10 Computing with Spiking Neuron Networks

Hélène Paugam-Moisy[1,2] · *Sander Bohte*[3]
[1]Laboratoire LIRIS – CNRS, Université Lumière Lyon 2, Lyon, France
[2]INRIA Saclay – lle-de-France, Université Paris-Sud, Orsay, France
helene.paugam-moisy@univ-lyon2.fr
hpaugam@lri.fr
[3]Research Group Life Sciences, CWI, Amsterdam, The Netherlands
s.m.bohte@cwi.nl

Abstract

Spiking Neuron Networks (SNNs) are often referred to as the third generation of neural networks. Highly inspired by 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 to memorize and a strong ability to do 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 ➲ Sect. 1 and summarizes the most currently-in-use models of neurons and synaptic plasticity in ➲ Sect. 2. The computational power of SNNs is addressed in ➲ Sect. 3 and the problem of learning in networks of spiking neurons is tackled in ➲ Sect. 4, with insights into the tracks currently explored for solving it. Finally, ➲ Sect. 5 discusses application domains, implementation issues and proposes several simulation frameworks.

1 From Natural Computing to Artificial Neural Networks

1.1 Traditional Neural Networks

Since the human brain is made up of a great many 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 and Pitts (1943) 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 (➲ *Fig. 1*, where index j has been omitted).

Subsequent neuronal models evolved where inputs and outputs were real-valued, and the nonlinear threshold function (Perceptron) was replaced by a linear input–output mapping (Adaline) or nonlinear functions like the sigmoid (Multilayer Perceptron). Alternatively, several connectionist models (e.g., RBF networks, Kohonen self-organizing maps (Kohonen 1982; Van Hulle 2000)) 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 $< X, W >$ (➲ *Fig. 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.

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

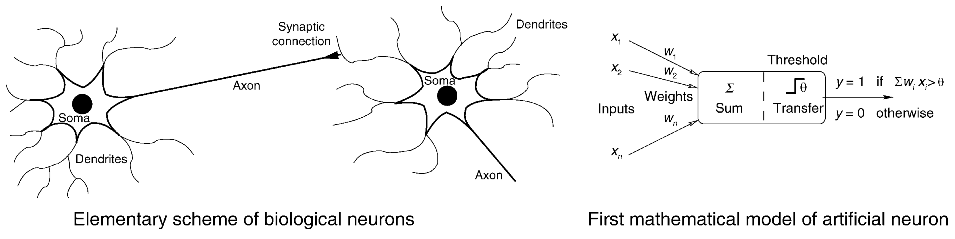
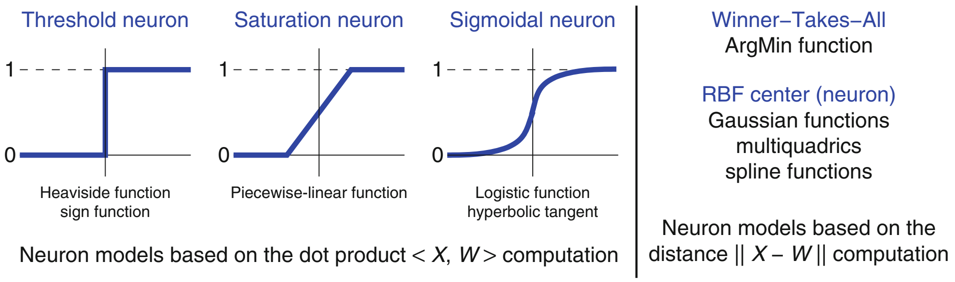


Fig. 2
Several variants of neuron models, based on a dot product or a distance computation, with different transfer functions.



A large number of learning rules have been proposed, both for teaching a network explicitly to perform some task (supervised learning), and for learning interesting features "on its own" (unsupervised learning). Supervised learning algorithms include gradient descent algorithms (e.g., error backpropagation) (Rumelhart et al. 1986) 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 (1949), 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 by this type of natural neural processing are referred to as Hebbian rules (e.g., in Hopfield's (1982) network model).

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

- Calculability: NNs computational power outperforms a Turing machine (Siegelmann 1999).
- Complexity: The "loading problem" is NP-complete (Blum and Rivest 1989; Judd 1990).

- Capacity: Multilayer Perceptrons (MLP), Radial Basis Function (RBF) network, and Wavelet Neural Networks (WNN) are universal approximators (Cybenko 1988; Funahashi 1989; Hornik et al. 1989).
- Regularization theory (Poggio and Girosi 1989); Probably Approximately Correct learning (PAC-learning) (Valiant 1984); Statistical learning theory, Vapnik–Chervonenkis dimension (VC-dimension), and Support Vector Machines (SVM) (Vapnik 1998).

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

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 and Pitts designed their model of the 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 (1939, 1950), they thought that complex, "intelligent" behavior could emerge from a very large network of neurons, combining huge numbers of elementary logic gates. History shows that such basic ideas have been very productive, even if effective learning rules for large networks (e.g., backpropagation for MLP) were discovered only at the end of the 1980s, 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 understand how the brain processes information. *Time* has become a central feature in cognitive processing (Abeles 1991). Brain imaging and a host of new technologies (microelectrode, LFP (local field potential) or EEG (electroencephalogram) recordings, fMRI (functional magnetic resonance imaging)) can now record rapid changes in the internal activity of the brain, and help elucidate the relation between the brain activity and the perception of a given stimulus. The current consensus is that cognitive processes are most likely based on the activation of transient assemblies of neurons (see ➋ Sect. 3.2), although the underlying mechanisms are not yet well understood.

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 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, which in turn is linked to the receiving end of other neurons, the dendrites (see ➋ *Fig. 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.

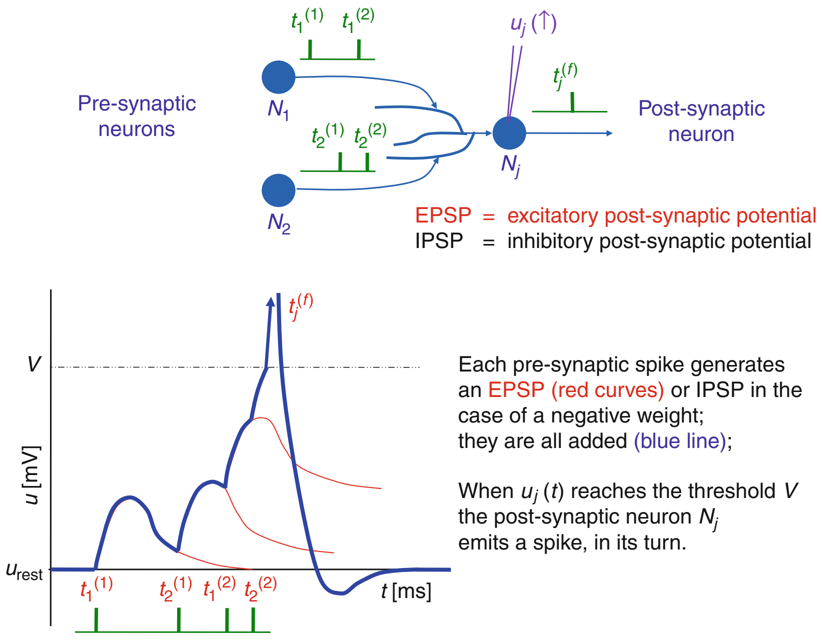
The transient impact a spike has on the neuron's membrane potential is generally referred to as the *postsynaptic potential*, or *PSP*, and the PSP can either inhibit the future firing – inhibitory postsynaptic potential, *IPSP* – or excite the neuron, making it more likely to fire – an excitatory postsynaptic 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 ➔ *Fig. 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 nonlinearly, or even multiply. More detailed accounts can be found in, for example, Maass and Bishop (1999).

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

Fig. 3

A model of spiking neuron: N_j fires a spike whenever the weighted sum of incoming EPSPs generated by its presynaptic neurons reaches a given threshold. The graphic (*bottom*) shows how the membrane potential of N_j varies through time, under the action of the four incoming spikes (*top*).


$t_1^{(1)}$ $t_1^{(2)}$
$u_j(\uparrow)$
Pre-synaptic neurons
N_1
$t_j^{(f)}$
Post-synaptic neuron
$t_2^{(1)}$ $t_2^{(2)}$
N_j
N_2
EPSP = excitatory post-synaptic potential
IPSP = inhibitory post-synaptic potential
$t_j^{(f)}$
V
u [mV]
u_{rest}
$t_1^{(1)}$ $t_2^{(1)}$ $t_1^{(2)}$ $t_2^{(2)}$
t [ms]
Each pre-synaptic spike generates an EPSP (red curves) or IPSP in the case of a negative weight; they are all added (blue line);
When $u_j(t)$ reaches the threshold V the post-synaptic neuron N_j emits a spike, in its turn.

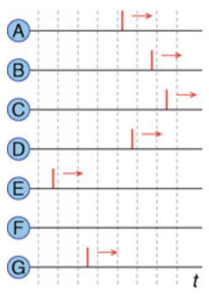
measures such as "firing rate." Rather than the form of the action potential, it is the number and the timing of spikes that matter. In fact, it has been established that *the exact timing of spikes* can be a means for coding information, for instance in the electrosensory system of electric fish (Heiligenberg 1991), in the auditory system of echo-locating bats (Kuwabara and Suga 1993), and in the visual system of flies (Bialek et al. 1991).

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. (1996) and van Rullen and Thorpe (2001) in the context of visual processing. Many physiologists subscribe to the idea of a Poisson-like rate code to describe the way 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 100–150 ms are sufficient for a human to respond selectively to complex visual stimuli (e.g., faces or food), but due to the feedforward architecture of the visual system, made up of multiple layers of neurons firing at an average rate of 10 ms, 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 (Olshausen 1996). 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 was pointed out by Recce (1999) and analyzed by Thorpe et al. (2001). 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 (➋ *Fig. 4*).

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



Count	Latency	Rank
1	5	3
1	6	5
1	7	6
1	5	4
1	1	1
0	–	–
1	3	2

Numeric examples:	Count code	Binary code	Timing code	Rank order
Left (opposite) figure $n = 7$, $T = 7$ ms	3	7	≈ 19	12.3
Thorpe et al. (2001) $n = 10$, $T = 10$ ms	3.6	10	≈ 33	21.8

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

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 the information-coding capacity. A greater 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, for example for a 1 ms 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, that is 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 the capacity is $n \log(n)$. However, this theoretical estimate must be reevaluated when considering the unavoidable bound on precision required for distinguishing two spike times, even in computer simulation (Cessac et al. 2010).

1.4 Spiking Neuron Networks

In *Spiking Neuron Networks* (SNNs), sometimes referred to as Pulsed-Coupled Neural Networks (PCNNs) in the literature, 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* (sometimes referred to as "latency coding" or "time-to-first-spike"). Temporal coding is a straightforward method for translating a vector of real numbers into a spike train, for example, for simulating traditional connectionist models using SNNs, as in Maass (1997a). The basic idea is biologically well founded: the more intensive the input, the earlier the spike transmission (e.g., in visual systems). 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 patterns $\mathbf{x} = (x_1, \ldots, x_n)$ – with all x_i inside a bounded interval of $\mathbb{R}$, for example $[0, 1]$ – 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 (➋ Fig. 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.

Many SNN approaches focus on the continuous computation that is carried out on such spike trains. Assigning meaning is then less straightforward, and depends 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* (➋ Fig. 6): With time on the abscissa, a small bar is plotted each time a neuron fires a spike (one line per neuron, numbered on the Y-axis). Variations 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

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

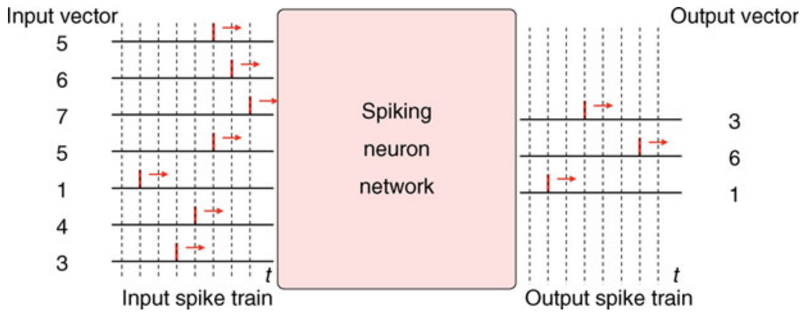
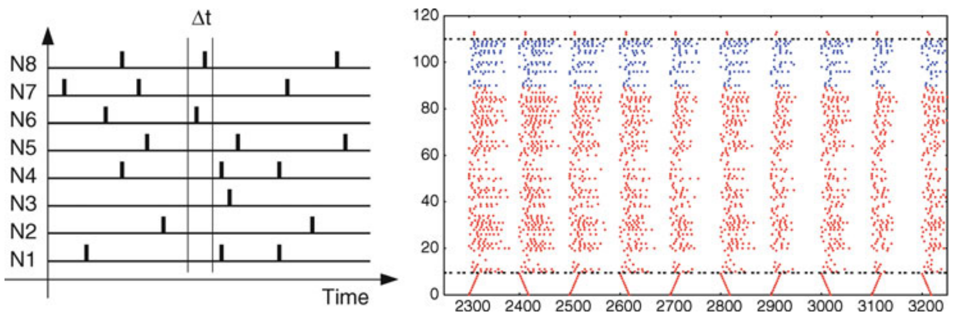


Fig. 6
On a spike raster plot, a small bar is plotted each time (in abscissa) that a neuron (numbered in ordinates) fires a spike. For computational purposes, time is often discretized in temporal Δt units (*left*). The dynamic answer of an SNN, stimulated by an input pattern in temporal coding – diagonal patterns, bottom – can be observed on a spike raster plot (*right*). (From Paugam-Moisy et al. 2008.)



plots drawn from multielectrode 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.

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 Sect. 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, and 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. ➋ Section 4 addresses different methods for learning in SNNs and presents the paradigm of *Reservoir Computing*. Finally, ➋ Sect. 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

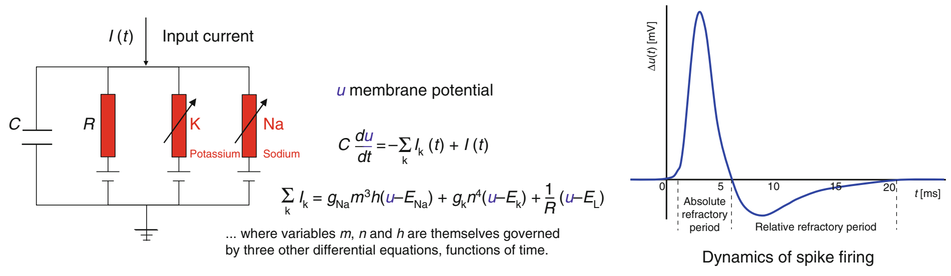
The fathers of the spiking neurons are the conductance-based neuron models, such as the well-known electrical model defined by Hodgkin and Huxley (1952) (➋ *Fig. 7*). Hodgkin and Huxley modeled the electrochemical information transmission of natural neurons with electrical circuits consisting of capacitors 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.

$$C\frac{du}{dt} = -g_{\mathrm{Na}}m^3h(u-E_{\mathrm{Na}}) - g_{\mathrm{K}}n^4(u-E_{\mathrm{K}}) - g_{\mathrm{L}}(u-E_{\mathrm{L}}) + I(t) \tag{1}$$

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

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

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, for example in response to an input current $I(t)$ sent during a small time, at $t < 0$.



I (t)
Input current
C
R
K
Potassium
Na
Sodium
u membrane potential
$C\frac{du}{dt} = -\sum_k I_k(t) + I(t)$
$\sum_k I_k = g_{Na}m^3h(u-E_{Na}) + g_k n^4(u-E_k) + \frac{1}{R}(u-E_L)$
... where variables m, n and h are themselves governed by three other differential equations, functions of time.
Δu(t) [mV]
0
5
10
15
20
t [ms]
Absolute refractory period
Relative refractory period
Dynamics of spike firing

properties of the membrane potential, respecting the behavior 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, that is, the *relative refractory period* (➲ *Fig. 7*).

The *Hodgkin–Huxley model* (HH) is realistic but far too complex for the simulation of SNNs. Although ordinary differential equations (ODE) 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

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

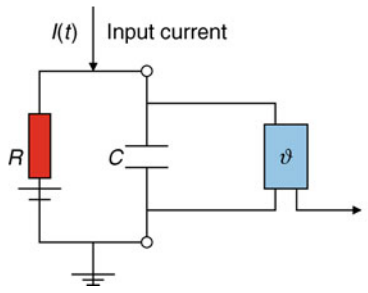
Simpler than the Hodgkin–Huxley neuron model are Integrate-and-Fire (I&F) neuron models, which are much more computationally tractable (see ➲ *Fig. 8* for equation and electrical model).

An important I&F neuron type is the *Leaky-Integrate-and-Fire* (LIF) neuron (Abbott 1999; Stein 1965). 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:

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

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 $u_{\text{rest}} = 0$ as a common assumption). An absolute refractory period can be modeled by forcing the neuron to a value $u = -u_{\text{abs}}$ during a time d_{abs} after a spike emission, and then restarting the integration with initial value $u = u_{\text{rest}}$.

Fig. 8
Electrical circuit and equation of the Integrate-and-Fire model (I&F).



u being the membrane potential,

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

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

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

2.2.2 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 (Brunel and Latham 2003). Via a simple transformation of the membrane potential u to a phase θ, the QIF neuron can be transformed to a Theta-neuron model (Ermentrout and Kopell 1986).

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

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.

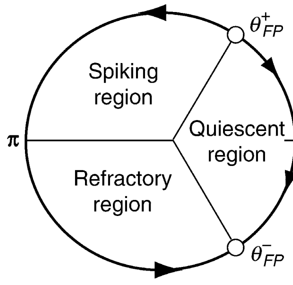
The main advantage of the Theta-neuron model is that neuronal spiking is described in a continuous manner, allowing for more advanced gradient approaches, as illustrated in ➋ Sect. 4.1.

2.2.3 Izhikevich's Neuron Model

In the class of spiking neurons defined by differential equations, the two-dimensional *Izhikevich neuron model* (Izhikevich 2003) is a good compromise between biophysical plausibility and computational cost. It is defined by the coupled equations

$$\frac{du}{dt} = 0.04u(t)^2 + 5u(t) + 140 - w(t) + I(t) \qquad \frac{dw}{dt} = a(bu(t) - w(t)) \tag{4}$$
$$\text{with after-spike resetting:} \qquad \text{if } u \geq \vartheta \text{ then } u \leftarrow c \text{ and } w \leftarrow w + d$$

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 the 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.



This neuron model is capable of reproducing many different firing behaviors that can occur in biological spiking neurons (➋ *Fig. 10*). (An electronic version of the original figure and reproduction permission are freely available at www.izhikevich.com.)

2.2.4 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., Izhikevich (2004) for a comprehensive review, or Standage and Trappenberg (2005) 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 extent – QIF models have been studied extensively due to their low complexity, making them relatively easy to understand. However, as argued by Izhikevich (2004), 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 be able to reproduce only three out of the 20 firing schemes displayed in ➋ *Fig. 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 – for example (K) "resonator" and (L) "integrator" – many such behaviors may be reachable with different parameter choices, for the 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 ➋ *Fig. 10*. Other intermediate models are currently being studied, such as the gIF model (Rudolph and Destexhe 2006).

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 1,200 floating point computations (FLOPS) per 1 ms simulation. Simplified to two differential equations, the Morris–Lecar or FitzHugh–Nagumo models still have a computational cost of one to several hundred FLOPS. Only five FLOPS are required by the LIF model, around 10 FLOPS for variants such as LIF-with-adaptation and quadratic or exponential Integrate-and-Fire neurons, and around 13 FLOPS for Izhikevich's model.

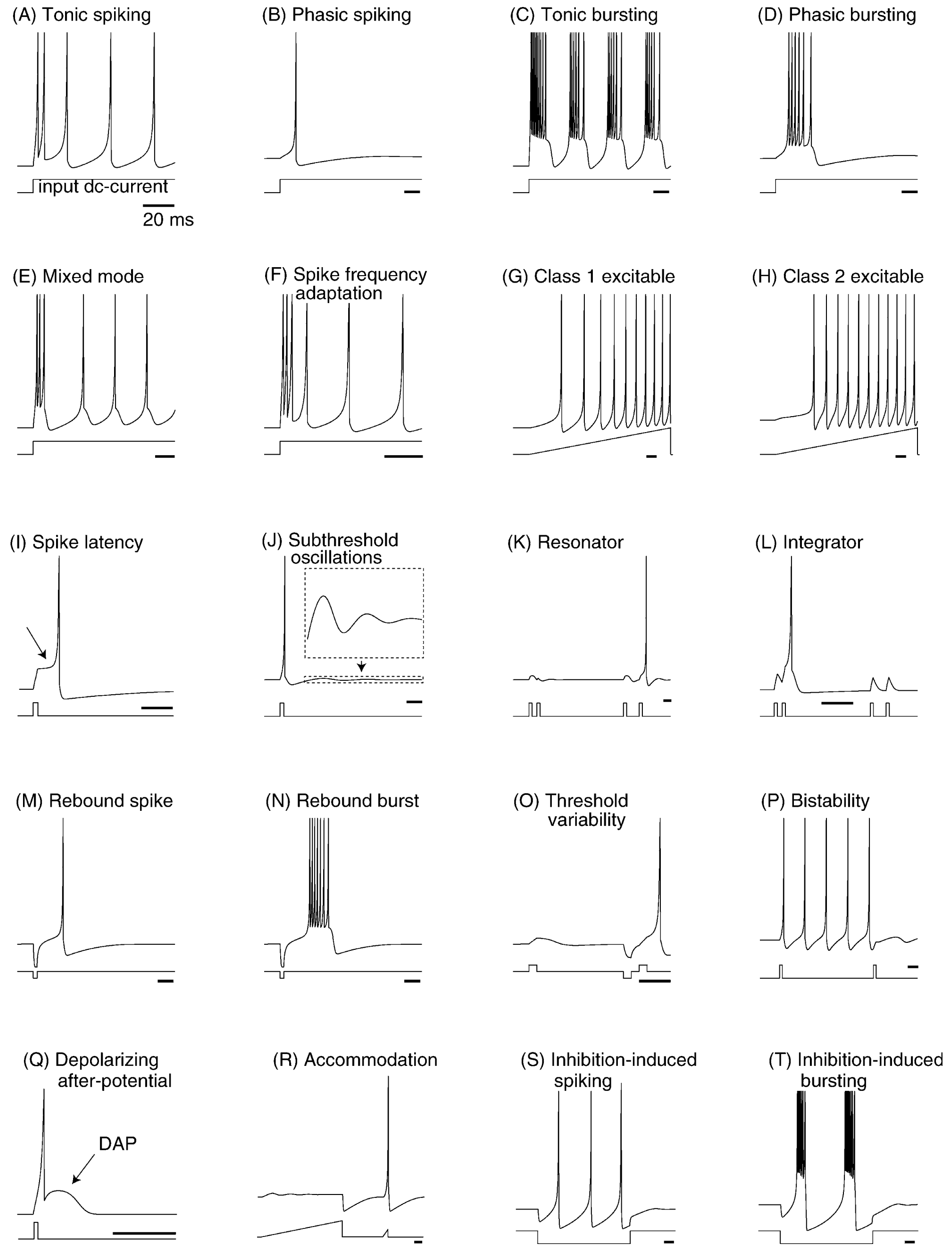
2.3 Spike Response Model

Compared to the neuron models governed by coupled differential equations, the *Spike Response Model* (SRM) as defined by Gerstner (1995) and Kistler et al. (1997) is more intuitive 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 occurrence of spike emissions. Let $\mathscr{F}_j = \left\{t_j^{(f)}; 1 \leq f \leq n\right\} = \left\{t \,|\, u_j(t) = \vartheta \;\wedge\; u_j'(t) > 0\right\}$ denote the set of all firing times of neuron N_j, and $\Gamma_j = \{i | N_i \text{ is presynaptic to } N_j\}$ define its set of presynaptic neurons. The state $u_j(t)$ of neuron N_j at time t is given by

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

Fig. 10

Many firing behaviors 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 [2004].)



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's potential response to presynaptic spikes, and κ_j describes the response of the membrane potential to an external input current (➋ *Fig. 11*). Some common choices for the kernel functions are:

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

or, somewhat more involved,

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

where $\mathcal{H}$ is the Heaviside function, ϑ is the threshold and τ a time constant, for neuron N_j. Setting $K \rightarrow \infty$ 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. (An α-function is like $\alpha(x) = x\exp^{-x}$.)

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

or, in a more general description,

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

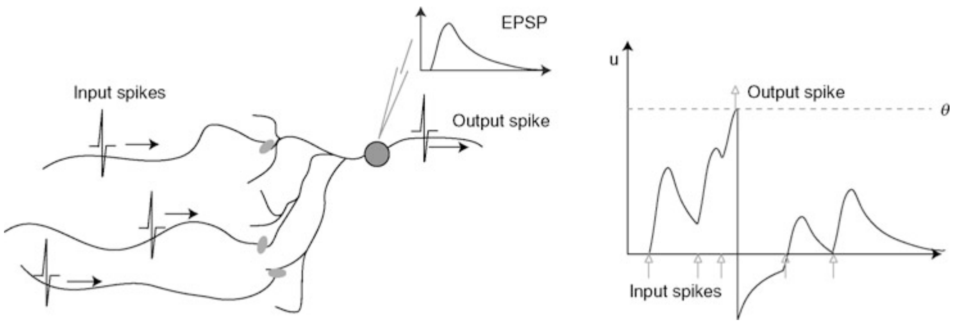
where τ_m and τ_s are time constants, and d_{ij}^{ax} describes the axonal transmission delay.

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 (➋ Eq. 5) by a

◘ Fig. 11

The Spike Response Model (SRM) is a generic framework to describe the spike process. (Redrawn after Gerstner and Kistler 2002b.)



single contribution. Moreover, integrating the equation on a small time window of 1 ms and assuming that each presynaptic neuron fires at most once in the time window (reasonable because of the refractoriness of presynaptic neurons) reduces the SRM to the simplified SRM_0 model:

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

Despite its simplicity, the SRM 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 is the 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 (Hebb 1949) (see ➋ Sect. 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 to 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. On the timescale of seconds or minutes, the weight changes are denoted as Short-Term Potentiation (STP) and Short-Term Depression (STD). Abbott and Nelson (2000) give a good review of the main synaptic plasticity mechanisms for regulating levels of activity in conjunction with Hebbian synaptic modification, for example, redistribution of synaptic efficacy (Markram and Tsodyks 1996) 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 (Markram et al. 1997; Bi and Poo 1998; Kempter et al. 1999).

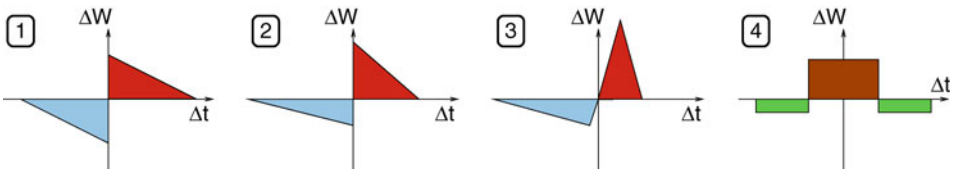
One important finding that is receiving increasing attention is *Spike-Timing Dependent Plasticity*, STDP, as discovered in neuroscientific studies (Markram et al. 1997; Kempter et al. 1999), especially in detailed experiments performed by Bi and Poo (1998, 2001). Often referred to as a *temporal Hebbian 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 a decrease in 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 (Markram et al. 1997; Kempter et al. 1999; Song et al. 2000; Senn et al. 2001; Buchs and Senn 2002; Izhikevich et al. 2004; Kistler 2002; Gerstner and Kistler 2002a; Nowotny et al. 2003; Izhikerich and Desai 2003; Saudargiene et al. 2004; Meunier and Paugam-Moisy 2005; Mouraud and Paugam-Moisy 2006): They are smooth versions of the shapes schematized by polygons in ➲ *Fig. 12*. The spike timing (X-axis) is the difference $\Delta t = t_{\text{post}} - t_{\text{pre}}$ 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, that is when $\Delta t > 0$ and is close to zero (windows 1–3 in ➲ *Fig. 12*), and decreased otherwise. The main differences in shapes 1–3 concern the symmetry or asymmetry of the LTP and LTD subwindows, and the discontinuity or not of the 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 $t_{\text{post}} - t_{\text{pre}}$ (window 4 in ➲ *Fig. 12*).

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, for example 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 are down) or have "epileptic" network activity (all weights are 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. (2003), for a smooth version of the temporal window 3 of ➲ *Fig. 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 the update rule and the shape of STDP windows, the reader could refer to Song et al. (2000), Rubin et al. (2000), and Câteau and Fukai (2003).

Fig. 12
Various shapes of STDP windows with LTP in blue and LTD in red for excitatory connections (1–3). 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) toward $\pm\infty$. Standard Hebbian rule (window 4) with brown LTP and green LTD are usually applied to inhibitory connections.



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 (Thorpe et al. 1996), 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, different approaches to determining the computational power and complexity of SNNs are described, and current thinking on how to exploit these properties is outlined, in particular with regard to dynamic cell assemblies.

In 1997, Maass (1997b, 2001) proposed to classify neural networks as follows:

- *1st generation:* Networks based on McCulloch and Pitts' neurons as computational units, that is, threshold gates, with only digital outputs (e.g., Perceptron, 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 and 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 very large-scale integrated (VLSI) circuits, new types of electronic solutions for encoding analog variables using 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 discussed in the following sections.

3.1 Complexity and Learnability Results

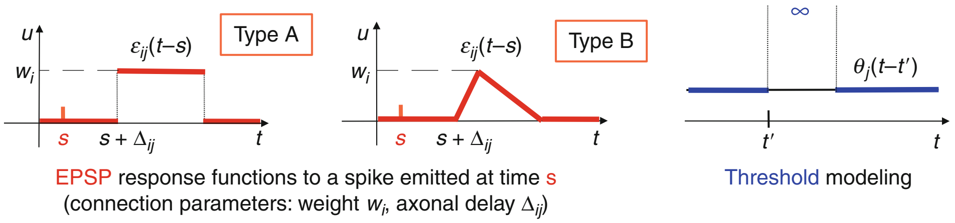
3.1.1 Tractability

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" (➲ *Fig. 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.

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 (Maass 1997b; Schmitt 1998). Since spiking neurons can behave as coincidence detectors (for a proper choice of weights, a spiking neuron can only fire when two or more input spikes are effectively coincident in time) it is

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 Maass (1997b).



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}):

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

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:

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

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 Sect. 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" (*Fig. 13*) has been proposed, as its triangular EPSP can shift the firing time of a targeted postsynaptic 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 1st and the 2nd generations of neural networks.

Schmitt develops a deeper study of type_A neurons with programmable delays in Schmitt (1998) and Maass and Schmitt (1997). 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 $n-1$.

3.1.2 Capacity

Maass (2001) considers *noisy spiking neurons*, a neuron model close to the SRM (cf. Sect. 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 Maass (2001) 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 $F : [0, 1]^n \rightarrow [0, 1]^k$ 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 et al. (2005) studied STDP learnability. They define a *Spiking Neuron Convergence Conjecture* (SNCC) and compare the behavior 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 (2001) 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.

3.1.3 VC-Dimension

The first attempt to estimate the VC-dimension (see http://en.wikipedia.org/wiki/VC_dimension for a definition) of spiking neurons is probably the work of Zador and Pearlmutter (1996), where they studied a family of integrate-and-fire neurons (cf. Sect. 2.2) with threshold and time constants as parameters. Zador and Pearlmutter proved that for an Integrate-and-Fire (I&F) model, VC_{dim}(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 (Maass and Schmitt 1997; Maass 2001) 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}(type_A neuron) is $\Omega(m \log(m))$ – even with fixed weights – whereas, with m variable weights, VC_{dim}(threshold gate) is $\Omega(m)$.
- With n real-valued inputs and a binary output, VC_{dim}(type_A neuron) is $O(n \log(n))$.
- With n real-valued inputs and a real-valued output, $pseudo_{\text{dim}}$(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 and Schmitt (1999) 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.

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 $\mathcal{S}_n^{xy}$, where x is b for Boolean values and a for analog (real) values and idem for y. Then the following holds:

- The classes $\mathcal{S}_n^{bb}$ and $\mathcal{S}_n^{ab}$ have VC-dimension $\Theta(n \log(n))$.
- The class $\mathcal{S}_n^{aa}$ 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 $\mathcal{N}$ 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 $\mathcal{N}$ directed acyclic graph. (The directed acyclic graph is the network topology that underlies the spiking neuron network dynamics.) Then $VC_{\text{dim}}(\mathcal{N})$ is $O(E^2)$, even for analog coding of the inputs (Maass and Schmitt 1999). Schmitt (2004) derived more precise results by considering a feedforward architecture of depth D, with nonlinear synaptic interactions between neurons.

It follows that the sample sizes required for the networks of fixed depth are not significantly larger than 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 (Schmitt 2004).

3.1.4 Loading Problem

In the framework of PAC-learnability (Valiant 1984; Blumer et al. 1989), only hypotheses from $\mathcal{S}_n^{bb}$ 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. Judd 1990):

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

In this PAC-learnability setting, the following results are proved in Maass and Schmitt (1999):

- The *consistency problem* for a spiking neuron with binary delays is *NP*-complete.
- The *consistency problem* for a spiking neuron with binary delays and fixed weights is *NP*-complete.

Several extended results were developed by Šíma and Sgall (2005), such as:

- The *consistency problem* for a spiking neuron with nonnegative delays is *NP*-complete ($d_{ij} \in \mathbb{R}^+$). The result holds even with some restrictions (see Šíma and Sgall (2005) 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.

3.1.5 Complexity Results Versus Real-World Performance

Non-learnability results, such as those outlined above, have of course been derived for classic NNs already, for example in Blum and Rivest (1989) and Judd (1990). 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 (Sporns et al. 2005; Eguíluz 2005; Achard and Bullmore 2007), the complex networks research area appears to provide valuable tools ("Small-Word" connectivity (Watts and Strogatz 1998), presence of clusters (Newman and Girvan 2004; Meunier and Paugam-Moisy 2006), presence of hubs (Barabasi and Albert 1999), etc., see Newman (2003) for a survey) for studying topological and dynamic complexities 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 (Legenstein and Maass 2005; Verstraeten et al. 2007; Schrauwen et al. 2009). 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.

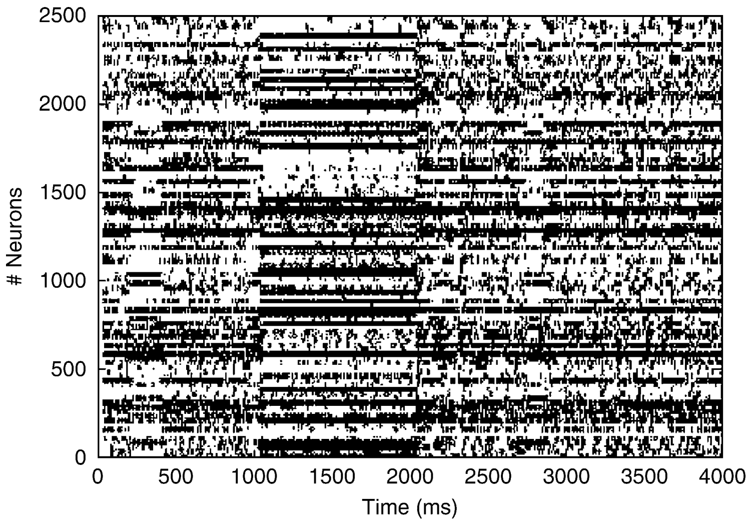
The concept of the *cell assembly* was introduced by Hebb (1949), more than half a century ago. (The word "cell" was used at that time, instead of "neuron.") However, the idea was not further developed, neither by neurobiologists – since they could not record the activity of more than one or a few neurons at a time, until recently – nor by computer scientists. New techniques of brain imaging and recording have boosted this area of research in neuroscience only recently (cf. Wenneker et al. 2003). In computer science, a theoretical analysis of assembly formation in spiking neuron network dynamics (with SRM neurons) has been discussed by Gerstner and van Hemmen (1994), 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, for example, by a Hebb's rule mechanism. Inherited from Hebb, current thinking about cell assemblies is that they could play the role of "grandmother neural groups" as a basis for memory encoding, instead of the old controversial notion of "grandmother cell," and that material entities (e.g., a book, a cup, or a dog) and even more abstract entities such as concepts or ideas could be represented by cell assemblies.

Within this context, synchronization of firing times for subsets of neurons inside a network has received much attention. Abeles (1991) 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 and Brody demonstrated *transient synchrony* as a means for collective spatiotemporal integration in neuronal circuits (Hopfield and Brody 2000, 2001). 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 (➋ *Fig. 14*).

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 (2006) who proposes the extended notion of *polychronization* 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

Fig. 14
A spike raster plot showing the dynamics of an artificial SNN: Erratic background activity is disrupted by a stimulus presented between 1,000 and 2,000 ms. (By courtesy of D. Meunier, reprint from his PhD thesis (2007), Université Lyon 2, France)



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 (LeCun et al. 1995). Similarly, traditional neural networks are preferred tools for function approximation, or regression. The best-known learning rules for achieving such networks 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, two main directions are distinguished. 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 and Natschläger (1997) propose a theoretical model for emulating arbitrary Hopfield networks in temporal coding (see ➋ Sect. 1.4). Maass (1997a) 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 ➋ Sect. 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 and 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 to be an artificial attempt to make SNNs computing like traditional neural networks, without taking advantage of SNNs intrinsic abilities to compute with time.

4.1.1 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 Hopfield's (1995) idea, Natschläger and Ruf (1998a) propose a learning algorithm that performs unsupervised clustering in spiking neuron networks, akin to RBF network, using spike times as input. Natschläger and Ruf's spiking neural network for unsupervised learning is a simple two-layer network of SRM 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 (➋ *Fig. 15*). The delay d^k of a synaptic terminal k is defined by the difference between the firing time of the presynaptic neuron i, and the time the postsynaptic 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 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 (Natschläger and Ruf 1998a).

Bohte et al. (2002b) 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

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 Natschläger and Ruf (1998b).

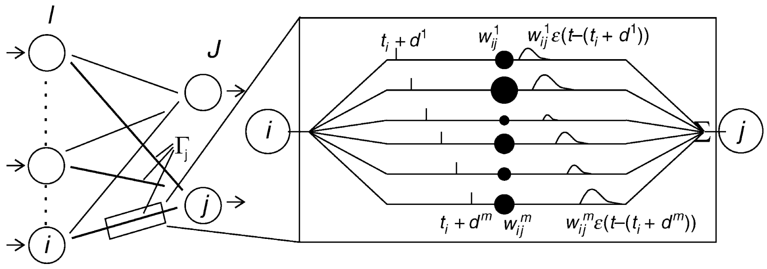
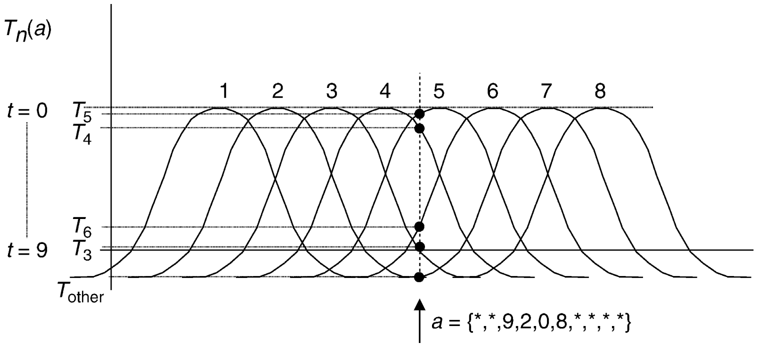

I
J
Γj
i
j
i
$t_i + d^1$
w_{ij}^1
$w_{ij}^1\varepsilon(t-(t_i + d^1))$
Σ
j
$t_i + d^m$
w_{ij}^m
$w_{ij}^m\varepsilon(t-(t_i + d^m))$

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.



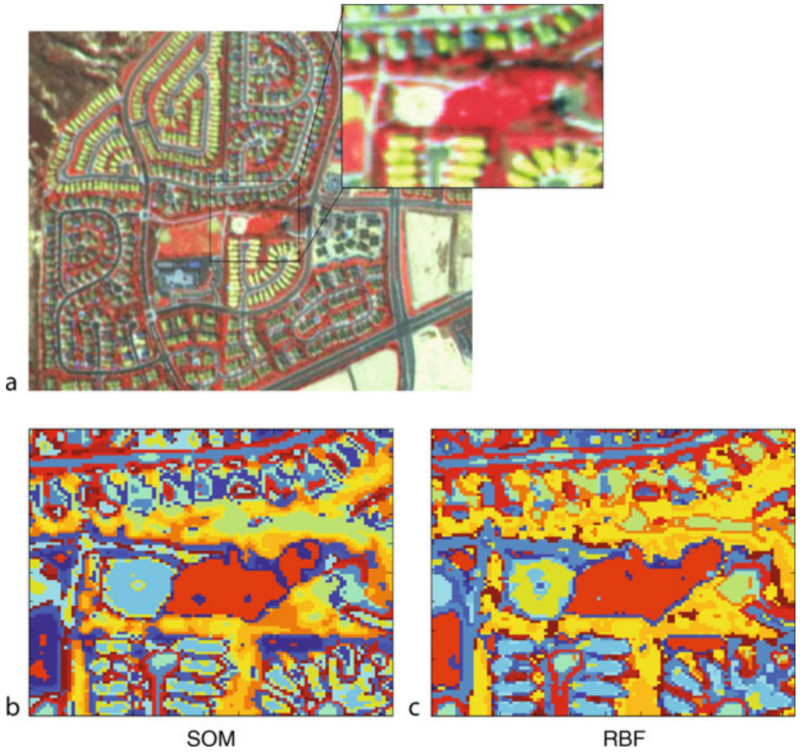
encoded into temporal spike-time patterns by population coding, using multiple local receptive fields like Radial Basis Functions. The translation of inputs into relative firing times is straightforward: An optimally stimulated neuron fires at $t = 0$, whereas a value up to say $t = 9$ is assigned to less optimally stimulated neurons (depicted in *Fig. 16*). With such encoding, spiking neural networks were shown to be effective for clustering tasks, for example *Fig. 17*.

4.1.2 Supervised Learning in Multilayer Networks

A number of approaches for supervised learning in standard multilayer feedforward networks have been developed based on gradient descent methods, the best known being error backpropagation. As developed in Bohte et al. (2002a), *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. (1986), but SpikeProp applies to spiking neurons of the SRM type. To overcome the discontinuous nature of spiking neurons, the thresholding function is

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-Organizing Map (SOM). (c) Spiking Neuron Network RBF classification of the cutout image.



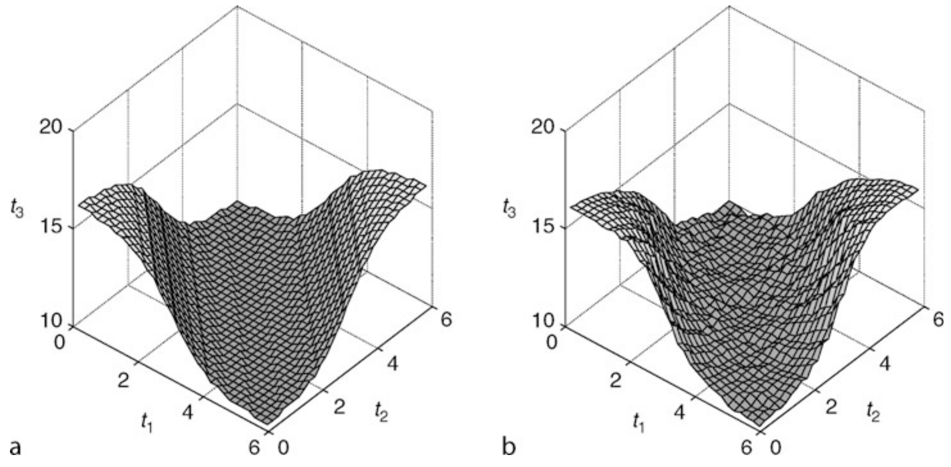
approximated, thus linearizing the model at a neuron's output spike times. As in the unsupervised SNN described above, each connection between neurons may have multiple delayed synapses with varying weights (see *Fig. 15*). The SpikeProp algorithm has been shown to be capable of learning complex nonlinear tasks in spiking neural networks with similar accuracy as traditional sigmoidal neural networks, including the archetypal XOR classification task (*Fig. 18*).

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 (Schrauwen and Van Campenhout 2004a, b). Xin and Embrechts (2001) have further shown that the addition of a simple momentum term significantly speeds up convergence of the SpikeProp algorithm. Booij and Nguyen (2005) 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 et al. (2009) derived a supervised Theta-learning rule for multilayer networks of Theta neurons. By mapping QIF neurons to the canonical Theta-neuron model (a nonlinear phase model, see Sect. 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.

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.



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 neural 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. (2009) 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 Belatreche et al. (2007), 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.

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 that 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 contribute to information processing. At a given time t, the sub-topology defined by active neurons can be very sparse and different from the

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 networks (taken from McKennoch et al. (2009))

Learning method	Network size	Epochs	Train (%)	Test (%)
Fisher Iris Dataset				
SpikeProp	$50 \times 10 \times 3$	1,000	97.4	96.1
BP A	$50 \times 10 \times 3$	2.6e6	98.2	95.5
BP B	$4 \times 8 \times 1$	1e5	98.0	90.0
Theta-Neuron BP	$4 \times 8 \times 1$	1,080	100	98.0
Wisconsin Breast Cancer Dataset				
SpikeProp	$64 \times 15 \times 2$	1,500	97.6	97.0
BP A	$64 \times 15 \times 2$	9.2e6	98.1	96.3
BP B	$9 \times 8 \times 1$	1e5	97.2	99.0
Theta-Neuron BP	$9 \times 8 \times 1$	3,130	98.3	99.0

underlying network architecture (e.g., local clusters, short or long path loops, and synchronized cell assemblies), comparable to the active brain regions that appear colored in brain imaging scanners. Clearly, an SNN architecture has no need to be regular. A network of spiking neurons can even be defined randomly (Maass et al. 2002b; Jaeger 2002) 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 (Meunier and Paugam-Moisy 2005). 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 a unifying term for which the precursor models are Echo State Networks (ESNs) and Liquid State Machines (LSMs). Note that the term "reservoir computing" is not reserved to SNNs, since ESN was first designed with sigmoidal neurons, but this chapter mainly presents reservoir computing with SNNs.

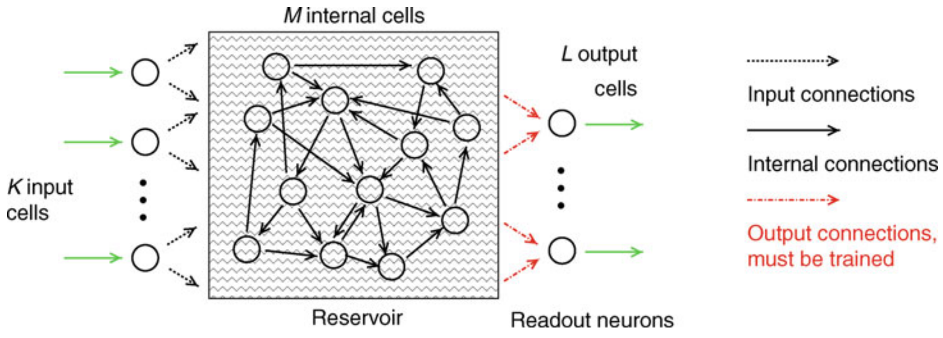
4.2.1 Main Characteristics of Reservoir Computing Models

The topology of a reservoir computing model (➲ *Fig. 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*, which 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 difficulty to find efficient supervised learning rules to train recurrent neural networks, as attested by the limited success of methods like Backpropagation Through Time (BPTT), Real-Time Recurrent Learning (RTRL) or Extended Kalman Filtering (EKF). The difficulty stems from lack of knowledge of how 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 was directly inspired by the observation of natural processing in the brain (see Sect. 2.4), its computation does not require supervised control or an understanding of the network dynamics.

The paradigm of "reservoir computing" is commonly referred to as such since approximately 2007, and it 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 were designed concurrently in the early 2000s, by Jaeger (2001) for the ESN and by Maass et al. (2002b) for the LSM.

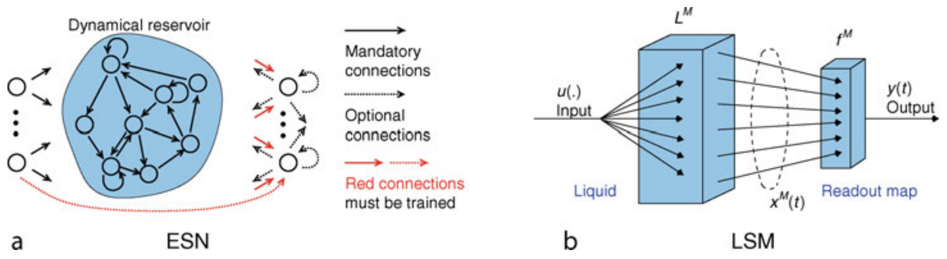
4.2.2 Echo State Network (ESN) and Liquid State Machine (LSM)

The original design of *Echo State Network*, proposed by Jaeger (2001), was intended to learn time series $\mathbf{u}(1), \mathbf{d}(1), \ldots, \mathbf{u}(T)$, and $\mathbf{d}(T)$ 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 *Fig. 20a*) and the network training dynamics is governed by the following equation:

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

Fig. 20

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



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, for example 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 $(W^{\text{out}})^t = M^{\dagger}T$ where $M^{\dagger}$ is the pseudo-inverse of M. In the exploitation phase, the network is driven by novel input sequences $u(t)$, with $t \geq T$ (desired output $d(t)$ is unknown), and it produces the computed output $y(t)$ with coupled equations like:

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

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

For the method to be efficient, the network must have the "Echo State Property," that is, the property of being state contracting, state forgetting, and input forgetting, which gives it a behavior of "fading memory." Choosing a reservoir weight matrix W with a spectral radius $|\lambda_{\max}|$ slightly lower than 1 is neither a necessary nor a sufficient condition (as discussed in Lukoševičius and Jaeger 2009) but can be applied as a rule of thumb for most practical cases: the common practice is to rescale W after randomly initializing the network connections. An important remark must be made: the condition on the spectral radius is no longer clearly relevant when the reservoir is an SNN with fixed weights, and it 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 Verstraeten et al. (2007).

ESNs have been successfully applied in many experimental settings, with networks no larger than 20–400 internal units, for example, in mastering the benchmark task of learning the Mackey–Glass chaotic attractor (Jaeger 2001). Although the first design of ESN was for networks of sigmoid units, Jaeger has also introduced spiking neurons (LIF model) in the ESNs (Jaeger 2002; Jaeger et al. 2007). Results improve substantially over standard ESNs, for example, in the task of generating a slow sinewave ($\mathbf{d}(n) = 1/5\sin(n/100)$), which becomes easy with a leaky integrator network (Jaeger 2002).

The basic motivation of the *Liquid State Machine*, defined by Maass et al. (2002b), was to explain how a continuous stream of inputs $u(\cdot)$ from a rapidly changing environment can be processed in real time by recurrent circuits of Integrate-and-Fire neurons (➲ *Fig. 20b*). The solution they propose is to build a "liquid filter" 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 an output, $y(t)$, that can appear to be 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 the 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 (Auer et al. 2008).

Often, in implementation, the neural network playing the role of the liquid filter is inspired by 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 the neurons.

The readout map is commonly task-specific. However, the hallmark of neural microcircuits is their ability to carry out several parallel real-time computations 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 nonlinear 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.

4.2.3 Related Reservoir Computing Work

An additional work that was linked to the family of "reservoir computing" models after being published is the Backpropagation DeCorrelation rule (BPDC), proposed by Steil (2004). As an extension of the Atiya–Parlos learning rule in recurrent neural networks (Atiya and Parlos 2000), 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. (2008), takes advantage of the theoretical results proving the importance of delays in computing with spiking neurons (see ➲ Sect. 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 it can be observed that polychronous groups (see ➲ Sect. 3.2) are activated more and more selectively as training goes on. The learning rule of the readout delays is based on a temporal margin criterion inspired by Vapnik's theory.

There exist reservoir computing networks that make use of evolutionary computation for training the weights of the reservoir, such as Evolino (Schmidhuber et al. 2007), and several other models that are currently proposed, with or without spiking neurons (Devert et al. 2007; Jiang et al. 2008a, b). Although the research area is in rapid expansion, several papers (Verstraeten et al. 2007; Schrauwen et al. 2007b; Lukoševičius and Jaeger 2009) offer valuable surveys.

4.3 Other SNN Research Tracks

Besides trying 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, being 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 (Barber 2003; Chechik 2003; Pfister et al. 2003, 2006; Bell and Parra 2005; Toyoizumi et al. 2005a, b; Pfister and Gerstner 2006; Bohte and Mozer 2007; Büsing and Maass 2008). The aim of this work on theoretically principled learning is, typically, to arrive at easily understood methods that have spiking neurons carry out some form of Independent Component Analysis (ICA) (Toyoizumi et al. 2005a; Klampfl et al. 2009), or Principal Component Analysis (PCA) (Büsing and Maass 2008), or focus on sparse efficient coding (Olshausen 1996; Volkmer 2004; Lewicki 2002; Smith 2006).

These methods have variable applicability to real world problems, though some have demonstrated excellent performance: Smith and Lewicki (2006) develop an efficient encoding of auditory signals 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 (Xie and Seung 2001; Izhikevich 2007; Legenstein et al. 2008). Many of these algorithms are highly technical and much of this research is still converging can practical algorithms. We only mention these directions here, the reader can pursue the state of the art in these areas.

Just as advances in neurosciences have contributed to the reevaluation 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 (Koerding and Wolpert 2004; Daw and Courvillie 2008). 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 networks is somehow related to 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 (Barber et al. 2005; Zemel et al. 1998; Sahani and Dayar 2003; Wu et al. 2003; Rao 2005; Gerwinn 2007; Hays et al. 2007; Ma et al. 2008). As noted by Deneve (2008a), 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 Deneve (2008a, b), 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 temporal dynamics. Interestingly, in this model neurons still exhibit a Poisson-like distribution of synaptic events. However, rather than reflecting stochasticity due to a 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 is 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 computers, and autonomous robotics.

5.1 Pattern Recognition with SNNs

Considerable work has focused on developing SNNs that are suitable for speech processing (Verstraeten et al. 2005; Holmberg et al. 2005; Wang and Parel 2005; Loiselle et al. 2005). Verstraeten et al. (2005) have developed 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. (2005) developed a method for automatic speech recognition grounded in SNNs. As a "front end," they simulated a part of the inner ear, and then simulated octopus spiking neurons to encode the inner-ear signal in a spike train. They subsequently used a fairly simple classifier to recognize speech from both the inner-ear simulation and the spiking neuron spike trains. Wang and Pavel (2005) used an SNN to represent auditory signals based on using the properties of the spiking neuron refractory period. In their SNN, they converted amplitude to temporal code while maintaining the phase information of the carrier. They proposed 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. (2005). 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 (Thorpe and Gautrais 1997); sparse image coding using an asynchronous spiking neural network (Perrinet and Samuelides 2002); a synchronization-based dynamic vision model for image segmentation (Azhar 2005); saliency extraction with a distributed spiking neural network (Chevallier et al. 2006; Masquelier et al. 2007; Chavallier and Tarroux 2008); and SNNs applied to character recognition (Wysoski et al. 2008).

SNN-based systems also develop increasingly in the area of robotics, where fast processing is a key issue (Maass et al. 2002a; Tenore 2004; Floreano et al. 2005, 2006; Panchev and Wermter 2006; Hartland and Bredeche 2007), from wheels to wings, or legged locomotion. The special abilities of SNNs for fast computing transient temporal patterns make them the first choice 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., Salmen and Plöger 2005), prediction in the context of wireless telecommunications (e.g., Jaeger and Haas 2004), or neuromorphic approaches to rehabilitation in medicine (e.g., Kutch 2004). In this context, the *Neuromorphic Engineer* newsletter often publishes articles on applications developed with SNNs (Institute of Neuromorphic Engineering newsletter: http://www.ine-web.org/).

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 audiovisual speech recognition was proposed by Séguier and Mercier (2002). Crépet et al. (2000) developed, with traditional NNs, a modular connectionist model of multimodal associative memory including temporal aspects of visual and auditory data processing (Bouchut et al. 2003). Such a multimodal framework, applied to a virtual robotic prey–predator environment, with spiking neuron networks as functional modules, has proved capable of simulating high-level natural behavior such as cross-modal priming (Meunier and Paugam-Moisy 2004) or real-time perceptive adaptation to changing environments (Chevallier et al. 2005).

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 purposes has to be revised also. The main interest of SNN simulation is to take into account the precise timing of the spike firing, hence the width of the time window used for discrete computation of the successive network states must remain narrow (see ➲ Sect. 1.4), and consequently only a few spike events occur at each time step: In ➲ *Fig. 6*, only two spikes were fired inside the Δt time range, among the 64 potential connections linking the eight 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 (Watts 1994; Mattia and Del Giudice 2000; Makino 2003; Rochel and Martinez 2003; Reutimann et al. 2003; McKennoch 2009), as long as the activity of an SNN can be fully described by a set of dated spikes. Nevertheless, event-driven programming that requires the next spike time can be explicitly computed in reasonable time, so that not all models of neurons can be used.

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 the membrane potential (e.g., for an SRM or LIF model neuron) is more complex than computing a weighted sum (e.g., for the threshold unit). Therefore, a communication time and computation cost are much better balanced in SNN parallel implementation as compared to traditional NNs, as proved by the parallel implementation of the *SpikeNET* software (Delorme 1999).

Well-known simulators of spiking neurons include *GENESIS* (Bower and Beeman 1998) and *NEURON* (Hines and Carnevale 1997), but they were 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 (Hines and Carnevale 2004) and a version for parallel machines on the other hand (Hines and Carnevale 2008). *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 (Roche and Mortinez 2003). *DAMNED* is a parallel event-driven simulator (Mouraud and Puzenat 2009). A comparative and experimental study of several SNN simulators can be found in Brette et al. (2007). 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 http://snn.elis.ugent.be/node/59 and briefly presented in the last section of Verstraeten (2007). 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 the book by Maass and Bishop (1999) are dedicated to this subject, and more recent work can be found in Upegui et al. (2004), Hellmich et al. (2005), Johnston et al. (2005), Chicca et al. (2003), Oster et al. (2005), Mitra et al. (2006), and Schrauwen et al. (2007a).

5.3 Conclusion

This chapter has given an overview of the 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

Abbott LF (1999) Brain Res Bull 50(5/6):303–304

Abbott LF, Nelson SB (2000) Synaptic plasticity: taming the beast. Nat Neurosci 3:1178–1183

Abeles M (1991) Corticonics: neural circuits of the cerebral cortex. Cambridge University Press, Cambridge

Achard S, Bullmore E (2007) Efficiency and cost of economical brain functional networks. PLoS Comput Biol 3(2):e17

Atiya A, Parlos AG (2000) New results on recurrent network training: unifying the algorithms and accelerating convergence. IEEE Trans Neural Netw 11(3): 697–709

Auer P, Burgsteiner H, Maass W (2008) A learning rule for very simple universal approximators consisting of a single layer of perceptrons. Neural Netw 21(5): 786–795

Azhar H, Iftekharuddin K, Kozma R (2005) A chaos synchronization-based dynamic vision model for image segmentation. In: IJCNN 2005, International joint conference on neural networks. IEEE–INNS, Montreal, pp 3075–3080

Barabasi AL, Albert R (1999) Emergence of scaling in random networks. Science 286(5439):509–512

Barber D (2003) Learning in spiking neural assemblies. In: Becker S, Thrun S, Obermayer K (eds) NIPS 2002, Advances in neural information processing systems, vol 15. MIT Press, Cambridge, MA, pp 165–172

Barber MJ, Clark JW, Anderson CH (2005) Neural representation of probabilistic information, vol 15. MIT Press, Cambridge, MA

Belatreche A, Maguire LP, McGinnity M (2007) Advances in design and application of spiking neural networks. Soft Comput-A Fusion Found Methodol Appl 11:239–248

Bell A, Parra L (2005) Maximising information yields spike timing dependent plasticity. In: Saul LK, Weiss Y, Bottou L (eds) NIPS 2004, Advances in neural information processing systems, vol 17. MIT Press, Cambridge, MA, pp 121–128

Bi G-q, Poo M-m (1998) Synaptic modification in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and polysynaptic cell type. J Neurosci 18(24):10464–10472

Bi G-q, Poo M-m (2001) Synaptic modification of correlated activity: Hebb's postulate revisited. Annu Rev Neurosci 24:139–166

Bialek W, Rieke F, de Ruyter R, van Steveninck RR, Warland D (1991) Reading a neural code. Science 252:1854–1857

Blum A, Rivest R (1989) Training a 3-node neural net is NP-complete. In: Proceedings of NIPS 1988, advances in neural information processing systems. MIT Press, Cambridge, MA, pp 494–501

Blumer A, Ehrenfeucht A, Haussler D, Warmuth MK (1989) Learnability and the Vapnik-Chervonenkis dimension. J ACM 36(4):929–965

Bobrowski O, Meir R, Shoham S, Eldar YC (2007) A neural network implementing optimal state estimation based on dynamic spike train decoding. In: Schölkopf B, Platt JC, Hoffman T (eds) NIPS 2006, Advances in neural information processing systems, vol 20. MIT Press, Cambridge, MA, pp 145–152

Bohte SM, Mozer MC (2007) Reducing the variability of neural responses: a computational theory of spike-timing-dependent plasticity. Neural Comput 19:371–403

Bohte SM, Kok JN, La Poutre H (2002a) Spike-prop: error-backpropagation in multi-layer networks of spiking neurons. Neurocomputing 48:17–37

Bohte SM, La Poutre H, Kok JN (2002b) Unsupervised clustering with spiking neurons by sparse temporal coding and multilayer RBF networks. IEEE Trans Neural Netw 13:426–435

Booij O, tat Nguyen H (2005) A gradient descent rule for spiking neurons emitting multiple spikes. Info Process Lett 95:552–558

Bouchut Y, Paugam-Moisy H, Puzenat D (2003) Asynchrony in a distributed modular neural network for multimodal integration. In: PDCS 2003, International conference on parallel and distributed computing and systems. ACTA Press, Calgary, pp 588–593

Bower JM, Beeman D (1998) The book of GENESIS: exploring realistic neural models with the General Simulation system, 2nd edn. Springer, New York

Brette R, Rudolph M, Hines T, Beeman D, Bower JM et al. (2007) Simulation of networks of spiking neurons: a review of tools and strategies. J Comput Neurosci 23(3):349–398

Brunel N, Latham PE (2003) Firing rate of the noisy quadratic integrate-and-fire neuron, vol 15. MIT Press, Cambridge

Buchs NJ, Senn W (2002) Spike-based synaptic plasticity and the emergence of direction selective simple cells: simulation results. J Comput Neurosci 13:167–186

Büsing L, Maass W (2008) Simplified rules and theoretical analysis for information bottleneck optimization and PCA with spiking neurons. In: NIPS 2007, Advances in neural information processing systems, vol 20. MIT Press, Cambridge

Câteau H, Fukai T (2003) A stochastic method to predict the consequence of arbitrary forms of spike-timing-dependent plasticity. Neural Comput 15(3):597–620

Cessac B, Paugam-Moisy H, Viéville T (2010) Overview of facts and issues about neural coding by spikes. J Physiol (Paris) 104:5–18

Chechik G (2003) Spike-timing dependent plasticity and relevant mutual information maximization. Neural Comput 15(7):1481–1510

Chevallier S, Tarroux P (2008) Covert attention with a spiking neural network. In: ICVS'08, Computer Vision Systems, Santorini, 2008. Lecture notes in computer science, vol 5008. Springer, Heidelberg, pp 56–65

Chevallier S, Paugam-Moisy H, Lemaître F (2005) Distributed processing for modelling real-time multimodal perception in a virtual robot. In: PDCN 2005, International conference on parallel and distributed computing and networks. ACTA Press, Calgary, pp 393–398

Chevallier S, Tarroux P, Paugam-Moisy H (2006) Saliency extraction with a distributed spiking neuron network. In: Verleysen M (ed) ESANN'06, Advances in computational intelligence and learning. D-Side Publishing, Evere, Belgium, pp 209–214

Chicca E, Badoni D, Dante V, d'Andreagiovanni M, Salina G, Carota L, Fusi S, Del Giudice P (2003) A VLSI recurrent network of integrate-and-fire neurons connected by plastic synapses with long-term memory. IEEE Trans Neural Netw 14 (5):1297–1307

Crépet A, Paugam-Moisy H, Reynaud E, Puzenat D (2000) A modular neural model for binding several modalities. In: Arabnia HR (ed) IC-AI 2000, International conference on artificial intelligence. CSREA Press, Las Vegas, pp 921–928

Cybenko G (1988) Approximation by superpositions of a sigmoidal function. Math Control Signal Syst 2:303–314

Daw ND, Courville AC (2008) The pigeon as particle filter. In: NIPS 2007, Advances in neural information processing systems, vol 20. MIT Press, Cambridge, MA

Delorme A, Gautrais J, Van Rullen R, Thorpe S (1999) SpikeNET: a simulator for modeling large networks of integrate and fire neurons. Neurocomputing 26–27:989–996

Deneve S (2008a) Bayesian spiking neurons. I. Inference. Neural Comput 20:91–117

Deneve S (2008b) Bayesian spiking neurons. II. Learning. Neural Comput 20:118–145

Devert A, Bredeche N, Schoenauer M (2007) Unsupervised learning of echo state networks: a case study in artificial embryogeny. In: Montmarché N et al. (eds) Artificial evolution, selected papers, Lecture Notes in Computer Science, vol 4926/2008, pp 278–290

Eguíluz VM, Chialvo GA, Cecchi DR, Baliki M, Apkarian AV (2005) Scale-free brain functional networks. Phys Rev Lett 94(1):018102

Ermentrout GB, Kopell N (1986) Parabolic bursting in an excitable system coupled with a slow oscillation. SIAM J Appl Math 46:233

Floreano D, Zufferey JC, Nicoud JD (2005) From wheels to wings with evolutionary spiking neurons. Artif Life 11(1–2):121–138

Floreano D, Epars Y, Zufferey J-C, Mattiussi C (2006) Evolution of spiking neural circuits in autonomous mobile robots. Int J Intell Syst 21(9):1005–1024

Funahashi K (1989) On the approximate realization of continuous mappings by neural networks. Neural Netw 2(3):183–192

Gerstner W (1995) Time structure of the activity in neural network models. Phys Rev E 51:738–758

Gerstner W, Kistler WM (2002a) Mathematical formulations of Hebbian learning. Biol Cybern 87(5–6): 404–415

Gerstner W, Kistler W (2002b) Spiking neuron models: single neurons, populations, plasticity. Cambridge University Press, Cambridge

Gerstner W, van Hemmen JL (1994) How to describe neuronal activity: spikes, rates or assemblies? In: Cowan JD, Tesauro G, Alspector J (eds) NIPS 1993, Advances in neural information processing system, vol 6. MIT Press, Cambridge, MA, pp 463–470

Gerwinn S, Macke JH, Seeger M, Bethge M (2007) Bayesian inference for spiking neuron models with a sparsity prior. In: NIPS 2006, Advances in neural information processing systems, vol 19. MIT Press, Cambridge, MA

Hartland C, Bredeche N (2007) Using echo state networks for robot navigation behavior acquisition. In: ROBIO'07, Sanya, China

Hebb DO (1949) The organization of behavior. Wiley, New York

Heiligenberg W (1991) Neural nets in electric fish. MIT Press, Cambridge, MA

Hellmich HH, Geike M, Griep P, Rafanelli M, Klar H (2005) Emulation engine for spiking neurons and adaptive synaptic weights. In: IJCNN 2005, International joint conference on neural networks. IEEE-INNS, Montreal, pp 3261–3266

Hines ML, Carnevale NT (1997) The NEURON simulation environment. Neural Comput 9:1179–1209

Hines ML, Carnevale NT (2004) Discrete event simulation in the NEURON environment. Neurocomputing 58–60:1117–1122

Hines ML, Carnevale NT (2008) Translating network models to parallel hardware in NEURON. J Neurosci Methods 169:425–455

Hodgkin AL, Huxley AF (1952) A quantitative description of ion currents and its applications to conduction and excitation in nerve membranes. J Physiol 117:500–544

Holmberg M, Gelbart D, Ramacher U, Hemmert W (2005) Automatic speech recognition with neural spike trains. In: Interspeech 2005 – Eurospeech, 9th European conference on speech communication and technology, Lisbon, pp 1253–1256

Hopfield JJ (1982) Neural networks and physical systems with emergent collective computational abilities. Proc Natl Acad Sci 79(8):2554–2558

Hopfield JJ (1995) Pattern recognition computation using action potential timing for stimulus representation. Nature 376:33–36

Hopfield JJ, Brody CD (2000) What is a moment? "Cortical" sensory integration over a brief interval. Proc Natl Acad Sci 97(25):13919–13924

Hopfield JJ, Brody CD (2001) What is a moment? Transient synchrony as a collective mechanism for spatiotemporal integration. Proc Natl Acad Sci 98(3):1282–1287

Hornik K, Stinchcombe M, White H (1989) Multilayer feedforward networks are universal approximators. Neural Netw 2(5):359–366

Huys QJM, Zemel RS, Natarajan R, Dayan P (2007) Fast population coding. Neural Comput 19:404–441

Izhikevich EM (2003) Simple model of spiking neurons. IEEE Trans Neural Netw 14(6):1569–1572

Izhikevich EM (2004) Which model to use for cortical spiking neurons? IEEE Trans Neural Netw 15(5): 1063–1070

Izhikevich EM (2006) Polychronization: computation with spikes. Neural Comput 18(2):245–282

Izhikevich EM (2007) Solving the distal reward problem through linkage of STDP and dopamine signaling. Cereb Cortex 17(10):2443–2452

Izhikevich EM, Desai NS (2003) Relating STDP and BCM. Neural Comput 15(7):1511–1523

Izhikevich EM, Gally JA, Edelman GM (2004) Spike-timing dynamics of neuronal groups. Cereb Cortex 14:933–944

Jaeger H (2001) The "echo state" approach to analysing and training recurrent neural networks. Technical Report TR-GMD-148, German National Research Center for Information Technology

Jaeger H (2002) 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

Jaeger H, Haas H (2004) Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless telecommunication. Science 304(5667):78–80

Jaeger H, Lukoševičius M, Popovici D, Siewert U (2007) Optimization and applications of echo state networks with leaky-integrator neurons. Neural Netw 20(3):335–352

Jiang F, Berry H, Schoenauer M (2008a) Supervised and evolutionary learning of echo state networks. In: Rudolph G et al. (eds) Proceedings of the 10th international conference on parallel problem solving from nature: PPSN X. Lecture notes in computer science, vol 5199. Springer, pp 215–224

Jiang F, Berry H, Schoenauer M (2008b) Unsupervised learning of echo state networks: balancing the double pole. In: GECCO'08: Proceedings of the 10th annual conference on genetic and evolutionary computation, ACM, New York, pp 869–870

Johnston S, Prasad G, Maguire L, McGinnity T (2005) Comparative investigation into classical and spiking neuron implementations on FPGAs. In: ICANN 2005, International conference on artificial neural networks. Lecture notes in computer science, vol 3696. Springer, New York, pp 269–274

Judd JS (1990) Neural network design and the complexity of learning. MIT Press, Cambridge, MA

Kempter R, Gerstner W, van Hemmen JL (1999) Hebbian learning and spiking neurons. Phys Rev E 59(4): 4498–4514

Kistler WM (2002) Spike-timing dependent synaptic plasticity: a phenomenological framework. Biol Cyber 87(5–6):416–427

Kistler WM, Gerstner W, van Hemmen JL (1997) Reduction of Hodgkin–Huxley equations to a single-variable threshold model. Neural Comput 9:1015–1045

Klampfl S, Legenstein R, Maass W (2009) Spiking neurons can learn to solve information bottleneck problems and extract independent components. Neural Comput 21(4):911–959

Koerding KP, Wolpert DM (2004) Bayesian integration in sensorimotor learning. Nature 427:244–247

Kohonen T (1982) Self-organized formation of topologically correct feature maps. Biol Cybern 43:59–69

Kutch JJ (2004) Neuromorphic approaches to rehabilitation. Neuromorphic Eng 1(2):1–2

Kuwabara N, Suga N (1993) Delay lines and amplitude selectivity are created in subthalamic auditory nuclei: the brachium of the inferior colliculus of the mustached bat. J Neurophysiol 69:1713–1724

LeCun Y, Jackel LD, Bottou L, Cortes C, Denker JS, Drucker H, Guyon I, Muller UA, Sackinger E, Simard P (1995) Learning algorithms for classification: a comparison on handwritten digit recognition, vol 276. World Scientific, Singapore

Legenstein R, Maass W (2005) What makes a dynamical system computationally powerful? In: Haykin S, Principe JC, Sejnowski TJ, McWhirter JG (eds) New directions in statistical signal processing: from systems to brain. MIT Press, Cambridge, MA

Legenstein R, Näger C, Maass W (2005) What can a neuron learn with spike-time-dependent plasticity? Neural Comput 17(11):2337–2382

Legenstein R, Pecevski D, Maass W (2008) Theoretical analysis of learning with reward-modulated spike-timing-dependent plasticity. In: NIPS 2007, Advances in neural information processing systems, vol 20. MIT Press, Cambridge, MA

Lewicki MS (2002) Efficient coding of natural sounds. Nat Neurosci 5:356–363

Loiselle S, Rouat J, Pressnitzer D, Thorpe S (2005) Exploration of rank order coding with spiking neural networks for speech recognition. In: IJCNN 2005, International joint conference on neural networks. IEEE–INNS Montreal, pp 2076–2080

Lukoševičius M, Jaeger H (July 2007) Overview of reservoir recipes. Technical Report 11, Jacobs University Bremen

Ma WJ, Beck JM, Pouget A (2008) Spiking networks for Bayesian inference and choice. Curr Opin Neurobiol 18(2):217–222

Maass W (1997a) Fast sigmoidal networks via spiking neurons. Neural Comput 10:1659–1671

Maass W (1997b) Networks of spiking neurons: the third generation of neural network models. Neural Netw 10:1659–1671

Maass W (2001) On the relevance of time in neural computation and learning. Theor Comput Sci 261:157–178 (extended version of ALT'97, in LNAI 1316:364–384)

Maass W, Bishop CM (eds) (1999) Pulsed neural networks. MIT Press, Cambridge, MA

Maass W, Natschläger T (1997) Networks of spiking neurons can emulate arbitrary Hopfield nets in temporal coding. Netw: Comput Neural Syst 8(4):355–372

Maass W, Schmitt M (1997) On the complexity of learning for a spiking neuron. In: COLT'97, Conference on computational learning theory. ACM Press, New York, pp 54–61

Maass W, Schmitt M (1999) On the complexity of learning for spiking neurons with temporal coding. Info Comput 153:26–46

Maass W, Steinbauer G, Koholka R (2002a) Autonomous fast learning in a mobile robot. In: Hager GD, Christensen HI, Bunke H, Klein R (eds) Sensor based intelligent robots, vol 2238. Springer, Berlin, pp 345–356

Maass W, Natschläger T, Markram H (2002b) Real-time computing without stable states: a new framework for neural computation based on perturbations. Neural Comput 14(11):2531–2560

Makino T (2003) A discrete event neural network simulator for general neuron model. Neural Comput Appl 11(2):210–223

Markram H, Tsodyks MV (1996) Redistribution of synaptic efficacy between neocortical pyramidal neurones. Nature 382:807–809

Markram H, Lübke J, Frotscher M, Sakmann B (1997) Regulation of synaptic efficacy by coincidence of postsynaptic APs and EPSPs. Science 275:213–215

Masquelier T, Thorpe SJ, Friston KJ (2007) Unsupervised learning of visual features through spike timing dependent plasticity. PLoS Comput Biol 3:e31

Mattia M, Del Giudice P (2000) Efficient event-driven simulation of large networks of spiking neurons and dynamical synapses. Neural Comput 12: 2305–2329

McCulloch WS, Pitts W (1943) A logical calculus of the ideas immanent in nervous activity. Bull Math Biophys 5:115–133

McKennoch S, Voegtlin T, Bushnell L (2009) Spike-timing error backpropagation in theta neuron networks. Neural Comput 21(1):9–45

Meunier D (2007) 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

Meunier D, Paugam-Moisy H (2004) A "spiking" bidirectional associative memory for modeling intermodal priming. In: NCI 2004, International conference on neural networks and computational intelligence. ACTA Press, Calgary, pp 25–30

Meunier D, Paugam-Moisy H (2005) Evolutionary supervision of a dynamical neural network allows learning with on-going weights. In: IJCNN 2005, International joint conference on neural networks. IEEE–INNS, Montreal, pp 1493–1498

Meunier D, Paugam-Moisy H (2006) Cluster detection algorithm in neural networks. In: Verleysen M (ed) ESANN'06, Advances in computational intelligence and learning. D-Side Publishing, Evere, Belgium, pp 19–24

Mitra S, Fusi S, Indiveri G (2006) A VLSI spike-driven dynamic synapse which learns only when necessary. In: Proceedings of IEEE international symposium on circuits and systems (ISCAS) 2006. IEEE Press, New York, p 4

Mouraud A, Paugam-Moisy H (2006) Learning and discrimination through STDP in a top-down modulated associative memory. In: Verleysen M (ed) ESANN'06, Advances in computational intelligence and learning. D-Side Publishing, Evere, Belgium, pp 611–616

Mouraud A, Paugam-Moisy H, Puzenat D (2006) A distributed and multithreaded neural event driven simulation framework. In: PDCN 2006, International conference on parallel and distributed computing and networks, Innsbruck, Austria, February 2006. ACTA Press, Calgary, 2006

Mouraud A, Puzenat D (2009) Simulation of large spiking neuron networks on distributed architectures, the "DAMNED" simulator. In: Palmer-Brown D, Draganova C, Pimenidis E, Mouratidis H (eds) EANN 2009, Engineering applications of neural networks. Communications in computer and information science, vol 43. Springer, pp 359–370

Natschläger T, Ruf B (1998a) Online clustering with spiking neurons using radial basis functions. In: Hamilton A, Smith LS (eds) Neuromorphic systems: engineering silicon from neurobiology. World Scientific, Singapore, Chap 4

Natschläger T, Ruf B (1998b) Spatial and temporal pattern analysis via spiking neurons. Netw: Comp Neural Syst 9(3):319–332

Newman MEJ (2003) The structure and function of complex networks. SIAM Rev 45:167–256

Newman MEJ, Girvan M (2004) Finding and evaluating community structure in networks. Phys Rev E 69:026113

Nowotny T, Zhigulin VP, Selverston AI, Abardanel HDI, Rabinovich MI (2003) Enhancement of synchronization in a hybrid neural circuit by spike-time-dependent plasticity. J Neurosci 23(30):9776–9785

Olshausen BA, Fields DJ (1996) Emergence of simple-cell receptive field properties by learning a sparse code for natural images. Nature 381:607–609

Oster M, Whatley AM, Liu S-C, Douglas RJ (2005) A hardware/software framework for real-time spiking systems. In: ICANN 2005, International conference on artificial neural networks. Lecture notes in computer science, vol 3696. Springer, New York, pp 161–166

Panchev C, Wermter S (2006) Temporal sequence detection with spiking neurons: towards recognizing robot language instructions. Connect Sci 18:1–22

Paugam-Moisy H, Martinez R, Bengio S (2008) Delay learning and polychronization for reservoir computing. Neurocomputing 71(7–9):1143–1158

Perrinet L, Samuelides M (2002) Sparse image coding using an asynchronous spiking neural network. In: Verleysen M (ed) ESANN 2002, European symposium on artificial neural networks. D-Side Publishing, Evere, Belgium, pp 313–318

Pfister J-P, Gerstner W (2006) Beyond pair-based STDP: a phenomenological rule for spike triplet and frequency effects. In: NIPS 2005, Advances in neural information processing systems, vol 18. MIT Press, Cambridge, MA, pp 1083–1090

Pfister J-P, Barber D, Gerstner W (2003) Optimal Hebbian learning: a probabilistic point of view. In: Kaynak O, Alpaydin E, Oja E, Xu L (eds) ICANN/ICONIP 2003, International conference on artificial neural networks. Lecture notes in computer science, vol 2714. Springer, Heidelberg, pp 92–98

Pfister J-P, Toyoizumi T, Barber D, Gerstner W (2006) Optimal spike-timing-dependent plasticity for precise action potential firing in supervised learning. Neural Comput 18(6):1318–1348

Poggio T, Girosi F (1989) Networks for approximation and learning. Proc IEEE 78(9):1481–1497

Rao RPN (2005) Hierarchical Bayesian inference in networks of spiking neurons. In: Saul LK, Weiss Y, Bottou L (eds) NIPS 2004, Advances in neural information processing systems, vol 17. MIT Press, Cambridge, MA, pp 1113–1120

Recce M (1999) Encoding information in neuronal activity. In: Maass W, Bishop CM (eds) Pulsed neural networks. MIT Press, Cambridge

Reutimann J, Giugliano M, Fusi S (2003) Event-driven simulation of spiking neurons with stochastic dynamics. Neural Comput 15(4):811–830

Rochel O, Martinez D (2003) An event-driven framework for the simulation of networks of spiking neurons. In: Verleysen M (ed) ESANN'03, European symposium on artificial neural networks. D-Side Publishing, Evere, Belgium, pp 295–300

Rubin J, Lee DD, Sompolinsky H (2001) Equilibrium properties of temporal asymmetric Hebbian plasticity. Phys Rev Lett 86:364–366

Rudolph M, Destexhe A (2006) Event-based simulation strategy for conductance-based synaptic interactions and plasticity. Neurocomputing 69: 1130–1133

Rumelhart DE, Hinton GE, Williams RJ (1986) Learning internal representations by back-propagating errors. Nature 323:533–536

Sahani M, Dayan P (2003) Doubly distributional population codes: Simultaneous representation of uncertainty and multiplicity. Neural Comput 15:2255–2279

Salmen M, Plöger PG (2005) Echo state networks used for motor control. In: ICRA 2005, International joint conference on robotics and automation. IEEE, New York, pp 1953–1958

Saudargiene A, Porr B, Wörgötter F (2004) How the shape of pre- and postsynaptic signals can influence STDP: a biophysical model. Neural Comput 16(3): 595–625

Schmidhuber J, Wiestra D, Gagliolo D, Gomez M (2007) Training recurrent networks by Evolino. Neural Comput 19(3):757–779

Schmitt M (1998) On computing Boolean functions by a spiking neuron. Ann Math Artif Intell 24: 181–191

Schmitt M (2004) On the sample complexity of learning for networks of spiking neurons with nonlinear synaptic interactions. IEEE Trans Neural Netw 15(5): 995–1001

Schrauwen B, Van Campenhout J (2004a) Extending SpikeProp. In: Proceedings of the international joint conference on neural networks, vol 1. IEEE Press, New York, pp 471–476

Schrauwen B, Van Campenhout J (2004b) Improving spikeprop: enhancements to an error-backpropagation rule for spiking neural networks. In: Proceedings of the 15th ProRISC workshop, vol 11

Schrauwen B, D'Haene M, Verstraeten D, Van Campenhout J (2007a) Compact hardware for real-time speech recognition using a liquid state machine. In: IJCNN 2007, International joint conference on neural networks, 2007, pp 1097–1102

Schrauwen B, Verstraeten D, Van Campenhout J (2007b) An overview of reservoir computing: theory, applications and implementations. In: Verleysen M (ed) ESANN'07, Advances in computational intelligence and learning. D-Side Publishing, Evere, Belgium, pp 471–482

Schrauwen B, Büsing L, Legenstein R (2009) On computational power and the order-chaos phase transition in reservoir computing. In: Koller D, Schuurmans D, Bengio Y, Bottou L (eds) NIPS'08, advances in neural information processing systems, vol 21. MIT Press, Cambridge, MA, pp 1425–1432

Séguie R, Mercier D (2002) Audio-visual speech recognition one pass learning with spiking neurons. In: ICANN'02, International conference on artificial neural networks. Springer, Berlin, pp 1207–1212

Senn W, Markram H, Tsodyks M (2001) An algorithm for modifying neurotransmitter release probability based on pre- and post-synaptic spike timing. Neural Comput 13(1):35–68

Siegelmann HT (1999) Neural networks and analog computation, beyond the Turing limit. Birkhauser, Boston, MA

Sima J, Sgall J (2005) On the nonlearnability of a single spiking neuron. Neural Comput 17(12):2635–2647

Smith EC, Lewicki MS (2006) Efficient auditory coding. Nature 439:978–982

Song S, Miller KD, Abbott LF (2000) Competitive Hebbian learning through spike-time dependent synaptic plasticity. Nat Neurosci 3(9):919–926

Sporns O, Tononi G, Kotter R (2005) The human connectome: a structural description of the human brain. PLoS Comp Biol 1(4):e42

Standage DI, Trappenberg TP (2005) Differences in the subthreshold dynamics of leaky integrate-and-fire and Hodgkin-Huxley neuron models. In: IJCNN 2005, International joint conference on neural networks. IEEE–INNS, Montreal, pp 396–399

Steil JJ (2004) Backpropagation-decorrelation: Online recurrent learning with O(n) complexity. In: IJCNN 2004, International joint conference on neural networks, vol 1. IEEE–INNS, Montreal, pp 843–848

Stein RB (1965) A theoretical analysis of neuronal variability. Biophys J 5:173–194

Tenore F (2004) Prototyping neural networks for legged locomotion using custom aVLSI chips. Neuromorphic Eng 1(2):4, 8

Thorpe SJ, Gautrais J (1997) Rapid visual processing using spike asynchrony. In: Mozer M, Jordan MI, Petsche T (eds) NIPS 1996, Advances in neural information processing systems, volume 9. MIT Press, Cambridge, MA, pp 901–907

Thorpe S, Fize D, Marlot C (1996) Speed of processing in the human visual system. Nature 381(6582): 520–522

Thorpe S, Delorme A, Van Rullen R (2001) Spike-based strategies for rapid processing. Neural Netw 14:715–725

Toyoizumi T, Pfister J-P, Aihara K, Gerstner W (2005a) Generalized Bienenstock-Cooper-Munro rule for spiking neurons that maximizes information transmission. Proc Natl Acad Sci USA 102(14): 5239–5244

Toyoizumi T, Pfister J-P, Aihara K, Gerstner W (2005b) Spike-timing dependent plasticity and mutual information maximization for a spiking neuron model. In: Saul LK, Weiss Y, Bottou L (eds) NIPS 2004, Advances in neural information processing systems, vol 17. MIT Press, Cambridge, MA, pp 1409–1416

Turing AM (1939) Systems of logic based on ordinals. Proc Lond Math Soc 45(2):161–228

Turing AM (1950) Computing machinery and intelligence. Mind 59:433–460

Upegui A, Peña Reyes CA, Sanchez E (2004) An FPGA platform for on-line topology exploration of spiking neural networks. Microprocess Microsyst 29:211–223

Valiant LG (1984) A theory of the learnable. Commun ACM 27(11):1134–1142

van Hulle M (2000) Faithful representations and topographic maps: from distortion- to information-based self-organization. Wiley, New York

Van Rullen R, Thorpe S (2001) Rate coding versus temporal order coding: what the retinal ganglion cells tell the visual cortex. Neural Comput 13: 1255–1283

Vapnik VN (1998) Statistical learning theory. Wiley, New York

Verstraeten D, Schrauwen B, Stroobandt D (2005) Isolated word recognition using a liquid state machine. In: Verleysen M (ed) ESANN'05, European symposium on artificial neural networks. D-Side Publishing, Evere, Belgium, pp 435–440

Verstraeten D, Schrauwen B, D'Haene M, Stroobandt D (2007) An experimental unification of reservoir computing methods. Neural Netw 20(3):391–403

Viéville T, Crahay S (2004) Using an Hebbian learning rule for multi-class SVM classifiers. J Comput Neurosci 17(3):271–287

Volkmer M (2004) A pulsed neural network model of spectro-temporal receptive fields and population coding in auditory cortex. Nat Comput 3: 177–193

Wang G, Pavel M (2005) A spiking neuron representation of auditory signals. In: IJCNN 2005, International joint conference on neural networks. IEEE–INNS, Montreal, pp 416–421

Watts L (1994) Event-driven simulation of networks of spiking neurons. In: Cowan JD, Tesauro G, Alspector J (eds) NIPS 1993, Advances in neural information processing systems, vol 6. MIT Press, Cambridge, MA, pp 927–934

Watts D, Strogatz S (1998) Collective dynamics of "small-world" networks. Nature 393:440–442

Wennekers T, Sommer F, Aertsen A (2003) Editorial: cell assemblies. Theory Biosci (special issue) 122:1–4

Wu S, Chen D, Niranjan M, Amari S (2003) Sequential Bayesian decoding with a population of neurons. Neural Comput 15:993–1012

Wysoski SG, Benuskova L, Kasabov N (2008) Fast and adaptive network of spiking neurons for multi-view visual pattern recognition. Neurocomputing 71(13–15):2563–2575

Xie X, Seung HS (2004) Learning in neural networks by reinforcement of irregular spiking. Phys Rev E 69 (041909)

Xin J, Embrechts MJ (2001) Supervised learning with spiking neuron networks. In: Proceedings of the IJCNN 2001 IEEE international joint conference on neural networks, Washington, DC, vol 3. IEEE Press, New York, pp 1772–1777

Zador AM, Pearlmutter BA (1996) VC dimension of an integrate-and-fire neuron model. Neural Comput 8(3):611–624

Zemel RS, Dayan P, Pouget A (1998) Probabilistic interpretation of population codes. Neural Comput 10:403–430