IDIAP RESEARCH REPORT

SPIKING NEURON NETWORKS
A SURVEY

Hélène Paugam-Moisy [1,2]

IDIAP–RR 06-11

FEBRUARY 2006

1 ISC-CNRS, Lyon, France, hpaugam@isc.cnrs.fr
2 IDIAP Research Institute, CP 592, 1920 Martigny, Switzerland and Ecole Polytechnique Fdrale de Lausanne (EPFL), Switzerland, Helene.Paugam-Moisy@idiap.ch

IDIAP Research Report 06-11

SPIKING NEURON NETWORKS
A SURVEY

Hélène Paugam-Moisy

FEBRUARY 2006

Abstract. Spiking Neuron Networks (SNNs) are often referred to as the 3^{rd} generation of neural networks. They derive their strength and interest from an accurate modelling of synaptic interactions between neurons, taking into account the time of spike emission. 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 developping models with an exponential capacity of memorizing and a strong ability to fast adaptation. Today, the main challenge is to discover efficient learning rules that might take advantage of the specific features of SNNs while keeping the nice properties (general-purpose, easy-to-use, available simulators, etc.) of current connectionist models (such as MLP, RBF or SVM).
The present survey relates the history of the "spiking neuron" and summarizes the most_currenlty_in_use models of neurons and networks, in Section 1. The computational power of SNNs is addressed in Section 2 and the problem of learning in networks of spiking neurons is tackled in Section 3, with insights into the tracks currently explored for solving it. Section 4 reviews the tricks of implementation and discuss several simulation frameworks. Examples of application domains are proposed in Section 5, mainly in speech processing and computer vision, emphasizing the temporal aspect of pattern recognition by SNNs.
Keywords. Spiking neurons, Spiking neuron networks, Pulsed neural networks, Synaptic plasticity, STDP.

Contents

1 Models of spiking neurons and networks

1.1 Traditional models of neurons

Starting from the mathematical model of McCulloch and Pitts [107] in 1943 (Figure 1), models of neuron evolved along the last six decades. The common basis consists in applying a non-linear function (threshold, sigmoid, or other) to a weighted sum of inputs. Both inputs and outputs can be either boolean (e.g. in the original threshold model) or real variables, but they are always numerical variables and all of them receive a value at each neural computation step. Several connectionist models (e.g. RBF networks, Kohonen self-organisation maps) make use of "distance neurons" where the dot product of the weights W and inputs X, denoted by $< X, W >$ or $W.X$, is replaced by the (usually quadratic) distance $\parallel X - W \parallel$ (Figure 2).

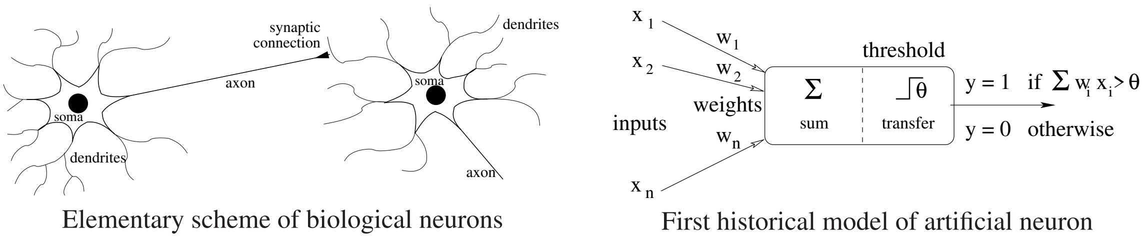


Figure 1: The first mathematical model of neuron picked up the most significant features of a natural neuron: All-or-none output resulting from a non-linear transfer function applied to a weighted sum of inputs.

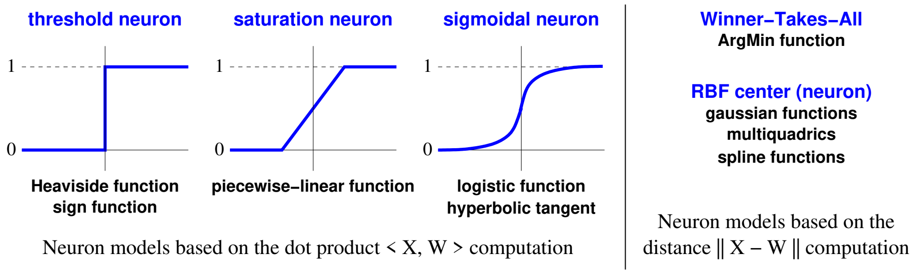


Figure 2: Several variants of neuron models, based on a dot product or a distance computation.

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

- Calculability: The computational power of NNs outperforms a common Turing machine [146, 145]
- Complexity: The "loading problem" is NP-complete [17, 80]
- Capacity: MLP, RBF and WNN[1] are universal approximators [35, 45, 67]
- Regularization theory [126]; PAC-learning[2] [161]; Statistical learning theory, VC-dim, SVM[3] [164]

nevertheless, they admit of intrinsic limitations, e.g. for processing large amount of data or for fast adaptation to changing environment.

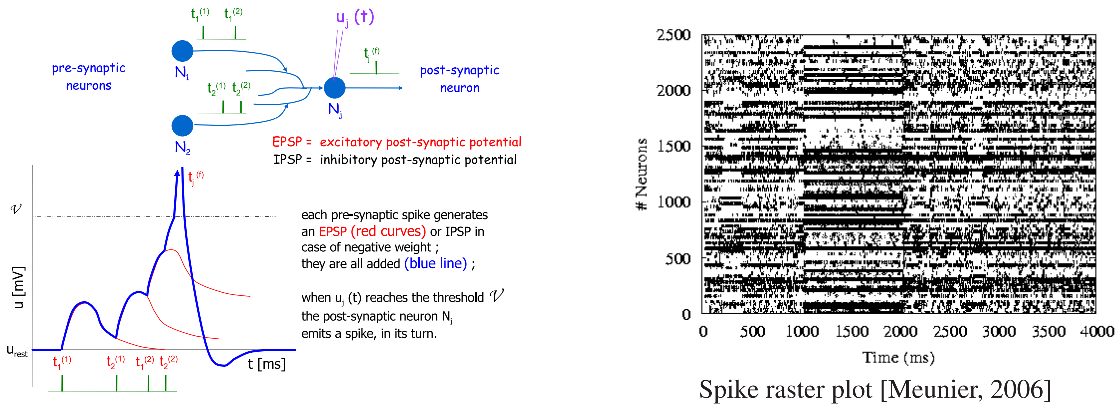
Usually, learning and generalisation phases of classic connectionist models are described by iterative algorithms. At each time step, each neuron receives information from each of its input channels and computes them according to its self formula and with its current weight values. Moreover, all the neurons of a given neural network usually belong to a unique model of neuron, or at most two models (e.g. RBF centers and linear output). Such characteristics are strongly restrictive compared with biological processing in natural neural networks.

[1] MLP = Multi-Layer Percpetrons - RBF = Radial Basis Function networks - WNN = Wavelet Neural Networks
[2] PAC learning = Probably Approximately Correct learning
[3] VC-dim = Vapnik-Chervonenkis dimension, for learning systems - SVM = Support Vector Machines

1.2 The biological inspiration, revisited

A new investigation in natural neuron processing is motivated by the evolution of thinking the basic principles of brain processing. At the time of the historical first neuron modelling, the common idea was that intelligence is based on reasoning and that logic is the fundation of reasoning. McCulloch and Pitts designed their model of neuron in order to prove that the elementary components of the brain were able to compute elementary logic functions. The first use they made of threshold neurons with n binary inputs was the synthesis of boolean functions of n variables. In the tradition of Turing's work [158, 159], they thought that a complex, may-be "intelligent" behaviour could emerge from a very large network of neurons, combining huge numbers of elementary logic gates. It's turned out that such basic ideas were very productive, even if effective learning rules for large networks (e.g. backpropagation, for MLP) have been discovered only at the end of the 80's, and even if the idea of boolean decomposition of tasks has been left out for long.

In the meantime, neurobiological research also made wide progress in fifty years. Notions such as associative memory, learning, adpatation, attention and emotions tend to supersede the place of logic and reasoning for better understanding how the brain processes, and **time** becomes a central feature in cognitive processing [3]. Brain imaging and a lot of new technologies (micro-electrode, LFP[4] or EEG[5] recordings, fMRI[6]) help to detect changes in the internal activity of brain, e.g. related to the perception of a given stimulus. Now it is currently agreed that most of cognitive processes are based on sporadic synchronization of transient assemblies of neurons, although the underlying mechanisms are not yet understood well. At the microscopic level, as extensively developed in [96], "neurons use action potentials[7] to signal over distances. The all-or-none nature of the action potential means that it codes information by its presence or absence, but not by its size or shape. The timing of spikes is already well established as a mean for coding information in the electrosensory system of electric fish [58], in the auditory system of echolocating bats [89], in the visual system of flies [15]", etc…



Spike raster plot [Meunier, 2006]

Figure 3: A spiking neuron N_j emits a spike whenever the weighted sum of incoming PSPs generated by its pre-synaptic neurons reaches a given threshold. A spike raster plot displays a bar each time a neuron emits a spike (one line per neuron). The firing pattern is modified when an input stimulus is applied to the network (here in the time range $1000 - 2000ms$): A synchronized cell assembly is formed and then disrupted.

The duration of an action potential emitted by a natural neuron is typically in the range of $1 - 2ms$. Rather than the form of the action potential, it is the number and the timing of spikes that matter. Hence the idea is to revisit the neuron modelling for taking into account the firing times of neurons (Figure 3), so defining the family of **spiking neurons**. The main difference consists in considering that neurons communicate information to each others by a *pulse code* rather than a *rate code*.

[4]LFP = Local Field Potential

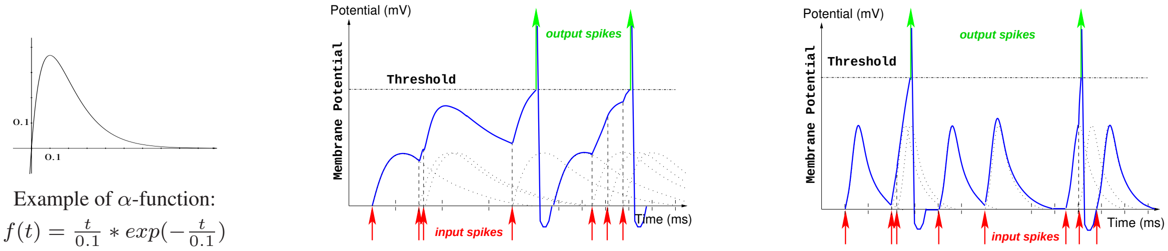
[5]EEG = ElectroEncephaloGram

[6]fMRI= functional Magnetic Resonance Imaging

[7]action potential = spike = pulse = wave of electric discharge that travels along the membrane of a neuron

At the neuron level, the membrane potential $u_j(t)$ of a postsynaptic neuron N_j varies continuously through time (cf. Figure 3, left). Each **action potential**, or **spike**, or **pulse**, emitted by a presynaptic neuron connected to N_j generates a weighted **PostSynaptic Potential** (PSP) which is function of time, usually modelled by an α-*function* like $x\ exp^{-x}$ where $x = t/\tau$ with an adjustable time constant τ (cf. left side of Figure 4). If the w_{ij} synaptic weight is excitatory, the EPSP is positive: Sharp increasing of the potential $u_j(t)$ and then smoothly decreasing back to null influence. If w_{ij} is inhibitory then the IPSP is negative: Sharp decreasing $u_j(t)$ and then smoothly increasing (not represented in the figure). At each time, the value of $u_j(t)$ results from the addition of the still active PSPs variations (blue curve in Figure 3). Whenever the potential $u_j(t)$ reaches the threshold value ϑ of N_j, the neuron **fires** or **emits a spike**, that corresponds to a sudden and very high increase of $u_j(t)$, followed by a strong depreciation and a smooth return to the resting potential u_{rest}.

The dynamics of neuron N_j, in Figure 3, has been good enough for explaining the general behaviour of a spiking neuron, but the changes of membrane potential can obey to slightly different laws and the firing behaviour of a neuron can vary, e.g. [86] from the role of "integrator" to the role of "coincidence detector". On the middle diagram of Figure 4, most of the incoming PSPs contribute to one or other spike emission. The neuron makes a temporal integration of incoming information. Conversely, right side of Figure 4 displays a coincidence detection behaviour. Only quasi-synchronously arriving PSPs trigger a very fast response by spike emission, whereas other PSPs have no output effect. Again different sets of parameters would make the neuron reproduce many other neuro-computational properties (see Figure 7, in Section 1.3.2).



Example of α-function: $f(t) = \frac{t}{0.1} * exp(-\frac{t}{0.1})$

Integrator

Coincidence detector

Figure 4: The role of a spiking neuron, for information coding, may depend on its parameters (threshold, time constant of membrane potential) and the inter-spike interval of presynaptic neurons emissions.

At the network level (Figure 3, right), a complete representation of network activity can be displayed on a spike raster plot: each neuron (number in Y-axis) is represented by a line varying with time (in abscissa) and a bar is ploted each time the neuron fires. The variations and frequencies of all the neurons activities can be observed on such diagrams of spikes, in the same way as natural neurons activities can be observed on spike rasters drawn from multi-electrode recordings. Likewise, other representations (e.g. time-frequency diagrams) can be drawn from simulation of artificial networks of spiking neurons, as well as in neuroscience, from experimental data. The spike raster of Figure 3 results from simulating a network of 2500 spiking neurons [109]: the injection of a perceptive input stimulus during time range $1000 - 2000ms$ clearly makes the temporal firing pattern different from background activity, hence highlighting the significance of pulse coding.

Strong arguments against rate coding have been given by Thorpe et al. [155, 163] in the context of visual processing. A Poisson-like rate code was a common idea admitted by many physiologists for describing the way that neurons transmit information. However Poisson rate codes are too inefficient to account for the rapid information transmission required for sensory processing in human vision. Only $100 - 150ms$ are sufficient for a neuron to respond selectively to complex visual stimuli (e.g. faces or food), but due to the feedforward architecture of visual system, made of multiple layers of neurons firing at an average rate of $10ms$, only one spike or none could be emitted by each neuron involved in the process during this time range. Hence it seems clear that the timing conveys informations, and not the number of spikes. Moreover, the rank order of the first spikes (e.g. after each visual saccade) could be significant enough for coding salient information [154].

In traditional models of neurons, the x_i component of an input vector X can be interpreted as an average firing rate of the pre-synaptic neuron N_i. This firing rate may result either from an average over time (spike count) or from an average over several runs (spike density), as precised by Gerstner in [96, 50]. Anyway, the basic principles of classic neural networks processing are related to a very large time scale, compared to the processing of **Spiking Neuron Networks** (SNNs)[8]. Since the basic principles are radically different, it is no surprise that the *old* material of neural network literature (learning rules, theoretical results, etc.) has to be adapted, or even more to be rethought fundamentally. The main purpose of this paper is to report as best as possible the state-of-the-art on different aspects concerning the SNNs, from theory to practice, via implementation. The first difficult task is to define "the" spiking neuron model because there exist numerous variants already.

1.3 Models of spiking neurons

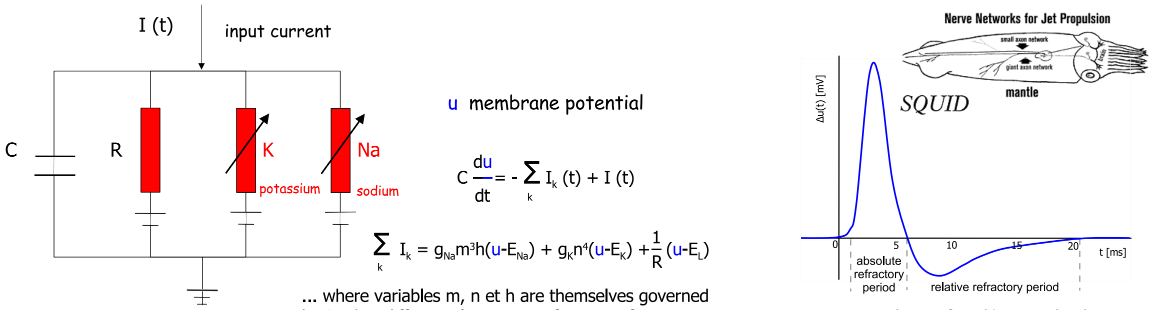
1.3.1 Hodgkin-Huxley model

The fathers of the spiking neurons are the conductance-based models, such as the well-known electrical model defined by Hodgkin and Huxley [61] in 1952 (Figure 5). The basic idea is to model electro-chemical information transmission of natural neurons by electric circuits made of capacitors and resistors: C is the capacitance of the membrane, the g_i are the conductance parameters for the different ion channels (sodium Na, potassium K, etc.) and the E_i are the corresponding equilibrium potentials. Variables m, h and n describe the opening and closing of the voltage dependent channels.

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

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

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



Dynamics of spike emission

Figure 5: Electrical model of "spiking" neuron [Hodgkin-Huxley, 1952]. The model is able to produce the dynamics of a spike emission, e.g. in response to an input current $I(t)$ sent during a small time, at $t < 0$.

The HH model has been compared successfully (with appropriate calibration of parameters) to numerous data from biological experiments on the giant axon of the squid. More generally, the HH model is able to model biophysically meaningful variations of the membrane potential, respecting the shape recordable from natural neurons: An abrupt, high increase at firing time, followed by a short time where the neuron is unable to spike again, the *absolute refractoriness*, and a further time range where the membrane is underpolarized, which makes a new firing more difficult, i.e. the *relative refractory period* (Figure 5).

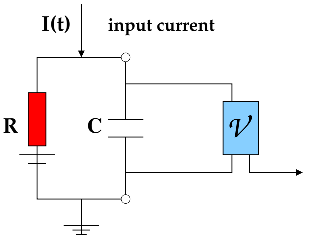
[8]sometimes, SNNs are also named Pulsed-Coupled Neural Networks (PCNNs)
[9]ODE = Ordinary Differential Equations

1.3.2 Integrate-and-Fire model

More tractable models are the **Integrate-and-Fire** (I&F) neurons [90, 151]. The most important simplification 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 basic circuit (Figure 6) consists of a capacitor C in parallel with a resistor R driven by an input current $I(t)$. The dynamics of the membrane potential can be described by a single first-order linear differential equation. Defining the time constant of the neuron membrane as $\tau_m = RC$, for modelling the voltage leakage, the usual formula for the **Leaky Integrate-and-Fire** neuron (LIF), can be written as follows (coherent with Figure 6, under the common assumption $u_{rest} = 0$):

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

In addition, 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_r. An absolute refractory period can be modelled by forcing the neuron to a value $u = -u_{abs}$ during a time d_{abs} after a spike emission, and then restarting the integration with initial value $u = u_r$.



$\mathbf{u}$ membrane potential

$$C\frac{d\mathbf{u}}{dt} = -\frac{1}{R}\mathbf{u}(t) + I(t)$$

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

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

Figure 6: The Leaky Integrate-and-Fire (LIF) model is a simplification of the Hodgkin-Huxley model.

There exist many variations between the HH and LIF models, with decreasing biophysical plausibility, but also with decreasing computational cost (see [70] for rather a complete review or [150] for in-depth comparison of HH and LIF subthreshold dynamics). Since it is defined by four differential equations, the HH model requires about 1200 floating point computations (FLOPS) per $1ms$ simulation. Simplified to two differential equations, the Morris-LeCar or FitzHugh-Nagamo models have still a computational cost of one or several hundreds FLOPS. Only 5 FLOPS are required by the LIF model, and around 10 FLOPS for its variants, the I&F with adaptation, the quadratic I&F (QIF), etc... But the LIF ability to reproduce neuro-computational properties is limited to 3 different firing schemes among the 20 possible behaviours inventoried by Izhikevich [70] (most of them on Figure 7[10]): the "tonic spiking" (A), the "class 1 excitable" (G) and the "integrator" (L).

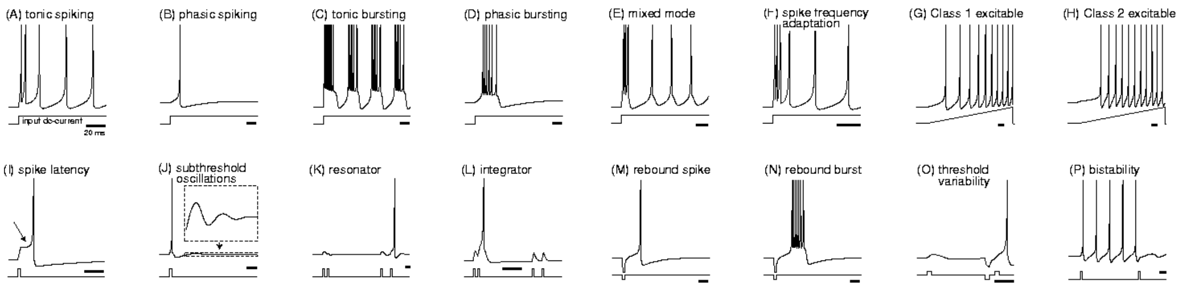


Figure 7: Many different neuro-computational properties and various firing behaviours can occur in biological spiking neurons. Shown are simulations of the Izhikevich neuron model, for different external input currents (displayed under each temporal firing pattern) [From *Izhikevich* [70]].

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

Note that a same neuron cannot be at once "integrator" and "resonator" since the properties are mutually exclusive, but a same neuron model can simulate all of them, with different choices of parameters. In the class of spiking neurons controlled by differential equations, the two-dimensional **Izhikevich neuron model** [69] defined by the coupled equations

$$\frac{du}{dt} = 0.04u(t)^2 + 5u(t) + 140 - w(t) + I(t) \qquad\qquad \frac{dw}{dt} = a\left(bu(t) - w(t)\right) \tag{3}$$

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

is a good compromise between biophysical plausibility and computational cost ($\sim$ 13 FLOPS).

1.3.3 Spike Response Model

More simple to understand and to implement is the **Spike Response Model** (SRM) defined by Gerstner [48, 85]. The model expresses the membrane potential u at time t as an integral over the past, including a model of refractoriness, but without any more differential equation. SRM is a phenomenological model of neuron, based on the occurence of spike emissions. Let $\mathcal{F}_j = \{t_j^{(f)}; 1 \leq f \leq n\} = \{t \mid u_j(t) = \vartheta \ \wedge \ u_j'(t) > 0\}$ denote the set of all firing times of neuron N_j, and $\Gamma_j = \{i \mid N_i \text{ is presynaptic to } N_j\}$ define its set of presynaptic neurons. The state $u_j(t)$ of neuron N_j at time t is given by

$$u_j(t) = \sum_{t_j^{(f)} \in \mathcal{F}_j} \eta_j\left(t - t_j^{(f)}\right) + \sum_{i \in \Gamma_j} \sum_{t_i^{(f)} \in \mathcal{F}_i} w_{ij}\epsilon_{ij}\left(t - t_i^{(f)}\right) + \underbrace{\int_0^\infty \kappa_j(r)I(t-r)dr}_{\text{if external input current}} \tag{4}$$

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 response to presynaptic spikes, and κ_j describes the response of the membrane potential to an external input current. For the kernel functions, a choice of usual expressions is given by:

$$\eta_j(s) = -\vartheta \, exp\left(-\tfrac{s}{\tau}\right)\mathcal{H}(s) \qquad \text{or} \qquad \eta_j(s) = -\eta_0 \, exp\left(-\tfrac{s-\delta^{abs}}{\tau}\right)\mathcal{H}(s-\delta^{abs}) - K\mathcal{H}(s)\mathcal{H}(\delta^{abs}-s)$$

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

$$\epsilon_{ij}(s) = \tfrac{s-d_{ij}^{ax}}{\tau_s} \, exp\left(-\tfrac{s-d_{ij}^{ax}}{\tau_s}\right) \qquad \text{or} \qquad \epsilon_{ij}(s) = \left[exp\left(-\tfrac{s-d_{ij}^{ax}}{\tau_m}\right) - exp\left(-\tfrac{s-d_{ij}^{ax}}{\tau_s}\right)\right]\mathcal{H}(s-d_{ij}^{ax})$$

α-function, or expression where τ_m and τ_s are time constants, with d_{ij}^{ax} the axonal transmission delay.

Kernel ϵ_{ij} describes the generic response of neuron N_j to spikes coming from presynaptic neurons N_i. For the sake of simplicity, $\epsilon_{ij}(s)$ can be assumed to have the same form $\epsilon(s - d_{ij}^{ax})$ 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 $w_{ij} < 0$).

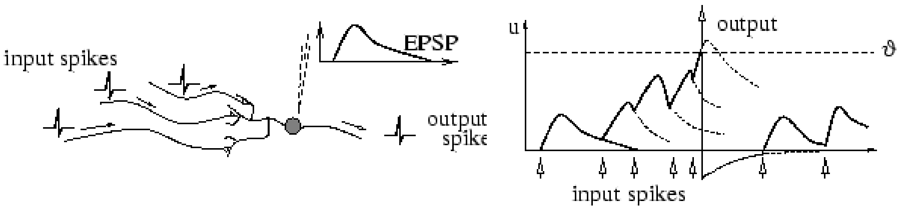


Figure 8: The Spike Response Model (SRM) is a generic framework to describe the spike process [From [50]].

A short term memory variant of SRM results from assuming that only the last firing $\hat{t}_j$ of N_j contributes to refractoriness, $\eta_j\left(t - \hat{t}_j\right)$ replacing the sum in formula (4). Moreover, integrating the equation on a small time window of $1ms$ and assuming that each presynaptic neuron emits at most once in the time window (reasonable since refractoriness of presynaptic neurons), we obtain the again simplified formula of model **SRM**$_0$, which is very close to the usual expression of neural activity, in rate coding

$$v_j(t) = \sum_{i \in \Gamma_j} w_{ij} \epsilon(t - \hat{t}_i - d_{ij}^{ax}) \quad \text{with next firing time} \quad t_j^{(f+1)} = t \Longleftrightarrow v_j(t) = \vartheta - \underbrace{\eta_j\left(t - \hat{t}_j\right)}_{threshold\ kernel} \tag{5}$$

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

1.4 Spiking neuron networks (SNNs)

Even if Izhikevich presently declares on his web site (http://www.nsi.edu/users/izhikevich/interest/index.htm):

> *On October 27, 2005 I finished simulation of a model that has the size of the human brain. The model has* $100,000,000,000$ *neurons (hundred billion or* 10^{11}*) and almost (one quadrillion or* 10^{15}*)* $1,000,000,000,000,000$ *synapses. It represents* $300 \times 300mm^2$ *of mammalian thalamo-cortical surface, specific, non-specific, and reticular thalamic nuclei, and spiking neurons with firing properties corresponding to those recorded in the mammalian brain. One second of simulation took* 50 *days on a beowulf cluster of* 27 *processors (*$3GHz$ *each). Indeed, no significant contribution to neuroscience could be made by simulating one second of a model, even if it has the size of the human brain. However, I learned what it takes to simulate such a large-scale system.*
> Take home message:
> *Size doesn't matter; it's what you put into your model and how you embed it into the environment.*

"large networks of spiking neurons" simulated in reasonable time, on a single computer, with LIF or SRM neuron models, may reach a size of several millions of synaptic connections. How to effectively and efficiently implement SNNs will be discussed in Section 4, whereas several case-studies of application will be developed in Section 5. Sections 2 and 3 are dedicated to theoretical issues, highlighting the strong computational power of SNNs beside the difficulty to control synaptic plasticity for designing new efficient learning rules.

1.4.1 Network topology and dynamics

Since networks of spiking neurons definitely behave in a different way than traditional neural networks, there is no reason to design SNNs in accordance with the usual schemes for architectures and dynamics, such as multilayer feedforward networks (e.g. MLP) or completely connected recurrent networks (e.g. Hopfield net). According to biological observations, the neurons of an SNN are sparsely and irregularly connected in space (network topology) and the variability of spike flows implies they communicate irregularly in time (network dynamics) with a low average activity. It is important to note that the network topology becomes a simple underlying support to the neural dynamics, but that active neurons only are decisive for information processing. At a given time t, the sub-topology defined by active neurons can be very sparse and different from the underlying network architecture (e.g. local clusters, short or long paths loops, synchronized cell assemblies), comparable to the active brain regions that appear coloured in brain imaging scanners. Hence an SNN architecture has no need to be regular. A SNN can even be defined randomly [98, 75] or by a loosely specified architecture, such as a set of neuron groups that are linked by projections, with average probability of connection from one group to the other [109]. However, the nature of a connection has to be prior defined as excitatory or inhibitory synaptic link, without subsequent change, except for the synaptic efficacy, i.e. the weight value can be modified, but not the weight sign.

The architecture of an SNN can be forced to match traditional connectionist models by temporal coding, as developed next (Section 1.4.2), but the cost of drastic simplifications in the model neuron is to lose precious features of firing times based computing. Otherwise, a new family of networks seems to be suitable for processing temporal input / output patterns with spiking neurons: The Echo State Networks (ESNs) and the Liquid State Machines (LSMs). The very similar two models will be shortly presented afterwards (Sections 1.4.3 and 1.4.4).

1.4.2 Computing with temporal patterns

Temporal coding[11] is a straightforward method for translating a vector of real numbers into a spike train, e.g. for simulating the traditional conectionist models by SNNs. The basic idea is biologically well-founded: more intensive the input, earlier the spike transmission (e.g. in visual system). Hence a network of spiking neurons can be designed with m input neurons $N_i \in \mathcal{N}_{in}$ whose firing times are determined through some external mechanism. The network is fed by successive m-dimensional input patterns $\mathbf{x} = (x_1, \ldots, x_m)$ - with all x_i inside a bounded interval of $\mathbb{R}$, e.g. $[0, 1]$ - that are translated into spike trains through successive temporal windows (comparable to successive steps of traditional NNs computation). In each time window, a pattern $\mathbf{x}$ is temporally coded relating 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.

Now consider that all the other neurons of the network ($N_j \notin \mathcal{N}_{in}$) compute the spikes incoming from their set Γ_j of presynaptic neurons, according to the SRM formula (cf. Equation 4)

$$u_j(t) = \sum_{t_j^{(f)} \in \mathcal{F}_j} \eta_j \left(t - t_j^{(f)}\right) + \sum_{i \in \Gamma_j} \sum_{t_i^{(f)} \in \mathcal{F}_i} w_{ij} \epsilon_{ij} \left(t - t_i^{(f)}\right)$$

where the first term of right-hand equation can be ignored (i.e. drop the refractory terms $\eta_j(t - t_j^{(f)})$), since the temporal window mechanism allows to suppose that neuron N_j has not fired for a while. For the same reason, the sets $\mathcal{F}_i$ of firing times of presynaptic neurons can be reduced to singletons. Another model simplification is to consider that PSP functions ϵ_{ij} rise (for an EPSP) or descent (for an IPSP) linearly with slope $\lambda_{ij} \in \mathbb{R}$ for a time range at least $R > 0$ (Figure 9).

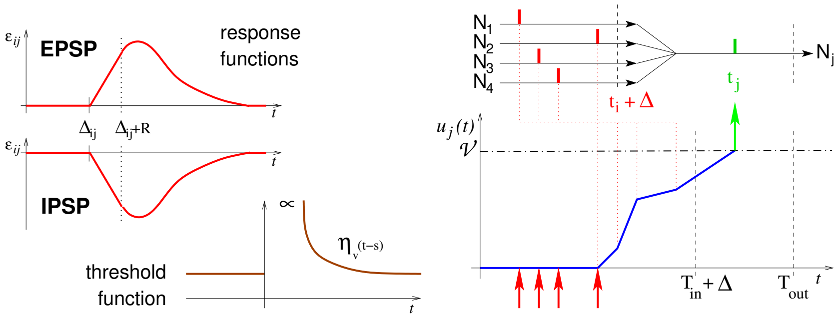


Figure 9: Shapes of postsynaptic potential (EPSP or IPSP) for computing a weighted sum in temporal coding. Right: Example variation of neuron N_j membrane potential for computing $\sum_{i \in \Gamma_j} \alpha_{ij} x_i$ and resulting firing time t_j. All the delays Δ_{ij} have been set equal to Δ. Neuron N_4 (third firing) is inhibitory whereas the other three are excitatory. The slopes of the PSPs are modulated by the synaptic efficacies w_{ij}.

Any neuron N_j can compute a weighted sum $\sum_{i \in \Gamma_j} \alpha_{ij} x_i$ in firing at a time t_j determined by the equation

$$\sum_{i \in \Gamma_j} w_{ij} \epsilon_{ij} \left(t_j - t_i\right) = \sum_{i \in \Gamma_j} w_{ij} \lambda_{ij} \left(t_j - t_i - \Delta_{ij}\right) = \vartheta \tag{6}$$

from which the output y_j of N_j can be derived, from temporal coding, as being the weighted sum to be computed:

$$t_j = \frac{\vartheta}{\lambda} + \sum_{i \in \Gamma_j} \frac{w_{ij} \lambda_{ij}}{\lambda} \left(T_{in} - x_i + \Delta_{ij}\right) = T_{out} - \sum_{i \in \Gamma_j} \alpha_{ij} x_i = T_{out} - y_j \tag{7}$$

where $\lambda = \sum_{i \in \Gamma_j} w_{ij} \lambda_{ij}$ and $T_{out} = \frac{\vartheta}{\lambda} + \sum_{i \in \Gamma_j} \frac{w_{ij} \lambda_{ij}}{\lambda} \left(T_{in} + \Delta_{ij}\right)$ are input-independent values and where the neuron parameters are associated to the weights α_{ij} by $\alpha_{ij} = \frac{w_{ij} \lambda_{ij}}{\lambda}$. Note that the mandatory normalization of the α_i resulting from the latter formula can be circumvented by employing an auxiliary input neuron (cf. [93]). Therefore, every traditional neural network can be emulated by a SNN, in temporal coding.

[11] sometimes referred as "delay coding", "latency coding" or "firing order coding"

1.4.3 Echo State Networks (ESNs)

The basic definition of an **Echo State Network** (ESN) has been given by Jaeger in [74], with the idea to discovering a more efficient solution than BPTT, RTRL or EKF[12] for supervised training of recurrent neural networks. The author considers discrete-time neural networks with K input units, N internal units and L output units (Figure 10). Real-valued connection weights are collected in W^{in} for input weights, W for internal connections, W^{out} for the connections towards the output units, with optional matrix W^{back} for connections from the output units back to the internal units. The set of internal units, called DR, for "dynamical reservoir", is a recurrent neural network. DR is a pool of neurons - sigmoidal units in the first version, integrate-and-fire neurons later - that are randomly, and maybe sparsely, connected by a non-learnable weight matrix W.

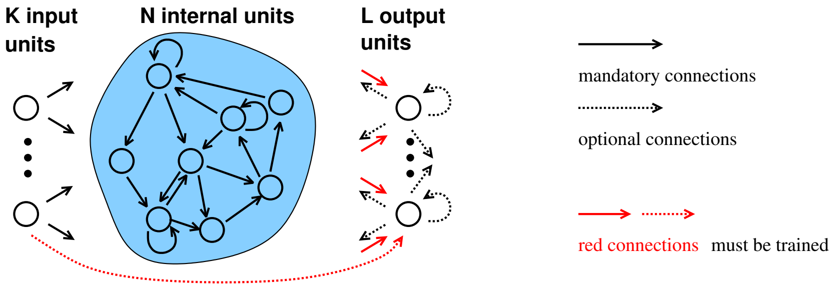


Figure 10: Architecture of an "Echo State Network". The blue central zone is the dynamical reservoir, DR.

From a teacher input / output time series $(\mathbf{u}(1), \mathbf{d}(1)), \ldots, (\mathbf{u}(T), \mathbf{d}(T))$, a trained ESN should outputs $\mathbf{y}(n)$ that approximates the teacher output $\mathbf{d}(n)$ when the ESN is driven by the training input $\mathbf{u}(n)$. The algorithm is as follows (for sigmoidal output units):

1. *Produce an echo state network,* i.e. construct a DR network which has the *echo state property*: for the ESN learning principle to work, the reservoir must asymptotically forget its input history. A necessary condition (seems to be sufficient... not yet theoretical proof!) is to choose a matrix W with a spectral radius $\mid \lambda_{max} \mid < 1$. Other matrices W^{in} and W^{back} can be generated randomly.

2. *Sample network training dynamics:* i.e. drive the network by presenting the teacher input $\mathbf{u}(n+1)$ and by teacher-forcing the teacher output $\mathbf{d}(n)$

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

   For each time larger than a washout time T_0, collect the network state $\mathbf{x}(n)$, row by row, in a matrix M and collect the sigmoid-inverted teacher output $tanh^{-1}\mathbf{d}(n)$ into a matrix T.

3. *Compute output weights* which minimize the training error. Any linear regression algorithm is convenient, such as LMS or RLS[13]. Concretely, W^{out} can result from multiplying the pseudoinverse of M by T to obtain $(W^{out})^t = M^{\dagger}T$.

4. *Exploitation:* ready for use, the network can be driven by novel input sequences $\mathbf{u}(n+1)$ with equations

$$\mathbf{x}(n+1) = f\left(W^{in}\mathbf{u}(n+1) + W\mathbf{x}(n) + W^{back}\mathbf{y}(n)\right) \tag{9}$$
$$\mathbf{y}(n+1) = f^{out}\left(W^{out}[\mathbf{u}(n+1), \mathbf{x}(n+1), \mathbf{y}(n)]\right) \tag{10}$$

Why "echo states"? The task, to combine $\mathbf{y}(n)$ from $\mathbf{x}(n)$, is solved by means (the internal states $x_i(n)$) which have been formed by the task itself (backprojection of $\mathbf{y}(n)$ in the DR). So, in intuitive terms, the target signal $\mathbf{y}(n)$ is re-constituted from its own echos $x_i(n)$ [75]. As stated by Jaeger, from the perspective of systems

[12] BPTT = Back-Propagation Through Time - RTRL = Real-Time Recurrent Learning - EKF = Extended Kalman Filtering
[13] LMS = Least Mean Squares - RLS = Recurrent Least Squares

engineering, the (unknown) system's dynamics is governed by $\mathbf{d}(n) = e\left(\mathbf{u}(n), \mathbf{u}(n-1), \ldots, \mathbf{d}(n-1), \mathbf{d}(n-2)\right)$ where e is a function of the previous inputs and system outputs. The task of finding a black-box model for an unknown system amounts to finding a good approximation to the system function e. The network output of a trained ESN with linear output units appears as a linear combination of the echo functions e_i:

$$\begin{aligned}
&e\left(\mathbf{u}(n), \mathbf{u}(n-1), \ldots, \mathbf{d}(n-1), \mathbf{d}(n-2)\right)\\
&= \mathbf{d}(n)\\
&\approx \mathbf{y}(n)\\
&= \sum w_i^{out} x_i(n)\\
&= \sum w_i^{out} e_i\left(\mathbf{u}(n), \mathbf{u}(n-1), \ldots, \mathbf{d}(n-1), \mathbf{d}(n-2)\right)
\end{aligned}$$

The basic idea of ESNs for black-box modelling can be condensed into the following statement [75]:

> *"Use an excitable system (the DR) to give a high-dimensional dynamical representation of the task input dynamics and / or output dynamics, and extract from this reservoir of task-related dynamics a suitable combination to make up the desired target signal."*

Many experiments have been tested successfully, with no larger networks than 20 to 400 internal units. Although the first design of ESN was for networks of sigmoid units, the similarity with Maass' approach (see just below) led Jaeger to introduce spiking neurons (LIF model) in the ESNs. Results are impressive, either in the task of generating a slow sinewave ($\mathbf{d}(n) = 1/5 \sin(n/100)$), hard or impossible to achieve with standard ESNs, but easy with a leaky integrator network [75], or in well mastering the benchmark task of learning the Mackey-Glass chaotic attractor [74].

Finally, ESNs are applicable in daily practice (many heuristics have been proposed for tuning the hyper-parameters - network size, spectral radius of W, scaling of inputs -) but, as confessed by Jaeger himself [76] in opening a special session dedicated to ESNs, at IJCNN'2005, the state of the art is clearly immature and, despite of many good results, there are still unexplained cases where ESNs work poorly.

1.4.4 Liquid State Machines (LSMs)

The **Liquid State Machine** (LSM) has been proposed by Maass, Natschläger and Markram [98] as a new framework for neural computation based on perturbations. The basic motivation was to explain how a continuous stream of inputs from a rapidly changing environment can be processed by stereotypical recurrent circuits of integrate-and-fire neurons in real time, as well as brain processes with biological neurons.

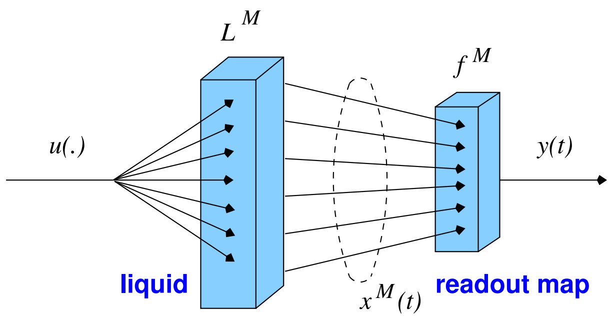


Figure 11: Architecture of a "Liquid State Machine". A continuous stream of values $u(.)$ is injected as input into the liquid filter L^M. A sufficiently complex excitable "liquid medium" creates, at time t, the *liquid state* $x^M(t)$, which is transformed by a memoryless *readout map* f^M to generate output $y(t)$.

Trying to capture a generic feature of neural microcircuits, which is their ability to carry out several real-time computations in parallel within the same circuitry, the authors have shown that a *readout neuron* (cf.

Figure 11) receiving inputs from hundreds or thousands of neurons in a neural microcircuit can learn to extract salient information from the high-dimensional transient states of the circuit and can transform transient circuit states into stable readouts [98]. 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.

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 *liquid neurons*), or even by real wet water in a bucket (!) as experimented by Fernando and Sojakka [41]. 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:

$$x^M(t) = \left(L^M(u)\right)(t) \tag{11}$$

$$y(t) = f^M\left(x^M(t)\right) \tag{12}$$

The readout map is usually task-specific. It can be implemented by one or several I&F neurons (the *readout neurons*) that can be trained to perform a specific task using, such as a linear regression or the *p-delta rule* [5], a perceptron-like local learning rule.

Why "liquid states"? The L^M 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, thus keeping a memory of past inputs in a dynamical system, making it possible to generate (role of readout units) stable and appropriately scaled output responses, even if the internal state never converges to a stable attractor.

Maass et al. proved that LSMs have a new form of universal computational power. More precisely, whereas Turing machines have universal computational power for off-line computation on discrete inputs, LSMs have (under conditions) universal computation power for real-time computing with fading memory on analog functions in continuous time. A theorem [98] guarantees that LSMs have this computational power, provided that:

1. the class of basis filters that composes the liquid filters L^M satisfies the point-wise *separation property*,
2. the class of functions from which the readout maps f^M are drawn satisfies the *approximation property*.

(SP), the *separation property*, addresses the amount of separation between the trajectories of internal states of the system that are caused by two different input streams, whereas (AP), the *approximation property*, addresses the capability of the readout mechanisms to distinguish and transform different internal states of the liquid into given target outputs (see Appendix A of [98] for exact definitions). Two versions of the universal approximation theorem hold, one for inputs that are continuous functions of time (e.g. time series) and the other where the inputs are finite or infinite spike trains (e.g. input spiking neurons).

Although the LSM model is not yet mature (in spite of some theoretical bases), it has been successfully applied to several benchmark non-linear problems such as the XOR [41], the Hopfield and Brody [64] (see Section 5) "zero-one" discrimination [41, 98], or to a task of texture recognition from artificial whiskers [167]. Subject to deeper understanding their underlying mechanisms and better mastering their implementation heuristics, LSMs should be efficient tools, especially for time series prediction and for temporal pattern recognition.

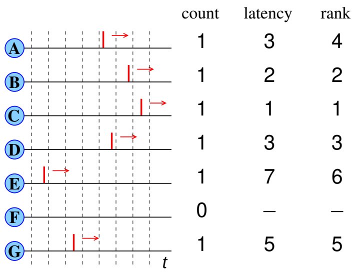
Finally, LSMs and ESNs are very similar models of recurrent neural networks that promise to be convenient for both exploiting and capturing most temporal features of spiking neuron processing, with a slight advantage to LSMs that have been inspired from the background idea to modelling dynamical and representational phenomena in biological neural networks. However both models are good candidates for engineering applications whenever temporally changing information has to be processed.

2 Computational power of neurons and networks

Since information processing with spiking neurons is based on the time of spike emissions (pulse coding) rather than the average numbers of spikes in a given time window (rate coding), two straightforward advantages of SNN processing are a possibility to very fast decoding sensory information, as in human visual system [155] (very precious for real-time signal processing), and a possibility to multiplexing information, e.g. like the auditory system combines amplitude and frequency very efficiently over one channel. Moreover, SNNs add a new dimension, the temporal axis, to the representation capacity and the processing abilities of neural networks. The present section proposes different attempts to grasp the new forms of computational power of SNNs and to surround how the paradigm of spiking neuron is a way to tremendously increase the computational power of neural networks. How to control and exploit the full capacity of this new generation of models raises many fascinating challenging questions that will be addressed in further sections (3, 4, 5).

2.1 Combinatorial point of view

Encouraging estimation of SNNs information coding ability can be drawn from considering the encoding capacity of neurons within a small set. The representational power of alternative coding schemes has been pointed out by Recce [131] and analysed by Thorpe et al. [154].



Numeric examples:	count code	binary code	timing code	rank order
left (opposite) figure $n = 7, T = 7ms$	3	7	≈ 19	12.3
Thorpe et al. [154] $n = 10, T = 10ms$	3.6	10	≈ 33	21.8

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

Figure 12: Comparing the representational power of spiking neurons, for different coding schemes. Count code: 6/7 spike per $7ms$, i.e. $\approx$ 122 spikes.s^{-1} - Binary code: 1111101 - Timing code: latency, here with a $1ms$ precision - Rank order code: $C > B > D > A > G > E > F$.

For instance, consider that a stimulus has been presented to a set of n neurons and that each of them emits at most one spike in the next $T(ms)$ time window (Figure 12). Let us discuss different ways to decode the temporal information that can be transmitted by the n neurons. If the code is to *count* the overall number of spikes emitted 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 case of *binary code*, the output is a n-digits binary number, with obviously n as information coding capacity. The maximum amount of information comes from *timing code*, provided an efficiently decoding mechanism is available for determining the precise times of each spike. In practical cases, the available code size depends on the latency, e.g. for a $1ms$ precision, an amount of information of $n \times log_2(T)$ can be transmitted in the T time window. Finally, in *rank order coding*, information is the order of the sequence of spike emissions, i.e. one among the $n!$ orders that can be obtained from n neurons, thus $\log_2(n!)$ bits can be transmitted.

More impressive estimation can be drawn from an opposite point of view, where large networks of spiking neurons (millions or billions of neurons and synapses) are considered as a whole. In first rough estimation, at each time t, the amount of information that can be encoded by a network of N spiking neurons has the combinatorial power $N!$ i.e. an exponential order of magnitude $O(e^{N \log N})$. This estimation results from guessing that a given information is encoded by the synchrony of a specific transient neural assembly, and counting that $N!$ different cell assemblies can be built from N neurons. Moreover, several assemblies can be simultaneously activated at any time, which again increases the potential power of SNNs for coding information. Such a statement could be, of course, inflected by deeper understanding of how cell assemblies work: The question is one of the burning issues in SNN research today and it is worth detailing the most promising research tracks.

2.2 "Infinite" capacity of cell assemblies

The concept of **cell assembly** has been already introduced by Hebb [57], more than half a century ago! However the idea had not been further developed, neither by neurobiologists - since they could not record the activity of more than one neuron at a time, until recently - nor by computer scientists. New techniques of brain imaging and recording have boosted this area of research in neuroscience for only a few years (cf. special issue 2003 of *Theory in Biosciences* [170]). 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 [53], who 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 cells 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. In this context, short term memory would be a persistent activity maintained by reverberations in assemblies, whereas long term memory would correspond to the formation of new assemblies, e.g. by a Hebb's rule mechanism. Inherited from Hebb, current thinkings about cell assemblies are that they could play a role of "grandmother neural groups" as basis of memory encoding, instead of the old debated notion of "grandmother cell", and that material entities (e.g. a book, a cup, a dog) and, even more, ideas (mental entities) could be represented by cell assemblies.

As related work, deep attention has been paid to synchronization of firing times for subsets of neurons inside a network. The notion of **synfire chain**, a pool of neurons firing synchronously, has been developed by Abeles [3]. Hopfield and Brody have studied how **transient synchrony** could be a collective mechanism for spatiotemporal integration [64, 65]. They tuned a network of spiking neurons (I&F), with an ad-hoc topology, for a multispeaker spoken digit recognition task (cf. Section 5), and they analysed the underlying computational principles. They show that, based on the variety of decay rates, the fundamental recognition event is the occurence of transient synchronization in pools of neurons with convergent firing rates. They claim that the resulting collective synchronization event is a basic computational building block, at the network level, for computing the operation: *Many variables are currently approximately equal*, with no resemblance in traditional computing. Related to other work, their method could be a way to control the formation of transient cell assemblies that synchronize selectively in response to specific spatiotemporal patterns.

However synchronization, even transient synchrony, appears to be a too restrictive notion for well understanding the full power of cell assemblies processing. This point has been well understood by Izhikevich who proposes the notion of **polychronization** [71]. From the simulation of a network of 1000 randomly connected (with 0.1 probability) Izhikevich's neurons (see Section 1.3.2), with fixed but different delays and weights adaptation by STDP[14] for the 80% excitatory neurons, he derived the emergence of over 5000 polychronous groups of neurons. Based on the connectivity between neurons, a polychronous group is a possible stereotypical time-locked firing pattern. Since its neurons have matching axonal conduction delays, the group can be activated more often than predicted by chance: Firing of the first few neurons with the right timing is enough to activate most of the group, w.r.t. the general network activity at a gamma rythm. Since any given neuron can be activated within several polychronous groups, at different times, 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 an unprecedented information capacity. Note that, due to STDP mechanism, the weights of excitatory connections vary, thus resulting in formation and disruption of groups, with a core of long lifetime survival groups. External inputs can drive the formation of groups that are activated when the stimulus is present. Hence, polychronous groups could represent memories and experience, and could be viewed as a computational implementation of cell assemblies.

[14]STDP = Spike Time Dependent Plasticity - see Section 3.

2.3 Complexity results

Since 1997, Maass [94, 95] has quoted that computation and learning has to procede quite differently in SNNs. He proposes to classify neural networks as follows

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

Maass also proposed a simplified model of spiking neuron with rectangular shape of EPSP, the "type_A spiking neuron" (Figure 13) that has been the support for many complexity results. A justification of the type_A neuron model is that it provides a link to silicon implementations of spiking neurons in analog VLSI. A fundamental point is that different transmision delays d_{ij} can be assigned to different presynaptic neurons N_i connected to a postsynaptic neuron N_j.

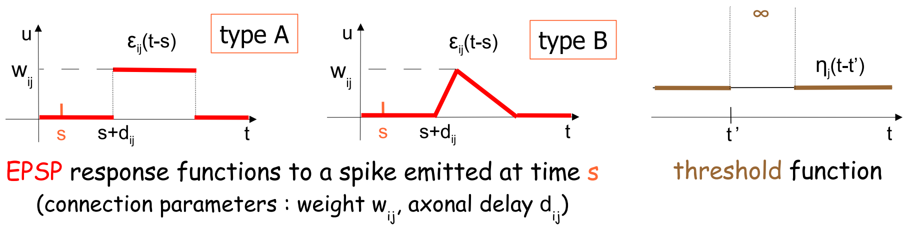


Figure 13: Very simple versions of spiking neurons: "type_A spiking neuron" (rectangular shaped pulse) and "type_B spiking neuron" (triangular shaped pulse), with elementary representation of refractoriness (threshold goes to infinity), as defined in [94].

Boolean input vectors $(x_1, \ldots, x_n)$ are 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 [94, 140]. Since spiking neurons are able to behave as coincidence detectors, it is straightforward to prove that the boolean funcion 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}):

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

However, the boolean function CD_n requires at least $\frac{n}{log(n+1)}$ threshold gates and at least $\Omega(n^{1/4})$ sigmoidal units to be computed.

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

Real-valued inputs $(x_1, \ldots, x_n)$ are presented to a spiking neuron by a set of input neurons $(N_1, \ldots, N_n)$ such that N_i fires at time $T_{in} - cx_i$ (cf. temporal coding, defined in Section 1.4.2). With positive real-valued inputs and a binary output, the function ED_n *(Element Distinctness function)* can be computed by a single type_A neuron, whereas at least $\Omega(nlog(n))$ threshold gates and at least $\frac{n-4}{2} - 1$ sigmoidal hidden units are required.

But for arbitrary real-valued inputs, type_A neurons are no longer able to compute threshold circuits. Hence the "type_B spiking neuron" (Figure 13) has been proposed since a triangular EPSP is able to shift the firing time in a continuous manner. Any threshold gate can be computed by $O(1)$ type_B spiking neurons. And, at the network level, any threshold circuit with s gates, for real-valued inputs $x_i \in [0,1]^n$ can be simulated by a network of $O(s)$ type_B spiking neurons.

All the above results drive Maass to conclude that spiking neuron networks are 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 [140, 99]. Results are:

- Every boolean function of n variables, computable by a 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 $\Theta(n)$.
- $(\forall n \geq 2)$ $\exists$ 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 $n-1$.
- $(\forall n \geq 2)$ $\exists$ a boolean function on n variables that has threshold order 2 and cannot be computed by a spiking neuron.

In [95], Maass considers **noisy spiking neurons**, a neuron model close to SRM (cf. Section 1.3.3), with a probability of *spontaneous* firing (even under threshold) or not firing (even above threshold) governed by the difference

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

The main result is that: For any given $\epsilon, \delta > 0$ one can simulate any given feedforward sigmoidal neural network $\mathcal{N}$ of s units with linear saturated activation function by a network $\mathcal{N}_{\epsilon,\delta}$ of $s + O(1)$ noisy spiking neurons, in temporal coding. As immediate consequence 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 $\epsilon > 0$ with arbitrarily high reliability, in temporal coding, by a network of noisy spiking neurons with a single hidden layer. As by-product, such a computation can be achieved within $20ms$ for biologically realistic values of spiking neuron time-constants.

Beyond this number of encouraging results, and others, Maass [95] points out that SNNs are able to encode time series in spike trains, but we have, in computational complexity theory, no standard reference models for analyzing computations on time series.

2.4 Learnability

Probably the first attempt (in 1996) to estimate the VC-dimension of spiking neurons is a work of Zador and Pearlmutter [172] who studied a family of integrate-and-fire neurons (cf. Section 1.3.2) with threshold and time-constants as parameters. They proved that the VC_{dim}(I&F) grows in $logB$ with the input signal bandwith B, which means that the VC_{dim} of a signal with infinite bandwith is unbounded, but the divergence to infinity is weak (logarithmic).

More conventional approaches [99, 95] consist in estimating bounds on the VC-dimension of neurons as functions of their programmable / learnable parameters that can be 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 $\Omega(m)$ only.

- 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, $pseudo_{dim}$(type_A neuron) is $O(n\,log(n))$.

Hence, the learning complexity of a single spiking neuron is higher than the learning complexity of a single threshold gate. As argumented by Maass and Schmitt [100], this should not be interpreted as saying that supervised learning is impossible for a spiking neuron, but it should become quite difficult to formulate rigorously provable learning results for spiking neurons. For summarizing, if the class of boolean functions, with n inputs and 1 output, that can be computed by a spiking neuron is 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 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 $\Theta(n\,log(n))$

Denoting by $\mathcal{T}_n^{bb}$ the class of boolean functions computable by a threshold gate, by μ_DNF_n the class of boolean functions definable by a DNF formula where each of the n variables occurs at most once, and by OR_of_$O(n)$_$\mathcal{T}_n^{bb}$ the class of boolean functions computable by a disjunction of $O(n)$ threshold gates, Schmitt and Maass summarize graphically (Figure 14) the bounds they proved, in the boolean domain [100].

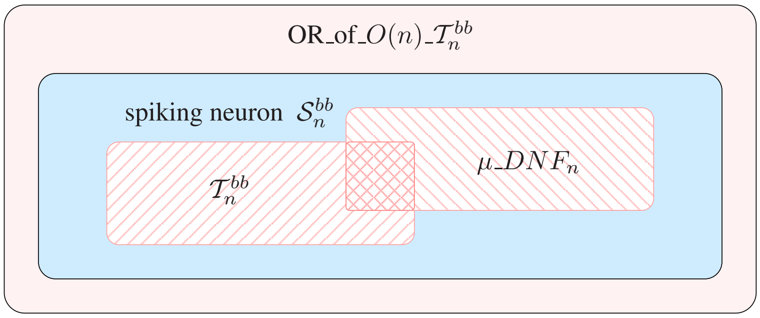


Figure 14: Upper and lower bounds for the computational power of a type_A spiking neuron in the boolean domain. (From [100])

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 a NN[15] 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[16]. Then $VC_{dim}(\mathcal{N})$ is $O(E^2)$, even for analog coding of the inputs [100]. In [141], Schmitt derives more precise results by considering a feedforward architecture of depth D, with nonlinear synaptic interactions between neurons:

- The pseudo-dimension of an SNN with W parameters (weights, delays, ...), depth D and rational synaptic interactions with degree no larger than p, is $O(WDlog(WDp))$. For fixed depth D and degree p, this entails the bound $O(Wlog(W))$.
- The pseudo-dimension of an SNN with W parameters and rational synaptic interactions with degree no larger than p, but with arbitrary depth, is bounded by $O(W^2 log(p))$. The pseudo-dimension is $\Theta(W^2)$ if the degree p is bounded by a constant.

It follows that the sample sizes required for the networks of fixed depth are not significantly larger than for classic 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 [141].

[15]NN = traditional Neural Networks
[16]The $\mathcal{N}$ directed acyclic graph is the network topology that underlies the spiking neuron network dynamics.

In the framework of PAC-learnability [161, 18], considering that only hypotheses from $\mathcal{S}_n^{bb}$ may be used by the learner, the computational complexity of training a spiking neuron can be analyzed within the formulation of the consistency or loading problem (cf. [80]): Given a training set T of labelled binary examples (X, b) with n inputs, does there exist parameters defining a neuron $\mathcal{N}$ in $\mathcal{S}_n^{bb}$ such that $(\forall (X, b) \in T)\ y_{\mathcal{N}} = b$? The following results are proved in [100]:

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

Extended results have been proposed recently by Šíma and Sgall [147]:

- The *consistency problem* for a spiking neuron with nonnegative delays (i.e. $d_{ij} \in \mathbb{R}^+$) is $NP\_complete$. The result holds even with some restrictions (see [147] for precise conditions) on bounded delays, unit weights or fixed threshold.
- A single spiking neuron $\mathcal{N}$ 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.
- The *representation problem* $^{(*)}$ for spiking neurons is $coNP\_hard$ and belongs[17] to Σ_2^p

$^{(*)}$ where the representation problem is defined by giving a boolean function of n variables, in DNF form, and looking for a spiking neuron able to compute it.

Nonlearnability results had been derived for classic NNs already [17, 80]. Nonlearnability results for SNNs should not curb the research of appropriate learning algorithms for SNNs. All the results stated in the present section are based on very restrictive models of SNNs and, apart the programmation of transmission delays of synaptic connections, they do not exploit all the capabilities of SNNs that could result from computational units based on firing times. Such a restriction can be explained by a lack of practice for building proves in such a context or, even more, by an incomplete and not adapted computational / learnable complexity theory. Indeed, learning in biological neural systems may employ rather different mechanisms and algorithms than usual computational learning systems. Therefore, several characteristics, especially the features related to computing in continuously changing time, will have to be fundamentally rethought for discovering efficient learning algorithms and ad-hoc theoretical models to understand and conquer the computational power of SNNs.

3 Learning in spiking neuron networks

A rough sketch of the multidisciplinary research tackling the question of learning and memorizing in spiking neuron networks can be presented as follows:

- Early work (end of the 90's) has been to find solutions for emulating traditional learning rules in SNNs (quickly presented in Section 3.1).
- Afterwards, new tracks have been searched in different ways to exploit the current knowledge about synaptic plasticity (Section 3.2)
- Recent studies propose computational justifications for plasticity-based learning rules and other tracks for efficient learning in SNNs (Section 3.3).

However, the old dilemma *plasticity / stability* (underlined by Grossberg since the 80's) reappears with enhanced intensity when learning with ongoing weights. The question is discussed in Section 3.4, with several approaches proposed to circumvent the problem.

[17] Σ_2^p is a complexity class from the polynomial time hierarchy (see [8])

3.1 Simulation of traditional models

Maass and Natschläger [97] propose a theoretical model for emulating arbitrary Hopfield networks in temporal coding. Maass [93] studies a "relatively realistic" mathematical model for biological neurons that can simulate arbitrary feedforward sigmoidal neural networks. Emphasize is put on the fast computation time that depends only on the number of layers of the sigmoidal network, no longer on the number of neurons or weigths. However, for the need of theoretical results, the model makes use of static reference times T_{in} and T_{out} and auxiliary neurons. Even if such artefacts can be withdrawn in practical computation, the method rather appears as an artificial attempt to make SNNs computing like traditional neural networks, without taking advantage of SNNs intrinsic abilities to computing with time. Similar techniques, still based on temporal coding (see Section 1.4.2), have been applied by Natschläger and Ruf to clustering RBF networks [115, 114] and by Ruf and Schmitt to Kohonen's self-organizing maps [136]. Therefore, SNNs are validated as universal approximators, and traditional supervised and unsupervised learning rules can be applied for training the synaptic weights.

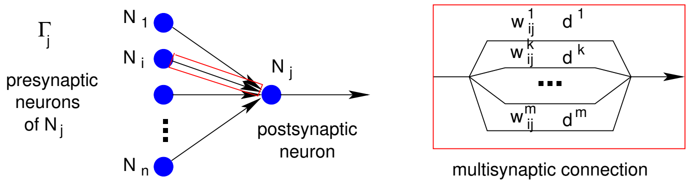


Figure 15: Any single connection can be considered as being multisynaptic, with random weights and a set of increasing delays, as defined in [115].

Although programmable delays have been pointed out for their interest in complexity analyses of SNNs computational power (Section 2.4), surprisingly synaptic transmission delays are fixed in these models, except in the RBF spiking network. Starting from an idea suggested by Hopfield in [63], Natschläger and Ruf [115, 114] exploit the possibility of a spiking neuron to behave as an RBF neuron and they consider multisynaptic connections (Figure 15). A set of delays $(d^1, \ldots, d^m)$ with fixed values is applied to m connections with random weights w_{ij}^m from a presynaptic neuron N_i towards a postsynaptic neuron N_j. The same idea has been extended to multilayer RBF networks by Bohte, La Poutré and Kok [21]. Breaking with the common representation of input data by temporal coding (as defined in Section 1.4.2), they propose to code each component of an input vector X by means of an array of spiking neurons with overlapping gaussian receptive fields (Figure 16). The higher the k^{th} receptive field is stimulated, the earlier the firing time of the k^{th} neuron. In addition to more biological plausibility, such a population coding helps their architecture of spiking RBF network to learn unsupervised clustering and, for instance of application, to classify features in color land images.

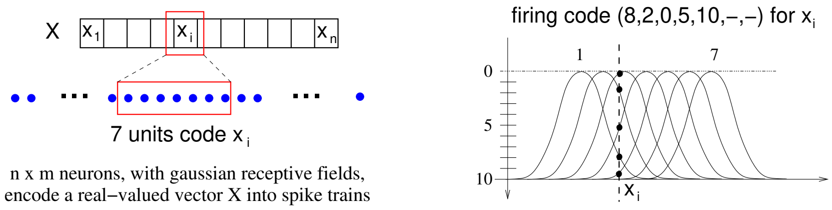


Figure 16: Each component of an input vector of real values can be encoded into a spike train by several neurons implementing overlapping gaussian receptive fields.

The same authors defined **SpikeProp**, a supervised learning rule adapted from error backpropagation to feedforward SNNs [20]. Applying the coarse coding of Figure 16 to usual benchmark databases, they show that complex, non-linear tasks can be learned with performance similar to traditional neural networks. In the range of work governed by the idea to transfer traditional NN expertise to SNN computation, a reinforcement learning rule [171] can be cited. The validity of the rule, based on a gradient descent of expected reward signals, strongly depends on the way of modelling the irregular spiking process of the neurons by a Poisson process.

3.2 Synaptic plasticity and STDP

From the early work presented by Hebb in 1949 [57], **synaptic plasticity** has been the main basis of learning rules by weight updating in artificial neural networks. However Hebb's ideas are poorly exploited by most of the current algorithms. In the context of SNNs these ideas have been revisited and the original sentence is often cited in recent articles:

> *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.*

Novel tracks for setting algorithms that control the synaptic plasticity are derived both from a deeper understanding of Hebb's lesson and from a bank of recent results in neuroscience, following the advances of experimental technology. Innovative principles are often qualified of **temporal Hebbian rules**. In the biological context of natural neurons, the changes of synaptic weights with effects lasting several hours are referred as LTP[18] if the weight values (also named *efficacies*) are strengthened, and LTD if the weight values are decreased. In the second or minute timescale, the weight changes are designed by STP and STD[19].

A good review of the main synaptic plasticity mechanisms for regulating levels of activity in conjunction with Hebbian synaptic modification has been developed by Abbott and Nelson in [2]:

- *Synaptic scaling:* Neurons (e.g. in the cortex) actively maintain an average firing rate by scaling their incoming weights. Synaptic scaling is multiplicative, in the sense that synaptic weights are changed by an amount proportional to their strength, and not all by the same amount (additive / subtractive adjustment). Synaptic scaling, in combination with basic Hebbian plasticity, seems to implement a synaptic modification comparable to Oja's rule [117] that generates, for simple neuron models, an input selectivity related to Principal Component Analysis (PCA).

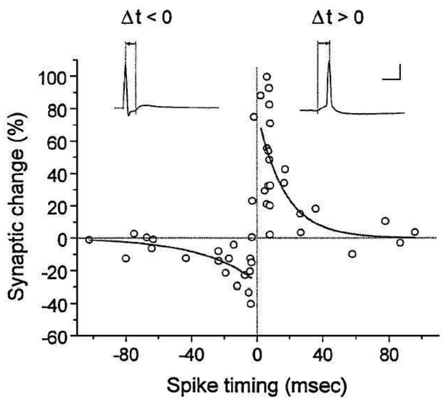
- *Synaptic redistribution:* Markram and Tsodyks' experiments [104] have critically challenged the conventional assumption that LTP reflects a general gain increase. The phenomenon of "Redistribution of Synaptic Efficacy" (RSE) designs the change in frequency dependence they have observed during synaptic potentiation. Synaptic redistribution could enhance the amplitude of synaptic transmission for the first spikes in a sequence, but with transient effect only. Possible positive consequences of RSE in the context of learning in neural networks is discussed relatively to the ART[20] model in [28].

- *Spike-timing dependent synaptic plasticity[21]:* **STDP** is far from beeing the most popular synaptic plasticity rule for a few years (first related articles [103, 13, 82]). STDP is a form of Hebbian synaptic plasticity sensitive to the precise timing of spike emission. It relies on local information driven by backpropagation of action potential (BPAP[22]) 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 an experiment to another, in neuroscience, a basic computational principle has emerged: A maximal increase of synaptic efficacy occurs on a connection when the presynaptic neuron fires a short time before the postsynaptic neuron, whereas a late presynaptic spike (just after the postsynaptic firing) leads to decrease the weight. If the two spikes (pre- and post-) are too much distant in time, then the weight leaves unchanged. This form of LTP / LTD timing dependency reflects a form of causal relationship in information transmission through action potentials.

As underlined by Abbott and Nelson, "STDP can act as a learning mechanism for generating neural responses selective to input timing, order and sequence. In general, STDP greatly expands the capability of Hebbian learning to address temporally sensitive computational tasks".

[18] LTP = Long Term Potentiation - LTD = Long Term Depression
[19] STP = Short Term Potentiation - STD = Short Term Depression
[20] ART = Adaptive Resonance Theory - see [27] for a definition of the Carpenter & Grossberg's model
[21] STDP = Spike-Time Dependent Plasticity
[22] BPAP = Back-Propagated Action Potential

The most commonly used temporal windows for controlling the weight LTP and LTD (Figure 17) are derived from the experiments performed by Bi and Poo in cultures of rat hippocampal neurons [13, 14].



The spike timing (in abscissa) is the difference $\Delta t = t_{post} - t_{pre}$ of firing times between the pre- and postsynaptic neurons. The synaptic weight is increased when the presynaptic spike is supposed to have a causal influence on the postsynaptic spike, i.e. when $\Delta t > 0$ and close to zero. The synaptic change ΔW, indicated in percentage, operates in a multiplicative way on the weight update (Y-axis).

Circles are real data recorded on a preparation (culture of rat hippocampal neurons) and numerical values result from biological experiments.

Figure 17: STDP window for synaptic modifications. LTP and LTD induced by correlated pre- and postsynaptic spiking at synapses between hippocampal glutamatergic neurons in culture [From Bi and Poo, 2001 [14]].

The modification ΔW is applied to a weight w_{ij}, according to either a multiplicative or an additive formula. Different shapes of STDP windows have been used in recent literature [103, 82, 149, 143, 26, 73, 84, 52, 116, 72, 138, 109, 111]. The main differences concern the symmetry or asymmetry of the LTP and LTD subwindows, and the discontinuity or not of ΔW function of Δt, near $\Delta t = 0$ (cf. Figure 18). Variants also exist in the meaning of the X-axis, now $t_{post} - t_{pre}$, now $t_{pre} - t_{post}$, without conventional representation, unfortunately. Note that STDP windows are often different for excitatory and inhibitory connections (Figure 17 and windows 1-3 on Figure 18 are applied to excitatory weights). For inhibitory synaptic connections, it is common to use a standard Hebbian rule, just strengthening the efficacy when the pre- and postsynaptic spikes occur close in time, whatever the sign of the difference $t_{post} - t_{pre}$ (cf. 4, most right on Figure 18).

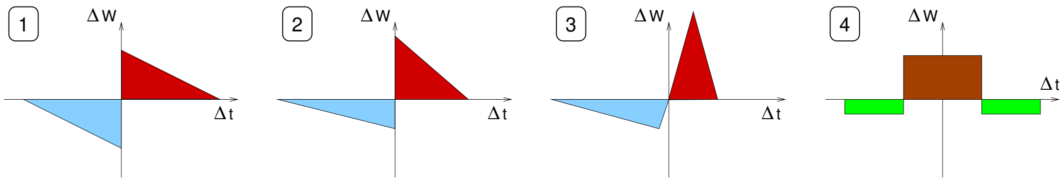


Figure 18: Various shapes of STDP windows, in $\Delta t = t_{post} - t_{pre}$ representation, with LTP in blue and LTD in red for excitatory connections (1 to 3). More realistic and smooth ΔW function of Δt are mathematically described by sharp rising slope near $\Delta t = 0$ and fast exponential decrease (or increase) towards $\pm\infty$. Standard Hebbian rule (4) with brown LTP and green LTD, usually applied to inhibitory connections.

STDP appears as a possible new basis for investigating innovative learning rules in SNNs. However a lot of questions arise and many problems remain unsolved. For instance, weight modifications according to STDP windows cannot be applied repeatedly in the same direction (e.g. always potentiation) without fixing bounds for the weight values, e.g. an arbitrary fixed range $[0, w_{max}]$ for excitatory synapses. Bounding both the weight increase and decrease is necessary for avoiding the network overall activity to fall into a silent state (all weights down) or an epileptic state (all weights up, disordered and frequent firing of allmost all the neurons), but in many STDP driven SNN models, a saturation of the weight values to 0 or w_{max} has been observed, which risk to convey reduced plasticity available for further adaptation of the network to new events to be learned. A regulatory mechanism, based on a triplet of spikes, has been described by Nowotny et al. [116], for a smooth version of the temporal window 3 of Figure 18, with an additive STDP learning rule.

A theoretical study of STDP learnability has been published in Autumn 2005 by Legenstein, Näger and Maass [91]. They define a *Spiking Neuron Convergence Conjecture* (SNCC) and compare the behaviour of STDP learning by teacher forcing with the Perceptron convergence theorem. They state that a spiking neuron can learn with STDP basically any map from input to output spike trains that it could possibly implement in a stable manner. They interpret the result as saying that STDP endows spiking neurons with **universal learning capabilities** for Poisson input spike trains. Beside this encouraging learnability result, recent work propose computational justifications for STDP as a new learning paradigm (next section).

3.3 Computational learning theory

As well summarized by Bohte and Mozer in [22], the numerous models developed around the different aspects of STDP can be classify in three groups:

- A number of studies focus on biochemical models that explain the underlying mechanisms giving rise to STDP [143, 12, 81, 127, 138] or that start from STDP mechanisms to derive explanations for various aspects of brain processing. ← This topic is not developed in the present survey since it is more related to neuroscience than to computer science.
- Many reseachers have also focused on models that explore the consequences of STDP-like learning rules in an ensemble of spiking neurons [51, 82, 149, 162, 130, 83, 72, 1, 144, 91, 111]. ← For further information, see [128] where Porr and Wörgötter develop an extensive review.
- A recent trend is to propose models that provide fundamental computational justifications for current models of synaptic plasticity, mainly for STDP [10, 30, 123, 11, 157, 156, 124, 22, 125]. ← This latter area of research is developed below, in present section.

For instance of STDP-like learning rule, Rao and Sejnowski [130] show that STDP in neocortical synapses can be interpreted as a form of temporal difference (TD), as defined by Sutton [152], for prediction of unput sequences. They show that a TD rule used in conjunction with dendritic BPAPs reproduces the temporally asymmetric window of Hebbian plasticity. They widely discuss possible biophysical mechanisms for implementing the TD rule.

In the scope of machine learning approach, Barber [10] proposes a statistical framework for adressing the question of learning in SNNs, in order to derive optimal algorithms as consequences of statistical learning criteria. Optimality is achieved w.r.t. a given neural dynamics and an assumed desired functionality (e.g. learning temporal sequences). The principle consists in adding hidden variables $\boldsymbol{h}(t)$ to the neural firing states $\boldsymbol{v}(t) = (v_i(t))_{i=1,\dots,V}$ of a fixed set of V neurons in a given time window $\{1, ..., T\}$. The temporal sequence $\mathcal{V}$ is represented in discrete time ($\Delta t = 1$) by boolean values: If neuron i spikes at time t then $v_i(t) = 1$ else $v_i(t) = 0$. Hence the model appears as a special case of Dynamic Bayesian networks with deterministic hidden variables updating (see [9]). Learning the sequence $\mathcal{V} = (\boldsymbol{v}(t))_{t=1,\dots,T}$ results from **maximizing the log-likelihood** $L(\theta_{\boldsymbol{v}}, \theta_{\boldsymbol{h}} \mid \mathcal{V})$ where $\theta_{\boldsymbol{v}}$ and $\theta_{\boldsymbol{h}}$ are the model parameters. Learning can be carried out by forward propagation through time, with the gradient ascent formula $\theta \leftarrow \theta + \eta \frac{\partial L}{\partial \theta}$. Batch learning rules are derived in case of a stochastically firing neuron, a simple model of threshold unit and a LIF[23] model is analyzed through the same principle. An extension of this work to continuous time (by taking the limit $\Delta t \to 0$) can be found in [123] where Pfister, Barber and Gerstner consider only one connection (weight w) between a pre- and a postsynaptic SRM_0[24] neurons. Assuming that the instantaneous firing rate of the postsynaptic neuron $\rho(t) = g(u(t))$ depends on the membrane potential $u(t)$ through an increasing function g, they search to maximize the probability that the postsynaptic spike train has been generated by the firing rate $\rho(t)$. They define the log-likelihood $\mathcal{L}(t_i \mid u(s))$ of the postsynaptic spike train, given the membrane potential, and derive a learning rule that tends to optimise the weight w in order to maximize the likelihood $w \leftarrow w + \frac{\partial \mathcal{L}}{\partial w}$ (gradient ascent). The interesting result has been to observe that, after simplifications and only in case of positive back-propagating

[23] LIF = Leaky Integrate and Fire (see Section 1.3.2)
[24] SRM = Spike Response Model (see Section 1.3.3)

action potential amplitude[25], the learning window obtained by this method presents a similar shape to the STDP window obtained by Bi and Poo (cf. Figure 17), but with a negative offset that is discussed in the paper.

Other approaches are based on the **maximization of mutual information**. Following Chechik [30], in [157], Toyoizumi, Pfister, Aihara and Gerstner ask what is the optimal synaptic update rule so as to maximize the mutual information between pre- and postsynaptic neurons. Considering a single postsynaptic neuron that receive input spike trains from N presynaptic neurons (modelled by independent Poisson spike trains), Chechik derives a learning algorithm that changes the weights by maximizing the input-output mutual information, so that the neural system be able to extract relevant information, instead of simply reproducing the representation of the inputs. The learning window resulting from his method presents only a positive part depending on the postsynaptic potential, but a flat negative part. Starting from a stochastically spiking neuron model with refractoriness, Toyoizumi et al. derive an online weight update rule that share properties with STDP, in particular a biphasic dependence upon the relative timing of pre- and postsynaptic spikes. They argue that the negative part of the learning window results from modelling the neuronal refractoriness.

An older model of synaptic LTP and LTD had been proposed by Bienenstock, Cooper and Munro in 1982, as a theory for the development of neuron selectivity [16]. Sometimes considered a precursory version of synaptic plasticity rule, the **BCM model** is a mathematical model presented as a mechanism of synaptic modification that results in a temporal competition between input patterns. A strong result of the BCM theory is a proof that once a neuron (defined by its synaptic efficacies) has reached a certain selectivity, it cannot switch to another selective region. The separation between selective regions depends on a threshold function of the average value of the postsynaptic firing rate. The BCM model has been related to STDP by Izhikevich and Desai who show in [72] that an application of the STDP rule restricted to only nearest-neighbour pairs of spikes, assuming Poisson spike trains, leads to a BCM-like stability diagram. Toyoizumi, Pfister, Aihara and Gerstner [156], starting from similar model and approach as in [157], show that, under the assumption of Poisson firing statistics, the synaptic update rule exhibits all the features of the Bienenstock-Cooper-Munro rule (BCM), in particular, regimes of synaptic potentiation and depression separated by a sliding threshold. For their model of stochastically spiking neuron with refractoriness, they calculate the probabilistic relation between an output spike train and an ensemble of input spike trains. Then they derive how the weights must change for maximizing information transmission under the constraint to be close to a fixed target firing rate. The resulting (by gradient ascent) online update rule depends on the state of the postsynaptic neuron (could account for homeostatic process), in addition to the usual measure of correlation between pre- and postsynaptic activity. The resulting control mechanism corresponds to the sliding threshold of the BCM rule, thus extending the BCM model to the case of spiking neurons. Artificial toy experiments of pattern discrimination (Poisson spike trains), rate modulation and spike-spike correlations are proposed as model applications.

Two of three not yet published articles deal with related questions or apply similar methods. In [124], Pfister and Gerstner propose a minimalist learning rule with only five parameters and study the effects of spike triplets (1 pre- and 2 postsynaptic spikes). They discuss their model in relation to the BCM rule and to other timing-based rules. Pfister, Toyoizumi, Barber and Gerstner, in [125], examine the ideal form of a STDP function for generating action potentials of the postsynaptic neuron with high temporal precision. With supervised learning in mind, they maximize the likelihood of emitting 1 desired postsynaptic spike train for a given set of N input spike trains. They derive learning rules for three different conditions imposed to the postsynaptic neuron behaviour and compare with STDP. A link to reinforcement learning is proposed, based on the Xie and Seung's approach (cf. end of Section 3.1). The third (already known) article to appear in 2006 is [22] where Bohte and Mozer propose a slightly different approach by **minimizing the entropy** of the postsynaptic neuron output in response to a spike pattern, incoming from several presynaptic neurons (with stochastic SRM neurons). Their goal is to reduce response variability to a given input pattern. They derive learning rule and obtain a robust modelling of STDP, both LTP and LTD curves (unlike Bell and Parga [11]). They compare and contrast their model to [157] and [11] and explain their contradictory result concerning the role of refractoriness

[25]BPAP amplitude is the constant η_0 of the detailed expression of the kernel function η, following formula 4 in Section 1.3.3.

as responsible for LTD in questioning the validity of some analytical approximations in [157]. However, it is hard to make a synthesis and to stand back from so much recent ideas and results, but their presentation helps to feel the effervescence of ongoing research on the burning topic of learning in SNNs.

3.4 Permanent memory storage

Therefore, synaptic plasticity, especially STDP, appears to be the main basis for defining new learning rules that match the temporal characteristics of spiking neuron networks. However, the major issue is to find a balance between permanently switching synaptic weights and ensuring long-lasting memory. The possibility to solve this problem from the results developed in previous sections (even in Section 3.3) is not straightforward.

As discussed by Fusi, Drew and Abbott [46], the challenge is to protect the memory trace from the ravage of ongoing activity (not from the ravage of time), in the case where new experiences are continually generating new memories. Studying a signal to noise ratio, the authors highlight that forgetting must occur, but according to a power-law rather than exponential dynamics. So for, they propose a cascade model of synaptically stored memories, with different levels of plasticity connected by metaplastic transitions.

The question is addressed another way by Hopfield and Brody [66] who study a self-repair rule in spiking neuron networks in order to prevent from degradation a connectivity pattern that implements a useful behavioral competency. Self-repair requires the use of timing information to specify the identity of appropriate functional connections. They derive a repair rule that has qualitative resemblance to experimentally described STDP rules.

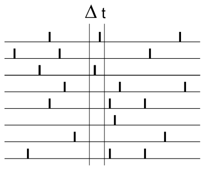
A promising approach consists in defining two concurrent learning mechanisms acting at different time scales. The idea is present in the Bienenstock-Cooper-Munro model, but not directly applicable to real-world problems. As alternative research track, combination of background running STDP and learning the network topology by evolutionary programming has been proved to be an interesting idea for learning in SNNs [43, 109].

4 Software and hardware implementation

Since computation in SNNs is based on spike timing and analog input values, it is no surprise that the area of SNN hardware implementation has known early and wide development, since the middle 90's. Indeed the historical models of spiking neurons (HH and LIF) had been designed as electrical circuits (cf. Section 1.3). Beside neural microcircuits, theoretical research, both in neuroscience and computer science, requires more and more powerful software for purpose of large scale simulation, since some macroscopic phenomena could emerge only from running very wide networks (several millions of neurons or billions of synapses) with fine tuning many microscopic model parameters. Next sections develop the specificities of simulating spiking neural network computation.

4.1 Event-driven simulation

The well known public simulators **GENESIS** [25] and **NEURON** [60] are convenient for programming precise biophysical models of neurons in order to forecast or analyse their precise behaviour, but they are not designed for fast simulation of very large scale SNNs. Actually such simulators are based on a **time-driven simulation**, which is proved to become quickly inefficient as the size of the network grows.



Only 2 spikes have been emitted in the time range Δt, among the potential 64 connections in a spiking neuron network of 8 neurons

$\Rightarrow$ Time-driven simulation yields 97 % useless computation !

Figure 19: A spike raster plot helps understanding why time-driven simulation is not convenient for SNNs.

In accordance with biological observations, the neurons of an SNN are sparsely and irregularly connected in space (network topology), and the variability of spike flows implies they communicate irregularly in time (network dynamics). Scrolling all the neurons and synapses of the network at each computation time step is useless time consuming since one of the major computing characteristics of SNNs is their low average activity (Figure 19). The sparse, but irregular, topology usually adopted for the underlying network architecture slightly compensates for this drawback. However the main interest of SNN simulation is to take into account the precise timing of spike emissions, hence the width of the time window used for discrete computation of the successive network states must remain narrow, so that only a few spike events occur at each time step, even in proportion to the effective connections.

Since the activity of an SNN can be fully described by emissions of dated spikes from pre-synaptic neurons towards post-synaptic neurons, an **Event-Driven Simulation** (EDS) is clearly suitable for sequential simulations of spiking neural networks [169, 106, 102, 135, 132]. More generally, event-driven approaches substantially reduce the computational charge of simulators that control exchanges of dated events between event-driven cells EC, without checking each cell at each time step.

In a typical spiking neuron network, at any time, each neuron can receive on its dendritic tree some signals emitted by other neurons. An incoming signal arrives with a delay d_{ij} and is weighted by a synaptic strength w_{ij} to be processed by the soma. The values of d_{ij} and w_{ij} are specific to a given connection, from a presynaptic neuron N_i to a postsynaptic neuron N_j. The membrane potential of a neuron varies in function of time and incoming signals. The neuron emits a spike, i.e. an outgoing signal on its axon, whenever its membrane potential overcomes a given threshold θ. In experimental setting, and thus for simulations, firing times are measured with some resolution Δt, yielding a discrete time representation. Hence each spike can be considered as an event, with a time stamp, and each neuron can be considered as an event-driven cell EC_j, able to forecast its next spike emission time, as result from the integration of incoming spikes.

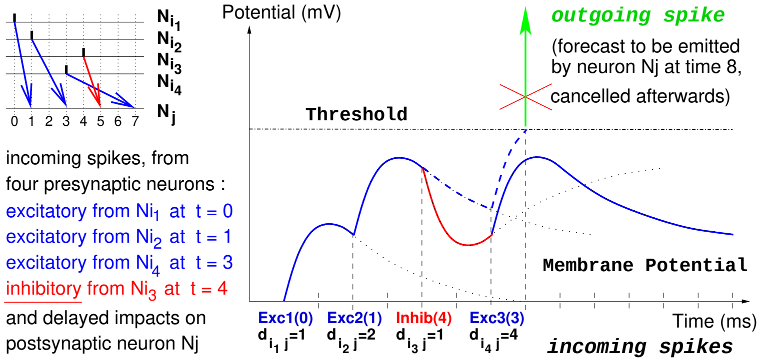


Figure 20: Variations of membrane potential for a postsynaptic neuron N_j. Three successive incoming excitatory spikes let forecast an outgoing spike that must be cancelled afterwards, due to a further incoming nhibitory spike (in red), with a smaller delay $d_{i_3 j}$.

However, the way to compute the future time of spike emission can be complex, depending on the model of neuron. For instance, if EC_j manages the incoming delays, a further incoming spike, with inhibitory synapse, can cancel the forecast of an outgoing spike before the stamp associated to this event (Figure 20). This issue is known under the name of **delayed firing problem** [55, 102, 106] and must be taken into account in simulations. Solutions can be found in the shape of the PSPs (if compatible with the neuron model), in the choice of the time stamp (e.g. the time when a spike impinges on the postsynaptic neuron), or by a posterior control with a sliding time window for further event validating.

4.2 Parallel computing

For developing very large scale SNNs, supporting a wide variety of spiking neuron models, a general purpose and fast running simulation framework is necessary. Since parallelism is an inherent feature of neural processing in brain, simulations of large scale neural networks could take advantage of parallel computing [23, 40]. It is known for long [54, 119] that a fine grain mapping of traditional neural networks (e.g. one neuron per processor, in rate coding) is dramatically inefficient, due to high communication overhead, the question is worth being revisited for spiking neuron networks.

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 t only a few neurons are active in an SNN, the classic bottleneck of message passing vanishes. Moreover, computing the updated state of membrane potential (e.g. SRM or LIF model neuron) is more complex than computing a weighted sum (e.g. threshold unit). Therefore communication time and computation cost are much more weell-balanced than in traditional NN simulation, and SNN simulation can highly benefit from parallel computing, as proved for instance by the parallel implementation of **SpikeNET** [39]. Jahnke et al. [78] propose a comparative study of SNN implementation on several parallel machines that were popular at end of the 90's. The neurocomputer CNAPS/256 (Adaptive Solutions), the systolic array processor SYNAPSE (Siemens AG), the parallel computer TMS320C80 (Texas Instruments), the 4xP90 and the SP2 that are MIMD parallel computer, and the SIMD computer CM-2 Connection Machine (Thinking Machines Corporation). Only the CM-2 exhibit enough performance for real-time simulation of large scale SNNs (up to 524 288 neurons), which confirms the intuitive advantage of fine-grain parallelism for neural computation.

When simulating SNNs on parallel systems, the new issue is to manage a correct matching between the running time on each processor (computer clocks) and the progress of the neural time in the network dynamics (simulated clocks), without irrecoverable divergence from a processor to another. As answer to the time control problem, several simulators [42, 129, 56] implement a unique controller or farmer processor for scheduling the network simulation. Other approaches, mainly developed by Grassmann et al. [55, 56], take advantage of the timing of spike events to direct the timing of execution, without an explicit synchronization barrier. A few studies coupled parallel computing and EDS, for general purpose systems [42], and for SNN simulation [55, 129, 56]. The more recent work in this direction is the **DAMNED simulator** [112] that combine event-driven simulation with two levels of parallelism. First, the SNN architecture is distributed onto P processors, according to a suitable mapping w.r.t. to the network topology. Second, on each processor, two concurrent processes run permanently for controlling either the message communication or the neural computation. Messages are packets of spike emission events, with time stamps, and local clock arrays. Furthermore, on each processor, all the neurons active at a local time t are computed by transient multithreaded concurrent processes. Although the computation is fully distributed, without central controller process, the scheduling of neural time simulation allows to solve the delayed firing problem (cf. Figure 20). Developed by Mouraud, Paugam-Moisy and Puzenat, DAMNED is written in C++, with the MPI[26] library to handle parallel computation, and can be implemented on parallel computers or workstation clusters (ongoing implementation).

4.3 Hardware circuits

Figure 21 illustrates a multi-chip implementation of SNNs [36] based on AER[27] systems. Among many works, a distinction can be done between systems based on FPGA gates (e.g. [160, 59, 79]) and systems based on VLSI circuits (e.g. [33, 118, 110]), or between systems that implement or not a synapse modification mechanism (learning rule, e.g. [59, 110]), or whether the system satisfies or not the real-time condition (e.g. [118]). Specific circuits are developed in purpose to real-time interface neuron models with biological cells (e.g [4]) for deeper understanding of spike-based dynamics and synaptic plasticity mechanisms such as STDP.

[26]MPI= Message Passing Interface, a system library for parallel programming (multiprocessor compilation, message-passing, etc.)
[27]AER = Address Event Representation

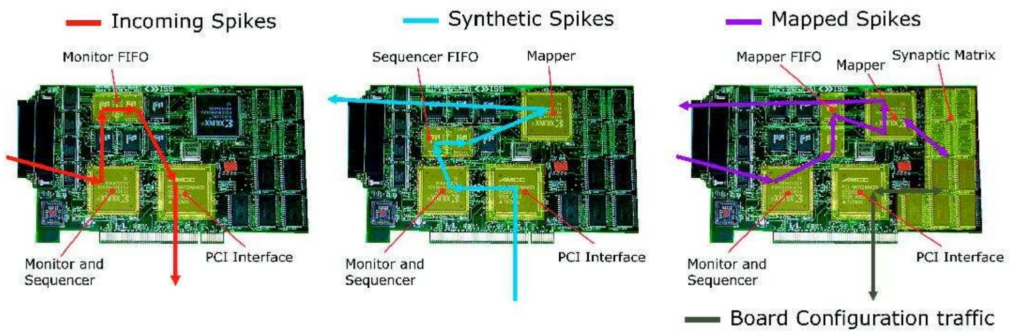


Figure 21: AER-based neuromorphic system. The three panels show, from left to right, information flow for the monitor, sequencer and mapper functionalities. [From Dante, Del Giudice and Whatley [36]]

The Maass and Bishop's book [96] contains several chapters about hardware implementation of SNNs, starting with a text by Murray as introduction [113]. An extensive study of Schäfer et al. [139] proposes a wide range of basic concepts for digital simulation of large SNNs. They present several hardware platforms for simulating complex networks and they investigate and compare the implementation concepts the different platforms are able to support: The SPIKE128k and the ParSPIKE-System, both developed at Paderborn; The NESPINN-System and the MASPINN-System, neurocomputers conceived in Berlin; A parallel PVM-Software-Simulator based on PVM[28] software development on a cluster of Sun workstations.

5 Application to temporal pattern recognition

SNNs appear to be able to give new computational solutions for **temporal pattern recognition**, i.e. for processing directly and efficiently the temporal aspects of data. But the way to do so is not yet straightforward. As quoted by Hopfield and Brody [65]:

> *How is information about spatiotemporal patterns integrated over time to produce responses selective to specific patterns and their natural variants?*

Temporal features are inherently present in almost all real-world data and human or animal processing:

- Visual perception of objects or structures vs motion …
- Auditory perception of phonemes, syllabes …
- Olfactory sensation of odors during a sniff
- Somatosensory feeling of textures

The major question is: How to capture online information contained in transient patterns? As stated in [91], STDP (see Section 3.2) enables spiking neurons to learn to predict even very complex temporal patterns of input currents that are provided to the neuron during training. However it is not usual to represent data by input currents. Even if patterns have been recorded with the help of captors (e.g. speech), usual recognition techniques start from applying standard preprocessing techniques to convert the data into real-valued vectors. Probably it is not the more efficient way to make use of SNNs and processing with spiking neurons should have to be rethought fundamentally.

The same idea has been developed by Hopfield and Brody who proposed an experimental spiking neuron network build for spoken digit recognition [64] and further analyzed the principles underlying the network processing [65]. The patterns are 10 spoken digits, taken from the TI46 database. The network has a rather

[28] PVM = Parallel Virtual Machine, a system library for parallel programming (similar to but slightly older than MPI)

complex architecture, biologically inspired, and the connections are designed (ad-hoc tuning) so that the output neurons are highly activated in response to the digit “one”.

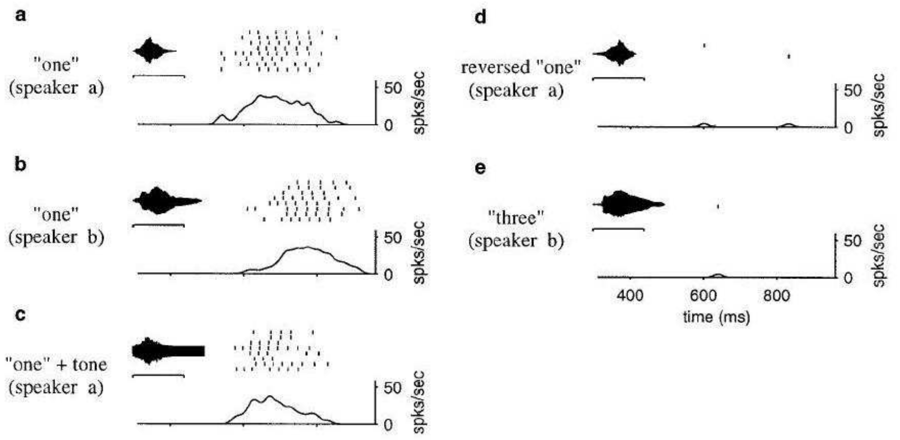


Figure 22: Responses (spike raster plots) of a network tuned on a single example, the “one” pronounced by speaker a. [From Hopfield and Brody [64]]

Experiments (Figure 22) show that the network is able to broad generalization, even from a single example, and is robust to noise. The network performs a temporal integration of transient synchrony. The underlying computational principle is that, based on the variety of decay rates in the network, the fundamental recognition event is the occurence of transient synchronization of neurons with convergent firing rates. Hence a subset of cells synchronize selectively for an input pattern. Detecting transient similar firing rates is an operation that is carried out very naturally and easily by networks of spiking neurons. The resulting collective synchronization event can be considered **a basic computational building block** for computing the operation: *“many variables are currently approximately equal”*. Advantages for temporal pattern recognition are valuable: The Hopfield-Brody network is a SNN that displays time-warp invariant spatiotemporal pattern recognition. The recognition process is also invariant with respect to pattern salience. Moreover experimental explorations with two enhancements (1. multiple events, 2. negative evidence) have suggested that interesting discrimination is achievable even with connected speech and speech-like noise in the background.

As concluded by Hopfield and Brody, the fuzzy “many are now equal” synchronization operation may be a fundamental and general computational building block, at the spiking neuron network level (as much important and new as the notion of firing threshold, at the neuron level), with no resemblance in traditional computing. Collective effects and decisions cannot be described in a similar fashion by the mathematical description of a single (even spiking) neuron.

Despite this prospective point of view, real-world problems are still hard to process that way. Nevertheless, several attempts to take advantage of multiplexed representation and fast processing of temporal patterns by SNNs have been developed successfully in different contexts of pattern recognition. Most of time, they combine traditional techniques with spike-based processing. Next two sections (5.1 and 5.2) present a sample of case studies as illustration of ongoing research.

5.1 Speech processing

Isolated word recognition using a Liquid State Machine, by Verstraeten, Schrauwen and Stroobandt [165]

- A “Lyon Passive Ear” is more suitable than a MFCC as front end for the LSM.
- Comparison of 3 ways to code stimuli into spike trains (the best is BSA).
- Conclude that the LSM is well suited for the task, with very good robustness for different types of noise.

Automatic speech recognition with neural spike trains, by Holmberg, Gelbart, Ramacher and Hemmert [62]

- Based on the difference between representation of sound signals in the human auditory system and in automatic speech recognition (ASR).
- Simulating spike trains of auditory nerve fibers → ASR results are demonstrated.
- Conclude to similar performance as MFCC and improvement in noise.
- Note that a large part of temporal information has been destroyed by use of HMM as ASR back end → could be improved, promising result.

A spiking neuron representation of auditory signals. by Wang and Pavel [168]

- Based on using the properties of the spiking neuron refractory period.
- Comparison between original input signal and reconstructed signal.
- Convert amplitude to temporal code while maintaining phase information of the carrier.
- Conclude that the narrow band envelope information could be encoded simply in the temporal inter-spike intervals.

Exploration of rank order coding with spiking neural networks for speech recognition, by Loiselle, Rouat, Pressnitzer and Thorpe [92]

- Based on rank order coding (times of first spikes - see Section 2.1), as for vision.
- Efficient on small training sets (fast response / adaptation).
- Complementary to usual methods (HMM, etc.), e.g. for a spoken "un".

Speech processing case studies presented above highlight that traditional preprocessing techniques are not optimal suited front end and back end for SNN processing. However interesting features have been pointed out, like robustness to noise, and combining SNN processing with other methods is proposed as a promising research area.

5.2 Computer vision

Rapid visual processing using spike asynchrony, by Thorpe and Gautrais [148]

- Based on the principle of rank order coding (times of first spikes) derived from the principle of natural vision processing [47].
- Powerful image compression and reconstruction techniques are derived (see also [163]).
- Importance of using rank order coding scheme in all the successive processing layers, not only at the input level.

This research area developed by Thorpe and colleagues has led them to start an engineering company, **SpikeNet Technology** [http://www.spikenet-technology.com] that develops and licences object recognition softwares for real-time vision, robust recognition and high speed tracking.

Sparse image coding using an asynchronous spiking neural network, by Perrinet and Samuelides [122]

- Based on rank order coding and propagation of spikes through a feed-forward SNN.
- Image reconstruction using a matching pursuit algorithm.
- Efficient compression and reconstruction technique, compared to image processing standards like JPEG.

A chaos synchronization-based dynamic vision model for image segmentation, by Azhar, Iftekharuddin and Kozma [6]

- Feature binding in spiking neurons using chaotic synchronization.
- Chaotic time series are generated from neurons coding the intensity (or color, texture) of image pixels.
- Segmented image generated from the largest cluster in the time series with similar chaotic synchronization parameter.
- Proof-of-concept results in segmenting medical images. Promising, but resource intensive technique.

Could early visual processes be sufficient to label motions?, by Kornprobst, Vieville and Dimov [87]

- Selection of relevant points for motion classification, from a video sequence as input.
- Method based on rank order coding, sequence description by spike responses.
- Information from early visual processes appears to be sufficient to classify biological motion.

Saliency extraction with a distributed spiking neural network, by Chevallier, Tarroux and Paugam-Moisy [32]

- Distributed SNN for extracting salient locations (e.g. based on intensity) in robot camera images.
- Spiking neurons efficiently detect conjunctions of high and low spatial frequency information (parallel processing of DOG and Gabor neural maps).
- Salient locations appear in a classified order, as result from rank order coding (order is hard to compute with traditional techniques).

5.3 Other application domains

SNN-based systems also develop increasingly in the area of robotics, where fast processing is a key issue [101, 153, 44, 43], from wheels to wings, or legged locomotion. The special abilities of SNNs for fast computing transient temporal patterns make them on first line for designing efficient systems in the area of autonomous robotics. This perspective is often cited but not yet fully developed.

Other research domains mention the use of SNNs, such as Echo State Netwroks for motor control (e.g. [137]), prediction in the context of wireless telecommunications (e.g. [77]) or neuromorphic approaches to rehabilitation, in medecine (e.g. [88]).

SNNs are able to draw collective decisions, based on transient synchronization, from local and transient properties of spatiotemporal patterns. Moreover, likewise the brain, they are able to process any kind of data by spike train encoding, whatever be the nature of input pattern, what opens up new horizons for computation. Since visual as well as auditory information, speech or other sensory data, could be preprocessed towards a uniform time-based representation, SNNs are ideal candidates for designing **multimodal interfaces**. For instance, an application of SNNs to audio-visual speech recognition has been proposed by Séguier and Mercier in [142]. Paugam-Moisy and her research group[29] has developed, for several years, a modular connectionist model of multimodal associative memory [34, 133, 120]. Recent work has focused on temporal aspects of visual and auditory data processing, in a virtual robotic prey-predator environment [134]: Distributed and concurrent processing [24] and spiking neuron networks as functional modules [108, 111], on purpose to develop real-time robotic multimodal interfaces [31, 32].

6 Conclusion

The present survey has focused on the computer science aspects of spiking neurons and networks, with special attention to machine learning and pattern recognition, the major research themes at IDIAP Research Institute. However it is proper to mention complementary active research areas, mainly developed by mathematicians or physicists. A special issue of the journal Natural Computation (2004) gathered extended versions papers [19, 37, 121, 166] from the "FUture of Neural Networks" workshop (FUNN). Some researchers analyse SNNs under the theory of dynamical systems, e.g. with mean field equations [38, 29, 7]. Other researchers take care of the oscillations that underlie input spike trains in olfactory system [105, 68] or auditory system[30]. For instance, the influence of the propagation delays on the weights modification, in phase with a background oscillatory signal, has been well described in the barn owl auditory system [49].

[29] Équipe "Connexionnisme et Modélisation Cognitive", institut des Sciences Cognitives, UMR CNRS 5015, Lyon (France)

[30] This research area will be highly represented at the 1st francophone conference in "Neurosciences Computationelles", in October 2006 [http://neurocomp.loria.fr/]

The numerous potential advantages of Spiking Neuron Networks have been widely discussed all along the report, in terms of computational complexity, learnability and temporal data representation and processing. In contrast, the novelty of the concept has for consequence that the computational power of SNNs is still hard to control, so that efficient ad-hoc techniques and algorithms remain to be discovered.

In conclusion:

Provided efficient information coding / decoding techniques, matching up with SNN computation specificities, and convenient machine learning algorithms, as easy to use as traditional NN learning rules, SNNs are the **tomorrow** computational device and a highway for **today**'s research !

References

[1] L. Abbott and W. Gerstner. Homeostasis and learning through spike-timing dependent plasticity. In C. Hansel, D. Chow, B. Gutkin, and C. Meunier, editors, *Methods and models in Neurophysics (Les Houches (F) summer school)*, 2004.

[2] L.F. Abbott and S.B. Nelson. Synaptic plasticity: taming the beast. *Nature Neuroscience*, 3:1178–1183, 2000.

[3] M. Abeles. *Corticonics: Neural Circuits of the Cerebral Cortex*. Cambridge Univ. Press, 1991.

[4] L. Alvado, J. Tomas, S. Saïghi, S. Renaud, T. Bal, A. Destexhe, and G. Le Masson. Hardware computation of conductance-based neuron models. *Neurocomputing*, 58–60:109–115, 2004.

[5] P. Auer, H. Burgsteiner, and W. Maass. A learning rule for very simple universal approximators consisting of a single layer of perceptrons. web available at http://www.igi.tugraz.at/maass/psfiles/126.pdf, submitted for publication (since 2001).

[6] H. Azhar, K. Iftekharuddin, and R. Kozma. A chaos synchronization-based dynamic vision model for image segmentation. In *IJCNN'2005, Int. Joint Conf. on Neural Networks*, pages 3075–3080. IEEE–INNS, 2005.

[7] M. Badoual, Q. Zou, A.P. Davison, M. Rudolph, T. Bal, Frgnac Y., and A. Destexhe. Biophysical and phenomenological models of multiple spike interactions in spike-timing dependent plasticity. *International Journal of Neural Systems*, 2006. (in press).

[8] J.L. Balcázar, J. Díaz, and J. Gabarró. *Structural complexity I (2nd edition)*. Springer-Verlag, 1995.

[9] D. Barber. Dynamic Bayesian networks withdeterministic latent tables. In S. Becker, S. Thrun, and K. Obermayer, editors, *NIPS*2002, Advances in Neural Information Processing Systems*, volume 15, pages 165–172. MIT Press, 2003.

[10] D. Barber. Learning in spiking neural assemblies. In S. Becker, S. Thrun, and K. Obermayer, editors, *NIPS*2002, Advances in Neural Information Processing Systems*, volume 15, pages 165–172. MIT Press, 2003.

[11] A. Bell and L. Parra. Maximising information yields spike timing dependent plasticity. In L.K. Saul, Y. Weiss, and L. Bottou, editors, *NIPS*2004, Advances in Neural Information Processing Systems*, volume 17, pages 121–128. MIT Press, 2005.

[12] G.-q. Bi. Spatiotemporal specificity of synaptic plasticity: cellular rules and mechanisms. *Biological Cybernetics*, 87:319–332, 2002.

[13] G.-q. Bi and M.-m. Poo. Synaptic modification in cultured hippocampal neurons: Dependence on spike timing, synaptic strength, and polysynaptic cell type. *J. of Neuroscience*, 18(24):10464–10472, 1998.

[14] G.-q. Bi and M.-m. Poo. Synaptic modification of correlated activity: Hebb's postulate revisited. *Annual Review of Neuroscience*, 24:139–166, 2001.

[15] W. Bialek, F. Rieke, R. de Ruyter, R.R. van Steveninck, and D. Warland. Reading a neural code. *Science*, 252:1854–1857, 1991.

[16] E.L. Bienenstock, L.N. Cooper, and P.W. Munro. Theory for the development of neuron selectivity: Orientation specificity and binocular interaction in visual cortex. *J. of Neuroscience*, 2(1):32–48, 1982.

[17] A. Blum and R. Rivest. Training a 3-node neural net is NP-complete. In *Proc. of NIPS*1988, Advances in Neural Information Processing Systems*, pages 494–501, 1989.

[18] A. Blumer, A. Ehrenfeucht, D. Haussler, and M.K. Warmuth. Learnability and the vapnik-chervonenkis dimension. *Journal of the ACM*, 36(4):929–965, 1989.

[19] S.M. Bohte. The evidence for neural information processing with precise spike-times: A survey. *Natural Computing*, 3(2):195–206, 2004.

[20] S.M. Bohte, Kok J.N., and H. La Poutré. Spikeprop: Error-backpropagation in temporally encoded networks of spiking neurons. *Neurocomputing*, 48:17–37, 2002.

[21] S.M. Bohte, H. La Poutré, and J.N. Kok. Unsupervised clustering with spiking neurons by sparse temporal coding and multilayer RBF networks. *IEEE Trans. on Neural Networks*, 13(2):426–435, 2002.

[22] S.M. Bohte and M.C. Mozer. A computational theory of Spike-Timing Dependent Plasticity: Achieving robust neural responses via conditional entropy minimization. Technical Report CWI TR-SEN E0505, The Netherlands Centre for Mathematics and Computer Science (NL), 2005. (submitted for publication).

[23] Y. Boniface, F. Alexandre, and S. Vialle. A library to implement neural networks on MIMD machines. In *Proc. of Euro-Par*, pages 935–938, 1999.

[24] Y. Bouchut, H. Paugam-Moisy, and D. Puzenat. Asynchrony in a distributed modular neural network for multimodal integration. In *PDCS'2003, Int. Conf. on Parallel and Distributed Computing and Systems*, pages 588–593. ACTA Press, 2003.

[25] J.M. Bower and D. Beeman. *The Book of GENESIS: Exploring Realistic Neural Models with the GEneral SImulation System*. Springer, 1998. 2nd edition.

[26] N.J. Buchs and W. Senn. Spike-based synaptic plasticity and the emergence of direction selective simple cells: Simulation results. *J. of Computational Neuroscience*, 13:167–186, 2002.

[27] G.A. Carpenter and S. Grossberg. *The Handbook of Brain Theory and Neural Networks (2nd edition) [M.A. Arbib, Ed.]*, chapter Adaptive Resonance Theory. MIT Press, 1998.

[28] G.A. Carpenter and B.L. Milenova. Redistribution of synaptic efficacy supports stable pattern learning in neural networks. *Neural Computation*, 14(4):873–888, 2002.

[29] B. Cessac and M. Samuelides. From neuron to neural network dynamics. (preliminary version of a review paper, available on http://www.inln.cnrs.fr/).

[30] G. Chechik. Spike-timing dependent plasticity and relevant mutual information maximization. *Neural Computation*, 15(7):1481–1510, 2003.

[31] S. Chevallier, H. Paugam-Moisy, and F. Lemaître. Distributed processing for modelling real-time multimodal perception in a virtual robot. In *PDCN'2005, Int. Conf. on Parallel and Distributed Computing and Networks*, pages 393–398. ACTA Press, 2005.

[32] S. Chevallier, P. Tarroux, and H. Paugam-Moisy. Saliency extraction with a distributed spiking neuron network. In *ESANN'06, Europ. Symp. on Artificial Neural Networks*, 2006. (to appear).

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

[34] A. Crpet, H. Paugam-Moisy, E. Reynaud, and D. Puzenat. A modular neural model for binding several modalities. In H. R. Arabnia, editor, *IC-AI'2000, Int. Conf. on Artificial Intelligence*, pages 921–928, 2000.

[35] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Math. Control, Signal Systems*, 2:303–314, 1988.

[36] V. Dante, P. Del Giudice, and A.M. Whatley. Hardware and software for interfacing to address-event based neuromorphic systems. *The Neuromorphic Engineer*, 2(1):5–6, 2005.

[37] E. Daucé. Short term memory in recurrent networks of spiking neurons. *Natural Computing*, 3(2):135–157, 2004.

[38] E. Daucé, M. Quoy, and B. Doyon. Resonant spatio-temporal learning in large random recurrent networks. *Biological Cybernetics*, 87:185–198, 2002.

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

[40] P.A. Estévez, H. Paugam-Moisy, D. Puzenat, and M. Ugarte. A scalable parallel algorithm for training a hierarchical mixture of neural networks. *Parallel Computing*, 28:861–891, 2002.

[41] C. Fernando and S. Sojakka. Pattern recognition in a bucket: a real liquid brain. In W. Banzhaf, T. Christaller, P. Dittrich, J.T. Kim, and J. Ziegler, editors, *ECAL'2003, Europ. Conf. on Artificial Life*, volume 2801 of *Lecture Notes in Computer Science*, pages 588–597. Springer, 2003.

[42] A. Fersha. Parallel and distributed simultation of discret event systems. In A. Y. Zomaya, editor, *Parallel and Distributed Computing Handbook*. McGraw-Hill, 1995.

[43] D. Floreano, Y. Epars, J.-C. Zufferey, and C. Mattiussi. Evolution of spiking neural circuits in autonomous mobile robots. *Int. J. of Intelligent Systems*, 2005. (in press).

[44] D. Floreano, J.C. Zufferey, and J.D. Nicoud. From wheels to wings with evolutionary spiking neurons. *Artificial Life*, 11(1-2):121–138, 2005.

[45] K. Funahashi. On the approximate realization of continuous mapings by neural networks. *Neural Networks*, 2(3):183–192, 1989.

[46] S. Fusi, P.J. Drew, and L.F. Abbott. Cascade models of synaptically stored memories. *Neuron*, 45:599–611, 2005.

[47] J. Gautrais and S.J. Thorpe. Rate coding vs temporal order coding: A theoretical approach. *Biosystems*, 48:57–65, 1998.

[48] W.. Gerstner. Time structure of the activity in neural network models. *Physical Review E*, 51:738–758, 1995.

[49] W. Gerstner, R. Kempter, and W. van Hemmen. *Hebbian learning of pulse timing in the barn owl auditory system*, chapter 14 in "Pulsed Neural Networks" (Maass & Bishop, Eds). MIT Press, 1999.

[50] W. Gerstner and W. Kistler. *Spiking Neuron Models: Single Neurons, Populations, Plasticity*. Cambridge University Press, 2002.

[51] W. Gerstner, W. Kistler, J.L. van Hemmen, and H. Wagner. A neural learning rule for sub-millisecond temporal coding. *Nature*, 383:76–78, 1996.

[52] W. Gerstner and W.M. Kistler. Mathematical formulations of hebbian learning. *Biological Cybernetics*, 87(5-6):404–415, 2002.

[53] W. Gerstner and J.L. van Hemmen. How to describe neuronal activity: Spikes, rates or assemblies? In J. D. Cowan, G. Tesauro, and J. Alspector, editors, *NIPS*1993, Advances in Neural Information Processing System*, volume 6, pages 463–470. MIT Press, 1994.

[54] J. Ghosh and K. Hwang. Mapping neural networks onto message-passing multicomputers. *Journal of Parallel Distributed Computing*, 6(2):291–330, 1989.

[55] C. Grassmann and J. K. Anlauf. Distributed, event-driven simulation of spiking neural networks. In *NC'98, International ICSC/IFAC Symposium on Neural Computation*, pages 100–105. ICSC Academic Press, 1998.

[56] C. Grassmann, T. Schönauer, and C. Wolff. Pcnn neurocomputeurs - event driven and parallel architectures. In *ESANN'02, European Symposium on Artificial Neural Network*, pages 331–336, 2002.

[57] D.O. Hebb. *The Organization of Behaviour*. Wiley, New York, 1949.

[58] W Heiligenberg. *Neural Nets in Electric Fish*. MIT Press, 1991.

[59] H. H. Hellmich, M. Geike, P. Griep, M. Rafanelli, and H. Klar. Emulation engine for spiking neurons and adaptive synaptic weights. In *IJCNN'2005, Int. Joint Conf. on Neural Networks*, pages 3261–3266. IEEE-INNS, 2005.

[60] M.L. Hines and N.T. Carnevale. The NEURON simulation environment. *Neural Computation*, 9:1179–1209, 1997.

[61] A.L. Hodgkin and A.F. Huxley. A quantitative description of ion currents and its applications to conduction and excitation in nerve membranes. *J. of Physiology*, 117:500–544, 1952.

[62] M. Holmberg, D. Gelbart, U. Ramacher, and W. Hemmert. Isolated word recognition using a liquid state machine. In *EuroSpeech'2005, European Conference on Speech Communication*, 2005.

[63] J.J. Hopfield. Pattern recognition computation using action potential timing for stimulus representation. *Nature*, 376(6535):33–36, 1995.

[64] J.J. Hopfield and C.D. Brody. What is a moment ? "Cortical" sensory integration over a brief interval. *Proc. Natl. Acad. Sci.*, 97(25):13919–13924, 2000.

[65] J.J. Hopfield and C.D. Brody. What is a moment ? Transient synchrony as a collective mechanism for spatiotemporal integration. *Proc. Natl. Acad. Sci.*, 98(3):1282–1287, 2001.

[66] J.J. Hopfield and C.D. Brody. Learning rules and network repair in spike-timing-based computation networks. *Proc. Natl. Acad. Sci.*, 101(1):337–342, 2004.

[67] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

[68] E. Hugues and D. Martinez. Encoding in a network of sparsely connected spiking neurons : application to locust olfaction neurocomputing. *Neurocomputing*, 65–66:537–542, 2005.

[69] E.M. Izhikevich. Simple model of spiking neurons. *IEEE Trans. in Neural Networks*, 14(6):1569–1572, 2003.

[70] E.M. Izhikevich. Which model to use for cortical spiking neurons? *IEEE Trans. in Neural Networks*, 15(5):1063–1070, 2004.

[71] E.M. Izhikevich. Polychronization: Computation with spikes. *Neural Computation*, 18(1):245–282, 2006.

[72] E.M. Izhikevich and N.S. Desai. Relating STDP and BCM. *Neural Computation*, 15(7):1511–1523, 2003.

[73] E.M. Izhikevich, J.A. Gally, and G.M. Edelman. Spike-timing dynamics of neuronal groups. *Cerebral Cortex*, 14:933–944, 2004.

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

[75] H. Jaeger. Tutorial on training recurrent neural networks, covering BPTT, RTRL, EKF and the "echo state network" approach. Technical Report TR-GMD-159, German National Research Center for Information Technology, 2002.

[76] H. Jaeger. Reservoir riddles: suggestions for Echo State Network research (extended abstract). In *IJCNN'2005, Int. Joint Conf. on Neural Networks*, pages 1460–1462. IEEE–INNS, 2005.

[77] H. Jaeger and H. Haas. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless telecommunication. *Science*, pages 78–80, 2004.

[78] A. Jahnke, T. Schoneauer, U. Roth, K. Mohraz, and H. Klar. Simulation of spiking neural networks on different hardware platforms. In *ICANN'1997, Int. Conf. on Artificial Neural Networks*, pages 1187–1192, 1997.

[79] S. Johnston, G. Prasad, L. Maguire, and McGinnity. Comparative investigation into classical and spiking neuron implementations on FPGAs. In *ICANN'2005, Int. Conf. on Artificial Neural Networks*, volume 3696 of *LNCS*, pages 269–274. Springer-Verlag, 2005.

[80] J.S. Judd. *Neural network design and the complexity of learning*. MIT Press, 1990.

[81] U. Karmarkar, M. Najarian, and D. Buonomano. Mechanisms and significance of spike-timing dependent plasticity. *Biological Cybernetics*, 87:373–382, 2002.

[82] R. Kempter, W. Gerstner, and J. L. van Hemmen. Hebbian learning and spiking neurons. *Physical Review E*, 59(4):4498–4514, 1999.

[83] R. Kempter, W. Gerstner, and J. L. van Hemmen. Intrinsic stabilization of output rates by spike-based hebbian learning. *Neural Computation*, 13:2709–2742, 2001.

[84] W.M. Kistler. Spike-timing dependent synaptic plasticity: a phenomenological framework. *Biological Cybernetics*, 87(5-6):416–427, 2002.

[85] W.M. Kistler, W. Gerstner, and J.L. van Hemmen. Reduction of hodgkin-huxley equations to a single-variable threshold model. *Neural Computation*, 9:1015–1045, 1997.

[86] P. König, A.K. Engel, and W. Singer. Integrator or coincidence detector? the role of the cortical neuron revisited. *Trends in Neuroscience*, 19(4):130–137, 1996.

[87] P. Kornprobst, T. Viéville, and I.K. Dimov. Could early visual processes be sufficient to label motions? In *IJCNN'2005, Int. Joint Conf. on Neural Networks*, pages 1687–1692. IEEE–INNS, 2005.

[88] J.J. Kutch. Neuromorphic approaches to rehabilitation. *The Neuromorphic Engineer*, 1(2):1–2, 2004.

[89] N. Kuwabara and N. Suga. Delay lines and amplitude selectivity are created in subthalamic auditory nuclei: the brachium of the inferior colliculus of the mustached bat. *J. of Neurophysiology*, 69:1713–1724, 1993.

[90] L. Lapicque. Recherches quantitatives sur l'excitation électrique des nerfs traité comme une polarization. *J. Physiol. Pathol. Gen.*, 9:620–635, 1907. cited by Abbott, L.F., in Brain Res. Bull. 50(5/6):303–304.

[91] R. Legenstein, C. Näger, and W. Maass. What can a neuron learn with Spike-Time-Dependent Plasticity? *Neural Computation*, 17(11):2337–2382, 2005.

[92] S. Loiselle, J. Rouat, D. Pressnitzer, and S. Thorpe. Exploration of rank order coding with spiking neural networks for speech recognition. In *IJCNN'2005, Int. Joint Conf. on Neural Networks*, pages 2076–2080. IEEE–INNS, 2005.

[93] W. Maass. Fast sigmoidal networks via spiking neurons. *Neural Computation*, 10:1659–1671, 1997.

[94] W. Maass. Networks of spiking neurons: The third generation of neural network models. *Neural Networks*, 10:1659–1671, 1997.

[95] W. Maass. On the relevance of time in neural computation and learning. *Theoretical Computer Science*, 261:157–178, 2001. (extended version of ALT'97, in LNAI 1316:364-384).

[96] W. Maass and C.M. Bishop, editors. *Pulsed Neural Networks*. MIT Press, 1999.

[97] W. Maass and T. Natschläger. Networks of spiking neurons can emulate arbitrary Hopfield nets in temporal coding. *Network: Computation in Neural Systems*, 8(4):355–372, 1997.

[98] W. Maass, T. Natschläger, and H. Markram. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Computation*, 14(11):2531–2560, 2002.

[99] W. Maass and M. Schmitt. On the complexity of learning for a spiking neuron. In *COLT'97, Conf. on Computational Learning Theory*, pages 54–61. ACM Press, 1997.

[100] W. Maass and M. Schmitt. On the complexity of learning for spiking neurons with temporal coding. *Information and Computation*, 153:26–46, 1999.

[101] W. Maass, G. Steinbauer, and R. Koholka. Autonomous fast learning in a mobile robot. In *Sensor Based Intelligent Robots*, pages 345–356. Springer, 2000.

[102] T. Makino. A discrete event neural network simulator for general neuron model. *Neural Computation and Applic.*, 11(2):210–223, 2003.

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

[104] H. Markram and M.V. Tsodyks. Redistribution of synaptic efficacy between neocortical pyramidal neurones. *Nature*, 382:807–809, 1996.

[105] D. Martinez. Oscillatory synchronization requires precise and balanced feedback inhibition in a model of the insect antennal lobe. *Neural Computation*, 17:2548–2570, 2005.

[106] M. Mattia and P. Del Giudice. Efficient event-driven simulation of large networks of spiking neurons and dynamical synapses. *Neural Computation*, 12:2305–2329, 2000.

[107] W.S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943.

[108] D. Meunier and H. Paugam-Moisy. A "spiking" Bidirectional Associative Memory for modeling intermodal priming. In *NCI'2004, Int. Conf. on Neural Networks and Computational Intelligence*, pages 25–30. ACTA Press, 2004.

[109] D. Meunier and H. Paugam-Moisy. Evolutionary supervision of a dynamical neural network allows learning with on-going weights. In *IJCNN'2005, Int. Joint Conf. on Neural Networks*, pages 1493–1498. IEEE–INNS, 2005.

[110] S. Mitra, S. Fusi, and G. Indiverti. A VLSI spike-driven dynamic synapse which learns only when necessary. In *ISCAS'2006, IEEE Int. Symp. on Circuits and Systems*, 2006. (to appear).

[111] A. Mouraud and H. Paugam-Moisy. Learning and discrimination through STDP in a top-down modulated associative memory. In *ESANN'06, Europ. Symp. on Artificial Neural Networks*, 2006. (to appear).

[112] A. Mouraud, H. Paugam-Moisy, and D. Puzenat. A Distributed And Multithreaded Neural Event Driven simulation framework. In *PDCN'2006, Int. Conf. on Parallel and Distributed Computing and Networks*, page (to appear), Innsbruck, AUSTRIA, February 2005. ACTA Press.

[113] A.F. Murray. *Pulse-based computation in VLSI neural networks*, chapter 3 in "Pulsed Neural Networks" (Maass & Bishop, Eds). MIT Press, 1999.

[114] T. Natschläger and B. Ruf. *Online clustering with spiking neurons using Radial Basis Functions*, chapter 4 in "Neuromorphic Systems: Engineering Silicon from Neurobiology" (Hamilton & Smith, Eds). World Scientific, 1998.

[115] T. Natschläger and B. Ruf. Spatial and temporal pattern analysis via spiking neurons. *Network: Comp. Neural Systems*, 9(3):319–332, 1998.

[116] T. Nowotny, V.P. Zhigulin, A.I. Selverston, H.D.I. Abardanel, and M.I. Rabinovich. Enhancement of synchronization in a hibrid neural circuit by Spike-Time-Dependent Plasticity. *The Journal of Neuroscience*, 23(30):9776–9785, 2003.

[117] E. Oja. A simplified neuron model as a principal component analyzer. *J. of Mathematical Biology*, 15:267–273, 1982.

[118] M. Oster, A.M. Whatley, S.-C. Liu, and R.J. Douglas. A hardware/software framework for real-time spiking systems. In *ICANN'2005, Int. Conf. on Artificial Neural Networks*, volume 3696 of *LNCS*, pages 161–166. Springer-Verlag, 2005.

[119] H. Paugam-Moisy. Multiprocessor simulation of neural networks. In M. Arbib, editor, *The Handbook of Brain Theory and Neural Networks*, pages 605–608. MIT Press, 1995.

[120] H. Paugam-Moisy and E. Reynaud. Multi-network system for sensory integration. In *IJCNN'2001, Int. Joint Conf. on Neural Networks*, pages 2343–2348, 2001.

[121] L. Perrinet. Finding independent components using spikes: A natural result of hebbian learning in a sparse spike coding scheme. *Natural Computing*, 3(2):159–175, 2004.

[122] L. Perrinet and M. Samuelides. Sparse image coding using an asynchronous spiking neural network. In *ESANN'2002, Europ. Symp. on Artificial Neural Networks*, pages 313–318, 2002.

[123] J.-P. Pfister, D. Barber, and W. Gerstner. Optimal hebbian learning: A probabilistic point of view. In O. Kaynak, E. Alpaydin, E. Oja, and L. Xu, editors, *ICANN/ICONIP 2003, Int. Conf. on Artificial Neural Networks*, volume 2714 of *Lecture Notes in Computer Science*, pages 92–98. Springer, 2003.

[124] J.-P. Pfister and W. Gerstner. Beyond pair-based STDP: a phenomenological rule for spike triplet and frequency effects. In *NIPS*2005, Advances in Neural Information Processing Systems*, volume 18. MIT Press, 2006. (to appear).

[125] J.-P. Pfister, T. Toyoizumi, D. Barber, and W. Gerstner. Optimal spike-timing dependent plasticity for precise action potential firing in supervised learning. Technical Report IDIAP-RR 05-88, IDIAP Research Institute, Martigny (CH), 2005. (to appear in Neural Computation).

[126] T. Poggio and F. Girosi. Networks for approximation and learning. *Proceedings of the IEEE*, 78(9):1481–1497, 1989.

[127] B. Porr and F. Wörgötter. Isotropic sequence order learning. *Neural Computation*, 15(4):831–864, 2003.

[128] B. Porr and F. Wörgötter. Temporal sequence learning, prediction, and control: A review of different models and their relation to biological mechanisms. *Neural Computation*, 17(2):245–319, 2005.

[129] R. Preis, K. Salzwedel, C. Wolff, and G. Hartmann. Efficient parallel simulation of pulse-coded neural networks (pcnn). In *PDPTA'2001, International Conference on Parallel and Distributed Processing Techniques and Applications*, 2001.

[130] R. Rao and T. Sejnowski. Spike-timing-dependent plasticity as temporal difference learning. *Neural Computation*, 13:2221–2237, 2001.

[131] M. Recce. *Encoding information in neuronal activity*, chapter 4 in “Pulsed Neural Networks” (Maass & Bishop, Eds). MIT Press, 1999.

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

[133] E. Reynaud, A. Crépet, H. Paugam-Moisy, and D. Puzenat. A modular neural network model for a multimodal associative memory. In *Proc. of 4th International Conference on Cognitive and Neural systems, ICCNS'2000*, page 17. Boston University, US, 2000.

[134] E. Reynaud and D. Puzenat. A multisensory identification system for robotics. In *IJCNN 2001, Int. Joint Conf. on Neural Networks*, pages 2924–2929, 2001.

[135] O. Rochel and D. Martinez. An event-driven framework for the simulation of networks of spiking neurons. In *ESANN'03, European Symposium on Artificial Neural Network*, pages 295–300, 2003.

[136] B. Ruf and M. Schmitt. Self-organization of spiking neurons using action potential timing. *IEEE Trans. on Neural Networks*, 9(3):575–578, 1998.

[137] M. Salmen and P.G. Plöger. Echo State Networks used for motor control. In *ICRA'2005, Int. Joint on Robotics and Automation*. IEEE, 2005.

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

[139] M. Schäfer, T. Schönauer, C. Wolff, G. Hartmann, H. Klar, and U. Rückert. Simulation of spiking neural networks - architectures and implementations. *Neurocomputing*, 48:647–679, 2002.

[140] M. Schmitt. On computing boolean functions by a spiking neuron. *Annals of Mathematics and Artificial Intelligence*, 24:181–191, 1998.

[141] M. Schmitt. On the sample complexity of learning for networks of spiking neurons with nonlinear synaptic interactions. *IEEE Trans. on Neural Networks*, 15(5):995–1001, 2004.

[142] R. Séguie and D. Mercier. Audio-visual speech recognition one pass learning with spiking neurons. In *ICANN '02, Int. Conf. on Artificial Neural Networks*, pages 1207–1212. Springer-Verlag, 2002.

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

[144] A. Shon, R. Rao, and T. Sejnowski. Motion detection and prediction through spike-timing dependent plasticity. *Network: Computation in Neural Systems*, 15:179–198, 2004.

[145] H.T. Siegelmann. *Neural networks and analog computation, beyond the Turing limit*. Birkhauser, 1999.

[146] H.T. Siegelmann and E.D. Sontag. On computational power of neural networks. *J. of Computer and Systems Sciences*, 50(1):132–150, 1995.

[147] J. Sima and J. Sgall. On the nonlearnability of a single spiking neuron. *Neural Computation*, 17(12):2635–2647, 2005.

[148] Thorpe S.J. and J. Gautrais. Rapid visual processing using spike asynchrony. In M. Mozer, Jordan M.I., and T. Petsche, editors, *NIPS*1996, Advances in Neural Information Processing Systems*, volume 9, pages 901–907. MIT Press, 1997.

[149] S. Song, K.D. Miller, and L.F. Abbott. Competitive hebbian learning through spike-time dependent synaptic plasticity. *Nature Neuroscience*, 3(9):919–926, 2000.

[150] D.I. Standage and T.P. Trappenberg. Differences in the subthreshold dynamics of leaky integrate-and-fire ans Hodgkin-Huxley neuron models. In *IJCNN'2005, Int. Joint Conf. on Neural Networks*, pages 396–399. IEEE–INNS, 2005.

[151] R.B. Stein. A theoretical analysis of neuronal variability. *Biophys. J.*, 5:173–194, 1965.

[152] R.S. Sutton. Learning to predict by the method of temporal difference. *Machine Learning*, 3(1):9–44, 1988.

[153] F. Tenore. Prototyping neural networks for legged locomotion using custom aVLSI chips. *The Neuromorphic Engineer*, 1(2):4, 8, 2004.

[154] S. Thorpe, A. Delorme, and R. Van Rullen. Spike-based strategies for rapid processing. *Neural Networks*, 14:715–725, 2001.

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

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

[157] T. Toyoizumi, J.-P. Pfister, K. Aihara, and W. Gerstner. Spike-timing dependent plasticity and mutual information maximization for a spiking neuron model. In L.K. Saul, Y. Weiss, and L. Bottou, editors, *NIPS*2004, Advances in Neural Information Processing Systems*, volume 17, pages 1409–1416. MIT Press, 2005.

[158] A.M. Turing. Systems of logic based on ordinals. *Proceedings of the London Mathematical Society*, 45(2):161–228, 1939.

[159] A.M. Turing. Computing machinery and intelligence. *Mind*, 59:433–460, 1950.

[160] A. Upegui, C.A. Peña Reyes, and E. Sanchez. An FPGA platform for on-line topology exploration of spiking neural networks. *Microprocessors and Microsystems*, 29:211–223, 2004.

[161] L.G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

[162] R. van Rossum, G.-q. Bi, and G. Turrigiano. Stable hebbian learning from spike time dependent plasticity. *J. Neuroscience*, 20:8812–8821, 2000.

[163] R. Van Rullen and S. Thorpe. Rate coding versus temporal order coding: what the retinal ganglion cells tell the visual cortex. *Neural Computation*, 13:1255–1283, 2001.

[164] V.N. Vapnik. *Statistical learning theory*. Wiley, 1998.

[165] D. Verstraeten, B. Schrauwen, and D. Stroobandt. Isolated word recognition using a liquid state machine. In *ESANN'05, European Symposium on Artificial Neural Network*, pages 435–440, 2005.

[166] M. Volkmer. A pulsed neural network model of spectro-temporal receptive fields and population coding in auditory cortex. *Natural Computing*, 3(2):177–193, 2004.

[167] J. Vreeken. On real-world temporal pattern recognition using Liquid State Machines. Master's thesis, Utrecht University (NL), 2004.

[168] G. Wang and M. Pavel. A spiking neuron representation of auditory signals. In *IJCNN'2005, Int. Joint Conf. on Neural Networks*, pages 416–421. IEEE–INNS, 2005.

[169] L. Watts. Event-driven simulation of networks of spiking neurons. In J. D. Cowan, G. Tesauro, and J. Alspector, editors, *Advances in Neural Information Processing System*, volume 6, pages 927–934. MIT Press, 1994.

[170] T. Wennekers, F. Sommer, and A. Aertsen. Editorial: Cell assemblies. *Theory in Biosciences (special issue)*, 122:1–4, 2003.

[171] X. Xie and H.S. Seung. Learning in neural networks by reinforcement of irregular spiking. *Physical Review E*, 69(041909), 2004.

[172] A.M. Zador and B.A. Pearlmutter. VC dimension of an integrate-and-fire neuron model. *Neural Computation*, 8(3):611–624, 1996.