Spiking neural networks, an introduction

Jilles Vreeken

Adaptive Intelligence Laboratory, Intelligent Systems Group,
Institute for Information and Computing Sciences, Utrecht University
Correspondence e-mail address: jvreeken@cs.uu.nl

Biological neurons use short and sudden increases in voltage to send information. These signals are more commonly known as action potentials, spikes or pulses. Recent neurological research has shown that neurons encode information in the timing of single spikes, and not only just in their average firing frequency. This paper gives an introduction to spiking neural networks, some biological background, and will present two models of spiking neurons that employ pulse coding. Networks of spiking neurons are more powerful than their non-spiking predecessors as they can encode temporal information in their signals, but therefore do also need different and biologically more plausible rules for synaptic plasticity.

You constantly receive sensory input from your environment. You process this information, recognizing food or danger, and take appropriate actions. Not only you; anything that interacts with its environment needs to do so. Mimicking such a seemingly simple mechanism in a robot proofs to be insanely difficult. Nature must laugh at our feeble attempts; animals perform this behaviour with apparent ease.

The reason for this mind-boggling performance lies in their neural structure or 'brain'. Millions and millions of neurons are interconnected with each other and cooperate to efficiently process incoming signals and decide on actions. A typical neuron sends its signals out to over 10.000 other neurons, making it clear to even to inexpert reader that the signal flow is rather complicated. To put it mildly: we do not understand the brain that well yet. In fact, we do not even completely understand the functioning of a single neuron. The chemical activity of the synapse already proves to be infinitely more complex than firstly assumed.

However, the rough concept of how neurons work is understood: neurons send out short pulses of electrical energy as signals, if they have received enough of these themselves. This basically simple mechanism has been moulded into a mathematical model for computer use. Artificial as these computerised neurons are, we refer to them as networks of artificial neurons, or artificial neural networks. We will sketch a short history of these now; the biological background of the real neuron will be drawn in the next chapter.

Generations of artificial neurons

Artificial neural networks are already becoming a fairly old technique within computer science; the first ideas and models are over fifty years old. The first generation of artificial neural networks consisted of McCulloch-Pitts threshold neurons [15], a conceptually very simple model: a neuron sends a binary 'high' signal if the sum of its weighted incoming signals rises above a threshold value. Even though these neurons can only give digital output, they have been successfully applied in powerful artificial neural networks like multi-layer perceptrons and Hopfield nets. For example, any function with Boolean output can be computed by a multi-layer perceptron with a single hidden layer; these networks are called universal for digital computations.

Neurons of the second generation do not use a step- or threshold function to compute their output signals, but a continuous activation function, making them suitable for analog in- and output. Commonly used examples of activation functions are the sigmoid and hyperbolic tangent. Typical examples of neural networks consisting of neurons of these types are feed-forward and recurrent neural networks. These are more powerful than their first generation predecessors: when equipped with a threshold function at the output layer of the network they are universal for digital computations, and do so with fewer neurons than a network of the first generation [14]. In addition they can approximate any analog function arbitrarily well, making these networks universal for analog computations.

Neuron models of the first two generations do not employ individual pulses, but their output signals typically lie between 0 and 1. These signals can be seen as normalized firing rates (frequencies) of the neuron within a certain period of time. This is a so-called rate coding, where a higher rate of firing correlates with a higher output signal. Rate coding implies an averaging mechanism, as real spikes work binary: spike, or no spike, there is no intermediate. Due to such an averaging window mechanism the output value of a neuron can be calculated in iteration. After such a cycle for each neuron the 'answer' of the network to the input values is known. Real neurons have a base firing-rate (an intermediate frequency of pulsing) and continuous activation functions can model these intermediate output frequencies. Hence, neurons of the second generation are more biologically realistic and powerful than neurons of the first generation [3].

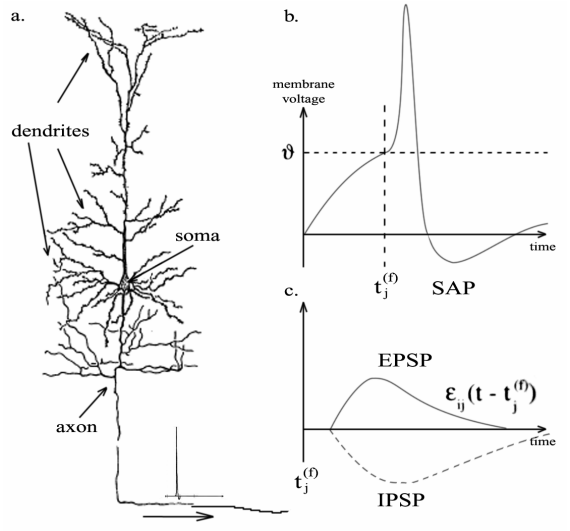

Figure 1. (a) Schematic drawing of a neuron. (b) Incoming postsynaptic potentials alter the membrane voltage so it crosses threshold value ϑ; the neuron spikes and goes into a refractory state. (c) Typical forms of excitatory and inhibitory postsynaptic potentials over time. [8]

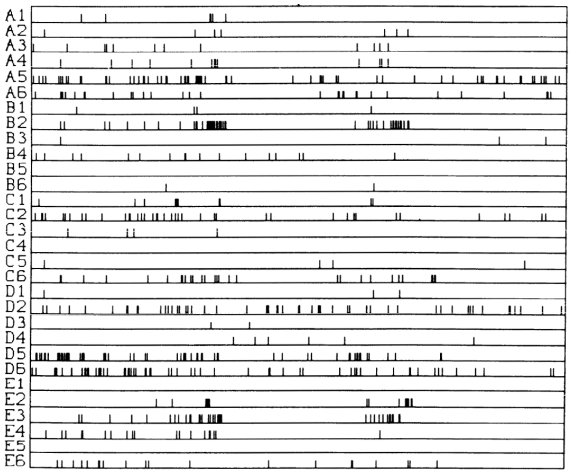

Figure 2. A 4 second recording of the neural activity recording from 30 neurons of the visual cortex of a monkey. Each vertical bar indicates a spike. The human brain can recognize a face within 150ms [24], which correlates to less than 3mm in this diagram; dramatic changes in firing frequency occur in this time span, neurons have to rely on information carried by solitary spikes. [13]

Adjusting the synaptic weights can alter the information flow through a neural network; the strengths of incoming signals of a neuron are altered, so most likely the output signal will also change in strength. This is a simplified basis for learning, also known as synaptic plasticity. Using a continuous activation function allows us to apply a gradient-descent learning algorithm like backward propagation [12]. This is a very commonly used and powerful supervised learning algorithm for training a network to give the desired output for a certain input vector.

The third generation of neural networks once again raises the level of biological realism by using individual spikes. This allows incorporating spatial-temporal information in communication and computation, like real neurons do [7]. So instead of using rate coding these neurons use pulse coding; mechanisms where neurons receive and do send out individual pulses, allowing multiplexing of information as frequency and amplitude of sound [9]. Recent discoveries in the field of neurology have shown that neurons in the cortex perform analog computations at incredible speed. Thorpe et al [24] demonstrated that humans analyse and classify visual input (i.e. facial recognition) in under 100ms. It takes at least 10 synaptic steps from the retina to the temporal lobe; this leaves about 10ms of processing-time per neuron. Such a time-window is much too little to allow an averaging mechanism like rate coding [9,24]. This does not mean that rate coding is not used, though when speed is an issue pulse coding schemes are favoured [24]. Before going into more detail about artificial spiking neurons, we will treat some more on the biological background of real neurons.

Biological background

Maass [19] correctly points out that it's a bad idea to pour water over your computer. It would most likely stop functioning, defining a sharp contrast with the absolute need for water all organisms in nature have. The neurons (see fig. 1a) in our brain find themselves surrounded by an artificial ocean of salty extra-cellular fluid. Our wetware, as we call our brain and parts of it, bears a lot of resemblance with the wetware of creatures that still live in the ocean. Squid have neurons up to 1.000 times larger than we do, making them much easier to examine. Despite the huge difference in size their functioning is alike; the equations Hodgkin and Huxley derived for squid neuron dynamics can also be used to describe the neurons in our brain. These similarities are extremely helpful in the research of how neurons, and the brain in general, function.

Computers communicate with bits; neurons use spikes. Incoming signals alter the voltage of the neuron and when this reaches above a threshold-value the neuron sends out an action potential itself. Such an action potential is a short (1ms) and sudden increase in voltage that is created in the cell body or soma. Due to their form and nature (see fig. 1b) we refer to them as spikes or pulses. The spike traverses down the axon of the neuron, axons being signal carriers that grow quite long before they start to branch: we have up to 4 kilometres of them in every cubic millimetre of our cortex. Bodies of Ranvier amplify the spike over the course of the axon, and at most branching points the spike is duplicated, so that minimal information loss occurs [4,10,15,19].

Spikes cannot just cross the gap between one neuron and the other. They have to be handled by the most complicated part of the neuron: the synapse [18,19], formed by the end of the axon, a synaptic gap and the first part of the dendrite. The synapse was first thought to only just transfer a signal from axon to the dendrite; it has proved to be a very complicated signal pre-processor and is crucial in learning and adaptation. When a spike arrives at the axonal (presynaptic) side of the synapse it is likely that some vesicles fuse with the cell membrane and release their neurotransmitter content into the extra-cellular fluid that fills the synaptic gap. Neurotransmitter molecules have to reach a matching receptor on the postsynaptic side of the gap to open an ion-gate on the neuron. Such a postsynaptic potential (see fig. 1c) can either be positive and called excitatory (EPSP) or negative and called inhibitory (IPSP). Once these charged particles enter the neuron they initiate a cascade that traverses the dendritic tree down to the trigger zone of the soma, altering the membrane potential. A single neuron receives potentials from roughly 10.000 synapses [19]. When the sum of these potentials reaches a threshold value the neuron sends out a spike down the axon. After which the neuron enters a short moment (10ms) of rest, the refractory period, in which it cannot send out a spike again.

Contrary to spikes, which are all very much alike, postsynaptic potentials differ in size. This is caused by the differ-

ences in amounts and types of neurotransmitters released and the resulting number of ion-channels activated, in short, the synaptic efficacy [8,10,19]. The long- and short-term history of the synapse and outside influences shape the role of a synapse as a pre-processor. The history of the synapse influences its characteristics and capabilities in the form of i.e. chance of vesicle deployment, regeneration and the amount of receptors. Neuro-hormones in the extra-cellular fluid can influence both the pre- and postsynaptic terminals temporarily by i.e. enhancing vesicle regeneration or blocking transmitters from activating ion-gate receptors. These are all examples of synaptic plasticity: influences on the effect of an incoming presynaptic spike on the postsynaptic neuron, which forms the basis of most models of learning and development of neural networks.

Now we know a bit more of what is going on 'under the hood' of a neuron, we can take another look at figure 2. We see that individual neurons send out erratic sequences of spikes, or spike-trains, which alter dramatically in frequency over a short period of time. Neurons have to use spatial and temporal information of incoming spike patterns to encode their message to other neurons [9,15,16]. Besides that figure 2 faintly resembles a musical score there are more parallels: in order to recognize a piece of music you have to hear more than just a single note, the melody and notes played by other musicians are even more important [19].

Spike coding

There are many different schemes for the use of spike timing information in neural computation. Because of the nature of this paper we'll only cover two models here: the spike response and the integrate-and-fire model. Both are instances of the general threshold-fire model. The integrate-and-fire model, which is very commonly used in networks of spiking neurons, will be covered after the conceptually more simple and general spike-response model. This model is simple to understand and implement. However, as it approximates the very detailed Hodgkin-Huxley model very well it captures generic properties of neural activity [8,9].

As we've seen in the previous chapter action potentials are all very similar. We can therefore forget about form and characterise these by their firing times $t_i^{(f)}$. The lower index i indicates the neuron, the upper index f the number of the spike. We can then describe the spike-train of a neuron as

$$F_i = \{t^{(1)},\ldots,t^{(n)}\} \tag{1.1}$$

The variable u_i is used to refer to the membrane potential, or internal state, of a neuron i. If a neuron's membrane potential crosses threshold value ϑ from below, it generates a spike. We add the time of this event to F_i, defining this set as

$$F_i = \{\, t \mid u_i(t) = \vartheta \wedge u_i'(t) > 0 \,\} \tag{1.2}$$

When a neuron generates an action potential, the membrane potential suddenly increases, followed by a long lasting negative after-potential (see fig. 1b). This sharp rise above the threshold value makes it is absolutely impossible for the neuron to generate another spike and is named absolute refractoriness. The period of relative refractoriness, which we call the negative spike after-potential (SAP), making it less likely that the neuron fires again. We can model this absolute and negative refractoriness with kernel η:

$$\eta(s) = -n_0 \exp\left(-\frac{s-\delta^{abs}}{t}\right)H(s-\delta^{abs}) - KH(s)H(\delta^{abs}-s) \tag{1.3}$$

$$H(s) = \begin{cases} 1 \text{ if } s > 0 \\ 0 \text{ if } s \le 0 \end{cases} \tag{1.4}$$

The duration of the absolute refractoriness is set by δ^{abs}, during which large constant K ensures that the membrane potential is vastly above the threshold value. Constant n_0 scales the duration of the negative after-potential. Having a description of what happens to a neuron when it fires itself we need one for the effect of incoming postsynaptic potentials.

$$\varepsilon_{ij}(s) = \left[\exp\left(-\frac{s-\Delta^{ij}}{\tau_m}\right) - \exp\left(-\frac{s-\Delta^{ij}}{\tau_s}\right)\right]H(s-\Delta^{ij}) \tag{1.5}$$

In equation 1.5 Δ^{ij} defines the transmission delay (axons and dendrites are fast, synapses relatively slow) and $0<\tau_s<\tau_m$ are time constants defining the duration of the effect of the postsynaptic potential. Kernel ε by default describes the effect of an excitatory postsynaptic potential; by using the negative value, we can model an IPSP from an inhibitory synapse (see fig. 1c). We use variable w_{ij} to model the synaptic efficacy or weight; with which we also can model inhibitory connections by using values lower than zero. It should be noted that real synapses are either excitatory or inhibitory; we know of no synapses changing effect during lifetime.

Neurons of the second generation work in an iterative, clock-based manner of digital computers, but can deal with analog input values; we can quite easily feed input neurons with digitised values from a dataset or a robot-sensor. Due to their iterative nature these networks are not very well suited for temporal tasks; they do not use time in their computation, whereas spiking neural networks do. However, such values cannot just be fed into a spiking neuron, in some way we'll either have to convert this information into spikes, or have to employ a method to alter the membrane-potential directly.

A general approach to achieve the latter is to use an extra function h^{ext} to describe the effect of an external influence on the membrane potential. These functions usually are too task-specific to be covered in this paper, so this leaves us with

$$h(t) = h^{ext}(t) + \sum_{j\in\Gamma_i}\sum_{t_j^{(f)}\in F_j} w_{ij}\varepsilon_{ij}(t-t_j^{(f)}) \tag{1.6}$$

For non-hardware solutions it might proof handier to convert analog signals into spikes that can be fed to the network directly. An often-used solution is to apply a Poisson-process for spike generation by the sensor neuron; a higher input signal correlates with a higher chance for a spike. Such a spike will then be processed and affect the membrane-potential of neurons normally. The current excitation of a neuron is described by

$$u_i(t) = \sum_{t_i^{(f)}\in F_i} n_i(t-t_i^{(f)}) + h(t) \tag{1.7}$$

where the refractory state, effects of incoming postsynaptic potentials and eventual external events are combined. Together with equation 1.2 this forms the spike-response model, a powerful though easy to implement model for working with spiking neural networks.

Short-term memory neurons

Analysis of neural networks has always been difficult and is even more so in a spatial-temporal domain as with networks of spiking neurons. An often-used simplification of the spike-response model only takes the refractory effects of the last pulse sent into account. Mathematically speaking, by replacing equation 1.7 with

$$u_i(t) = n_i(t-\hat{t}_i) + h(t) \tag{1.8}$$

we are already finished. Forgetting about the effects of earlier refractory periods is not a capital crime; for normal operation

the model is still quite realistic, while analysis is been made easier. Due to the 'bad' memory of this model these are called 'short term memory' neurons.

Integrate-and-fire neurons

The most widely used and best-known model of threshold-fire neurons, and spiking neurons in general, is the integrate-and-fire neuron [5,6]. This model is based on, and most easily explained by, principles of electronics. Figure 3 shows schematic drawings of both a real and an integrate-and-fire neuron. A spike travels down the axon and is transformed by a low-pass filter, which converts the short pulse into a current pulse $I(t\text{-}t_j^{(f)})$ that charges the integrate-and-fire circuit. The resulting increase in voltage there can be seen as postsynaptic potential $\varepsilon(t\text{-}t_j^{(f)})$. Once the voltage over the capacitor goes above threshold value ϑ the neuron sends out a pulse itself. Mathematically we write

$$\tau_m \frac{\partial u}{\partial m} = -u(t) + RI(t) \qquad (1.9)$$

to describe the effects on membrane potential u over time, with τ_m being the membrane time constant in which voltage 'leaks' away. As with the spike-response model the neuron fires once u crosses threshold ϑ and a short pulse δ is generated. To force a refractory period after firing we set u to $K<0$ for a period of δ^{abs}.

$$I_i(t) = \sum_{j \in \Gamma_i} c_{ij} \sum_{t_j^{(f)} \in F_j} \delta(t - t_j^{(f)}) \qquad (1.10)$$

The input current I for neuron i will often be 0, as incoming pulses have a finite short length. Once a spike arrives, it is multiplied by synaptic efficacy factor c_{ij} forming the post-synaptic potential that charges the capacitor. This model is computationally simple and can easily be implemented in hardware. It is closely linked to the more general spike-response model and can be used like it by rewriting it into the correct kernels η and ε [8].

Spiking neurons in hardware

Very Large-Scale Integration (VLSI) technology integrates many powerful features into a small microchip like a microprocessor. Such systems can use data representations of either binary (digital VLSI) or continuous (analog VLSI) voltages. Progress in digital technology has been tremendous, providing us with ever faster, more precise and smaller equipment. In digital systems an energy-hungry synchronisation clock makes it certain that parts are ready for action. Analog systems consume much less power and space on silicon than digital systems (in many orders of magnitude) and are easily interfaced with the analog real world. However, their design is hard, due to noise computation is fundamentally (slightly) inaccurate and sufficiently reliable non-volatile analog memory does not (yet) exist [20,4].

Noise is the influence of random effects that affect *everything* in the real world that operates in normal (so, above the absolute zero) working environment temperatures. For digital systems this is not much of a problem, as extra precision can be acquired by using more bits for more precise data encoding. In analog systems such a simple counter-measure is not at hand, there are no practical ways of eliminating noise; at normal temperatures noise has to be accepted as a fact of life. Our brain is a perfect example of an analog system that operates quite well with noise, like neural networks do in general.

In fact, performance of neural networks increases with noise present [11]. Spiking neuron models can easily be equipped with noise-models like noisy threshold, reset or integration. The interested reader can find more de-tails on the modelling of noise in spiking neurons in Gerstner's excellent review on neuron models [8].

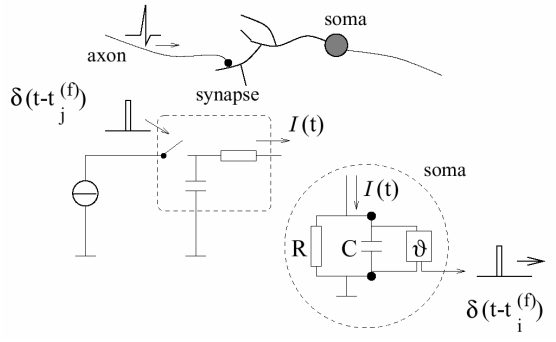

Fig. 3. Schematic drawing of the integrate-and-fire neuron. On the left side, the low-pass filter that transforms a spike to a current pulse $I(t)$ that charges the capacitor. On the right, the schematic version of the soma, which generates a spike when voltage u over the capacitor crosses threshold [10].

Hybrid systems can provide a possibly perfect solution, operating with reliable digital communication and memory while using fast, reliable and cheap analog computation and interfacing. In such a solution, neurons can send short digital pulses, much like we've seen before in the integrate-and-fire model. This model can be implemented in VLSI systems quite well [2]. VLSI systems usually work parallel, a very welcome fact for simulation of neural systems, which are inherently massively parallel. A significant speed gain can be acquired by using a continuous hardware solution; by definition digital simulation will have to recalculate each time-slice iteratively [20]. Though computer simulations have an advantage in adaptability, scaling a network up to more neurons (1000+) often means leaving the domain of real-time simulation. VLSI systems can be specifically designed to be able to link up, easily forming a scalable set-up that consists of many parts operating like they are one big system [2,11,20].

Synaptic plasticity

We saw that synapses are very complex signal pre-processors that they play an important role in development, memory and learning of neural structures. Synaptic plasticity is a form of change of the pre-processing, which is a preferred word for *'learning'* as it better describes what is at hand: long- or short-term change in synaptic efficacy [1,4,24].

Hebbian plasticity is a local form of long-term potentiation (LTP) and depression (LTD) of synapses and is based on the correlation of firing activity between pre- and postsynaptic neurons. This is usually, and easily, implemented with rate coding; similar neuron activity means a strong correlation. As we are using pulse-coding schemes, we have to think about how to define correlations in neural activity using single spikes. Pure Hebbian plasticity acts locally at each individual synapse, making it both very powerful and difficult to control; it is a positive-feedback process that can destabilize postsynaptic firing rates by endlessly strengthening effective and weakening ineffective synapses. If possible one has to avoid such behaviour, most desirably by a biologically plausible local rule.

Spike-timing dependent synaptic plasticity (STDP) is a form of competitive Hebbian learning that uses the exact spike timing information [1,23]. Experiments in neuroscience have shown that long-term strengthening occurs if presynaptic action potentials arrive within 50ms before of a postsynaptic spike and a weakening if it arrives late. Due to this mechanism STDP can lead to stable distributions of LTP and LDP, making postsynaptic neurons sensitive to the tim-

ing of incoming action potentials. This sensitivity leads to competition among the presynaptic neurons, resulting in shorter latencies, spike synchronization and faster information propagation through the network [1,23].

Hebbian plasticity is a form of unsupervised learning, which is useful for clustering input data but less appropriate when a desired outcome for the network is known in advance. Back-propagation [21] is a widely known and often used supervised learning algorithm. Due to the very complex spatial-temporal dynamics and continuous operation it cannot be directly applied to spiking neural networks, adaptations [12] exist in which individual spikes and their timing are taken into account.

Discussion

Neural structures are very well suited for complex information processing. Animals show an incredible ease in coping with dynamic environments, raising interest for the use of artificial neural networks in tasks that deal with real-world interactions. Over the years, three generations of artificial neural networks have been proposed, each generation biologically more realistic and computationally more powerful. Spiking neural networks use the element of time in communicating by sending out individual pulses. Spiking neurons can therefore multiplex information into a single stream of signals, like the frequency and amplitude of sound in the auditory system [9].

We have covered the very general and realistic spike-response model as well as the more common integrate-and-fire model for using pulse coding in neurons. Both models are powerful, realistic and easy to implement in both computer simulation and hardware VLSI systems. Standard neural network training algorithms use rate coding and cannot be directly used satisfactory for spiking neural networks. Spike-timing dependent synaptic plasticity uses exact spike timing to optimise information-flow through the network, as well as impose competition between neurons in the process of unsupervised Hebbian learning.

Pulse coding is computationally powerful [15,16,17] and very promising for tasks in which temporal information needs to be processed. We conclude with the remark that this is the case for virtually any task or application that deals with in- or output from the real world.

References

1. Abbot, L. F. & Nelson, S. B. *Synaptic Plasticity: taming the beast*, Nature Neuroscience Review, vol. 3 p.1178-1183 (2000).
2. Christodoulou C., Bugmann, G. & Clarkson, T. G. *A spiking neuron model: applications and learning*, Neural Networks, in press (2002).
3. DasGupta, B. & Schnitger, G. *The power of approximating: a comparison of activation functions*, Advances in Neural Information Processing Systems, vol. 5 p.363-374 (1992).
4. Elias, J. G. & Northmore, D. P. M. *Building Silicon Nervous Systems with Dendritic Tree Neuromorphs* in Maass, W. & Bishop, C. M. (eds.) *Pulsed Neural Networks*, MIT-press (1999).
5. Feng, J. & Brown, D. *Integrate-and-fire Models with Nonlinear Leakage*, Bulletin of Mathematical Biology vol. 62, p.467-481 (2000).
6. Feng, J. *Is the integrate-and-fire model good enough? – a review*, Neural Networks vol. 14, p.955-975 (2001).
7. Ferster, D. & Spruston, N. *Cracking the neuronal code*, Science, vol. 270 p.756-757 (1995).
8. Gerstner, W. *Spiking Neurons* in Maass, W. & Bishop, C. M. (eds.) *Pulsed Neural Networks*, MIT-press (1999).
9. Gerstner, W. , Kempter, R., Leo van Hemmen, J. & Wagner, H. *Hebbian Learning of Pulse Timing in the Barn Owl Auditory System* in Maass, W. & Bishop, C. M. (eds.) *Pulsed Neural Networks*, MIT-press (1999).
10. Gerstner, W., Kistler, W. *Spiking Neuron Models*, Cambridge University Press (2002).
11. Horn, D. & Opher, I. *Collective Excitation Phenomena and Their Applications* in Maass, W. & Bishop, C. M. (eds.) *Pulsed Neural Networks*, MIT-press (1999).
12. Kempter, G. & van Hemmen, J. L. *Hebbian Learning and Spiking Neurons*, Physical Review E4, p. 4498-4514 (1999).
13. Kruger, J. & Aiple, F. *Multimicroelectrode investigation of monkey striate cortex: spike train correlations in the infra-granular layers,* Journal of Neurophysiology vol. 60, p.798-828 (1988).
14. Maass, W., Schnitger, G. & Sontag, E. *On the computational power of sigmoid versus boolean threshold circuits*, Proc. of the 32nd Annual IEEE Symposium on Foundations of Computer Science, p. 767-776 (1991).
15. Maass, W. *The Third Generation of Neural Network Models*, Technische Universität Graz (1997).
16. Maass, W. *Computation with Spiking Neurons* in Maass, W. & Bishop, C. M. (eds.) *Pulsed Neural Networks*, MIT-press (1999).
17. Maass, W. On the Computational Power of Recurrent Circuits of Spiking Neuron, Technische Universität Graz (2000).
18. Maass, W. Synapses as Dynamic Memory Buffers, Technische Universität Graz (2000).
19. Maass, W. *Computing with Spikes*, Technische Universität Graz (2002).
20. Murray, A. F. *Pulse-Based Computation in VLSI Neural Networks* in Maass, W. & Bishop, C. M. (eds.) *Pulsed Neural Networks*, MIT-press (1999).
21. Rumelhart, D.E., Hinton, G.E. & Williams, R.J. *Learning representations by back-propagating errors,* Nature Vol. 323 (1986).
22. Sala, D., Cios, K. & Wall, J. *Self-Organization in Networks of Spiking Neurons*, University of Toledo, Ohio (1998).
23. Song, S., Miller, K. D. & Abbott, L. F. *Competetive Hebbian Learning through spike-timing-dependent synaptic plasticity*, Nature Neuroscience, vol. 3 p. 919-926 (2000).
24. Thorpe, S., Delorme, A., Van Rullen, R. *Spike based strategies for rapid processing*, Neural Networks, vol. 14(6-7), p.715-726 (2001).