Paradigms for Computing with Spiking Neurons

Wolfgang Maass

1 Introduction

Spiking neurons differ in essential aspects from the familiar computational units of common neural network models, such as McCulloch-Pitts neurons or sigmoidal gates. Therefore the question arises how one can *compute* with spiking neurons, or with related computational units in electronic hardware, whose input and output consists of trains of pulses. Furthermore the question arises how the computational power of networks of such units relates to that of common reference models, such as threshold circuits or multi-layer perceptrons. Both of these questions will be addressed in this chapter.

2 A Formal Computational Model for a Network of Spiking Neurons

A neuron model whose dynamics is described in terms of differential equations, such as the Hodgkin-Huxley model, the Fitzhugh-Nagumo model, or the integrate-and-fire model, makes it quite difficult to analyze computations in networks of such neurons. Hence the formulation of the spike-response model (see [Gerstner and van Hemmen, 1992, Gerstner, 1999]) has greatly facilitated the investigation of computations in networks of spiking neurons. Although this model has a mathematically much simpler formulation, it is able to capture the dynamics of Hodgkin-Huxley neurons quite well [Kistler et al., 1997]. In addition, by choosing suitable response functions one can readily adapt this model to reflect the dynamics of a large variety of different biological neurons.

For the sake of completeness we quickly review the definition of the spike response model. Let I be a set of neurons, and assume that one has specified for each neuron $i \in I$ a set $\Gamma_i \subseteq I$ of immediate predecessors ("presynaptic neurons") in the network. The firing times $t_i^{(f)} \in \mathcal{F}_i$ for all neurons $i \in I$ are defined recursively (by simultaneous recursion along the time axis). The

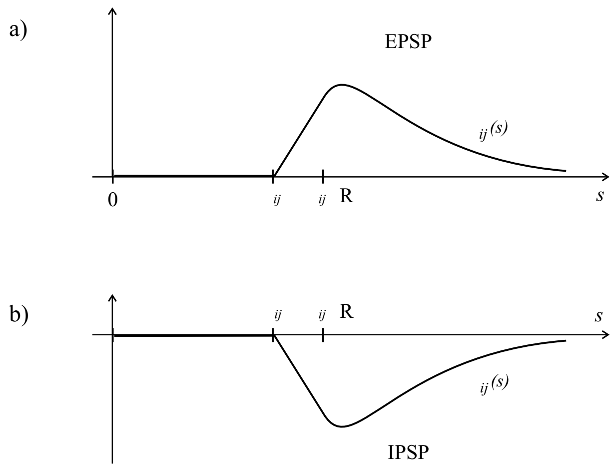

FIGURE 1. a) Typical shape of an excitatory postsynaptic potential (EPSP). b) Typical shape of an inhibitory postsynaptic potential (IPSP). The delay Δ_{ij} and the length R of such PSP will turn out to be important for the computations discussed in sections 4 and 5.

neuron i fires whenever the state variable

$$u_i(t) = \sum_{t_i^{(f)} \in \mathcal{F}_i} \eta_i(t - t_i^{(f)}) + \sum_{j \in \Gamma_i} \sum_{t_j^{(f)} \in \mathcal{F}_j} w_{ij}\, \epsilon_{ij}(t - t_j^{(f)})\,. \tag{1.1}$$

reaches the firing threshold ϑ of neuron i.

The response functions $\epsilon_{ij}(t - t_j^{(f)})$ model excitatory and inhibitory postsynaptic potentials (EPSP's and IPSP's, see Figure 1) at the soma of neuron i, which result from the firing of a presynaptic neuron j at time $t_j^{(f)}$. The function η_i models the response of neuron i to its own firing, in particular refractory effects. One typically assumes that η_i assumes a strongly negative value for values of $t - t_i^{(f)}$ in the range of a few ms, and then gradually returns to a value near 0 [Gerstner, 1999]. If the neuron i has not fired for a while, one can ignore the first summand with the refractory terms $\eta_i(t - t_i^{(f)})$ in (1.1).

In order to complete the definition of a network of spiking neurons as a formal computational model one has to specify its network input and output. We assume that subsets of neurons $I_{input} \subseteq I$ and $I_{output} \subseteq I$ have been fixed, and that the firing times $\mathcal{F}_i$ for the neurons $i \in I_{input}$ constitute the network input. Thus we assume that these firing times are determined through some external mechanism, rather than computed according to the previously described rules. The firing times $\mathcal{F}_i$ of the neurons $i \in I_{output}$ constitute the network output. These firing times are computed in the previously described way (like for all neurons $i \in I - I_{input}$) with the help of the state variable (1.1).

Thus from a mathematical point of view, a network of spiking neurons computes a function which maps a vector of several time series $\langle \mathcal{F}_i \rangle_{i \in I_{input}}$ on a vector of several other time series $\langle \mathcal{F}_i \rangle_{i \in I_{output}}$. In the first part of this chapter we will focus on the mathematically simpler case where these time series $\mathcal{F}_i$ consist of at most one spike each.

3 McCulloch-Pitts Neurons versus Spiking Neurons

The simplest computational unit of traditional neural network models is a *McCulloch-Pitts neuron*, also referred to as *threshold gate* or *perceptron*. A McCulloch-Pitts neuron i with real valued weights α_{ij} and threshold ϑ receives as input n binary or real valued numbers $x_1, \ldots, x_n$. Its output has the value

$$\begin{cases} 1 \ , & \text{if} \quad \sum_{j=1}^{n} \alpha_{ij} \cdot x_j \geq \vartheta \\ 0 \ , & \text{otherwise} \ . \end{cases} \tag{1.2}$$

In multilayer networks one usually considers a variation of the threshold gate to which we will refer as a *sigmoidal gate* in the following. The output of a sigmoidal gate is defined with the help of some non-decreasing continuous *activation function* $g : \mathbf{R} \to \mathbf{R}$ with bounded range as

$$g(\sum_{j=1}^{n} \alpha_{ij} \cdot x_j - \vartheta) \ . \tag{1.3}$$

By using sigmoidal gates instead of threshold gates one can not only compute functions with analog output, but also increase the computational power of neural nets for computing functions with boolean output [Maass et al., 1991; DasGupta and Schnitger, 1996].

In the following we will compare the computational capabilities of these two computational units of traditional neural network models with that of a spiking neuron. One immediately sees that a spiking neuron i can in principle simulate any given threshold gate (1.2) with positive threshold ϑ for binary input. For that we assume that the response functions $\epsilon_{ij}(x)$ are all identical except for their sign (which we choose to be positive if $\alpha_{ij} > 0$ and negative if $\alpha_{ij} \leq 0$), and that all presynaptic neurons j which fire, fire at the same time $t_j = T_{input}$. In this case the spiking neuron i fires if and only if

$$\sum_{j \text{ fires at time } T_{input}} w_{ij} \cdot \epsilon_{ij} \geq \vartheta \ , \tag{1.4}$$

where ϵ_{ij} is the extremal value of $\epsilon_{ij}(s)$ (i.e., $\epsilon_{ij} = \max_s \epsilon_{ij}(s)$ if $\epsilon_{ij}(s)$ represents an EPSP, $\epsilon_{ij} = min_s \ \epsilon_{ij}(s)$ if $\epsilon_{ij}(s)$ represents an IPSP), $w_{ij} \geq 0$

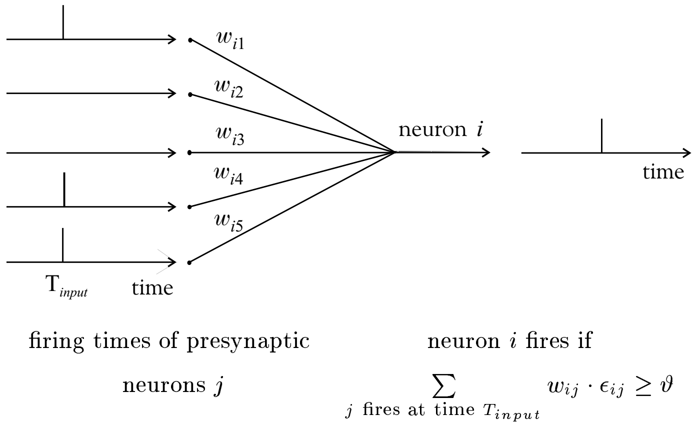

FIGURE 2. Simulation of a threshold gate by a spiking neuron.

is the synaptic weight, and $\vartheta > 0$ is the firing threshold of neuron i, see Figure 2. Then for $w_{ij} := \alpha_{ij}/\epsilon_{ij}$ the spiking neuron i can simulate any given threshold gate defined by (1.2) if the input bits $x_1, \ldots, x_n$ are encoded by the firing or non-firing of presynaptic neurons $j = 1, \ldots, n$ at a common time T_{input}, and if the output bit of the threshold gate is encoded by the firing or non-firing of the spiking neuron i during the relevant time window.

A closer look shows that it is substantially more difficult to simulate in the same manner a *multi-layer* network of threshold gates (i.e., a threshold circuit) by a network of spiking neurons. The exact firing time of the previously discussed spiking neuron i depends on its concrete input $x_1, \ldots, x_n$. If $\sum_{j \text{ fires at time } T_{input}} w_{ij} \cdot \epsilon_{ij} - \vartheta$ has a value well above 0, then the state variable $u_i(t) = \sum_{j \text{ fires at time } T_{input}} w_{ij} \cdot \epsilon_{ij}(t - T_{input})$ will cross the firing threshold ϑ earlier, yielding an *earlier* firing time of neuron i, compared with an input where $\sum_{j \text{ fires at time } T_{input}} w_{ij} \cdot \epsilon_{ij} - \vartheta$ is positive but close to 0. Therefore, if one employs several spiking neurons to simulate the threshold gates on the first layer[1] of a threshold circuit, those neurons on the first layer which do fire (corresponding to threshold gates with output 1) will in general fire at slightly different time points. This will occur even if all input neurons j of the network have fired at the same time T_{input}. Therefore the timing of such straightforward simulation of a multi-layer threshold circuit is unstable: even if all firings in one layer occur synchronously, this synchronicity will in general get lost at the next layer. Similar problems arise

[1]We will not count the layer of input neurons in this chapter, and hence refer to the *first hidden layer* as the *first layer* of the network.

in a simulation of other types of multi-layer boolean circuits by networks of spiking neurons.

Consequently one needs a separate *synchronization mechanism* in order to simulate a multi-layer boolean circuit – or any other common model for digital computation – by a network of spiking neurons with bits $0, 1$ encoded by firing and nonfiring. One can give a mathematically rigorous proof that such synchronization mechanism can in principle be provided by using some auxiliary spiking neurons [Maass, 1996]. This construction exploits the simple fact that the double-negation $\neg\neg b$ of a bit b has the same value as b. Therefore instead of making a spiking neuron i fire in the direct way described by equation (1.4), one can make sure that if $\sum_j \alpha_{ij} \cdot x_j \geq \vartheta$, then the spiking neuron i is *not prevented* from firing by auxiliary inhibitory neurons. These auxiliary inhibitory neurons are connected directly to the input neurons whose firing/nonfiring encodes the input bits $x_1, \ldots, x_n$, whereas the driving force for the firing of neuron i comes from input-independent excitatory neurons. With this method one can simulate any given boolean circuit, and in the absence of noise even any Turing machine, by a finite network of spiking neurons [Maass, 1996] (we refer to [Judd and Aihara, 1993] for earlier work in this direction). On this basis one can also implement the constructions of [Valiant, 1994] in a network of spiking neurons.

Before we leave the issue of synchronization we would like to point out that in a practical context one can achieve a near-synchronization of firing times with the help of some common background excitation, for example an excitatory background oscillation $\sin(\omega\, t)$ that is added to the membrane potential of all spiking neurons i [Hopfield, 1995]. With proper scaling of amplitudes and firing thresholds one can achieve that for all neurons i the state variable $u_i(t)$ can cross the firing threshold ϑ only when the background oscillation $sin(\omega\, t)$ is close to its peak. Our preceding discussion also points to a reason why it may be *less advantageous* for a network of spiking neurons to employ a forced synchronization. The small temporal differences in the firing times of the neurons i that simulate the first layer of a threshold circuit according to equation (1.4) contain valuable *additional information* that is destroyed by a forced synchronization: these temporal differences contain information about *how much* larger the weighted sum $\sum_j \alpha_{ij} \cdot x_j$ is compared with ϑ. Thus it appears to be advantageous for a network of spiking neurons to employ instead of a synchronized digital mode an asynchronous or loosely synchronized analog mode where subtle differences in firing times convey additional *analog* information. We will discuss in the next section computational operations which spiking neurons can execute in this quite different computational mode, where *analog* values are encoded in *temporal patterns* of firing times.

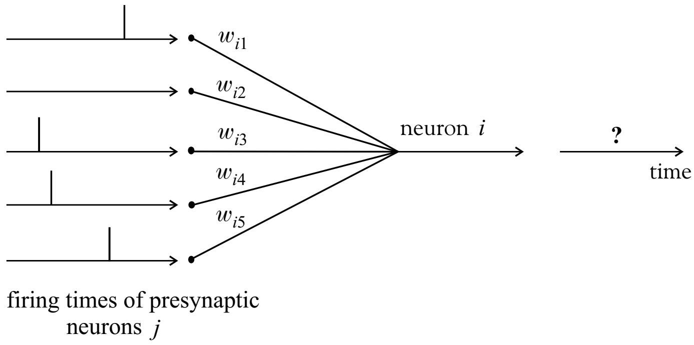

FIGURE 3. Typical input for a biological spiking neuron i, where its output cannot be easily described in terms of conventional computational units.

4 Computing with Temporal Patterns

4.1 Conincidence Detection

We will now consider the more typical scenario for a biological neuron, where preceding neurons do not fire in a synchronized manner, see Figure 3. In this case the computational operation of a spiking neuron cannot be easily described with the help of common computational operations or computational models.

We will show that in an asynchronous mode, with analog values encoded by a temporal pattern of firing times, a spiking neuron has in principle not only more computational power than a McCulloch-Pitts neuron, but also more computational power than a sigmoidal gate.

One new feature of a *spiking* neuron – which does not correspond to any feature of the computational units of traditional neural network models – is that it can act as *coincidence detector* for incoming pulses [Abeles, 1982]. Hence if the arrival times of the incoming pulses encode *numbers*, a spiking neuron can detect whether some of these numbers have (almost) equal value. On the other hand we will show below that this operation on numbers is a rather "expensive" computational operation from the point of view of traditional neural network models.

We will now make these statements more precise. Assume that $\{1, \ldots, n\} = \Gamma_i$ are the immediate predecessors of a spiking neuron i, that their connections to neuron i all have the same transmission delay Δ_{ij}, and that $w_{ij} = 1$ for all $j \in \Gamma_i$. Furthermore, assume that the response functions $\epsilon_{ij}(s)$ are defined in some common way (see [Gerstner, 1999]), for example as a difference of two exponentially decaying functions

$$\epsilon_{ij}(s) = \frac{1}{1 - (\tau_s/\tau_m)} \left[\exp\left(-\frac{s - \Delta_{ij}}{\tau_m} \right) - \exp\left(-\frac{s - \Delta_{ij}}{\tau_s} \right) \right] \mathcal{H}(s - \Delta_{ij})$$

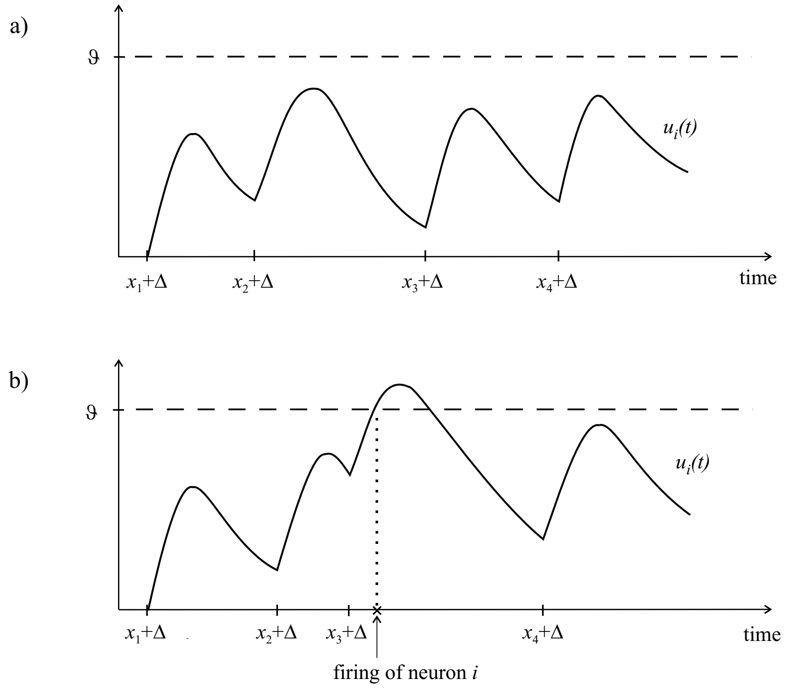

FIGURE 4. a) Typical time course of the state variable $u_i(t)$ if $ED_4(x_1, x_2, x_3, x_4) = 0$. b) Time course of $u_i(t)$ in the case where $ED_4(x_1, x_2, x_3, x_4) = 1$ because $|x_3 - x_2| \leq c_1$.

with time constants $0 < \tau_s < \tau_m$ and the heaviside function $\mathcal{H}$. A plot of an EPSP of such shape is shown in Figure 1a). It consists of an almost linearly rising phase for small s, exponential decay for large s, and a smooth transition between both phases when it reaches its maximal value in between.

For every given values of the time constants τ_s, τ_m with $\tau_s < \tau_m$ one can find values $0 < c_1 < c_2$ and ϑ so that $u_i(t) < \vartheta$ for any input consisting of an arbitrary number of EPSP's with distance $\geq c_2$, whereas $u_i(t)$ reaches a value $> \vartheta$ for two EPSP's in distance $\leq c_1$. Then the spiking neuron i does not fire if the neurons $j \in \Gamma_i$ fire (each at most once) in temporal distance $\geq c_2$ (see Figure 4a)), but it fires whenever two presynaptic neurons $j \in \Gamma_i$ fire in temporal distance $\leq c_1$, see Figure 4b)). Consequently, if for example one encodes n real numbers $x_1, \ldots , x_n$ through the firing times of the n neurons in Γ_i , and decodes the output of neuron i as "1" if it fires and "0" if it does not fire, the neuron i computes the following function

$ED_n : \mathbf{R}^n \to \{0,1\}$:

$$ED_n(x_1,\ldots,x_n) \;=\; \begin{cases} 1 & , \quad \text{if there are } j \neq j' \text{ so that } |x_j - x_{j'}| \leq c_1 \\ 0 & , \quad \text{if } |x_j - x_{j'}| \geq c_2 \text{ for all } j \neq j' \quad . \end{cases}$$

Note that this function $ED_n(x_1,\ldots,x_n)$ (where ED stands for "**e**lement **d**istinctness") is in fact a partial function, which may output arbitrary values in case that $c_1 < \min\{|x_j - x_{j'}| : j \neq j' \text{ and } j,j' \in \Gamma_i\} < c_2$. Therefore hair-trigger situations can be avoided, and a single spiking neuron can compute this function ED_n even if there is a small amount of noise on its state variable $u_i(t)$.

On the other hand the following results show that the same partial function ED_n requires a substantial number of neurons if computed by neural networks consisting of McCulloch-Pitts neurons (threshold gates) or sigmoidal gates. These lower bounds hold for *arbitrary* feedforward architectures of the neural net, and *any values* of the weights and thresholds of the neurons. The inputs $x_1,\ldots,x_n$ are given to these neural nets in the usual manner as analog input variables.

Theorem 1.1 *Any layered threshold circuit that computes ED_n needs to have at least* $\log(n!) \geq \frac{n}{2} \cdot \log n$ *threshold gates on its first layer.*

The *proof* of Theorem 1.1 relies on a geometrical argument, see [Maass, 1997b].

Theorem 1.2 *Any feedforward neural net consisting of arbitrary sigmoidal gates needs to have at least $\frac{n-4}{2}$ gates in order to compute ED_n.*

The *proof* of Theorem 1.2 is more difficult, since the gates of a sigmoidal neural net (defined according to (1.3) with some smooth gain function g) output *analog numbers* rather than *bits*. Therefore a multilayer circuit consisting of sigmoidal gates may have larger computational power than a circuit consisting of threshold gates. The proof procedes in an indirect fashion by showing that any sigmoidal neural net with m gates that computes ED_n can be transformed into another sigmoidal neural net that "shatters" *every* set of $n-1$ different inputs with the help of $m+1$ programmable parameters. According to [Sontag, 1997] this implies that $n-1 \leq 2(m+1)+1$. We refer to [Maass, 1997b] for further details. ∎

4.2 RBF-Units in the Temporal Domain

We have demonstrated in the preceding subsection that for some computational tasks a single spiking neuron has more computational power than a fairly large neural network of the conventional type. We will show in this subsection that the preceeding construction of a spiking neuron that detects coincidences among incoming pulses can be expanded to yield detectors for more complex temporal patterns.

Instead of a common delay Δ between presynaptic neurons $j \in \Gamma_i$ and neuron i (which appears in our formal model as the length of the initial flat part of the response function $\epsilon_{ij}(x)$) one can employ for different j *different* delays Δ_{ij} between neurons j and i. These delays Δ_{ij} represent a new set of parameters that have no counterpart in traditional neural network models[2]. There exists evidence that in some biological neural systems these delays Δ_{ij} can be tuned by "learning algorithms". In addition one can tune the firing threshold ϑ and/or the weights w_{ij} of a spiking neuron to increase its ability to detect specific temporal patterns in the input. In the extreme case one can raise the firing threshold ϑ so high that *all* pulses from presynaptic neurons have to arrive nearly simultaneously at the soma of i to make it fire. In this case the spiking neuron can act in the temporal domain like an RBF-unit (i.e., radial basis function unit) in traditional neural network models: it will fire only if all presynaptic neurons $j \in \Gamma_i$ fire at times t_j so that for some constant T_{input} one has $t_j \approx T_{input} - \Delta_{ij}$ for all $j \in \Gamma_i$, where the vector $(\Delta_{ij})_{j \in \Gamma_i}$ of transmission delays plays now the role of the *center* of an RBF-unit. This possibility of using spiking neurons as RBF-like computational units in the temporal domain was first observed by Hopfield [Hopfield, 1995]. In the same article Hopfield demonstrates an advantageous consequence of employing a *logarithmic* encoding $x_j = \log y_j$ of external sensory input variables y_j through firing times $t_j = T_{input} - x_j$. Since spiking neurons have the ability to detect temporal patterns irrespective of a common additive constant in their arrival times, they can with the help of logarithmic encoding ignore constant *factors* λ in sensory input variables $\langle \lambda \cdot y_j \rangle_{j \in \Gamma_i}$. It has been argued that this useful mechanisms may be related to the amazing ability of biological organisms to classify patterns over a very large scale of intensities, such as for example visual patterns under drastically different lighting conditions.

In [Natschläger and Ruf, 1998] the previous construction of an RBF-unit for temporal patterns has been extended to a an RBF-network with the help of lateral inhibition between RBF-units (see Section 2.4.5). Alternatively one can add linear gates on the second layer of an RBF-network of spiking neurons with the help of the construction described in the following section.

4.3 Computing a Weighted Sum in Temporal Coding

A characteristic feature of the previously discussed computation of the function ED_n and the simulation of an RBF-unit is the *asymmetry* between coding schemes used for *input* and *output*. Whereas the input consisted of a vector of analog numbers, encoded through temporal delays, the output

[2]Theoretical results about the Vapnik-Chervonenkis dimension (VC-dimension) of neural nets suggest that tuning of delays enhances the flexibility of spiking neurons for computations (i.e., the number of different functions they can compute) even more than tuning the weights [Maass and Schmitt, 1997].

of the spiking neuron was just binary, encoded through firing or nonfiring of that neuron. Obviously for multilayer or recurrent computations with spiking neurons it is desirable to have mechanisms that enable a layer of spiking neurons to *output* a vector of analog numbers encoded in the same way as the input. For that purpose one needs mechanisms for *shifting* the firing time of a spiking neuron i in dependence of the firing times t_j of presynaptic neurons, in a manner that can be controlled through the internal parameters w_{ij} and Δ_{ij}. As an example for that we will now describe a simple mechanism for computing for arbitrary parameters $\alpha_{ij} \in \mathbf{R}$ and inputs $x_j \in [0,1]$ the weighted sum $\sum\limits_j \alpha_{ij} \cdot x_j$ through the firing time of neuron i.

We assume that each response function $\epsilon_{ij}(s)$ has a shape as shown in Figure 1: $\epsilon_{ij}(s)$ has value 0 for $s \leq \Delta_{ij}$ and then rises approximately lineary (in the case of an EPSP) or descends approximately lineary (in the case of an IPSP) with slope $\lambda_{ij} \in \mathbf{R}$ for an interval of length at least $R > 0$. Assume that the presynaptic neurons $j \in \Gamma_i$ fire at times $t_j = T_{input} - x_j$. If the state variable $u_i(t) = \sum\limits_{j \in T_i} w_{ij} \cdot \epsilon_{ij}(t - t_j)$ of neuron i reaches the threshold ϑ at a time t_i when the response functions $\epsilon_{ij}(t - t_j)$ are all in their initial linear phase of length $\geq R$, then t_i is determined by the equation

$$\sum_{j \in \Gamma_i} w_{ij} \cdot \epsilon_{ij}(t_i - t_j) = \sum_{j \in \Gamma_i} w_{ij} \cdot \lambda_{ij} \cdot (t_i - t_j - \Delta_{ij}) = \vartheta \,. \tag{1.5}$$

Obviously (1.5) implies that

$$t_i \;=\; \frac{\vartheta}{\sum\limits_{j \in \Gamma_i} w_{ij} \cdot \lambda_{ij}} \;+\; \frac{\sum\limits_{j \in \Gamma_i} w_{ij} \cdot \lambda_{ij} \cdot (t_j + \Delta_{ij})}{\sum\limits_{j \in \Gamma_i} w_{ij} \cdot \lambda_{ij}} \;. \tag{1.6}$$

Then by writing λ for $\sum_{j \in \Gamma_i} w_{ij} \cdot \lambda_{ij}$ and expressing t_j as $T_{input} - x_j$ we get

$$t_i \;=\; \frac{\vartheta}{\lambda} + \sum_{j \in \Gamma_i} \frac{w_{ij} \cdot \lambda_{ij}}{\lambda} \cdot (T_{input} - x_j + \Delta_{ij}) \,,$$

or equivalently

$$t_i \;=\; T_{output} \;-\; \sum_{j \in \Gamma_i} \alpha_{ij} \cdot x_j \tag{1.7}$$

for some input-independent constant $T_{output} := \frac{\vartheta}{\lambda} + \sum\limits_{j \in \Gamma_i} \frac{w_{ij} \cdot \lambda_{ij}}{\lambda}(T_{input} + \Delta_{ij})$, and formal "weights" α_{ij} defined by $\alpha_{ij} := \frac{w_{ij} \cdot \lambda_{ij}}{\lambda}$. These "weights" α_{ij} are automatically normalized: by the definition of λ they satisfy

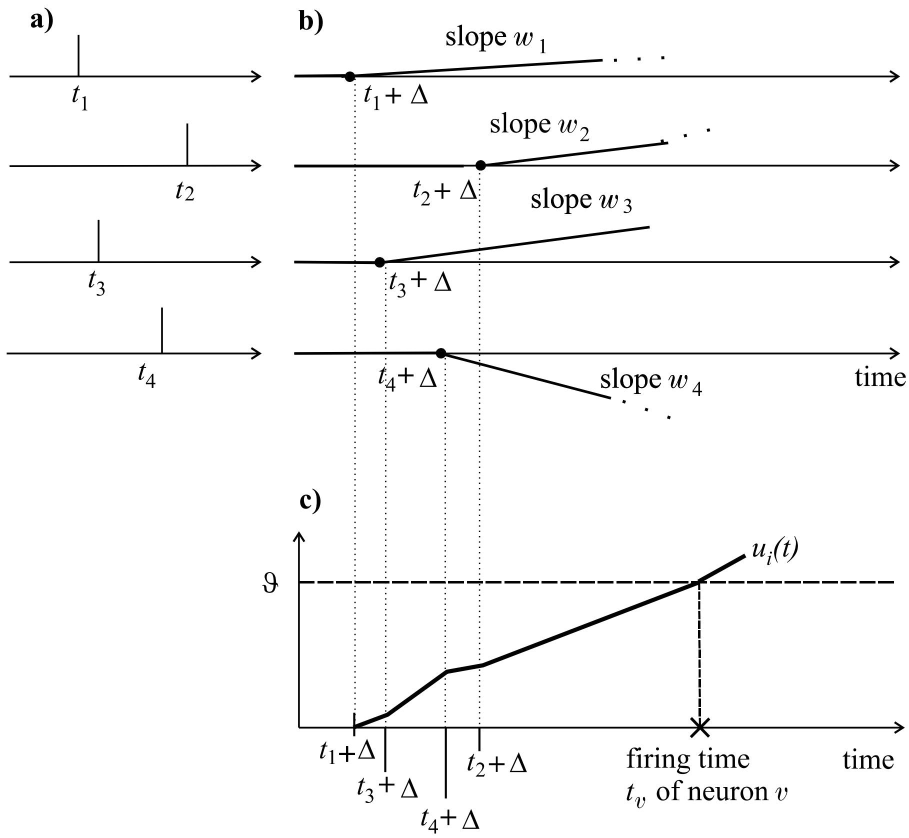

FIGURE 5. Mechanisms for computing a weighted sum in temporal coding according to equation (1.6). a) Firing times t_j of presynaptic neurons j. b) Initial linear segments of the weighted response functions $w_{ij} \cdot \epsilon_{ij}(t - t_j)$ at the soma of neuron i. c) State variable $u_i(t)$ and resulting firing time t_i of neuron i.

$\sum_{j \in \Gamma_i} \alpha_{ij} = 1$. Such automatic weight normalization may be desirable in some situations [Haefliger et al., 1997]. One can circumvent it by employing an auxiliary input neuron (see [Maass, 1997a]). In this way one can compute an arbitrary given weighted sum $\sum_{j \in T_i} \alpha_{ij} \cdot x_j$ in *temporal* coding by a *spiking* neuron. Note that in this construction the analog output $\sum_{j \in \Gamma_i} \alpha_{ij} \cdot x_j$ is encoded in exactly the same way as the analog inputs x_j.

4.4 Universal Approximation of Continuous Functions with Spiking Neurons in the Temporal Domain

We will show in this subsection that on the basis of the computational mechanism described in the preceding subsection one can build networks of spiking neurons that can approximate arbitrary given bounded continuous functions in the temporal domain. We first observe that one can expand the previously described mechanism for computing a weighted sum $\sum_{j \in \Gamma_i} \alpha_{ij} \cdot x_j$ in the temporal domain to yield for *temporal coding* also a simulation of an arbitrary given sigmoidal neuron with the piecewise linear gain function

$$sat(x) = \begin{cases} x & , \quad \text{if} \quad 0 \leq x \leq 1 \\ 0 & , \quad \text{if} \quad x \leq 0 \\ 1 & , \quad \text{if} \quad x \geq 1 \end{cases} \quad .$$

In this case we want that neuron i responds to firing of its presynaptic neurons at times $t_j = T_{input} - x_j$ by firing at time

$$t_i = T_{output} - sat(\sum_{j \in \Gamma_i} \alpha_{ij} \cdot x_j) \quad .$$

For that purpose one just needs auxiliary mechanisms that support an approximation of the given sigmoidal neuron in the saturated regions of its gain function *sat*, i.e. for $x \leq 0$ and $x \geq 1$. Translated into the temporal domain this requires that the spiking neuron i does not fire before some fixed time T (simulating $sat(x) = 1$ for $x \geq 1$) and by the latest at some fixed time $T_{output} > T$ (simulating $sat(x) = 0$ for $x \leq 0$). This can easily be achieved with the help of auxiliary spiking neurons. Computer simulations suggest that in a practical situation such auxiliary neurons may not even be necessary [Maass and Natschläger, 1997].

According to the preceding construction one can simulate any sigmoidal neuron with the piecewise linear gain function *sat* by spiking neurons with analog inputs *and* outputs encoded by temporal delays of spikes. Since inputs and outputs employ the same coding scheme, the outputs from a first layer of spiking neurons (that simulate a first layer of sigmoidal gates) can be used as inputs for another layer of spiking neurons, simulating another layer of sigmoidal gates. Hence on the basis of the assumption that the initial segments of response functions $\epsilon_{ij}(s)$ are linear one can show with a rigorous mathematical proof [Maass, 1997a]:

Theorem 1.3 *Any feedforward or recurrent analog neural net (for example any multilayer perceptron), consisting of s sigmoidal neurons that employ the gain function sat, can be simulated arbitrarily closely by a network of $s + c$ spiking neurons (where c is a small constant) with analog inputs and outputs encoded by temporal delays of spikes. This holds even if the spiking neurons are subject to noise.* ∎

Theorem 1.2 and 1.3 together exhibit an interesting *asymmetry* regarding the computational power of standard sigmoidal neural nets (multilayer perceptrons) and networks of spiking neurons: Whereas any sigmoidal neural net can be simulated by an insignificantly larger network of spiking neurons (with temporal coding), certain networks of spiking neurons can only be simulated by substantially larger sigmoidal neural nets.

It is wellknown that feedforward sigmoidal neural nets with gain function *sat* can approximate any given continuous function $F : [0,1]^n \rightarrow [0,1]^m$ with any desired degree of precision. Hence Theorem 1.3 implies:

Corollary 1.4 *Any given continuous function* $F : [0,1]^n \rightarrow [0,1]^m$ *can be approximated arbitrarily closely by a network of spiking neurons with inputs and outputs encoded by temporal delays.*

Remarks:

a) The construction that yields the proof of Theorem 1.3 shows that a network of spiking neurons can *change* the function: $F : [0,1]^n \rightarrow [0,1]^m$ that it computes in the same way as a traditional neural net: by changing the synaptic "weights" w_{ij} that scale the slopes of the initial segments of postsynaptic pulses. The delays Δ_{ij} between neurons need not be changed for that purpose (but they *could* be used to modulate the effective "weights" of the simulated sigmoidal neural net by additive constants). From that point of view this construction is complementary to the simulation of RBF-units by spiking neurons described in subsection 4.2: there the "program" of the encoded function was encoded exclusively in the delays Δ_{ij}.

b) It turns out that the network of spiking neurons constructed for the proof of Theorem 1.3 computes approximately the same function in rate-coding *and* in temporal coding.

4.5 Other Computations with Temporal Patterns in Networks of Spiking Neurons

The previously described method for emulating classical artificial neural networks in the temporal domain with spiking neurons can also be applied to Hopfield nets [Maass and Natschläger, 1997], Kohonen's self-organizing map [Ruf and Schmitt, 1998] and RBF-networks [Natschläger and Ruf, 1998]. The latter construction refines Hopfield's construction of an RBF-unit in the temporal domain. It simulates RBF-units by neurons that output an analog number (encoded in its firing time), rather than a single bit (encoded by firing/nonfiring). They implement a competition among different RBF-units through lateral inhibition. Furthermore they show through computer simulations that a variation of the Hebb-rule for spiking neurons with temporal coding, that has been experimentally observed for biological neurons (see [Markram and Sakmann, 1995] and [Markram et al., 1997]), yields good performance for unsupervised learning of temporal input patterns. It is of interest for applications that their RBF-network also exhibits some robustness with regard to warping of temporal input patterns.

Simon Thorpe and his collaborators have independently explored several options for computing with information encoded in the timing of spikes ([Thorpe and Imbert, 1989, Thorpe and Gautrais, 1997, Samuelides et al., 1997]). The goal of their constructions is that the output neurons of the net respond in a given way to the firing *order* of the input neurons. This is a special case of the computations with spiking neurons considered in Sections 4.3 and 4.4. Obviously each firing order is naturally encoded in the vector $\underline{x}$ of delays of these spikes. Hence any classification task for spike orders can be viewed as a special case of a classification task for delay vectors $\underline{x}$. Theorem 1.3 shows that networks of spiking neurons can apply to this task the full classification power of multilayer perceptrons.

Finally we would like mention that [Watanabe and Aihara, 1997] have explored *chaos* in the temporal pattern of firing in a network of spiking neurons.

5 Computing with a Space-Rate Code

The second model for fast analog computation with spiking neurons that we will discuss is more suitable for neural systems consisting of a large number of components which are not very reliable. In fact this model, which was recently developed in collaboration with Thomas Natschläger [Maass and Natschläger, 1998], *relies* on the assumption that individual synapses are “unreliable”. It takes into account evidence from [Dobrunz and Stevens, 1997] and others, which shows that individual synaptic release sites are highly stochastic: they release a vesicle (filled with neurotransmitter) upon the arrival of a spike from the presynaptic neuron u with a certain probability (called *release probability*). This release probability varies among different synapses between values less than 0.1 and more than 0.9.

This computational model is based on a space-rate encoding (also referred to as population coding) of analog variables, i.e., an analog variable $x \in [0, 1]$ is encoded by the percentage of neurons in a population that fire within a short time window (say, of length 5 ms).

Although there exists substantial empirical evidence that many cortical systems encode relevant analog variables by such space-rate code, it has remained unclear how networks of spiking neurons can *compute* in terms of such a code. Some of the difficulties become apparent if one just wants to understand for example how the trivial linear function $f(x) = x/2$ can be computed by such a network if the input $x \in [0, 1]$ is encoded by a space-rate code in a pool U of neurons and the output $f(x) \in [0, 1/2]$ is supposed to be encoded by a space-rate code in another pool V of neurons. If one assumes that all neurons in V have the same firing threshold and that reliable synaptic connections from all neurons in U to all neurons in V exist with approximately equal weights, a firing of a percentage x of neurons in U during a short time interval will typically trigger *almost none* or *almost all* neurons in V to fire, since they all receive about the same input from U.

Several mechanisms have already been suggested that could in principle achieve a *smooth* graded response in terms of a space-rate code in V instead of a binary "all or none" firing, such as strongly varying firing thresholds or different numbers of synaptic connections from U for different neurons $v \in V$ [Wilson and Cowan, 1972]. Both of these options would fail to spread average activity over all neurons in V, and hence would make the computation less robust against failures of individual neurons.

We assume that n pools $U_1, \ldots, U_n$ consisting of N neurons each are given, and that all neurons in these pools have synaptic connections to all neurons in another pool V of N neurons.[3] We assume that for each pool U_j all neurons in U_j are excitatory, or all neurons in U_j are inhibitory. We will first investigate the question which functions $\langle x_1, \ldots, x_n \rangle \to y$ can be computed by such a network if x_j is the firing probability of a neuron in pool U_j during a short time interval I_{input} and y is the firing probability of a neuron in pool V during a slightly later time interval I_{output}.

Consider an idealized mathematical model where all neurons which fire in the pool U_j fire synchronously at time T_{input}, and the probability that a neuron $v \in V$ fires (at time T_{output}) can be described by the probability that the sum h_v of the amplitudes of EPSP's and IPSP's resulting from firing of neurons in the pools $U_1, \ldots, U_n$ exceeds the firing threshold θ (which is assumed to be the same for all neurons $v \in V$). We assume in this section that the firing rates of neurons in pool V are relatively low, so that the impact of their refractory period can be neglected. We investigate refractory effects in section 6. The random variable (r.v.) h_v is the sum of random variables h_{vu} for all neurons $u \in \bigcup_{j=1}^{n} U_j$, where h_{vu} models the contribution of neuron u to h_v. We assume that h_{vu} is nonzero only if neuron $u \in U_j$ fires at time T_{input} (which occurs with probability x_j)[4] and if the synapse between u and v releases one or several vesicles (which occurs with probability r_{vu} whenever u fires). If both events occur then the value of h_{vu} is chosen according to some probability density function ϕ_{vu}. The functions ϕ_{vu}, as well as the parameters r_{vu}, are allowed to vary arbitrarily for different pairs u, v of neurons. For each neuron $v \in V$ we

[3]Our results remain valid if one considers instead connections by fixed random graphs with lower density between pools U_j and V.

[4]This holds if the pool size is large enough such that we can treat x_j (y) as the probability that a neuron $u \in U_j$ ($v \in V$) will fire once during a certain input (output) interval of length Δ.

consider the sum $h_v = \sum_{j=1}^{n} \sum_{u \in U_j} h_{vu}$ of the r.v.'s h_{vu} and we assume that v fires at time T_{output} if and only if $h_v \geq \theta$. Although the r.v.'s h_{vu} may have quite different distributions (for example due to different ϕ_{vu} and r_{vu}), their stochastic independence allows us to approximate the firing probability $\mathrm{P}\{h_v \geq \theta\}$ through a normal distribution Φ. The Berry-Esseen Theorem [Petrov, 1995] implies that

$$|\mathrm{P}\{h_v \geq \theta\} - (1 - \Phi(\theta;\, \mu_v, \sigma_v))| \;\leq\; 0.7915 \frac{\rho_v}{\sigma_v^3} \;, \tag{1.8}$$

where $\Phi(\cdot;\, \mu_v, \sigma_v)$ denotes the normal distribution function with mean μ_v and variance σ_v^2. The three moments occurring in (1.8) can be related to the r.v.'s h_{vu} through the equations $\mu_v = \sum_{j=1}^{n} \sum_{u \in U_j} \mathrm{E}[h_{vu}]$, $\sigma_v^2 = \sum_{j=1}^{n} \sum_{u \in U_j} \mathrm{Var}[h_{vu}]$, and $\rho_v = \sum_{j=1}^{n} \sum_{u \in U_j} \mathrm{E}[|h_{vu} - \mathrm{E}[h_{vu}]|^3]$. According to the definition of the r.v. h_{vu} we have $\mathrm{E}[h_{vu}] = x_j r_{vu} \bar{a}_{vu}$ and $\mathrm{E}\left[h_{vu}^2\right] = x_j r_{vu} \hat{a}_{vu}$ where $\bar{a}_{vu} = \int a \phi_{vu}(a) da$ denotes the mean EPSP (IPSP) amplitude and $\hat{a}_{vu} = \int a \phi_{vu}(a) da$ denotes the second moment. Hence we can assign to μ_v and σ_v in (1.8) the values

$$\mu_v = \sum_{j=1}^{n} \sum_{u \in U_j} x_j r_{vu} \bar{a}_{vu} \quad , \tag{1.9}$$

$$\sigma_v^2 = \sum_{j=1}^{n} \sum_{u \in U_j} (x_j r_{vu} \hat{a}_{vu} - x_j^2 r_{vu}^2 \bar{a}_{vu}^2) \quad . \tag{1.10}$$

A closer look reveals that the right hand side of (1.8) scales like $N^{-1/2}$.[5] Hence, equation (1.8) implies that for large N we can approximate the firing probability $\mathrm{P}\{h_v \geq \theta\}$ by the term $1 - \Phi(\theta;\, \mu_v, \sigma_v)$, which smoothly grows with μ_v. The gain of this sigmoidal function depends on the size of σ_v. In particular, if synaptic transmission were reliable, this function would degenerate to a step function. With the definition of the formal weights $w_{vj} := \sum_{u \in U_j} r_{vu} \bar{a}_{vu}$ we have $\mu_v = \sum_{j=1}^{n} w_{vj} x_j$, and hence $1 - \Phi(\theta;\, \mu_v, \sigma_v)$ smoothly grows with the weighted sum $\sum_{j=1}^{n} w_{vj} x_j$ of the inputs x_j.

So far we have just considered the probability $\mathrm{P}\{h_v \geq \theta\}$ that a single neuron $v \in V$ will fire, but we are really interested in the expected fraction y of neurons in pool V which will fire in response to a firing of a fraction x_j of neurons in the pools U_j for $j = 1, \ldots, n$. According to equation (1.8)

[5] More precisely, the right hand side of (1.8) scales like $N^{-1/2}$ if for all N the average value of the terms $\mathrm{E}[|h_{vu} - \mathrm{E}[h_{vu}]|^3]$ is uniformly bounded from above and the average value of the terms $\mathrm{Var}[h_{vu}]$ is uniformly bounded from below by a constant > 0 for $j \in \{1, \ldots, n\}$ and $u \in \bigcup_{j=1}^{n} U_j$. The latter can only be achieved for inputs where $x_j > 0$ for some j, since otherwise $\mathrm{Var}[h_{vu}] = 0$ for all $u \in U_j$ and all $j \in \{1, \ldots, n\}$. But in the case $x_j = 0$ for all $j \in \{1, \ldots, n\}$ both $\mathrm{P}\{h_v \geq \theta\}$ and $1 - \Phi(\theta;\, \mu_v, \sigma_v)$ have value 0 if $\theta > 0$ and hence the left hand side of (1.8) has value 0.

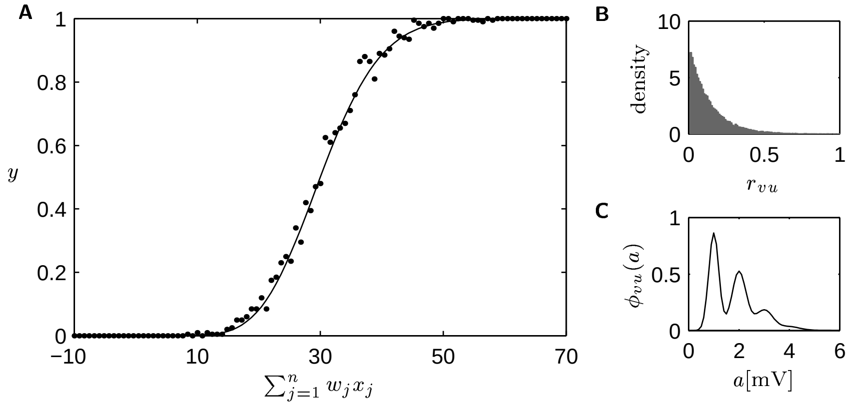

FIGURE 6. **A** Computer simulation of the model described in section 5 with a time interval Δ of length $\Delta = 5\,\text{ms}$ for space-rate coding and neurons modeled by the spike response model of [Gerstner, 1999]. We have chosen $n = 6$, a pool size $N = 200$ and $\langle w_1, \ldots, w_6 \rangle = \langle 10, -20, -30, 40, 50, 60 \rangle$ for the effective weights. Each dot is the result of a simulation with an input $\langle x_1, \ldots, x_6 \rangle$ selected randomly from $[0, 1]^6$ in such a way that $\sum_{j=1}^{n} w_j x_j$ covers the range $[-10, 70]$ almost uniformly. The y-axis shows the fraction y of neurons in pool V that fire during a 5 ms time interval in response to the firing of a fraction x_j of neurons in pool U_j for $j = 1, \ldots, 6$ during an earlier time interval of length 5 ms. **B** Distribution of non-failure probabilities r_{vu} for synapses between the pools U_4 and V underlying this simulation. **C** Example of a probability density function ϕ_{vu} of EPSP amplitudes as used for this simulations. This corresponds to a synapse with 5 release sites and a release probability of 0.3.

one can approximate y for sufficiently large pool sizes N by

$$y = \frac{1}{N} \sum_{v \in V} \mathrm{P}\{h_v \geq \theta\} = \frac{1}{N} \sum_{v \in V} 1 - \Phi(\theta;\, \mu_v, \sigma_v) \ .$$

Hence y is approximated by an average of the N sigmoidal functions $1 - \Phi(\theta;\, \mu_v, \sigma_v)$. If the weights w_{vj} have similar values for different $v \in V$ one can expect that y grows smoothly with the weighted sum $\bar{\mu} = \sum_{j=1}^{n} w_j x_j$, where we write $w_j = \sum_{v \in V} w_{vj}/N$ for the "effective weights" w_j between the pools of neurons U_j and V.

In order to test these theoretical predictions for an idealized mathematical model we have carried out computer simulations of a more detailed model consisting of more realistic models for spiking neurons and time intervals I_{input} and I_{output} of length $\Delta = 5\,\text{ms}$ for space-rate coding (see Figure 6).

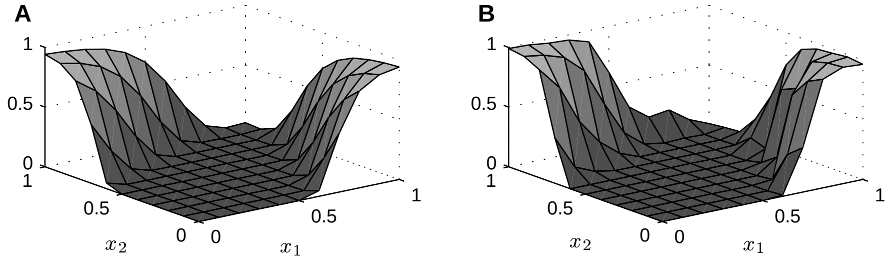

FIGURE 7. **A**) Plot of a function $f(x_1, x_2) : [0,1]^2 \to [0,1]$ which interpolates XOR. Since f interpolates XOR it cannot be computed by a single sigmoid unit. **B**) Computation of f by a 3-layer network in space-rate coding with spike response model neurons ($N = 200$) according to our model (for details see [Maass and Natschläger, 1998]).

5.1 Multi-Layer Computations

The preceding arguments show that approximate computations of functions of the form $\langle x_1, \ldots, x_n \rangle \to y = \sigma\left(\sum_{j=1}^{n} w_j x_j\right)$, with inputs and output in space-rate code, can be carried out within 10 ms by a network of spiking neurons. Hence the universal approximation theorem for multi layer perceptrons implies that *arbitrary continuous functions* $f : [0,1]^n \to [0,1]^m$ can be approximated with a computation time of not more than 20 ms by a network of spiking neurons with 3 layers. Thus our model provides a possible theoretical explanation for the empirically observed very fast multi-layer computations in biological neural systems that were mentioned in the introduction.

Results of computer simulations of the computation of a specific function f in space-rate coding that requires a multi-layer network because it interpolates the boolean function XOR is shown in Figure 7.

6 Analog Computation on Time Series in a Space-Rate Code

We have shown that biological networks of spiking neurons with space-rate coding have at least the computational power of multi layer perceptrons. In this section we will demonstrate that they have strictly more computational power. This becomes clear if one considers computations on *time series*, rather than on *static batch inputs*.

We now analyze the behavior of our computational model if the firing probabilities in the pools U_i change with time. Writing $x_i(t)$ $(y(t))$ for the probability that a neuron in pool U_i (V) fires during the t-th time window

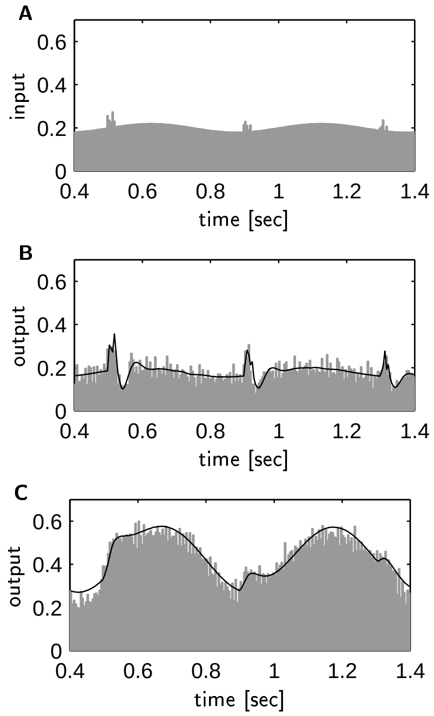

FIGURE 8. Response of two different networks to the same input (panel A). Panel B shows the response of a network which approximates a bandpass filter, whereas panel C shows the response of a network which approximates a lowpass filter. The gray shaded bars in panels B and C show the actual measured fraction of neurons which fire in pool V of the network during a time interval of length 5 ms in response to the input activity in pool U_1 shown in panel A. The solid lines in panels B and C are plots of the theoretically predicted output (see [Maass and Natschläger, 1998]).

of length Δ (e.g. for Δ in the range between 1 and 5 ms), our computational model from section 5 maps a vector of n analog time series $\{x_i(t)\}_{t\in\mathbf{N}}$ onto an output time series $\{y(t)\}_{t\in\mathbf{N}}$ (where ($\mathbf{N} = \{0, 1, 2, \dots\}$).

As an example consider a network which consists of one presynaptic pool U_1 connected to the output pool V with the same type of synapses as discussed before. In addition there are *feedback connections* between individual neurons $v \in V$. The results of simulations reported in Figure 8 show that this network computes an interesting map in the time series domain: The space-rate code in pool V represents a sigmoidal function σ applied to the output of a *bandpass filter*. Figure 8B shows the response of such network to a sine wave with some bursts of activity added on top (Figure 8A). Figure 8C shows the output of another network of spiking neurons (which approximates a *lowpass* filter) to the same input (shown in Figure 8A).

One can prove that in principle any linear filter with finite or infinite impulse response can be implemented with any desired degree of precision by recurrent networks of spiking neurons with space-rate coding [Maass and Natschläger, 1998].

7 Computing with Firing Rates

Traditionally a link between sigmoidal neural networks and biological neural systems is established by interpreting the firing rate (i.e., the spike count over time) of a spiking neuron as an analog number between 0 and 1. In this interpretation one gets a plausible correspondence between the dependence of the output value $g(\sum_{j\in\Gamma_i} w_{ij}\, x_j)$ of a sigmoidal gate on its input values x_j on one hand, and the dependence of the firing rate of a spiking neuron i on the firing rates of presynaptic neurons $j \in \Gamma_i$ on the other hand.

There exists ample biological evidence that information about a stimulus is in many biological neural systems encoded in the firing rates of neurons. However recent empirical results from neurophysiology have raised doubts whether the firing rate of a biological neuron i does in fact depend on the firing rates x_j of presynaptic neurons $j \in \Gamma_i$ in a way that can be described by an expression of the form $g(\sum_{j\in\Gamma_i} w_{ij}\, x_j)$. Results of [Abbott et al., 1997] and others about the dynamic behavior of biological synapses show that for some neural systems above a "limiting frequency" of about 10 Hz the amplitudes of postsynaptic potentials are inversely proportional to the firing rate x_j of the presynaptic neuron $j \in \Gamma_i$. These results suggest that instead of a fixed parameter w_{ij} one has to model the "strength" of a biological synapse for rate coding by a quantity $w_{ij}(x_j)$ that *depends* on the firing rate x_j of the presynaptic neuron, and that this quantity $w_{ij}(x_j)$ is proportional to $\frac{1}{x_j}$. But then the weighted sum $\sum_{j\in\Gamma_i} w_{ij}(x_i)\cdot x_j$, which models the average membrane potential at the soma of a spiking neuron i, does no longer depend on the firing rates x_j of those presynaptic neurons j that fire above the limiting frequency. We refer to [Maass and Zador, 1998] for a survey of computational implications of synaptic dynamics.

8 Computing with Firing Rates and Temporal Correlations

We will discuss in this section computations that employ a quite different type of "temporal coding". Communication by spike trains offers a direct way to encode transient *relations* between different neurons: through coincidences (or near coincidences) in their firing times. Hence computations with spiking neurons may in principle also involve complex operations on *relations between computational objects*, a computational mode which has no parallel in traditional neural network models – or any other common computational model. This type of temporal coding need not necessarily take place on the microscopic level of coding by single spikes, but can also take place on a macroscopic level of statistical correlations between firing times of different neurons. Milner had conjectured already in 1974 that visual input might be encoded in the visual cortex in a way where "cells fired by the same figure fire together but not in synchrony with cells fired by other figures" ([Milner, 1974]). This conjecture has subsequently been elaborated by von der Malsburg [von der Malsburg, 1981] and has been supported more recently by experimental data from several labs (see for example [Eckhorn et al., 1988; Gray et al., 1989; Kreiter and Singer, 1996; Vaadia et al., 1995]).

A variety of models have been proposed in order to shed light on the possible *organization* of *computing with firing correlations* in networks of spiking neurons. We have already shown in the preceding sections that spiking neurons are well-suited for *detecting* firing correlations among preceeding neurons. They also can *induce* firing correlations in other neurons k by sending the same output spike train to several other neurons k. But the question remains what exactly can be computed with firing correlations in a *network* of spiking neurons.

In [Eckhorn et al., 1990] a computational model was introduced whose computational units are modifications of integrate-and-fire neurons that receive two types of input: *feeding input* and *linking input*. Both types of inputs are processed in this model by leaky integrators with different time constants, and are then multiplied to produce the potential $u_i(t)$ of an integrate-and-fire neuron i. In networks of such computational units the feeding input typically is provided by feedforward connections, starting from the stimulus, whereas the linking input comes through feedback connections from higher layers of the network. These higher layers may represent information stored in an associative memory like in a Hopfield net. Computer simulations have shown that this model is quite successful in reproducing firing patterns in response to specific stimuli that match quite well firing patterns that have been experimentally observed in the visual cortex of cat and monkey. So far no theoretical results have been derived for this model.

A related model, but on a more abstract level without spiking neurons was proposed in [Kay and Phillips, 1996] (see also [Phillips and Singer, 1996] for a survey of this and related models). That model also involves two types of input, called RF and CF, where RF corresponds to "feeding input" and CF corresponds to "linking input". No computational model has been specified for the generation of the

CF-values. A computational unit in their model outputs a continuous value $2 \cdot g(\frac{r}{2} \cdot (1 + e^{2r \cdot c})) - 1$ that ranges between -1 and 1. In this formula r is a weighted sum of RF-input, s is a weighted sum of CF-input to that unit, and g is a sigmoidal gain function. In this computational unit the RF-input r determines the sign of the output. Furthermore $r = 0$ implies that the output has value 0, independently of the value of the CF-input c. However for $r \neq 0$ the size of the output is increased through the influence of the CF-input c if c has the same sign as r, and decreased otherwise. Computer simulations of large networks of such computational units have produced effects which are expected on the basis of psychological studies of visual perception in man [Phillips and Singer, 1996].

Other models aim directly at simulating effects of computing with firing correlations on an even more abstract conceptual level, see for example [von der Malsburg, 1981; Shastri and Ajjanagadde, 1993]. In [Shastri and Ajjanagadde, 1993] a formal calculus is developed for exploiting the possibility to compute with *relations* encoded by temporal coincidences.

No rigorous results are available so far which show that the previously described models have more computational power than conventional neural network models. In principle every function that is computed on any of the previously discussed models can also be computed by a conventional sigmoidal neural net, i.e., by an abstract model for a network of spiking neurons that encode all information in their firing rates. This follows from the simple fact that a sigmoidal neural net has the "universal approximation property", i.e., it can approximate any given continuous function. Thus the question about a possible increase in computational power through the use of firing correlations boils down to a *quantitative* rather than *qualitative* question: How much hardware, computation time, etc. can a neural network save by computing with firing correlations in addition to firing rates?

We will sketch a new model from [Maass, 1998] that provides some first results in this direction. We write $\nu(i)$ for the output of a sigmoidal gate i, which is assumed to range over $[0, 1]$. One may interpret $\nu(i)$ as the firing rate of a spiking neuron i. We now introduce for certain sets S of neurons a new variable $c(S)$, also ranging over $[0, 1]$, whose value models in an abstract way the current amount of *temporal correlation* in the firing times of the neurons $i \in S$. For example, for some time internal A of 5 msec one could demand that $c(S) = 0$ if

$$\frac{Pr[\text{ all } j \in S \text{ fire during } A]}{\Pi_{j \in S} \ \ Pr[j \text{ fires during } A]} \leq 1 \quad ,$$

and that $c(S)$ approaches 1 when this quotient approaches infinity. Thus we have $c(S) = 0$ if all neurons $j \in S$ fire stochastically independently.

One then needs computational rules that extend the standard rule

$$\nu(i) = g(\sum_{j \in \Gamma_i} w_{ij} \cdot \nu(j))$$

for sigmoidal gates so that they also involve the new variables $c(S)$ in a meaningful way. In particular, one wants to have that the firing rate $\nu(i)$ is increased if one or several subsets $S \subseteq \Gamma_i$ of preceding neurons fire with temporal correlation $c(S) > 0$. This motivates the first rule of our model:

$$\nu(i) = g(\sum_{j \in \Gamma_i} w_{ij} \cdot \nu(j) + \sum_{S \subseteq \Gamma_i} w_{iS} \cdot c(S) \cdot \Pi_{j \in S}\, \nu(j) + \vartheta) \quad . \tag{1.11}$$

The products $c(S) \cdot \Pi_{j \in S}\, \nu(j)$ in the second summand of (1.11) reflect the fact that statistical correlations in the firing times of the neurons $j \in S$ can only increase the firing rate of neuron i by a significant amount if the firing rates of all neurons $j \in S$ are sufficiently high. These products also arise naturally if $c(S)$ is interpreted as being proportional to

$$\frac{Pr[\text{ all } j \in S \text{ fire during } A]}{\Pi_{j \in S}\ Pr[j \text{ fires during } A]} \quad , \tag{1.12}$$

and $\nu(j)$ is proportional to $Pr[j$ fires during $A]$. Hence multiplying (1.12) with $\Pi_{j \in S}\, \nu(j) \approx \Pi_{j \in S}\, Pr[j$ fires during $A]$ yields a term proportional to $Pr[$all $j \in S$ fire during $A]$. This term is the one that really determines by how much the firing rate of neuron i may increase through correlated firing of neurons in S: If the neurons $j \in S$ fire almost simultaneously, this will move the state variable $u_i(t)$ of neuron i to a larger peak value compared with a situation where the neurons $j \in S$ fire in a temporally dispersed manner.

In order to complete the definition of our model for computing with firing rates $\nu(i)$ and firing correlations $c(S)$ one also has to specify how the correlation variable $c(S)$ is computed for a set S of "hidden" units i. Two effects have to be modelled:

(a) $c(S)$ increases if all neurons $i \in S$ receive common input from some other neuron k.

(b) $c(S)$ increases if there is a set S' of other neurons with significant correlation (i.e., $c(S') > 0$) so that each neuron $i \in S$ has some neuron $i' \in S'$ as predecessor (i.e., $\forall\, i \in S\ \exists\, i' \in S' (i' \in \Gamma_i)$).

These two effects give rise to the two terms in the following rule:

$$c(S) = g(\sum_{k} w_{Sk} \cdot o(k) + \sum_{S'} w_{SS'} \cdot c(S') \cdot \Pi_{i' \in S'}\, o(i') + \vartheta_S) \quad . \tag{1.13}$$

From the point of view of computational complexity it is interesting to note that in a network of spiking neurons no additional units are needed to compute the value of $c(S)$ according to (1.13). The new parameters $w_{Sk}, w_{SS'}$ can be chosen so that they encode the relevant information about the connectivity structure of the net, for example $w_{Sk} = 0$ if not $\forall\ i \in S\,(k \in \Gamma_i)$ and $w_{SS'} = 0$ if not $\forall\ i \in\ S\ \exists i' \in S'\,(i' \in \Gamma_i)$. Then the rule (1.13) models the previously described effects (a) and (b).

The rules (1.11) and (1.12) involve besides the familiar "synaptic weights" w_{ij} also new types of parameters w_{iS}, w_{Sk}, and $w_{SS'}$. The parameter w_{iS} scales the influence that correlated firing of the presynaptic neurons $j \in S$ has on the firing rate of neuron i. Thus for a biological neuron this parameter w_{iS} not only depends on the connectivity structure of the net, but also on the geometric and biochemical structure of the dendritic tree of neuron i and on the locations of the synapses from the neurons $j \in S$ on this dendritic tree. For example correlated firing of neurons $j \in S$ has a larger impact if these neurons either have synapses that are clustered together on a single subtree of the dendritic tree of i that contains voltage-gated channels ("hot spots"), or if they have synapses onto disjoint subtrees of the dendritic tree (thus avoiding sublinear summation of their EPSP's in the Hodgkin-Huxley model. Taking into account that very frequently pairs of biological neurons are not connected just by one synapse, but by multiple synapses that may lie on different branches of their dendritic tree, one sees that in the context of computing with firing correlations the "program" of the computation can be encoded through these additional parameters w_{iS} in much more subtle and richer ways than just through the "synaptic weights" w_{ij} . Corresponding remarks apply to the other new parameters w_{Sk} and $w_{SS'}$ that arise in the context of computing with firing correlations.

One should add that the interpretation of $c(S)$ becomes more difficult in case that one considers correlation variables $c(S')$ for a family of sets S' whose intersection contains more than a single neuron. For example, if $|S'| \geq 2$ and $S' \subsetneq S$ then $c(S)$ should be interpreted as the impact of correlated firing of neurons in S *beyond* the impact that correlated firing of the neurons in S' already has. One can escape this technical difficulty by considering for example in a simplified setting only correlation variables $c(S)$ for sets S of size 2.

The following result shows that a computational unit i that computes its output $\nu(i)$ according to rule (1.11) has more computational power than a sigmoidal gate (or even a small network of sigmoidal gates) that receives the same numerical variables $\nu(j), c(S)$ as input. This arises from the fact that the computational role (2.10) involves a *product* of input variables.

Consider the boolean function $F : \{0,1\}^{n+\binom{n}{2}} \rightarrow \{0,1\}$ that outputs 1 for n boolean input variables $\nu(j)$, $j \in \{1,\ldots,n\}$, and $\binom{n}{2}$ boolean input variables $c(S)$ for all subsets $S \subseteq \{1,\ldots,n\}$ of size 2 *if and only if* $c(S) = 1$ and $o(j_1) = o(j_2) = 1$ for some subset $S = \{j_1, j_2\}$. It is obvious from equation (1.11) that if one takes as gain function the Heaviside function $\mathcal{H}$, then a *single* computational unit i of the type described by equation (1.11) can compute the function F_n. On the other hand the following result shows that a substantial number of threshold gates or sigmoidal gates are needed to compute the same function F_n. Its proof can be found in [Maass, 1998].

Theorem 1.5 *The function F_n can be computed by a single neuron that carries out computations with firing rates and firing correlations according to rule (1.11).*

On the other hand any feedforward threshold circuit that computes the function F_n needs to have on the order of $n^2/\log n$ gates. Any feedforward circuit consisting of sigmoidal gates [6] *needs to have at least proportional to n many gates to compute F_n.* ∎

9 Networks of Spiking Neurons for Storing and Retrieving Information

Synfire chains [Abeles, 1991] are models for networks of spiking neurons that are well-suited for storing and retrieving information from a network of spiking neurons. A synfire chain is a chain of pools of neurons with a rich ("diverging/converging") pattern of excitatory feedforward connection from each pool to the next pool, that has a similar effect as complete connectivity between successive pools: an almost synchronous firing of most neurons in one pool in a synfire chain triggers an almost synchronous firing of most neurons in the next pool in the chain. Neurons may belong to different synfire chains, which has the consequence that the activation of one synfire chain may trigger the activation of another synfire chain (see [Bienenstock, 1995]). In this way a pointer from one memory item (implemented by one synfire chain) to another memory item (implemented by another synfire chain) can be realized by a network of spiking neurons. A remarkable property of synfire chains is that the temporal delay between the activation time of the first pool and the k-th pool in a synfire chain has a very small variance, even for large values of k. This is due to the temporal averaging of EPSP's from neurons in the preceding pool that is carried out through rich connectivity between successive pools.

An *analog* version of synfire chains results from the model for computing with space-rate coding discussed in section 5. If one takes synaptic unreliability into account for a synfire chain, one can achieve that the percentage of firing neurons of a later pool in the synfire chain becomes a smooth function of the percentage of firing neurons in the first pool. This variation of the synfire chain model predicts that precisely timed firing patterns of a fixed set of 2 or 3 neurons in different pools of a biological neural system

[6] with piecewise rational activation functions

occur more often than can be expected by chance, but not *every* time for the same stimulus. This prediction is consistent with experimental data [Abeles et al., 1993].

Other types of networks of spiking neurons that are useful for storing and retrieving information are various implementations of attractor neural networks with recurrent networks of spiking neurons, see for example [Maass and Natschläger, 1997].

10 Computing on Spike Trains

We still know very little about the power of networks of biological neurons for computations on spike trains, for example for spike trains that encode a time series of analog numbers as for example the spike trains from neuron $H1$ in the blowfly (see [Rieke et al., 1997]). One problem is that the previously discussed formal models for networks of spiking neurons are not really adequate for modeling computations by biological neural systems on *spike trains*, because they are based on the assumption that synaptic weights w_{ij} are *static* during a computation. Some consequences of the inherent temporal dynamics of biological synapses on the computational power of networks of spiking neurons are discussed in [Maass and Zador, 1998, Maass and Sontag, 1998, Zador et al., 1999]. In particular, it is shown in [Maass and Sontag, 1998] that theoretically a single layer of spiking neurons with dynamic synapses can approximate any nonlinear filter given by an arbitrary Volterra series.

11 Conclusions

The results of this chapter show that networks of spiking neurons present a quite interesting new class of computational models. In particular, they can carry out analog computation not only under a rate code, but also under temporal codes where the timing of spikes carries analog information. We have presented theoretical evidence which suggests that through the use of temporal coding a network of spiking neurons may gain for certain computational tasks more computational power than a traditional neural network of comparable size.

The models for networks of spiking neurons that we have discussed in this chapter originated in the investigation of biological neurons. However it is obvious that many of the computational ideas and architectures presented in this chapter are of a more general nature, and can just as well be applied to implementations of pulsed neural nets in electronic hardware, such as those surveyed in [Maass and Bishop, 1999].

12 References

[Abeles, 1982] Abeles, M. (1982). Role of the cortical neuron: integrator or coincidence detector? *Israel J. Med. Sci.*,18:83–92.

[Abeles, 1991] Abeles, M. (1991). *Corticonics*. Cambridge University Press, Cambridge.

[Abeles et al., 1993] Abeles, M., Bergmann, H., Margalit, E., and Vaadia, E. (1993). Spatiotemporal firing patterns in the frontal cortex of behaving monkeys. *J. of Neurophysiology*, 70(4), 1629–1638.

[Abbott et al., 1997] Abbott, L. F., Sen, K., Varela, J. A., and Nelson, S. B. (1997). Synaptic depression and cortical gain control. *Science*, 275:220–222.

[Bienenstock, 1995] Bienenstock, E. (1995). A model of neocortex. *Network*, 6:179–224.

[DasGupta and Schnitger, 1996] DasGupta, B. and Schnitger G. (1996). Analog versus discrete neural networks. *Neural Computation*, 8(4), 805–818.

[Dobrunz and Stevens, 1997] Dobrunz, L. and Stevens, C. (1997). Heterogenous release probabilities in hippocampal neurons. *Neuron*, 18:995–1008.

[Eckhorn et al., 1988] Eckhorn, R., Bauer, R., Jordan, W., Brosch, M., Kruse, W., Munk, M., and Reitboeck, H. J. (1988). Coherent oscillations: A mechanism of feature linking in the visual cortex? Multiple electrode and correlation analysis in the cat. *Biological Cybernetics*, 60:121–130.

[Eckhorn et al., 1990] Eckhorn, R., Reitboeck, H. J., Arndt, M., and Dicke, P. (1990). Feature linking via synchronization among distributed assemblies: simulations of results from cat visual cortex. *Neural Computation*, 2:293–307.

[Gerstner, 1999] Gerstner, W. (1999) Spiking Neurons. In: *Pulsed Neural Networks*, W. Maass and C. Bishop, eds., MIT Press (Cambridge), 3–53.

[Gerstner and van Hemmen, 1992] Gerstner, W. and van Hemmen, J. L. (1992). Associative memory in a network of “spiking” neurons. *Network: Computation in Neural Systems*, 3:139–164.

[Gray et al., 1989] Gray, C. M., König, P., Engel, A. K., and Singer, W. (1989). Oscillatory responses in cat visual cortex exhibit intercolumnar synchronization which reflects global stimulus properties. *Nature*, 338:334–337.

[Haefliger et al., 1997] Haefliger, P., Mahowald, M., and Watts, L. (1997). A spike based learning neuron in analog VLSI. *Advances in Neural Information Processing Systems, vol. 9*, MIT Press, (Cambridge), 692–698.

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

[Hopfield and Herz, 1995] Hopfield, J. J., Herz, A. V. M. (1995). Rapid local synchronization of action potentials: Toward computation with coupled integrate-and-fire neurons. *Proc. Natl. Acad. Sci. USA*, 92:6655–6662.

[Judd and Aihara, 1993] Judd, K. T. and Aihara, K. (1993). Pulse propagation networks: A neural network model that uses temporal coding by action potentials. *Neural Networks*, 6:203–215.

[Kay and Phillips, 1996] Kay, J. and Phillips, W. A. (1997). Activation functions, computational goals, and learning rules for local processors with contextual guidance. *Neural Computation*, 9(4):895–910.

[Kreiter and Singer, 1996] Kreiter, A. K. and Singer, W. (1996). Stimulus-dependent synchronization of neuronal responses in the visual cortex of the awake macaque monkey. *The Journal of Neuroscience*, 16(7):2381–2396.

[Kistler et al., 1997] Kistler, W., Gerstner, W., and van Hemmen, J. L. (1997). Reduction of Hodkin-Huxley equations to a single-variable threshold model. *Neural Computation*, 9:1015–1045.

[Maass, 1996] Maass, W. (1996). Lower bounds for the computational power of networks of spiking neurons. *Neural Computation*, 8(1):1–40.

[Maass, 1997a] Maass, W. (1997). Fast sigmoidal networks via spiking neurons. *Neural Computation*, 9:279–304.

[Maass, 1997b] Maass, W. (1997). Networks of spiking neurons: the third generation of neural network models. *Neural Networks*, 10(9):1659–1671. Extended abstract (with a different title) appeared in: *Advances in Neural Information Processing Systems, vol. 9*, MIT Press, (Cambridge), 211–217.

[Maass, 1998] Maass, W. (1998). A simple model for neural computation with firing rates and firing correlations. *Network: Computation in Neural Systems*, 9:1–17.

[Maass et al., 1991] Maass, W., Schnitger, G., and Sontag, E. (1991). On the computational power of sigmoid versus boolean threshold circuits. Proc. of the *32nd Annual IEEE Symposium on Foundations of Computer Science* 1991, 767–776; extended version appeared in: *Theoretical Advances in Neural Computation and Learning*, V. P. Roychowdhury, K. Y. Siu, A. Orlitsky, eds., Kluwer Academic Publishers (Boston, 1994), 127–151.

[Maass and Bishop, 1999] Maass, W., and Bishop, C., eds. (1999) *Pulsed Neural Networks*, MIT Press, (Cambridge).

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

[Maass and Natschläger, 1998] Maass, W. and Natschläger, T. (1998). A model for fast analog computation based on unreliable synapses. *Neural Computation*, in press.

[Maass and Schmitt, 1997] Maass, W. and Schmitt, M. (1997). Complexity of learning for networks of spiking neurons, *Proc. of the Tenth Annual Conference on Computational Learning Theory*, ACM, New York, 54–61; journal version to appear in *Information and Computation*.

[Maass and Sontag, 1998] Maass, W. and Sontag, E. D. (1998). Neural systems as nonlinear filters, submitted for publication.

[Maass and Zador, 1998] Maass, W. and Zador, A. (1998). *Dynamic stochastic synapses as computational units*, Advances in Neural Information Processing Systems, vol. 10, MIT-Press (Cambridge), 194–200.

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

[Markram and Sakmann, 1995] Markram, H. and Sakmann, B. (1995). Action potentials propagating back into dendrites triggers changes in efficacy of single-axon synapses between layer V pyramidal neurons. *Society for Neuroscience Abstracts*, 21:2007.

[Milner, 1974] Milner, P. M. (1974). A model for visual shape recognition. *Psychological Review*, 81(6):521–535.

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

[Petrov, 1995] Petrov, V. V. (1995). *Limit Theorems of Probability Theory.* Oxford University Press.

[Phillips and Singer, 1996] Phillips, W. A. and Singer, W. (1996). In search of common foundations for cortical computation. *Behavioral and Brain Sciences*, in press.

[Rieke et al., 1997] Rieke, F., Warland, D., de Ruyter van Steveninck, R. R., and Bialek, W. (1997). *Spikes – Exploring the Neural Code*. MIT Press, (Cambridge), MA.

[Ruf and Schmitt, 1998] Ruf, B., and Schmitt, M. (1998). Self-organizing maps of spiking neurons using temporal coding. In *Computational Neuroscience: Trends in Research*, J. M. Bower, ed., Plenum Press, New York, 509–514,.

[Samuelides et al., 1997] Samuelides, M. and Thorpe, S., Veneau, E. (1997). Implementing hebbian learning in a rank-based neural network. *Proc. 7th Int. Conference on Artificial Neural Networks - ICANN'97* in Lausanne, Switzerland, Springer, Berlin, 145–150.

[Shastri and Ajjanagadde, 1993] Shastri, L. and Ajjanagadde, V. (1993). From simple associations to systematic reasoning: a connectionist representation of rules, variables and dynamic bindings using temporal synchrony. *Behavioural and Brain Sciences*, 16:417–494.

[Sontag, 1997] Sontag, E. D. (1997). Shattering all sets of 'k' points in "general position" requires $(k-1)/2$ parameters. *Neural Computation*, 9(2):337–348.

[Thorpe and Imbert, 1989] Thorpe, S. J. and Imbert, M. (1989). Biological constraints on connectionist models. *Connectionism in Perspective*. R. Pfeifer, Z. Schreter, F. Fogelman-Soulié, L. Steels, eds. Elsevier, Amsterdam, (1989), pp. 63–92.

[Thorpe and Gautrais, 1997] Thorpe, S. J. and Gautrais, J. (1997). Rapid visual processing using spike asynchrony. *Advances in Neural Information Processing Systems, vol. 9*, MIT Press, (Cambridge), MA, 901–907.

[Vaadia et al., 1995] Vaadia, E., Aertsen, A., and Nelken, I. (1995). Dynamics of neuronal interactions cannot be explained by neuronal transients. *Proc. Royal Soc. of London B*, 261:407–410.

[Valiant, 1994] Valiant, L. G. (1994). *Circuits of the Mind*, Oxford University Press, Oxford.

[von der Malsburg, 1981] von der Malsburg, C. (1981). The correlation theory of brain function. *Internal Report 81-2 of the Dept. of Neurobiology of the Max Planck Institute for Biophysical Chemistry in Göttingen*, Germany. Reprinted in *Models of Neural Networks II*, Domany et al., eds., Springer, 1994, 95–119.

[Watanabe and Aihara, 1997] Watanabe, M., and Aihara, K. (1997). Chaos in neural networks composed of coincidence detector neurons. *Neural Networks*, 10(8), 1353-1359.

[Wilson and Cowan, 1972] Wilson, H. R. and Cowan, J. D. (1972). Excitatory and inhibitory interactions in localized populations of model neurons. *Biophysics Journal*, 12:1–24.

[Zador et al., 1999] Zador, A., Natschläger, T., and Maass, W., (1999). Temporal processing in networks coupled by dynamic synapses, in preparation.