SLAC-PUB-5035
4 October 1989
(A)

Computation and Control with Neural Nets*

A. Corneliusen†, P. Terdal†, T. Knight and J. Spencer
Stanford Linear Accelerator Center, Stanford University, Stanford, California 94309

Abstract

As energies have increased exponentially with time so have the size and complexity of accelerators and control systems. NN may offer the kinds of improvements in computation and control that are needed to maintain acceptable functionality. For control their associative characteristics could provide signal conversion or data translation. Because they can do any computation such as least squares, they can close feedback loops autonomously to provide intelligent control at the point of action rather than at a central location that requires transfers, conversions, hand-shaking and other costly repetitions like input protection. Both computation and control can be integrated on a single chip, printed circuit or an optical equivalent that is also inherently faster through full parallel operation. For such reasons one expects lower costs and better results. Such systems could be optimized by integrating sensor and signal processing functions. Distributed nets of such hardware could communicate and provide global monitoring and multiprocessing in various ways e.g. via token, -slotted or parallel rings(or Steiner trees) for compatibility with existing systems. Problems and advantages of this approach such as an optimal, real-time Turing machine are discussed. Simple examples are simulated and hardware implemented using discrete elements that demonstrate some basic characteristics of learning and parallelism. Future 'microprocessors' are predicted and requested on this basis.

Invited *talk* presented at the Accelerator Control: International Conference on Accelerator and *Large Experimental* Physics *Control Systems*, Vancouver, Canada., October *30–November* 3, 1989.

*This work was supported by U.S. Department of Energy contract DE-AC03-76SF00515

†Summer students at SLAC from Avila and Reed Colleges, respectively.

1. Introduction

Except for space and time limitations, the conventional von Neumann computer can simulate any physically realizable process of interest here. However, since physical processes compute much faster, there has been a consistent drive toward increasing time and space complexity in computers. At the same time, there is no question that NN are possible that could do any conventional computer calculation[l, 2]. Further, by integrating memory and processor in a fully parallel way, NN could provide a Turing-equivalent machine that eliminates the von Neumann bottleneck[3]. Failing this, techniques like hierarchical memory(cache), vector processing, pipelining and multiprocessing are needed to minimize the associated bottlenecks that arise in different implementations of the von Neumann architecture. While such developments may continue, Fig. 1 shows an exponential growth in integrated circuit complexity every bit as rapid as that observed for accelerators[3, 4]. This is both good and bad. Good because larger rings need better microprocessors in increasing numbers to operate efficiently and they in turn can produce the kinds of beams that are needed to make higher density chips. Bad because the growing complexity ultimately conspires to worsen the effective throughput and reliability through a more ill defined, hard to control problem environment.

While it is unlikely that storage rings will continue increasing at this rate[5], one could see a 64-bit, 5-million transistor micro by 1993 through a doubling of word size every 6-8 years[6] as well as a 100-million micro and one gigabit RAM chip around 2000 through doubling the density every l-2 years. The success of such predictions depends more on impetus than technical problems like connectivity or power dissipation, even though these are considerable. While some may prefer a 64-bit word, the majority probably prefer more memory at all levels of the heirarchy. Thus, the real questions are not when but what and how. The *what* and *how* are closely related and this is where NN appear to be important. As a metaphor for large complex systems(LCS) that operate in a highly connected and massively parallel way, they provide an ideal model for the architecture needed to make and perhaps justify a billion-transistor chip because more transistors don't necessarily mean greater speed or reliability which are both important for computers and real-time control[3].

Our approach to optimal control systems for LCS is similar to that for large computers because this is what they are in many respects when we generalize our ideas of computer storage to include real-time data. One negative implication is that they have the same indigenous bottlenecks unless concurrency is exploited - at least to the extent that concurrent hardware can be kept running in a productive way. Also, because such hardware should be kept as close as practicable to the sensors and data sources that feed it, as well as to the actuators and systems it controls, this implies distributed nets of intelligent controllers – **not** general purpose computers or micros. Improvements in IC technology such as shown in Fig. 1 make this feasible i.e. possible at reasonable costs.

Perhaps the first and in many ways the most important bottleneck is a kind of mindset that preordains certain approaches. Due to its impressive longevity, the von Neumann computer model and its process-at-a-time sequential logic is a remarkably pervasive example. As a result we begin by comparing current manifestations of AI and NN. Next, we consider combinational logic functions, which require no memory in the sense that the output is solely a function of the inputs. This provides an alternative model for both hardware and software that avoids many complexities and bottlenecks of the von Neumann model[7]. Multiplication as a basic computation process demonstrates different alternatives and architectures e.g. the difference between calculating a result or remembering it. Back Propagation[8] and Adaptive Resonance Theory[9] have been used to simulate such processes. They illustrate some basic ideas like delearning, learning, unlearning and generalization as well as some weaknesses and possible improvements.

The goal is a method that allows computation, comparison and optimization for LCS even though they may be 'unpredictable' in some important way such as long term stability. This implies only that the design, simulation and control are inseparable because feedback, when fast enough and clean enough, 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. Combinational NN provide a model that is faster, less expensive and ultimately less complex than present alternatives.

2. Relation Between AI and NN - Expert Systems

A recent survey of AI software revenues(Computerworld/Sept.4) showed the two lowest categories to be AI(5%) and NN(11%) with Expert Systems and Applications the highest with 70% of the market. One wonders if this is due to the growth of complexity or efficacy of expert systems and whether it reflects the relative merits?

Can 'Expert systems capture human and other expertise and make it available, upon demand, and under almost all circumstances?' Can any system? This implies that response times can be accelerated and made more reliable using stress-free micros and state-of-the-art methods. A few questions that need to be asked concern:

- Accurate Decision/Control Systems for Changing Data and Environments,
- Maintaining Efficiencies for Ever Increasing Amounts of Data,
- Acquiring Expert Knowledge to Model Decision Making Processes, or
- Acquiring Expert Knowledge Where It May Not Exist or
- Involves Proprietary or Confidential Information,
- Resolving Apparent or Real Differences between Experts,
- Cost and Time of Incorporating into Existing Systems and
- Other Fundamental Problems and Dichotomies.

The time and difficulty of implementing expert or other algorithmic systems as well as problems that arise when the underlying systems or data change can be significant. -Experts may not know or be willing to admit how they actually do some jobs. The 'expert' is a unique NN that may be difficult to reconcile with others – either experts or interrogators. In fact, it may require an artificial NN(ANN) to learn via training rather than with expert system software.

Many distinctions have been made between AI and NN. While they share the goal of simulating intelligent behavior their means differ widely. Many are directly related to the available hardware, the brain and conventional digital computer(CC). Like the CC, expert systems are stepwise, heirarchical, sequential processes. The basic distinctions and dichotomies are summarized in Fig. 2. While we have discussed this before[10, 3] we reference an interesting comment by Freud[ll].

3. Description of NN for Computation and Control

The acronym NN originated from simple models of the neuron and its connectivity in the central nervous system. The acronym PDP for parallel distributed processing is also used[8]. Being equivalent to any finite state Turing machine[1], they provide an alternative to the conventional von Neumann computer. Although McCulloch and Pitts[l] appear to have been the first to use mathematical logic beyond propositional calculus, there is still no satisfactory way to characterize finite automatons that optimizes convergence, minimizes complexity or guarantees that there will be no -spurious results. To the extent that the Turing-Church hypothesis is valid, so is the original criticism of Minsky and Papert[12] if it argues against NN for general purpose systems. However, one can argue in favor of NN for computational problems with high algorithmic complexity e.g. where a satisfactory algorithm isn't known such as pattern recognition or for complex adaptive control problems.

A McCulloch-Pitts neuron[l] can be represented by Rosenblatt perceptrons[13] or Widrow/Hoff adaptive linear neurons (Adalines)[14]. In Fig. 3 there are inputs $x_{i=1,2}$ or outputs from the axons of two neurons in a preceding layer which feed synapses ω_{ij} and the cell through dendrites. Such inputs are called links or edges 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. Such elements are often given an adjustable bias for fixing thresholds that is equivalent to a self-excitation ω_{jj} for fixed unit input that can latch or inhibit. This is the simplest example of a learning element from which more complex NN or automatons can be constructed.

For digital systems with n binary inputs (0,1) or (fl) there are 2" possible input states. In the (0,1) basis, weights are excitatory or inhibiting depending on the sign of w. The output y_j is some function of the weighted, linear sum of inputs i_j:

$$y_j(t+1) = f_j(i_j) \text{ where } i_j = \sum_{0}^{n-1} \omega_{ij} x_i(t) + b_j \ . \qquad (1)$$

f is the transfer function which maps x to y and the network problem is to obtain a desired mapping from a limited training set that does not produce spurious results when extended. The connectivity matrix ω_{ij} generally allows n^2 interactions or couplings between n neurons. This is reduced to O(n) i.e. $\leq 2(n-1)$ in a feedforward or forward directed graph.

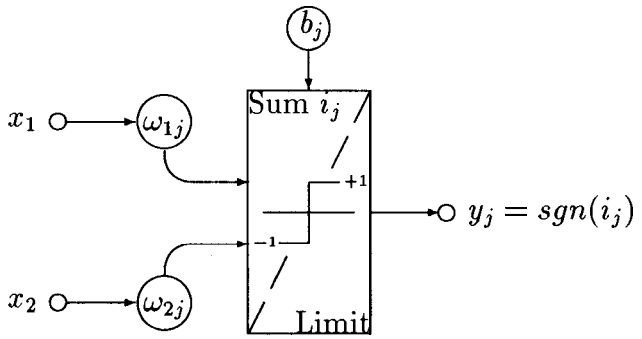


Figure 3: Two-Input, One-Output Feedforward Neuron.

A classic test for such a device is whether it can solve the XOR problem[12, 8] which is not linearly separable. To obtain all such states from the 2^{2^n} possible, it is necessary to have nonlinear inputs *or* another layer of 'hidden' neurons e.g. Fig. 4 shows how an AND neuron, embedded in a single hidden layer, functions to give XOR. It is not possible, by any method such as Back-Propagation or ART1 with only two layers and linear inputs such as shown in Fig. 3 to compute states like XOR. A possible exception occurs if cross links within the input layer are allowed; timing considerations make this equivalent to another layer although such a configuration would have fewer neurons for comparable power.

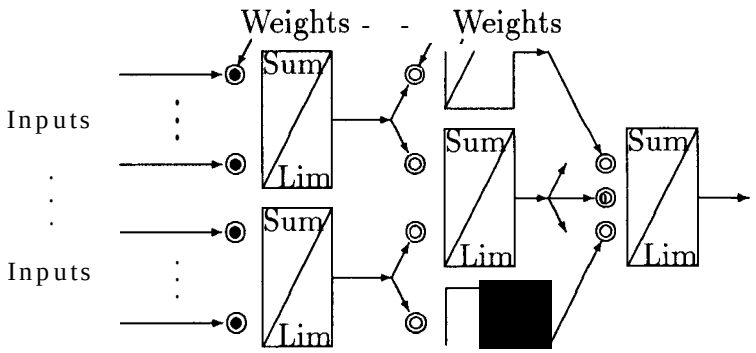


Figure 4: Schematic, General Neural Net with XOR.

A general purpose controller providing both feedforward and feedback elements might look as shown in Fig. 5. In feedforward mode, it tracks input commands in the presence of measureable or predictable perturbations of plant characteristics such as near term history, temperature and the like. The feedback controller, like the others, can be activated with a stored state or for training or deactivated by the 'Desired Response' input. Feedback provides stability, reduces sensitivity and can be used to augment other NN such as the perturbation nets. It's also possible to have an overseer net for such control. Adaptive signal processing is treated in Ref. [15].

4. Some Examples

Among many possible questions one might raise is why they should consider this approach as opposed to increasingly powerful microprocessors that make distributed control based on expert systems possible? Some reasons in favor of the micro based expert systems approach are their availability and our growing experience with them. Some reasons against are the speed and vulnerability of general purpose systems. NN could supply superior front end control and monitoring while micros supply overall direction and feedback of performance data or abnormal or out-of-tolerance characteristics.

Another question is what needs to be done to new or conventional systems to simplify their subsequent extension to include NN techniques? How might one incorporate or integrate these into existing systems without disruption or necessarily being forced to use them. Since most 'NN' are software simulations this is simpler than for hardware but many of the advantages are also sacrificed. For systems integrated on a single PC-board such as data acquisition and analysis for beam position monitors it seems a reasonable beginning – especially when such measurements are critical to the feedback and control process.

With increasing accuracy, it becomes increasingly difficult to specify or qualify conventional hardware because it may be changing with time and conditions. In many cases there is no hardware yet available that meets the specifications for the next generation system. Integrating them with something that has the capacity to undergo adaptive learning is a new way of improving the accuracy of old devices and designing new ones.

We begin by looking at linearly separable logic functions to test and improve methods like back-propagation because such methods provide a basic control system model. It is therefore interesting to use them to simulate the basic components of the system including multiplication which is used throughout the process.

4.1 Learning with Back-Propagation(BP)

Back Propagation [8] is probably the most popular method for supervised learning now being used. Typically, it has a representational or input layer for conditioning

and fanout, some intermediate or hidden layers and an output or decisional layer. There is no feedback, which could make the graph cyclic, nor linkage sideways. Time is measured in discrete units[16]: the output state $\{y_j(t+\mathrm{m})\}$ for an m-layer system corresponds to the input state {s;(t)}. For a monotonically increasing function like Fermi-Dirac for activation, the weights can be adjusted by a gradient method[8] that generally finds solutions given enough neurons. Although Back-Propagation tends to be slow[8] 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[3]. -Although this provides the highest speed it would also have the worst reliability.

In Fig. 6 we show some results for the minimum logical AND using this method. In Case (A), all possible input patterns were used together with a fast learning rate. While TSS, the total sum of squares, decreases monotonically there was only slow, partial learning or what we will call delearning for pattern P11 initially. This is shown by both the output activation A_{11} and the partial sum of squares PSS_{11} for this pattern. The initial values for the two input weights were randomly selected and left unconstrained throughout the calculation. The greatest learning rates, defined by $\dot{A}_{11}$, $\dot{P}$ and $\dot{T}$, occur when we = ω_1 on the 6th iteration. The maximum rate of delearning occurs near iteration 12 where the difference between the w's reaches a local maximum.

In another case(not shown) with everything identical to Case (A) except that the weights were constrained equal, we observed comparable results but more emphatic -oscillations.- In Case (B), weights were constrained equal and pattern P00 was replaced with P11 so that target activations(T_{pj}) of 0 and 1 were equally weighted. This reduces delearning in all patterns to negligible amounts and provides an example of generalization because it responds correctly for pattern P00 without being trained. While there is still some modulation, this would be quite noise immune. This problem is eliminated by constraining the bias to be $b_2=1.5\omega$ as shown by the dot-dashed lines for A_{11} and PSS.

Finally, we show a three-input, four-neuron case in Fig. 7 corresponding to Case (A) above i.e. all 8 possible input patterns were used with equal weighting. The first few iterations are comparable to Case (A) but more emphatic so that by the 7th or

8th cycle, A_{111} appeared to be stably trapped at 0 with PSS_{111} totally dominating TSS. This is a case of rapid but partial learning with one error in eight. This situation remained unchanged for the next 8000 iterations. The characteristics then become similar to previous cases. The only exception is the oscillation in TSS. This can happen near a trend reversal that continues improving P11 $\bar{1}$ at the expense of 7 other patterns when weights are changed according to[8]:

$$\Delta\omega_{ij}(n + 1) = \alpha\Delta\omega_{ij}(n) + l_r \sum_{P}(\delta_{pj}x_{pi}) , \qquad (2)$$

with high momentum $\alpha = 0.9$. δ_{pj} is defined as $(T_{pj}\text{-}A_{pj})A_{pj}(1\text{-}A_{pj})$. Decreasing both the learning rate l_r and α generally smooths the results but causes a proportionate increase in the number of iterations n.

In another case(not shown) with random, unconstrained input weights but only four patterns P110, P101, PO11 and Plll(weighted 3-fold), A_{111} converged to ≥ 0.9 in about 22 iterations quite similar to Case (A) with 3 neurons which also had 4 patterns but in this case, 4 other patterns were predicted correctly. In all cases, the solutions satisfy the relations:

$$\omega_{ij} = \omega_{kj} \text{ for i, } k < j \text{ and } b_j = (\frac{1}{2} - n)\omega_{ij} , \qquad (3)$$

where n is the number of inputs. These are just the slope and intercept relations for an n-dimensional hyperplane separating P1· · ·1 from all other permutations. The dot-dashed lines in Fig. 6B show this case for the same values of 1, and α.

The results demonstrate a number of characteristics reminescent of human learning such as-the difficulty of unlearning something or discriminating essential properties and symmetries. We know that real neural systems take about 100 steps to execute many complex, real-time tasks[3]. For this example, it should take only two steps or a couple of milliseconds once we are properly trained. However, it seems to take an inordinately long time to learn and it is hard to see how we learn in this way. A more relevant question here is how we should set up an artificial NN that implements BP and minimizes cycle time. It is hard to avoid multiplication in such a net and, depending on the specific implementation e.g. analog/digital, it also seems hard to avoid multi-input, combinational AND/NAND logic.

4.2 Parallel Multiplication

Multiplication is a natural extension of the above examples. We considered it a good benchmark for comparing serial and parallel computations[3] as well as for comparing different network implementations. However, many processes are multiplication intensive such as DSP, differential equation solving and graphics; In adaptive signal processing(ASP) we have to repeatedly compute the weighted sum of inputs as well as do some kind of least squares optimization.

Although one could use associative memory to remember an answer rather than calculate it, this would be very inefficient and no justification for gigabit memories. Most schemes follow the basic way we first learn to multiply and so provide a good, literal example of a Turing machine. The largest number, called the multiplicand with N_1 digits, is put on top and the multiplier N_2 below. One forms the $N_1 \times N_2$ partial products and then forms the N_1+N_2 digit result from N_1+N_2-1 sums and carries. Assuming that the N^2 products(for $N_1=N_2=N$) are all done initially in parallel, the time is proportional to 2N for the sums which are done sequentially(because of the carries) with N(N-1) full adders[17]. The procedure is the same regardless of the number system.

The question of how the partial products are done is important because with enough combinational logic we get the result independent of the usual sequence of sums and carries. One might try to reduce the $2N-1$ sum cycles to order $\log_2 N$ via parallel adders that sum partial products of the same order with a binary tree structure. While this is useful for large N, an alternative is a *fully* parallel multiply. Either way, connectivity could be a significant problem for large enough N.

One can deduce a result for a fully parallel multiply of two N-bit numbers that takes only 3-layers independent of N assuming the appropriate AND's are formed by the intermediate or hidden layer[3]. The 2N-bit product $\sum z_i 2^i$ is:

$$z_0 = x_0 \cdot y_0 \equiv x_0 y_0$$

$$z_1 = x_1 y_0 x_0 y_1 + x_0 y_1 x_1 y_0 = x_1 y_0(\bar{x}_0 + \bar{y}_1) + x_0 y_1(\bar{x}_1 + \bar{y}_0) \qquad (4)$$

$$z_2 = x_2 y_0[\overline{x_0 y_2 + x_1 y_1 + (x_0 y_1)(y_0 x_1)}] + \cdots = x_2 y_0[\bar{x}_0\bar{x}_1 + x_0\bar{\bar{y}}_1 + y_1\bar{\bar{y}}_2 + x_0\bar{y}_2] +$$
$$x_1 y_1[\bar{\bar{x}}_0\bar{x}_2 + x_0\bar{\bar{y}}_0 + \bar{y}_0\bar{y}_2] + x_0 y_2[\bar{x}_1\bar{x}_2 + \bar{\bar{x}}_1 y_0 + y_0\bar{\bar{y}}_1 + y_0\bar{x}_2]$$

The dot implies a logical multiply or AND while the plus sign implies a logical sum i.e. an inclusive OR. Also, we consider only the integral part($i \geq 0$) for simplicity. Each successive order incorporates the previous variables to that order. If the input layer provides $\{x,\overline{x},y,\overline{y}\}$ and the hidden layer the multi-input AND's, then the output layer OR's the proper combinations into $\{z\}$. We can also form the pairwise AND's and NAND's in the hidden layer, combine them in the 'output' layer and form the OR as a literal sum of signals from the proper 'output' neurons. This is the case shown in Fig. 8 although, in principle, it takes another layer.

Fully parallel schemes generally imply 3-dimensions which raises many questions e.g. VLSI is 2-D and this limits the degree of parallelism and the resources it can bring to a problem in any fixed time period. However, 3-dimensions and high speeds can cause many unforeseen problems so we decided to test the schema. Fig. 8 shows a <2.5 ns multiply going from the output of the TTL-to-ECL to the output of the last AND gate with a spacious PC layout. Since adequate pulsers and probes weren't available, we didn't try to improve the pulse shape. We estimate it is possible to reduce this number to picoseconds depending on the switching times achievable.

Although the i486 chip includes RISC features, cache and a math coprocessor, it's integer multiply between a register and accumulator takes >460 ns for the 25 MHz i486. In big, general purpose computers like the IBM 3090 at SLAC in scalar mode, it takes 5 cycles while in vector mode it could take longer because it takes 30 cycles to start the pipeline(3090/120E). Fairchild also has a 2x8 Recode Multiplier which provides a comparable speed but the timing compounds with N e.g. a 16×16 takes **24** ns. Honeywell has used a heterojunction or superlattice structure of GaAs to obtain a 1.8 ns, 5x5-bit multiplier.

4.2.1 Multiply Simulations

With insight into such a multi-layer problem, it is interesting to try to use NN to simulate the design of VLSI and other circuits. This was explored only briefly before finding sufficient flaws to lay it aside. For instance, we couldn't use constraints such as Eq.[3] in a natural way nor use more general minimization constraints without more software work which is in progress. The goal is a method with the power to generalize results such as represented by Eq.[4] where a single AND for the binary

function z_o predicts $2^{2(N-1)}$ nonzero results for N×N multiplication. We note that the problem can't be done in two layers for the same reason as the XOR problem.

4.3 Other Examples

Beam position monitors(BPM's) are a good example of a distributed system that is important, expensive and often inadequate due to noise and error sources that are difficult to measure or unfold from one another. Many of these are a direct result of the sequential nature of the control system that ignores the parallel nature of the problem. The trigger system of a large particle detector which decides whether to readout and store megabits of information on a possible event is another example.

One criticism of fully parallel schemes that often arises is the expense of implementing them due to the growth in the number of components i.e. their spatial complexity and the increased power dissipation this implies. For control applications, this number is relatively small because the required word sizes are often smaller and the number of functions required are much smaller than in a conventional computer.

5. Conclusions and Possibilities

Minsky and Papert have reissued *Perceptrons*[12] recently with a prologue in which they remark: "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 they are often given considerable credit for this, it is fair to say that little hardware has been developed nor some of the theoretical questions resolved. While it is clear that a more analytical framework is needed to study and understand the topology of NN, it is still impressive how well combinational methods work. One might question in reply: Where are the parallel circuits and techniques that could have been in the i486 but may only appear in the i686 – if we're lucky[6].

Using the example of parallel multiplication we have tried to show how one obtains a real-time Turing machine i.e. one that minimizes time requirements if not the space or number of neurons. This is important for both computation and control. Further, since this is provided by an NN which can also provide reliability through high connectivity[3], one can postulate an optimal control system. The con-

straints(correlated) of reliability and spatial complexity demonstrate the importance of minimizing expressions such as Eq.[4] for single (and multi) output functions of many variables. More generally, this is a problem associated with the need for a more analytic, inductive mathematical logic. The efficient use of PLA's in todays VLSI chips is one application but the need to extend such chips to 3-D or increase their fractal dimensionality(> 2) is clear. This also explain the current interest in optical circuits.

Many current problems in NN are directly related to weaknesses observed with some NN algorithms. The Hopfield model[18] is easy to understand physically with ω_{ij} a symmetric matrix. The Hebb learning algorithm[8] seems natural and a Lyapunov function exists which guarantees that all attractors in the difference equations are stable fixed points. Nevertheless, although stable associative memory is possible, this still doesn't preclude spurious states.

Other methods like BP often fail to get the preferred answer or may take an inordinately long time for problems a person can solve by inspection. Because this depends on how we set them up, we have taken a combinational approach that emphasizes symmetry and repetitiveness for its usefulness in circuit design. This is especially important where size and complexity are significant because they influence time response and reliability. The error correcting algorithm needs improvement in a number of ways that make it more immune to a poor choice of training set or network. This often leads to delearning in some patterns initially and can even lead to trapping in semistable or stable states that may not allow unlearning even when new training data are added. The system becomes infexible or unadaptive. This can also worsen the net's capacity to generalize outside the training set. Poor use of constraints and correlations between variables can lead to the same result.

Nevertheless, many interesting characteristics of human learning are demonstrated. It also provides a straightforward way to implement expert systems in cases where the knowledge or data bases exist that requires no direct access to the expert. However, explaining the resulting system's decisions is not so straightforward. This is another symptom of our limited understanding of how we should construct the net itself in terms of the number and distribution of neurons or the impact and interplay

of the training set. The example of multiplication shows the mechanics of sums and carries quite clearly. The feedforward net with combinational logic simplifies the situation but still provides a powerful tool.

Of course, there are many technical problems with implementing NN in hardware. Recurrent training based on a standardization procedure consistent with ones such as used for setting magnets is relevant since it relates to the overall problem of remembering a solution once trained e.g. what happens when one loses power? Are there effects equivalent to hysteresis that require an annealing procedure to forget or wipe out the perturbation and return the system to its initial state? We could also use the terms 'reconditioning' for 'standardization' and 'training' for 'magnetization' as well as 'operant conditioning' for their combination.

Besides technical problems there are the risks of attempting to implement any new approach. An important presupposition that often blocks acceptance of NN as a control procedure is the belief in the existence of well-defined, reproducible standards. In such cases, an important and necessary goal is the establishment of a robust, reproducible control standard. Even though the standard may change, its existence is a fundamental premise for many expert systems that are buttressed by phrases like 'paradise' and 'golden' solutions[19]. In a microscopic sense these don't exist and their probability macroscopically decreases with increasing size and complexity.

Because there are a number of problems that are common to different laboratories, it seems practical to try to explore the possibility of joint projects – especially in the area of hardware. While there are undoubtedly broader concerns that should be explored, this one is lucrative enough to interest private industry in collaborating as well.

Acknowledgements

The authors would like to thank Terry Anderson and the Graphic Arts Group, Don Briggs, Dave Gustavson, Hamid Shoaee, George Spalek, Nancy Spencer and John Tinsman for various contributions to this work.

References

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

[2] 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. See also S.C. Kleene's contribution there.

[3] J.E. Spencer, Real-Time Applications of Neural Nets, IEEE Trans. on Real-Time Computer Applications in Nuclear, Particle and Plasma Sciences, Vol. NS-36-5(1989)1480.

[4] Data was obtained from Intel Corp. and Catalogues of High-Energy Accelerators, Int'l. Conf. on High Energy Accel's. For electron rings, the length in terms of the equivalent radius in mm is plotted assuming the linear density of components and the various data and control signals are roughly constant. Units of cm would be more accurate but would make the graph unnecessarily complicated. We note that both IC's and storage rings were 'invented' in the mid-to-late fifties so their two curves begin near one another and nearly overlay.

[5] The large proton ring SSC(Superconducting Super Collider) proposed for 1995 is more than three times larger than LEP and intrinsically more complex. The exponential growth in energy of proton rings is shown in W.K.H. Panofsky, Phys. Today(1983)34.

[6] Intel's i860 RISC chip **is** a 64-bit microprocessor with 1 million transistors capable of concurrent integer math, floating point addition and multiplication as well as pipelining so that it could provide burst rates of up to 120 million results

a second with a 40 MHz clock. The i586 may well combine this chip with the i486. The i686 could have 50-100 million transistors with parallel processing and NN capabilities in time for the next generation linear collider. A 4-million bit RAM chip for the 386 PS/2 series is also available that has an 80 ns access time.

[7] John Backus, Can Programming be Liberated from the von Neumann Style?, ACM Comm., 21(1978)613.

[8] James L. McClelland and David E. Rumelhart, *Explorations in Parallel Distributed Processing*, The MIT Press, Cambridge, MA(1988).

[9] S. Grossberg, Adaptive Pattern Classification and Universal Recoding, II, Biol. Cyber., 23(1976)187.

[10] J.E. Spencer, Accelerator Diagnosis and Control by NN, IEEE Part. Accel. Conf., Vol. 89CH2389-9(1989).

[11] Sigmund Freud, *A General Introduction to Psychoanalysis*, Boni and Liveright Publishers, New York(1920). "With two of its assertions, psychoanalysis offends the whole world and draws aversion upon itself. One of these assertions offends an intellectual prejudice, the other an aesthetic-moral one. Let us not think too lightly of these prejudices; they are powerful things, remnants of useful, even necessary, developments of mankind. They are retained through powerful affects, and the battle against them is a hard one. The first of these displeasing assertions of psychoanalysis is this, that the psychic processes are in themselves unconscious, and that those which are conscious are merely isolated acts and parts of the total psychic life. Recollect that we are, on the contrary, accustomed to identify the psychic with the conscious... and psychology as the science of the content of consciousness. Indeed, so obvious does this identification seem to us that we consider its slightest contradiction obvious nonsense, . ..and yet I can assure you that by the acceptance of unconscious processes you have paved the way for a decisively new orientation in the world and in science."

[12] M.Minsky and S.Papert, *Perceptrons - An Introduction to Computational Geometry*, MIT Press, Cambridge(l988).

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

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

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

[16] This is a big assumption for parallel schemes. It says that the neuron dominates the delay as opposed to the length of dendrite or wiring contribution. 'Layers' traversed and the corresponding time delays need to be considered in 3-D since they may easily depend on the structure of the net and number of neurons. Connections per second(COPS) can then be used in analogy to FLOPS as a figure of merit.

[17] Shinji Nakamura and Kai-Yu Chu, A Single Chip Parallel Multiplier by MOS Technology, IEEE Trans. on Comp.,Vol. 37(1988)274.

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

[19] T. Higo, H. Shoaee and J. Spencer, Some Applications of AI to the Problems of Accelerator Physics, IEEE Part. Accel. Conf., Vol. 87CH2387-9(1987)707.

Figure Captions

Fig. 1: Complexity, as measured by the number of elements, versus time for Intel microprocessors, random access memory chips(dashed) and electron storage rings.

Fig. 2: Schematic comparison of AI, shown as left side activities, and NN as right side activities.

Fig. 3: Two-Input, One-Output, Feedforward Neuron.

Fig. 4: Schematic, General Neural Net with XOR.

Fig. 5: Schematic, general purpose control system using NN for feedforward and feedback control optimization. The neuron-like elements are schematic NN such as shown in Fig. 4. Their controllers can select states or allow training.

Fig. 6: Back Propagation method for a 2-input, S-neuron logical AND with learning rate l_r=1.0 and momentum α=0.9. Case (A) uses all 4 possible patterns for learning with random starting weights $\omega_{0,1}$= 0.23,-0.37 and bias b_2=-0.14. Case (B) is the identical setup except that only 3 training patterns were used and weighted as $2P_{11}$, P_{01} and P_{10}. Also, the synapse weights were constrained to be equal with starting values $\omega_{0,1}$=0.23. The dotdash curves result from the added constraint of Eq. 3 on the bias.

Fig. 7: Back Propagation method for a 3-input, $-neuron logical AND with random starting weights $\omega_{0,1,2}$= 0.33,0.41,-0.39 and bias b_3=-0.31. All 8 possible patterns were used for learning with equal weighting and l_r=1.0 and α=0.9.

Fig. 8: Tektronix 2465B scope trace obtained by switching between multiplying 3x3 and 3x1 using three Fairchild 100K series chips. The F100124 TTL-to-ECL Hex -Translator provides the input pattern and two F100104 Quint AND/NAND ECL Gates provide the internal and output layers. The vertical scale 200mV and the time scale is 5 ns per major division.

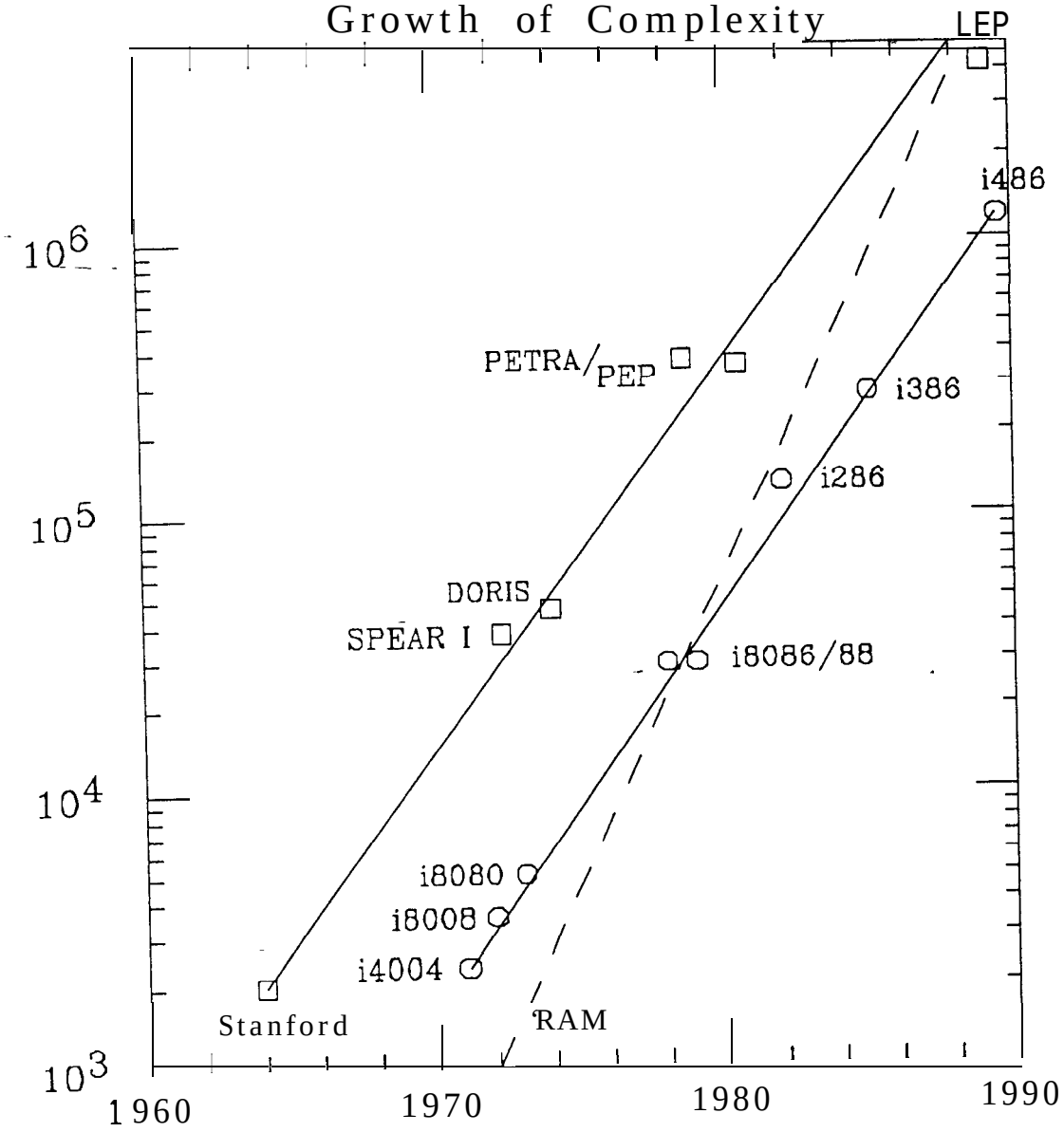

Growth of Complexity
LEP
i486
10^6
PETRA/PEP
i386
i286
10^5
DORIS
SPEAR I
i8086/88
10^4
i8080
i8008
i4004
Stanford
RAM
10^3
1960
1970
1980
1990

Fig. 1

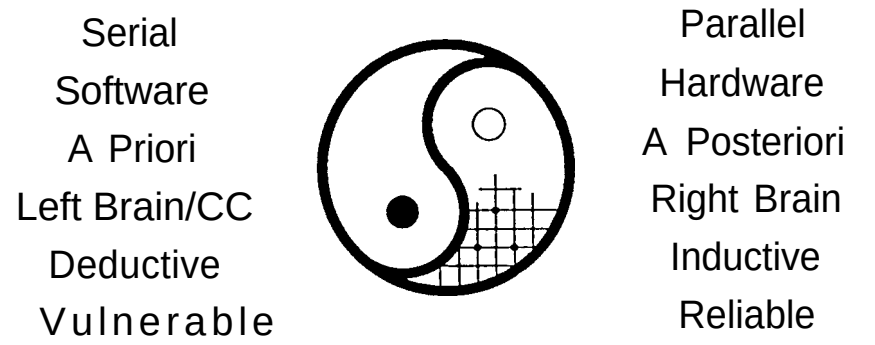
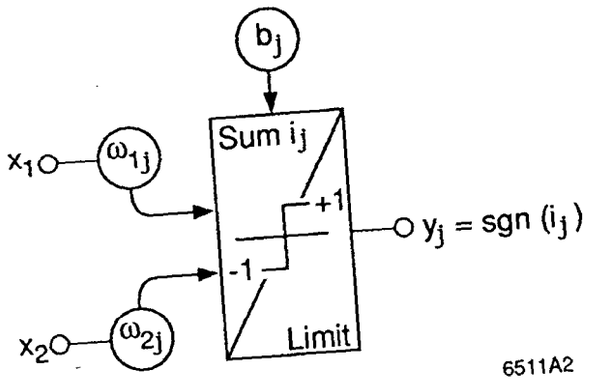

Serial
Software
A Priori
Left Brain/CC
Deductive
Vulnerable
Parallel
Hardware
A Posteriori
Right Brain
Inductive
Reliable
lo-89
6511A1

Fig. 2



lo-69

Fig. 3

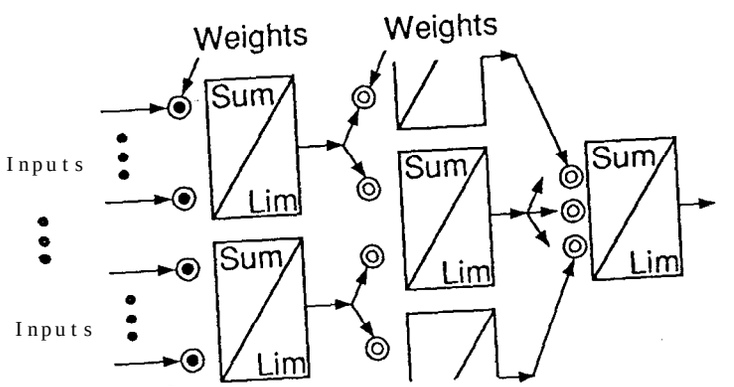

Weights
Weights
Inputs
Inputs
Sum
Lim
Sum
Lim
Sum
Lim
Sum
Lim
6511A3
10-89

Fig. 4

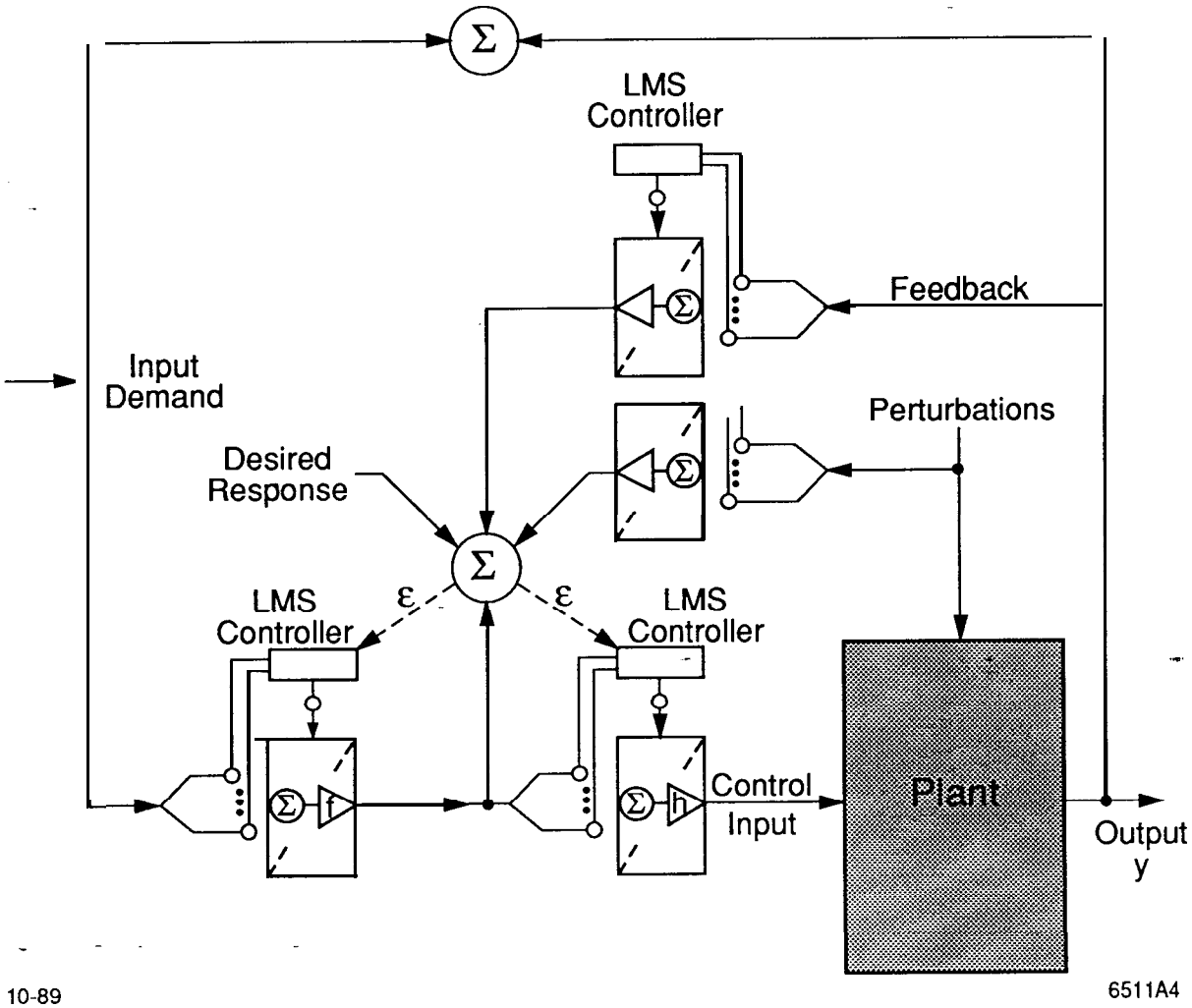

Σ
LMS
Controller
Feedback
Input
Demand
Perturbations
Desired
Response
Σ
ε
ε
LMS
Controller
LMS
Controller
Σ
f
Σ
h
Control
Input
Plant
Output
y
10-89
6511A4

Fig. 5

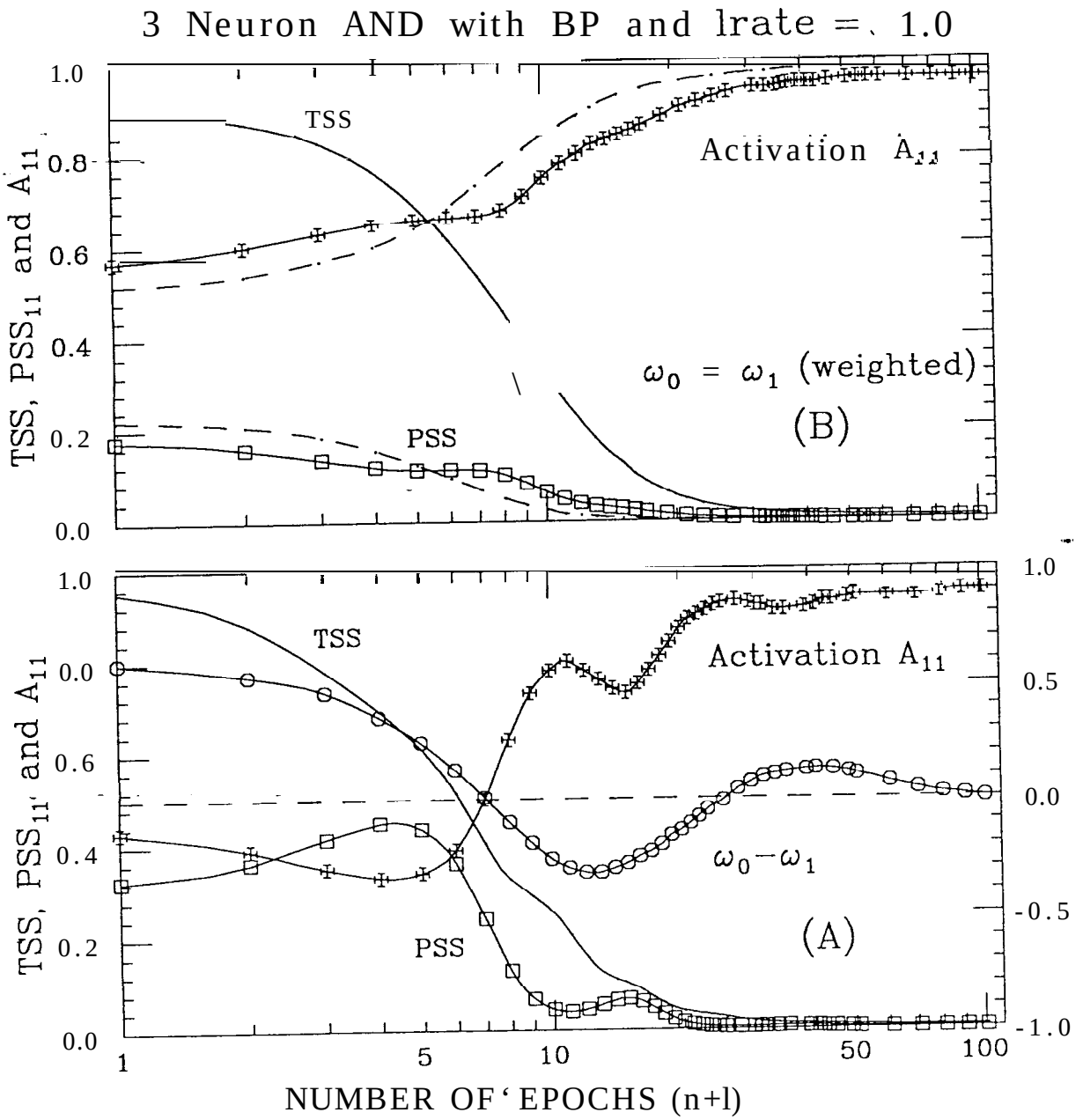

3 Neuron AND with BP and lrate =、1.0
TSS, PSS11 and A11
TSS
Activation A11
ω0 = ω1 (weighted)
(B)
PSS
TSS, PSS11' and A11
TSS
Activation A11
ω0−ω1
(A)
PSS
NUMBER OF ' EPOCHS (n+l)

Fig. 6

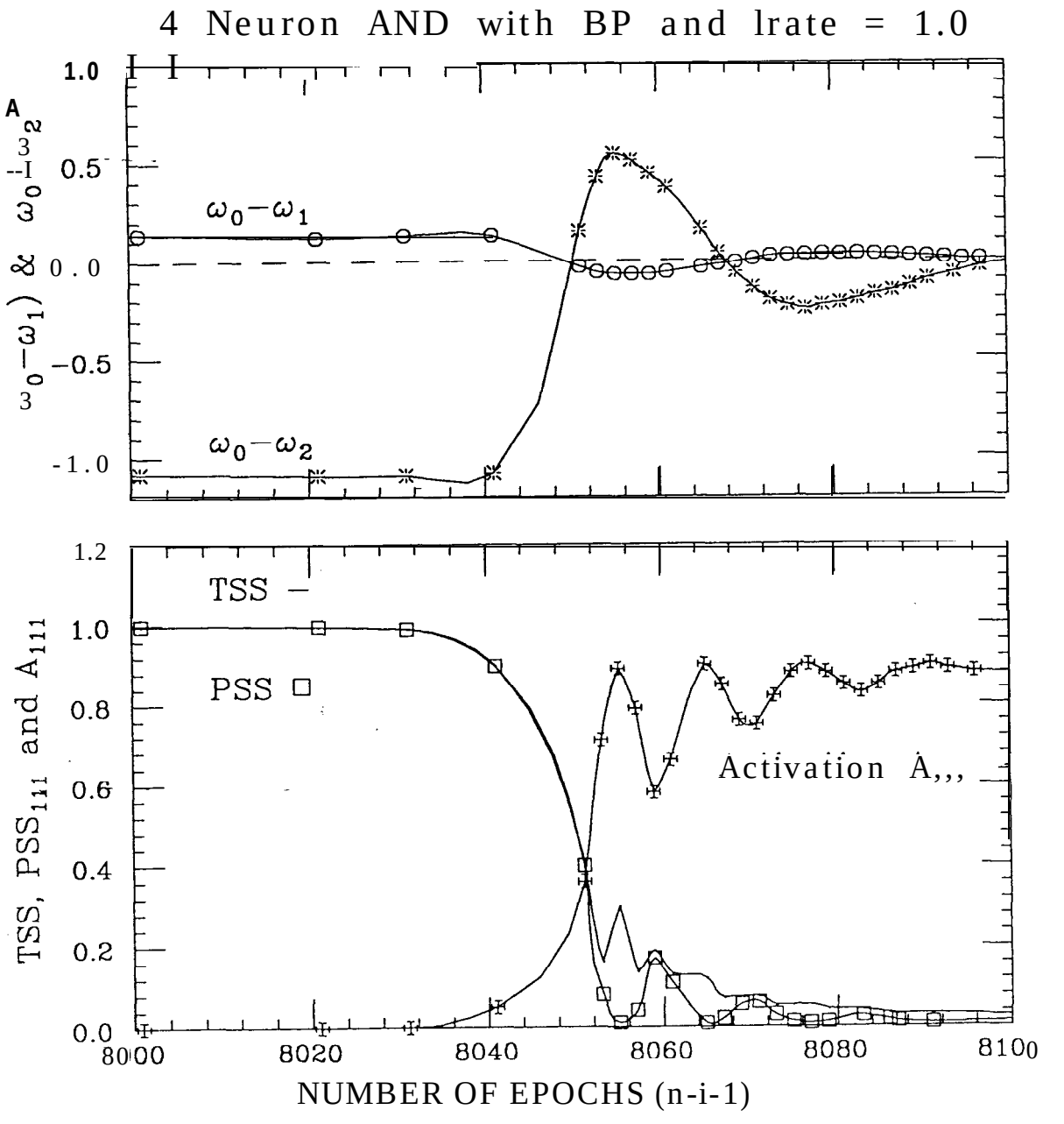
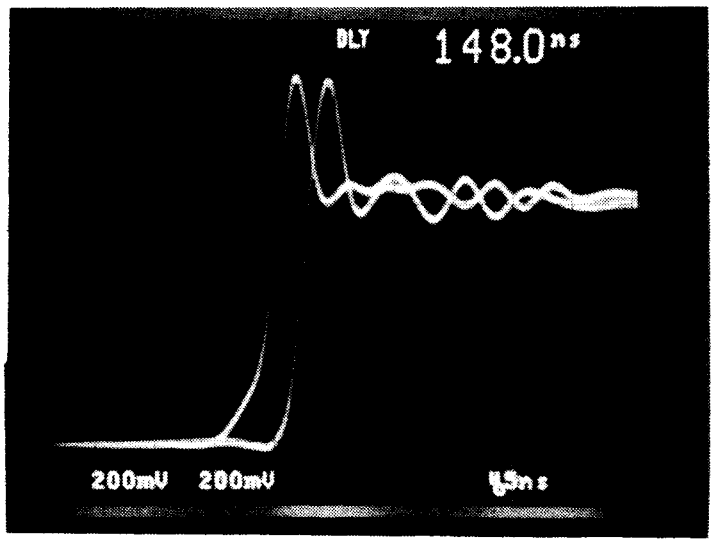

4 Neuron AND with BP and lrate = 1.0
1.0
0.5
0.0
-0.5
-1.0
$\omega_0-\omega_1$
$\omega_0-\omega_2$
TSS
PSS
Activation A_{111}
1.2
1.0
0.8
0.6
0.4
0.2
0.0
8000
8020
8040
8060
8080
8100
NUMBER OF EPOCHS (n-i-1)

Fig. 7



10-89 6489A2

Fig. 8