Overview of neurocomputers

L. M. Patnaik

Microprocessor Applications Laboratory, Indian Institute of Science, Bangalore 560 012, India

This paper presents an introduction to neurocomputers and an overview of the history of neurocomputers. Direct implementation methods of neurocomputers using techniques from microelectronics and photonics are discussed. Emulation methods using special-purpose hardware are highlighted. The role of parallel computing systems for improved performance is introduced. Some commercially available neurocomputers and performance issues of such systems are also presented.

FROM the advent of the first useful electronic digital computer (ENIAC) in 1946 until the late 1980s, essentially all information processing applications used a single basic approach: programmed computing. Solving a problem using programmed computing involves devising an algorithm and/or a set of rules for solving the problem and then correctly coding these in software (and making necessary revisions and improvements).

A new approach to information processing that does not require algorithm or rule development and that often reduces significantly the quantity of software that must be developed has recently become available[1]. This approach, called neurocomputing, allows for some types of problems (typically in areas such as sensor processing, pattern recognition, data analysis and control), the development of information processing capabilities for which the algorithms or rules are not known (or where they might be known but the software to implement them would be too expensive, time-consuming or inconvenient to develop). Since current computers operate on a totally logical basis, software must be virtually perfect if it is to work. The exhaustive design, testing and iterative improvement that routine software development demands makes it a lengthy and expensive process. The computer-aided software engineering (CASE) tools often used with neurocomputing systems can frequently be utilized to build these routine software modules in a few hours. These properties make neurocomputing an interesting alternative to programmed computing, at least for those areas where it is applicable.

Formally, neurocomputing is the engineering discipline concerned with nonprogrammed adaptive information processing systems – neutral networks – that autonomously develop operational capabilities in adaptive response to an information environment. Instead of being given a step-by-step procedure for carrying out the desired transformation, the neural network itself generates its own internal rules governing the association and refines those rules by mapping or associations of objects or representations in one set (such as written words) with objects or representations in another set (such as spoken sounds). Through trial and error, the network literally teaches itself how to do the task.

In the sections to follow, we describe the history of neurocomputing, architectures of neurocomputers and practical implementation attempts.

History of neurocomputers

The first proposal to use neural networks as a basis for computation was made in 1943 by McCulloch and Pitts, who proposed to describe neural computation by a network of binary threshold elements. More specifically, each element could compute a weighted sum of inputs from other elements and output a 0 or a 1 according to whether this sum was below or above a certain threshold.

In 1957 the perceptron was invented by Frank Rosenblatt. In addition to the discovery of the perceptron and his visionary view of the future of neurocomputing, Rosenblatt also designed and helped to build the first successful neurocomputer, the Mark I Perceptron, which functioned as a character recognizer. It was built and successfully demonstrated in 1957 and 1958.

One of the first approaches to an electronic model of the neuron was an adaptive circuit theory pioneered by Bernard Widrow in the early 1960s. Widrow worked on a simple model of a neuron for pattern recognition and signal processing (Adaline). While Adaline's capabilities were impressive, its initial applications were limited because each neuron required an operator to adjust its weights. John Hopfield was one of the first to cast neural designs into microchips. His analog-to-digital converter uses the principle of energy minimization, in which neural networks maintain a state of lowest energy. The only inherent limitation for this design method is the power dissipation cost created by the resistive interconnections. Resistors create heat, and because these circuits require high degree of interconnection – every amplifier must be connected through a resistor to every other amplifier – heat dissipation in the interconnect grid is the limiting factor on this sort of cognizer chip.

Most of the components in Hopfield's simple design already exist in other integrated circuits; hence, the methods of constructing them are well established. Accurate resistors, however, are often not needed and the problem of building them into microchip silicon had never been fully addressed. A team of Bell Lab's technicians, however, found one way of building such resistive interconnects with advanced semiconductor research apparatus. This newly developed method has allowed Bell Lab's researchers to build a 512-neuron chip with over 256,000 connections. The 512 amplifiers are arranged in blocks of 128 around the edges of the chip. Resistors are built vertically at the 256,000 intersections of a central wire grid. A special low-temperature ion beam process was used to place the resistors.

To build Hopfield-style chips, different microchip manufacturing techniques are being tried at other research labs. Carver Mead fabricated an associative memory chip that included several clever design features. The chip consisted of 22 analog amplifiers arranged along the diagonal of a 22 × 22 cell array. In 1986, Robert Hecht-Nielsen built the first commercial neural network, which is described later.

Research in ways to capture neural network in silicon has made remarkable progress in the last few years, and with major research efforts being made in the United States, Europe and Japan, even more progress will be forthcoming.

Implementation aspects of neurocomputers

The progress of neurocomputing is inexorably related to the progress in neurocomputer design. Historically, the first real-world applications of neurocomputing became possible only with the advent of sufficiently powerful implementation hardware. Most future applications are likely to require ever more powerful neurocomputers. This section discusses the implementation of neurocomputers and their performance measures.

The fact that Artificial Neural Networks (ANN) are composed of a very large number of comparatively simple processing elements leads us to the conclusion that present-day digital computers may not be particularly well suited as an implementation medium for neural nets. Neurocomputers can be built either as hard-wired machines (direct implementation techniques) capable of implementing only a limited set of neural network architectures, or as flexible, reconfigurable machines (emulation approach) that can run a wide variety of neural network architectures. For those applications where running multiple network types is important, reconfigurability is essential. However, hardware designed expressly for running a specific network architecture can sometimes be more efficient. We will begin this section with a definition and a brief analysis of direct ANN implementation approaches; we will then define and focus on emulation-based implementations.

Direct implementation techniques

In direct implementation of ANNs, individual physical elements (such as wires, transistors, light-emitting diodes, etc.) are allocated to represent each element of the neural network. Such an implementation is inherently parallel, in that each implementation element processes information simultaneously with respect to other elements. The most commonly encountered direct implementation schemes are based on optical or analog electronic mechanisms.

In analog VLSI (Very Large Scale Integration) microelectronics, semiconductor technology is used to produce integrated circuits containing a large number of analog components (amplifiers, comparators, etc.). Such circuits operate on continuous-valued signals, rather than on discrete-valued binary data. This makes analog VLSI attractive as an implementation medium for neural networks, which also usually operate on continuous-valued signals. Unfortunately, analog microelectronics techniques face severe technological hurdles in neural-network implementation. The existing and the projected circuit technologies impose severe constraints in wiring together large networks. As these and other technological constraints are addressed, analog VLSI will become an increasingly attractive means of realizing artificial neural networks. Prof. Carver Mead of Caltech and Synaptics Inc. has applied this analog technology in the realization of several innovative biologically inspired processing systems, including an artificial ear and an artificial retina (see Figure 1). This

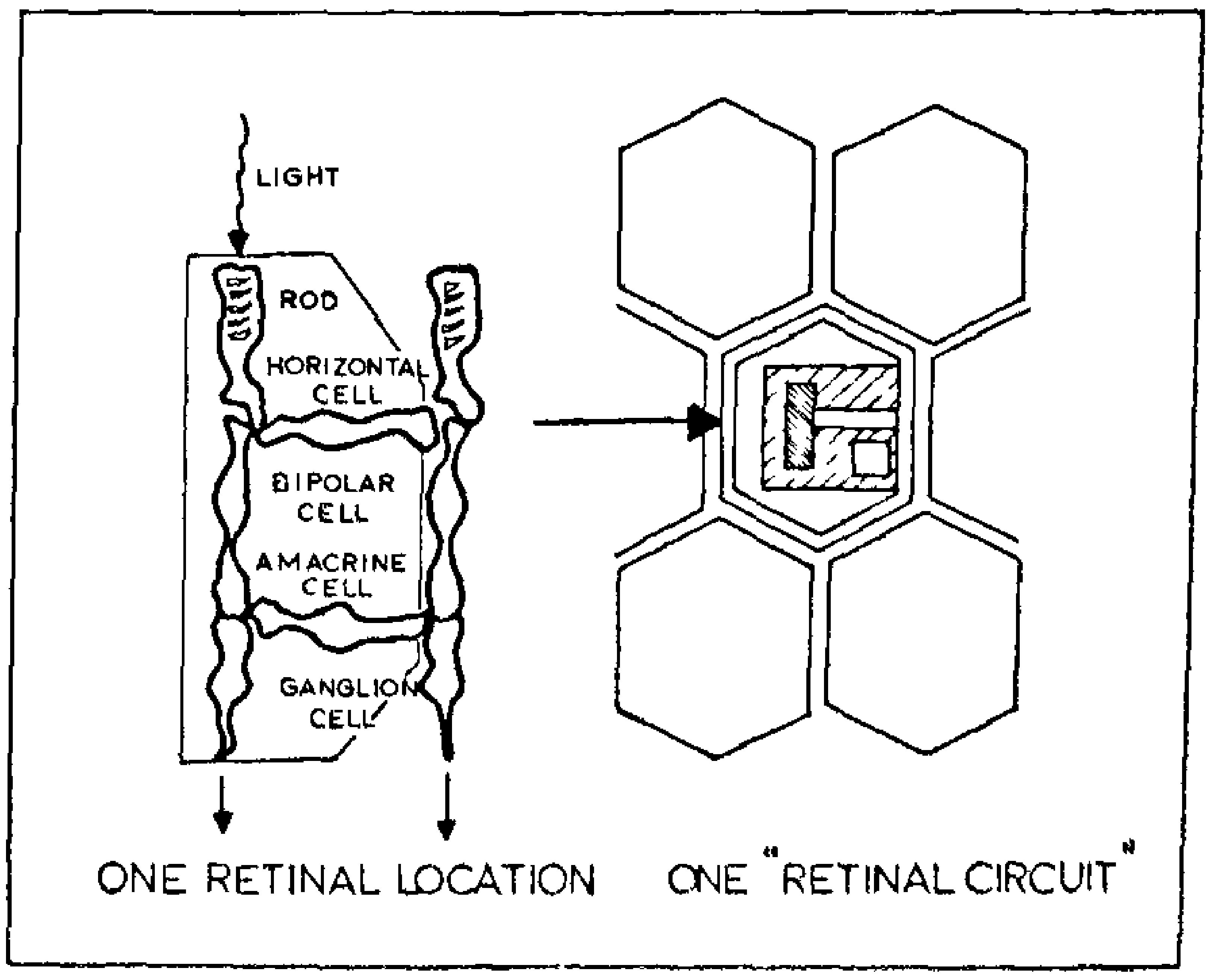

Figure 1. Analog VLSI retina chip

vision-processing chip consists of an array of interconnected analog circuits, each of which captures the operation of the neural circuitry contained in a small path of retina. Because this chip is a direct neural-network implementation, it exhibits the real-time processing speed which is possible with true parallel implementations. Mead's work exemplifies the value of biology in guiding ANN applications development.

If analog VLSI faces severe hurdles as an ANN direct-implementation medium, optical methods are perhaps even less well developed. Optical ANN implementations suffer from spatial-resolution problems (engendered, for example, by optical diffraction limits), which restrict the number of neural elements which may be implemented. Also, there are very few optical mechanisms for implementing the long-term memory required by adaptive neural nets. Optical methods exploit the fact that light beams can pass through one another without destroying their information content (though there are possible interference effects). This opens up the possibility of using optics as a high-capacity communication medium for implementing the links which connect the nodes (processing elements) of a neural network.

One type of optical neural-net implementation is illustrated in Figure 2. This example uses a hybrid electronic/optic system, in which the input and output signals for the net are implemented electronically. The inputs are produced by a set of electrically driven light-emitting diodes (LEDs). The light produced by each of these elements is conveyed to the appropriate elements of a weight matrix by lenses or by optical fibres. The weight matrix is simply a transparent optical mask whose matrix elements consist of regions shaded or tinted to a degree that corresponds to the desired matrix weight. To form an associate system, these outputs must be fed back to the net's inputs as shown in Figure 2.

Typical optical ANN implementation involves large arrays of light-emitting diodes and photodiodes, or complex nonlinear optical elements, all of which are difficult to produce. Besides the initial expenses of this exotic technology, there appears to be no way to significantly cost-reduce such systems, as would be required in high-volume ANN applications.

Direct ANN implementation is a difficult task. It is probable that direct implementations of specific carefully chosen applications will be realized in the near future. However, these techniques are not likely to soon provide an attractive means of realizing general ANN applications. As an alternative to direct implementation, we turn to digital emulation techniques.

Network emulation techniques

Network emulation techniques are quite general, permitting the implementation of arbitrary network models and arbitrary network organizations. While emulation lacks speed and perhaps the low cost characteristic of some direct implementations, it is extremely versatile and builds on the existing digital technology. This is the approach taken by most of the existing ANN companies. Figure 3 shows several methods of pursuing an emulation approach to neural-network implementation.

Emulating neural network using a standard Von Neumann computer approach suffers from both speed and memory utilization constraints, which can severely limit the real-world applications of neural networks. Even so, this approach is a logical and attainable starting point for neurocomputer development. It may be sufficient for certain neural network applications which do not require large nets or high processing throughput.

Another possible type of emulation is the use of numeric coprocessors or an array processor attached to the host system. Most of the neurocomputers that have been built are configured as coprocessors to a standard computer acting as a host (Figure 4). As a coprocessor, a neurocomputer can be thought of as just another type of peripheral (like a printer) attached to a computer.

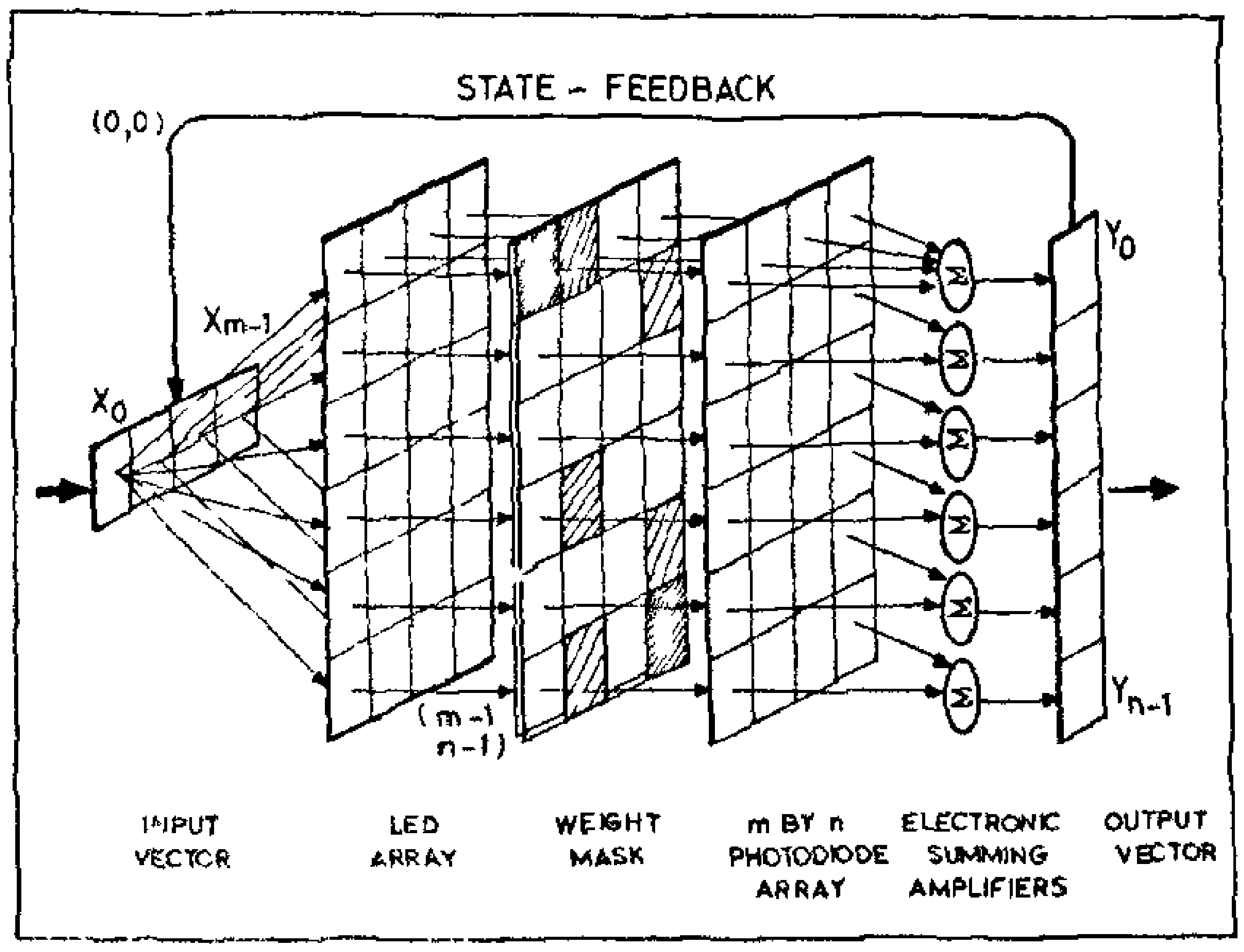

Figure 2. Electrooptical network implementation.

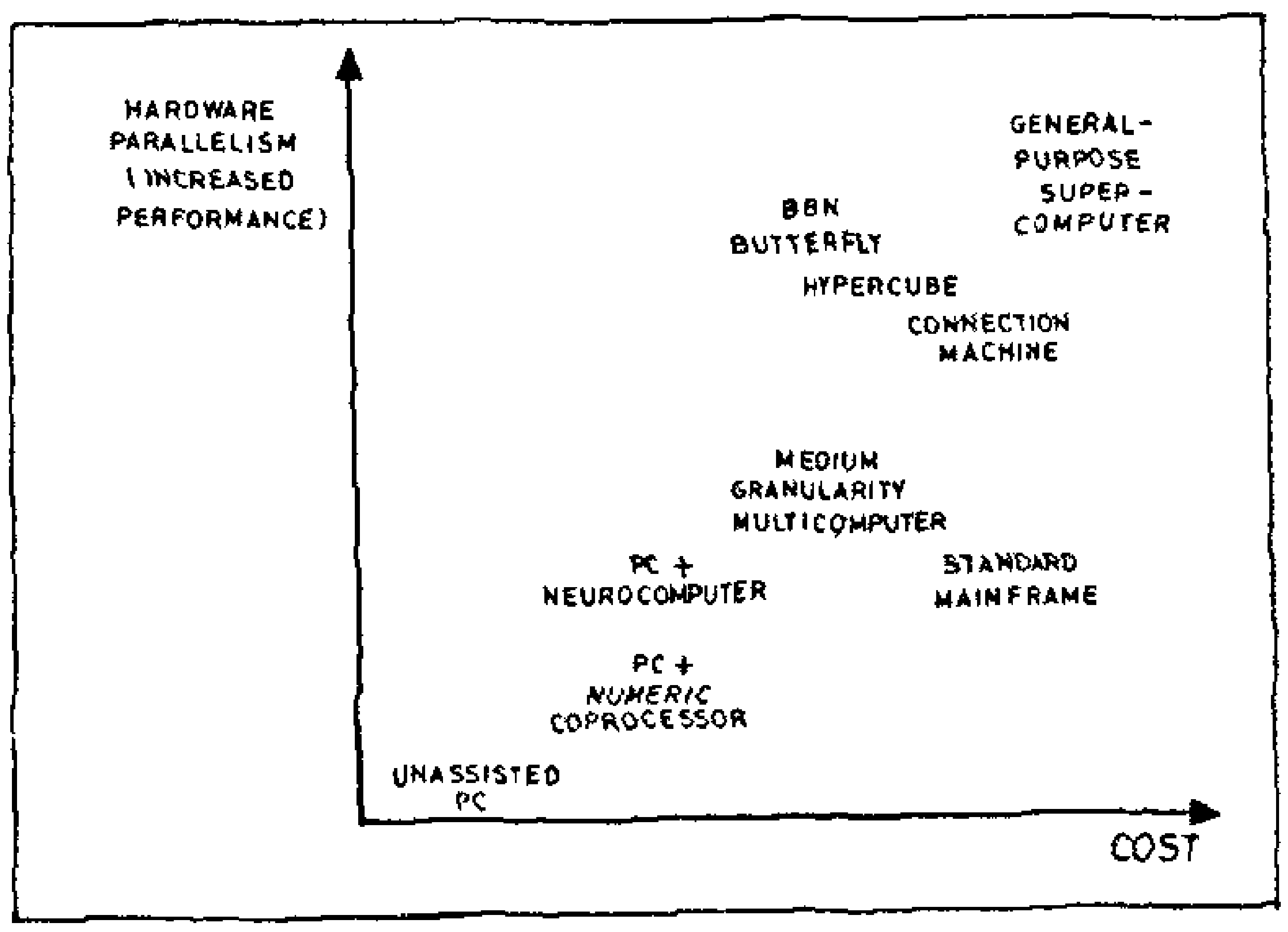

Figure 3. Electronic neurocomputers

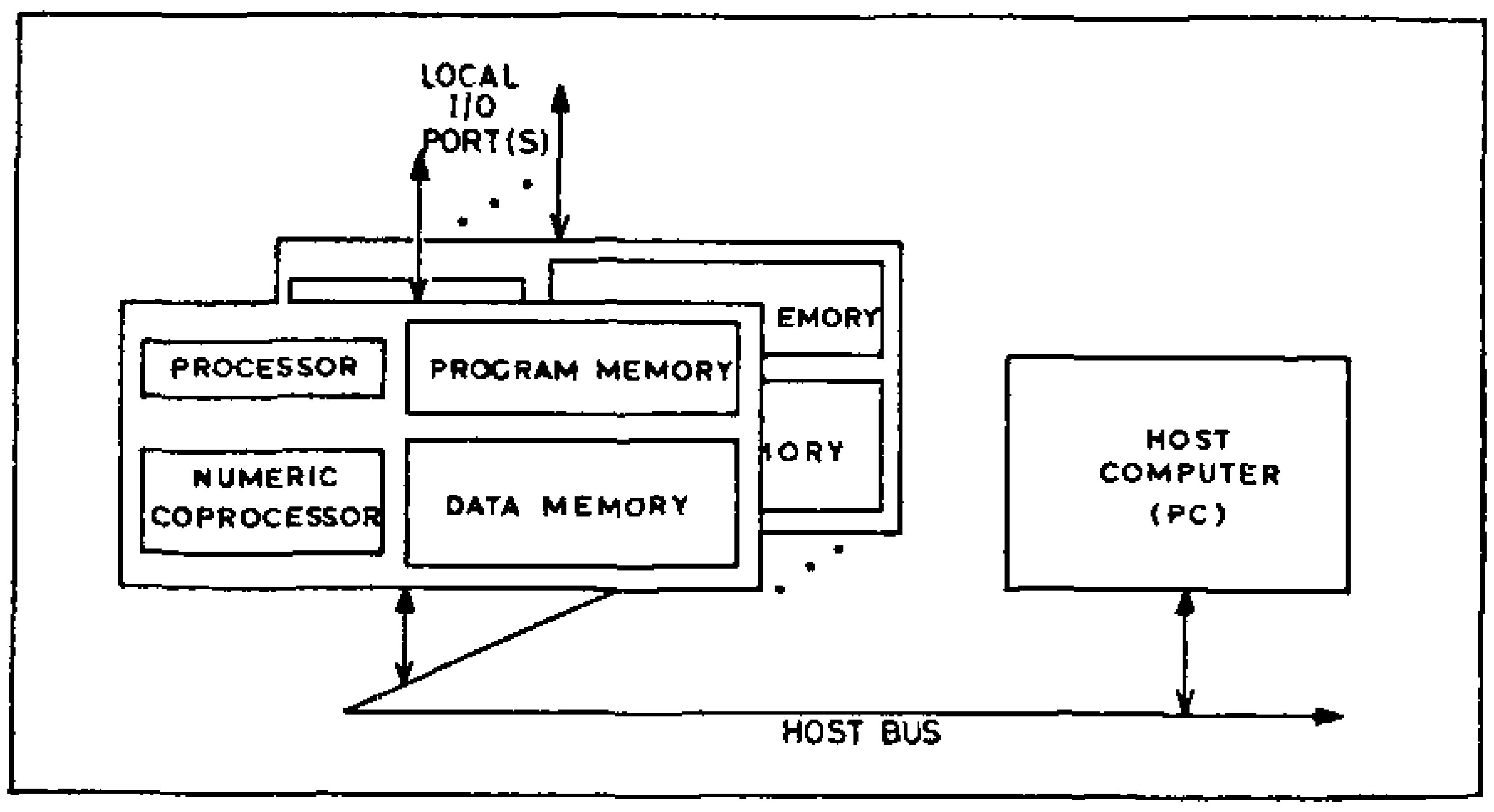

Figure 4. Neurocomputing coprocessor

Hetch-Nielsen Neurocomputers Inc. (HNC)[2] offers a neurocomputing coprocessor card for IBM PCs and clones (ANZA). This coprocessor is a fairly simple numeric-oriented computer which can exchange data and control information with its host PC. Its utility derives from its hardware support for such numerically intensive signal processing operations as sum-of-products, which are extensively used in neural-net emulation. The processing performance of the ANZA is derived from the on-board 20 MHz Motorola MC68020 microprocessor plus its associated 68881 numeric coprocessor chip. Data memory and program memory are implemented in 4 MB (4 million bytes) of RAM. Several ANZA cards (or its successor ANZA Plus) are installable within one host PC, increasing the size of the nets which may be emulated. The ANZA was designed to make easy neurocomputing integration into existing software environments. Interaction between user software on the ANZA is accomplished through a set of callable subroutines that comprise the User Interface Subroutine Library (UISL). The UISL provides access to all of the ANZA's network data structures and functions. This allows the user to add quickly and easily neural network capability to new or existing software. As a general-purpose neurocomputing coprocessor, the ANZA can implement anyone of the known network paradigms.

As a third alternative to the emulation techniques, multiprocessor systems are used. Each processor in such a configuration is much faster than a microprocessor-based numeric coprocessor, and each processor also has a much larger amount of local storage (and thus the ability to describe a larger piece of network). However, such a machine is also much more expensive than a coprocessor-based system. Some of the multiprocessor systems used for neural-network implementation include the transputer-based computing surface with 40 transputers, the WARP machine, the BBN Butterfly, the connection machine, the hypernet and the hierarchical network of hypercubes. In the next section, we discuss some of these architectures for the simulation of neural networks.

The BBN Butterfly machine

The BBN Butterfly machine is characterized by a higher degree of hardware parallelism, and by the nature of the communication mechanism which links its constituent processors. Such a machine is much more expensive than a neurocomputing coprocessor, but is also significantly more powerful. This class of machines is also typically general-purpose, rather than being specialized for neurocomputing. The Butterfly machine usually contains 4 to 128 individual processing nodes. Each such node is a complete computer containing a Motorola 68020 processing unit and an optional numeric coprocessor, local data memory and local program memory. ANN emulation can take advantage of many of the Butterfly's architectural features, provided that the needed supporting software is available. As in any general-purpose system, however, neural-network emulation can be even better implemented on a parallel system which is architecturally specialized for such a task.

Hypercube architecture

Let us now consider hypercube machines, yet another type of parallel digital computers which are potentially useful for neural-network emulation[3]. Like Butterfly, existing hypercubes consist of a few tens to a few hundred processing nodes. Each such node is like the nodes used in the Butterfly (i.e. each contains a processor, memory, etc.). The price of a hypercube is also similar to that of a Butterfly containing a like number of processor nodes. The distinction between these classes of architectures revolves around the nature of the internode communication mechanism. The Butterfly uses a particular type of nonblocking multiport message switches, whereas the hypercube machines use high-speed point-to-point serial communication links between nodes. In an n-dimensional hypercube design, each processing node has a dedicated link to n neighbouring nodes. Most hypercube machines are of degree 5 or 6; each node has direct links to 5 or 6 other nodes.

Like the Butterfly machine, hypercubes are usually implemented as general-purpose computers. ANN simulation may be readily performed on such machines, given the necessary software. While the parallelism of hypercubes makes them attractive for neural-network emulation, their communication mechanism is less than ideal for this task.

Hierarchical network of hypercubes

The hierarchical network of hypercube proposed by Kumar *et al.*[4,5] is a low-cost, expandable and fault-

Table 1. Comparison of backpropagation implementations on different systems

Computer	Performance in CUPS**
Connection machine 2	280 M
WARP (10)	17 M
ANZA Plus	10 M
Butterfly 64	8 M
SUN 3	250 K
EH (3, 2)	9 45 K
FH (3, 3)	60–64 K*
EH (3, 4)	400–480 K*

**CUPS Connection updates per second
*Estimated values

tolerant parallel architecture suitable for neural-network simulation. A hierarchical network of hypercubes is represented by EH(k, l), where EH stands for Extended Hypercube, k is the dimension of the basic module and there are $l + 1$ levels of hierarchy in the network (l is often referred to as the degree of EH). Several neural network paradigms have been simulated on this architecture with significantly improved performance compared to that obtained on CM-2 and iWARP machines. Table 1 compares the backpropagation implementations on different systems.

TMI connection machine

We will briefly consider one more parallel computer which is usable in ANN emulation: the Thinking machines, Inc. (TMI) Connection Machine. This system is quite different from any other machine considered previously. The Connection Machine is architecturally quite innovative, but acceptance of the machine has been relatively slow. (Writing software for this system has proven challenging.) It is an expensive machine. The connection machine is a true massively parallel computer, consisting of up to 64 K (65,536) individual processing nodes, each of which is much simpler than a full stand-alone computer. In fact, each such node contains a bit-serial processor (i.e. one which performs its operations, such as addition, one bit at a time) as well as a small amount of memory. The resulting processing nodes are small enough for several of them to be built on a single chip. The overall machine contains many such processing-element chips. The processing nodes do not contain enough local memory to emulate a reasonably large chunk of network, forcing an extensive use of slow serial interprocess communication. Programming the machine is difficult, though software support is increasing. Finally, the connection machine is so large and expensive that its use is impractical for many interesting neural-network applications (e.g. military signal processing systems, or embedded machine controllers).

Performance measures of neurocomputers

A number of neurocomputer performance measures that have been found to be useful in assessing the capabilities and costs are presented in the following paragraphs.

Capacity

There are two measures of capacity. The first concerns the sizes of specified types of networks that can be implemented by the neurocomputer. The second measure of capacity is the total number of networks of specified types and sizes that can be stored ready for immediate use within the neurocomputer. This second measure of capacity is important, since most applications of neurocomputing require multiple networks, and since loading networks into a neurocomputer over an interface is usually a time-consuming process.

Speed

Speed is rated by how quickly a specified network of a specified size can update the interconnections. Commercial neurocomputers are typically benchmarked using a large backpropagation network of a stated number of layers and processing elements per layer. The total number of interconnections that can be updated in one second is often used as the speed metric.

Cost

Cost is one of the most difficult performance measures to determine in a fair and accurate way. Further, there are legitimately different approaches to cost measurement. The measurement approach that most comprehensively incorporates all of the costs involved is the calculation of the delivered-capability cost. Given a neural network of a certain type and size that is activated from software, the delivered-capability cost is the total cost of running that network through some stated number of update iterations. This cost should include the purchase price of the neurocomputer (including any taxes), the cost of any required software, neurosoftware languages, cables, LAN hardware, and so on, and neurocomputer system administration costs. Maintenance costs during the useful life of the machine must also be incorporated.

Accuracy

Accuracy is a neurocomputer capability performance measure that assesses the numerical accuracy of the machine in terms of the needs of certain specified networks. The measure is typically binary. Either the

Table 2. Neurocomputers built to date*

Neurocomputer	Year intro-duced	Technology	Capacity			Speed	Developers	Status[§]
			Number of processing elements	Number of connections	Number of networks[†]	Connections per second[‡]		
Perceptron	1957	Electromechanical and electronic	8	512	1	10^3	Frank Rosenblatt, Charles Wightman, Cornell Aeronautical Laboratory	Experimental
Adaline/Madaline	1960/62	Electrochemical (now electronic)[‖]	1/8	16/128	1	10^4	Bernard Widrow, Stanford Univ.	Commercial
Electrooptic crossbar	1984	Electrooptic	32	10^3	1	10^5	Demitri Psaltis, California Inst of Technology	Experimental
Mark III	1985	Electronic	8×10^3	4×10^5	1	3×10^5	Robert Hecht- Nielsen, Todd Gutschow, Michael Myers, Robert Kuczewski, TRW	Commercial
Neural emulation processor	1985	Electronic	4×10^3	1.6×10^4	1	4.9×10^5	Claude Cruz, IBM	Experimental
Optical resonator	1985	Optical	6.4×10^3	1.6×10^7	1	1.6×10^5	Bernard Soffer, Yuri Owechko, Gilbert Dunning, Hughes Malibu Research Labs	Experimental
Mark IV	1986	Electronic	2.5×10^5	5×10^6	1	5×10^6	Robert Hecht- Nielsen, Todd Gutschow, Michael Myers, Robert Kuczewski, TRW	Experimental
Odyssey	1986	Electronic	8×10^3	2.5×10^5	1	2×10^6	Andrew Penz, Richard Wiggins, Texas Instruments Central Research Labs	Commercial
Crossbar chip	1986	Electronic	256	6.4×10^4	1	6×10^9	Larry Jackel, John Denker and others, AT&T Bell Labs	Experimental
Optical novelty filter	1986	Optical	1.6×10^4	2×10^6	1	2×10^7	Dana Anderson, Univ. of Colorado	Experimental
ANZA	1987	Electronic	3×10^4	5×10^5	No limit	2.5×10^4 (1.4×10^5)	Robert Hecht- Nielsen, Todd Gutschow, Hecht-Nielsen Neurocomputer Corp.	Commercial
Parallon 2	1987	Electronic	10^4	5.2×10^4	No limit	1.5×10^4 (3×10^4)	Sam Bogoch, Oren Clark, Iain Bason, Human Devices	Commercial
Parallon 2 x	1987	Electronic	9.1×10^4	3×10^5	No limit	1.5×10^4 (3×10^4)		Commercial
Delta floating-point processor	1987	Electronic	10^6	10^6	No limit	2×10^6 (10^7)	George A, Works, Willaim L, Hicks, Stephen Deiss, Richard Kasbo, Science Applications Int'l Corp	Commercial
ANZA plus	1988	Electronic	10^6	1.5×10^6	No limit	1.5×10^6 (6×10^6)	Robert Hecht- Nielsen, Todd Gutschow, Hecht-Nielsen Neurocomputer Corp.	Commercial

*Numbers given pertain to individual boards or chips. More than one board may be used to build an individual machine.
[†]Number of networks that can be simultaneously resident on the board, without going to an outside memory peripheral
[‡]Speed outside the parentheses is with learning; speed inside the parentheses is without learning
[§]Experimental describes a one-of-a-kind device or machine built to explore an idea or prove a point, 'commercial' describes a device or machine that has been offered for sale.
[‖]Early versions required continuous electroplating lasting about a minute for full-scale change.

neurocomputer can implement a particular specified neural network to the necessary degree of accuracy, or it cannot do so.

Generality

Some neurocomputers are suitable for implementing only certain neural-network architectures. Other neurocomputers can implement essentially any architecture. Given a specific list of architectures, generality is the binary measurement of whether the neurocomputer can or cannot implement those network types.

Software interface provisions

For software programs to be able to call neural networks as subroutines, it is necessary to have software routines that can be linked with the user software and then called whenever needed to control the neurocomputer. The simplicity and ease of use of these software interface provisions are of great importance.

Configuration provision

For a computer to be used, it must be configured to run the desired neural networks. There are two basic approaches to this problem. First, most of the commonly used neural networks should be available in highly efficient (in other words, microcoded or otherwise optimized) prepackaged form. This coding efficiency ensures that the networks used most often execute at maximum possible speed on the hardware. As with software interface provisions, configuration provisions are best judged on a relative-merit basis. The availability of a general-purpose neural network description language capable of describing a large percentage of neural-network architectures efficiently for use with the neurocomputer is an important consideration

Table 2 shows the different neurocomputers built so far, their performance in terms of speed and capacity, and the year introduced.

Conclusions

We discussed here the different approaches for implementing neurocomputers. The future development of neurocomputers will be governed by the development of accurate mathematical models for artificial neurons. Since it is predicted that the models will involve more of analog computations, it is hoped that more emphasis will be laid on analog hardware design and photonic computing for the design of efficient neurocomputers.

1. Lippman, R , *IEEE ASSP Mag* , April 1987, pp 4–22
2 Hecht-Nielsen, R , *IEEE Spectrum*, March 1988, 36–41
3. Mahadevan, Indu and Patnaik, L M , *J Parallel Comput* , 1992, **18**, 401–413.
4. Mohan Kumar, J., PhD Dissertation, Indian Institute of Science, Bangalore, 1992
5. Patnaik, L. M and Mohan Kumar, J , *J Comput Electr Eng* (to appear).

Mathematical modelling of neurons and neural networks

Chandan Dasgupta

Department of Physics, Indian Institute of Science, Bangalore 560 012, India and Jawaharlal Nehru Centre for Advanced Scientific Research, Bangalore 560 012, India

The basic concepts and techniques involved in the development and analysis of mathematical models for individual neurons and networks of neurons are reviewed. Some of the interesting results obtained from recent work in this field are described. The current status of research in this field in India is discussed.

THE development of an understanding how the human brain functions is one of the most important and challenging tasks faced by modern science. During recent years, a great deal of progress has been made in experimental neurobiology. At the same time, efforts towards the development of mathematical models for some of the experimentally observed phenomena have gained momentum. Neurosciences cover a vast range of phenomena extending from the molecular to the behavioural level. In this paper, I have made an attempt to provide a brief overview of the concepts and techniques involved in the development and analysis of