On the Computational Complexity of Networks of Spiking Neurons

(Extended Abstract)

Wolfgang Maass

Institute for Theoretical Computer Science
Technische Universitaet Graz
Klosterwiesgasse 32/2
A-8010 Graz, Austria
e-mail: maass@igi.tu-graz.ac.at

Abstract

We investigate the computational power of a formal model for networks of spiking neurons. It is shown that simple operations on phase-differences between spike-trains provide a very powerful computational tool that can in principle be used to carry out highly complex computations on a small network of spiking neurons. We construct networks of spiking neurons that simulate arbitrary threshold circuits, Turing machines, and a certain type of random access machines with real valued inputs. We also show that relatively weak basic assumptions about the response- and threshold-functions of the spiking neurons are sufficient in order to employ them for such computations.

Furthermore we prove upper bounds for the computational power of networks of spiking neurons with arbitrary piecewise linear response- and threshold-functions, and show that they are with regard to real-time simulations computationally equivalent to a certain type of random access machine, and to recurrent analog neural nets with piecewise linear activation functions. In addition we give corresponding results for networks of spiking neurons with a limited timing precision, and we prove upper and lower bounds for the VC-dimension and pseudo-dimension of networks of spiking neurons.

1 Introduction and Basic Definitions

There exists substantial evidence that timing phenomena such as temporal differences between spikes and frequencies of oscillating subsystems are integral parts of various information processing mechanisms in biological neural systems (for a survey and references see e.g. Kandel et al., 1991; Abeles, 1991; Churchland and Sejnowski, 1992; Aertsen, 1993). Furthermore simulations of a variety of specific mathematical models for networks of spiking neurons have shown that temporal coding offers interesting possibilities for solving classical benchmark-problems such as associative memory, binding, and pattern segmentation (for an overview see Gerstner et al., 1992). Very recently one has also started to build *artificial* neural nets that model networks of spiking neurons (see e.g. Watts, 1994). Some aspects of these models have also been studied analytically (see e.g. Gerstner and van Hemmen, 1994), but almost nothing is known about their computational complexity (see Judd and Aihara, 1993, for some first results in this direction). In this article we investigate a simple formal model SNN for networks of spiking neurons that allows us to model the most important timing phenomena of neural nets (including synaptic modulation at various time-scales), and we prove lower and upper bounds for its computational power.

Definition of a Spiking Neuron Network (SNN): *An SNN $\mathcal{N}$ consists of*

- *a finite directed graph $\langle V, E\rangle$ (we refer to the elements of V as "neurons" and to the elements of E as "synapses")*
- *a subset $V_{in} \subseteq V$ of input neurons*
- *a subset $V_{out} \subseteq V$ of output neurons*
- *for each neuron $v \in V - V_{in}$ a threshold-function $\Theta_v : \boldsymbol{R}^+ \to \boldsymbol{R} \cup \{\infty\}$ (where $\boldsymbol{R}^+ := \{x \in \boldsymbol{R} : x \geq 0\}$)*
- *for each synapse $\langle u, v\rangle \in E$ a response-function $\varepsilon_{u,v} : \boldsymbol{R}^+ \to \boldsymbol{R}$ and a weight-function $w_{u,v} : \boldsymbol{R}^+ \to \boldsymbol{R}$.*

We assume that the firing of the input neurons $v \in V_{in}$ is determined from outside of $\mathcal{N}$, i.e. the sets $F_v \subseteq \boldsymbol{R}^+$ of firing times ("spike trains") for the neurons $v \in V_{in}$ are given as the input of $\mathcal{N}$. Furthermore we assume that a set $T \subseteq \boldsymbol{R}^+$ of potential firing times has been fixed (we will consider only the cases $T = \boldsymbol{R}^+$ and $T = \{i \cdot \mu \,:\, i \in \boldsymbol{N}\}$ for some $\mu > 0$).

For a neuron $v \in V - V_{in}$ one defines its set F_v of firing times recursively. The first element of F_v is $\inf\{t \in T : P_v(t) \geq \Theta_v(0)\}$, and for any $s \in F_v$ the next larger element of F_v is $\inf\{t \in T : t > s \text{ and } P_v(t) \geq \Theta_v(t-s)\}$, where the *potential function* $P_v : \boldsymbol{R}^+ \to \boldsymbol{R}$ *is defined by*

$$P_v(t) := 0 + \sum_{u\,:\,\langle u,v\rangle \in E} \;\; \sum_{s \in F_u\,:\,s<t} w_{u,v}(s) \cdot \varepsilon_{u,v}(t-s) \ .$$

The firing times ("spike trains") F_v of the output neurons $v \in V_{out}$ that result in this way are interpreted as the output of $\mathcal{N}$.

Regarding the set T of potential firing times we consider in this article the case $T = \mathbf{R}^+$ (SNN with continuous time) and the case $T = \{i \cdot \mu : i \in \mathbf{N}\}$ for some μ with $1/\mu \in \mathbf{N}$ (SNN with discrete time).

We assume that for each SNN $\mathcal{N}$ there exists a bound $\tau_{\mathcal{N}} \in \mathbf{R}$ with $\tau_{\mathcal{N}} > 0$ such that $\Theta_v(x) = \infty$ for all $x \in (0, \tau_{\mathcal{N}})$ and all $v \in V - V_{in}$ ($\tau_{\mathcal{N}}$ may be interpreted as the minimum of all "refractory periods" τ_{ref} of neurons in $\mathcal{N}$). Furthermore we assume that all "input spike trains" F_v with $v \in V_{in}$ satisfy $|F_v \cap [0,t]| < \infty$ for all $t \in \mathbf{R}^+$. On the basis of these assumptions one can also in the continuous case easily show that the firing times are well-defined for all $v \in V - V_{in}$ (and occur in distances of at least $\tau_{\mathcal{N}}$).

In models for *biological neural systems* one assumes that if x time-units have passed since its last firing, the current threshold $\Theta_v(x)$ of a neuron v is "infinite" for $x < \tau_{ref}$ (where τ_{ref} = refractory period of neuron v), and then approaches quite rapidly from above some constant value. A neuron v "fires" (i.e. it sends an "action potential" or "spike" along its axon) when its current membrane potential $P_v(t)$ at the axon hillock exceeds its current threshold Θ_v. $P_v(t)$ is the sum of various postsynaptic potentials $w_{u,v} \cdot \varepsilon_{u,v}(t-s)$. Each of these terms describes an *excitatory* (EPSP) or *inhibitory* (IPSP) *postsynaptic potential* at the axon hillock of neuron v at time t, as a result of a spike that had been generated by a "presynaptic" neuron u at time s, and which has been transmitted through a synapse between both neurons. Recordings of an EPSP typically show a function that has a constant value c (c = resting membrane potential; e.g. $c = -70mV$) for some initial time-interval (reflecting the axonal and synaptic transmission time), then rises to a peak-value, and finally drops back to the same constant value c. An IPSP tends to have the negative shape of an EPSP. For the sake of mathematical simplicity we assume in the SNN-model that the constant initial and final value of all response-functions $\varepsilon_{u,v}$ is equal to 0 (in other words: $\varepsilon_{u,v}$ models the *difference* between a postsynaptic potential and the resting membrane potential c). Different presynaptic neurons u generate postsynaptic potentials of different sizes at the axon hillock of a neuron v, depending on the size, location and current state of the synapse (or synapses) between u and v. This effect is modelled by the weight-factors $w_{u,v}(s)$.

The precise shapes of threshold-, response-, and weight-functions vary among different biological neural systems, and even within the same system. Fortunately one can prove significant *upper bounds* for the computational complexity of SNN's $\mathcal{N}$ without *any* assumptions about the *specific shapes* of these functions of $\mathcal{N}$. We show that for such upper bounds one only has to assume that they are of a reasonably simple *mathematical structure.*

In order to prove *lower bounds* for the computational complexity of an SNN $\mathcal{N}$ one is forced to make more specific assumptions about these functions. However we show that significant (and in some cases *optimal*, see Theorem 3) lower bounds can be shown under some rather weak *basic assumptions* about these functions, which are further relaxed in section 4 of Maass, 1994c. These basic assumptions (see section 2) mainly require that EPSP's have an arbitrarily small time-segment where they increase linearly, and some arbitrarily small time-segment where they decrease linearly. Since the computational power of SNN's may potentially increase through the use of time-dependent weights, *lower* bounds for their computational power are more significant if they do *not* involve the use of time-dependent weights. Hence we will assume in our *lower* bound results that *all weight-functions* $w_{u,v}(s)$ *have a constant value* $w_{u,v}$ *which does not depend on the time* s.

Apart from the abovementioned condition on the existence of linear segments in EPSP's, the basic assumptions which underlie our lower bound results involve no other significant conditions on the shape of response- and threshold-functions. Hence one may argue that these basic assumptions are biologically plausible. In addition the same lower bounds can be shown if also phenomena such as "adaptation" of neurons, or a "reset" of the potential after a firing are taken into account (see Maass, 1994c). Thus the more critical points with regard to the biological interpretation of these lower bound results appear to be the relatively simple firing mechanism of the SNN-model, which for example ignores for the sake of simplicity nonlinear interactions among postsynaptic potentials such as integration of potentials within the dendritic tree of a neuron, and various possible sources of "imprecision" in the determination of the firing times. The latter issue can partially be taken into account by considering the variation of the SNN-model with *discrete* firing times as in section 3 and 4 (although the implicit global synchronization of this version is not completely satisfactory). In this variation of the SNN-model with discrete firing times $i \cdot \mu$ for $i \in \mathbf{N}$ one can view a firing of a neuron at time $i \cdot \mu$ as representing a somewhat imprecise firing time in a small interval *around* time $i \cdot \mu$.

The model SNN that we consider in this article is very closely related to the model that was previously considered by Buhmann and Schulten, 1986, and especially to the *spike response model* of Gerstner, 1991, Gerstner, Ritz, van Hemmen, 1992, and Gerstner, van Hemmen, 1994. Similarly as in Buhmann and Schulten, 1986, we consider in this article only the deterministic case (which corresponds to the limit case $\beta \to \infty$ in the stochastic spike response model of Gerstner et al.). However in contrast to these preceding models we do not fix particular (necessarily somewhat arbitrarily chosen) response- and threshold-functions in our model SNN. Instead, we want to have the possibility to use the SNN-model as a framework for *investigating* the computational power of various *different* response- and threshold-functions. In addition, we would like to make sure that various *different* response- and threshold-functions that are observed in specific biological neural systems are in fact special cases of the response- and threshold-functions in the here considered formal model SNN.

The model SNN is also suitable for investigating algorithms that involve *synaptic modulation* at various time-scales. Hence one can investigate within this framework not only the complexity of algorithms for supervised and unsupervised learning, but also the potential power of rapid weight-changes *within* the course of a computation (concrete algorithms of this type were proposed by von der Malsburg and Schneider, 1986; Sporns et al., 1991; and many others). In the *upper* bound results of this paper we allow that the value of a weight $w_{u,v}(s)$ at a firing time $s \in F_u$ is defined by an *algebraic computation tree* (see van Leeuwen, 1990) in terms of its previous value at firing times $s' \in F_u$ with $s' < s$, the time-difference $s - \tilde{s}$ to some preceding firing times $\tilde{s} < s$ of arbitrary other neurons, and arbitrary real-valued parameters. Such algebraic computation tree may compare the size of any two reals (respectively the value of any two previously computed terms) at its branching nodes, and add, subtract, multiply or divide any two reals (respectively previously computed terms) at its other nodes. In this way $w_{u,v}(s)$ can be defined by arbitrary piecewise rational functions of the abovementioned arguments where the domain of each rational piece of that function is characterized by algebraic inequalities in the considered arguments. As a simple special case one can for example increase $w_{u,v}$ (perhaps up to some specified saturation-value) as long as neurons u and v fire coherently, and decrease $w_{u,v}$ otherwise.

For the sake of simplicity in the statements of our results we assume in this extended abstract that for each weight $w_{u,v}$ the algebraic computation tree that describes the changes of this weight has size $O(1)$, and only involves reals of bounded absolute value. Furthermore we assume in Theorems 4, 5 and 6 that either each weight is an arbitrary time-invariant real, or that each current weight is rounded off to bit-length poly$(\log p_{\mathcal{N}})$ in binary representation, and does not depend on the firing times of firings that occured longer than time $O(1)$ ago. In addition we assume in Theorems 4 and 6 that the parameters in the algebraic computation tree are rationals of bit-length $O(\log p_{\mathcal{N}})$.

It is well-known that the *Vapnik-Chervonenkis dimension ("VC-dimension")* of a neural net $\mathcal{N}$ (and the *pseudo-dimension* for the case of a neural net $\mathcal{N}$ with *real-valued* output) can be used to bound the number of examples that are needed to train $\mathcal{N}$ (see Haussler, 1992, Maass, 1995). Obviously these notions have to be *defined differently* for a network with *time-dependent* weights.

We propose to define the VC-dimension (pseudo-dimension) of a SNN $\mathcal{N}$ with time-dependent weights as the VC-dimension (pseudo-dimension) of the class of all functions that can be computed by $\mathcal{N}$ with different assignments of values to the real-valued (or rational-valued) parameters of $\mathcal{N}$ that are involved in the definitions of the piecewise rational response-, threshold-, and weight-functions of $\mathcal{N}$. In a biological neural system $\mathcal{N}$ these parameters might for example reflect the concentrations of certain chemical substances that are known to modulate the behavior of $\mathcal{N}$.

The computational complexity of another neural network model where timing plays an important role has previously been investigated by Judd and Aihara, 1993. Their model PPN is also motivated by biological spiking neurons, but it employs a completely different firing mechanism. There are no response-functions in this model, and instead of integrating all incoming EPSP's and IPSP's in order to determine whether it should "fire", a neuron in a PPN randomly selects a *single* one of the incoming "stimulations" of maximal size, and determines on the basis of that stimulation whether it should fire. Consequently, computations in this model PPN proceed quite differently from computations in models of spiking neurons such as the spike response model of Gerstner and van Hemmen, 1994, or the here considered model SNN. Judd and Aihara, 1993, construct PPN's which can simulate Turing machines that use at most a *constant* number s of cells on their tapes, where s is bounded by the number of neurons in the simulating PPN. However a Turing machine with a constant bound s on its number of tape cells is just a special case of a finite automaton, and hence this result does not show that a PPN of finite size can have the computational power of an arbitrary Turing machine.

In contrast to the quoted result about PPN's, it is shown in Theorem 1 of this article that with arbitrary response- and threshold-functions which satisfy the basic assumptions of section 2 one can construct for any given Turing machine M an SNN $\mathcal{N}_M$ of *finite size* that can simulate *any* computation of M in real-time (even if the number of tape cells that M uses is much larger than the number of neurons in $\mathcal{N}_M$).

The focus in the investigation of computations in biological neural systems differs in two essential aspects from that of classical computational complexity theory. First, one is not only interested in single computations of a neural net for unrelated inputs x, but also in its ability to process an interrelated sequence $(\langle x(i), y(i)\rangle)_{i\in\mathbf{N}}$ of inputs and outputs, which may for example be the protocol of some *adaption-* or *learning process*. Obviously the processing of arbitrary sequences $(\langle x(i), y(i)\rangle)_{i\in\mathbf{N}}$ of inputs $x(i)$ and outputs $y(i)$ contains as a special case not only the scenario of *unsupervised* learning (where the $x(i)$ are the inputs for the unsupervised learning), but also that of *supervised* learning processes. For the case of supervised learning (e.g. PAC-learning) one may assume that for some $m \in \mathbf{N}$ the initial segment $\langle x(1), \ldots, x(m)\rangle$ represents some *training sequence* of length m, where each $x(i)$ also provides the target-output, whereas $x(i)$ for $i > m$ just represents a *test-example* which does not indicate the corresponding target output. Apart from the advantage that it allows us to compare not only the computational complexity of *computations* but also of *learning processes*, the subsequent notions of a real-time computation and real-time simulation are also therefore particularly suitable for the analysis of computations in biological neural systems because the *exact timing* of computations is all-important in biology, and many tasks have to be solved within a specific number of steps. We will show after these definitions, that these notions also allow us to analyze computations in the usual sense of computational complexity theory.

Definition of real-time computation and real-time simulation

Fix some arbitrary (finite or infinite) input alphabet A_{in} and output-alphabet A_{out} (for example they can be chosen to be $\{0,1\}$, $\{0,1\}^*$ or $\boldsymbol{R}$). We say that a machine M processes a sequence $(\langle x(i), y(i)\rangle)_{i\in \boldsymbol{N}}$ of pairs $\langle x(i), y(i)\rangle \in A_{in} \times A_{out}$ in real-time r, if M outputs $y(i)$ for every $i \in \boldsymbol{N}$ within r computation steps after having received input $x(i)$ (for $i > 0$ we assume that $x(i)$ is presented at the next step after M has given output $y(i-1)$).

We say that a machine M' simulates a machine M in real-time (with delay-factor Δ) if for every $r \in \boldsymbol{N}$ and every sequence that is processed by M in real-time r, M' can process the same sequence in real time $\Delta \cdot r$.

In the case of SNN's M we count each spike in M as a computation step.

We first would like to point out that these notions contain the usual notions of a computation respectively simulation as special cases. If M computes a boolean function $F : \{0,1\}^* \to \{0,1\}$ in time $t(n)$ (in the usual sense of computational complexity theory), then one can identify each input $\langle z_1, \ldots, z_n\rangle \in \{0,1\}^*$ with an infinite sequence $(x(i))_{i\in\mathbf{N}}$ where $x(i) = z_i$ for $i \leq n$ and $x(i) = B$ for $i > n$ (assume that M gets one input bit per step, $B :=$ "blank"). Furthermore one can set $y(i) = B$ for those steps i where M's computation is not yet finished, and $y(i) = F(\langle z_1, \ldots, z_n\rangle)$ for all later i (in particular for all $i \geq t(n)$). Obviously M processes this sequence $(\langle x(i), y(i)\rangle)_{i\in\mathbf{N}}$ in real-time 1. Hence, if another machine M' can simulate M in real-time with delay-factor Δ, then M' can compute the same function $F : \{0,1\}^* \to \{0,1\}$ in time $\Delta \cdot t(n)$. This implies that a real-time simulation is a special case of a linear-time simulation. In particular, every computational problem that can be solved by M within a certain time complexity, can be solved by M' within the same time complexity (up to a constant factor).

In addition, the remarks before the definition imply that when we show that M' can simulate M in *real-time*, we may conclude that any *adaptive behavior* (or *learning* algorithm) of M can also be implemented on M'. Finally we would like to point out that for the investigation of specific computational- and learning-problems on specific models for biological neural nets one would like to get eventually also estimates for the *size* of the constant r in real-time processing, respectively the *size* of the delay-factor Δ in a real-time simulation. Such refined analysis (which will not be carried out in this paper) appears to be also of interest, since it is likely to throw some light on the specific advantages and disadvantages of different models for biological neural systems (e.g. networks of spiking neurons versus analog neural nets).

In contrast to the usual notion of a simulation, a *real-time* simulation of another computational model M by an SNN implies that the simulation of each computation step of M requires only a *fixed* number of spikes in the SNN. In particular the required number of spikes does not become larger for the simulation of *later* computation steps of M.

Input- and Output-Conventions: For simulations between SNN's and Turing machines we assume that the SNN either gets an input (or produces an output) from $\{0,1\}^*$ in the form of a spike-train (i.e. one bit per unit of time), or encoded into the phase-difference of just two spikes. The former convention is suitable for comparisons with Turing machines that receive at each computation step a single input bit and produce a single output bit. For comparisons with Turing machines that start with the whole input written on a specified tape, and have their whole output written on another tape when the machine halts, it is more adequate to assume that the SNN receives at the beginning of a computation the whole tape content of the input tape encoded into the time-difference φ between two spikes (using the same encoding as we will use in the proof of Theorem 1 in order to represent the content of a stack), and that the SNN also provides the final content of the output tape in the same form. *Real-valued* input or output for an SNN is always encoded into the phase-difference of two spikes.

This extended abstract is a slightly expanded version of Maass, 1994b. Further details can be found in Maass, 1994c.

2 Networks of Spiking Neurons with Continuous Time

In order to carry out computations on an SNN, *some* assumptions have to be made about the structure of the response- and threshold-functions of its neurons. It is obvious that for example neurons with everywhere constant response-functions cannot carry out *any* computation. We will specify in the following a set of **basic assumptions**, which suffice for the proofs of lower bounds for the computational power or VC-dimension in this article. Some variations of these conditions are discussed in Maass, 1994c.

We assume in our constructions of SNN's for the proofs of lower bounds that there exist some arbitrary given constants $\Delta_{\min}, \Delta_{\max} \in \mathbf{R}$ with $0 \le \Delta_{\min} < \Delta_{\max}$ so that we can choose for each "synapse" $\langle u, v\rangle \in E$ an individual "*delay*" $\Delta_{u,v} \in [\Delta_{\min}, \Delta_{\max}]$ with $\varepsilon_{u,v}(x) = 0$ for all $x \in [0, \Delta_{u,v}]$. This parameter $\Delta_{u,v}$ corresponds in biology to the time-span between the firing of the presynaptic neuron u and the moment when its effect reaches the trigger zone (axon hillock) of the postsynaptic neuron v. This time-span is known to vary for individual neurons in biological neural systems, depending on the type of synapse and the geometrical constellation. The constants $\Delta_{\min}$ and $\Delta_{\max}$ can be interpreted as biological constraints on the possible lengths of such time-spans. No requirements about $\Delta_{\min}$ and $\Delta_{\max}$ are needed for our construction, except that $\Delta_{\min} < \Delta_{\max}$.

We also assume for our proofs of lower bounds that except for their individual delays $\Delta_{u,v}$ the response-functions $\varepsilon_{u,v}$ (as well as the threshold functions Θ_v) are *stereotyped*, i.e. that their shape is determined by some general functions $\varepsilon^E, \varepsilon^I$ and Θ which do not depend on u or v. More precisely, we assume that we can decide for any pair $\langle u, v\rangle \in E$ whether $\varepsilon_{u,v}$ should represent an excitatory "*EPSP-response-function*", or an inhibitory "*IPSP-response-function*". In the EPSP-case we assume that

$$\varepsilon_{u,v}(\Delta_{u,v} + x) = \varepsilon^E(x) \quad \text{for all } x \in \mathbf{R}^+,$$

and in the IPSP-case we assume that

$$\varepsilon_{u,v}(\Delta_{u,v} + x) = \varepsilon^I(x) \quad \text{for all } x \in \mathbf{R}^+.$$

In either case we assume that

$$\varepsilon_{u,v}(x) = 0 \quad \text{for all } x \in [0, \Delta_{u,v}].$$

Furthermore we assume for all neurons $v \in V - V_{in}$ that

$$\Theta_v(x) = \Theta(x) \quad \text{for all } x \in \mathbf{R}^+.$$

Finally we assume for the proofs of our lower bounds for the computational complexity of SNN's that the three functions $\varepsilon^E : \mathbf{R}^+ \to \mathbf{R}^+$, $\varepsilon^I : \mathbf{R}^+ \to \{x \in \mathbf{R} : x \leq 0\}$, and $\Theta : \mathbf{R}^+ \to \mathbf{R}^+ \cup \{\infty\}$ are some *arbitrary* functions with the following properties: There exist some arbitrary strictly positive real numbers τ_{ref}, τ_{end}, σ_1, σ_2, σ_3, τ_1, τ_2, τ_3, L, s_{up}, s_{down} with $0 < \tau_{ref} < \tau_{end}$, $\sigma_1 < \sigma_2 < \sigma_3$, $\tau_1 < \tau_2 < \tau_3$ which satisfy the following five conditions:

(1) $\Theta(x) \geq \Theta(0) > 0$ *for all* $x \in \boldsymbol{R}^+$, $\Theta(x) = \infty$ *for all* $x \in (0, \tau_{ref})$, *and* $\Theta(x) = \Theta(0) < \infty$ *for all* $x \in [\tau_{end}, \infty)$

(2) $\varepsilon^E(0) = \varepsilon^E(x) = 0$ *for all* $x \in [\sigma_3, \infty)$, *and there exists some* $\varepsilon_{\max} \in \boldsymbol{R}^+$ *so that* $\exists x \in \boldsymbol{R}^+(\varepsilon^E(x) = \varepsilon_{\max})$ *and* $\forall\, y \in \boldsymbol{R}^+(\varepsilon^E(y) \leq \varepsilon_{\max})$

(3) $\varepsilon^E(\sigma_1 + z) = \varepsilon^E(\sigma_1) + s_{up} \cdot z$ *for all* $z \in [-L, L]$

(4) $\varepsilon^E(\sigma_2 + z) = \varepsilon^E(\sigma_2) - s_{down} \cdot z$ *for all* $z \in [-L, L]$

(5) $\varepsilon^I(0) = \varepsilon^I(x) = 0$ *for all* $x \in [\tau_3, \infty)$, $\varepsilon^I(x) < 0$ *for all* $x \in (0, \tau_3)$, ε^I *is non-increasing in* $[0, \tau_1]$ *and non-decreasing in* $[\tau_2, \tau_3]$.

We assume in addition that $\Theta(0)$, σ_1, σ_2, $\varepsilon^E(\sigma_1)$, $\varepsilon^E(\sigma_2)$, s_{up}, $s_{down} \in \boldsymbol{Q}$.

It should be pointed out that no conditions about the smoothness, the continuity, or the number of extrema of the functions Θ, ε^E, ε^I are made in the preceding basic assumptions. However if one demands in addition that ε^E is piecewise linear and continuous, then the conditions (3) and (4) become redundant. The assumption that $\Theta(0), \sigma_1$, σ_2, $\varepsilon^E(\sigma_1)$, $\varepsilon^E(\sigma_2)$, s_{up}, s_{down} are rationals will only be needed to ensure that certain weights and delays can be chosen to be *rationals*.

There exist of course extremely simple response- and threshold-functions which satisfy our basic assumptions, and our subsequent results demonstrate that these can in principle be used in order to build an artificial neural network with some finite number $n_{\mathcal{U}}$ of spiking neurons that can simulate in real time any other digital computer (even computers that employ many more than $n_{\mathcal{U}}$ memory cells or computational units).

We have formulated the preceding basic assumptions on the response- and threshold-functions in a rather general fashion in order to make sure that they can in principle be satisfied by a wide range of EPSP's, IPSP's and threshold-functions that have been observed in a number of biological neural systems.

The currently available findings about biological neural systems (see e.g. Kandel et al., 1991, and the discussions in Valiant, 1994) indicate that in general a single EPSP alone cannot cause a neuron to fire. In fact, it is commonly reported that 50 to 100 EPSP have to arrive within a short time-span at a neuron in order to trigger its firing. These reports indicate that the weights $w_{u,v}$ in our model should be assumed to be relatively small, since they cannot amplify a single EPSP to yield an arbitrarily high potential P_v. Hence for the sake of biological plausibility one should assume for all lower bound results that the values of all weights $w_{u,v}$ in an SNN belong to some bounded interval $[0, w_{\max}]$. For simplicity we assume in the following lower bound results that $w_{\max} = 1$. This convention just amounts to a certain scaling of the values of the response-functions in relation to the threshold-functions. In any version of this model where a single neuron is not able to cause the firing of another neuron, one necessarily has to assume that each input spike is simultaneously received by *several* neurons (since otherwise it cannot have any effect).

Theorem 1: *If the response- and threshold-functions of the neurons satisfy the previously described basic assumptions, then one can build from such neurons for any given* $d \in \boldsymbol{N}$ *an SNN* $\mathcal{N}_{TM}(d)$ *of finite size with rational delays that can simulate with a suitable assignment of rational values from* $[0,1]$ *to its weights any Turing machine with at most* d *tapes in real-time.*

Furthermore $\mathcal{N}_{TM}(2)$ *can compute* *<u>any</u> function* $F : \{0,1\}^* \to \{0,1\}^*$ *with a suitable assignment of real values from* $[0,1]$ *to its weights.*

The constructions in the proof of Theorem 1 employ an "oscillator" as "pacemaker", and $O(1)$ other "oscillators" with the same oscillation-frequency, whose phase-difference $\sum_{i=1}^{\ell} b_i \cdot 2^{-i-c}$ to the pacemaker encodes the current content $\langle b_1, \ldots, b_\ell \rangle \in \{0,1\}^*$ of a stack (for arbitrary $\ell \in \mathbf{N}$). In order to simulate the usual stack operations one has to build SNN-modules that can multiply such a phase-difference with the constants $1/2$, respectively 2. In order to simulate the control of a Turing machine one has to build a synchronization module that can bring arbitrary spikes into synchronization with the pacemaker. Such module is needed, since the exact *firing times* of neurons that simulate boolean gates in a straightforward manner are in general dependent on the *values* of its inputs, and they may even vary for different inputs that should yield the same output. This causes problems if one wants to simulate the gates on higher layers of a multi-layer boolean circuit, since their "input-spikes" will in general not arrive simultaneously. However with the help of synchronization modules an SNN can even simulate *threshold circuits* in an efficient manner.

The last part of Theorem 1 implies that the VC-dimension of some finite SNN's is infinite. In contrast to that the following result shows that one can give finite bounds for the VC-dimension of those SNN's that only use a bounded numbers of spikes in their computation. Furthermore the last part of the claim of Theorem 2 implies that their VC-dimension may in fact grow with the number S of spikes that occur in a computation.

Theorem 2: *The VC-dimension and pseudo-dimension of any SNN $\mathcal{N}$ with piecewise linear response- and threshold-functions, arbitrary real-valued parameters and time-dependent weights (as specified in section 1) can be bounded (even for real-valued inputs and outputs) by $O(|E| \cdot |V| \cdot S(\log|V| + \log S))$ if $\mathcal{N}$ uses in each computation at most S spikes.*

Furthermore one can construct SNN's (with any response- and threshold-functions that satisfy our basic assumptions, with fixed rational parameters and rational time-invariant weights) whose VC-dimension is for computations with up to S spikes as large as $\Omega(|E| \cdot S)$.

We have shown in Theorem 1 that one can build from arbitrary neurons, whose response- and threshold-functions satisfy certain basic assumptions, an SNN that can simulate any Turing machine. However SNN's are strictly more powerful than Turing machines for two reasons:

i) An SNN can receive *real* numbers as input, and give *real* numbers as output (in the form of time-differences between pairs of spikes).

ii) One can construct from any neurons which satisfy our basic assumptions modules for an SNN that can ADD, SUBTRACT, or COMPARE any two phase-differences from $[0, L/4]$, as well as a module for MULTIPLY(β) (multiplication of a phase-difference with an arbitrary constant $\beta > 0$). If such operations are applied to a phase-difference of the form $\sum_{i=1}^{\ell} b_i \cdot 2^{-i-c}$, this will in general affect more than $O(1)$ of the stored bits $\langle b_1, \ldots, b_\ell \rangle$ (which can never happen on a Turing machine).

It turns out that one can in fact *characterize exactly* the computational power of SNN's with arbitrary piecewise linear response- and threshold-functions which satisfy our basic assumptions. For that purpose we consider arbitrary random access machines (RAM's) with $O(1)$ registers that can store in their registers, use as constants, receive as input, and give as their output arbitrary real numbers of bounded absolute value, and which employ arbitrary finite programs with the instructions ADD, SUBTRACT, COMPARE, MULTIPLY(β) for arbitrary real-valued constants β, HALT. These instructions may involve direct and indirect addressing, as well as conditional jumps. We will use the *unit-cost criterion* (i.e. one unit is charged for each execution of an instruction), and refer to the here described RAM's as N-RAM's (because of their intimate connection to neural nets, as shown in Theorem 3). Obviously this model is closely related to that of Blum, Shub and Smale (Blum et al., 1989). It is easy to see that for boolean valued input it can simulate any Turing machine in real-time (representing finite stack-contents by rational numbers as in the proof of Theorem 1).

Besides providing a tight upper bound for the computational power of a large class of SNN's, the following result also establishes a relationship between the computational power of SNN's and that of recurrent *analog* neural nets.

In the latter model no "spikes" or other non-trivial timing-phenomena occur, but the output of a gate consists of the "analog" value of some squashing- or *activation function* that is applied to the weighted sum of its inputs. This output value may be interpreted as the current firing-frequency of a neuron. See e.g. (Siegelmann and Sontag, 1992) or (Maass, 1993) for recent results about the computational power of such models.

Theorem 3: *The following three classes of computational models have the same computational power, in the sense that for any model M from one of these classes one can construct models from each of the other two classes that can simulate M in real-time:*

- *SNN's of finite size with piecewise linear response- and threshold-functions and time-invariant weights*
- *recurrent analog neural nets of finite size with piecewise linear activation functions*
- *N-RAM's.*

This equivalence holds both for the case of arbitrary real-valued parameters respectively constants in all three types of models, and if all parameters and constants are required to be rationals.

The **proof** of the preceding result shows in particular that any type of piecewise linear response- and threshold-function that satisfies our basic assumptions is *universal* for *all* piecewise linear response- and threshold-functions, in the sense that *any* SNN with *arbitrary* piecewise linear response- and threshold functions can be simulated in real-time by an SNN with response- and threshold-functions of that particular type. This proof also shows that the activation-functions π (saturated linear function) and $\mathcal{H}$ (heaviside) together are in an analogous sense *universal* for all piecewise linear activation functions for recurrent analog neural nets. For the previously described simulations between RAM's, SNN's, and analog neural nets one has to adopt suitable input- and output-conventions for real numbers. All three types of machines can input and output arbitrary real numbers of bounded absolute value. However the corresponding bound depends in the case of neural nets on their specific response- and threshold-functions (respectively their specific activation functions). Therefore in order to be able to choose these functions independently of the simulated RAM, one has to allow that the simulating neural net changes the *scale* of inputs and outputs (via multiplication with a suitable constant factor that may depend on the simulated RAM).

An interesting consequence of the proof of the preceding result is that SNN's with piecewise linear *continuous* response-functions are computationally equivalent to recurrent analog neural nets with arbitrary piecewise linear activation functions (that may be *discontinuous*). We also would like to point out that N-RAM's with the additional instruction MULTIPLY (for arbitrary real-valued operands of bounded absolute value) are with regard to real-time simulations equivalent to recurrent analog neural nets with arbitrary piecewise *polynomial* activation functions. It should be noted that Siegelmann and Sontag, 1994, and Koiran, 1993, had already established before some other relationships between analog neural nets and variations of the model by Blum et al., 1989.

Regarding other *upper* bounds for the computational power of SNN's we would like to mention also that for the case of boolean input and output any SNN with arbitrary piecewise linear response- and threshold-functions with arbitrary rational parameters and time-invariant weights can be simulated by a Turing machine in such a way that arbitrary computations of the SNN with S spikes are simulated by polynomial(S) steps on the Turing machine. Furthermore for the case of real-valued input and output SNN's with arbitrary piecewise linear response- and threshold-functions and arbitrary real-valued parameters and *time-dependent* weights (as described in section 1, with weight-changes described by algebraic computation trees of size $O(1)$ that involve only real numbers of bounded absolute value) are with regard to real-time simulations *equivalent* to N-RAM's with the additional instructions MULTIPLY and DIVIDE for real numbers of bounded absolute value.

3 Networks of Spiking Neurons with Discrete Time

In this section we consider the case where all firing times of neurons in $\mathcal{N}$ are multiples of some μ with $1/\mu \in \mathbf{N}$. We restrict our attention to the biologically plausible case where there exists some $t_{\mathcal{N}} \geq 1$ such that for all $x > t_{\mathcal{N}}$ all response functions $\varepsilon_{u,v}(x)$ have the value 0, and also all threshold functions $\Theta_v(x)$ have some arbitrary constant value. If $t_{\mathcal{N}}$ is chosen minimal with this property, we refer to

$p_{\mathcal{N}} := \lceil t_{\mathcal{N}}/\mu \rceil$ as the *timing-precision of* $\mathcal{N}$. Obviously for $p_{\mathcal{N}} = 1$ the SNN is equivalent to a "non-spiking" neural net that consists of linear threshold gates, whereas an SNN with continuous time may be viewed as the opposite extremal case for $p_{\mathcal{N}} \to \infty$.

The following result provides a significant upper bound for the computational power of an SNN with discrete time, even in the presence of arbitrary real-valued parameters and weights. Its proof is technically rather involved. One first carries out a suitable transformation of the parameters to replace the given real-valued parameters by rationals of bounded bit-length, using a similar method as in (Maass, 1993). One then computes firing times in an efficient manner by combining a qualitative analysis of the derivatives of the involved polynomials to guide binary searches for those intervals of the form $[i \cdot \mu, (i+1) \cdot \mu)$ where they assume the value 0.

Theorem 4: *Assume that $\mathcal{N}$ is an SNN with timing-precision $p_{\mathcal{N}}$, arbitrary piecewise polynomial response- and piecewise rational threshold-functions with arbitrary real-valued parameters, and weight-functions as specified in section 1.*

Then one can simulate $\mathcal{N}$ for boolean valued inputs in real-time (with delay-factor poly$(|V|, \log p_{\mathcal{N}}, t_{\mathcal{N}}/\tau_{\mathcal{N}})$) by a Turing machine with poly$(|V|, \log p_{\mathcal{N}}, \log 1/\tau_{\mathcal{N}})$ states and poly$(|V|, \log p_{\mathcal{N}}, t_{\mathcal{N}}/\tau_{\mathcal{N}})$ tape-cells.

On the other hand any Turing machine with q states that uses at most s tape-cells can be simulated in real-time (with delay-factor $O(1)$) by an SNN $\mathcal{N}$ with any response- and threshold-functions that satisfy our basic assumptions with rational parameters and time-invariant rational weights, with $O(q)$ neurons, $\log p_{\mathcal{N}} = O(s)$, and $t_{\mathcal{N}}/\tau_{\mathcal{N}} = O(1)$.

The next result shows that the VC-dimension of an SNN with discrete time is finite, and grows proportionally to $\log p_{\mathcal{N}}$. The upper bound requires a somewhat delicate structural analysis of computations in an SNN in combination with Milnor's Theorem (see Goldberg and Jerrum, 1993). With the help of the forthcoming results of Karpinski and Macintyre, 1994, one can extend the upper bound of Theorem 5 to hold also for SNN's whose response- and threshold-functions involve the exponential function. The lower bound of Theorem 5 employs a new explicit construction in addition to that of (Maass, 1994a).

Theorem 5: *Assume that the SNN $\mathcal{N}$ has the same properties as in Theorem 4. Then the VC-dimension and the pseudo-dimension of $\mathcal{N}$ (for arbitrary real valued inputs) can be bounded by $O(|E| \cdot |V| \cdot \log p_{\mathcal{N}})$, independently of the number of spikes in its computations.*

Furthermore one can construct SNN's $\mathcal{N}$ of this type with any response- and threshold-functions that satisfy our basic assumptions with rational parameters and time-invariant rational weights, so that $\mathcal{N}$ has (already for boolean inputs) a VC-dimension of at least $\Omega(|E|(\log p_{\mathcal{N}} + \log |E|))$.

4 Relationships between SNN's with Discrete Time and other Computational Models

We consider here the relationship between SNN's with discrete time and analog neural nets. We consider in this section a perhaps more "realistic" version of recurrent analog neural nets, where the output of each gate is rounded off to an integer multiple of some $\frac{1}{a}$ (with $a \in \mathbf{N}$). We refer to a as the *number of activation levels of the analog neural net.*

It is an interesting open problem whether analog neural nets (with gate-outputs interpreted as firing rates) or networks of spiking neurons provide a more adequate computational model for biological neural systems. Theorem 6 shows (like Theorem 3) that in spite of their quite different structure the computational power of these two models is in fact closely related.

On the side the following theorem also exhibits a new subclass of deterministic finite automata (DFA's) which turns out to be of particular interest in the context of neural nets. We say that a DFA M is a *sparse DFA of size s* if M can be realized by a Turing machine with s states and space-bound s (such that each step of M corresponds to one step of the Turing machine). Note that a sparse DFA may have exponentially in s many states, but that only poly(s) bits are needed to describe its transition function. Sparse DFA's are relatively easy to construct, and hence are very useful for demonstrating (via Theorem 6) that a specific task can be carried out on a neural net with a realistic timing precision (respectively a realistic number of different activation levels).

Theorem 6: *The following classes of machines have closely related computational power in the sense that there is a polynomial p such that each computational model from any of these classes can be simulated in real-time (with delay-factor $\leq p(s)$) by some computational model from any other class (with the size-parameter s replaced by $p(s)$):*

- *sparse DFA's of size s*
- *SNN's with $O(1)$ neurons and timing precision 2^s*
- *analog neural nets that consist of $O(1)$ gates with piecewise rational activation functions with 2^s activation levels and parameters and weights of bit-length $\leq s$*
- *neural nets that consist of s linear threshold gates (with recurrencies) with arbitrary real weights.*

The result of Theorem 6 is remarkably stable since it holds no matter whether one considers just SNN's $\mathcal{N}$ with $O(1)$ neurons that employ very simple specific piecewise linear response- and threshold-functions with parameters of bit-length $O(1)$ (with $t_{\mathcal{N}}/\tau_{\mathcal{N}} = O(1)$ and time-invariant weights of bit-length $\leq s$), or if one considers SNN's $\mathcal{N}$ with s neurons with arbitrary piecewise polynomial response- and piecewise rational threshold-functions with arbitrary real-valued parameters, $t_{\mathcal{N}}/\tau_{\mathcal{N}} \leq s$, and time-dependent weights (as specified in section 1).

5 Conclusion

We have introduced a simple formal model SNN for networks of spiking neurons, and have shown that significant upper bounds for its computational power and sample complexity can be derived from rather weak assumptions about the mathematical structure of its response-, threshold-, and weight-functions. We have also proven lower bounds for the computational power and sample complexity of SNN's, and we have shown that any neurons whose response- and threshold-functions satisfy our basic assumptions can be employed for the constructions of the corresponding SNN's.

The results of this article have two interesting consequences. One is, that in order to show that a network of spiking neurons can carry out some specific task (e.g. in pattern recognition or pattern segmentation, or solving some binding problem; see e.g. Malsburg and Schneider, 1986, or Gerstner et al., 1992) it now suffices to show that a finite automaton, a threshold circuit, a Turing machine or an N-RAM (see section 2) can carry out that task in an efficient manner. Furthermore the simulation results of this article allow us to relate the computational *resources* that are needed on the latter more convenient models (e.g. the required work space on a Turing machine) to the required resources needed by the SNN (e.g. the timing precision of the SNN, see Theorem 6). In other words, one may view N-RAM's and the other mentioned common computational models as "higher programming languages" for the construction of networks of spiking neurons. The real-time simulations of this article provide automatic methods for translating any program that is written in such higher programming language into the construction of a corresponding SNN. In this way the "user" of an SNN may choose to ignore all worrisome implementation details on SNN's such as timing (potentially at the cost of some efficiency). Furthermore the matching upper bound result for N-RAM's (see Theorem 3) shows that the corresponding "higher programming language" is able to exploit *all* computational abilities of SNN's.

Secondly, the combination of our lower and upper bounds for the computational power of SNN's provides for a large class of response- and threshold-functions *exact* characterizations (up to real-time simulations) of the computational power of SNN's with real valued inputs, and for SNN's with bounded timing precision. As a consequence of these results, one can then also relate the computational power of SNN's to that of recurrent *analog* neural nets with various activation functions, thereby throwing some light on the relationships between the computational power of networks of neurons with *spike-coding* (SNN's) and networks of neurons with *frequency-coding* (analog neural nets). Furthermore the combination of our lower and upper bound results implies that there are extremely simple piecewise linear response- and threshold-functions that are *universal* in the sense that with these functions an SNN can simulate in real-time any SNN that employs *arbitrary* piecewise linear response- and threshold-functions. Equivalence-results of this type induce some structure in the "zoo" of response- and threshold-functions that are mathematically interesting or occur in biological neural systems, and they allow us to focus on those aspects of these functions which are *essential* for the computational power of spiking neurons.

Finally we would like to point out that since we have based all of our investigations on the rather fine notion of a *real-time simulation* (see section 1), our results provide information not just about the relationships between the *computational* power of the previously mentioned models for neural networks, but also about their capability to execute *learning* algorithms (i.e. about their *adaptive* qualities).

Acknowledgement

I would like to thank Wulfram Gerstner, Eduardo D. Sontag, and John G. Taylor for helpful discussions.

References

M. Abeles. (1991) Corticonics: Neural Circuits of the Cerebral Cortex. *Cambridge University Press.*

A. Aertsen. ed. (1993) Brain Theory: Spatio-Temporal Aspects of Brain Function. *Elsevier.*

L. Blum, M. Shub, S. Smale. (1989) On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines. *Bull. Amer. Math. Soc.* 21(1): 1-46.

J. Buhmann, K. Schulten. (1986) Associative recognition and storage in a model network of physiological neurons. *Biol. Cybern.* 54: 319-335.

P. S. Churchland, T. J. Sejnowski. (1992) The Computational Brain. *MIT-Press.*

W. Gerstner. (1991) Associative memory in a network of "biological" neurons. *Advances in Neural Information Processing Systems, vol. 3, Morgan Kaufmann*: 84-90.

W. Gerstner, R. Ritz, J. L. van Hemmen. (1992) A biologically motivated and analytically soluble model of collective oscillations in the cortex. *Biol. Cybern.* 68: 363-374.

W. Gerstner, J. L. van Hemmen. (1994) How to describe neuronal activity: spikes, rates, or assemblies?. *Advances in Neural Information Processing Systems, vol. 6, Morgan Kaufmann*: 463-470.

D. Haussler. (1992) Decision theoretic generalizations of the PAC model for neural nets and other learning applications. *Inf. and Comput.* 95: 129-161.

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

E. R. Kandel, J. H. Schwartz, T. M. Jessel. (1991) Principles of Neural Science. *Prentice-Hall.*

M. Karpinski, A. Macintyre. (1994) Quadratic bounds for VC-dimension of sigmoidal neural networks. Preprint (preliminary version).

P. Koiran. (1993) A weak version of the Blum, Shub, Smale model. *Proc. of the 34th Annual IEEE Symp. on Found. of Comp. Sci., IEEE Computer Society Press*: 486-495.

W. Maass. (1993) Bounds for the computational power and learning complexity of analog neural nets. *Proc. 25th Annual ACM Symposium on the Theory of Computing*: 335-344.

W. Maass. (1994a) Neural Nets with Superlinear VC-Dimension. *Proc. of The European Conference on Artificial Neural Networks 1994 (ICANN '94)*; journal version appeared in *Neural Computation* 6: 875-882.

W. Maass. (1994b) On the computational complexity of networks of spiking neurons (extended abstract). *TR 393 from May 1994 of the Institutes for Information Processing Graz*; to appear in the *Proceedings of NIPS '94.*

W. Maass. (1994c) Lower bounds for the computational power of networks of spiking neurons. Submitted for publication.

W. Maass. (1995) Vapnik-Chervonenkis Dimension of Neural Nets. To appear in the *Handbook of Brain Theory and Neural Networks, M. A. Arbib, ed., MIT-Press.*

H. T. Siegelmann, E. D. Sontag. (1992) On the computational power of neural nets. *Proc. 5th ACM-Workshop on Computational Learning Theory*: 440-449.

H. T. Siegelmann, E. D. Sontag. (1994) Analog computation via neural networks. *Theoretical Computer Science* 131: 331-360.

O. Sporns, G. Tononi, G. M. Edelmann. (1991) Modeling perceptual grouping and figure-ground segregation by means of active reentrant connections. *Proc. Natl. Acad. Sci. USA 88*: 129-133.

L. G. Valiant. (1994) Circuits of the Mind. To appear in *Oxford University*

Press.

J. van Leeuwen, ed. (1990) Handbook of Theoretical Computer Science, vol. A: Algorithms and Complexity. *MIT-Press.*

C. von der Malsburg, W. Schneider. (1986) A neural cocktail-party processor. *Biol. Cybern.* 54: 29-40.

L. Watts. (1994) Event-driven simulation of networks of spiking neurons. *Advances in Neural Information Processing Systems, vol. 6, Morgan Kaufmann*: 927-934.