SLAC-PUB-4963
May 1989
(A)

Real-Time Applications of Neural Nets*

J.E. Spencer

Stanford Linear Accelerator Center, Stanford University, Stanford, CA 94309

Abstract: Producing, accelerating and colliding very high power, low emittance beams for long periods is a formidable problem in real-time control. As energy has grown exponentially in time so has the complexity of the machines and their control systems. Similar growth rates have occurred in many areas e.g. improved integrated circuits have been paid for with comparable increases in complexity. However, in this case, reliability, capability and cost have improved due to reduced size, high production and increased integration which allow various kinds of feedback. In contrast, most large complex systems(LCS) are perceived to lack such possibilities because only one copy is made. Neural nets, as a metaphor for LCS, suggest ways to circumvent such limitations. It is argued that they are logically equivalent to multi-loop feedback/forward control of faulty systems. While complimentary to AI, they mesh nicely with characteristics desired for real-time systems. Such issues are considered, examples given and possibilities discussed.

1. Introduction

A recent workshop[1] agreed on three major concerns for real-time systems: ~~time, r~~eliability and environment in this order. If high performance and reliability are defined by meeting the needs of the specific problem environment, neural nets(NN) appear ideal on all points. The growing importance of real-time computing and control makes this a good area for exploration. A specific concern is the growing size and complexity of the systems that have to be supported. This worsens time response and reliability and makes a more ill-defined problem environment which makes control more complicated and hard to verify. As a result, systems become increasingly unresponsive and 'user unfriendly'.

Since this is a common problem, new concepts, hardware and software seem necessary. We compare AI and NN which are complementary in many respects. NN *finds* rules for LCS rather than follows models which might be too simple or rigid or too detailed and complicated for the time scales involved. Where AI is synonymous with environments that are knowable *a priori*, adaptive learning in unknown, incomplete or noisy environments is the domain of NN, which can learn and improve or modify rules in a natural way when the knowledge domain increases or changes. A corollary: it can find more efficient i.e. faster and less complex control schemes.

Science and its machines have been a major force pushing the limits of both computing and control. Various von Neumann bottlenecks, in precisely the areas of concern here, have occurred repeatedly from his first justifications[2] for the 'von Neumann computer' to the present. His architecture was to: "revolutionize the purely mathematical approach to...nonlinear differential equations...extend quantum theory to systems of more particles and more degrees of freedom...and remove many bottlenecks in the computing approach to ordinary and electron optics." Such computers use central processors operating sequentially via program counters on stored data and instructions. However, von Neumann was also aware of alternatives like NN that combine processor and memory in a Turing-equivalent machine that could make real-time systems more efficient *and* reliable[3].

*Funded by U.S. Department of Energy contract DE-AC03-76SF00515.

1.1 Growth of Complexity

Size and complexity are two important environmental factors that strongly influence time and reliability. Figure 1 considers complexity for two very different size scales – the growth in the number of transistors per IC and the average radius of electron storage rings in mm[4]. The assumption is that the number of elements per unit length stays roughly constant. Only Intel μ's are shown beginning with the first one – the 4-bit *i*4004 that was used in calculators. The *i*80486 is 2-4 times faster than the *i*80386 for about 2.5 times the cost but includes RISC features, memory management functions and a math coprocessor.

The control system for the Large Electron-Positron ring LEP at CERN utilizes more than 20 workstations and PC's as user interfaces in their CCR with another 50 for process control and some 2000 micros for local equipment control. One can make a case that these large systems require a network of many local as well as centrally located control points to be able to run i.e. that there is a correlation between the curves of Fig. 1. The large Superconducting Super Collider(SSC) proposed recently is about three times larger and intrinsically more complex than LEP.

Our goal is a method that is computable, consistent and decidable i.e. capable of computation, comparison and optimization. However, lattice calculations for LCS such as SSC provide no guarantees of important parameters even at the level of single particle dynamics. That LCS are often indeterminant implies only that the design and control are inseparable because feedback, when fast enough, clean enough and based on a sufficiently accurate time history, can subvert such effects. This is why we repeatedly link computation and control and why one needs new technology that integrates data acquisition, analysis and control in a *real-time,* Turing-equivalent machine.

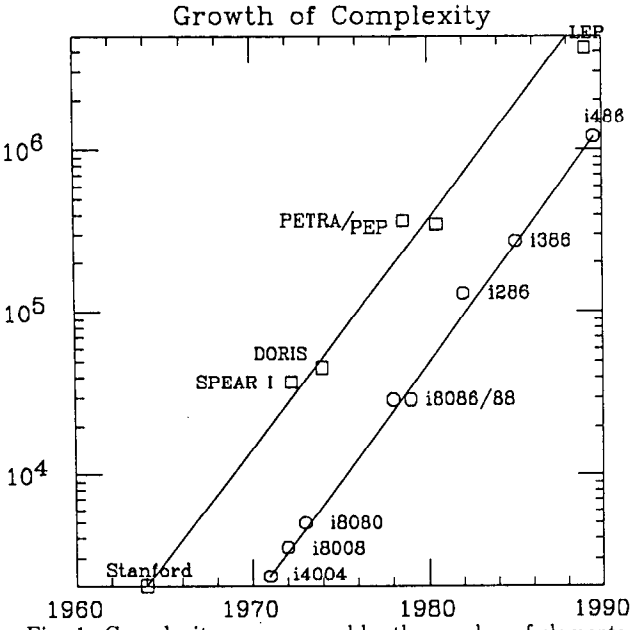

Fig. 1: Complexity, as measured by the number of elements, versus time for Intel microprocessors and electron storage rings labelled by their various acronyms.

Invited talk presented at the Real Time Computer Applications in Nuclear Particle and Plasma Physics, Williamsburg, VA., May 16–19, 1989

2. Background/History Lessons

The acronym NN originated from simple models of the neuron and its connectivity in the central nervous system. They were demonstrated to be capable of complete propositional logic[5] and equivalent to any finite state Turing machine. Thus they provide an alternative means of computing to the von Neumann machine. It's applicability to parallel processing machines and their programming has produced the acronym PDP(Parallel Distributed Processing)[6] which helps in territorial disputes between some neurophysiologists and computer scientists.

Because it involves certain aspects difficult to formulate or conceptualize, NN has produced a number of diverse approaches and 'spinoffs' each of which has its own machinery e.g. spin models based on statistical mechanics consider neurons as quasi-particles or microscopic, magnetic moments. Some prefer the broad group called cellular automata where the cell and its interactions can be quite simple. Many of these approaches are remote from real neurons and many are not exactly soluble by present methods except through significant simplifications e.g. nearest neighbor interactions within structures like strings that make this well defined. They provide interesting insights into collective effects like phase transitions and offer the possibility of 'solving' incredibly complex problems in real-time via their basic, physics-like architectures.

However, it is useful to recall that Turing-Church hypothesize that programmable simulations of any physically realizable process are possible only with unlimited space and time[7]. If one makes the plausible assumption that the simulation will generally proceed at a slower pace than the actual process, then simplifications are necessary and these may be gross when structured programs are used for real-time applications. Thus, while non-analytic methods may seem abhorrent, the alternative of oversimplified models must also be questioned because it is inconsistent with the basic hypothesis or definition of simulation.

It would be nice to have a simple, lucid image but this may require a far more complete understanding, progressing through successive levels from the atomic or molecular to the subcell/chromosome, single cell and many cell domain. It will probably reveal new concepts along the way. Here we assume a uniform, parallel architecture based on locally interacting cells which all produce small but finite time delays of equal length and explore some of the time and reliability consequences e.g. the time dispersion, the gain and loss of synchronization and dependence of excitation on distance. Feedback can provide clock cycles with 'programmable' delays and interrupts in such schemes.

Minsky and Papert have just reissued *Perceptrons*[8] with an historical prologue some twenty years after its first appearance with some remarks worth pondering: "One reason why progress has been so slow in this field is that researchers unfamiliar with its history have continued to make many of the same mistakes that others have made before them. Some readers may be shocked to hear it said that little of significance has happened in this field (since publication almost twenty years ago)." While this may seem harsh, consider how little basic hardware has been developed since von Neumann's preamble to his 1952 paper[3]: "The subject-matter, as the title suggests, is the role of error in logics, or in the physical implementation of logics — in automata-synthesis. Error is viewed, therefore, not as an extraneous and misdirected or misdirecting accident, but as an essential part of the process under consideration — its importance in the synthesis of automata being fully comparable to that of the factor which is normally considered, the intended and correct logical structure."

3. Relation Between AI and NN

The growth of complexity has many implications. Experts may become more specialized so they individually support less and less of the total system and are also more vulnerable to it. This may explain the remarkable interdisciplinary interest in NN. Beyond this comes the time and difficulty of implementing expert or other algorithmic systems as well as the problems that arise when the underlying systems or data change. One suspects that these are symptoms of more significant underlying problems. First, there may not be experts for some subsystems and in particular for the complete LCS. Second, the experts may not know *or* be willing to admit how they actually accomplish certain jobs e.g. it may be by heuristic search but whatever it is, it is their method. This leads to a more fundamental problem. The 'expert' is himself a unique neural net that may be difficult to reconcile with other NN e.g. experts or interrogators. In particular, it may require another NN to properly simulate it via training rather than software. When Bertrand Russell said: "Reason is the proper means to an end" he did not mean that all ends should be achieved by deductive means but rather that only *proper* means are rational[9].

The availability of the PC together with minimal needs for application specific software makes NN very attractive for data intensive problems. Physics differs only superficially from other fields in that its methods are more direct, simpler and thus controllable and quantifiable in time and scale. However, as its machines (systems) get bigger and more complex, they get harder to design and control so the situations become more analogous. Modeling permits a variety of ideas and interrelationships to be evaluated in comparatively short times using NN in conjunction with available databases. This modeling includes studies on NN because any parallel network architecture can be simulated on the von Neumann architecture – given time and memory.

Many distinctions have been made between AI and NN e.g. the rather ironic one that NN is not a branch of AI. Seymour Papert has described them as two distinct offsprings of cybernetics with AI being the more productive and popular. While they both share the common goal of simulating intelligent behavior their means are very different. The historical development is an interesting story which explains the clear but controversial distinctions. Many are directly related to the available 'hardware' i.e. the brain and the conventional digital computer(CC). Some distinctions which seem valid – at least for now, can be represented as follows:

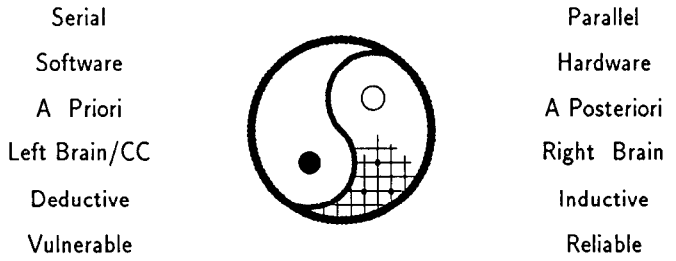

Figure 2: Very Schematic Comparison Between AI and NN.

One easily diagnoses a split personality here with many apparent complimentary aspects such as graphic/numeric, visual/verbal and macroscopic/microscopic conflicts. Minsky and Papert argue against such symbolist/connectionist dialectics in a new prologue to *Perceptrons*[8]. However, they mention localized/distributed as well as heirarchical/heterarchical. While such 'broad divisions make no sense' to them, one can argue that they do exist and run so deep as to be worth studying in their own right[16].

4. Parallelism, Connectivity and Feedback

It is important to make clear distinctions between such features. One can have a highly parallel system with no connectivity and a highly connected system with no feedback. Further, all of them are supposed to improve the robustness of a computer or control system. For instance, adaptive control, considered in its broadest sense, is supposed to extend the bandwidth of conventional linear control and deal with changing, uncertain environments in a timely, reliable way. This means providing parallel acquisition, analysis and control which implies both connectivity and multiloop feedback. Currently, there is a growing degree of parallelism but very restricted connectivity and feedback. As a result, most artificial topologies don't correspond at all closely to real NN i.e. they neither simulate nor model them for reasons as indicated in Fig. 2. Indeed, the brain is truly different from a conventional computer in these respects.

"Every medical student is aware that information is transmitted in the central nervous system through a succession of neurons, one after another"[10]. Also, everybody knows they have about 10^{10} neurons – give or take and that their 'switching times' are 10^{-3} seconds or so. An immediate question is why a large computer with cycle times of order 10^{-9} seconds and 10^9 transistors can't do even easy pattern recognition and associated learning in comparable times? This is where connectivity and von Neumann bottlenecks enter. While computers may have a billion transistors, only a percent or so are busy at any time because of the memory/processor bottleneck. This worsens as systems grow larger or get more data intensive(cellular automata simulations, I/O bottlenecks, etc.). But it is parallel acquisition and processing of the non-trivial type that makes the difference.

4.1 Importance of Connectivity – The 'Rule of 100'

Neurons fire in ≈1 ms and 'decisions' take about 100 steps e.g. response times take ≈100 ms[11] for perception, analysis and response. If we cease functioning due to loss of decision capacity e.g. control of various functions, then for 10^{10} neurons in the brain and a loss rate of $\approx 3\times10^3$ per day(2 per minute) we have a life expectancy of 100 years assuming very high connectivity. Assuming more initial neurons allows more units of 100 with differing roles and locations. Thus, while we may not use our neurons effectively we definitely don't have more than enough!

4.2 Graph Theory Representation - Undirected Graphs

A connected graph G is composed of sets of links and vertices $\{\mathcal{L}, \mathcal{V}\}$ which can be characterized by the number of vertices $\{n_\ell\}$ each having ℓ legs. The links or edges are the axons or dendrites and the vertices are the cell body or computation unit. The total number of links is $2N = \sum \ell \cdot n_\ell$. In a real neuron, the number of links per vertex can be as high as $10^4 - 10^5$ which implies very high connectivity. The number of loops is:

$$L(G) = 1 + \frac{1}{2}\sum(\ell - 2)n_\ell \,. \qquad (1)$$

This is the number of possible feedback loops that don't include nested loops. Connectivity is defined by the minimum cardinality of either the set of nodes $\kappa(G)$ which separate a graph into subgraphs or set of links $\lambda(G)$ which separate a node from the graph. These integers are a measure of a graph's node and link vulnerability so we want to make them as large as practicable. Complexity can be taken as the minimum cardinality of vertices $\chi(G)$ connecting input/output terminals e.g. the shortest path. Reliability relates to the number of such paths or spanning trees while cost can be taken as proportional to a combination of the total number of links and nodes[16].

4.3 Reliability

A particularly useful expression[12] for the reliability of a graph G was given in the context of electronic networks:

$$\mathcal{R}(G) = p_i\mathcal{R}(G * i) + (1 - p_i)\mathcal{R}(G - i) \qquad (2)$$

where i is any link and p_i its reliability. $G*i$ and G-i are graphs with i contracted and deleted. Full dots in Fig. 3 are 'terminals' that must communicate and R is the probability of doing so:

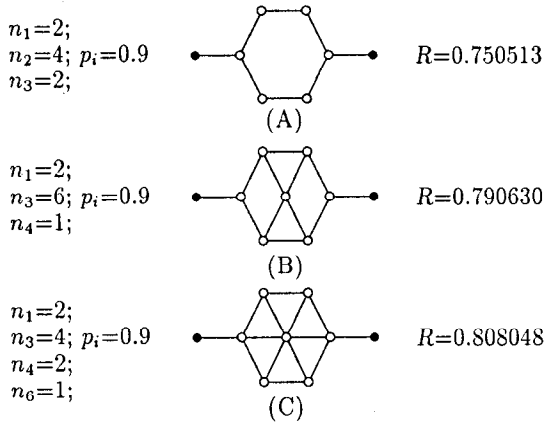

Figure 3: Some Complicated Series-Parallel Graphs.

Such graphs can represent any number of possibilities e.g. the accelerator complex we want to control or one of its subsystems. The graphs in Fig. 3 are dominated by their serial links which could represent ion sources(easily made redundant) or the two opposing linacs of a linear collider. Here we use them as highly linearized versions of NN. All of these graphs have low connectivity and gain little in reliability for significant increases in cost. Figure 4 shows a graph with similar χ's as Fig. 3 but with higher reliabilities. The amplitude of the leading order term in the polynomial for $R(G)$ gives the number of different spanning trees having the same complexity $\chi(G)$. For $\chi = 5$ in Fig's. 3A&B this is 2 and for the output terminal in the upper righthand corner of Fig. 4 it is 10.

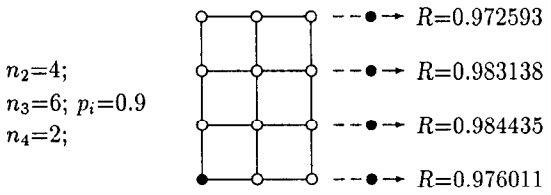

Figure 4: Reliability versus Output Terminal for a 3×4 Net.

One can make such graphs arbitrarily reliable but not perfect. Many are impressed with anything that fails once in every 10^6 trys but in LCS this is often inadequate e.g. it's only about a second's worth of turns in the storage rings of Fig. 1 or 1 ms's operation of a high-speed IC element in a mainframe.

5. Computation and Control with Neural Nets

What should one use for an artificial neuron and how should they be connected to make NN[13]? A McCulloch-Pitts formal neuron[5] can be represented by Rosenblatt perceptrons[14] or Widrow/Hoff adaptive linear neurons (Adalines)[15]. In Fig. 5 there are outputs $x_{i=1,2}$ from the axons of two neurons in the preceding layer which feed synapses ω_{ij} and the cell through dendrites. This input is called a link and the cell body, which does the computation and sends it out on axon j according to some transform or activation function f_j, is a node. This is the simplest example of a learning element from which more complex NN or automatons can be constructed. Any Boolean function and any computation can be done with a net of such elements.

They often have an adjustable bias for fixing thresholds that is equivalent to a weight ω with fixed input. Such neurons are often combined into sequential, layered, feedforward nets without self-excitation loops or feedback i.e. $\omega_{jj} = \omega_{ji} = 0$. The number of states are finite and usually discrete with binary values (0,1) or (± 1) but they can also be analog functions – especially when they're software neurons and nets on PC's. The output y_j is the weighted, linear sum of inputs:

$$y_{j(t+1)} = f_j(x_i) = \sum \omega_{ij} x_{i(t)} + b_j \quad . \tag{3}$$

The weights ω_{ij} (connectivity matrix) of a net fix the couplings between the j-th neuron and a preceding one i with their different layers distinguished by x and y. Weights and states(e.g. ± 1) are the knowledge base. Weights occur on inputs because of possible ambiguities and the assumed feedforward or directed character of the graph. When f_j is a step function $y_j \rightarrow sgn(y_j)$.

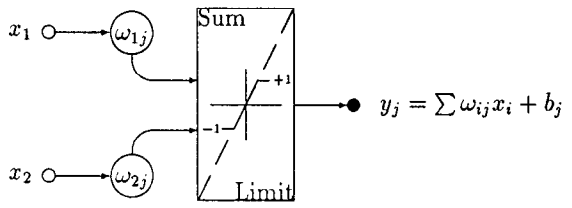

Figure 5: Two-Input, One-Output Feedforward Perceptron.

A classic test for such a device is whether it can solve the XOR problem[8, 6]. To obtain all 2^{2^n} possibilities for n inputs, it is necessary to have nonlinear inputs *or* hidden neurons e.g. Fig. 6 'shows' how an AND neuron, embedded in a single hidden layer, functions to give XOR. In this way we can compute any Boolean function, do binary arithmetic or any computation.

The Back Propagation Method [6] has a representational or input layer for conditioning and fanout, some hidden layers and an output or decisional layer. There is no feedback, which would make the graph cyclic, nor linkage sideways. For a monotonically increasing function like Fermi-Dirac for activation, the weights can be adjusted by a gradient method[6] that generally finds solutions given enough neurons.

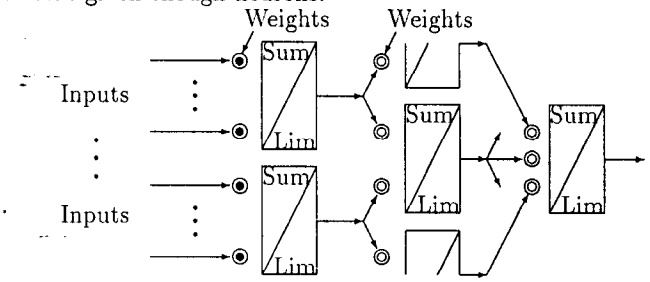

Figure 6: Schematic, General Neural Net (for XOR).

Although Back-Propagation tends to be slow[6] and often hard to understand physically when not set up carefully, it is easy to use and appears capable of computing any function with one hidden layer[17]. However, as indicated in Sect. 4.1, real neural systems take about 100 steps for many complex, real-time tasks and it is hard to see how they could learn in this way.

The Hopfield model[18] is easy to understand physically with ω_{ij} a symmetric matrix i.e. the Hebb learning algorithm[6] seems natural and more importantly, a Lyapunov function exists which guarantees that all attractors in the difference equations [3] are fixed points so that stable, associative memory is possible. Extension to adaptive control follows from models such as two-layer, bidirectional associative memory[19] e.g. from Eq. [3]

$$\begin{aligned} \dot{x}_i &= -x_i + \sum \omega_{ij} y_j(t) + a_i \\ \dot{y}_j &= -y_j + \sum \omega_{ij} x_i(t) + b_j \end{aligned} \tag{4}$$

These are ordinary, linear differential equations until one imposes a step function or sigmoid on the sum(as done above) or on the individual activations u_i. In all cases they retain stable control properties as shown by Cohen and Grossberg[20] for fairly general nonlinear equations:

$$\dot{u}_i = \alpha_i(u_i)[\beta_i(u_i) + \sum \omega_{ij} \delta_j(u_j)] \tag{5}$$

where α, β and δ are nonlinear functions of a node's activation u_i and δ is a monotonically increasing function for stability. These are the dynamical equations for updating the activation and the learning rule can be written in various ways related to Hebb's algorithm as:

$$\dot{\omega}_{ij} = \omega_{ij} + \mathcal{F}(\omega_{ij}, u_i, u_j) \tag{6}$$

The Maxwell-Lorentz control problem for electron optics involves nonlinear equations where the $\mathcal{EM}$ fields are the control elements and the particle parameters are the training variables for feedback and optimization. Writing the Lorentz equation in spatial components gives six simultaneous, first-order equations with the substitutions $\dot{x} = u,\ \dot{y} = v,\ \dot{z} = w$:

$$\begin{aligned} \dot{u} &= k(E_x - \dot{\gamma} u + B_z v - B_y w) \\ \dot{v} &= k(E_y - \dot{\gamma} v + B_x w - B_z u) \\ \dot{w} &= k(E_z - \dot{\gamma} w + B_y u - B_x v) \end{aligned} \tag{7}$$

$\vec{E}$ and $\vec{B}$ are the fields, $k = q/\gamma m$ is the charge-to-mass ratio (energy dependent with stochastic variations) and $\dot{\gamma} = \dot{E}/E$. These are **all** local functions of the position $\vec{r}$.

To solve such equations, the initial values of x,y,z,u,v,w and the energy are necessary as well as the field functions. The computer solution, throughout the region of interest, can be used as reference for comparison to feedback measurements in the least squares optimization step of the control cycle[21]. Also, general purpose magnets are available for control purposes[22].

Unfortunately, the Eq's. [7] don't quite match Eq's. [4] or [5]. The weight matrix is not symmetric. However, reducing the problem to one dimension e.g. by transforming to cylindrical coordinates for a betatron(or storage ring) provides sufficient conditions for a stable fixed point without violating Maxwell's equations. We can use the two layer BAM of Eq. [4] for turn-to-turn simulation which is stable in the linear approximation i.e. once trained the ring can, at least in principle, remember how to operate. The question is how to extend and relate this approach to control e.g. add realistic perturbations.

The differential equations for betatron oscillations from Eq.[7] have a number of control applications:

$$\ddot{q}_i + \alpha_q \dot{q}_i + K(q_i) = F(t) \quad . \tag{8}$$

K is the focusing (possibly nonlinear), F some externally applied forces and $\alpha_q \propto \dot{\gamma}$ is assumed to vary slowly. F can be a Fourier series to cancel global error harmonics or a noise source which mirrors one measured with the beam. The subscript specifies bunch number or measurements around the ring for least squares.

With $\dot{q} = p$ and H_o the time independent Hamiltonian for the undamped, conservative system one has:

$$\dot{q}_i = \frac{\partial H_o}{\partial p_i} \quad ; \qquad \dot{p}_i = -\frac{\partial H_o}{\partial q_i} - \alpha_q p_i + F(t) \quad . \tag{9}$$

We can add various types of stochastic effects to the equations and study their effect on stability and emittance and how to minimize them. Without such effects, the time evolution is well defined and we can set the system up and study perturbations to the regions of attraction with damping strength α, nonlinearities in K and convergence of different learning algorithms and how they influence feedback. Multibunch interaction and corrective feedback studies are a natural extension with a rich spectrum of attractors and strategies to populate them.

6. Some Possibilities

I have used 'Turing-machine' to mean any machine capable of general computation rather than a generic, serial, programming approach to computation. NN offer the possibility of fast, reliable, real-time Turing machines that combine computation and control needed for increasingly complex environments. Today, they are usually multiple layers with sequential fanout and feedforward that are sparsely connected and may not adjust weights or change the numbers of layers or neurons because they use only fixed logic or have been trained for speed and reliability. These are deterministic, pattern recognition and mapping problems based on associative memory models for noisy data. Autonomous controllers can function in this way with incomplete or mixed information as in complex hybrids involving logic as well as analog and digital inputs. They can also change naturally when the problem changes due to bad input channels or new ones.

Just as fast, inexpensive micros based on serial processing have made AI practical, the advent of single-chip parallel processors can be expected to make large scale PDP practical. However, parallelism and connectivity can help at many levels. Simply duplicating existing serial logic (trivial parallelism) is only a limited step toward the next generation machine or of solving *the* von Neumann bottleneck. PC's have made it easy to ignore hardware implications such as how to implement input nonlinearities e.g. do binary multiplication with neurons or the related problem of hardware least squares at the output. Equally important for real-time applications is the potential integration of signal sensing, processing and feedback(ISP) on a single chip.

For illustration, consider multiplying two n-bit numbers. One can *deduce* a result for a parallel multiply assuming that n^2 AND's are done in one cycle followed by a sequence of adds with $n(n-1)$ full adders for a total of 2n cycles. With enough connectivity and numbers of neurons it is possible to get the answer in m+2 cycles with m hidden layers independent of n or just 3 cycles with combinational logic gates e.g. the $2n$-bit result $z_i 2^i$ has $z_o = x_o \cdot y_o$, $z_1 = x_1 \cdot y_o \cdot (\overline{y_1 \cdot x_o}) + x_o \cdot y_1 \cdot (\overline{y_o \cdot x_1})$, ... which has many interesting implications as indicated in Sect's. 4 & 5.

Techniques for design and mass production of microstructures exist which could be applied to a virtually unlimited variety of problems. Some relevant examples include chips for high-resolution mechanical alignment of components, beam position sensing and field probe arrays as well as temperature, pressure, vacuum or flow sensors with unprecedented speeds, accuracy and reliability. Intelligent, solid-state devices with no moving parts to wear out, break or stick which can be pretested and debugged through millions of cycles will ultimately be cheaper and more reliable and could mean the success or failure of large complex systems such as the next generation of accelerators.

Intelligent controllers in extreme environments can save a lot of money. Effects such as aging and annealing could be dealt with in various ways because such devices can be self-monitoring. Further, small size and low cost allow more monitoring and redundancy and could be expected to minimize radiation effects, an added constraint for many accelerator applications.

Because there are a number of problems that are common to different laboratories, it seems practical to try to explore them together to find where joint projects are possible. While there are probably broader concerns encompassing other areas and agencies, this one is potentially important and lucrative enough to interest private industry in collaborating as well. The American Scientist expressed the situation well: "The media hype, and that of some entrepreneurs, is not so much wrong as premature."

References

[1] Kang G. Shin, Introduction to Issue on Real-Time Systems, IEEE Trans. on Computers, Vol. C-36(1987)901.

[2] J. von Neumann, IAS Memorandum on the Program of the High Speed Computer, Princeton Univ., Nov. 1945.

[3] J. von Neumann, Probabilistic Logics and the Synthesis of Reliable Organisms From Unreliable Components,*Automata Studies*, Edited by C.E.Shannon and J. McCarthy, Annals of Math. Studies, No. 34, Princeton Univ. Press, 1956.

[4] Data was obtained from Intel Corp. and Catalogues of High-Energy Accelerators, Int'l. Conf. on High Energy Accel's.

[5] Warren S. McCulloch and Walter Pitts, A Logical Calculus of the Ideas Immanent in Nervous Activity, Bull. Math. Biophys., Vol. 5(1943)115.

[6] David E. Rumelhart and James L. McClelland(Eds.), *Parallel Distributed Processing–Explorations in the Microstructure of Cognition,* MIT Press, Cambridge, MA(1986).

[7] Marvin L. Minsky, *Computation: Finite and Infinite Machines*, Prentice-Hall, Inc., Englewood Cliffs, NJ(1967).

[8] M.Minsky and S.Papert, *Perceptrons - An Introduction to Computational Geometry*, MIT Press, Cambridge(1988).

[9] Bertrand Russell, *Human Society in Ethics and Politics,* G. Allen & Unwin Ltd., London(1955).

[10] A. C. Guyton, *Basic Human Physiology,* W.H. Freeman and Co., San Francisco, CA, 1971.

[11] J.A. Feldman, Connectionist Models and their Applications, Cog. Sci., Vol. 9(1985)1.

[12] Fred Moskowitz, The Analysis of Redundancy Networks, AIEE Trans on Comm. Elect., Vol. 35(1958)627. See also: L.B. Page and J.E. Perry, IEEE Trans on Rel., Vol. 37(1988)259.

[13] A fundamental question: What is the simplest primitive required to simulate the brain in the Turing-Church sense?

[14] Frank Rosenblatt, *Principles of Neurodynamics*, Spartan Books, New York(1962).

[15] B. Widrow and M.E. Hoff, Adaptive Switching Circuits, IRE WESCON, Part 4(1960)96.

[16] J.E. Spencer, Accelerator Diagnosis and Control by Neural Nets, IEEE Particle Accelerator Conf., Chicago(1989).

[17] Robert Hecht-Nielsen, Kolmogorov's Mapping NN Existence Theorem, IEEE Conf. on NN, Cat. #87TH0191-7(1987).

[18] J.J. Hopfield, Neural Networks and Physical Systems with Emergent Collective Computational Abilities, Proc. Nat. Acad. Sci., Vol 79(1982)2554.

[19] Bart Kosko, Bidirectional Associative Memories, IEEE Trans. Syst., Man and Cyber.,Vol. 18(1988)49.

[20] M.A. Cohen and S. Grossberg, Absolute Stability of Global Pattern Formation and Parallel Memory Storage by NN, IEEE Trans. Syst., Man and Cyber.,Vol. 13(1983)815.

[21] B. Widrow and S.D. Stearns, *Adaptive Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ(1985).