The Computational Capabilities of Neural Networks

(extended abstract)

Jiří Šíma*[1]

*Institute of Computer Science, Academy of Sciences of the Czech Republic, P.O. Box 5, 182 07 Prague 8, Czech Republic, sima@cs.cas.cz, *and* Institute for Theoretical Computer Science (ITI), Charles University

1 Introduction

The (artificial) neural networks represent a widely applied computational paradigm that is an alternative to the conventional computers in many areas of artificial intelligence. By analogy with classical models of computation such as Turing machines which are useful for understanding the computational potential and limits of conventional computers, the capability of neural networks to realize general computations have been studied for more than decade and many relevant results have been achieved [1, 2, 3, 5]. The neural networks are classified into a computational taxonomy according to the restrictions that are imposed on their parameters. Thus, various models are obtained which have different computational capabilities and enrich the traditional repertoire of computational means. In particular, the computational power of neural networks have been investigated by comparing their variants with each other and with more traditional computational tools including finite automata, Turing machines, Boolean circuits, etc. The aim of this approach is to find out what is, in principle, or efficiently, computable by particular neural networks, and how to optimally implement required functions.

In the present paper a taxonomy of neural network models is reviewed from the computational point of view. The *digital* computations with binary inputs and outputs are mostly assumed here although their values may be encoded by analog states. The respective theoretical results concerning the computational power of neural networks are briefly surveyed. The main focus is on the *perceptron networks* in Section 2 whose computational properties are now fairly well understood. The computational taxonomy is extended in Section 3 with other neural network models whose computational capabilities have only recently been analyzed including the *RBF*, *winner-take-all*, and *spiking* networks. The full version of this paper appeared as technical report [4] which contains relevant references.

2 Perceptron Networks

The *perceptron network* consists of s computational *units (neurons, perceptrons)* $V = \{1, \ldots, s\}$ where s is its *size*. Assume that n *input* and m *output* units serve as an interface while the remaining ones are called *hidden* neurons. The units are connected into an oriented graph—*architecture* with each edge from i to j labeled with a *weight* $w_{ji} \in \Re$ ($w_{ji} = 0$ iff there is no edge). The *computational dynamics* determines for each $j \in V$ the evolution of its *state (output)* $y_j^{(t)} \in \Re$ in time $t \geq 0$ which establishes the *network state* $\mathbf{y}^{(t)} = (y_1^{(t)}, \ldots, y_s^{(t)}) \in \Re^s$. At $t = 0$, the network is placed in an *initial state* $\mathbf{y}^{(0)}$ (e.g. including an external input). An *excitation* $\xi_j^{(t)} = \sum_{i=0}^{s} w_{ji} y_i^{(t)}$ is assigned to each $j \in V$ at time $t \geq 0$ as the weighted sum of outputs from incident units including a *bias* $w_{j0} \in \Re$ which can be viewed as the weight of formal input $y_0^{(t)} \equiv 1$. In the **discrete-time** dynamics, the network state is updated at time instants $t = 1, 2, \ldots$ so that units $j \in \alpha_{t+1}$ from a certain subset $\alpha_{t+1} \subseteq V$ compute their new outputs $y_j^{(t+1)}$ by applying an *activation function* $\sigma : \Re \longrightarrow \Re$ to $\xi_j^{(t)}$, i.e. $y_j^{(t+1)} = \sigma(\xi_j^{(t)})$ while $y_j^{(t+1)} = y_j^{(t)}$ for $j \notin \alpha_{t+1}$. The *binary* perceptron networks with $y_j \in \{0, 1\}$ usually employ the *hard limiter* activation function: $\sigma(\xi) = 1$ for $\xi \geq 0$ and $\sigma(\xi) = 0$ for $\xi < 0$; while the *analog* networks use continuous *sigmoid* functions, e.g. the *saturated-linear* function: $\sigma(\xi) = 1$ for $\xi \geq 1$, $\sigma(\xi) = \xi$ for $0 < \xi < 1$, and $\sigma(\xi) = 0$ for $\xi < 0$; or the *standard sigmoid* $\sigma(\xi) = 1/(1+e^{-\xi})$. For the purpose of computing Boolean functions the analog states y_j of output units j are interpreted as binary outputs 0 or 1 with *separation* $\varepsilon > 0$ if $y_j \leq h - \varepsilon$ or $y_j \geq h + \varepsilon$, respectively, for some fixed threshold $h \in \Re$.

A **single perceptron** computes an n-variable *linearly separable* Boolean function $f : \{0, 1\}^n \longrightarrow$

[1]Research partially supported by project LN00A056 of The Ministry of Education of the Czech Republic.

$\{0,1\}$. The class of such functions contains only a small fraction $2^{\Theta(n^2)}$ of all the 2^{2^n} Boolean functions, including AND, OR, NOT, and is closed under the negation. Any linearly separable Boolean function can be realized by using only *integer* weights with $\Theta(n \log n)$ bits in their binary representations. The parity (XOR), on the other hand, represents the most prominent example among the functions not computable by a single perceptron. In general, the issue of whether a given Boolean function is linearly separable, is co-NP-complete problem.

The architecture of *feedforward* perceptron networks is an acyclic graph which is split into $d+1$ pairwise disjoint *layers* $\alpha_0, \ldots, \alpha_d \subseteq V$ where d is called its *depth*, so that units in any layer α_t are connected only to neurons in subsequent layers α_u, $u > t$. Usually, α_0 is the input layer (not counted in the number of layers) and α_d represents the output layer. The *network function* $\mathbf{f} : \{0,1\}^n \longrightarrow \{0,1\}^m$ is evaluated, layer after layer, in parallel time d. *Any* Boolean input-output mapping can be implemented with a **binary feedforward network** of $\Theta(\sqrt{m2^n/(n - \log m)})$ perceptrons in depth 4 (constant time) which, in the worst case, cannot be reduced even for unbounded depth (this size is optimal also for $m = 1$). The integer weights in feedforward networks are usually bounded by a polynomial in terms of input length n which clearly decreases their computational power. However, unbounded weights can still be replaced with polynomial ones by adding one more layer while only a polynomial overhead in the network size is needed. Also the exponential size in the previous universal networks is far from being practically realizable for larger input lengths although there is no specific function known so far that would require superlinear number of units. In fact, many important functions can be computed by the feedforward perceptron networks of *polynomial size* and *constant depth* d which, for *polynomial weights*, correspond to the complexity classes TC_d^0, $d \geq 1$. Thus, TC_1^0 contains all the functions computable by single perceptrons with polynomial weights, and $TC_1^0 \subsetneq TC_2^0$ since $XOR \in TC_2^0 \setminus TC_1^0$. A less known result is a deeper separation $TC_2^0 \subsetneq TC_3^0$. The witnessing function from $TC_3^0 \setminus TC_2^0$ is the *Boolean inner product* $IP : \{0,1\}^{2k} \longrightarrow \{0,1\}$ ($k \geq 1$) defined as $IP(x_1, \ldots, x_k, x_1', \ldots, x_k') = \bigoplus_{i=1}^{k} AND(x_i, x_i')$ where $\bigoplus$ stands for the parity. This means that polynomial-size and polynomial-weight three-layered perceptron networks are computationally more powerful than two-layered ones. The separation above depth 3 is unknown and it is even conceivable that NP-complete problems can be computed in three layers with a linear number of perceptrons. Another important case is the class of *symmetric Boolean functions* (including the parity) whose values depend only on the number of 1s within the input, which can be implemented by polynomial-weight three-layered networks with $O(\sqrt{n})$ perceptrons. This size cannot generally be reduced below $\Omega(\sqrt{n/\log n})$ even if arbitrary depth and unbounded weights are allowed. Hence, the feedforward perceptron networks are computationally more powerful than the *AND-OR circuits* of logical gates AND, OR, NOT which cannot compute the parity within polynomial size and constant depth. In addition, the feedforward perceptron networks of small constant depths compute efficiently basic *arithmetic functions* and thus, implement analytic functions with a large precision. For example, the two-layered perceptron networks of polynomial size and weights can be constructed for comparing two n-bit binary numbers, or for computing the multiple sum of n such numbers. Further, the product and division of two n-bit binary numbers, powering to n, and sorting of n such numbers can be achieved with three-layered perceptron networks whose depth (except for powering) cannot be reduced when polynomial size and weights are required. In some cases, one can build smaller networks if the optimal depth is increased by few layers (e.g. quadratic-size fourlayred perceptron multiplier). Even more general *trade-off* results are known among different complexity measures such as size, depth, and connectivity, e.g. for every $d > 0$, the n-variable parity can be computed by a depth-$O(d \log n / \log f)$ feedforward network of size $O(dn/f^{1-1/d})$ with $O(dnf^{1/d})$ polynomial-weight edges, and *fan-in* (the maximum number of inputs to a single unit) bounded by f.

The computational capabilities of **analog feedforward networks** with e.g. the standard sigmoid which is the most widely applied model in practical neurocomputing (back-propagation), can be compared to that of binary perceptrons. Obviously, binary units can be replaced with analog ones within the same architecture. The real states, on the other hand, may bring more efficiency in analog networks of *constant size*: the *unary squaring* functions $SQ_k : \{0,1\}^{k^2+k} \longrightarrow \{0,1\}$ ($k \geq 1$) defined as $SQ_k(x_1, \ldots, x_k, z_1, \ldots, z_{k^2}) = 1$ iff $(\sum_{i=1}^{k} x_i)^2 \geq \sum_{i=1}^{k^2} z_i$, can be computed by only two analog units with polynomial weights and separation $\varepsilon = \Omega(1)$ while any binary feedforward network computing SQ_k requires $\Omega(\log k)$ units even for unbounded depth and weights. Hence, the size of feedforward networks can sometimes be reduced by logarithmic factor when binary units are replaced by analog ones. However, the analog versions $TC_d^0(\sigma)$ with separation $\varepsilon = \Omega(1)$ equal TC_d^0 for every $d \geq 1$.

The architecture of *recurrent* perceptron networks is, in general, a cyclic graph, and hence, their computations may not terminate. *Sequential* updates satisfy $|\alpha_t| \leq 1$ for every $t \geq 1$ while *synchronous (fully) parallel* computations with $\alpha_t = V$ for all $t \geq 1$ will mostly be assumed here. Thus, a network *converges* or reaches a *stable state* $\mathbf{y}^{(t^\star)}$ at time $t^\star \geq 0$ if $\mathbf{y}^{(t^\star)} = \mathbf{y}^{(t^\star+k)}$ (within some precision for analog states) for all $k \geq 1$. To decide whether a given binary recurrent network has a stable state is NP-complete whereas the *halting problem* which is the issue of whether a network converges for a given input, is computationally even harder: PSPACE-complete for binary networks and algorithmically undecidable for analog nets with 25 units and rational weights. For the purpose of universal computations over binary inputs of arbitrary lengths, different input protocols have been introduced. In *finite* networks the input is either presented *online*, each bit every *period* of $p \geq 1$ steps, or in analog nets the input string can be encoded into a real input state. The computational power of **finite recurrent perceptron networks** with the saturated-linear function depends on the descriptive complexity of weights. For *integer weights* such (binary) networks are computationally equivalent to finite automata and are shortly called *neuromata*. In particular, a neuromaton of size $\Theta(\sqrt{q})$ can be constructed which, for period $p = 4$, simulates a given deterministic finite automaton with q states, and in the worst case, this size cannot be reduced if either $p = O(\log q)$ or polynomial weights are assumed. Also an optimal-size neuromaton with $\Theta(\ell)$ units can be build recognizing a regular language described by a given regular expression of length ℓ. Furthermore, with *rational weights* arbitrary Turing machines can be simulated step per step. Thus, any function computable by a Turing machine in time $T(n)$ can be computed by a fixed *universal* analog recurrent network with only 886 units in time $O(T(n))$ whose size can be reduced to 25 neurons at the cost of $O(n^2T(n))$ time overhead. The Turing universality is also proved for the standard sigmoid although the simulation requires exponential time per each step. Finally, with *arbitrary real weights* the finite analog nets have even super-Turing computational capabilities: polynomial time corresponds to the nonuniform complexity class P/poly and within exponential time *any* input-output mapping can be computed. Moreover, a proper hierarchy of complexity classes between P and P/poly has been shown for polynomial-time computations of analog nets with increasing Kolmogorov complexity (information contents) of real weights. However, any amount of *analog noise* reduces their power to that of neuromata or even more (definite languages).

Special attention has been paid to symmetric **Hopfield nets** with undirected architectures whose weights satisfy $w(i,j) = w(j,i)$ for all $i, j \in V$. The fundamental property of binary Hopfield nets with $w(j,j) \geq 0$ for every $j \in V$ is that, under sequential update, they always converge towards some stable state corresponding to a local minimum of bounded *Liapunov (energy)* function E defined on the state space which decreases along any nonconstant computation, e.g. $E(\mathbf{y}) = -1/2 \sum_{j=1}^{s} \sum_{i=0}^{s} w_{ji} y_i y_j$. The parallel binary Hopfield nets either reach a stable state (e.g. when E is negative definite) or eventually alternate between two different states. These convergence results hold also for analog symmetric networks which, under mild hypotheses, converge to a fixed point or to a limit cycle of length at most two for parallel updates. The *convergence time* in binary Hopfield nets of size s is trivially upper bounded by 2^s and may indeed be exponential, e.g. $\Omega(2^{s/3})$ parallel steps. In the average case, on the other hand, a very fast convergence $O(\log\log s)$ is guaranteed under reasonable assumptions. The previous bounds can be expressed in terms of the network *weight* $W = \sum_{j,i \in V} |w_{ji}|$. An upper bound $O(W)$ on the convergence time of binary Hopfield nets follows from the characteristics of the energy which yields polynomial-time convergence for networks with polynomial weights. Moreover, an analog Hopfield net can be constructed whose computation terminates later than that of any other binary symmetric network of the same representation size (the number of bits in the binary representation of weights). Further, the *capacity* of the Hopfield memory is upper bounded by the number of stable states. In average, there are asymptotically $1.05 \times 2^{0.2874s}$ many stable states in binary Hopfield nets of size s, with $w_{jj} = w_{j0} = 0$ $(j \in V)$ whose other weights are independent identically zero-mean Gaussian random variables. For a particular binary symmetric network, however, the issues of deciding whether there are at least one, two, or three stable states, are NP-complete. Also the problem of *attraction radius*, i.e. how many bits may be flipped in a given stable state so that the Hopfield net still converges to it, is NP-hard and probably cannot even be efficiently approximated. Hopfield nets have alternatively been applied to the fast approximate *combinatorial opti-*

mization. Hence, the issue of finding a state with *minimal energy* for a given Hopfield net is of special interest. However, this problem is in general NP-complete although it can be solved in polynomial time for binary nets with planar architectures. In addition, there is a polynomial-time *approximate* algorithm that solves the minimum energy problem within absolute error less than $0.243W$ in binary Hopfield nets with weight W. Finally, the *computational power* of finite Hopfield nets will be reviewed. For binary states, the Hopfield nets are computationally equivalent with *convergent* asymmetric networks within only a linear overhead in time and size. This means that not only do all binary symmetric networks converge, but all convergent computations can be realized efficiently with symmetric weights (for feedforward networks even within the same architecture). However, the finite binary Hopfield nets recognize a strict subclass of the regular languages (*Hopfield languages*). Only if the analog symmetric networks with rational weights are augmented with an external oscillator of a certain type then such devices are Turing universal.

In an alternative input protocol, **infinite families of binary recurrent networks** $\{N_n\}$, each N_n with n input units for one input length $n \geq 0$, are exploited for universal computations. The computational power of such families of polynomial-size networks correspond to the nonuniform complexity class PSPACE/poly even if one restricts only to Hopfield nets. However, their power reduces to P/poly for polynomial symmetric weights.

The computational properties of various stochastic versions of perceptron networks have also been analyzed for which the reference model of **probabilistic networks** is considered. In particular, the corresponding deterministic network is augmented with additional *random binary* input units i so that for all $t \geq 0$ the probability that $y_i^{(t)} = 1$, is given by a fixed $p_i \in [0,1]$ while $y_i^{(t)} = 0$ with probability $1 - p_i$. This probabilistic model is polynomially (in parameters) related to neural networks with other stochastic behavior including Boltzmann machines, networks with unreliable states and connections, etc. Thus, required Boolean functions are computed by such networks possibly with an *error* whose probability is bounded by $0 < \varepsilon < 1/2$. This error probability can be reduced in *probabilistic binary feedforward networks* of size s and depth d to arbitrarily small $0 < \lambda < \varepsilon$ at the cost of size $\lceil 2\log_{4\varepsilon(1-\varepsilon)} \lambda \rceil s + 1$ and depth $d+1$. Hence, for n binary (deterministic) inputs, $\lceil 8\varepsilon \ln 2/(1-2\varepsilon)^2 + 1 \rceil ns + 1$ units should in principle suffice to make such networks deterministic. Also, the complexity classes TC_d^0 ($d \geq 1$) are generalized to their stochastic versions RTC_d^0 with error probabilities $\varepsilon(n) = 1/2 - 1/n^{O(1)}$ depending on input length $n \geq 0$. Thus, $RTC_d^0 \subseteq TC_{d+1}^0$ for every $d \geq 1$ (nonuniform networks) and the probabilistic feedforward networks may indeed be more efficient since $IP \in RTC_2^0 \setminus TC_2^0$. The computational analysis has been extended to *probabilistic recurrent networks* with the saturated-linear function. With integer weights such networks recognize regular languages with a bounded error probability. For rational weights, polynomial time corresponds to the nonuniform complexity class Pref-BPP/log where a weak super-Turing capability originates from arbitrary reals p_i while rational values p_i reduce the power to the recursive class BPP. For arbitrary real weights, polynomial time corresponds to $P/poly$.

The computational dynamics of **continuous-time perceptron networks** determines analog states $\mathbf{y}(t) \in \Re^s$ for every real time $t > 0$, e.g. by a system of s differential equations $dy_j/dt(t) = -y_j(t) + \sigma(\xi_j(t))$ ($j \in V$) with boundary conditions given by $\mathbf{y}(0)$ and the saturated-linear function σ. By Liapunov function argument any continuous-time symmetric network converges to some stable state with $dy_j/dt = 0$ for all $j \in V$ which can take exponential time in terms of s. Moreover, a continuous-time asymmetric network can be constructed that simulates a given finite binary discrete-time recurrent network within only a linear-size overhead. For convergent computations such a simulation can even be achieved with the continuous-time Hopfield nets. Hence, the polynomial-size infinite families of continuous-time (symmetric) networks compute at least PSPACE/poly.

3 Other Neural Network Models

The computational capabilities of neural network models that are alternative to perceptron networks have recently been analyzed. For example, unit $j \in V$ in the **RBF networks** (*radial basis functions*) computes its "excitation" as $\xi_j^{(t)} = \|\mathbf{x}_j^{(t)} - \mathbf{w}_j\|/w_{j0}$ where $\mathbf{x}_j^{(t)} = (y_{ji_1}^{(t)}, \ldots, y_{ji_{n_j}}^{(t)}) \in \Re^{n_j}$ consists of states of units $i_1, \ldots, i_{n_j} \in V$ incident to j, and the "weight" vector $\mathbf{w}_j = (w_{ji_1}, \ldots, w_{ji_{n_j}}) \in \Re^{n_j}$ is called its *center* while the positive "bias" $w_{j0} > 0$ determines its *width*. Also, the shape of activation function differs from that of sigmoid functions, e.g. the *Gaussian function* $\sigma(\xi) = e^{-\xi^2}$ is employed. For the maximum norm and representation of binary values by two different analog states, the RBF unit can robustly implement the universal Boolean

NAND gate over multiple literals for a large class of activation functions. Thus, the deterministic finite automata with q states can be implemented by recurrent networks with $O(\sqrt{q \log q})$ RBF units.

The **winner-take-all networks** represent another neural network model that employs powerful k-WTA_n gates computing a mapping k-$WTA_n : \Re^n \longrightarrow \{0,1\}^n$ defined as k-$WTA_n(x_1, \ldots, x_n) = (y_1, \ldots, y_n)$ where $y_i = 1$ $(1 \leq i \leq n)$ iff $|\{j;\, x_j > x_i,\, 1 \leq j \leq n\}| \leq k-1$. In particular, even a simple $1 - WTA_n$ $(n \geq 3)$ cannot be implemented by any feedforward perceptron network of size less than $\binom{n}{2} + n$. Moreover, any Boolean function from TC_0^2 can be computed by a *single* k-WTA_r gate applied to polynomially many r weighted sums of inputs with positive polynomial weights.

The most prominent position among alternative models is occupied by **networks of spiking neurons** which are supposed to be more biologically plausible units than perceptrons. The states in spiking networks (including interface units) are encoded by temporal differences between so-called *spikes (firing times)* of neurons. Thus, a sequence of firing times $0 \leq y_j^{(1)} < y_j^{(2)} < \cdots < y_j^{(\tau)} < ...$ is associated with each $j \in V$, and for continuous time $t \geq 0$ denote $Y_j(t) = \{y_j^{(\tau)} < t;\, \tau \geq 1\}$ and $y_j(t) = \max Y_j(t)$ for $Y_j(t) \neq \emptyset$. Formally choose $y_j^{(0)} < 0$ and define $y_j(t) = t$ for all $0 \leq t < y_j^{(1)}$ $(j \in V)$. In addition, for every $j, i \in V$ a *response function* $\varepsilon_{ji} : \Re_0^+ \longrightarrow \Re$ of j to spikes from i in continuous time $t \geq 0$ is either excitatory (EPSP) $\varepsilon_{ji} \geq 0$ or inhibitory (IPSP) $\varepsilon_{ji} \leq 0$ ($\varepsilon_{ji}(t) \equiv 0$ for $w_{ji} = 0$). The examples of the EPSP functions include e.g. $\varepsilon_{ji}(t) = 0$ for $0 \leq t \leq \Delta_{ji}$ or $t \geq \Delta_{ji} + 2$, and $\varepsilon_{ji}(t) = 1 - |t - \Delta_{ji} - 1|$ for $\Delta_{ji} < t < \Delta_{ji} + 2$ where $0 < \Delta_{\min} \leq \Delta_{ji} \leq \Delta_{\max}$ is a *delay* of connection from i to j while the IPSP can be chosen as $-\varepsilon_{ji}$. Furthermore, for each non-input unit j a function $w_{j0} : \Re_0^+ \longrightarrow \Re_0^- \cup \{-\infty\}$ determines its nonpositive bias at time $t \geq 0$ while $w_{ji} \geq 0$ for all $i \in V$, e.g. $w_{j0}(t) = w_{j0}(0) < 0$ for $t \geq t_{ref}$, and $w_{j0}(t) = -\infty$ for a *refrectory period* $0 < t < t_{ref}$. The spikes $y_j^{(\tau)}$, $\tau \geq 1$ for each non-input j are computed recursively as $y_j^{(\tau)} = \inf\{t \geq 0;\, t > y_j^{(\tau-1)} \,\&\, \xi_j(t) \geq 0\}$ according to its excitation $\xi_j(t) = w_{j0}(t - y_j(t)) + \sum_{i=1}^{s} \sum_{y \in Y_i(t)} w_{ji} \cdot \varepsilon_{ji}(t-y)$. First the *lower bounds* on the computational power of spiking networks are reviewed. Any binary feedforward perceptron network of size s and depth d can be simulated by a network of $O(s)$ spiking units within time $O(d)$. Or any deterministic finite automaton with q states can be implemented with $O(\sqrt{q})$ spiking neurons. Finally, one can build a finite spiking network with rational weights from $[0,1]$ that simulates any Turing machine in linear time, and any input-output mapping can be computed with arbitrary real weights. The computational power of spiking networks can completely be characterized by introducing the *upper bounds*. The spiking networks with any piecewise linear response- and bias-functions are computationally equivalent to discrete-time analog recurrent perceptron networks with any piecewise linear activation functions (e.g. the hard limiter and saturated-linear functions together are universal in this context). Such networks are further equivalent to so called *N-RAM*s, which are random access machines with $O(1)$ registers working with arbitrary reals of bounded absolute values. This equivalence is shown by linear-time pairwise simulations and is still valid when all the numerical parameters are rational numbers. The Turing universality holds even for *piecewise constant* response functions but with exponential-time overhead. Indeed, the small temporal differences in the firing times (e.g. different Δ_{ji} in the piecewise constant response functions) may bring additional power: the Boolean *coincidence-detection* function $CD_k : \{0,1\}^{2k} \longrightarrow \{0,1\}$ $(k \geq 1)$ that formalizes a simple pattern-matching task as $CD_k(x_1, \ldots, x_k, x_1', \ldots, x_k') = 1$ iff there is $1 \leq i \leq k$ such that $x_i = x_i' = 1$ can easily be implemented by a single spiking neuron whose inputs are suitably delayed. Any feedforward perceptron network computing CD_k, on the other hand, requires at least $k/log(k+1)$ or $\Omega(\sqrt{k})$ or $\Omega(k^{1/4})$ units with the hard limiter or saturated-linear or standard sigmoid activation function, respectively.

In more realistic **noisy spiking networks** the excitation of unit j just governs the probability that j fires. Any Boolean function can be implemented by a sufficiently large network of noisy spiking neurons with arbitrarily high probability of correctness.

References

[1] I. Parberry: *Circuit Complexity and Neural Networks.* The MIT Press, Cambridge, MA, 1994.

[2] V.P. Roychowdhury, K.-Y. Siu, A. Orlitsky (eds.): *Theoretical Advances in Neural Computation and Learning.* Kluwer Academic Publishers, 1994.

[3] H.T. Siegelmann: *Neural Networks and Analog Computation: Beyond the Turing Limit.* Birkhäuser, Boston, 1999.

[4] J. Šíma: The computational theory of neural networks. TR V–823, ICS, AS CR, Prague, 2000.

[5] K.-Y. Siu, V.P. Roychowdhury, T. Kailath: *Discrete Neural Computation: A Theoretical Foundation.* Prentice Hall, Englewood Cliffs, NJ, 1995.