"Progress in Neural Networks"
Vol. 1, ed. O. Omidvar, Ablex Publishing Corporation, 1991.

8
VLSI Chips for Neural Networks

Michel Verleysen
Paul G. A. Jespers
Université Catholique de Louvain, Belgium

1. INTRODUCTION

The pioneering work of McCullogh and Pitts [1] raised the interest in the study of the human nervous system. Over the past 50 years, a number of considerable efforts have been made in order to increase the computational power of artificial systems. Classical computers, based on Von Neumann's architectures, seem to be appropriated when the speed of achieving repetitive tasks is important. However, the most sophisticated computers cannot solve perception problems that are obvious for a 5-year-old child. The need for finding new computer architectures is thus increasing since the number of tasks we want to be solved by artificial devices is increasingly growing.

Over the past few years, the number of researchers exploring artificial neural networks has increased spectacularly. Various research fields are studied, such as theoretical background, convergence properties, applications, electronic and optic implementations, and so on. Results are very promising in any of these ways, and one can expect that the number of industrial applications will be important in the next few years.

Most of the actual research in artificial neural networks have been, up to now, achieved by simulations on classical computers. However, the models became too large and too complicated to be handled by conventional machines; furthermore, two main characteristics of neuromorphic systems are their speed and their fault tolerance, which are difficult to appreciate with digital systems. Artificial neural networks need, thus, implementations more adapted to their inherent properties, since effective simulations exceed the limits of conventional machines.

The analogy between the human nervous system and artificial neural networks is restricted to some precise characteristics. In fact, because of the inherent complexity of the nervous system, it is impossible to consider an artificial device, namely a neural network, with all the possibilities of a human brain. Only some of its characteristics will therefore be emulated, and the most interesting

features will be incorporated in artificial neural networks. Their attractiveness resides in the fact that some of these characteristics are nonexistent in classical artificial devices (fault tolerance, recognition facilities, etc.), precisely, nervous systems and electronics have radically different structures: Millions of years of evolution of the human brain and 50 years experience in computer electronics lead to radically different architectures. The following table shows the main differences between a neural system and digital electronics [2].

It is obvious that these two approaches are radically different and seem to be incompatible. The challenge is thus to design electronic structures with the interesting characteristics of neural networks, such as, their parallel, speed, fault tolerance, and retrieval properties. Because of the features listed above, digital electronics is not well-suited to such realizations; this chapter will point out the advantages of analog electronics in the design of artificial neural networks.

2. MODELS OF NEURONS AND SYNAPSES

A wide variety of neural networks has been described recently in the literature. The most commonly used are the Hopfield's net, feed-forward nets, such as the multilevel perceptron, and locally interconnected nets, which are the closest to the human biological model. Although a lot of other neural networks exists, almost all of them can be related to one of these three models, sometimes through some generalizations.

A Hopfield's neural net is a fully interconnected network, where each neuron is connected to each other [3]. Figure 8.1 represents a Hopfield's network with N neurons and N^2 synapses. Once the values of each neuron are computed, they are fed back into the network through the synapses, and a new cycle can occur.

An example of a feed-forward net is the multilevel perceptron [4]. In such nets (Figure 8.2), the output of a neuron in layer #n is connected to the input of neurons in layer #n + 1; there is therefore no feedback, such as in the Hopfield's net.

Table 8.1. Comparison between neural systems and digital electronics.

Neural	Electronic
analog	digital
parallel	serial
low power, speed	high power, speed
fuzzy	accurate
redundant	fault sensitive
monolithic	modular
very high fanout	sparsely connected

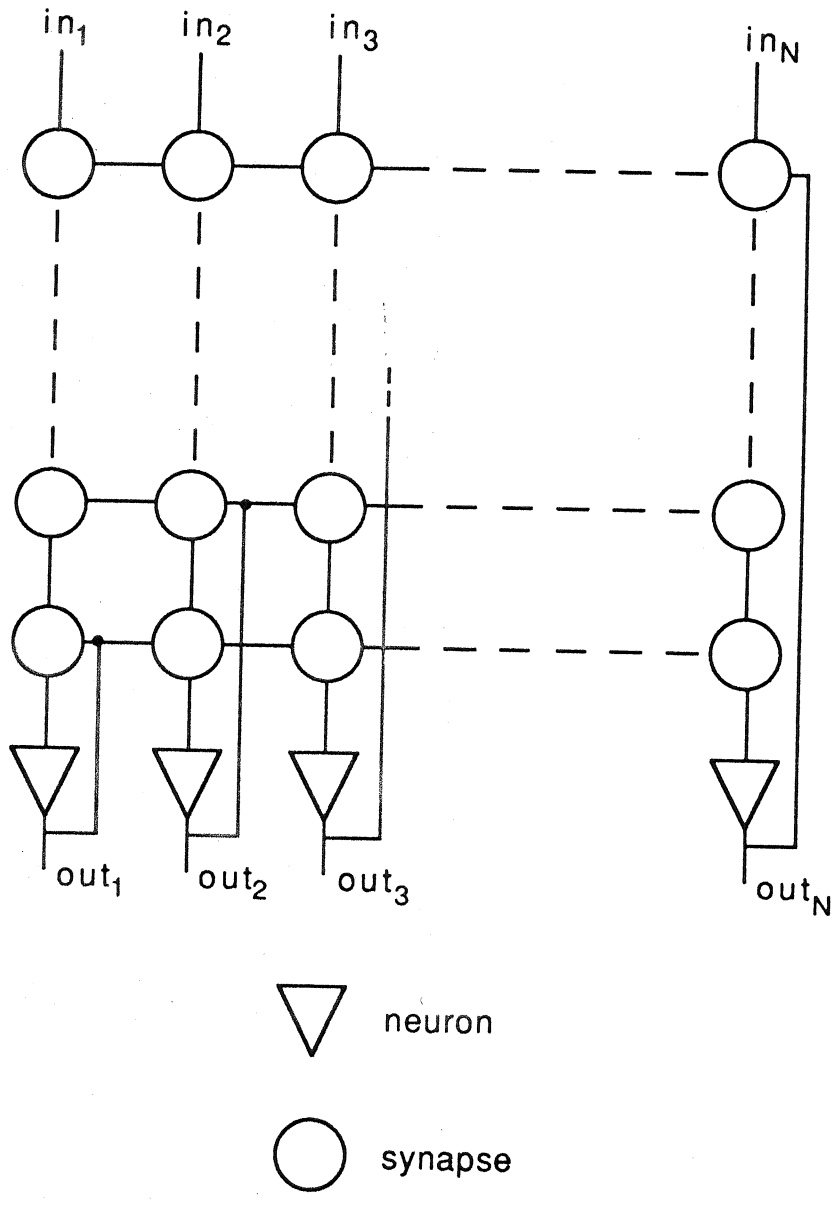

Figure 8.1. Hopfield's network.

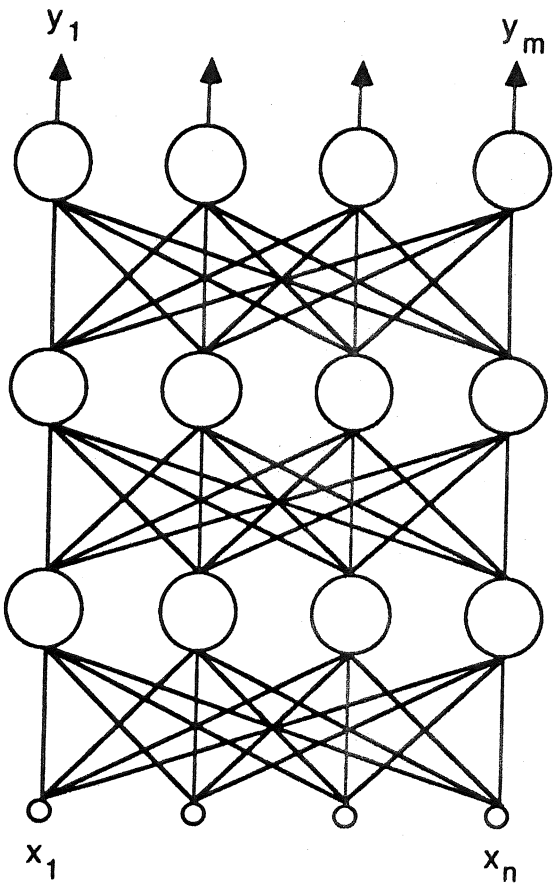

Figure 8.2. Multilevel perceptron.

The third sort of mentioned net is the locally interconnected network (Figure 8.3). It is similar to Hopfield's net, except that each neuron is no more connected to all the other neurons, but only to a small part of them. An example of such a network is the silicon retina implemented by C. Mead [5].

The functionality of neurons and synapses in these models are roughly the same. Each neuron is connected to some others through programmable connections called synapses. The function of a neuron is to realize a weighted sum of its inputs; this sum goes through a nonlinearity, usually a threshold function or a sigmoïd, and the output of the neuron serves as input to the connected synapses. The function of a synapse is to realize an operation, usually a simple multiplication, between the output value of the connected neuron and a weight contained in the synapse. Figure 8.4 shows a neuron whose output is determined by a weighted sum of its inputs; examples of threshold and sigmoïd functions are given in

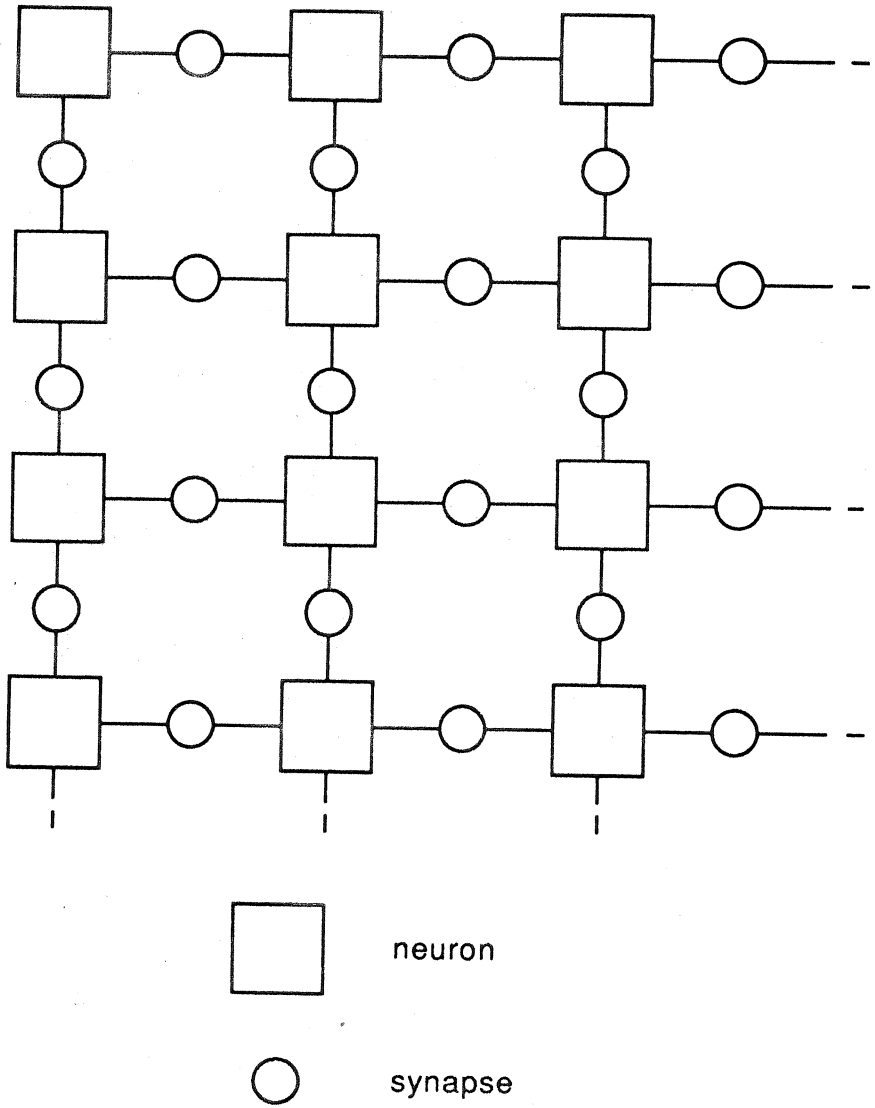

Figure 8.3. Locally interconnected network.

Figure 8.5. A learning rule is generally associated with each type of neural network in order to compute the weights contained in each synapse. This rule can be offline: The weights are computed separately before using the neural network in the convergence process. It can also be online: The weights are slightly adapted each time a new pattern is presented to the network. The learning rule and the connectivity (i.e., the way the neurons are connected to each other) is characteristic of each type of neural network.

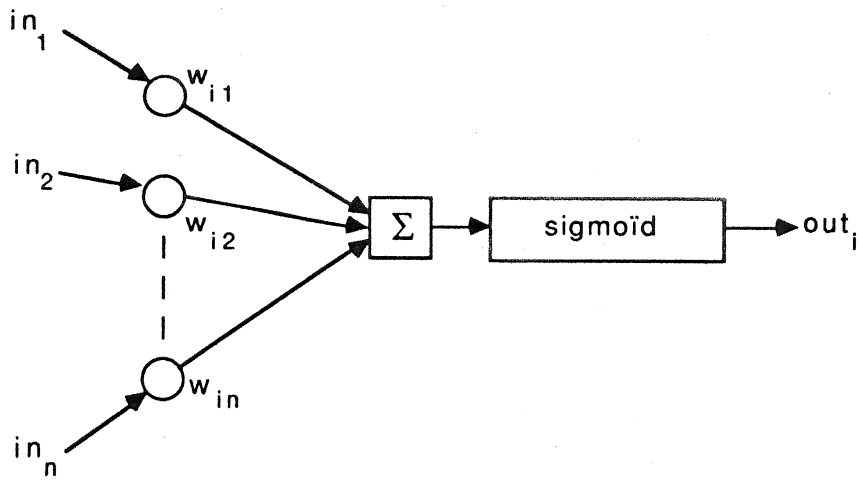

Figure 8.4. Structure of a neuron.

3. ELECTRONIC IMPLEMENTATIONS

3.2. Resistor Arrays

The first and simplest idea for implementing neural networks is to use resistors as synapses (Figure 8.6). A neuron is supposed to realize the function:

$$V_i = f\left(\sum_j T_{ij} V_j + I_i\right)$$

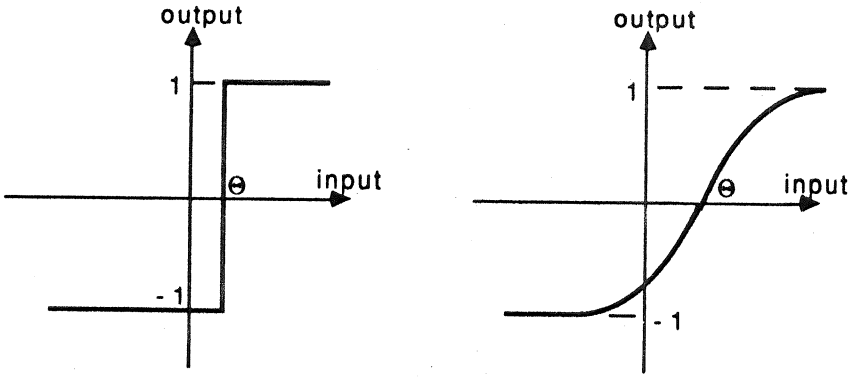

Figure 8.5. Activation functions.

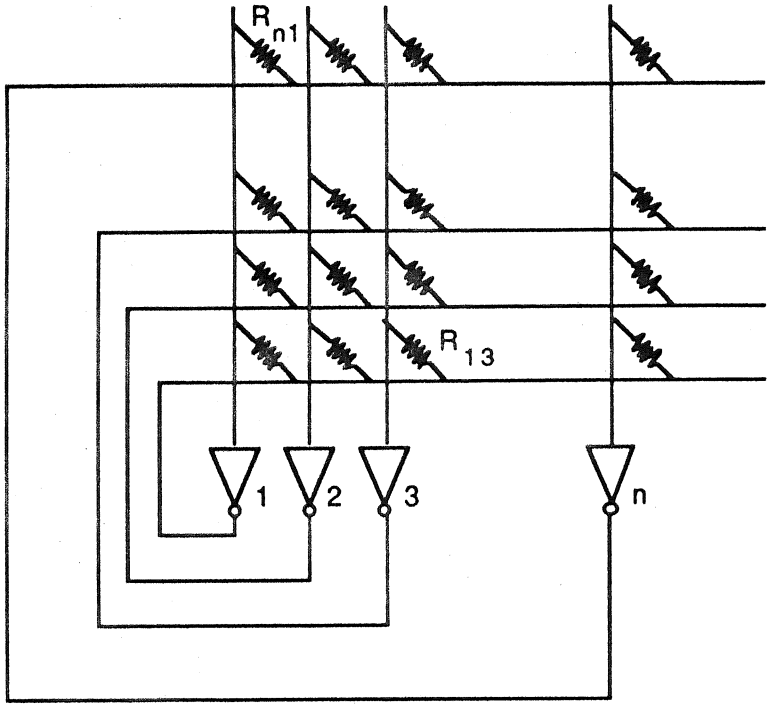

Figure 8.6. Resistor array.

where

V_i is the output of neuron i;
T_{ij} is the weight of the synapse connecting neurons i and j;
I_i is the input of neuron i;
f is the nonlinear transfer function of the neuron (threshold function, sigmoïd, etc.).

The value of each coupling resistor is given as

$$R_{ij} = 1 / T_{ij}.$$

In the synapses, the output voltage of each neuron is thus converted to an input current for another neuron i:

$$I_{ij} = V_j / R_{ij} = T_{ij} V_j.$$

All the currents derived from the synapses connected to neuron i and the input current are summed, and the neuron must therefore be a nonlinear conductance which converts its input current into an output voltage:

$$V_i = f\left(\sum_j V_j/R_{ij} + I_i \right).$$

The main problem with this kind of implementation is that resistors are not commonly used in standard CMOS technology: They usually take a large area on the chip, and it is therefore impossible to implement networks with a huge number of interconnections. Furthermore, to program the network, the resistors should be variable, which is very difficult to achieve.

A solution to these problems is given by Howard et al. [6]. They implemented a resistor array by deposing amorphous silicon by e-beam evaporation onto a patterned poliamide layer. The density of integration corresponds to about 6.10^6 resistors per square centimeter, and is thus suited for large arrays of synapses. However, the synapses are not programmable since the connection values have to be chosen during the fabrication processes; this sort of realization must thus be restricted to optimization problems where the weights must not be changed, and is not suited for pattern recognition where the weights depend on which patterns to store in the network.

3.2. Digital Networks

In order to realize programmable connections, several digital chips have been fabricated. They all realize a compromise between the spatial complexity of neural networks which is impossible to achieve by digital processors and a temporal complexity, that is, a temporal multiplexing of signals to avoid a huge number of interconnections.

A first realization is the best match classifier implemented by S. Mackie and J. Denker [7]. The global design is an implementation of a classifier whose architecture is a pipeline of digital processors. Figure 8.7 shows the architecture of one of the 50 processors contained in this chip. The 128-bits patterns are stored in a ring shift register (1). The input and feature vectors are compared serially (2) by AND and NXOR functions (to compute respectively the number of common 1's and the number of bits the same). The total dot product is then loaded into a comparison register together with a tag or name associated with the stored vector (3). The dot product is then compared to the five best products coming from the previous processor (4); if the new product is better than one of the five previous products, the new one is inserted on the match list and one is deleted (5). The next processor can then compared the input to another stored vector, and the process is repeated.

Since a lot of computations are made serially, it is important to have high clock frequency and an internal pipelining in the processors. The authors announce a clock frequency up to 100 MHz; this can be achieved because no stage contains more than two gate delays. Since each processor is internally pipelined,

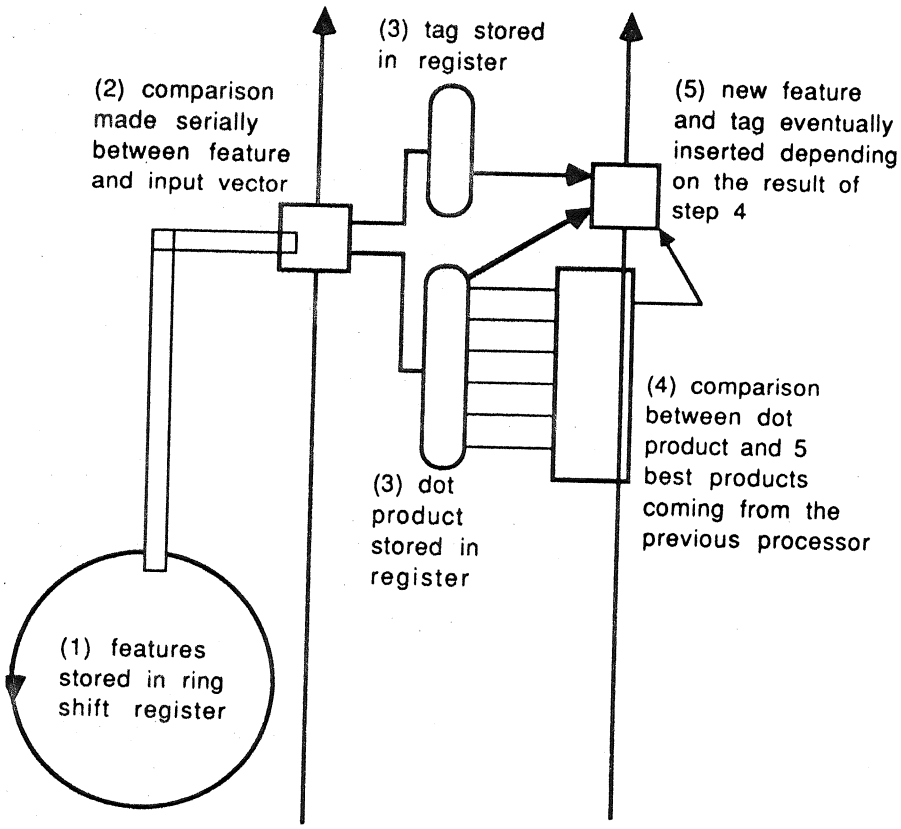

Figure 8.7. Best match classifier.

the comparison with one stored vector can begin during dot product computation in the previous processor. Because of the internal structure and the high clock frequency, the chip will produce a list of five best distances with tag strings every 1.3 μs, with a latency of about 2.5 μs; furthermore, chips are cascadable so that more than 50 vectors can be stored.

Another digital realization of a neural network is proposed by M. Weinfeld et al. [8]. It consists in a Hopfield-like architecture, although the neurons are not fully interconnected on the chip: Like in the precedent architecture, a part of the computation is serial, and the processing time is thereby increased.

To illustrate the method, we choose a network with eight neurons (Figure 8.8). Each neuron i contains a register, which is used to store successively the value of neuron i, $i + 1$, $i + 2$, and so on, during the different iterations. At each iteration, each neuron is updated with the value contained in its register; after

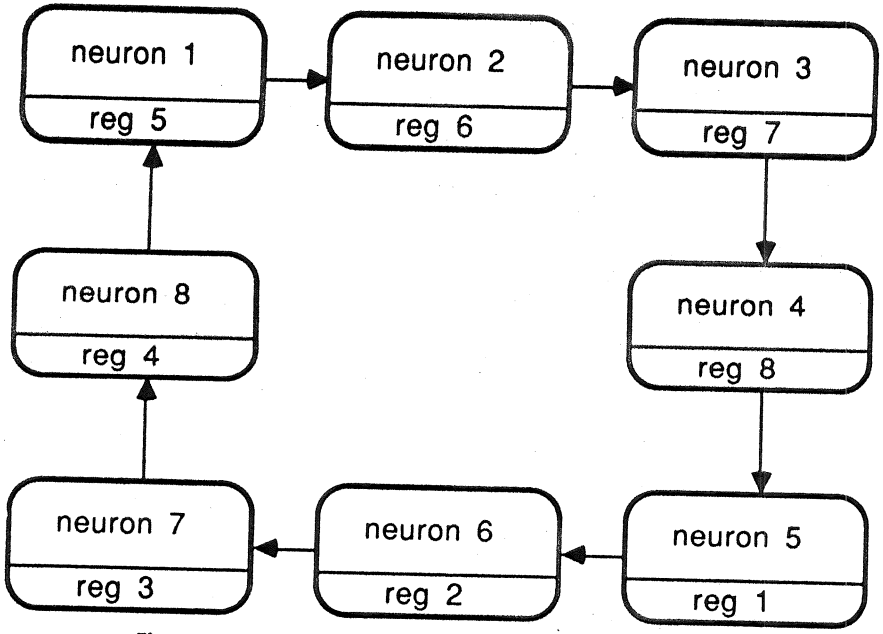

Figure 8.8. Weinfeld's architecture after 5 iterations.

eight iterations, the cycle is finished and the value of each neuron is correct. The process is repeated until a stable state is reached.

Figure 8.9 shows the schematic of a neuron which can be complex because an ALU, registers, and many logical functions are needed; a single neuron can contain up to 4,000 transistors. However, because of the simplicity of the interconnections, this solution can be used when the increased convergence time is not a problem.

A third realization is proposed by F. Blayo and P. Hurat [9]. It consists in a systolic array where the input vector propagates in 2.N steps through the network (Figure 8.10). The vector X propagates vertically, while the intermediate subtotals propagate horizontally. It is also possible to feed back the result into the network, avoiding to wait 2.N steps before beginning a new cycle (N steps is, however, a minimum). Depending on the number of bits required for the intermediate sum-of-products, the cells can be much smaller than in the previous example. Furthermore, the average convergence time can be strongly increased by pipelining the vectors in the network.

Another interesting realization is the chip presented by J. Rasure and J. Salas [10]. It implements a three-layer feed-forward perceptron, used to classify binary images of handwritten numerals 0 through 9. The implemented network contains 24 nodes in the first layer, 16 nodes in the second layer, and 10 nodes in the

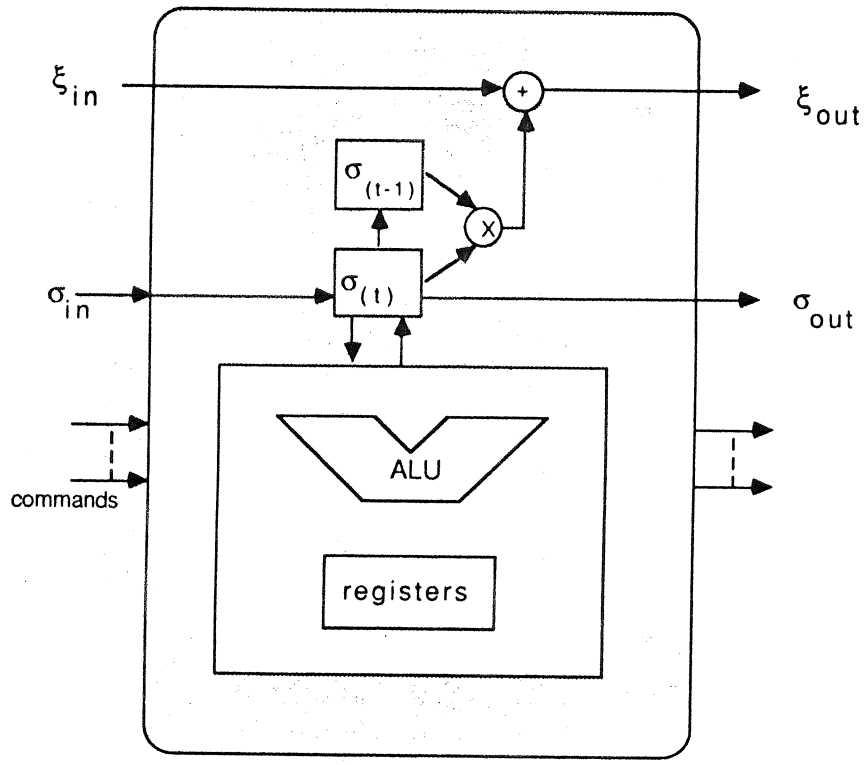

Figure 8.9. Weinfeld's architecture.

output layer. By using recirculating weights and circulating input data, this circuit avoids global communications between the 50 nodes. At a clock frequency of 20 MHz, this circuit can classify an input pattern every 0.4 milliseconds. The principle of recirculating data and weights is roughly the same as in the previous realization, but this circuit shows that through some generalization, this method is applicable to other architectures than Hopfield's networks.

To conclude the discussion about digital implementations of neural networks, we must mention the study made by B. Potu and P. Ramamoorthy [11]. They considered the use of signed-digit and modified signed-digit arithmetic to implement multistate neural networks. The chip architecture is also based on digit-serial pipelining; its main drawback is that the conventional learning rules are not adapted to multistate neurons, and therefore, special algorithms need thus to be developed.

Many other digital realizations of neural networks have been made. Although each architecture is different, they are all similar to one of the chips described above. Since digital chips cannot contain the huge number of connections re-

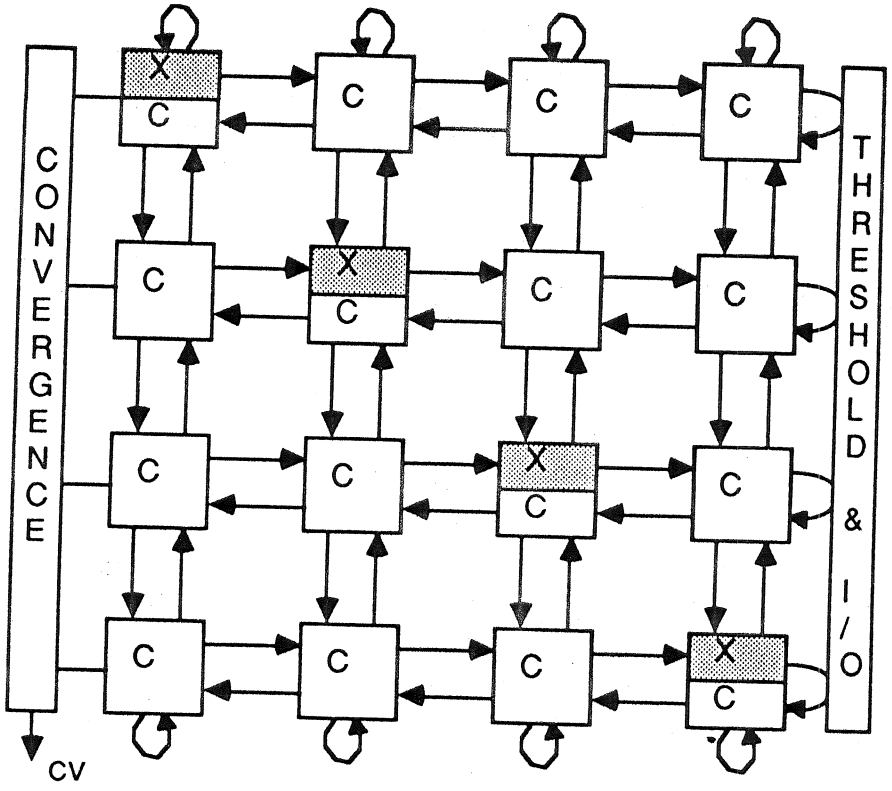

Figure 8.10. Systolic neural network [9].

quired for neural networks, the spatial complexity is replaced by a temporal complexity; this increases the convergence time, but such solutions are interesting when precision is more necessary than speed.

3.3. Analog Networks

As described in the first part of this chapter, fully interconnected networks, or any type of networks with large number of connections, are very difficult to realize with digital VLSI; this is due to the fact that digital techniques use, in general, many transistors or other devices, and that synaptic or neuronic cells take a large area on the chip. On the other hand, analog cells can be more compact, and despite of the increase of complexity during chip design, can be more suited to neural networks implementations. Of course, careful design must be done to avoid electrical problems which can occur with large systems. For example, if a synaptic cell is designed to have current output, the neuron cell, which will determine the value of the total current driven by all the synapses which are connected to this neuron, must be more accurate if 100 synapses are connected than if 10 are. This means that crucial electrical problems will occur in large networks, and that a lot of simulations and chip fabrications are needed before making a correct general design.

In the following discussion, Hopfield's fully interconnected neural networks will be studied. Generalization can be made of other types of neural networks, since the main problem consists of connecting a huge number of synapses to the neurons. Once this problem is solved, the same structure or design can be used for any type of neural network (feed-forward multilayer perception, classifier, Hamming net, etc.).

The first problem when implementing analog neural networks is the choice of the sort of memory points which will be used to store the synaptic weights. Before attempting to solve this problem, the number of precision bits needed per synaptic weight must be known: A neural network where only one or two precision bits are needed will have a totally different design than a neural network where a precision of 8 or 12 bits is needed. It is obvious that in the first case the size of a memory points will be less crucial than in the second one.

Digital memory points are of course the simplest to use; *digital* here means that only logic values can be stored in such memories, and thus that n identical cells are needed to store one n-bit synaptic weight. Static memory points are generally used, while dynamic ones also exist in some implementations. The reason why dynamic memory points are not often used is the following: Neural networks are in general asynchronuous; computations and lecture of the memory points can thus occur at any time, and the design of a refreshment system would enhance the complexity of the system. Since most of the actual chips implementing neural networks are prototypes and text chips, the design is maximally simplified, and dynamic memory points with a complex refreshment logic are generally avoided.

Analog memory points may be used when more than a one-bit precision is needed. Analog memory points consist of a capacitor storing an analog synaptic weight; the value of this capacitor is then multiplied in the synapse by the output of the connected neuron. An ideal capacitor would store an exact weight with an infinite precision. In a VLSI chip, we must of course cope with the technological limitations: Every capacitor has a leakage current, and its value must be refreshed periodically. The designer must therefore try to accord this circuit to the following rule: During the various cycles necessary of the convergence of the neural network, the percentage of the charge loss of the capacitor must be small enough so that its values before and after the convergence can be considered as exact as the precision needed for the network. Several techniques are proposed to achieve this goal; they all rely either on technological improvements to reduce the leakage currents of the capacitors or on specific design in order to compensate such currents.

The technological improvements try to obtain accurate capacitors with a leakage current as reduced as possible; this difficulty is achieved with standard MOS devices. Special technological processes are then used, for example, the FLOTOX [12] which consists of floating-gate structures to implement capacitors with a retention time of several days. This solution presents the drawback that standard CMOS technology cannot support these supplementary technological steps without important cost increase, and that limits the use of this solution.

Another way to obtain high-precision capacitors is to adjust their design so that leakage currents will have no more effect on the behavior of the circuit. Since synaptic weights can, in general, be positive or negative, a convenient way is to use two capacitors [13]; the difference of their charges will determine the value of the connection. The charge loss is then determined by the difference of the leakage currents, which is much better than of the solution with only one capacitor. Transistors and small variable capacitors are added for weight update and decay (Figure 8.11). This solution can be combined with high precision capacitors with reduced leakage currents to obtain accurate synaptic weights.

Although the solution with analog synaptic weights is seductive to reduce the number of memory points when a precision of several bits is needed, the best solution would be to use only one digital memory point; of course, this offers less precision and the global performance will decrease. However, algorithms coping with this restriction can be used to program the network [14]. In such algorithms, only two or three different synaptic weights are needed without necessary performance decrease in comparison with networks where continuous synaptic weights are used.

Once the type of memory points has been chosen, the synapse must be designed; this will realize a logical function between the value of the neuron which is connected to the synapse and an internal weight stored in the memory points. First, the case where the output value of the neuron can take two different values and where the synaptic weight can take three is examined. The logical function which will be realized between the neuron output value (+1 or −1) and the synaptic weight (+1, 0, or −1) is either AND or XOR (both of these functions are supposed to give 0 when the synaptic weight is 0). However,

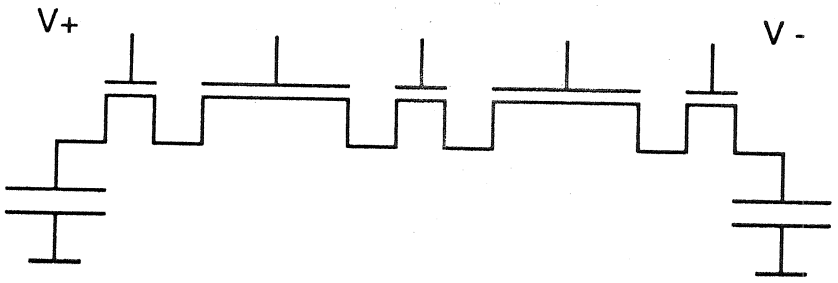

Figure 8.11. Capacitive memory points.

simulations showed [15] that in the general case the XOR function is preferred in a Hopfield's network, because the storage capacity is increased; the circuit shown here implements a XOR function, but can be easily transposed to an AND function.

Since three different synaptic weights are allowed, either one analog or two digital memory points are needed; the last solution is preferred because of the reduced number of precision bits. The architecture described below [16] is based on the principle of current summation: Synapses are programmable current sources, and all the synaptic currents are summed in the neuron, an amplifier which realizes the nonlinear transfer function (sigmoïd, threshold, etc.). One solution is to sum all the synaptic currents on the input line of the neuron [17]: If the synapse is excitatory (positive result of the XOR function), a current is sourced to this input line; if the synapse is inhibitory (negative result), a current is sunk from the input line (Figure 8.12).

If we suppose that the neuron realizes a pure threshold function, with the threshold equal to 0, it must be able to discriminate if the total current on its input line is positive or negative; this can be done by a simple amplifier. The major drawback of this architecture is the following: In standard CMOS technology, the currents trough P- and N-type transistors are very different. Compensation is achieved by adjusting the size of the transistors (usually a factor 2.5 . . . 3 between the two types of transistors); however, this ratio is never known exactly

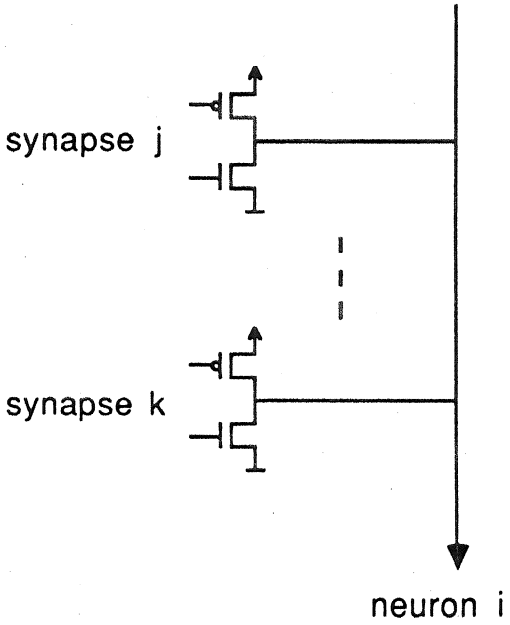

Figure 8.12. Synapse architecture.

before the technological processes, and the sourced and sunk currents are slightly different. When increasing the number of synapses connected to the same neuron, these differences will be summed, and the total mismatching can be greater than one synaptic current; the neuron can thus switch to a wrong state if the input current is close to the threshold. A solution to this problem is presented in [18]; it consists of summing all the excitatory currents on one input line, and all the inhibitory currents on another one; because of the two-line system, the same type of transistors can be used, and the mismatching problem no longer exists.

We briefly describe here an architecture of neurons and synapses which can be used for implementing networks with up to 500 fully interconnected neurons. The synapses is described in Figure 8.13.

Each synapse is a programmable current source controlling a differential pair. Three connection values are allowed in each synapse. If "mem1" = 1, current is delivered to one of the two lines with the sign of the connection determined by the product of "mem2" and the output of the neuron to which the synapse is connected. If "mem1" = 0, no connection exists between neurons i and j, and no current flows either to the excitatory or inhibitory lines.

Depending on the state of the XOR function, the current may be sourced either on the line $i+$ or on the line $i-$. In the neuron, the comparison of the two total currents on the lines $i+$ and $i-$ must be achieved. This is done by means of the current reflector shown in Figure 8.14. The currents on the lines are convert-

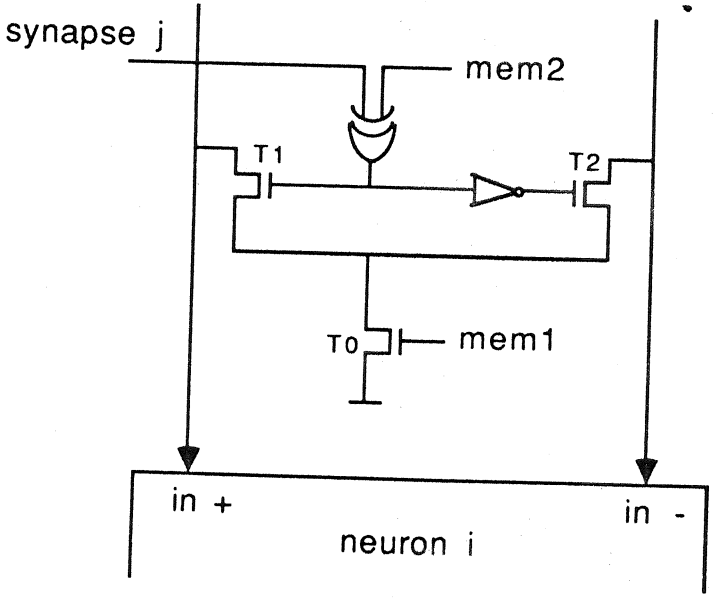

Figure 8.13. Synapse [18].

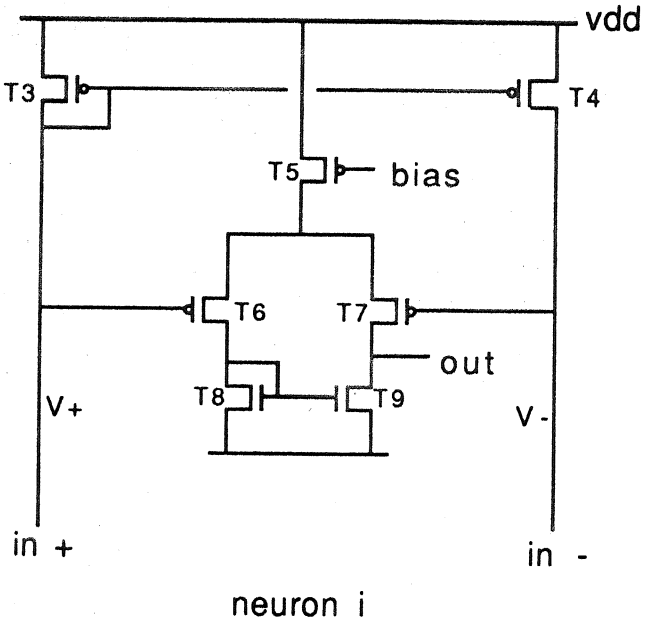

Figure 8.14. Neuron [18].

ed into voltages across transistors *T3* and *T4;* these voltages themselves are compared in the differential input reflector formed by transistors *T5* to *T9*. Because of the two-stage architecture of the neuron, the gain may be very large and the output (*out*) is either 5V if the current in *neuron* $i-$ is greater than the one in *neuron* $i+$, or 0V in the opposite case.

With increasing numbers of synapses connected to the same neuron, voltage drops V+ and V− across T_3 and T_4 tend to increase. Because V+ and V− are in fact the drain voltages of transistors T1 and T2 (Figure 8.13), saturation of the synaptic transistors must be avoided. A feedback loop was introduced to keep the voltage V* (Figure 8.15) fixed to Vref. While no high gain is needed for the feedback loop, the amplifier shown in Figure 8.15 can be very simple (even a single transistor can be used).

The architecture presented here is suited for networks up to about 500 fully interconnected neurons. The only limitation in the number of synapses which can be connected to the same neuron is the matching between the different synaptic current sources.

An analog neural network is not complete with only neurons and synapses. Some circuitry must be added for the programmation of the chip. First, all memory points contained in the synapses must be programmed; due to the high number of such memory points, this can only be achieved by an appropriate

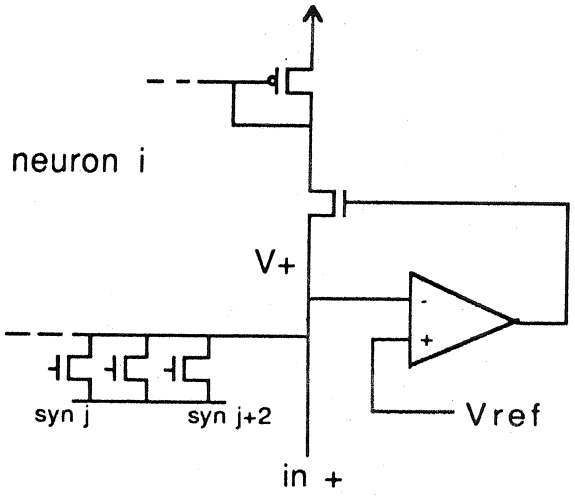

Figure 8.15. Feedback loop.

multiplexing of the programmation circuit. Secondly, although networks with hundreds of neurons can be considered with such circuits, it is more difficult to consider chips with hundreds of output pads. Another multiplexing must thus be realized at the output of the chip. Furthermore, it can be interesting to cascade several chips in order to realize bigger networks. For this purpose, each output of the chip representing the value of a neuron must be analog; for example, if four chips are connected together to double the size of a fully interconnected network, the inputs of the neurons (the sum of all the synaptic currents) must be connected, and not their outputs! The output pads of the chip must therefore be analog, and this enhances the complexity of the system. Finally, an analog neural network is normally asynchronuous; however, clocks and master-slave registers are generally added for testing purposes and for checking the different intermediate states of the network's convergence process.

Other analog neural networks have, of course, been realized and reported in the literature. J. Paulos and P. Hollis [19] presented a mixed digital-analog architecture to implement feedforward networks with integral back-propagation. The circuit for neurons and synapses is similar to the previous architecture, but the number of inputs and outputs is decreased by using an analog delay line. The complexity of the chip is thus reduced, but it limits the waveform processing to a sampling rate of 10 kHz.

An architecture for a switched-capacitor neural network (Figure 8.16) has also been presented by Y. Tsividis and D. Anastassiou [20]. They show how the techniques used in switched-capacitor filters can also be used to implement neural networks. Such circuit appears to have a number of advantages: No loop is

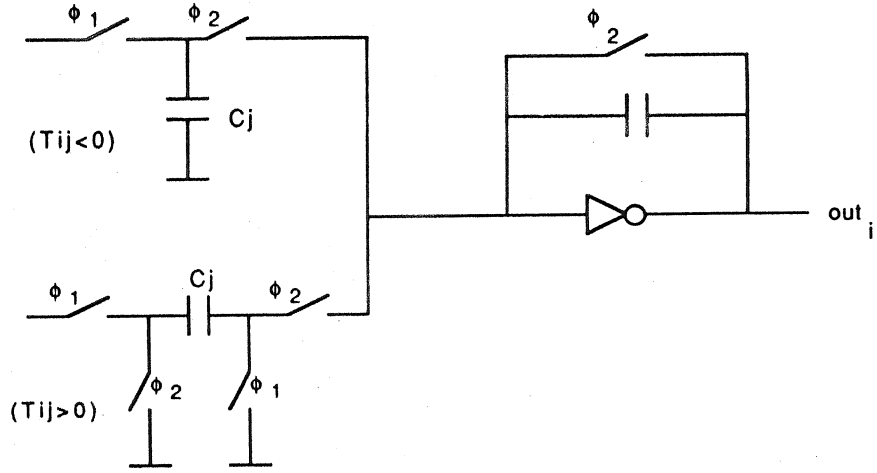

Figure 8.16. Switched-capacitor architecture.

ever closed, so the possibility of unwanted continuous-time oscillations is avoided. Secondly, the number of interconnections can be reduced, because multiplexing is easier in discrete-time implementations. Although this architecture presents a number of important advantages, implementations of switch-capacitor neural networks have not yet been reported in the literature.

To conclude this section on analog realizations of neural networks, we must mention the architecture proposed by A. Murray and A. Smith [21]. This circuit uses a "pulse stream" signaling mechanism that mimics that of a biological neural network. The output of the neuron is a stream of pulses; its frequency represents the value of the neuron. This stream of pulses is modulated in the synapse by the connection weight, and the pulses coming from all the connected synapses are added in the neuron. Although this architecture is more complicated than the other circuits described previously, it is much more insensitive to the matching of the components contained in the chip, and it can thus be a good solution for large networks.

4. CONCLUSION

The circuits described in this chapter represent some of the first distributed computation chips. They all differ in the methods of storing the connection weights, in realizing the sum-of-products, and by the input-output devices. However, they all try to reach a common goal—a high density of neurons and synapses.

Neural networks simulated on conventional computers show interesting properties unknown in other architecture; however, it is only with VLSI implementations that such properties will be fully exploited. The algorithms actually developed to program neural networks require, in general, a great number of neurons. The challenge is therefore to design networks with an important computational power, and to reduce as much as possible the complexity of the basic cells (neurons and synapses).

Distributed computation chips can be compared with RAMs and CPUs. In RAMs, data storage is important, but there is no processing power; on the contrary, CPU chips are essentially one very complex node. The common point between these structures is the total number of transistors, which only depends on the technology. Figure 8.17 shows that when the design of neural networks will be mature, they will contain about 10^3—10^4 synapses [22]; this proves that a lot of efforts still have to be made before the realization of a super-neurocomputer.

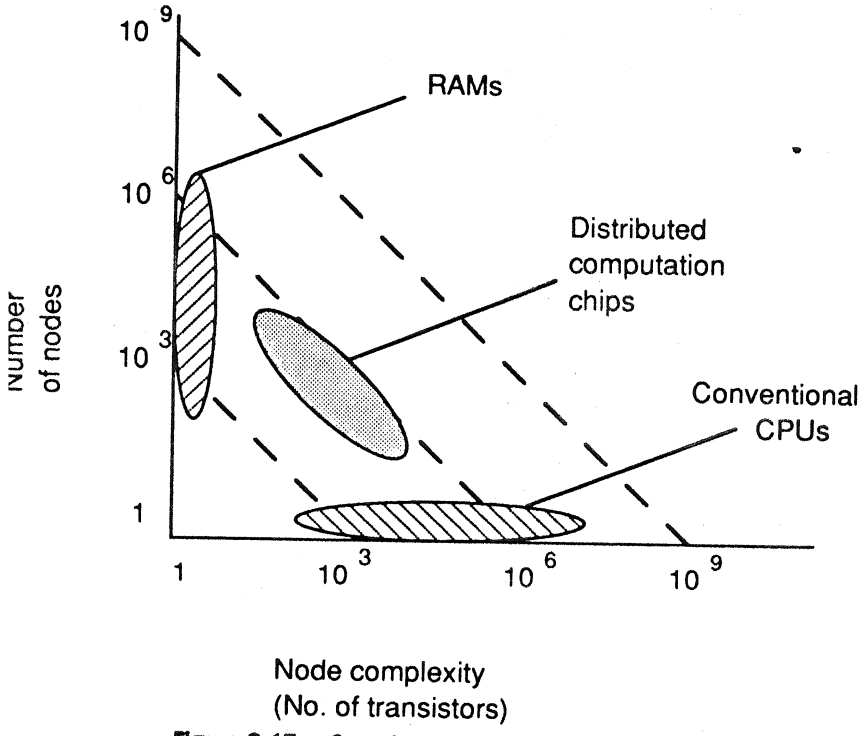

Figure 8.17. Complexity of neural networks.

REFERENCES

1. W.S. McCullogh and W. Pitts, "A Logical Calculus of the Ideas Immanent in Nervous Activity," *Bulletin of Mathematical Biophysics*, Vol. 5, 1943, pp. 113–133.
2. J. Raffel, "Electronic Implementations of Neuromorphic Systems," *Proceedings of the 1988 Custom Integrated Circuit Conference*. Piscataway, NJ: IEEE Publications, 1988, pp. 10.1.1–10.1.7.
3. J.J. Hopfield, "Neural Networks and Physical Systems with Emergent Collective Computational Abilities," *Proceedings of the National Academy of Science USA*, Vol. 79, April 1982, pp. 2554–2558.
4. R.P. Lippmann, "An Introduction to Computing with Neural Nets," *IEEE ASSP Magazine*, April 1987, pp. 4–22.
5. M. Sivilotti, M. Mahowald and C. Mead, "Real-Time Visual Computations Using Analog CMOS Processing Arrays," *Proceedings of the 1987 Stanford Conference on Advanced Research in VLSI*, P. Losleben (Ed.). Cambridge, MA: MIT Press, 1987.
6. R.E. Howard, D.B. Schwartz, J.S. Denker, R.W. Epworth, H.P. Graf, W.E. Hubbard, L.D. Jackel, B.L. Straughn and D.M. Tennant, "An Associative Memory Based on an Electronic Neural Network Architecture," *IEEE Transactions on Electron Devices*, Vol. ED-34, No. 7, July 1987, pp. 1553–1556.
7. S. Mackie and J. Denker, "A Digital Implementation of a Best Match Classifier," *Proceedings of the 1988 Custom Integrated Circuit Conference*. Piscataway, NJ: IEEE Publications, 1988, pp. 10.4.1–10.4.4.
8. M. Weinfeld, "A Fully Digital Integrated CMOS Hopfield Neural Network Including the Learning Algorithm," *VLSI for Artificial Intelligence*, J.G. Delgado-Frias and W.R. Moore (Eds.). Dordrecht, West Germany: Kluwer Academic Publishers, 1988, pp. 169–178.
9. F. Blayo and P. Hurat, "A VLSI Systolic Array Dedicated to Hopfield Neural Network," *VLSI for Artificial Intelligence*, J.G. Delgado-Frias and W.R. Moore (Eds.). Dordrecht, West Germany: Kluwer Academic Publishers, 1988, pp. 255–264.
10. J. Rasure and J. Salas, "A VLSI Three Layer Artificial Neural Network for Binary Image Classification," *Proceedings of the First International Neural Network Society Annual Meeting*. New York: Pergamon Press, 1988, p. 402.
11. B. Potu and P. Ramamoorthy, "A Fully Digital Architecture for Multi-State Neural Networks," *Proceedings of the First International Neural Network Society Annual Meeting*. New York: Pergamon Press, 1988, p. 400.
12. C. Bleiker, "Contribution to the Characterization of Programming, Storage and Retention in EEPROM-cells with Floating-gate Structures," Ph.D. Thesis, Eidgenössischen Technischen Hochschule Zürich, Zurich, 1987.
13. D. Schwartz and R. Howard, "A Programmable Analog Neural Network Chip," *Proceedings of the 1988 Custom Integrated Circuit Conference*. Piscataway, NJ: IEEE Publications, 1988, pp. 10.2.1–10.2.4.
14. B. Sirletti, M. Verleysen, A. Vandemeulebroecke and P. Jespers, "An Algorithm for Pattern Recognition with VLSI Neural Networks," *Proceedings of the First*

International Neural Network Society Annual Meeting. New York: Pergamon Press, 1988, p. 53.
15. J.J. Hopfield, "Neurons with Graded Response Have Collective Computational Properties Like Those of Two-State Neurons," *Proceedings of the National Academy of Science USA*, Vol. 81, May 1984, pp. 3088–3092.
16. M. Verleysen, B. Sirletti and P. Jespers, "A Large VLSI Fully Interconnected Neural Network," *Proceedings of the 1988 Symposium on VLSI Circuits*. Piscataway, NJ: IEEE Publications, 1988, pp. 27–28.
17. H.P. Graf and P. de Vegvar, "A CMOS Associative Memory Chip Based on Neural Networks," *Proceedings of the 1987 IEEE International Solid-State Circuits Conference*. New York: Pergamon Press, 1987, pp. 304–305.
18. M. Verleysen, B. Sirletti and P. Jespers, "A New VLSI Architecture for Neural Associative Memories," *Neural Networks from Models to Applications*, L. Personnaz and G. Dreyfus (Eds.). Paris: I.D.S.E.T., pp. 692–700.
19. J. Paulos and P. Hollis, "A VLSI Architecture for Feedforward Networks with Integral Back-Propagation," *Proceedings of the First International Neural Network Society Annual Meeting*. New York: Pergamon Press, 1988, p. 399.
20. Y. Tsividis and D. Anastassiou, "Switched-Capacitor Neural Networks," *Electronics Letters*, Vol. 23, No. 18, 1987, pp. 958–959.
21. A. Murray and A. Smith, "A Novel Computational and Signalling Method for VLSI Neural Networks," *Proceeding of ESSCIRC 1987*, Bad Soden (RFA), pp. 19–22.
22. M. Recce and P. Treleaven, "Parallel Architectures for Neural Computers," *Neural Computer*, R. Eckmiller and C.v.d.Malsburg (Eds.). Heudelberg: Springer-Verlag, 1988, pp. 487–495.

9
Neural Networks for Pattern Recognition*

Arun D. Kulkarni
Department of Mathematics and Computer Science
University of Texas at Tyler, TX

1. INTRODUCTION

Neural networks are a technology which has been inspired by studies of brain and nervous systems. The usage of neural networks for pattern recognition may be traced back to the perceptron models originated by Rosenblatt in 1950 [1,2]. The perceptron models used the concept of reward and punishment. In late 1960s, the progress in neural network models slowed down due to the limited capabilities of the early single layer perceptron models. In the mid-1970s and early 1980s, with the availability of enhanced computing power the progress in the development of neural network models accelerated. Researchers were able to model and test their theories about the functioning of the brain.

Today the number of well-developed theories and models of Artificial Neural Network (ANN) are available [3]. These networks consist of a large number of simple processing elements called neurodes that represent the neurons. These neurodes are interconnected by the synoptic connections. These models are capable of learning and making decisions; and are suitable for a variety of pattern recognition tasks. The classifiers in a pattern recognition system enable it to make the decision about the pattern presented to the classifier. Most of these decisions are based on the previous experience.

The ANNs consist of Parallel Distributed Processing (PDP) models. The PDP models are well described in the definitive work of Rumelhart and McClelland [4]. The functional synthesis of these models consists of establishing a relationship between the several inputs and one or more outputs. In ANN, the neurodes are connected to each other by the synoptic connections or the

* The author is thankful for the editorial assistance and for the comments and suggestions from the anonymous referee. The author is also thankful to Mr. Nishimura and Mr. Singh for their software development work.