Computing with Spiking Neuron Networks

Hélène Paugam-Moisy[1] and Sander Bohte[2]

Abstract Spiking Neuron Networks (SNNs) are often referred to as the 3^{rd} generation of neural networks. Highly inspired from natural computing in the brain and recent advances in neurosciences, they derive their strength and interest from an accurate modeling of synaptic interactions between neurons, taking into account the time of spike firing. SNNs overcome the computational power of neural networks made of threshold or sigmoidal units. Based on dynamic event-driven processing, they open up new horizons for developing models with an exponential capacity of memorizing and a strong ability to fast adaptation. Today, the main challenge is to discover efficient learning rules that might take advantage of the specific features of SNNs while keeping the nice properties (general-purpose, easy-to-use, available simulators, etc.) of traditional connectionist models. This chapter relates the history of the "spiking neuron" in Section 1 and summarizes the most currently-in-use models of neurons and synaptic plasticity in Section 2. The computational power of SNNs is addressed in Section 3 and the problem of learning in networks of spiking neurons is tackled in Section 4, with insights into the tracks currently explored for solving it. Finally, Section 5 discusses application domains, implementation issues and proposes several simulation frameworks.

[1] Professor at Universit de Lyon Laboratoire de Recherche en Informatique - INRIA - CNRS bat. 490, Universit Paris-Sud
Orsay cedex, France e-mail: hpaugam@lri.fr

[2] CWI
Amsterdam, The Netherlands e-mail: sbohte@cwi.nl

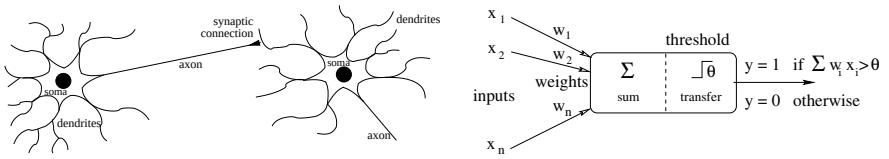
Contents

1 From natural computing to artificial neural networks

1.1 Traditional neural networks

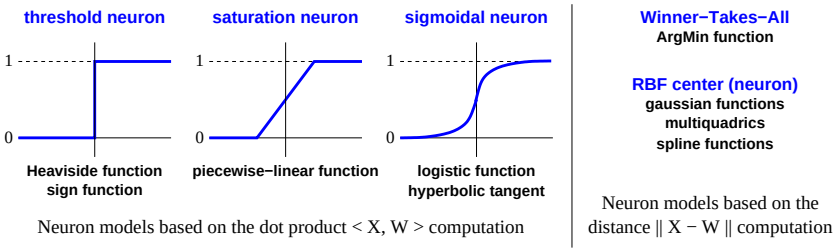
Since the human brain is made up of a great many of intricately connected neurons, its detailed workings are the subject of interest in fields as diverse as the study of neurophysiology, consciousness, and of course artificial intelligence. Less grand in scope, and more focused on the functional detail, artificial neural networks attempt to capture the essential computations that take place in these dense networks of interconnected neurons making up the central nervous systems in living creatures.

The original work of McCulloch & Pitts in 1943 [110] proposed a neural network model based on simplified "binary" neurons, where a single neuron implements a simple thresholding function: a neuron's state is either "active" or "not active", and at each neural computation step, this state is determined by calculating the weighted sum of the states of all the afferent neurons that connect to the neuron. For this purpose, connections between neurons are directed (*from* neuron N_i *to* neuron N_j), and have a weight (w_{ij}). If the weighted sum of the states of all the neurons N_i connected to a neuron N_j exceeds the characteristic threshold of N_j, the state of N_j is set to active, otherwise it is not (Figure 1, where index j has been omitted).



Elementary scheme of biological neurons — First mathematical model of artificial neuron

Fig. 1 The first model of neuron picked up the most significant features of a natural neuron: All-or-none output resulting from a non-linear transfer function applied to a weighted sum of inputs.



Neuron models based on the dot product < X, W > computation — Neuron models based on the distance || X − W || computation

Fig. 2 Several variants of neuron models, based on a dot product or a distance computation, with different transfer functions.

Subsequent neuronal models evolved where inputs and outputs were real-valued, and the non-linear threshold function (Perceptron) was replaced by a linear input-

output mapping (Adaline) or non-linear functions like the sigmoid (Multi-Layer Perceptron). Alternatively, several connectionist models (e.g. RBF networks, Kohonen self-organizing maps [84, 172]) make use of "distance neurons" where the neuron output results from applying a transfer function to the (usually quadratic) distance $\| X - W \|$ between the weights W and inputs X instead of the dot product, usually denoted by $< X, W >$ (Figure 2).

Remarkably, networks of such simple, connected computational elements can implement a wide range of mathematical functions relating input states to output states: With algorithms for setting the weights between neurons, these artificial neural networks can "learn" such relations.

A large number of learning rules have been proposed, both for teaching a network explicitly to do perform some task (supervised learning), and for learning interesting features "on its own" (unsupervised learning). Supervised learning algorithms are for example gradient descent algorithms (e.g. error backpropagation [140]) that fit the neural network behavior to some target function. Many ideas on local unsupervised learning in neural networks can be traced back to the original work on synaptic plasticity by Hebb in 1949 [51], and his famous, oft repeated quote:

> *When an axon of cell A is near enough to excite cell B or repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased.*

Unsupervised learning rules inspired from this type of natural neural processing are referred to as Hebbian rules (e.g. in Hopfield's network model [60]).

In general, artificial neural networks (NNs) have been proved to be very powerful, as engineering tools, in many domains (pattern recognition, control, bioinformatics, robotics), and also in many theoretical issues:

- Calculability: NNs computational power outperforms a Turing machine [154]
- Complexity: The "loading problem" is NP-complete [15, 78]
- Capacity: MLP, RBF and WNN[1] are universal approximators [35, 45, 63]
- Regularization theory [132]; PAC-learning[2] [171]; Statistical learning theory, VC-dimension, SVM[3] [174]

Nevertheless, traditional neural networks suffer from intrinsic limitations, mainly for processing large amount of data or for fast adaptation to a changing environment. Several characteristics, such as iterative learning algorithms or artificially designed neuron model and network architecture, are strongly restrictive compared with biological processing in natural neural networks.

---

[1] MLP = Multi-Layer Perceptrons - RBF = Radial Basis Function networks - WNN = Wavelet Neural Networks

[2] PAC learning = Probably Approximately Correct learning

[3] VC-dimension = Vapnik-Chervonenkis dimension - SVM = Support Vector Machines

1.2 The biological inspiration, revisited

A new investigation in natural neuronal processing is motivated by the evolution of thinking regarding the basic principles of brain processing. When the first neural networks were modeled, the prevailing belief was that intelligence is based on reasoning, and that logic is the foundation of reasoning. In 1943, McCulloch & Pitts designed their model of neuron in order to prove that the elementary components of the brain were able to compute elementary logic functions: Their first application of thresholded binary neurons was to build networks for computing boolean functions. In the tradition of Turing's work [168, 169], they thought that complex, "intelligent" behaviour could emerge from a very large network of neurons, combining huge numbers of elementary logic gates. History shows us that such basic ideas have been very productive, even if effective learning rules for large networks (e.g. backpropagation for MLP) have been discovered only at the end of the 1980's, and even if the idea of boolean decomposition of tasks has been abandoned for a long time.

Separately, neurobiological research has greatly progressed. Notions such as associative memory, learning, adaptation, attention and emotions have unseated the notion of logic and reasoning as being fundamental to understanding how the brain processes information, and *time* has become a central feature in cognitive processing [2]. Brain imaging and a host of new technologies (micro-electrode, LFP[4] or EEG[5] recordings, fMRI[6]) can now record rapid changes in the internal activity of brain, and help elucidate the relation between brain activity and the perception of a given stimulus. The current consensus agrees that cognitive processes are most likely based on the activation of transient assemblies of neurons (see Section 3.2), although the underlying mechanisms are not yet understood well.

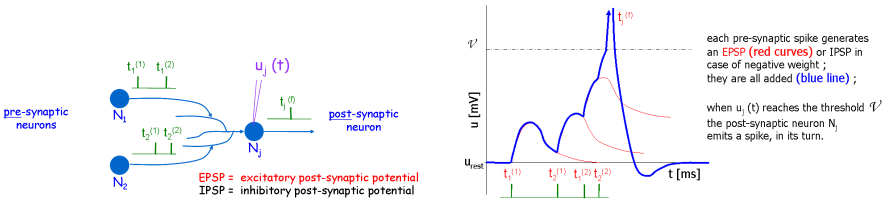


Fig. 3 A model of spiking neuron: N_j fires a spike whenever the weighted sum of incoming EPSPs generated by its pre-synaptic neurons reaches a given threshold. The graphic (right) shows how the membrane potential of N_j varies through time, under the action of the four incoming spikes (left).

With these advances in mind, it is worth recalling some neurobiological detail: real neurons spike, at least, most biological neurons rely on pulses as an important part of information transmission from one neuron to another neuron. In a rough and

[4] LFP = Local Field Potential

[5] EEG = ElectroEncephaloGram

[6] fMRI= functional Magnetic Resonance Imaging

non-exhaustive outline, a neuron can generate an action potential – the *spike* – at the soma, the cell body of the neuron. This brief electric pulse (1 or 2*ms* duration) then travels along the neuron's axon, that in turn is linked up to the receiving end of other neurons, the dendrites (see Figure 1, left view). At the end of the axon, synapses connect one neuron to another, and at the arrival of each individual spike, the synapses may release neurotransmitters along the *synaptic* cleft. These neurotransmitters are taken up by the neuron at the receiving end, and modify the state of that *postsynaptic* neuron, in particular the *membrane potential*, typically making the neuron more or less likely to fire for some duration of time.

The transient impact a spike has on the neuron's membrane potential is generally referred to as the *postsynaptic potential*, or *PSP*, and the PSP can either inhibit the future firing – inhibitory postsynaptic potential, *IPSP* – or excite the neuron, making it more likely to fire – an excitatory postsynaptic potential, *EPSP*. Depending on the neuron, and the specific type of connection, a PSP may directly influence the membrane potential for anywhere between tens of microseconds and hundreds of milliseconds. A brief sketch of the typical way a *spiking neuron* processes is depicted in Figure 3. It is important to note that the firing of a neuron may be a deterministic or stochastic function of its internal state.

Many biological details are omitted in this broad outline, and they may or may not be relevant for computing. Examples are the stochastic release of neurotransmitter at the synapses: depending on the firing history, a synaptic connection may be more or less reliable, and more or less effective. Inputs into different parts of the dendrite of a neuron may sum non-linearly, or even multiply. More detailed accounts can be found in for example [99].

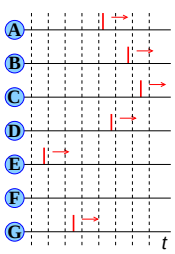
Evidence from the field of neuroscience has made it increasingly clear that in many situations, information is carried in the individual action potentials, rather than aggregate measures such as "firing rate". Rather than the form of the action potential, it is the number and the timing of spikes that matter. In fact, it has been established that *the exact timing of spikes* can be a means for coding information, for instance in the electrosensory system of electric fish [52], in the auditory system of echo-locating bats [86], and in the visual system of flies [14].

1.3 Time as basis of information coding

The relevance of the timing of individual spikes has been at the center of the debate about rate coding versus spike coding. Strong arguments against rate coding have been given by Thorpe et al. [165, 173] in the context of visual processing. Many physiologists subscribe to the idea of a Poisson-like rate code to describe the way that neurons transmit information. However, as pointed out by Thorpe et al., Poisson rate codes seem hard to reconcile with the impressively efficient rapid information transmission required for sensory processing in human vision. Only $100-150ms$ are sufficient for a human to respond selectively to complex visual stimuli (e.g. faces or food), but due to the feedforward architecture of visual system, made up of

multiple layers of neurons firing at an average rate of $10ms$, realistically only one spike or none could be fired by each neuron involved in the process during this time window. A pool of neurons firing spikes stochastically as a function of the stimulus could realize an instantaneous rate code: a *spike density code*. However, maintaining such a set of neurons is expensive, as is the energetic cost of firing so many spikes to encode a single variable [124]. It seems clear from this argument alone that the presence and possibly timing of individual spikes is likely to convey information, and not just the number, or rate, of spikes.

From a combinatorial point of view, precisely timed spikes have a far greater encoding capacity, given a small set of spiking neurons. The representational power of alternative coding schemes has been pointed out by Recce [134] and analysed by Thorpe et al. [164]. For instance, consider that a stimulus has been presented to a set of n spiking neurons and that each of them fires at most one spike in the next $T(ms)$ time window (Figure 4).



	count	latency	rank
A	1	5	3
B	1	6	5
C	1	7	6
D	1	5	4
E	1	1	1
F	0	–	–
G	1	3	2

Numeric examples:	count code	binary code	timing code	rank order
left (opposite) figure $n=7$, $T=7ms$	3	7	≈ 19	12.3
Thorpe et al. [164] $n=10$, $T=10ms$	3.6	10	≈ 33	21.8

Number of bits that can be transmitted by n neurons in a T time window.

Fig. 4 Comparing the representational power of spiking neurons, for different coding schemes. Count code: 6/7 spike per $7ms$, i.e. ≈ 122 spikes.s^{-1} - Binary code: 1111101 - Timing code: latency, here with a $1ms$ precision - Rank order code: $E \geq G \geq A \geq D \geq B \geq C \geq F$.

Consider some different ways to decode the temporal information that can be transmitted by the n spiking neurons. If the code is to *count* the overall number of spikes fired by the set of neurons (population rate coding), the maximum amount of available information is $\log_2(n+1)$, since only $n+1$ different events can occur. In the case of a *binary code*, the output is an n-digits binary number, with obviously n as information coding capacity. A higher amount of information is transmitted with a *timing code*, provided that an efficient decoding mechanism is available for determining the precise times of each spike. In practical cases, the available code size depends on the decoding precision, e.g. for a $1ms$ precision, an amount of information of $n \times \log_2(T)$ can be transmitted in the T time window. Finally, in *rank order coding*, information is encoded in the order of the sequence of spike emissions, i.e. one among the $n!$ orders that can be obtained from n neurons, thus $\log_2(n!)$ bits can be transmitted, meaning that the order of magnitude of capacity is $n\log(n)$. However this theoretical estimate must be alleviated when considering the unavoidable bound on precision required for distinguishing two spike times [177], even in computer simulation.

1.4 Spiking Neuron Networks

In *Spiking Neuron Networks* (SNNs)[7], the presence and *timing* of individual spikes is considered as the means of communication and neural computation. This compares with traditional neuron models where analog values are considered, representing the *rate* at which spikes are fired.

In SNNs, new input-output notions have to be developed that assign meaning to the presence and timing of spikes. One example of such coding that easily compares to traditional neural coding, is *temporal coding*[8]. Temporal coding is a straightforward method for translating a vector of real numbers into a spike train, for example for simulating traditional connectionist models by SNNs, as in [96]. The basic idea is biologically well-founded: the more intensive the input, the earlier the spike transmission (e.g. in visual system). Hence a network of spiking neurons can be designed with n input neurons N_i whose firing times are determined through some external mechanism. The network is fed by successive n-dimensional input analog patterns $\mathbf{x} = (x_1, \ldots, x_n)$ - with all x_i inside a bounded interval of $\mathbb{R}$, e.g. $[0,1]$ - that are translated into spike trains through successive temporal windows (comparable to successive steps of traditional NNs computation). In each time window, a pattern $\mathbf{x}$ is temporally coded relative to a fixed time T_{in} by one spike emission of neuron N_i at time $t_i = T_{in} - x_i$, for all i (Figure 5). It is straightforward to show that with such temporal coding, and some mild assumptions, any traditional neural network can be emulated by an SNN. However, temporal coding obviously does not apply readily to more continuous computing where neurons fire multiple spikes, in spike trains.

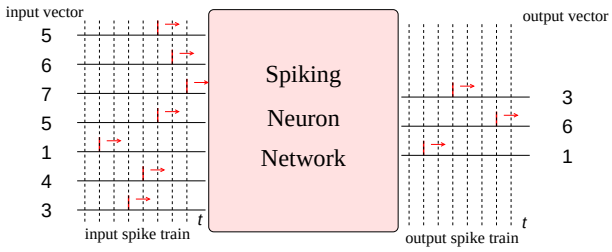


Fig. 5 Illustration of the temporal coding principle for encoding and decoding real vectors in spike trains.

Many SNN approaches focus on the continuous computation that is carried out on such spike trains. Assigning meaning is then less straightforward, and depending on the approach. However, a way to visualize the temporal computation processed by an SNN is by displaying a complete representation of the network activity on a *spike raster plot* (Figure 6): With time on the abscissa, a small bar is plotted each time a neuron fires a spike (one line per neuron, numbered in Y-axis). Variations

[7] SNNs are sometimes referred to as Pulsed-Coupled Neural Networks (PCNNs) in literature

[8] sometimes referred to as "latency coding" or "time-to-first-spike"

and frequencies of neuronal activity can be observed in such diagrams, in the same way as natural neurons activities can be observed in spike raster plots drawn from multi-electrode recordings. Likewise, other representations (e.g. time-frequency diagrams) can be drawn from simulations of artificial networks of spiking neurons, as is done in neuroscience from experimental data.

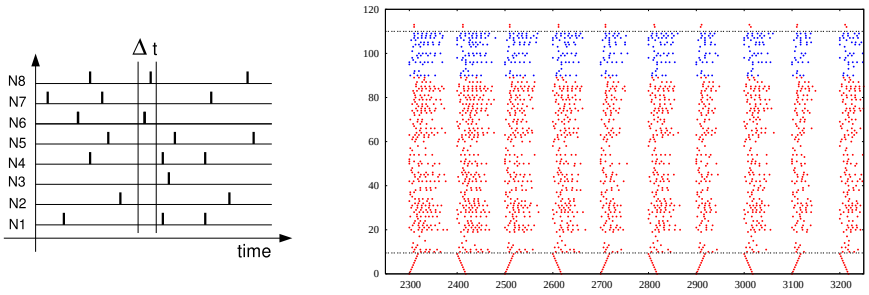


Fig. 6 On a spike raster plot, a small bar is plotted each time (in abscissa) that a neuron (numbered in ordinates) fires a spike. For computational purpose, time is often discretized in temporal Δt units (left). The dynamic answer of an SNN, stimulated by an input pattern in temporal coding - diagonal patterns, on bottom - can be observed on a spike raster plot (right) [from *Paugam-Moisy et al.* [127]].

Since the basic principle underlying SNNs is so radically different, it is not surprising that much of the work on traditional neural networks, such as learning rules and theoretical results, has to be adapted, or even has to be fundamentally rethought. The main purpose of this Chapter is to give an exposition on important state-of-the-art aspects of computing with SNNs, from theory to practice and implementation.

The first difficult task is to define "the" model of neuron, as there exist numerous variants already. Models of spiking neurons and synaptic plasticity are the subject of Section 2. It is worth mentioning that the question of network architecture has become less important in SNNs than in traditional neural networks. Section 3 proposes a survey of theoretical results (capacity, complexity, learnability) that argue for SNNs being a new generation of neural networks that are more powerful than the previous ones, and considers some of the ideas on how the increased complexity and dynamics could be exploited. Section 4 addresses different methods for learning in SNNs and presents the paradigm of *Reservoir Computing*. Finally, Section 5 focuses on practical issues concerning the implementation and use of SNNs for applications, in particular with respect to temporal pattern recognition.

2 Models of spiking neurons and synaptic plasticity

A spiking neuron model accounts for the impact of impinging action potentials – spikes – on the targeted neuron in terms of the internal state of the neuron, as well

as how this state relates to the spikes the neuron fires. There are many models of spiking neurons, and this section only describes some of the models that have so far been most influential in Spiking Neuron Networks.

2.1 Hodgkin-Huxley model

The fathers of the spiking neurons are the conductance-based neuron models, such as the well-known electrical model defined by Hodgkin & Huxley [57] in 1952 (Figure 7). Hodgkin & Huxley modeled the electro-chemical information transmission of natural neurons with electrical circuits consisting of capacitors and resistors: C is the capacitance of the membrane, g_{Na}, g_K and g_L denote the conductance parameters for the different ion channels (sodium Na, potassium K, etc.) and E_{Na}, E_K and E_L are the corresponding equilibrium potentials. The variables m, h and n describe the opening and closing of the voltage dependent channels.

$$C\frac{du}{dt} = -g_{Na}m^3h(u-E_{Na}) - g_Kn^4(u-E_K) - g_L(u-E_L) + I(t) \tag{1}$$

$$\tau_n\frac{dn}{dt} = -[n-n_0(u)], \quad \tau_m\frac{dm}{dt} = -[m-m_0(u)], \quad \tau_h\frac{dh}{dt} = -[h-h_0(u)]$$

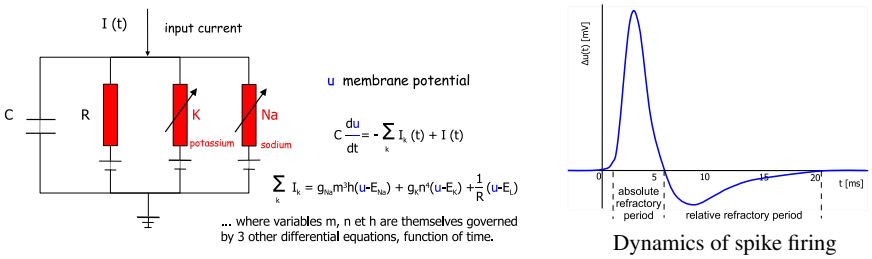


Fig. 7 Electrical model of "spiking" neuron as defined by Hodgkin and Huxley. The model is able to produce realistic variations of the membrane potential and the dynamics of a spike firing, e.g. in response to an input current $I(t)$ sent during a small time, at $t < 0$.

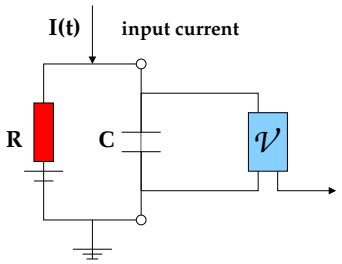
Appropriately calibrated, the Hodgkin-Huxley model has been successfully compared to numerous data from biological experiments on the giant axon of the squid. More generally, it has been shown that the Hodgkin-Huxley neuron is able to model biophysically meaningful properties of the membrane potential, respecting the behaviour recordable from natural neurons: an abrupt, large increase at firing time, followed by a short period where the neuron is unable to spike again, the *absolute refractoriness*, and a further time period where the membrane is depolarized, which makes renewed firing more difficult, i.e. the *relative refractory period* (Figure 7).

The *Hodgkin-Huxley model* (HH) is realistic but far too much complex for the simulation of SNNs. Although ODE[9] solvers can be applied directly to the system of differential equations, it would be intractable to compute temporal interactions between neurons in a large network of Hodgkin-Huxley models.

2.2 Integrate-and-Fire model and variants

Integrate-and-Fire (I&F) and Leaky-Integrate-and-Fire (LIF)

Derived from the Hodgkin-Huxley neuron model are Integrate-and-Fire (I&F) neuron models, that are much more computationally tractable (see Figure 8 for equation and electrical model).



u being the membrane potential,

$$C\frac{du}{dt} = -\frac{1}{R}(u(t) - u_{rest}) + I(t)$$

spike firing time $t^{(f)}$ is defined by

$$u(t^{(f)}) = \vartheta \quad \text{with} \quad u'(t^{(f)}) > 0$$

Fig. 8 The Integrate-and-Fire model (I&F) is a simplification of the Hodgkin-Huxley model.

An important I&F neuron type is the *Leaky-Integrate-and-Fire* (LIF) neuron [87, 162]. Compared to the Hodgkin-Huxley model, the most important simplification in the LIF neuron implies that the shape of the action potentials is neglected and every spike is considered as a uniform event defined only by the time of its appearance. The electrical circuit equivalent for a LIF neuron consists of a capacitor C in parallel with a resistor R driven by an input current $I(t)$. In this model, the dynamics of the membrane potential in the LIF neuron are described by a single first-order linear differential equation:

$$\tau_m \frac{du}{dt} = u_{rest} - u(t) + RI(t), \tag{2}$$

where $\tau_m = RC$ is taken as the time constant of the neuron membrane, modeling the voltage leakage. Additionally, the firing time $t^{(f)}$ of the neuron is defined by a threshold crossing equation $u(t^{(f)}) = \vartheta$, under the condition $u'(t^{(f)}) > 0$. Immediately after $t^{(f)}$, the potential is reset to a given value u_{rest} (with $u_{rest} = 0$ as a common assumption). An absolute refractory period can be modeled by forcing the neuron

[9] ODE = Ordinary Differential Equations

to a value $u = -u_{abs}$ during a time d_{abs} after a spike emission, and then restarting the integration with initial value $u = u_{rest}$.

Quadratic-Integrate-and-Fire (QIF) and Theta neuron

Quadratic-Integrate-and-Fire (QIF) neurons, a variant where $\frac{du}{dt}$ depends on u^2, may be a somewhat better, and still computationally efficient, compromise. Compared to LIF neurons, QIF neurons exhibit many dynamic properties such as delayed spiking, bi-stable spiking modes, and activity dependent thresholding. They further exhibit a frequency response that better matches biological observations [25]. Via a simple transformation of the membrane potential u to a phase θ, the QIF neuron can be transformed to a Theta neuron model [42].

In the Theta neuron model, the neuron's state is determined by a *phase*, θ. The Theta neuron produces a spike with the phase passes through π. Being one-dimensional, the Theta neuron dynamics can be plotted simply on a phase circle (Figure 9).

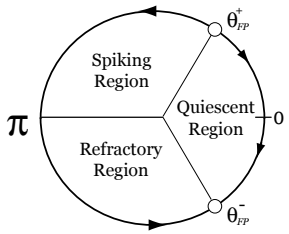


Fig. 9 Phase circle of the Theta neuron model, for the case where the baseline current $I(t) < 0$. When the phase goes through π, a spike is fired. The neuron has two fixed points: a saddle point θ_{FP}^{+}, and an attractor θ_{FP}^{-}. In the spiking region, the neuron will fire after some time, whereas in the quiescent region, the phase decays back to θ_{FP}^{-} unless input pushes the phase into the spiking region. The refractory phase follows after spiking, and in this phase it is more difficult for the neuron to fire again.

The phase-trajectory in a Theta-neuron evolves according to:

$$\frac{d\theta}{dt} = (1-\cos(\theta)) + \alpha I(t)(1+\cos(\theta)), \tag{3}$$

where θ is the neuron phase, α is a scaling constant, and $I(t)$ is the input current.

The main advantage of the Theta-neuron model is that neuronal spiking is described in a continuous manner, allowing for more advanced gradient approaches, as illustrated in Section 4.1.

Izhikevich's neuron model

In the class of spiking neurons defined by differential equations, the two-dimensional *Izhikevich neuron model* [66] is a good compromise between biophysical plausibility and computational cost. It is defined by the coupled equations

$$\frac{du}{dt} = 0.04u(t)^2 + 5u(t) + 140 - w(t) + I(t) \qquad \frac{dw}{dt} = a\,(bu(t) - w(t)) \tag{4}$$

$$\text{with after-spike resetting:} \qquad \text{if } u \geq \vartheta \text{ then } u \leftarrow c \text{ and } w \leftarrow w + d$$

This neuron model is capable to reproducing many different firing behaviors that can occur in biological spiking neurons (Figure 10)[10].

On spiking neuron model variants

Besides the models discussed here, there exist many different spiking neuron models that cover the complexity range between the Hodgkin-Huxley model and LIF models, with decreasing biophysical plausibility, but also with decreasing computational cost (see e.g. [67] for a comprehensive review, or [160] for an in-depth comparison of Hodgkin-Huxley and LIF subthreshold dynamics).

Whereas the Hodgkin-Huxley models are the most biologically realistic, the LIF and - to a lesser extend - QIF models have been studied extensively due to their low complexity, making them relatively easy to understand. However, as argued by Izhikevic [67], LIF neurons are a simplification that no longer exhibit many important spiking neuron properties. Where the full Hodgkin-Huxley model is able to reproduce many different neuro-computational properties and firing behaviors, the LIF model has been shown to only be able to reproduce 3 out of the 20 firing schemes displayed on Figure 10: the "tonic spiking" (A), the "class 1 excitable" (G) and the "integrator" (L). Note that although some behaviors are mutually exclusive for a particular instantiation of a spiking neuron model - e.g. (K) "resonator" and (L) "integrator" - many such behaviors may be reachable with different parameter choices, for a same neuron model. The QIF model is already able to capture more realistic behavior, and the Izhikevich neuron model can reproduce all of the 20 firing schemes displayed in Figure 10. Other intermediate models are currently being studied, such as the gIF model [138].

The complexity range can also be expressed in terms of the computational requirements for simulation. Since it is defined by four differential equations, the Hodgkin-Huxley model requires about 1200 floating point computations (FLOPS) per $1ms$ simulation. Simplified to two differential equations, the Morris-LeCar or FitzHugh-Nagumo models have still a computational cost of one to several hundreds FLOPS. Only 5 FLOPS are required by the LIF model, around 10 FLOPS for

[10] Electronic version of the original figure and reproduction permission are freely available at *www.izhikevich.com*

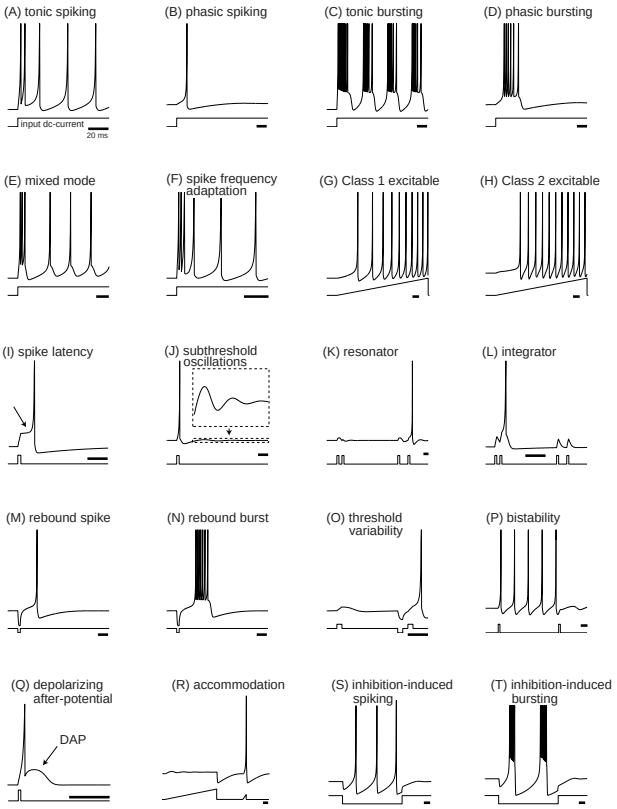


Fig. 10 Many firing behaviours can occur in biological spiking neurons. Shown are simulations of the Izhikevich neuron model, for different external input currents (displayed under each temporal firing pattern) [From *Izhikevich* [67]].

variants such as LIF-with-adaptation and quadratic or exponential Integrate-and-Fire neurons, and around 13 FLOPS for Izhikevich's model.

2.3 Spike Response Model

Compared to the neuron models governed by coupled differential equations, the *Spike Response Model* (SRM) as defined by Gerstner [46, 81] is more intuitive to understand and more straightforward to implement. The SRM model expresses the membrane potential u at time t as an integral over the past, including a model of refractoriness. The SRM is a phenomenological model of neuron, based on the occur-

rence of spike emissions. Let $\mathscr{F}_j = \{t_j^{(f)}; 1 \leq f \leq n\} = \{t \mid u_j(t) = \vartheta \wedge u_j'(t) > 0\}$ denote the set of all firing times of neuron N_j, and $\Gamma_j = \{i \mid N_i \text{ is presynaptic to } N_j\}$ define its set of presynaptic neurons. The state $u_j(t)$ of neuron N_j at time t is given by

$$u_j(t) = \sum_{t_j^{(f)} \in \mathscr{F}_j} \eta_j(t - t_j^{(f)}) + \sum_{i \in \Gamma_j} \sum_{t_i^{(f)} \in \mathscr{F}_i} w_{ij} \varepsilon_{ij}(t - t_i^{(f)}) + \underbrace{\int_0^\infty \kappa_j(r) I(t-r) dr}_{\text{if external input current}} \quad (5)$$

with the following kernel functions: η_j is non-positive for $s > 0$ and models the potential reset after a spike emission, ε_{ij} describes the membrane potential's response to presynaptic spikes, and κ_j describes the response of the membrane potential to an external input current. Some common choices for the kernel functions are:

$$\eta_j(s) = -\vartheta \exp\left(-\frac{s}{\tau}\right) \mathscr{H}(s),$$

or, somewhat more involved,

$$\eta_j(s) = -\eta_0 \exp\left(-\frac{s - \delta^{abs}}{\tau}\right) \mathscr{H}(s - \delta^{abs}) - K \mathscr{H}(s) \mathscr{H}(\delta^{abs} - s),$$

where $\mathscr{H}$ is the Heaviside function, ϑ is the threshold and τ a time constant, for neuron N_j. Setting $K \to \infty$ ensures an absolute refractory period δ^{abs} and η_0 scales the amplitude of relative refractoriness.

Kernel ε_{ij} describes the generic response of neuron N_j to spikes coming from presynaptic neurons N_i, and is generally taken as a variant of an α-function[11]:

$$\varepsilon_{ij}(s) = \frac{s - d_{ij}^{ax}}{\tau_s} \exp\left(-\frac{s - d_{ij}^{ax}}{\tau_s}\right) \mathscr{H}(s - d_{ij}^{ax}),$$

or, in a more general description:

$$\varepsilon_{ij}(s) = \left[\exp\left(-\frac{s - d_{ij}^{ax}}{\tau_m}\right) - \exp\left(-\frac{s - d_{ij}^{ax}}{\tau_s}\right)\right] \mathscr{H}(s - d_{ij}^{ax}),$$

where τ_m and τ_s are time constants, and d_{ij}^{ax} describes the axonal transmission delay.

For the sake of simplicity, $\varepsilon_{ij}(s)$ can be assumed to have the same form $\varepsilon(s - d_{ij}^{ax})$ for any pair of neurons, only modulated in amplitude and sign by the weight w_{ij} (excitatory EPSP for $w_{ij} > 0$, inhibitory IPSP for $w_{ij} < 0$).

A short term memory variant of SRM results from assuming that only the last firing $\hat{t}_j$ of N_j contributes to refractoriness, $\eta_j(t - \hat{t}_j)$ replacing the sum in formula (5) by a single contribution. Moreover, integrating the equation on a small time window of $1ms$ and assuming that each presynaptic neuron fires at most once in the time window (reasonable since refractoriness of presynaptic neurons), reduces the SRM to the simplified SRM_0 model:

[11] An α-*function* is like $\alpha(x) = x\, exp^{-x}$

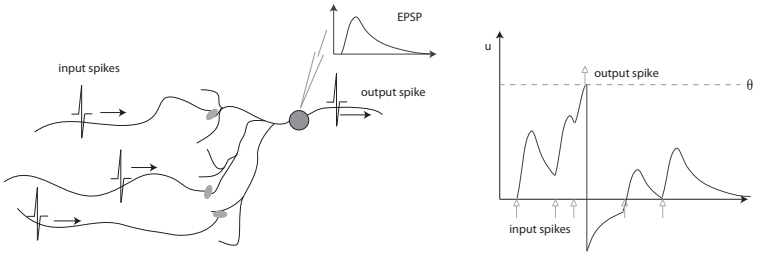


Fig. 11 The Spike Response Model (SRM) is a generic framework to describe the spike process (redrawn after [46]).

$$u_j(t) = \eta_j(t-\hat{t}_j) + \sum_{i \in \Gamma_j} w_{ij}\varepsilon(t-\hat{t}_i - d_{ij}^{ax}) \quad \text{next firing time} \quad t_j^{(f)} = t \Longleftrightarrow u_j(t) = \vartheta \tag{6}$$

Despite its simplicity, the Spike Response Model is more general than Integrate-and-Fire neuron models and is often able to compete with the Hodgkin-Huxley model for simulating complex neuro-computational properties.

2.4 Synaptic plasticity and STDP

In all the models of neurons, most of the parameters are constant values, and specific to each neuron. The exception are synaptic connections that are the basis of adaptation and learning, even in traditional neural network models where several synaptic weight updating rules are based on Hebb's law [51] (see Section 1). *Synaptic plasticity* refers to the adjustments and even formation or removal of synapses between neurons in the brain. In the biological context of natural neurons, the changes of synaptic weights with effects lasting several hours are referred as Long Term Potentiation (LTP) if the weight values (also called *efficacies*) are strengthened, and Long Term Depression (LTD) if the weight values are decreased. In the second or minute timescale, the weight changes are denoted as Short Term Potentiation (STP) and Short Term Depression (STD). In [1], Abbott & Nelson give a good review of the main synaptic plasticity mechanisms for regulating levels of activity in conjunction with Hebbian synaptic modification, e.g. redistribution of synaptic efficacy [107] or synaptic scaling. Neurobiological research has also increasingly demonstrated that synaptic plasticity in networks of spiking neurons is sensitive to the presence and precise timing of spikes [106, 12, 79].

One important finding that is receiving increasing attention is *Spike-Timing Dependent Plasticity*, STDP, as discovered in neuroscientific studies [106, 79], especially in detailed experiments performed by Bi & Poo [12, 13]. Often referred to as a *temporal Hebbian rule*, STDP is a form of synaptic plasticity sensitive to the

precise timing of spike firing relative to impinging presynaptic spike times. It relies on local information driven by backpropagation of action potential (BPAP) through the dendrites of the postsynaptic neuron. Although the type and amount of long-term synaptic modification induced by repeated pairing of pre- and postsynaptic action potential as a function of their relative timing vary from one neuroscience experiment to another, a basic computational principle has emerged: a maximal increase of synaptic weight occurs on a connection when the presynaptic neuron fires a short time before the postsynaptic neuron, whereas a late presynaptic spike (just after the postsynaptic firing) leads to decrease the weight. If the two spikes (pre- and post-) are too distant in time, the weight remains unchanged. This type of LTP / LTD timing dependency should reflect a form of causal relationship in information transmission through action potentials.

For computational purposes, STDP is most commonly modeled in SNNs using temporal windows for controlling the weight LTP and LTD that are derived from neurobiological experiments. Different shapes of STDP windows have been used in recent literature [106, 79, 158, 153, 26, 70, 80, 47, 123, 69, 143, 114, 117]: They are smooth versions of the shapes schematized by polygons in Figure 12. The spike timing (X-axis) is the difference $\Delta t = t_{post} - t_{pre}$ of firing times between the pre- and postsynaptic neurons. The synaptic change ΔW (Y-axis) operates on the weight update. For excitatory synapses, the weight w_{ij} is increased when the presynaptic spike is supposed to have a causal influence on the postsynaptic spike, i.e. when $\Delta t > 0$ and close to zero (windows 1-3 in Figure 12) and decreased otherwise. The main differences between shapes 1 to 3 concern the symmetry or asymmetry of the LTP and LTD subwindows, and the discontinuity or not of ΔW function of Δt, near $\Delta t = 0$. For inhibitory synaptic connections, it is common to use a standard Hebbian rule, just strengthening the weight when the pre- and postsynaptic spikes occur close in time, regardless of the sign of the difference $t_{post} - t_{pre}$ (window 4 in Figure 12).

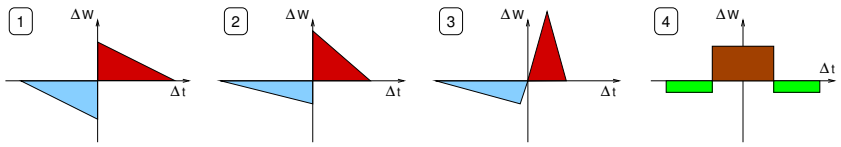


Fig. 12 Various shapes of STDP windows with LTP in blue and LTD in red for excitatory connections (1 to 3). More realistic and smooth ΔW function of Δt are mathematically described by sharp rising slope near $\Delta t = 0$ and fast exponential decrease (or increase) towards $\pm\infty$. Standard Hebbian rule (window 4) with brown LTP and green LTD are usually applied to inhibitory connections.

There exist at least two ways to compute with STDP: The modification ΔW can be applied to a weight w according to either an additive update rule $w \leftarrow w + \Delta W$ or a multiplicative update rule $w \leftarrow w(1 + \Delta W)$.

The notion of temporal Hebbian learning in the form of STDP appears as a possible new direction for investigating innovative learning rules in SNNs. However, many questions arise and many problems remain unresolved. For example, weight modifications according to STDP windows cannot be applied repeatedly in the same

direction (e.g. always potentiation) without fixing bounds for the weight values, e.g. an arbitrary fixed range $[0, w_{max}]$ for excitatory synapses. Bounding both the weight increase and decrease is necessary to avoid either silencing the overall network (when all weights down) or have "epileptic" network activity (all weights up, causing disordered and frequent firing of almost all neurons). However, in many STDP driven SNN models, a saturation of the weight values to 0 or w_{max} has been observed, which strongly reduces further adaptation of the network to new events. Among other solutions, a regulatory mechanism, based on a triplet of spikes, has been described by Nowotny et al. [123], for a smooth version of the temporal window 3 of Figure 12, with an additive STDP learning rule. On the other hand, applying a multiplicative weight update also effectively applies a self-regulatory mechanism. For deeper insights into the influence of the nature of update rule and the shape of STDP windows, the reader could refer to [158, 137, 28].

3 Computational power of neurons and networks

Since information processing in spiking neuron networks is based on the precise timing of spike emissions (pulse coding) rather than the average numbers of spikes in a given time window (rate coding), there are two straightforward advantages of SNN processing. First, SNN processing allows for the very fast decoding of sensory information, as in the human visual system [165], where real-time signal processing is paramount. Second, it allows for the possibility of multiplexing information, for example like the auditory system combines amplitude and frequency very efficiently over one channel. More abstractly, SNNs add a new dimension, the temporal axis, to the representation capacity and the processing abilities of neural networks. Here, we describe different approaches to determining the computational power and complexity of SNNs, and outline current thinking on how to exploit these properties, in particular in dynamic cell assemblies.

In 1997, Maass [97, 98] proposed to classify neural networks as follows:

- *1st generation:* Networks based on McCulloch and Pitts' neurons as computational units, *i.e.* threshold gates, with only digital outputs (e.g. perceptrons, Hopfield network, Boltzmann machine, multilayer networks with threshold units).
- *2nd generation:* Networks based on computational units that apply an activation function with a continuous set of possible output values, such as sigmoid or polynomial or exponential functions (e.g. MLP, RBF networks). The real-valued outputs of such networks can be interpreted as *firing rates* of natural neurons.
- *3rd generation of neural network models:* Networks which employ spiking neurons as computational units, taking into account the precise *firing times* of neurons for information coding. Related to SNNs are also pulse stream VLSI circuits, new types of electronic software that encode analog variables by time differences between pulses.

Exploiting the full capacity of this new generation of neural network models raises many fascinating and challenging questions that will be addressed in further sections.

3.1 Complexity and learnability results

Tractability

To facilitate the derivation of theoretical proofs on the complexity of computing with spiking neurons, Maass proposed a simplified spiking neuron model with a rectangular EPSP shape, the "type_A spiking neuron" (Figure 13). The type_A neuron model can for instance be justified as providing a link to silicon implementations of spiking neurons in analog VLSI neural microcircuits. Central to the complexity results is the notion of transmission delays: different transmission delays d_{ij} can be assigned to different presynaptic neurons N_i connected to a postsynaptic neuron N_j.

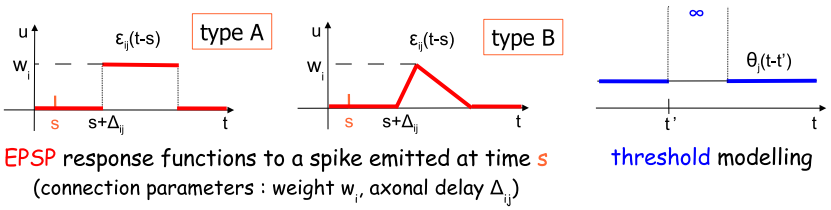


Fig. 13 Very simple versions of spiking neurons: "type_A spiking neuron" (rectangular shaped pulse) and "type_B spiking neuron" (triangular shaped pulse), with elementary representation of refractoriness (threshold goes to infinity), as defined in [97].

Let boolean input vectors $(x_1,\ldots,x_n)$ be presented to a spiking neuron by a set of input neurons $(N_1,\ldots,N_n)$ such that N_i fires at a specific time T_{in} if $x_i = 1$ and does not fire if $x_i = 0$. A type_A neuron is at least as powerful as a threshold gate [97, 145]. Since spiking neurons can behave as coincidence detectors[12] it is straightforward to prove that the boolean function CD_n *(Coincidence Detection function)* can be computed by a single spiking neuron of type_A (the proof relies on a suitable choice of the transmission delays d_{ij}):

$$CD_n(x_1,\ldots,x_n,y_1,\ldots,y_n) = \begin{cases} 1, & \text{if } (\exists i)\ x_i = y_i \\ 0, & \text{otherwise} \end{cases}$$

[12] For a proper choice of weights, a spiking neuron can only fire when two or more input spikes are effectively coincident in time.

In previous neural network generations, the computation of the boolean function CD_n required many more neurons: At least $\frac{n}{\log(n+1)}$ threshold gates and at least an order of magnitude of $\Omega(n^{1/4})$ sigmoidal units.

Of special interest is the *Element Distinctness function*, ED_n:

$$ED_n(x_1,\ldots,x_n) = \begin{cases} 1, & \text{if } (\exists i \neq j)\ x_i = x_j \\ 0, & \text{if } (\forall i \neq j)\ \mid x_i - x_j \mid \geq 1 \\ \text{arbitrary}, & \text{otherwise} \end{cases}$$

Let real-valued inputs $(x_1,\ldots,x_n)$ be presented to a spiking neuron by a set of input neurons $(N_1,\ldots,N_n)$ such that N_i fires at time $T_{in} - cx_i$ (cf. temporal coding, defined in Section 1.4). With positive real-valued inputs and a binary output, the ED_n function can be computed by a single type_A neuron, whereas at least $\Omega(n\log(n))$ threshold gates and at least $\frac{n-4}{2} - 1$ sigmoidal hidden units are required.

However, for arbitrary real-valued inputs, type_A neurons are no longer able to compute threshold circuits. For such settings, the "type_B spiking neuron" (Figure 13) has been proposed, as its triangular EPSP can shift the firing time of a targeted post-synaptic neuron in a continuous manner. It is easy to see that any threshold gate can be computed by $O(1)$ type_B spiking neurons. Furthermore, at the network level, any threshold circuit with s gates, for real-valued inputs $x_i \in [0,1]$ can be simulated by a network of $O(s)$ type_B spiking neurons.

From these results, Maass concludes that spiking neuron networks are computationally more powerful than both the 1*st* and the 2*nd* generations of neural networks.

Schmitt develops a deeper study of type_A neurons with programmable delays in [145, 102]. Some results are:

- Every boolean function of n variables, computable by a single spiking neuron, can be computed by a disjunction of at most $2n-1$ threshold gates.
- There is no $\Sigma\Pi$-unit with fixed degree that can simulate a spiking neuron.
- The *threshold number* of a spiking neuron with n inputs is $\Theta(n)$.
- The following relation holds: $(\forall n \geq 2)\ \exists$ a boolean function on n variables that has threshold number 2 and cannot be computed by a spiking neuron.
- The *threshold order* of a spiking neuron with n inputs is $\Omega(n^{1/3})$.
- The *threshold order* of a spiking neuron with $n \geq 2$ inputs is at most $n-1$.

Capacity

In [98], Maass considers *noisy spiking neurons*, a neuron model close to the SRM (cf. Section 2.3), with a probability of *spontaneous* firing (even under threshold) or not firing (even above threshold) governed by the difference:

$$\sum_{i\in\Gamma_j} \sum_{s\in\mathcal{F}_i, s<t} w_{ij}\varepsilon_{ij}(t-s) - \underbrace{\eta_j(t-t')}_{\text{threshold function}}$$

The main result from [98] is that for any given $\varepsilon, \delta > 0$ one can simulate any given feedforward sigmoidal neural network $\mathcal{N}$ of s units with linear saturated activation function by a network $\mathcal{N}_{\varepsilon,\delta}$ of $s + O(1)$ noisy spiking neurons, in temporal coding. An immediate consequence of this result is that SNNs are *universal approximators*, in the sense that any given continuous function $F : [0,1]^n \to [0,1]^k$ can be approximated within any given precision $\varepsilon > 0$ with arbitrarily high reliability, in temporal coding, by a network of noisy spiking neurons with a single hidden layer.

With regard to synaptic plasticity, Legenstein, Näger and Maass studied STDP learnability in [90]. They define a *Spiking Neuron Convergence Conjecture* (SNCC) and compare the behaviour of STDP learning by teacher-forcing with the Perceptron convergence theorem. They state that a spiking neuron can learn with STDP basically any map from input to output spike trains that it could possibly implement in a stable manner. They interpret the result as saying that STDP endows spiking neurons with *universal learning capabilities* for Poisson input spike trains.

Beyond these and other encouraging results, Maass [98] points out that SNNs are able to encode time series in spike trains, but there are, in computational complexity theory, no standard reference models yet for analyzing computations on time series.

VC-dimension

[13]

The first attempt to estimate the VC-dimension of spiking neurons is probably the work of Zador & Pearlmutter in 1996 [187], where they studied a family of integrate-and-fire neurons (cf. Section 2.2) with threshold and time-constants as parameters. Zador & Pearlmutter proved that for an Integrate-and-Fire (I&F) model, the $VC_{dim}(\text{I\&F})$ grows as $\log(B)$ with the input signal bandwidth B, which means that the VC_{dim} of a signal with infinite bandwidth is unbounded, but the divergence to infinity is weak (logarithmic).

More conventional approaches [102, 98] estimate bounds on the VC-dimension of neurons as functions of their programmable / learnable parameters, such as the synaptic weights, the transmission delays and the membrane threshold:

- With m variable positive delays, $VC_{dim}(\text{type\_A neuron})$ is $\Omega\left(m\log(m)\right)$ - even with fixed weights - whereas, with m variable weights, $VC_{dim}(\text{threshold gate})$ is $\Omega(m)$.
- With n real-valued inputs and a binary output, $VC_{dim}(\text{type\_A neuron})$ is $O\left(n\log(n)\right)$.
- With n real-valued inputs and a real-valued output, $pseudo_{dim}(\text{type\_A neuron})$ is $O(n\log(n))$.

The implication is that the learning complexity of a single spiking neuron is greater than the learning complexity of a single threshold gate. As Maass & Schmitt [103] argue, this should not be interpreted as saying that supervised learning is impossible for a spiking neuron, but rather that it is likely quite difficult to formulate rigorously provable learning results for spiking neurons.

---

[13] see *http://en.wikipedia.org/wiki/VC_dimension* for a definition

To summarize Maass and Schmitt's work: let the class of boolean functions, with n inputs and 1 output, that can be computed by a spiking neuron be denoted by $\mathscr{S}_n^{xy}$, where x is b for boolean values and a for analog (real) values and idem for y. Then the following holds:

- The classes $\mathscr{S}_n^{bb}$ and $\mathscr{S}_n^{ab}$ have VC-dimension $\Theta(n\ log(n))$.
- The class $\mathscr{S}_n^{aa}$ has pseudo-dimension $\Theta(n\ log(n))$.

At the network level, if the weights and thresholds are the only programmable parameters, then an SNN with temporal coding seems to be nearly equivalent to traditional Neural Networks (NNs) with the same architecture, for traditional computation. However, transmission delays are a new relevant component in spiking neural computation and SNNs with programmable delays appear to be more powerful than NNs.

Let $\mathscr{N}$ be an SNN of neurons with rectangular pulses (e.g. type_A), where all delays, weights and thresholds are programmable parameters, and let E be the number of edges of the $\mathscr{N}$ directed acyclic graph[14]. Then $VC_{dim}(\mathscr{N})$ is $O(E^2)$, even for analog coding of the inputs [103]. Schmitt derived more precise results by considering a feedforward architecture of depth D, with nonlinear synaptic interactions between neurons, in [146].

It follows that the sample sizes required for the networks of fixed depth are not significantly larger than for traditional neural networks. With regard to the generalization performance in pattern recognition applications, the models studied by Schmitt can be expected to be at least as good as traditional network models [146].

Loading problem

In the framework of PAC-learnability [171, 16], only hypotheses from $\mathscr{S}_n^{bb}$ may be used by the learner. Then, the computational complexity of training a spiking neuron can be analyzed within the formulation of the *consistency* or *loading problem* (cf. [78]):

> *Given a training set T of labeled binary examples (X,b) with n inputs, does there exist parameters defining a neuron $\mathscr{N}$ in $\mathscr{S}_n^{bb}$ such that $(\forall (X,b) \in T)\ y_{\mathscr{N}} = b$?*

In this PAC-learnability setting, the following results are proved in [103]:

- The *consistency problem* for a spiking neuron with binary delays is *NP*-complete ($d_{ij} \in \{0,1\}$).
- The *consistency problem* for a spiking neuron with binary delays and fixed weights is *NP*-complete.

Several extended results have been developed by Šíma and Sgall [155], such as:

[14] The directed acyclic graph is the network topology that underlies the spiking neuron network dynamics.

- The *consistency problem* for a spiking neuron with non-negative delays is *NP*-complete ($d_{ij} \in \mathbb{R}^+$). The result holds even with some restrictions (see [155] for precise conditions) on bounded delays, unit weights or fixed threshold.
- A single spiking neuron with programmable weights, delays and threshold does not allow robust learning unless $RP = NP$. The *approximation problem* is not better solved even if the same restrictions as above are applied.

Complexity results versus real-world performance

Non-learnability results such as those outlined above have of course been derived for classic NNs already, e.g. in [15, 78]. Moreover, the results presented in this section apply only to a restricted set of SNN models and, apart from the programmability of transmission delays of synaptic connections, they do not cover all the capabilities of SNNs that could result from computational units based on firing times. Such restrictions on SNNs can rather be explained by a lack of practice for building proofs in such a context or, even more, by an incomplete and unadapted computational complexity theory or learning theory. Indeed, learning in biological neural systems may employ rather different mechanisms and algorithms than common computational learning systems. Therefore, several characteristics, especially the features related to computing in continuously changing time, will have to be fundamentally rethought to develop efficient learning algorithms and ad-hoc theoretical models to understand and master the computational power of SNNs.

3.2 Cell assemblies and synchrony

One way to take a fresh look at SNNs complexity is to consider their dynamics, especially the spatial localization and the temporal variations of their activity. From this point of view, SNNs behave as *complex systems*, with emergent macroscopic-level properties resulting from the complex dynamic interactions between neurons, but hard to understand just looking at the microscopic-level of each neuron processing. As biological studies highlight the presence of a specific organization in the brain [159, 41, 3], the Complex Networks research area appears to provide with valuable tools ("Small-Word" connectivity [180], presence of clusters [121, 115], of hubs [7]... see [122] for a survey) for studying topological and dynamic complexity of SNNs, both in natural and artificial networks of spiking neurons. Another promising direction for research takes its inspiration from the area of Dynamic Systems: Several methods and measures, based on the notions of phase transition, edge-of-chaos, Lyapunov exponents or mean-field predictors, are currently proposed to estimate and control the computational performance of SNNs [89, 175, 147]. Although these directions of research are still in their infancy, an alternative is to revisit older and more biological notions that are already related to the network topology and dynamics.

The concept of the *cell assembly* has been introduced by Hebb [51] in 1949, more than half a century ago[15]. However the idea had not been further developed, neither by neurobiologists - since they could not record the activity of more than one or a few neurons at a time, until recently - nor by computer scientists. New techniques of brain imaging and recording have boosted this area of research in neuroscience for only a few years (cf. special issue 2003 of *Theory in Biosciences* [182]). In computer science, a theoretical analysis of assembly formation in spiking neuron network dynamics (with SRM neurons) has been discussed by Gerstner & van Hemmen in [48], where they contrast ensemble code, rate code and spike code, as descriptions of neuronal activity.

A cell assembly can be defined as a group of neurons with strong mutual excitatory connections. Since a cell assembly, once a subset of its neurons are stimulated, tends to be activated as a whole, it can be considered as an operational unit in the brain. An association can be viewed as the activation of an assembly by a stimulus or another assembly. Then, short term memory would be a persistent activity maintained by reverberations in assemblies, whereas long term memory would correspond to the formation of new assemblies, e.g. by a Hebb's rule mechanism. Inherited from Hebb, current thinking about cell assemblies is that they could play a role of "grandmother neural groups" as a basis of memory encoding, instead of the old controversial notion of "grandmother cell", and that material entities (e.g. a book, a cup, a dog) and, even more abstract entities such as concepts or ideas could be represented by cell assemblies.

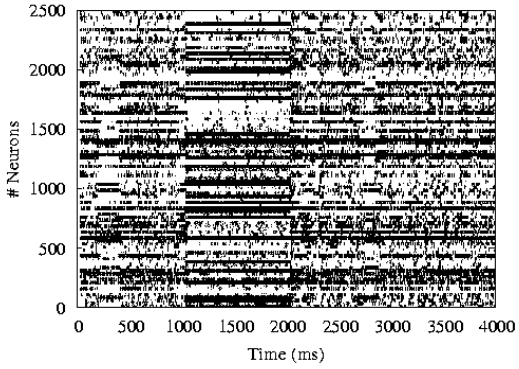


Fig. 14 A spike raster plot showing the dynamics of an artificial SNN: Erratic background activity is disrupted by a stimulus presented between 1000 and 2000 ms [From *Meunier* [112]].

Within this context, synchronization of firing times for subsets of neurons inside a network has received much attention. Abeles [2] developed the notion of *synfire chains*, which describes activity in a pool of neurons as a succession of synchronized firing by specific subsets of these neurons. Hopfield & Brody demonstrated

[15] The word "cell" was in used at that time, instead of "neuron".

transient synchrony as means for collective spatio-temporal integration in neuronal circuits [61, 62]. The authors claim that the event of collective synchronization of specific pools of neurons in response to a given stimulus may constitute a basic computational building block, at the network level, for which there is no resemblance in traditional neural computing.

However, synchronization per se – even transient synchrony – appears to be too restrictive a notion for fully understanding the potential capabilities of information processing in cell assemblies. This has been comprehensively pointed out by Izhikevich, who proposes the extended notion of *polychronization* [68] within a group of neurons that are sparsely connected with various axonal delays. Based on the connectivity between neurons, a polychronous group is a possible stereotypical time-locked firing pattern. Since the neurons in a polychronous group have matching axonal conduction delays, the group can be activated in response to a specific temporal pattern triggering very few neurons in the group, other ones being activated in a chain reaction. Since any given neuron can be activated within several polychronous groups, the number of coexisting polychronous groups can be far greater than the number of neurons in the network. Izhikevich argues that networks with delays are "infinite-dimensional" from a purely mathematical point of view, thus resulting in much greater information capacity as compared to synchrony based assembly coding. Polychronous groups represent good candidates for modeling multiple trace memory and they could be viewed as a computational implementation of cell assemblies.

Notions of cell assemblies and synchrony, derived from natural computing in the brain and biological observations, are inspiring and challenging computer scientists and theoretical researchers to search for and define new concepts and measures of complexity and learnability in dynamic systems. This will likely bring a much deeper understanding of neural computations that include the time dimension, and will likely benefit both computer science as well as neuroscience.

4 Learning in spiking neuron networks

Traditionally, neural networks have been applied to pattern recognition, in various guises. For example, carefully crafted layers of neurons can perform highly accurate handwritten character recognition [88]. Similarly, traditional neural networks are preferred tool for function approximation, or regression. The best-known learning rules for achieving such network are of course the class of error-backpropagation rules for supervised learning. There also exist learning rules for unsupervised learning, such as Hebbian learning, or distance based variants like Kohonen self-organizing maps.

Within the class of computationally oriented spiking neuron networks, we distinguish two main directions. First, there is the development of learning methods equivalent to those developed for traditional neural networks. By substituting traditional neurons with spiking neuron models, augmenting weights with delay lines,

and using temporal coding, algorithms for supervised and unsupervised learning have been developed. Second, there are networks and computational algorithms that are uniquely developed for networks of spiking neurons. These networks and algorithms use the temporal domain as well as the increased complexity of SNNs to arrive at novel methods for temporal pattern detection with spiking neuron networks.

4.1 Simulation of traditional models

Maass & Natschläger [100] propose a theoretical model for emulating arbitrary Hopfield networks in temporal coding (see Section 1.4). Maass [96] studies a "relatively realistic" mathematical model for biological neurons that can simulate arbitrary feedforward sigmoidal neural networks. Emphasis is put on the fast computation time that depends only on the number of layers of the sigmoidal network, and no longer on the number of neurons or weights. Within this framework, SNNs are validated as universal approximators (see Section 3.1), and traditional supervised and unsupervised learning rules can be applied for training the synaptic weights.

It is worth remarking that, to enable theoretical results, Maass & Natschläger's model uses static reference times T_{in} and T_{out} and auxiliary neurons. Even if such artifacts can be removed in practical computation, the method rather appears as an artificial attempt to make SNNs computing like traditional neural networks, without taking advantage of SNNs intrinsic abilities to computing with time.

Unsupervised learning in spiking neuron networks

Within this paradigm of computing in SNNs equivalently to traditional neural network computing, a number of approaches for unsupervised learning in spiking neuron networks have been developed, based mostly on variants of Hebbian learning. Extending on an Hopfield's idea [59], Natschläger & Ruf [119] propose a learning algorithm that performs unsupervised clustering in spiking neuron networks, akin to RBF network, using spike-times as input. Natschläger & Ruf's spiking neural network for unsupervised learning is a simple two-layer network of Spike Response Model neurons, with the addition of multiple delays between the neurons: An individual connection from a neuron i to a neuron j consists of a fixed number of m synaptic terminals, where each terminal serves as a sub-connection that is associated with a different delay d^k and weight w_{ij}^k (figure 15). The delay d^k of a synaptic terminal k is defined by the difference between the firing time of the pre-synaptic neuron i, and the time the post-synaptic potential of neuron j starts rising.

A Winner-Takes-All learning rule modifies the weights between the source neurons and the neuron first to fire in the target layer using a time-variant of Hebbian learning: If the start of the PSP at a synapse slightly precedes a spike in the target neuron, the weight of this synapse is increased, as it exerted significant influence on the spike-time via a relatively large contribution to the membrane potential. Earlier

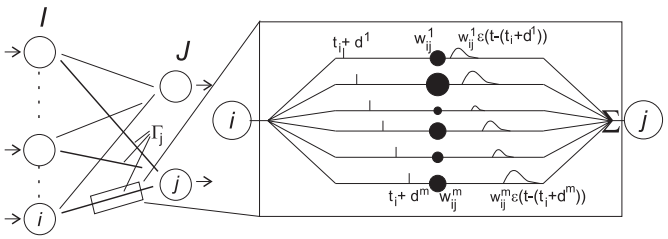


Fig. 15 Unsupervised learning rule in SNNs: Any single connection can be considered as being multisynaptic, with random weights and a set of increasing delays, as defined in [120].

and later synapses are decreased in weight, reflecting their lesser impact on the target neuron's spike time. With such a learning rule, input patterns can be encoded in the synaptic weights such that, after learning, the firing time of an output neuron reflects the distance of the evaluated pattern to its learned input pattern thus realizing a kind of RBF neuron [119].

Bohte *et al.,* [20] extend on this approach to enhance the precision, capacity and clustering capability of a network of spiking neurons by developing a temporal version of population coding. To extend the encoding precision and clustering capacity, input data is encoded into temporal spike-time patterns by population coding, using multiple local receptive fields like Radial Basis Functions. The translation of inputs into relative firing-times is straightforward: An optimally stimulated neuron fires at $t = 0$, whereas a value up to say $t = 9$ is assigned to less optimally stimulated neurons (depicted in Figure 16). With such encoding, spiking neural networks were shown to be effective for clustering tasks, e.g. Figure 17.

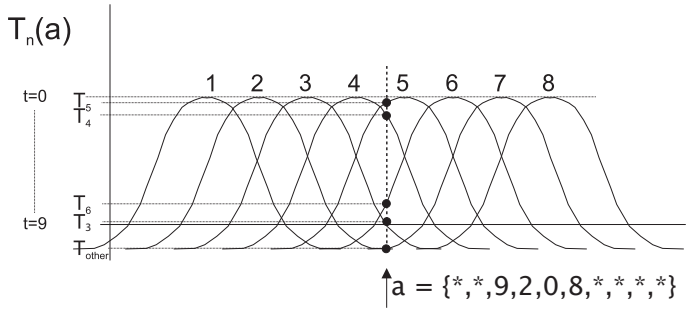


Fig. 16 Encoding with overlapping Gaussian receptive fields. An input value a is translated into firing times for the input-neurons encoding this input-variable. The highest stimulated neuron (neuron 5), fires at a time close to $T = 0$, whereas less stimulated neurons, as for instance neuron 3, fire at increasingly later times.

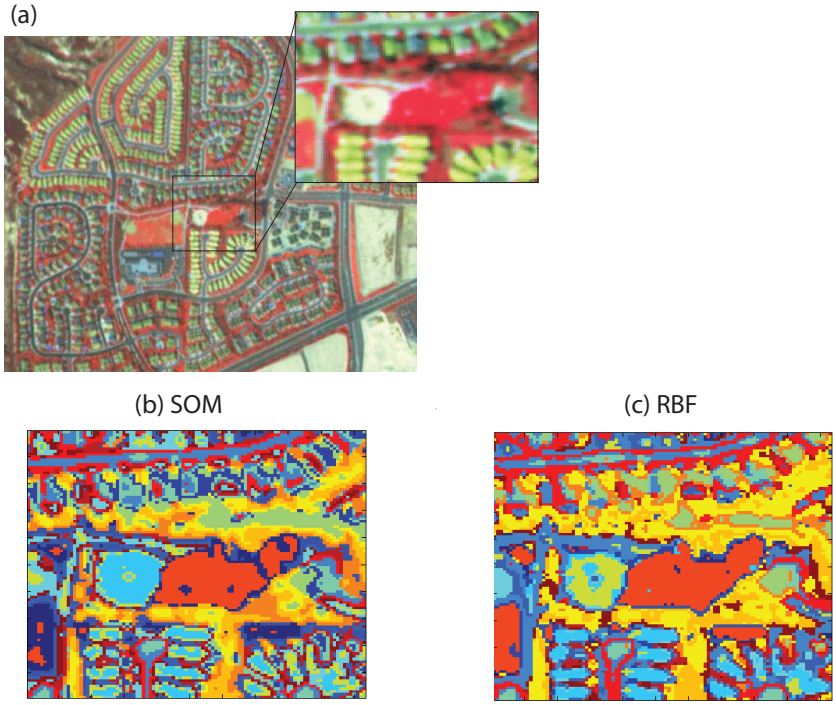


Fig. 17 Unsupervised classification of remote sensing data. (a) The full image. Inset: image cutout that is actually clustered. (b) Classification of the cutout as obtained by clustering with a Self-Origanizing Map (SOM) (c) Spiking Neuron Network RBF classification of the cutout image.

Supervised learning in multi-layer networks

A number of approaches for supervised learning in standard multi-layer feedforward networks have been developed based on gradient descent methods, the best known being error backpropagation. As developed in [18], *SpikeProp* starts from error backpropagation to derive a supervised learning rule for networks of spiking neurons that transfer the information in the timing of a single spike. This learning rule is analogous to the derivation rule by Rumelhart *et al.* [139], but SpikeProp applies to spiking neurons of the SRM type. To overcome the discontinuous nature of spiking neurons, the thresholding function is approximated, thus linearizing the model at a neuron's output spike times. As in the unsupervised SNN described above, each connection between neurons may have multiple delayed synapses with varying weights (see Figure 15). The SpikeProp algorithm has been shown to be capable of learning complex non-linear tasks in spiking neural networks with similar accuracy as traditional sigmoidal neural networks, including the archetypical XOR classification task (Figure 18).

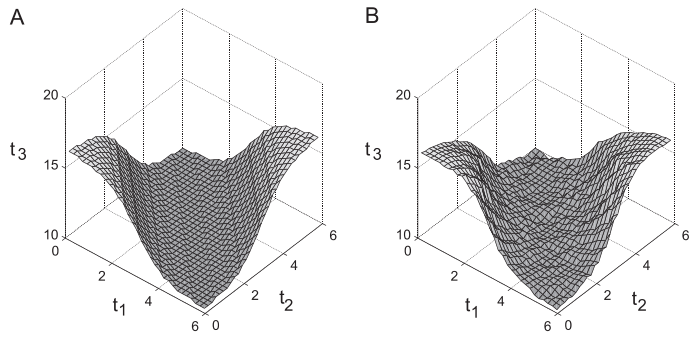


Fig. 18 Interpolated XOR function $f(t_1,t_2) : [0,6] \rightarrow [10,16]$. a) Target function. b) Spiking Neuron Network output after training.

The SpikProp method has been successfully extended to adapt the synaptic delays along the error-gradient, as well as the decay for the α-function and the threshold [149, 148]. Xin *et al.* [186] have further shown that the addition of a simple momentum term significantly speeds up convergence of the SpikeProp algorithm. Booij & Nguyen [21] have, analogously to the method for BackPropagation-Through-Time, extended SpikeProp to account for neurons in the input and hidden layer to fire multiple spikes.

McKennoch, Voegtlin and Bushnell [111] derive a supervised Theta-learning rule for multi-layer networks of Theta-neurons. By mapping QIF neurons to the canonical Theta neuron model (a non-linear phase model - see Section 2.2), a more dynamic spiking neuron model is placed at the heart of the spiking neuron network. The Theta neuron phase model is cyclic and allows for a continuous reset. Derivatives can then be computed without any local linearization assumptions.

Some sample results showing the performance of both SpikeProp and the Theta Neuron learning rule as compared to error-backpropagation in traditional neural networks is shown in Table 1. The more complex Theta-neuron learning allows for a smaller neuronal network to optimally perform classification.

As with SpikeProp, Theta-learning requires some careful fine-tuning of the network. In particular, both algorithms are sensitive to *spike-loss*, in that no error-gradient is defined when the neuron does not fire for any pattern, and hence will never recover. McKennoch *et al.* heuristically deal with this issue by applying alternating periods of coarse learning, with a greater learning rate, and fine tuning, with a small learning rate.

As demonstrated in [10], non-gradient based methods like Evolutionary Strategies do not suffer from these tuning issues. For MLP networks based on various spiking neuron models, performance comparable to SpikeProp is shown. An evolutionary strategy is however very time consuming for large-scale networks.

Table 1 Classification results for the SpikeProp and Theta-neuron supervised learning methods on two benchmarks, the Fisher Iris dataset and the Wisconsin Breast Cancer dataset. The results are compared to standard error-backpropagation, *BP A* and *BP B* denoting the standard Matlab backprop implementation with default parameters, where their respective network sizes are set to correspond to either the SpikeProp or the Theta-neuron neural networks. (taken from [111]).

Learning Method	Network Size	Epochs	Train	Test
Fisher Iris Dataset				
SpikeProp	50x10x3	1000	97.4%	96.1%
BP A	50x10x3	2.6e6	98.2%	95.5%
BP B	4x8x1	1e5	98.0%	90.0%
Theta Neuron BP	4x8x1	1080	100%	98.0%
Wisconsin Breast Cancer Dataset				
SpikeProp	64x15x2	1500	97.6%	97.0%
BP A	64x15x2	9.2e6	98.1%	96.3%
BP B	9x8x1	1e5	97.2%	99.0%
Theta Neuron BP	9x8x1	3130	98.3%	99.0%

4.2 Reservoir Computing

Clearly, the architecture and dynamics of an SNN can be matched, by temporal coding, to traditional connectionist models, such as multilayer feedforward networks or recurrent networks. However, since networks of spiking neurons behave decidedly different as compared to traditional neural networks, there is no pressing reason to design SNNs within such rigid schemes.

According to biological observations, the neurons of biological SNNs are sparsely and irregularly connected in space (network topology) and the variability of spike flows implies they communicate irregularly in time (network dynamics) with a low average activity. It is important to note that the network topology becomes a simple underlying support to the neural dynamics, but that only active neurons are contributing to information processing. At a given time t, the sub-topology defined by active neurons can be very sparse and different from the underlying network architecture (e.g. local clusters, short or long path loops, synchronized cell assemblies), comparable to the active brain regions that appear coloured in brain imaging scanners. Clearly, an SNN architecture has no need to be regular. A network of spiking neurons can even be defined randomly [101, 72] or by a loosely specified architecture, such as a set of neuron groups that are linked by projections, with a given probability of connection from one group to the other [114]. However, the nature of a connection has to be prior defined as an excitatory or inhibitory synaptic link, without subsequent change, except for the synaptic efficacy. That is the weight value can be modified, but not the weight sign.

With this in mind, a new family of networks has been developed that is specifically suited to processing temporal input / output patterns with spiking neurons. The new paradigm is named *Reservoir Computing* as an unifying term for which the precursor models are Echo State Networks (ESNs) and Liquid State Machines

(LSMs). Note that the terms "reservoir computing" are not reserved to SNNs since ESN has been first designed with sigmoidal neurons, but the present chapter mainly presents reservoir computing with SNNs.

Main characteristics of reservoir computing models

The topology of a reservoir computing model (Figure 19) can be defined as follows:

- a layer of K neurons with input connections toward the reservoir,
- a recurrent network of M neurons, interconnected by a random and sparse set of weighted links: the so-called *reservoir*, that is usually left untrained,
- a layer of L *readout neurons* with trained connections from the reservoir.

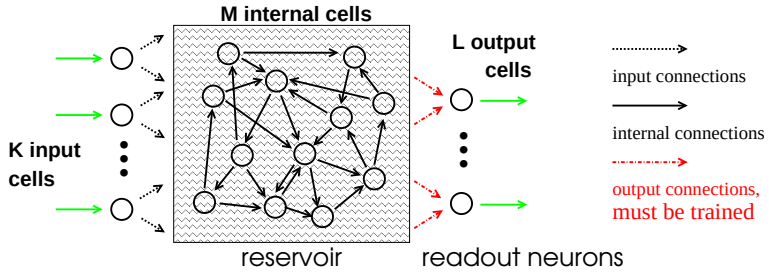


Fig. 19 Architecture of a Reservoir Computing network: the "reservoir" is a set of M internal neurons, with random and sparse connectivity.

The early motivation of reservoir computing is the well-known hardness to find efficient supervised learning rules to train recurrent neural networks, as attested by the limited success of methods like Back-Propagation Through Time (BPTT), Real-Time Recurrent Learning (RTRL) or Extended Kalman Filtering (EKF). The difficulty stems from the lack of knowledge on the way to control the behavior of the complex dynamic system resulting from the presence of cyclic connections in the network architecture. The main idea of reservoir computing is to renounce training the internal recurrent network and only to pick out, by way of the readout neurons, the relevant part of the dynamic states induced in the reservoir by the network inputs. Only the reading-out of this information is subject to training, usually by very simple learning rules, such as linear regression. The success of the method is based on the high power and accuracy of self-organization inherent to a random recurrent network.

In SNN versions of reservoir computing, a soft kind of unsupervised, local training is often added by applying a synaptic plasticity rule like STDP inside the reservoir. Since STDP has been directly inspired from the observation of natural processing in the brain (see Section 2.4), its computation does not require supervised control nor understanding the network dynamics.

The paradigm of "reservoir computing" is only commonly referred to as such since approximately 2007, and encompasses several seminal models in the literature that predate this generalized notion by a few years. The next section describes the two founding models that have been designed concurrently in the early 2000's, by Jaeger for the ESN [71] and by Maass *et al.* for the LSM [101].

Echo State Network (ESN) and Liquid State Machine (LSM)

The original design of *Echo State Network*, proposed by Jaeger in 2001 [71], has been intended to learn time series $(\mathbf{u}(1),\mathbf{d}(1)),\ldots,(\mathbf{u}(T),\mathbf{d}(T))$ with recurrent neural networks. The internal states of the reservoir are supposed to reflect, as an "echo", the concurrent effect of a new teacher input $u(t+1)$ and a teacher-forcing output $d(t)$, related to the previous time. Therefore, Jaeger's model includes backward connections from the output layer toward the reservoir (see Figure 20 (a)) and the network training dynamics is governed by the following equation:

$$\mathbf{x}(t+1) = f\left(W^{in}\mathbf{u}(t+1)+W\mathbf{x}(t)+W^{back}\mathbf{d}(t)\right) \tag{7}$$

where $x(t+1)$ is the new state of the reservoir, W^{in} is the input weight matrix, W the matrix of weights in the reservoir and W^{back} the matrix of feedback weights, from the output layer to the reservoir. The learning rule for output weights W^{out} (feedforward connections from reservoir to output) consists of a linear regression algorithm, e.g. Least Mean Squares: At each step, the network states $x(t)$ are collected into a matrix M, after a washout time t_0, and the sigmoid-inverted teacher output $tanh^{-1}\mathbf{d}(n)$ into a matrix T, in order to obtain $(W^{out})^t = M^{\dagger}T$ where $M^{\dagger}$ is the pseudo-inverse of M. In exploitation phase, the network is driven by novel input sequences $u(t)$, with $t \geq T$ (desired output $d(t)$ are unknown), and produces computed output $y(t)$ with coupled equations like:

$$\mathbf{x}t+1) = f\left(W^{in}\mathbf{u}(t+1)+W\mathbf{x}(t)+W^{back}\mathbf{y}(t)\right) \tag{8}$$

$$\mathbf{y}(t+1) = f^{out}\left(W^{out}[\mathbf{u}(t+1),\mathbf{x}(t+1),\mathbf{y}(t)]\right) \tag{9}$$

For the method to be efficient, the network must have the "Echo State Property", i.e. the properties of being state contracting, state forgetting and input forgetting, that give it a behavior of "fading memory". As stated by Jaeger, a necessary (and usually sufficient) condition is to choose a reservoir weight matrix W with a spectral radius $\mid \lambda_{max} \mid$ slightly lower than 1. Since the weights are randomly chosen, this condition is not straightforward; common practice however is to rescale W after randomly initializing the network connections. An important remark must made: the condition on the spectral radius is no longer clearly relevant when the reservoir is an SNN with fixed weights, and totally vanishes when an STDP rule is applied to the reservoir. A comparative study of several measures for the reservoir dynamics, with different neuron models, can be found in [175].

ESNs have been successfully applied in many experimental settings, with networks no larger than 20 to 400 internal units, e.g. in mastering the benchmark task of learning the Mackey-Glass chaotic attractor [71]. Although the first design of ESN was for networks of sigmoid units, Jaeger has also introduced spiking neurons (LIF model) in the ESNs [72, 74]. Results improve substantially over standard ESNs, e.g. in the task of generating a slow sinewave ($\mathbf{d}(n) = 1/5\sin(n/100)$), that becomes easy with a leaky integrator network [72].

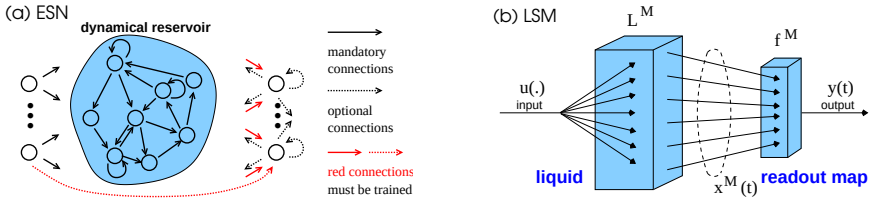


Fig. 20 Architecture of the two founding models of reservoir computing: ESN and LSM.

The basic motivation of the *Liquid State Machine*, defined by Maass, Natschläger and Markram in 2002 [101], was to explain how a continuous stream of inputs $u(.)$ from a rapidly changing environment can be processed in real time by recurrent circuits of Integrate-and-Fire neurons (Figure 20 (b)). The solution they propose is to build a "liquid fiter" L^M - the reservoir - that operates similarly to water undertaking the transformation from the low-dimensional space of a set of motors stimulating its surface into a higher dimensional space of waves in parallel. The liquid states $x^M(t)$ are transformed by a readout map f^M to generate output $y(t)$ that can appear as stable and appropriately scaled responses given by the network, even if the internal state never converges to a stable attractor. Simulating such a device on neural microcircuits, Maass *et al.* have shown that a readout neuron receiving inputs from hundreds or thousands of neurons can learn to extract salient information from the high-dimensional transient states of the circuit and can transform transient circuit states into stable outputs.

In mathematical terms, the liquid state is simply the current output of some operator L^M that maps input functions $u(.)$ onto functions $x^M(t)$. The L^M operator can be implemented by a randomly connected recurrent neural network. The second component of an LSM is a "memoryless readout map" f^M that transforms, at every time t, the current liquid state into the machine output, according to equations:

$$x^M(t) = \left(L^M(u)\right)(t) \tag{10}$$

$$y(t) = f^M\left(x^M(t)\right) \tag{11}$$

The readout is usually implemented by one or several Integrate-and-Fire neurons that can be trained to perform a specific task using very simple learning rules, such as a linear regression or the p-delta rule [5].

Often, in implementation, the neural network playing the role of liquid filter is inspired from biological modeling cortical columns. Therefore, the reservoir has a 3D topology, with a probability of connection that decreases as a gaussian function of the distance between neurons.

The readout map is commonly task-specific. However, the hallmark feature of neural microcircuits is their ability to carry out several real-time computations in parallel within the same circuitry. It appears that a readout neuron is able to build a sort of equivalence class among dynamical states, and then to well recognize similar (but not equal) states. Moreover, several readout neurons, trained to perform different tasks, may enable parallel real-time computing.

LSMs have been successfully applied to several non-linear problems, such as the XOR and many others. LSMs and ESNs are very similar models of reservoir computing that promise to be convenient for both exploiting and capturing most temporal features of spiking neuron processing, especially for time series prediction and for temporal pattern recognition. Both models are good candidates for engineering applications that process temporally changing information.

Related reservoir computing work

An additional work that has been linked to the family of "reservoir computing" models after being published, is the Back-Propagation DeCorrelation rule (BPDC), proposed by Steil in 2004 [161]. As an extension of the Atiya-Parlos' s learning rule in recurrent neural networks [4], the BPDC model is based on a multilayer network with fixed weights until the last layer. Only this layer has learnable weights both from the reservoir (the multilayer network) to the readout (the last layer) and recurrently inside the readout layer. However the BPDC model has not been proposed with spiking neurons so far, even if that appears to be readily feasible.

Another approach, by Paugam-Moisy *et al.* [127], takes advantage of the theoretical results proving the importance of delays in computing with spiking neurons (see Section 3) for defining a supervised learning rule acting on the delays of connections (instead of weights) between the reservoir and the readout neurons. The reservoir is an SNN, with an STDP rule for adapting the weights to the task at hand, where can be observed that polychronous groups (see Section 3.2) are activated more and more selectively as training goes on. The learning rule of readout delays is based on a temporal margin criterion inspired from Vapnik's theory.

There exist reservoir computing networks that make use of evolutionary computation for training the weights of the reservoir, such as Evolino [144], and several other models are currently proposed, with or without spiking neurons [40, 76, 75]. Although the research area is in rapid expansion, several papers [175, 151, 94] propose valuable surveys.

4.3 Other SNN research tracks

Besides that efforts to apply traditional learning rules to SNNs, and the development of reservoir computing, there are many research efforts that relate to learning with spiking neurons.

Much research is for instance carried out on deriving theoretically principled learning rules for spiking neurons, for instance on Information Bottleneck learning rules that attempt to maximize measures of mutual information between input and output spike trains [8, 29, 129, 11, 167, 166, 130, 19, 131, 27]. The aim of this work on theoretically principled learning is typically to come to easily understood methods that have spiking neurons carry out some form of Independent Component Analysis (ICA)[166, 82], or Principal Component Analysis (PCA) [27], or focus on sparse efficient coding [124, 178, 92, 157].

These methods have variable applicability to real world problems, though some have demonstrated excellent performance: Smith & Lewicki [157] develop an efficient encoding of auditory in spike-trains based on sparse over-complete dictionaries that outperforms many standard filter based approaches. Forms of reinforcement learning have been developed based on the combination of reward modulation and STDP [185, 65, 91]. Many of these algorithms are highly technical and much of this research is still converging to practical algorithms. We only mention these directions here, and leave it to the reader to pursue the current state-of-the-art in these areas.

Just as advances in neurosciences have contributed to the re-evaluation of the significance of the timing and presence of single spikes in neuronal activity, advances in neuropsychology suggest that brain-like systems are able to carry out at least some forms of Bayesian inference [83, 36]. As a result, the implementation of Bayesian inference algorithms in neural networks has received much attention, with a particular emphasis on networks of spiking neurons.

In this line of research, the activity in neural network is somehow related representing probability distributions. Much of this research however relies on noisy, stochastic spiking neurons that are characterized by a spike-density, and Bayesian inference is implicitly carried out by large populations of such neurons [9, 188, 141, 183, 133, 49, 17, 64, 95]. As noted by Deneve [38], coding probabilities with stochastic neurons "has two major drawbacks. First,[...], it adds uncertainty, and therefore noise, to an otherwise deterministic probability computation. Second, [...], the resulting model would not be self-consistent since the input and output firing rates have different meanings and different dynamics."

In [38, 39], an alternative approach is developed for binary log-likelihood estimation in an SNN. Such binary log-likelihood estimation in an SNN has some known limitations: It can only perform exact inference in a limited family of generative models, and in a hierarchical model, only the objects highest in the hierarchy truly have a temporal dynamic. Interestingly, in this model neurons still exhibit a Poisson-like distribution of synaptic events. However, rather than reflecting stochasticity due to noisy firing mechanism, it reflects the sensory input-noise. Still, this type of SNN is at the forefront of current developments and many advances in this direction are to be expected.

5 Discussion

This chapter has given an outline of some of the most important ideas, models and methods in the development of Spiking Neuron Networks, with a focus on pattern recognition and temporal data processing, such as time series. By necessity, many related subjects are not treated in detail here. Variants of the models and methods described in this chapter, and variants thereof, are increasingly being applied to real world pattern recognition. Section 4.1 listed some results on SNN algorithms applied to traditional pattern recognition, where a dataset of numeric vectors are mapped to a classification. However, as was emphasized in the section on reservoir computing, many interesting application domains have an additional temporal dimension: Not just the immediate data are important, but the *sequence* of data. An increasing amount of work deals with applying SNN concepts to various application domains with important temporal dynamics, such as speech processing, active vision for computer, and autonomous robotics.

5.1 Pattern recognition with SNNs

Considerable work has focused on developing SNNs that are suitable for speech processing [176, 58, 179, 93]. Verstraeten *et al.* [176] develop a model based on Liquid State Machines that is trained to recognize isolated words. They compare several front-end signal encoding methods, and find that a nature-inspired front-end like a "Lyon Passive Ear" outperforms other methods when an LSM is applied. Similarly, Holmberg, *et al* [58] develop a method for automatic speech recognition grounded in SNNs. As a "front-end", they simulate a part of the inner ear, and then simulate octopus spiking neurons to encode the inner-ear signal in a spike-train. They subsequently use a fairly simple classifier to recognize speech from both the inner-ear simulation and the spiking neuron spike-trains. Wang & Pavel [179] use an SNN to represent auditory signals based on using the properties of the spiking neuron refractory period. In their SNN, they convert amplitude to temporal code while maintaining phase information of the carrier. They propose that for auditory signals, the narrow band envelope information could be encoded simply in the temporal inter-spike intervals. Rank order coding with spiking neural networks has been explored for speech recognition by Loiselle *et al.* [93]. They show it is an efficient method (fast response / adaptation ability) when having only small training sets.

One important fact that the speech processing case studies highlight is that traditional preprocessing techniques do not provide optimal front-ends and back-ends for subsequent SNN processing. Still, many promising features have been pointed out, like robustness to noise, and combining SNN processing with other methods is proposed as a promising research area.

In parallel, a number of SNN-based systems have been developed for computer vision, for example using spike asynchrony [156]; sparse image coding using an asynchronous spiking neural network [128]; a synchronization-based dynamic vi-

sion model for image segmentation [6]; saliency extraction with a distributed spiking neural network [32, 108, 31]; and SNNs applied to character recognition [184].

SNN-based systems also develop increasingly in the area of robotics, where fast processing is a key issue [104, 163, 44, 43, 126, 50], from wheels to wings, or legged locomotion. The special abilities of SNNs for fast computing transient temporal patterns make them on first line for designing efficient systems in the area of autonomous robotics. This perspective is often cited but not yet fully developed.

Other research domains mention the use of SNNs, such as Echo State Networks for motor control (e.g. [142]), prediction in the context of wireless telecommunications (e.g. [73]) or neuromorphic approaches to rehabilitation, in medecine (e.g. [85]). In this context, the "Neuromorphic Engineer" newsletter[16] often publishes articles on applications developed with SNNs.

It is worth remarking that SNNs are ideal candidates for designing *multimodal interfaces*, since they can represent and process very diverse information in a unifying manner based on time, from such different sources as visual, auditory, speech or other sensory data. An application of SNNs to audio-visual speech recognition has been proposed by Séguier and Mercier in [152]. Paugam-Moisy *et al* [34] have developed, with traditional NNs, a modular connectionist model of multimodal associative memory including temporal aspects of visual and auditory data processing [22]. Such a multi-modal framework, applied to a virtual robotic prey-predator environment, with spiking neuron networks as functional modules, has proved capable to simulate high-level natural behavior such as cross-modal priming [113] or real-time perceptive adaptation to changing environment [30].

5.2 Implementing SNNs

Since SNNs perform computations in such a different way as compared to traditional NNs, the way to program an SNN model for application purpose has to be revised also. The main interest of SNN simulation is to take into account the precise timing of spike firing, hence the width of the time window used for discrete computation of the successive network states must remain narrow (see Figure 6 in Section 1.4), and consequently only a few spike-events occur at each time step: In Figure 6, only 2 spikes were fired inside the Δt time range, among the 64 potential connections linking the 8 neurons. Hence, inspecting all the neurons and synapses of the network at each time step is exceedingly time consuming: in this example, a clock-based simulation (i.e. based on a time window) computes zero activity in 97% of the computations ! An event-driven simulation is clearly more suitable for sequential simulations of spiking neural networks [181, 109, 105, 136, 135, 111], as long as the activity of an SNN can be fully described by a set of dated spikes. Nevertheless, event-driven programming requires the next spike time can be explicitly computed in reasonable time, so that not all models of neurons can be used.

[16] Institute of Neuromorphic Engineering newsletter: *http://www.ine-web.org/*

At the same time, SNN simulation can highly benefit from parallel computing, substantially more so than traditional NNs. Unlike a traditional neuron in rate coding, a spiking neuron does not need to receive weight values from each presynaptic neuron at each computation step. Since at each time step only a few neurons are active in an SNN, the classic bottleneck of message passing is removed. Moreover, computing the updated state of membrane potential (e.g. for a SRM or LIF model neuron) is more complex than computing a weighted sum (e.g. for threshold unit). Therefore communication time and computation cost are much more well-balanced in SNN parallel implementation as compared to traditional NNs, as proved by the parallel implementation of the *SpikeNET* software [37].

Well-known simulators of spiking neurons are for example *GENESIS* [23] and *NEURON* [54], but they have been designed principally for programming detailed biophysical models of isolated neurons, rather than for fast simulation of very large scale SNNs. However, NEURON has been updated with an event-driven mechanism on the one hand [55] and a version for parallel machines on the other hand [56]. *BRIAN* (*http://brian.di.ens.fr/*) is a mainly clock-based simulator with an optional event-driven tool, whereas *MVASpike* (*http://mvaspike.gforge.inria.fr/*) is a purely event-driven simulator [136]. *DAMNED* is a parallel event-driven simulator [118]. A comparative and experimental study of several SNN simulators can be found in Brette *et al.* [24]. Most simulators are currently programmed in C or C++. Others are Matlab toolboxes, such as Jaeger's toolbox for ESNs available from the web page (*http://www.faculty.jacobs-university.de/hjaeger/esn_research.html*) or the *Reservoir Computing Toolbox*, available at URL *http://snn.elis.ugent.be/node/59* and briefly presented in the last section of [175]. A valuable tool for developers could be *PyNN* (*http://neuralensemble.org/trac/PyNN*), a Python package for simulator-independent specification of neuronal network models.

Hardware implementations of SNNs are also being actively pursued. Several chapters in [99] are dedicated to this subject, and more recent work can be found for instance in [170, 53, 77, 33, 125, 116, 150].

5.3 Conclusion

This chapter has given an overview of the current state-of-the-art in Spiking Neuron Networks: its biological inspiration, the models that underlie the networks, some theoretical results on computational complexity and learnability, learning rules, both traditional and novel, and some current application areas and results. The novelty of the concept of SNNs means that many lines of research are still open and are actively being pursued.

References

1. L.F. Abbott and S.B. Nelson. Synaptic plasticity: taming the beast. *Nature Neuroscience*, 3:1178–1183, 2000.
2. M. Abeles. *Corticonics: Neural Circuits of the Cerebral Cortex*. Cambridge Univ. Press, 1991.
3. S. Achard and E. Bullmore. Efficiency and cost of economical brain functional networks. *PLoS Computational Biology*, 3(2):e17, 2007.
4. A. Atiya and A.G. Parlos. New results on recurrent network training: Unifying the algorithms and accelerating convergence. *IEEE Trans. on Neural Networks*, 11(3):697–709, 2000.
5. P. Auer, H. Burgsteiner, and W. Maass. A learning rule for very simple universal approximators consisting of a single layer of perceptrons. *Neural Networks*, 21(5):786–795, 2008.
6. H. Azhar, K. Iftekharuddin, and R. Kozma. A chaos synchronization-based dynamic vision model for image segmentation. In *IJCNN'2005, Int. Joint Conf. on Neural Networks*, pages 3075–3080. IEEE–INNS, 2005.
7. A. L. Barabasi and R. Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999.
8. D. Barber. Learning in spiking neural assemblies. In S. Becker, S. Thrun, and K. Obermayer, editors, *NIPS*2002, Advances in Neural Information Processing Systems*, volume 15, pages 165–172. MIT Press, 2003.
9. M. J. Barber, J. W. Clark, and C. H. Anderson. *Neural Representation of Probabilistic Information*, volume 15. MIT Press, 2003.
10. A. Belatreche, L. P. Maguire, and M. McGinnity. Advances in design and application of spiking neural networks. *Soft Computing-A Fusion of Foundations, Methodologies and Applications*, 11:239–248, 2007.
11. A. Bell and L. Parra. Maximising information yields spike timing dependent plasticity. In L.K. Saul, Y. Weiss, and L. Bottou, editors, *NIPS*2004, Advances in Neural Information Processing Systems*, volume 17, pages 121–128. MIT Press, 2005.
12. G.-q. Bi and M.-m. Poo. Synaptic modification in cultured hippocampal neurons: Dependence on spike timing, synaptic strength, and polysynaptic cell type. *J. of Neuroscience*, 18(24):10464–10472, 1998.
13. G.-q. Bi and M.-m. Poo. Synaptic modification of correlated activity: Hebb's postulate revisited. *Annual Review of Neuroscience*, 24:139–166, 2001.
14. W. Bialek, F. Rieke, R. de Ruyter, R.R. van Steveninck, and D. Warland. Reading a neural code. *Science*, 252:1854–1857, 1991.
15. A. Blum and R. Rivest. Training a 3-node neural net is NP-complete. In *NIPS*1988, Advances in Neural Information Processing Systems*, pages 494–501, 1989.
16. A. Blumer, A. Ehrenfeucht, D. Haussler, and M.K. Warmuth. Learnability and the vapnik-chervonenkis dimension. *Journal of the ACM*, 36(4):929–965, 1989.
17. O. Bobrowski, R. Meir, S. Shoham, and Y. C. Eldar. A neural network implementing optimal state estimation based on dynamic spike train decoding. In *NIPS*2006, Advances in Neural Information Processing Systems*, volume 20, 2007.
18. S. M. Bohte, J. N. Kok, and H. La Poutre. Spike-prop: errorbackpropagation in multi-layer networks of spiking neurons. *Neurocomputing*, 48:17–37, 2002.
19. S. M. Bohte and M. C. Mozer. Reducing the variability of neural responses: A computational theory of spike-timing-dependent plasticity. *Neural Computation*, 19:371–403, 2007.
20. S. M. Bohte, H. La Poutre, and J. N. Kok. Unsupervised clustering with spiking neurons by sparse temporalcoding and multilayer rbf networks. *Neural Networks, IEEE Transactions on*, 13:426–435, 2002.
21. O. Booij and H. tat Nguyen. A gradient descent rule for spiking neurons emitting multiple spikes. *Information Processing Letters*, 95:552–558, 2005.
22. Y. Bouchut, H. Paugam-Moisy, and D. Puzenat. Asynchrony in a distributed modular neural network for multimodal integration. In *PDCS'2003, Int. Conf. on Parallel and Distributed Computing and Systems*, pages 588–593. ACTA Press, 2003.

23. J.M. Bower and D. Beeman. *The Book of GENESIS: Exploring Realistic Neural Models with the GEneral SImulation System*. Springer, 1998. 2nd edition.
24. R. Brette, M. Rudolph, T. Hines, D. Beeman, J.M. Bower, and et al. Simulation of networks of spiking neurons: A review of tools and strategies. *J. of Computational Neuroscience*, 23(3):349–398, 2007.
25. N. Brunel and P. E. Latham. *Firing Rate of the Noisy Quadratic Integrate-and-Fire Neuron*, volume 15. MIT Press, 2003.
26. N.J. Buchs and W. Senn. Spike-based synaptic plasticity and the emergence of direction selective simple cells: Simulation results. *J. of Computational Neuroscience*, 13:167–186, 2002.
27. L. Büsing and W. Maass. Simplified rules and theoretical analysis for information bottleneck optimization and pca with spiking neurons. In *NIPS*2007, Advances in Neural Information Processing Systems*, volume 20. MIT Press, 2008.
28. H. Câteau and T. Fukai. A stochastic method to predict the consequence of arbitrary forms of Spike-Timing-Dependent Plasticity. *Neural Computation*, 15(3):597–620, 2003.
29. G. Chechik. Spike-timing dependent plasticity and relevant mutual information maximization. *Neural Computation*, 15(7):1481–1510, 2003.
30. S. Chevallier, H. Paugam-Moisy, and F. Lemaître. Distributed processing for modelling real-time multimodal perception in a virtual robot. In *PDCN'2005, Int. Conf. on Parallel and Distributed Computing and Networks*, pages 393–398. ACTA Press, 2005.
31. S. Chevallier and P. Tarroux. Covert attention with a spiking neural network. In *ICVS'08, Computer Vision Systems*, volume 5008 of *Lecture Notes in Computer Science*, pages 56–65. Springer, 2008.
32. S. Chevallier, P. Tarroux, and H. Paugam-Moisy. Saliency extraction with a distributed spiking neuron network. In *ESANN'06, Advances in Computational Intelligence and Learning*, pages 209–214, 2006.
33. E. Chicca, D. Badoni, V. Dante, M. d'Andreagiovanni, G. Salina, L. Carota, S. Fusi, and P. Del Giudice. A VLSI recurrent network of integrate-and-fie neurons connected by plastic synapses with long-term memory. *IEEE Trans. on Neural Networks*, 14(5):1297–1307, 2003.
34. A. Crépet, H. Paugam-Moisy, E. Reynaud, and D. Puzenat. A modular neural model for binding several modalities. In H. R. Arabnia, editor, *IC-AI'2000, Int. Conf. on Artificial Intelligence*, pages 921–928, 2000.
35. G. Cybenko. Approximation by superpositions of a sigmoidal function. *Math. Control, Signal Systems*, 2:303–314, 1988.
36. N. D. Daw and A. C. Courville. The pigeon as particle filter. In *NIPS*2007, Advances in Neural Information Processing Systems*, volume 20. MIT Press, 2008.
37. A. Delorme, J. Gautrais, R. Van Rullen, and S. Thorpe. SpikeNET: A simulator for modeling large networks of integrate and fire neurons. *Neurocomputing*, 26–27:989–996, 1999.
38. S. Deneve. Bayesian spiking neurons i: Inference. *Neural Computation*, 20:91–117, 2008.
39. S. Deneve. Bayesian spiking neurons ii: Learning. *Neural Computation*, 20:118–145, 2008.
40. A. Devert, N. Brèdeche, and M. Schoenauer. Unsupervised learning of Echo State Networks: A case study in artificial embryogeny. In N. Montmarché et al., editor, *Artificial Evolution, Selected Papers*, volume 4926/2008 of *Lecture Notes in Computer Science*, pages 278–290, 2007.
41. V. M. Eguíluz, G. A. Chialvo, D. R.and Cecchi, M. Baliki, and A. V. Apkarian. Scale-free brain functional networks. *Physical Review Letters*, 94(1):018102, 2005.
42. G. B. Ermentrout and N. Kopell. Parabolic bursting in an excitable system coupled with a slow oscillation. *SIAM Journal on Applied Mathematics*, 46:233, 1986.
43. D. Floreano, Y. Epars, J.-C. Zufferey, and C. Mattiussi. Evolution of spiking neural circuits in autonomous mobile robots. *Int. J. of Intelligent Systems*, 21(9):1005–1024, 2006.
44. D. Floreano, J.C. Zufferey, and J.D. Nicoud. From wheels to wings with evolutionary spiking neurons. *Artificial Life*, 11(1-2):121–138, 2005.
45. K. Funahashi. On the approximate realization of continuous mapings by neural networks. *Neural Networks*, 2(3):183–192, 1989.

46. W.. Gerstner. Time structure of the activity in neural network models. *Physical Review E*, 51:738–758, 1995.
47. W. Gerstner and W.M. Kistler. Mathematical formulations of hebbian learning. *Biological Cybernetics*, 87(5-6):404–415, 2002.
48. W. Gerstner and J.L. van Hemmen. How to describe neuronal activity: Spikes, rates or assemblies? In J. D. Cowan, G. Tesauro, and J. Alspector, editors, *NIPS*1993, Advances in Neural Information Processing System*, volume 6, pages 463–470. MIT Press, 1994.
49. S. Gerwinn, J. H. Macke, M. Seeger, and M. Bethge. Bayesian inference for spiking neuron models with a sparsity prior. In *NIPS*2006, Advances in Neural Information Processing Systems*, volume 19. MIT Press, 2007.
50. C. Hartland and N. Bredèche. Using Echo State Networks for robot navigation behavior acquisition. In *ROBIO'07*, Sanya, Chine, 2007.
51. D.O. Hebb. *The Organization of Behaviour*. Wiley, New York, 1949.
52. W Heiligenberg. *Neural Nets in Electric Fish*. MIT Press, 1991.
53. H. H. Hellmich, M. Geike, P. Griep, M. Rafanelli, and H. Klar. Emulation engine for spiking neurons and adaptive synaptic weights. In *IJCNN'2005, Int. Joint Conf. on Neural Networks*, pages 3261–3266. IEEE-INNS, 2005.
54. M.L. Hines and N.T. Carnevale. The NEURON simulation environment. *Neural Computation*, 9:1179–1209, 1997.
55. M.L. Hines and N.T. Carnevale. Discrete event simulation in the NEURON environment. *Neurocomputing*, pages 1117–1122, 2004.
56. M.L. Hines and N.T. Carnevale. Translating network models to parallel hardware in NEURON. *J. Neurosci. Meth.*, 169:425–455, 2008.
57. A.L. Hodgkin and A.F. Huxley. A quantitative description of ion currents and its applications to conduction and excitation in nerve membranes. *J. of Physiology*, 117:500–544, 1952.
58. M. Holmberg, D. Gelbart, U. Ramacher, and W. Hemmert. Isolated word recognition using a liquid state machine. In *EuroSpeech'2005, European Conference on Speech Communication*, 2005.
59. J. J. Hopfield. Pattern recognition computation using action potential timing for stimulus representation. *Nature*, 376:33–36, 1995.
60. J.J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proc. Natl. Acad. Sci.*, 79(8):2554–2558, 1982.
61. J.J. Hopfield and C.D. Brody. What is a moment ? "Cortical" sensory integration over a brief interval. *Proc. Natl. Acad. Sci.*, 97(25):13919–13924, 2000.
62. J.J. Hopfield and C.D. Brody. What is a moment ? Transient synchrony as a collective mechanism for spatiotemporal integration. *Proc. Natl. Acad. Sci.*, 98(3):1282–1287, 2001.
63. K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.
64. Q. J. M. Huys, R. S. Zemel, R. Natarajan, and P. Dayan. Fast population coding. *Neural Computation*, 19:404–441, 2007.
65. E. M. Izhikevich. Solving the distal reward problem through linkage of STDP and dopamine signaling. *Cerebral Cortex*, 2007.
66. E.M. Izhikevich. Simple model of spiking neurons. *IEEE Trans. in Neural Networks*, 14(6):1569–1572, 2003.
67. E.M. Izhikevich. Which model to use for cortical spiking neurons? *IEEE Trans. in Neural Networks*, 15(5):1063–1070, 2004.
68. E.M. Izhikevich. Polychronization: Computation with spikes. *Neural Computation*, 18(2):245–282, 2006.
69. E.M. Izhikevich and N.S. Desai. Relating STDP and BCM. *Neural Computation*, 15(7):1511–1523, 2003.
70. E.M. Izhikevich, J.A. Gally, and G.M. Edelman. Spike-timing dynamics of neuronal groups. *Cerebral Cortex*, 14:933–944, 2004.
71. H. Jaeger. The "echo state" approach to analysins and training recurrent neural networks. Technical Report TR-GMD-148, German National Research Center for Information Technology, 2001.

72. H. Jaeger. Tutorial on training recurrent neural networks, covering BPTT, RTRL, EKF and the "echo state network" approach. Technical Report TR-GMD-159, German National Research Center for Information Technology, 2002.
73. H. Jaeger and H. Haas. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless telecommunication. *Science*, pages 78–80, 2004.
74. H. Jaeger and M. Lukoševičius. Optimization and applications of echo state networks with leaky-integrator neurons. *Neural Networks*, 20(3):335–352, 2007.
75. F. Jiang, H. Berry, and M. Schoenauer. Supervised and evolutionary learning of Echo State Networks. In G. Rudolph et al., editor, *Parallel Problem Solving from Nature (PPSN'08)*, Lecture Notes in Computer Science, 2008.
76. F. Jiang, H. Berry, and M. Schoenauer. Unsupervised learning of Echo State Networks: Balancing the double pole. In C. Ryan et al., editor, *Genetic and Evolutionary Computation Conference (GECCO)*, 2008.
77. S. Johnston, G. Prasad, L. Maguire, and McGinnity. Comparative investigation into classical and spiking neuron implementations on FPGAs. In *ICANN'2005, Int. Conf. on Artificial Neural Networks*, volume 3696 of *LNCS*, pages 269–274. Springer-Verlag, 2005.
78. J.S. Judd. *Neural network design and the complexity of learning*. MIT Press, 1990.
79. R. Kempter, W. Gerstner, and J. L. van Hemmen. Hebbian learning and spiking neurons. *Physical Review E*, 59(4):4498–4514, 1999.
80. W.M. Kistler. Spike-timing dependent synaptic plasticity: a phenomenological framework. *Biological Cybernetics*, 87(5-6):416–427, 2002.
81. W.M. Kistler, W. Gerstner, and J.L. van Hemmen. Reduction of hodgkin-huxley equations to a single-variable threshold model. *Neural Computation*, 9:1015–1045, 1997.
82. S. Klampfl, R. Legenstein, and W. Maass. Spiking neurons can learn to solve information bottleneck problems and to extract independent components. *Neural Computation*, 2008. in press.
83. K. P. Koerding and D. M. Wolpert. Bayesian integration in sensorimotor learning. *Nature*, 427:244–247, 2004.
84. T. Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43:59–69, 1982.
85. J.J. Kutch. Neuromorphic approaches to rehabilitation. *The Neuromorphic Engineer*, 1(2):1–2, 2004.
86. N. Kuwabara and N. Suga. Delay lines and amplitude selectivity are created in subthalamic auditory nuclei: the brachium of the inferior colliculus of the mustached bat. *J. of Neurophysiology*, 69:1713–1724, 1993.
87. L. Lapicque. Recherches quantitatives sur l'excitation électrique des nerfs traité comme une polarization. *J. Physiol. Pathol. Gen.*, 9:620–635, 1907. cited by Abbott, L.F., in Brain Res. Bull. 50(5/6):303–304.
88. Y. LeCun, L. D. Jackel, L. Bottou, C. Cortes, J. S. Denker, H. Drucker, I. Guyon, U. A. Muller, E. Sackinger, and P. Simard. *Learning algorithms for classification: A comparison on handwritten digit recognition*, volume 276. Singapore, 1995.
89. R. Legenstein and W. Maass. What makes a dynamical system computationally powerful? In S. Haykin, J. C. Principe, T.J. Sejnowski, and J.G. McWhirter, editors, *New Directions in Statistical Signal Processing: From Systems to Brain*. MIT Press, 2005.
90. R. Legenstein, C. Näger, and W. Maass. What can a neuron learn with Spike-Time-Dependent Plasticity? *Neural Computation*, 17(11):2337–2382, 2005.
91. R. Legenstein, D. Pecevski, and W. Maass. Theoretical analysis of learning with reward-modulated Spike-Timing-Dependent Plasticity. In *NIPS*2007, Advances in Neural Information Processing Systems*, volume 20. MIT Press, 2008.
92. M. S. Lewicki. Efficient coding of natural sounds. *Nature Neuroscience*, 5:356–363, 2002.
93. S. Loiselle, J. Rouat, D. Pressnitzer, and S. Thorpe. Exploration of rank order coding with spiking neural networks for speech recognition. In *IJCNN'2005, Int. Joint Conf. on Neural Networks*, pages 2076–2080. IEEE–INNS, 2005.
94. M. Lukoševičius and H. Jaeger. Overview of reservoir recipes. Technical Report 11, Jacobs University Bremen, July 2007.

95. W. J. Ma, J. M. Beck, and A. Pouget. Spiking networks for bayesian inference and choice. *Current Opinion in Neurobiology*, 2008.
96. W. Maass. Fast sigmoidal networks via spiking neurons. *Neural Computation*, 10:1659–1671, 1997.
97. W. Maass. Networks of spiking neurons: The third generation of neural network models. *Neural Networks*, 10:1659–1671, 1997.
98. W. Maass. On the relevance of time in neural computation and learning. *Theoretical Computer Science*, 261:157–178, 2001. (extended version of ALT'97, in LNAI 1316:364-384).
99. W. Maass and C.M. Bishop, editors. *Pulsed Neural Networks*. MIT Press, 1999.
100. W. Maass and T. Natschläger. Networks of spiking neurons can emulate arbitrary Hopfield nets in temporal coding. *Network: Computation in Neural Systems*, 8(4):355–372, 1997.
101. W. Maass, T. Natschläger, and H. Markram. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Computation*, 14(11):2531–2560, 2002.
102. W. Maass and M. Schmitt. On the complexity of learning for a spiking neuron. In *COLT'97, Conf. on Computational Learning Theory*, pages 54–61. ACM Press, 1997.
103. W. Maass and M. Schmitt. On the complexity of learning for spiking neurons with temporal coding. *Information and Computation*, 153:26–46, 1999.
104. W. Maass, G. Steinbauer, and R. Koholka. Autonomous fast learning in a mobile robot. In *Sensor Based Intelligent Robots*, pages 345–356. Springer, 2000.
105. T. Makino. A discrete event neural network simulator for general neuron model. *Neural Computation and Applic.*, 11(2):210–223, 2003.
106. H. Markram, J. Lübke, M. Frotscher, and B. Sakmann. Regulation of synaptic efficacy by coincidence of postsynaptic APs and EPSPs. *Science*, 275:213–215, 1997.
107. H. Markram and M.V. Tsodyks. Redistribution of synaptic efficacy between neocortical pyramidal neurones. *Nature*, 382:807–809, 1996.
108. T. Masquelier, S. J. Thorpe, and K. J. Friston. Unsupervised learning of visual features through spike timing dependent plasticity. *PLoS Comput Biol*, 3:e31, 2007.
109. M. Mattia and P. Del Giudice. Efficient event-driven simulation of large networks of spiking neurons and dynamical synapses. *Neural Computation*, 12:2305–2329, 2000.
110. W.S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943.
111. S. McKennoch, T. Voegtlin, and L. Bushnell. Spike-timing error backpropagation in theta neuron networks. *Neural Computation*, pages 1–37, 2008.
112. D. Meunier. *Une modélisation évolutionniste du liage temporel (in French)*. PhD thesis, University Lyon 2, http://demeter.univ-lyon2.fr/sdx/theses/lyon2/2007/meunier_d, 2007.
113. D. Meunier and H. Paugam-Moisy. A "spiking" Bidirectional Associative Memory for modeling intermodal priming. In *NCI'2004, Int. Conf. on Neural Networks and Computational Intelligence*, pages 25–30. ACTA Press, 2004.
114. D. Meunier and H. Paugam-Moisy. Evolutionary supervision of a dynamical neural network allows learning with on-going weights. In *IJCNN'2005, Int. Joint Conf. on Neural Networks*, pages 1493–1498. IEEE–INNS, 2005.
115. D. Meunier and H. Paugam-Moisy. Cluster detection algorithm in neural networks. In *ESANN'06, Advances in Computational Intelligence and Learning*, pages 19–24, 2006.
116. S. Mitra, S. Fusi, and G. Indiverti. A VLSI spike-driven dynamic synapse which learns only when necessary. In *ISCAS'2006, IEEE Int. Symp. on Circuits and Systems*, 2006. (to appear).
117. A. Mouraud and H. Paugam-Moisy. Learning and discrimination through STDP in a top-down modulated associative memory. In *ESANN'06, Europ. Symp. on Artificial Neural Networks*, pages 611–616, 2006.
118. A. Mouraud, H. Paugam-Moisy, and D. Puzenat. A Distributed And Multithreaded Neural Event Driven simulation framework. In *PDCN'2006, Int. Conf. on Parallel and Distributed Computing and Networks*, pages 212–217, Innsbruck, AUSTRIA, February 2006. ACTA Press.

119. T. Natschläger and B. Ruf. *Online clustering with spiking neurons using Radial Basis Functions*, chapter 4 in "Neuromorphic Systems: Engineering Silicon from Neurobiology" (Hamilton & Smith, Eds). World Scientific, 1998.
120. T. Natschläger and B. Ruf. Spatial and temporal pattern analysis via spiking neurons. *Network: Comp. Neural Systems*, 9(3):319–332, 1998.
121. M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Phys. Rev. E*, 69:026113, 2004.
122. M.E.J. Newman. The structure and function of complex networks. *SIAM Rev.*, 45:167–256, 2003.
123. T. Nowotny, V.P. Zhigulin, A.I. Selverston, H.D.I. Abardanel, and M.I. Rabinovich. Enhancement of synchronization in a hibrid neural circuit by Spike-Time-Dependent Plasticity. *The Journal of Neuroscience*, 23(30):9776–9785, 2003.
124. B. A. Olshausen. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381:607–609, 1996.
125. M. Oster, A.M. Whatley, S.-C. Liu, and R.J. Douglas. A hardware/software framework for real-time spiking systems. In *ICANN'2005, Int. Conf. on Artificial Neural Networks*, volume 3696 of *LNCS*, pages 161–166. Springer-Verlag, 2005.
126. C. Panchev and S. Wermter. Temporal sequence detection with spiking neurons: towards recognizing robot language instructions. *Connection Science*, 18:1–22, 2006.
127. H. Paugam-Moisy, R. Martinez, and S. Bengio. Delay learning and polychronization for reservoir computing. *Neurocomputing*, 71(7-9):1143–1158, 2008.
128. L. Perrinet and M. Samuelides. Sparse image coding using an asynchronous spiking neural network. In *ESANN'2002, Europ. Symp. on Artificial Neural Networks*, pages 313–318, 2002.
129. J.-P. Pfister, D. Barber, and W. Gerstner. Optimal hebbian learning: A probabilistic point of view. In O. Kaynak, E. Alpaydin, E. Oja, and L. Xu, editors, *ICANN/ICONIP 2003, Int. Conf. on Artificial Neural Networks*, volume 2714 of *Lecture Notes in Computer Science*, pages 92–98. Springer, 2003.
130. J.-P. Pfister and W. Gerstner. Beyond pair-based STDP: a phenomenological rule for spike triplet and frequency effects. In *NIPS*2005, Advances in Neural Information Processing Systems*, volume 18, pages 1083–1090. MIT Press, 2006.
131. J.-P. Pfister, T. Toyoizumi, D. Barber, and W. Gerstner. Optimal Spike-Timing-Dependent Plasticity for precise action potential firing in supervised learning. *Neural Computation*, 18(6):1318–1348, 2006.
132. T. Poggio and F. Girosi. Networks for approximation and learning. *Proceedings of the IEEE*, 78(9):1481–1497, 1989.
133. R. P. N. Rao. Hierarchical bayesian inference in networks of spiking neurons. *NIPS*2004, Advances in Neural Information Processing Systems*, 17:1113–1120, 2005.
134. M. Recce. *Encoding information in neuronal activity*, chapter 4 in "Pulsed Neural Networks" (Maass & Bishop, Eds). MIT Press, 1999.
135. J. Reutimann, M. Giugliano, and S. Fusi. Event-driven simulation of spiking neurons with stochastic dynamics. *Neural Computation*, 15(4):811–830, 2003.
136. O. Rochel and D. Martinez. An event-driven framework for the simulation of networks of spiking neurons. In *ESANN'03, European Symposium on Artificial Neural Network*, pages 295–300, 2003.
137. J. Rubin, D.D. Lee, and H. Sompolinsky. Equilibrium properties of temporal asymmetric hebbian plasticity. *Physical Review Letters*, 86:364–366, 2001.
138. M. Rudolph and A. Destexhe. Event-based simulation strategy for conductance-based synaptic interactions and plasticity. *Neurocomputing*, 69:1130–1133, 2006.
139. D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by back-propagating errors. *Nature*, 323:533–536, 1986.
140. D.E. Rumelhart, G.E. Hinton, and R.J. Williams. *Parallel Distributed Processing: Explorations in the microstructure of cognition*, volume I, chapter Learning internal representations by error propagation, pages 318–362. MIT Press, 1986.

141. M. Sahani and P. Dayan. Doubly distributional population codes: Simultaneous representation of uncertainty and multiplicity. *Neural Computation*, 15:2255–2279, 2003.
142. M. Salmen and P.G. Plöger. Echo State Networks used for motor control. In *ICRA'2005, Int. Joint on Robotics and Automation*, pages 1953–1958. IEEE, 2005.
143. A. Saudargiene, B. Porr, and F. Wörgötter. How the shape of pre- and postsynaptic signals can influence STDP: a biophysical model. *Neural Computation*, 16(3):595–625, 2004.
144. J. Schmidhuber, D. Wiestra, D. Gagliolo, and M. Gomez. Training recurrent networks by evolino. *Neural Computation*, 19(3):757–779, 2007.
145. M. Schmitt. On computing boolean functions by a spiking neuron. *Annals of Mathematics and Artificial Intelligence*, 24:181–191, 1998.
146. M. Schmitt. On the sample complexity of learning for networks of spiking neurons with nonlinear synaptic interactions. *IEEE Trans. on Neural Networks*, 15(5):995–1001, 2004.
147. B. Schrauwen, L. Büsing, and R. Legenstein. On computational power and the order-chaos phase transition in Reservoir Computing. In *NIPS*08*, 2009. (to appear).
148. B. Schrauwen and J. Van Campenhout. Extending spikeprop. In *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*, volume 1, 2004.
149. B. Schrauwen and J. Van Campenhout. Improving spikeprop: Enhancements to an error-backpropagation rule for spiking neural networks. In *Proceedings of the 15th ProRISC Workshop*, volume 11, 2004.
150. B. Schrauwen, M. D'Haene, D. Verstraeten, and J. Van Campenhout. Compact hardware for real-time speech recognition using a liquid state machine. In *Neural Networks, 2007. IJCNN 2007. International Joint Conference on*, pages 1097–1102, 2007.
151. B. Schrauwen, D. Verstraeten, and J. Van Campenhout. An overview of reservoir computing: theory, applications and implementations. In *ESANN'07, Advances in Computational Intelligence and Learning*, pages 471–482, 2007.
152. R. Séguie and D. Mercier. Audio-visual speech recognition one pass learning with spiking neurons. In *ICANN '02, Int. Conf. on Artificial Neural Networks*, pages 1207–1212. Springer-Verlag, 2002.
153. W. Senn, H. Markram, and M. Tsodyks. An algorithm for modifying neurotransmitter release probability based on pre- and post-synaptic spike timing. *Neural Computation*, 13(1):35–68, 2001.
154. H.T. Siegelmann. *Neural networks and analog computation, beyond the Turing limit*. Birkhauser, 1999.
155. J. Sima and J. Sgall. On the nonlearnability of a single spiking neuron. *Neural Computation*, 17(12):2635–2647, 2005.
156. Thorpe S.J. and J. Gautrais. Rapid visual processing using spike asynchrony. In M. Mozer, Jordan M.I., and T. Petsche, editors, *NIPS*1996, Advances in Neural Information Processing Systems*, volume 9, pages 901–907. MIT Press, 1997.
157. E. C. Smith and M. S. Lewicki. Efficient auditory coding. *Nature*, 439:978–982, 2006.
158. S. Song, K.D. Miller, and L.F. Abbott. Competitive hebbian learning through spike-time dependent synaptic plasticity. *Nature Neuroscience*, 3(9):919–926, 2000.
159. O. Sporns, G. Tononi, and R. Kotter. The human connectome : A structural description of the human brain. *PLoS Comp. Biology*, 1(4):e42, 2005.
160. D.I. Standage and T.P. Trappenberg. Differences in the subthreshold dynamics of leaky integrate-and-fire ans Hodgkin-Huxley neuron models. In *IJCNN'2005, Int. Joint Conf. on Neural Networks*, pages 396–399. IEEE–INNS, 2005.
161. J.J. Steil. Backpropagation-Decorrelation: Online recurrent learning with O(n) complexity. In *IJCNN'2004, Int. Joint Conf. on Neural Networks*, volume 1, pages 843–848. IEEE–INNS, 2004.
162. R.B. Stein. A theoretical analysis of neuronal variability. *Biophys. J.*, 5:173–194, 1965.
163. F. Tenore. Prototyping neural networks for legged locomotion using custom aVLSI chips. *The Neuromorphic Engineer*, 1(2):4, 8, 2004.
164. S. Thorpe, A. Delorme, and R. Van Rullen. Spike-based strategies for rapid processing. *Neural Networks*, 14:715–725, 2001.

165. S. Thorpe, D. Fize, and C. Marlot. Speed of processing in the human visual system. *Nature*, 381(6582):520–522, 1996.
166. T. Toyoizumi, J.-P. Pfister, K. Aihara, and W. Gerstner. Generalized Bienenstock-Cooper-Munro rule for spiking neurons that maximizes information transmission. *Proc. Natl. Acad. Sci. USA*, 102(14):5239–5244, 2005.
167. T. Toyoizumi, J.-P. Pfister, K. Aihara, and W. Gerstner. Spike-timing dependent plasticity and mutual information maximization for a spiking neuron model. In L.K. Saul, Y. Weiss, and L. Bottou, editors, *NIPS*2004, Advances in Neural Information Processing Systems*, volume 17, pages 1409–1416. MIT Press, 2005.
168. A.M. Turing. Systems of logic based on ordinals. *Proceedings of the London Mathematical Society*, 45(2):161–228, 1939.
169. A.M. Turing. Computing machinery and intelligence. *Mind*, 59:433–460, 1950.
170. A. Upegui, C.A. Peña Reyes, and E. Sanchez. An FPGA platform for on-line topology exploration of spiking neural networks. *Microprocessors and Microsystems*, 29:211–223, 2004.
171. L.G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.
172. M. van Hulle. *Faithful Representations and topographic maps: From distortion- to information-based self-organization*. New York: Wiley, 2000.
173. R. Van Rullen and S. Thorpe. Rate coding versus temporal order coding: what the retinal ganglion cells tell the visual cortex. *Neural Computation*, 13:1255–1283, 2001.
174. V.N. Vapnik. *Statistical learning theory*. Wiley, 1998.
175. D. Verstraeten, B. Schrauwen, M. D'Haene, and D. Stroobandt. An experimental unification of reservoir computing methods. *Neural Networks*, 20(3):391–403, 2007.
176. D. Verstraeten, B. Schrauwen, and D. Stroobandt. Isolated word recognition using a liquid state machine. In *ESANN'05, European Symposium on Artificial Neural Network*, pages 435–440, 2005.
177. T. Viéville and S. Crahay. Using an hebbian learning rule for multi-class SVM classifers. *J. of Computational Neuroscience*, 17(3):271–287, 2004.
178. M. Volkmer. A pulsed neural network model of spectro-temporal receptive fields and population coding in auditory cortex. *Natural Computing*, 3:177–193, 2004.
179. G. Wang and M. Pavel. A spiking neuron representation of auditory signals. In *IJCNN'2005, Int. Joint Conf. on Neural Networks*, pages 416–421. IEEE–INNS, 2005.
180. D. Watts and S. Strogatz. Collective dynamics of "small-worl" networks. *Nature*, 393:440–442, 1998.
181. L. Watts. Event-driven simulation of networks of spiking neurons. In J. D. Cowan, G. Tesauro, and J. Alspector, editors, *NIPS:1993, Advances in Neural Information Processing System*, volume 6, pages 927–934. MIT Press, 1994.
182. T. Wennekers, F. Sommer, and A. Aertsen. Editorial: Cell assemblies. *Theory in Biosciences (special issue)*, 122:1–4, 2003.
183. S. Wu, D. Chen, M. Niranjan, and S. Amari. Sequential bayesian decoding with a population of neurons. *Neural Computation*, 15:993–1012, 2003.
184. S. G. Wysoski, L. Benuskova, and N. Kasabov. Fast and adaptive network of spiking neurons for multi-view visual pattern recognition. *Neurocomputing*, 71(13-15):2563–2575, 2008.
185. X. Xie and H.S. Seung. Learning in neural networks by reinforcement of irregular spiking. *Physical Review E*, 69(041909), 2004.
186. J. Xin and M. J. Embrechts. Supervised learning with spiking neuron networks. In *Proceedings of the IJCNN 2001 IEEE International Joint Conference on Neural Networks, Washington DC*, volume 3, pages 1772–1777, 2001.
187. A.M. Zador and B.A. Pearlmutter. VC dimension of an integrate-and-fire neuron model. *Neural Computation*, 8(3):611–624, 1996.
188. R. S. Zemel, P. Dayan, and A. Pouget. Probabilistic interpretation of population codes. *Neural Computation*, 10:403–430, 1998.