Cauwenberghs, "Event-driven contrastive divergence for spiking neuromorphic systems," *Front. Neurosci.*, vol. 72013, DOI: 10.3389/fnins.2013.00272.

[13] D. Nikolic, S. Haeusler, W. Singer, and W. Maass, "Distributed fading memory for stimulus properties in the primary visual cortex," *PLoS Biol.*, vol. 7, no. 12, pp. 1–19, 2009.

[14] D. Pecevski, L. Buesing, and W. Maass, "Probabilistic inference in general graphical models through sampling in stochastic networks of spiking neurons," *PLoS Comput. Biol*, vol. 7, no. 12, 2011 Art. ID e1002294.

[15] D. Pecevski and W. Maass, "Learning probabilistic inference through STDP," 2015, under review.

[16] M. A. Petrovici, J. Bill, I. Bytschok, J. Schemmel, and K. Meier, "Stochastic inference with deterministic spiking neurons," 2013. [Online]. Available: http://arxiv.org/abs/1311.3211

[17] T. Pfeil *et al.* "The effect of heterogeneity on decorrelation mechanisms in spiking neural networks: A neuromorphic-hardware study," 2014. [Online]. Available: http://arxiv.org/abs/1411.7916

[18] K. Yang *et al.* "A 23 mb/s 23 pj/b fully synthesized true-random-number generator in 28 nm and 65 nm CMOS," in *Dig. Tech. Papers Solid-IEEE Int. State Circuits Conf.* 2014, pp. 280–281.

[19] J. Binas, G. Indiveri, and M. Pfeiffer, "Spiking analog VLSI neuron assemblies as constraint satisfaction problem solvers," 2015, arXiv preprint arXiv:1511.00540.

[20] H. Mostafa, L. K. Muller, and G. Indiveri, "An event-based architecture for solving constraint satisfaction problems," *Nature Commun.*, 2015, to be published.