Neural and Super-Turing Computing

HAVA T. SIEGELMANN
University of Massachusetts at Amherst, Department of Computer Science, Amherst, MA 01003, USA; E-mail: hava@cs.umass.edu

Abstract. "Neural computing" is a research field based on perceiving the human brain as an information system. This system reads its input continuously via the different senses, encodes data into various biophysical variables such as membrane potentials or neural firing rates, stores information using different kinds of memories (e.g., short-term memory, long-term memory, associative memory), performs some operations called "computation", and outputs onto various channels, including motor control commands, decisions, thoughts, and feelings. We show a natural model of neural computing that gives rise to hyper-computation. Rigorous mathematical analysis is applied, explicating our model's exact computational power and how it changes with the change of parameters. Our analog neural network allows for supra-Turing power while keeping track of computational constraints, and thus embeds a possible answer to the superiority of the biological intelligence within the framework of classical computer science. We further propose it as standard in the field of analog computation, functioning in a role similar to that of the universal Turing machine in digital computation. In particular an analog of the Church-Turing thesis of digital computation is stated where the neural network takes place of the Turing machine.

Key words: analog computation, computational theory, chaos, dynamical systems, neuron

1. Neural Computing: An Introduction

The brain is unique information machinery. We understand that a nearby cat is smaller than a far-away truck even when the two retinal pictures have equal size, we notice a running tiger behind the bushes from minute hints only, and we identify a female face – tasks that are hard for any artificial vision system. Even in the early stages of visual processes, the visual system executes computation, association, and filtering, apparently to benefit action in the real world and contribute to the survival of the race. Humans, unlike classical computers, are able to mine data. Informally, *data-mining* is our ability to understand all the many green dots of a *pointilliste* painting as, for example, a single tree, and to filter out all the "irrelevant data" such as the blue and brown intermediate pieces. Data-mining enables us divide these dots into one or several trees, or even to note a general pattern in the treetops. For artificial computers, data-mining is done by complicated algorithms, which are far from being executed in real time. No classical algorithm can execute this complicated task. Another astonishing feature of the brain as an information system is its ability to work in *different modes of operation* and to automatically switch between them. When crossing a busy street, we are likely not to notice the nightingale singing and the details of the full moon, although we see them physically. Our level of sensitivity and choice of attention are chosen on-line in

Minds and Machines **13:** 103–114, 2003.

accordance with the tasks that are to be done and with the amount of background noise from either the environment or internal processes – this without investing any energy or conscious effort. Technologies that introduce any kind of adaptivity and mode changes are extremely hard to design in artificial systems.

How does the brain compute, and what makes it so superior to computers for intelligent tasks such as multi-modality sensory fusion, filtering, data-mining, and recognition? The brain seems to rely on different strategies from contemporary computers. The neural cortex is composed of a highly parallel architecture of about 10^{11} neurons. Neurons receive their input through "input channels" (dendrites) where the average in-degree is 10^4 per neuron; each neuron processes its input data and outputs a value via its "output channel" (axon). The biological nervous system proceeds by transferring information continuously like a dynamical system without relying on any static memory; yet it remembers. The system can incorporate analog physical values, continuous time updating, and various analog media of encoding and fusing data, which digital models of computation cannot handle. A neuron is slower by six orders of magnitude than a basic silicon gate, implying that brain superiority is not due to fast operation. An axon transfers information to the next dendrites through chemical gates (synapses), giving rise to the adaptation and changing of the architecture as part of the learning experience. This is in contrast to the static architecture of a present-day computer.

Neural computing combines research on different levels: the molecular level, the synaptic level, the levels of the single neuron, cortical circuits, cortical maps, and neural maps, and up to the system level. It combines the work of biochemists, neurophysiologists, neurobiologists, computer scientists, engineers, psychologists, and philosophers. On the lower levels, attention is given to the detailed structure of the single neuron and to the development of bottom-up approaches to computation: what functions can be executed on neuronal "wetware"? On the highest level, philosophers and computer scientists focus on the type of computation and functions enabled by neural concepts.

This paper focuses on the high level research of neural computation and describes new computational theories that have been developed to explain and analyze brain-like computational paradigms as well as respective innovative technologies.

2. Computational Models

Computational theory has developed hand in hand with the field of neural computation. The young but fascinating research fields of *analog computational* theory and of *hyper-computation* are closely related to neural computation.

The Turing machine was suggested in 1935–36 as a model of a mathematician who solves problems by following a specifiable fixed algorithm and using unlimited time, energy, pencils, and paper. Turing's 1938 search for models of hypercomputers (that outperform this mathematician), together with his later emphasis

on learning and adaptation, probably reflects his understanding that there are other kinds of computation beyond the static, fully specifiable algorithm (Copeland, 2000; Copeland et al., 1999). The brain, for example, could be perceived as a powerful computer with its excellent ability for speech recognition, image recognition, and the human ability to develop new theories. The nervous system, constituting an intricate web of neurons with 10^{14}synaptic connections that adapt with experience, *cannot be perceived as a static algorithm*; the chemical and physical processes affecting the neuronal states, like other natural processes, are based on exact real values and are thus *not specifiable by finite means.*

The second meeting of the Josiah Macy Jr. scientific conference was held in 1943. Its aim was to identify applicable parallels between brain and computer. In this meeting, which was attended by researchers from diverse areas of science, social science, and engineering. McCulloch and Pitts demonstrated the first computational model of the neuron. They argued that because of the on/off nature of action potentials (as seen by Hodgkin and Huxley), the nervous system could be thought of as a finite interconnection of logical devices. The theory of McCulloch and Pitts was controversial. Many stressed that while digital behavior is a recognized component of cerebral functioning, any theory that did not take into account the relevant continuous variables (i.e., the underlying chemical and physical phenomena) could not be considered a faithful model. From ensuing debates and discussions, two different approaches emerged. The first, led by Gerard and Bateson, emphasized the significance of continuous-valued computation; the second, headed by McCulloch, Pitts, and von Neumann, stressed the advantages in the simplicity of digital computation Nyce (1992, p. 32) (although von Neumann acknowledged analog values to exist in neural functioning (e.g., von Neumann, 1958). However, following the development of von Neumann's model of universal computation, based on the principle of the McCulloch-Pitts neuron and on the universal Turing machine, the digital approach prevailed in the field of cybernetic research. The digital model laid the groundwork both for the 20th century's computer paradigm and for an enhanced understanding of computational processes in the brain.

Towards the end of the 1980s, the continuous neural network model was revived. Unlike the output values 0,1 of the McCulloch-Pitts neuron, the newer neural models calculate continuous values between 0 and 1. These models were crucial for the development of adaptive technologies, including the classical back-propagation algorithm that learns neural parameters (e.g. Rumelhart et al., 1986), and they enabled new theoretical foundations to be laid for machine learning and brought about engineering tools such as optimal controllers. The same advance from the discrete to the continuous model was adopted concurrently in biological modeling of the nervous system.

Soon afterwards, the unavoidable fundamental question was raised: how to characterize computational models described by networks of continuous neurons? The need for new theories describing the operations and capabilities of machines that are analog or adaptive have become imperative if one wants to describe and

analyze nature's computation as well as the already developing analog chips and adaptive technologies. This is the goal of the work described in this paper (see also Siegelmann, 1998).

3. Analog Computation

In the field of analog computation, any physical system observed by an experimenter in a laboratory and any dynamical activity in nature is perceived as performing a computational process. Beginning from an initial state (input), a physical system evolves in its state space according to an update equation (the computation process) until it reaches some designated state (the output). Such natural processes can be modeled with dynamical systems of the form $x(t+1) = f(x(t))$ or $\mathrm{d}x/\mathrm{d}t = f(x(t))$ by identifying a set of internal variables x together with a rule f that describes the transformation from state to state.

Three main properties distinguish analog from digital models (Siegelmann, 1998):

1. Analog computational models are defined on a *continuous phase space* (e.g. where the variables x may assume analog values), while the phase space of a digital model is inherently discrete.
2. Physical dynamics are characterized by the existence of *real constants* that influence the macroscopic behavior of the system. In contrast, in digital computation all constants are in principle accessible to the programmer.
3. The motion generated by a physical system is "locally continuous" in the dynamics. That is, unlike the flow in digital computation, statements of the following forms are not allowed in the analog setup: "tests for 0" or "if $x > 0$ then compute one thing and if $x < 0$ then continue in another computation path."

Continuous time dynamics is part of many analog systems as well, and we distinguish between continuous versus discrete time analog computational models.

Note that although the physical system contains internal continuous values, *discreteness of the output* may be dictated by the limited precision of the measurement tools used to probe the continuous phase space. This brings us to a discrete I/O, as in the definition of digital computation.

Blum, Shub, and Smale introduced a discrete time computational model that operates in each time step on real valued registers, irrespective of their binary representation, and allows for real constants as well (Blum et al., 1989). The BSS model is considered a model of computation over the real numbers, rather than a model of analog computation, because it lacks the property of local continuity (item 3 above), which is crucial in a framework of analog computing. *Hybrid models* of computation behave similarly. These combine discrete and continuous time dynamics, usually by means of ordinary differential equations (ODEs) that are governed by finite automata. Due to their finite automaton component, hybrid systems also do not adhere to local continuity. A *Coupled Map Lattice* is the analog version of

a cellular automaton. This model is composed of an infinite lattice of variables in which all variables are updated by the same local transition rule. This definition is general enough to include any discrete time model. Other models of analog computers were proposed in Shannon (1941); Pour-El (1974) and were found to be computationally weak.

Next we will describe the model called the *Analog Recurrent Neural Network* (Siegelmann et al., 1994). This model was shown to be hyper-computational.

4. The Analog Recurrent Neural Network Model

A mathematically simple but biologically relevant model is a finite assembly of simple processors (or neurons) called the Analog Recurrent Neural Network (ARNN). This network was suggested jointly with Eduardo Sontag, a mathematician and control-theorist from Rutgers University, and the material in this section summarizes the work in Siegelmann et al. (1994); Siegelmann (1998).

Unlike the von Neumann computer model, the structure of the neural network cannot be separated into a memory region and a processing unit; memory and processing are strongly coupled. Each neuron is part of the processing unit, and the memory is implicitly encoded in the mutual influence between any pair of neurons. The influence is represented by a real number weight. The status of the weights, whether perceived as unknown parameters that are to be estimated or as fixed constants (after being learnt), prompts the two different views of the neural model: when the weights are considered unknown parameters, the network is an adaptive technology that is to approximate input–output mappings by means of parameter estimation – also called *learning*. When the weights are considered constant, the networks can perform exact computations rather than mere approximations. The network is “analog” in that its neurons update within a continuous phase space, it updates with local continuity, and it incorporates real constant weights. The adjective "recurrent” emphasizes that the interconnection is general, rather than layered or symmetrical. Such architecture allows for an internal state representation (x) and describes its evolution over time.

(Comment: A related model is the *network of spiking neurons*. Here the neurons output binary values, but the time intervals between consecutive spikes are exact reals. The computational analysis of the two models is equivalent under the transformation of neural value and time interval Maass (1996) and we thus consider the analog recurrent networks only.)

Formally, the ARNN consists of a finite number of simple neurons updating in discrete time. There are two input channels into which inputs are presented. Each neuron updates its activation value (state) according to a function of the activations (x_j), inputs (u_j), and a set of real coefficients/weights (a_{ij}, b_{ij}, c_i):

$$x_i(t+1) = \sigma\left(\sum_{i=1}^{N} a_{ij}x_j(t) + \sum_{j=N}^{M} b_{ij}u_j(t) + c_i\right) \quad i = 1, 2, \ldots$$

where N is the number of processors, M (=2) is the number of external input signals, and σ a very simple "sigmoid-like" function, the saturated-linear function:

$$\sigma(x) = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } \leq x \leq 1 \\ 1 & \text{if } x > 1 \end{cases}$$

As part of the description, we assume that we have singled out a subset of the N processors, say $x_{i1}...x_{ip}$; these are the p output processors that communicate the output of the network to the environment. Thus a particular network of this type is specified by its weights and the set of output processors.

The structure of the network, including the values of the interconnection weights, does not change in time, but rather remains constant. What changes in time are the activation values, or outputs of each processor, which are used in the next iteration. In this sense our model is "uniform". We next show this model to be hyper-computational and analyze its exact computational power.

5. Analyzing the Analog Recurrent Neural Network

A Nonuniform model of computation was introduced by Karp and Lipton (1982). It generalizes the concept of monotonic response time as a function of the input length, such as the class P (the class of all recursive functions that are computable in polynomial time), to include growing hardware: there exists a function f such that output for an input of length n is calculated by an acyclic interconnection of $f(n)$ digital components (e.g., McCulloch-Pitts neurons or logical gates). Inputs of different length are computed by different hardware, and the different interconnections can not be all described by a Turing machine. Karp and Lipton (1982) showed that such a family of nonuniform acyclic digital circuits is able to compute a larger set of functions than the classical Turing machine. When f denote polynomial functions, the resulting families of circuits compute a class called P/poly: the letter "P" is reminiscent of computation in polynomial time, and the term "poly" is to note that the sizes of the different circuits are polynomial in the input length. P/poly was proved to contain all functions in P, some but not all recursive functions above the class P, and even some nonrecursive (super-Turing) functions as well.

This ARNN model was proved to compute in polynomial time exactly the functions in P/poly. That is, the ARNNs are hyper-computational but still have a well-defined computational class (Siegelmann et al., 1994).

We next try to explain this class of functions P/poly using the model of a Turing machine with an external advisor who wants to help a Turing machine to compute more input-output maps than there are in the class P. This model may help some of us comprehend what the analog networks compute. So, assume that a Turing machine is given, and that in addition to the input, the machine also receives some extra bits of advice.

1. Assume first that for each input, the advisor can provide one bit of advice, and the advisor knows everything with no limits. Under such general terms, he may describe any binary map he wants (by providing the output bit for any input string) and the machine would be able to compute every binary language, including nonrecursive ones. So, we want to put some constraints on the advice.
2. A reasonable rule is that the advisor will not supply a bit for every input, but rather one bit for all inputs having the same length. Inputs of different length may receive independent advice. (This is called *nonuniform advice*.) The machine can now still compute more than P, but less than before, since the 2^n words of length n receive only one bit of advice.
3. The analog neural network computes more than the class in item 2 above. Because for larger n there are more input strings of length n, the advisor will now give more advice bits as n grows. He provides n^2 bits, or n^3 or any polynomially long advice, but still with the constraints that all input words of the same length receive the same advice. The resulting super-Turing class of nonuniform polynomial advice is what the analog neural network computes in polynomial time.

6. Connecting with Classical Computability

One may wonder about the connection between these computationally powerful neural networks and classical computability. We have proved (Siegelmann et al., 1995) that the classical Turing Model is a subset of the recurrent networks. If one restricts attention to networks whose interconnection weights are all rational numbers, then one obtains a model of computation that is polynomially related to Turing Machines. In particular, we proved that given any multi-tape Turing Machine, one can simulate it in realtime by some network with rational weights, and of course the converse simulation in polynomial time is obvious. We counted and found a network made up of 886 neurons that computes all partial recursive functions.

In the Turing model, to get the P/poly power, one has to add nonuniformity. The neural network framework, on the other hand, is both natural and elegant for computation beyond the Turing limit: a change in the weights from rational numbers to real numbers is all that is needed to change the power of the system from that of a Turing Machine to the class P/poly.

Two questions arise from considering Turing Machines as a subset of recurrent networks. First, are there some networks whose computational power lies between P and P/poly? Second, what is the essence of the difference between rational and real numbers that causes such a discrepancy in computational power? Together with Jose Balcázar and Ricard Gavaldà we have proved (Balcázar et al., 1997) that the equivalence between neural networks and either Turing Machines (P) or Turing Machines with polynomial advice (P/poly) constitute only the two extremes of the

computational hierarchy spanned by the neural networks. To do this, we recruited a tool from information theory, the Kolmogorov characterization.

Kolmogorov complexity is a measure of the information contained in an individual object. We used a variant of Kolmogorov characterization to measure the information encoded in numbers; this variant takes into account computation efficiency as well. We then defined rational numbers, real numbers and an infinite hierarchy in the real numbers (between rationals and "total-reals") in terms of their *resource-bounded Kolmogorov complexity*. Rationals have the complexity [1,n] since rationals and any other linear time Turing generated number can be described in constant time and linear generation time. Reals have the complexity [n,n] since all the bits have to be described and there is no way around it. Middle complexity can be for example [log$n$,n]. In this way, we were able to reveal an infinite hierarchy of computational classes associated with neural networks that have weights of increasing Kolmogorov characterization; the hierarchy lies between Turing Machines and Turing Machines with polynomial advice.

7. Real Values in Analog Networks

One may argue that the analog networks, which require infinite bit description, are, for all practical purposes, useless since systems with infinitely precise constants cannot be built. However, the real weights are appealing for the *mathematical modeling of analog computation* that occurs in nature. In nature, the fact that the constants are not known to us, or cannot even be measured, is irrelevant for the true evolution of the system. For example, the planets revolve according to the exact values of G, π, and their masses, regardless of our inability to gauge these values. Although one could replace these constants with rational numbers and observe similar qualitative behavior in finite time simulation, the long-term infinite-time characteristics of the system depend on the precise real values. Put differently, the analog network can be thought of as a mathematical idealization that assumes continuous phase space and precise data. This assumption of continuity is common in mathematics. Once it allowed the introduction of integrals to approximate volumes; another time it enabled the design of polynomial algorithms to solve linear programming, where the previously used discrete algorithms were exponentially slow. Perhaps this continuous framework is what makes many neural networks to be so efficient as well.

Because analog neural networks are defined with unbounded precision, one could think, naively, that infinite precision is required to fully describe their computation; however, we found that this is not the case. We have proved that "linear precision suffices" (Siegelmann et al., 1994) that is, up to the qth step of the computation, only the first $O(q)$ (e.g., number which is linear in q) bits in both weights and activation values of the neurons influence the result.

This property of being indifferent to small changes in the internal value, where "small" is measured as a function of the computation time, can be interpreted as a

weak property of robustness. This robustness includes changes in the precise form of the activation function, in the weights of the network, and even an error in the update. In the classical model of (digital) computation, in contrast to in our model, this type of robustness cannot even be properly defined.

The amount of information necessary for the neural network is identical to the precision required by chaotic systems. Here we call it linear precision, where we mean linearly many bits as a function of the computation time. In chaotic systems the jargon is 'exponential precision', which is calculated in terms of the value of the computation time. These are equivalent quantities in different wordings. For chaotic systems, if you want to know the behavior after t steps, you have to know the variable values in precision 2^t. Higher precision will not change the result and smaller precision will not suffice to describe the dynamics. Due to the equivalence of descriptive precisions, neural networks may constitute a framework for the modeling of physical dynamics.

8. Hyper-Computation via Analog Neural Networks

Many dynamical systems and idealized chaotic systems that cannot be described by the universal Turing machine are well captured within the framework of the analog neural network (Siegelmann, 1995). Thus the network of continuous neurons can be considered as standard in the field of analog computation, functioning in a role similar to that of the universal Turing machine in digital computation. An analog of the Church-Turing thesis of digital computation can be formulated as follows:

> No possible abstract analog device can have more computational capabilities (up to polynomial time) than Analog Recurrent networks.

There is an alternative interpretation of this thesis. It can be perceived as differentiating between static computational models and dynamically evolving ones with learning capabilities. The classical computing paradigms are static; they include only rational constants and are bounded by the power of the universal Turing machine. Evolving machines, on the other hand, can tune their internal constants/parameters, possibly on some continuum where the values would not be measurable by an external observer.

Since the networks have the property "linear precision suffices", their parameters need not be fully tuned when beginning the computation. This implies that an interleaved process of learning (the qth bits of the constants) and computing (the qth step) has super-Turing power. It is reasonable to suggest that the speed and accuracy of the learning process are the two parameters that determine the network's exact location on the computational continuum, ranging from the static classical models all the way up to adaptive models.

We conclude that – as with analog architectures – neural networks, Bayesian networks and other adaptive algorithms, if adapting on a continuum, are mathematically proven to outperform the static digital model.

In Siegelmann (1995) a chaotic dynamical system that is computationally equivalent to the recurrent neural network model was presented. This system, called "the analog shift map", is an almost classical chaotic dynamical system. It is also associated with the analog computation models suggested in this paper (and hence the term 'analog'). More than that, it is a mathematical formulation that is conjectured to describe idealized physical phenomena. To the extent that the analog shift map does describe idealized physical phenomena, the same is true for neural networks.

The work described in this paper constitutes the rigorous mathematical foundation of the analog recurrent neural network model. This model is particularly interesting because of its popularity in engineering applications for automatic learning and time series prediction; and also because it can model natural biological systems. It is especially interesting mathematically, as it allows for unbounded (analog) precision while still being bounded in processing elements and having no long-term memory registers. Our model may also be thought of as a possible answer to Penrose's recent claim Penrose (1989) that the standard model of computing is not appropriate for modeling true biological intelligence. Penrose argues that physical processes, evolving at a quantum level, may result in computations which cannot be incorporated in Church's Thesis. The analog neural network does allow for non-Turing power while keeping track of computational constraints, and thus embeds a possible answer to Penrose's challenge within the framework of classical computer science.

9. Comment on Stochastic Analog Neural Networks

It is interesting to consider what happens to the analog networks when they exhibit stochastic and random behavior. To obtain such a model, we extend in a natural way the von Neumann model of unreliable interconnection of components to the area of neural networks, and incorporate Shannon's random-noise approach of independent fixed noise.

The model results from adding to the network a heads/tails coin, or a stochastic neuron, where the probability of falling on heads or tails is a real number.

$$b = \begin{cases} 0 & \text{with probability } p \\ 1 & \text{with probability } 1 - p \end{cases}$$

Although p in this model is a real number, it is never accessed by any neuron: the coin is binary so that the other neurons receive only digital values from it.

We say that the language L is epsilon-recognized in time T by a stochastic network if every input string of length n is classified in time $T(n)$ by every computation path of that input, and the error probability (e.g., the fraction of runs in which output is wrong) in deciding the input relative to the language is bounded by $\epsilon < 1/2$.

While for real weight analog networks, the class of stochastic languages is computationally equivalent to the class of deterministic languages, when the weights

are rational stochasticity does add power. Stochastic networks with rational weights and real probabilities compute an intermediate super-Turing class between P and P/poly. This class is called BPP/log (Siegelmann, 1998, 1999).

We conclude that in any process that is affected by a real number, explicitly (like as a weight) or implicitly (like in the stochastic process that emits binary values according to a real stochasticity) the real value brings on nonrecursive computation (Siegelmann, 1999). In these terms the analog recurrent neural networks are the high end of the nonrecursive efficiently computed classes, where the stochastic and the deterministic versions are equal.

Note that the stochasticity described in this section is very particular in the sense that it is modeled by a coin (as in the von Neumann noise model). When the noise itself is defined by a richer continuous function, the computational power of the network may be reduced to regular or even definite languages, which are a strict subset of regular languages (see, for example, Siegelmann, 2002). A language is called definite if for some integer r, any two strings coinciding on the last r symbols are either both or neither in the language. Definite languages are reminiscent of short-term memory. The type of stochasticity and noise is what decides the computational power from the weak definite languages all the way up to hyper-computation.

References

Balcázar, J.L., Gavaldà, R. and Siegelmann, H.T. (1997), 'Computational Power of Neural Networks: A Characterization in Terms of Kolmogorov Complexity', *IEEE Transactions on Information Theory* 43(4), pp. 1175–1183.

Blum, L., Shub, M. and Smale, S. (1989), 'On a Theory of Computation and Complexity Over the Real Numbers: NP Completeness, Recursive Functions, and Universal Machines,' *Bull. A.M.S.* 21, pp. 1–46.

Copeland, B.J. (2000), 'Narrow Versus Wide Mechanism', *Journal of Philosophy* 97, pp. 5–32.

Copeland, B.J. and Proudfoot, D. (1999), 'Alan Turing's Forgotten Ideas in Computer Science', *Scientific American* 280, pp. 99–103.

Hopfield, J.J. and Tank, D.W. (1985), 'Neural Computation of Decisions in Optimization Problems', *Biological Cybernetics* 52, pp. 141–152.

Karp, R.M. and Lipton, R. (1982), 'Turing Machines That Take Advice', *Enseignment Mathematique* 28, pp. 191–209.

Kay, L.M., Lancaster, L.R. and Freeman, W.J. (1996), 'Reafference and Attractors in the Olfactory System During Odor Recognition', *International Journal of Neural Systems* 7(4), pp. 489–495.

Koch, C. and Crick, F.C. (2000), in M.S. Gazzaniga, ed., *Some Thoughts on Consciousness and Neuroscience The Cognitive Neurosciences*, 2nd edition, MIT Press, Cambridge, MA, pp. 1285–1294.

Maass, W. (1996), 'Networks of Spiking Neurons: The Third Generation of Neural Network Models', *Electronic Colloquium on Computational Complexity (ECCC)* 3 (031).

Nyce, J. (1992), 'Analogy or Identity: Brain and Machine' at the *Macy Conferences on Cybernetics SIGBIO Newsletter: Published by the Association for Computing Machinery*, Special Interest Group on Biomedial Computing 12, pp. 32–37.

Orponen, P. (1997), 'A Survey of Continuous-Time Computation Theory,' in D.-Z. Du and K.-I Ko, eds, *Advances in Algorithms, Languages, and Complexity*, Dordrecht: Kluwer Academic Publishers, pp. 209–224.

Penrose, R. (1989), *The Emperor's New Mind*, Oxford: Oxford University Press.

Pour-El, M.B. (1974), 'Abstract Computability and its Relation to the General Purpose Analog Computer (Some Connections Between Logic, Differential Equations and Analog Computers)', *Transactions of the American Mathematical Society* 199, pp. 1–29.

Rumelhart, D.E., Hinton, G.E. and Williams, R.J. (1986), 'Learning Representations by Back-Propagating Errors', *Nature* 323, pp. 533–536.

Shannon, C.E. (1941), 'Mathematical Theory of the Differential Analyzer', *Journal of Mathematics and Physics of the Massachusetts Institute of Technology* 20, pp. 337–354.

Siegelmann, H.T. (1995), 'Computation Beyond the Turing Limit', *Science* 238(28), pp. 632–637.

Siegelmann, H.T. (1998), *Neural Networks and Analog Computation: Beyond the Turing Limit*, Boston MA: Birkhauser.

Siegelmann, H.T. (1999), 'Stochastic Analog Networks and Computational Complexity', *Journal of Complexity* 15(4), pp. 451–475.

Siegelmann, H.T. (2002), 'Neural Automata and Analog Computational Complexity', in M.A. Arbib, ed., 2nd edition, *The Handbook of Brain Theory and Neural Networks*, Cambridge, MA: MIT Press, in press.

Siegelmann, H.T., Ben-Hur, A. and Fishman, S. (1999), 'Computational Complexity for continuous Time Dynamics,' *Physical Review Letters* 83(7), pp. 1463–1466 (Full version to appear in *Journal of Complexity*).

Siegelmann H.T. and Fishman S. (1998), 'Computation by Dynamical Systems,' *Physica D* 120, pp. 214–235

Siegelmann, H.T. and Sontag, E.D. (1994), 'Analog Computation via Neural Networks,' *Theoretical Computer Science* 131, pp. 311–360.

Siegelmann, H.T. and Sontag, E.D. (1995), 'Computational Power of Neural Networks,' *Journal of Computer System Sciences* 50(1), pp. 132–150.

von Neumann, J. (1958), *The Computer and the Brain*, New Haven: Yale University Press.