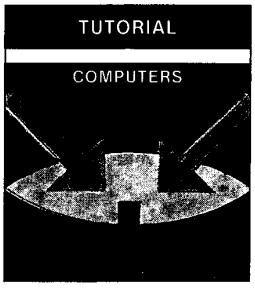
Neurocomputing: picking the human brain

Borrowing from biology, researchers are exploring neural networks—a new, nonalgorithmic approach to information processing



Imagine a computer that learns. Information is fed into it, along with examples of the conclusions it should be reaching or feedback on how it is doing—or the machine may even be left to its own devices. The computer simply runs through the material again and again, making myriads of mistakes but learning from them, until finally it gets itself into proper shape to carry out the task successfully. Such behavior is quite human, and naturally so; for the design of the machine's information-processing system, a neural network, was inspired by the structure of the human brain—its nerve cells, their interconnections, and their interactions—and by envy of what the brain can do.

As an alternative form of information processing, neurocomputing is fast becoming an established discipline, and some neural networks are already on the market. Neural networks are good at some things that conventional computers are bad at. They do well, for instance, at solving complex pattern-recognition problems implicit in understanding continuous speech, identifying handwritten characters, and determining that a target seen from different angles is in fact one and the same object.

Neural networks parallel-process immense quantities of information. Yet for a long time the only way to implement them was by simulating them laboriously, inefficiently, and at great expense on standard, serial computers. That situation is changing. Neurocomputers—hardware on which neural networks can be implemented efficiently—have reached the prototype stage at several companies, and some are already commercially available. All are coprocessor boards that plug into conventional machines. Developers include Hecht-Nielsen Neurocomputer Corp. (HNC), IBM Corp., Science Applications International Corp. (SAIC), Texas Instruments Corp., and TRW Inc.

Meanwhile, researchers at Boston University, the California Institute of Technology, the Helsinki University of Technology, Johns Hopkins University, the University of California at San Diego, and other universities have been investigating the theory behind neural networks and exploring their potential to solve problems that have stumped algorithmic computing for decades.

What is neurocomputing?

Nearly all automated information processing is at present based upon John von Neumann's "glorified adding machine" concept. But before such a computer can be programmed to carry out an information-processing function, some person has both to understand that function and to devise an algorithm for implementing it. For complex functions, such as computed axial tomography, development waits on the birth of geniuses capable of propounding the needed algorithm—in this case, Johan Radon and Alan Cormack.

Even worse, there may be tasks for which algorithms do not yet exist, or for which it is virtually impossible to write down a series of logical or arithmetic steps that will arrive at the answer. Yet in some of these cases it is possible to specify the tasks exactly and even develop an endless set of examples of the function being carried out. Many such tasks exist. There is no algorithmic software as yet for an automobile autopilot, a handwritten-character reader, a spoken-language translator, a system that can identify enemy airplanes or ships, or a system capable of recognizing continuous speech, regardless of who is speaking.

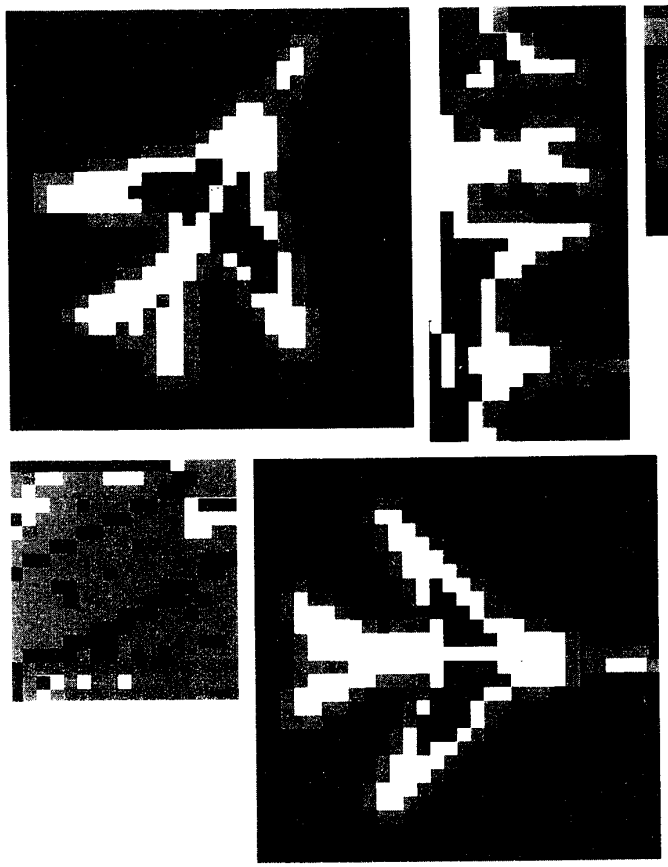
These tasks do have three important characteristics in common, however: humans know how to do them; large sets of examples of the tasks being carried out can be generated; and each task involves associating objects in one set with objects in another set. Such associations are known as mappings or transformations. For example, a computer that can read aloud must somehow associate groups of written letters, spaces, and punctuation with specific sounds, pauses, and intonations.

Robert Hecht-Nielsen
Hecht-Nielsen Neurocomputer Corp.

Defining terms

Adaptive coefficients: values of the previous computations (weights) of a processing element stored in its local memory, which modify subsequent computations.
Connection: a signal transmission pathway between processing elements, corresponding to the axons and synapses of neurons in a human brain, that connects the processing elements into a network.
Learning law: an equation that modifies all or some of the adaptive coefficients (weights) in a processing element's local memory in response to input signals and the values supplied by the transfer function. The equation enables the network to adapt itself to examples of what it should be doing and to organize information within itself, and thereby learn.
Processing element: an artificial neuron in a neural network, consisting of a small amount of local memory and processing power. The output from a processing element is fanned out and becomes the input to many other elements.
Scheduling function: a function that determines if and how often a processing element is to apply its transfer function.
Transfer function: a mathematical formula that, among other things, determines a processing element's output signal as a function of the most recent input signals and the adaptive coefficients (weights) in local memory. The transfer function includes the learning law of the processing element.
Transformations: mappings or associations of objects or representations in one set (such as written words) with objects or representations in another set (such as spoken sounds) according to some rule, which is typically not known to a human programmer, but is implicit in the training data.
Weight: within a processing element, an adaptive coefficient associated with a single input connection. The weight determines the intensity of the connection, depending on the network's design and the information it has learned.

0018-9235/88/0300-0036$1.00©1988 IEEE



Michael Myers, TRW Inc.

Types of aircraft can be identified with 95-percent accuracy by the TRW Mark IV neurocomputer. The input image of an airplane taken from directly overhead with the nose at any angle (top left), *is digitized into picture elements (pixels), and their brightness is color coded. A neural network takes each pixel from the image, calculates its radius from the center and its angular coordinate on a polar scale, and enters this information on a graph* (top middle). *Next, the network does a Fourier transform on each vertical column, giving a processed image* (top right) *that is independent of the rotation of the plane in the input image. The processed image is fed into a second neural network, one trained to recognize images and classify them; this network mulls the problem* (bottom left), *picks a category for the new image, and produces one of the training images of a plane in that category* (bottom right). *(All the training images had the aircraft noses at the 3 o'clock position.)*

In formal terminology, neurocomputing is the engineering discipline concerned with nonprogrammed adaptive information-processing systems—neural networks—that develop associations (transformations or mappings) between objects in response to their 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 comparing its results to the examples. Through trial and error, the network literally teaches itself how to do the task.

Neurocomputing is a fundamentally new and different information-processing paradigm—the first alternative to algorithmic programming. Wherever it is applicable, totally new information-processing capabilities can be developed, and development costs and time often shrink by an order of magnitude.

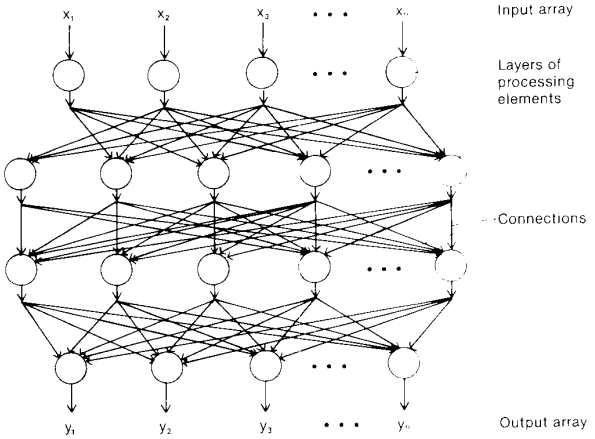
Neurocomputing does not, however, replace algorithmic programming. For one thing, neurocomputing is still in its infancy and is currently applicable to only certain classes of problems. More important, on a philosophical level, it is now suspected that neurocomputing and algorithmic programming may be conceptually incompatible. Transformations often prove impossible to describe satisfactorily in terms of an algorithm, and vice versa. This fact often unsettles people who think solely in procedural terms, since it means that neurocomputing may solve important information-processing problems without disclosing the rules used in the solution (at least in terms of present-day information-processing concepts). Neuroscience itself may run into this same problem in the future: an accurate understanding of how individual nerve cells interact in the brain may reveal virtually nothing about how brains process information.

How a neural network works

A neural network is modeled on the gross structure of the brain: a collection of nerve cells, or neurons, each of which is connected to as many as 10 000 others, from which it receives stimuli—inputs and feedback—and to which it sends stimuli. Some of those connections are strong; others are weak. The brain accepts inputs and generates responses to them, partly in accordance with its genetically programmed structure, but mainly through learning, organizing itself in reaction to input rather than by doing only by rote what it is told.

The neural networks used by engineers are only loosely based upon biology. At best, the only fair comparison is that they behave in a vaguely similar way. Since we have almost no idea how brains work, it will be a long time before we can re-create in a machine all the capabilities of the brain. Even so, neurocomputing is already offering some valuable, specialized, brain-like capabilities that in all likelihood lie beyond the reach of algorithmic programming.

A neural network consists of a collection of processing elements. Each processing element has many input signals, but only



Conceptually, a neural network consists of many processing elements (circles), *each connected to many others. An input array, or sequence of numbers, is entered into the network. Each processing element in the first layer takes a component of the input array, operates on it in parallel with the other processing elements in the layer according to the transfer function, and delivers a single output to processing elements in a layer below. The result is an output array representing some characteristic associated with the input. Since inputs and adaptive coefficients (weights) can change over time, the network adapts and learns.*

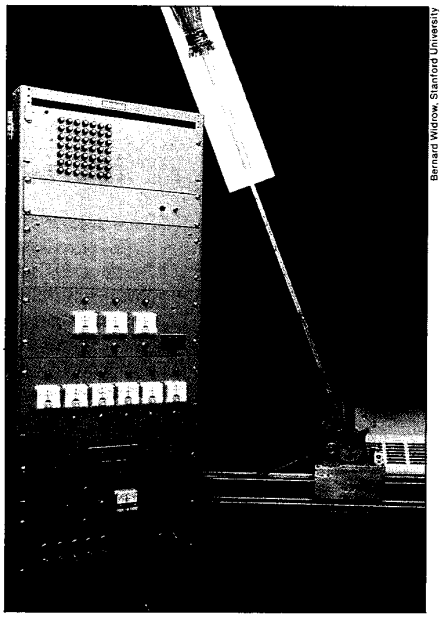
a single output signal. The output signal fans out along many pathways to provide input signals to other processing elements. These pathways connect the processing elements into a network [see figure, lower right, p. 37].

Each processing element typically has its own small local memory, which stores the values of some previous computations along with the adaptive coefficients basic to neural-network learning. The processing that each element does is determined by a transfer function—a mathematical formula that defines the element's output signal as a function of whatever input signals have just arrived and the adaptive coefficients present in the local memory. Often a neural network is divided into layers—groups of processing elements all having the same transfer function.

Depending on the design of the neural network, the processing elements either operate continuously or are updated episodically. A scheduling function determines in which way and how often each processing element is to apply its transfer function.

Each processing element is completely self-sufficient and works away in total disregard of the processing going on inside its neighbors. In any neural network a great deal of independent parallel computation is usually under way.

At the same time, all the processing elements intimately affect the behavior of the entire network, since each element's output becomes the input to many others. The topology of the connections among processing elements influences what information-processing functions a neural network can carry out, as it determines what data each processing element receives and therefore the information on which it can act.



Bernard Widrow, Stanford University

Broomstick balancing has become a classic test of a neural network's performance in adaptive control. The original experiment, conducted in 1962 by Bernard Widrow at Stanford University (now professor of electrical engineering), used his Madaline (Multiple ADAptive LINear Elements) neurocomputer (left)*, with sensors on the broomstick and cart indicating position, angle, velocity, and acceleration.*

By and large, every connection entering a processing element has an adaptive coefficient called a weight assigned to it. This weight, which is stored in the local memory of the processing element, is generally used to amplify, attenuate, and possibly change the sign of the signal in the incoming connection. Often, the transfer function sums this and other weighted input signals to determine the value of the processing element's next output signal. Thus the weights determine the strength of the connections from neighboring processing elements.

Learning without programming

The weights, moreover, are not fixed but may change. Most transfer functions include a learning law—an equation that modifies all or some of the weights in the local memory in response to the input signals and the values supplied by the transfer function. In effect, the learning law allows the processing element's response to change with time, depending on the nature of the input signals. It is the means by which the network adapts itself to the answers desired and so organizes information within itself—in short, learns.

A neural network learns how to process information usually by being given either supervised training or graded training. In both, it runs through a series of trials. In supervised training, the network is supplied with both input data and desired output data (correct answers as examples). After each trial, the network compares its own output with the right answers, corrects any differences, and tries again, iterating until the output error reaches an acceptable level. In graded training, the network is given input data but no desired output data; instead, after each trial or series of trials it is given a grade or performance score that tells it how well it is doing.

In either case, after training, the network is ready to process genuine inputs. At this point, depending upon the task to be done, a human operator may disable the learning law and "freeze the weights" of the connections, so that the network will stop adapting itself to new data and speed up its processing. For example, if the network has been well trained to read aloud from written text, it need not continue to learn on real data. On the other hand, if the network is to control the attitude of an orbiting spacecraft whose mass will decrease as fuel is spent, it should continue to adapt itself to changing conditions.

From concept to hardware

Neural networks are varied. At least 50 different types are being explored in research or being developed for applications. Of these, 13 are in common use [Table I]. Although all consist of processing elements joined by a multiplicity of connections, they differ in the learning laws incorporated into their transfer functions, the topology of their connections, and the weights assigned to their connections. In fact, some neural networks that learn may not be trained (self-organizing map) and some do not even learn at all (Hopfield network). The result: a host of networks, each suited to different types of tasks.

Any neural-network architecture can take different physical forms: electronic (in which everything consists of electronic devices and circuitry), electrooptical (in which optical signals link electronic processing elements), or entirely optical (in which light signals link optical processing elements made out of some nonlinear optical material). Experimenters at various institutions are working on all of these approaches. But problems with materials make the physical hardware of a network less straightforward than the conceptual architecture ["Neural networks: the physical reality," p. 40].

A number of neurocomputers—specialized machines able to efficiently and cost-effectively implement neural networks—have been built, and a few are now becoming commercially available [Table II]. All of them are configured as coprocessors to a standard serial computer, which acts as the host. The neurocomputer coprocessor, usually a board that looks much like any other circuit board, is connected to the host through a shared data bus or through a standard peripheral interconnection—PC-bus, Ethernet, or DRV-11.

As coprocessors, neurocomputers can be thought of as just another type of peripheral, like a printer or external disk drive. Data are shuffled into and out of the neurocomputer by the host computer through software routines supplied by the neurocomputer's manufacturer, even though the neurocomputer does its actual work on the data nonalgorithmically. No one has proposed a stand-alone neurocomputer—mainly because present networks do not handle input-output processing, and host computers already do that very well.

Most of the neurocomputer coprocessors built so far have been hard-wired designs optimized for implementing one type of neural network or a small selection of types. A few neurocomputers—notably those developed at IBM, Texas Instruments, and TRW—can implement several classes of networks, and a very few—such

as those developed at HNC and SAIC—can implement essentially any neural network.

Such flexibility is valuable because the ability to quickly modify the neural network being used is critically important in research, application studies, and early applications. As with von Neumann machines, though, the more general-purpose a neurocomputer is, the slower it is. After the applications of neural networks are understood more fully, it may be possible to substitute specialized neurocomputers in those applications where only a particular network, or small range of networks, is needed.

For those who wish to get deeply involved in altering the structure and thus the behavior of a network, there are languages designed expressly for describing neural networks in a high-level, machine-independent way. The field is very new, and so far only four neurosoftware languages have been introduced: P3, Panspec, AnSpec, and Axon.

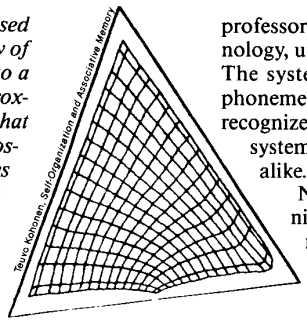
Neural networks applied

Several organizations are trying to apply neural networks to information-processing problems in commerce and industry that have proved intractable or far too expensive with algorithmic computers. Preliminary results have been encouraging.

Behavioristics Inc., Silver Spring, Md., has demonstrated a neural network for scheduling airline flights. Airlines sell seats at

I. Thirteen best-known neural networks

Name of network	Inventors and developers	Years introduced	Primary applications	Limitations	Comments
Adaptive resonance theory	Gail Carpenter, Northeastern U.; Stephen Grossberg, Boston U.	1978–86	Pattern recognition, especially when pattern is complicated or unfamiliar to humans (radar or sonar readouts, voiceprints)	Sensitive to translation, distortion, changes in scale	Very sophisticated; not yet applied to many problems
Avalanche	Stephen Grossberg, Boston U.	1967	Continuous-speech recognition; teaching motor commands to robotic arms	Literal playback of motor sequences—no simple way to alter speed or interpolate movements	Class of networks—no single network can do all these tasks
Back propagation	Paul Werbos, Harvard U.; David Parker, Stanford U.; David Rumelhart, Stanford U.	1974–85	Speech synthesis from text; adaptive control of robotic arms; scoring of bank loan applications	Supervised training only—correct input-output examples must be abundant	The most popular network today—works well, simple to learn
Bidirectional associative memory	Bart Kosko, U. of Southern California	1985	Content-addressable associative memory	Low storage density; data must be properly coded	Easiest network to learn—good educational tool; associates fragmented pairs of objects with complete pairs
Boltzmann and Cauchy machines	Jeffrey Hinton, U. of Toronto; Terry Sejnowsky, Johns Hopkins U.; Harold Szu, Naval Research Lab	1985–6	Pattern recognition for images, sonar, radar	Boltzmann machine: long training time. Cauchy machine: generating noise in proper statistical distribution	Simple networks in which noise function is used to find a global minimum
Brain state in a box	James Anderson, Brown U.	1977	Extraction of knowledge from data bases	One-shot decision making—no iterative reasoning	Similar to bidirectional associative memory in completing fragmented inputs
Cerebellatron	David Mar, MIT; James Albus, NBS; Andres Pellionez, NYU	1969–82	Controlling motor action of robotic arms	Requires complicated control input	Similar to avalanche network; can blend several command sequences with different weights to interpolate motions smoothly as needed
Counterpropagation	Robert Hecht-Nielsen, Hecht-Nielsen Neurocomputer Corp.	1986	Image compression; statistical analysis; loan application scoring	Large number of processing elements and connections required for high accuracy for any size of problem	Functions as a self-programming look-up table; similar to back propagation only simpler, although also less powerful
Hopfield	John Hopfield, California Inst. of Technology and AT&T Bell Labs	1982	Retrieval of complete data or images from fragments	Does not learn—weights must be set in advance	Can be implemented on a large scale
Madaline	Bernard Widrow, Stanford U.	1960–62	Adaptive nulling of radar jammers; adaptive modems; adaptive equalizers (echo cancellers) in telephone lines	Assumes a linear relationship between input and output	Acronym stands for multiple adaptive linear elements; powerful learning law; in commercial use for more than 20 years
Neocognitron	Kunihiko Fukushima, NHK Labs	1978–84	Handprinted-character recognition	Requires unusually large number of processing elements and connections	Most complicated network ever developed; insensitive to differences in scale, translation, rotation; able to identify complex characters (such as Chinese)
Perceptron	Frank Rosenblatt, Cornell U.	1957	Typed-character recognition	Cannot recognize complex characters (such as Chinese); sensitive to difference in scale, translation, distortion	The oldest neural network known; was built in hardware; rarely used today
Self-organizing map	Teuvo Kohonen, Helsinki U. of Technology	1980	Maps one geometrical region (such as a rectangular grid) onto another (such as an aircraft)	Requires extensive training	More effective than many algorithmic techniques for numerical aerodynamic flow calculations



Self-organizing map, a neural network devised by Teuvo Kohonen of the Helsinki University of Technology, can map a rectangular grid onto a nonrectangular shape. Here, the map is approximating a triangle, having arranged itself so that the squares remain equal in area so far as is possible. In modeling three-dimensional shapes (such as airframes) for numerical analysis, the network performs the mappings more accurately than most algorithmic techniques.

different fares depending how far in advance a reservation is made. The system, called the Airline Marketing Tactician, optimizes over time the allocation of seats between discount and standard fare classes to maximize the airline's profits. At present, several major airlines are considering the system.

Murray Smith, president of Adaptive Decision Systems Inc., Andover, Mass., has shown how accurately neural networks can score applications for bank loans. The network is given relevant data from a loan application form, and judges the applicant as a good or bad credit risk in terms of the examples on which it has been trained. The scoring system based on neurocomputing performed much more accurately than an existing operational point-scoring system based upon a combination of an expert system and a statistical model. A neurocomputer loan-scoring application has been developed for a major finance company, which plans to install it in the field around midyear.

A great challenge has been to get a computer to recognize human speech and translate it into written text—especially when a person speaks naturally, running words together in a continuous stream rather than articulating every syllable and pausing between individual words. Jeffrey Elman, associate professor of linguistics of the University of California at San Diego, has shown that neural networks can pick individual words out of connected streams and devise representations for them. The neural-network speech-recognition system with the highest accuracy and largest vocabulary was developed by Teuvo Kohonen, research professor in technical physics at the Helsinki University of Technology, under contract to Asahi Chemical Co. of Tokyo, Japan. The system has a front-end network that recognizes short, phoneme-like fragments of speech and a back-end network that recognizes strings of the fragments as words. A post-processing system uses context to distinguish between words that sound alike.

Neural networks have also broken ground in image recognition. Kohonen has shown that an associative memory network can take the image of a partially obscured face, complete it, and identify it with an image in a memory of 500 different people's faces. Kunihiko Fukushima, senior research scientist of NHK Laboratories in Tokyo, and Sei Miyake, a research director at the Automated Telecommunications Research Center in Osaka, have demonstrated a network, the neocognitron, that can identify hand-printed characters with 95-percent accuracy, regardless of shifts in position, changes in scale, and even small distortions.

Adaptive-control problems have been solved by neural networks for more than 25 years [see photograph, p. 38]. The early demonstrations have prompted efforts to apply neural networks to the practical "eye-hand coordination" of robot arms moving in response to feedback from camera images. Remarkable progress has been made by Andres Pellionez, research associate professor of physiology and biophysics of New York University, New York City, who has demonstrated the ability to move a finger of a robot arm in a beautifully coordinated perfect straight line at any angle.

Complementary to algorithmic computing

Algorithmic computing and neurocomputing complement each other nicely. The first is ideal for accounting, aerodynamic and hydrodynamic modeling, and the like. The second is ideal for pattern recognition, fuzzy knowledge processing, and adaptive control. Neurocomputers should not be used to balance checkbooks. Algorithmic computers should not be used to recognize speech. The two can also be integrated easily in hardware.

A technology succeeds, it seems, if it fits smoothly into an existing infrastructure and important applications are developed quickly. Neurocomputing appears to fill the bill. And if the his-

Neural networks: the physical reality

A neural network should in principle be simple to build. In the most basic networks, a processing element needs only to take all its incoming signals, multiply them by the weights of the connections over which they entered, add up the intermediate answers, and multiply the total by a nonlinear function to give its single output. If the incoming signals are voltages, then the weights can be represented by resistors, and by Ohm's law the intermediate answers are currents; by Kirchoff's law, all these currents can be summed by connecting the currents together at one terminal to give the output. Seemingly, then, that processing elements in the simplest neural networks should be resistors at the intersections of connecting wires.

A number of investigators—notably John Hopfield and others at AT&T Bell Laboratories—have indeed built neural networks of wires and resistors. But translating them into chips has proven difficult because it is virtually impossible to build accurate resistors on silicon wafers.

As a result, some neural-network engineers have resorted to "faking" accurate resistances on silicon. For example, Hans Peter Graf, member of the technical staff, and Larry Jackel, head of the device structure research department, at AT&T Bell Laboratories at Holmdel, N.J., have designed a combination analog-and-digital chip in which a digital gate at the crosspoint of the conductors acts in effect like a resistor with a value appropriate to the processing element. Carver Mead, the Gordon and Eddy Moore professor of computer science at the California Institute of Technology in Pasadena, has designed a neurocomputing chip on which each accurate linear "resistor" is simulated by a group of seven digital transistors. Jay Sage, staff member of the analog device technology group at MIT Lincoln Laboratory, Lexington, Mass., has designed a chip in which the same kind of field-effect transistor used in electrically erasable programmable ROMs acts as a resistor whose value is determined by charge buried under the gate.

Other neural network researchers have gotten around the problem by abandoning traditional VLSI processing materials. For example, Satish Khanna, technical group supervisor of advanced materials and devices section, and his colleagues at Jet Propulsion Laboratory, Pasadena, Calif., have devised resistors out of silicon hydride compounds, which are deposited on the spots where the rows and columns of connections meet. For any given network, current pulses sent along the pairs of vertical and horizontal conductors heat the spots of silicon hydride and drive off hydrogen, tailoring the resistance by up to three orders of magnitude.

Still other researchers have circumvented the problem by creative computation techniques. For example, in the Anza Plus neurocomputer manufactured by the Hecht-Nielsen Neurocomputer Corp., a processing element is not a physical entity at all. Instead, it is a slice of time on a VLSI digital arithmetic processing chip: its values are computed in a few microseconds, and then those values are stored in memory until needed again. Thus many processing elements share the same physical hardware, creating a "virtual network"—an approach that has proven to be economic, since VLSI is so fast and powerful. —*R.H.-N.*

II. Neurocomputers built to date*

Neurocomputer	Year introduced	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 U.	Commercial
Electro-optic crossbar	1984	Electro-optic	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, U. 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 2x	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, William 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 parentheses is with learning; speed inside 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.

tory of computing is any guide, the capabilities of neural networks are likely to grow with every successive generation.

To probe further

An excellent review of neurocomputing in the 1950s and 1960s can be found in *Learning Machines* by Nils Nilsson, McGraw-Hill, N.Y., 1965. James Anderson and Edward Rosenfeld have collected classic papers about neural networks in their book *Neurocomputing*, MIT Press, Cambridge, Mass., 1988.

David Tank and John Hopfield review their work in neurocomputing in "Collective Computation in Neuronlike Circuits," *Scientific American*, December 1987, pp. 104–114.

The IEEE 1988 International Conference on Neural Networks, cosponsored by half a dozen IEEE societies, will be held July 24–27 in San Diego. Contact Nomi Feldman, IEEE ICNN-88 Conference Secretariat, 3770 Tansy St., San Diego, Calif. 92121.

A new quarterly journal, *Neural Networks*, began publication in January 1988. A subscription to it is included in the annual membership fee ($45 regular, $35 student) of the International Neural Network Society, founded last year; for information, write to the society's secretary-treasurer Harold Szu, Naval Research Laboratory, Code 5756, Washington, D.C. 20375. The society will hold its 1988 annual meeting in Boston, Sept. 6–10.

The state of the art in neurocomputing is detailed in the four-volume *Proceedings of the IEEE First International Conference on Neural Networks*, held in San Diego, Calif., June 21–24, 1987. The set, IEEE Catalog No. 87TH0191-7, is available from the IEEE Service Center, 445 Hoes Lane, Piscataway, N.J. 08854.

About the author

Robert Hecht-Nielsen [M] is chair of the board of the Hecht-Nielsen Neurocomputer Corp. in San Diego, Calif. Before co-founding HNC with Todd Gutschow in October 1986, he founded and ran the neurocomputing programs from 1979 to 1983 at the Motorola Government Electronics Group in Tempe, Ariz., and from 1983 to 1986 at the TRW Electronics Systems Group in San Diego. He has B.S. and Ph.D. degrees in mathematics (1971 and 1974) from Arizona State University in Tempe. He teaches neurocomputing at the University of California at San Diego and is a member of the Del Mar Surf Club. ◆