Artificial Neural Networks

John J. Hopfield

Introduction

The following text is taken from an invited talk at the 1987 IEDM in Washington, DC, by Dr. John J. Hopfield, Professor of chemistry and biology at the California Institute of Technology. This talk has been transcribed for publication in *IEEE Circuits and Devices Magazine*. It covered the computational characteristics of neurobiological systems and attempted to show how these systems might be described in terms of equivalent electrical circuits. Silicon integrated circuits have been fabricated based on these circuit descriptions, and results have been obtained that are remarkably similar to the most simple neural networks. This is a fascinating field of study and illustrates the possibilities of improved understanding of neural systems by applying concepts from an apparently disparate field, namely, electrical circuits and computer science. And, of course, a long-term goal is to configure a computer to operate with the advantageous characteristics of neural networks.

Some Questions

I will attempt to answer five questions associated with artificial neural networks:

- Why are some people—who may be either visionaries or lunatics—interested in artificial neural networks?
- What are artificial neural networks, from the point of view of electronic circuits? How do such networks compute?
- How can they be programmed and how can they be made to solve particular problems?
- Can interesting problems actually be put on such networks?

Finally, I will turn to the current state of artificial neural network technology and the resulting challenges to people working on electronic devices.

To arrive at some of these answers, let's look first at the brain, shown with its "computer case" off in Fig. 1. It is a computer made of organic materials and wet chemistry but, nonetheless, is one of the world's best computers. There are problems in which, like multiplication, every data bit must be taken very seriously. Brains are very bad at such problems. There are other problems—such as the recognition of faces—which people do well but which digital electronic machines do very ineffectively, if at all.

One can try to make quantitative the idea of how good or how bad brains and digital machines are in computation. In so doing, however, one should factor out the issue of intrinsic device speed. For comparison, silicon devices have an intrinsic speed about 100,000 times faster than that of natural neurobiological devices. (For the sake of convenience, we can think about computation per clock cycle or characteristic time. Although neural biology has no such clock, it does have a characteristic time to make elementary decisions.) If we then compare—counting the hardware components used and multiplying by the number of clock cycles it takes to do things—we find that for arithmetic silicon VLSI—even discounting its speed—is a million times more effective than our brains.

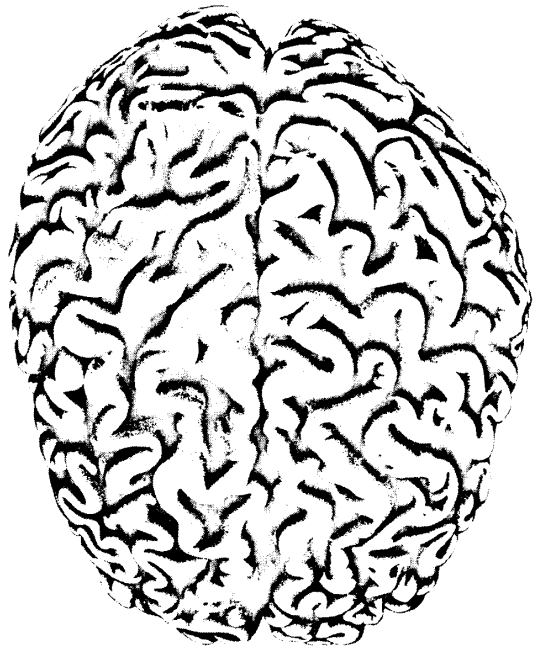

Fig. 1 A top view of the human brain. The computations are carried out in a surface layer about 2 mm thick.

On the other hand, for solving problems like face recognition—which a computer program has yet to achieve, but which even a pigeon can do—it becomes apparent that the neurobiological system is more effective by a factor of 10^8. Speed is not the essence of the problem: the pigeon can recognize a face in the equivalent of 50 clock cycles.

These facts illustrate why people are interested in artificial neural networks. Increasingly, computers are being asked to do tasks that were formerly only done by their biological counterparts. And digital computers, as often as not, can't do those tasks. The question then arises as to whether we can make electronic architectures that are more like neurobiological architectures and that are better than conventional architectures for addressing fuzzy and ill-defined problems, like face recognition or the decoding of natural language.

Architectural Problems

To begin, we must understand the essence of neurobiological architecture. To relate architecture to function,

8755-3996/88/0900-0003$01.00 © 1988 IEEE

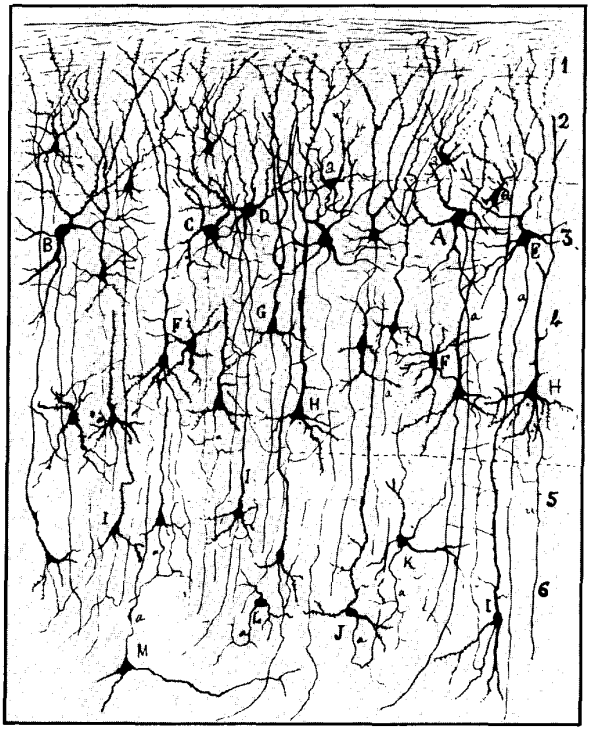

Fig. 2 A stained section of brain taken from cat cortex. The large bulges are the cell bodies; the thin processes are axons (outputs), and the knobby thick filaments are dendrites where inputs are made. The staining procedure makes about 1 percent of the cell totally visible while others are unstained.

one must start at the micro level. Figure 2 shows a magnified thin section of the cortex from a cat brain. The section has been stained so that only about one cell in 100 is visible. Each of the small knobs or projections corresponds to a point where brain cells (neurons) contact one another. Each neuron receives input from literally thousands of others.

A neuron can be studied as an input/output device. While details vary, neurons fundamentally transmit pulse-coded analog information. The input/output relation of this pulse-coded system is a simple sigmoid (Fig. 3), easily built electronically. It is *not* simply a linear amplifier. The sigmoid characteristic—the fact that the output saturates at both ends—is very important to neural computational properties and to computational properties in general. For the most part, linear systems don't make decisions and, thus, do not compute, although they are important adjuncts to computational systems.

Details, such as the structure and the electrical processing of neurons, can be examined and compared with silicon VLSI. Some differences, such as connectivity, are quite significant. In the typical VLSI circuit, the output current of a typical transistor feeds two or three other transistors and receives its input from a similar number. In neurobiology, that number is on the scale of 3,000 rather than 3, a qualitative difference. In addition, the connectivity pattern in digital processing is chiefly *feed-forward*, with the output of one logical decision feeding into the next logical decision. There is a clear forward direction of information flow. Neurobiology has an immense amount of *feedback* in

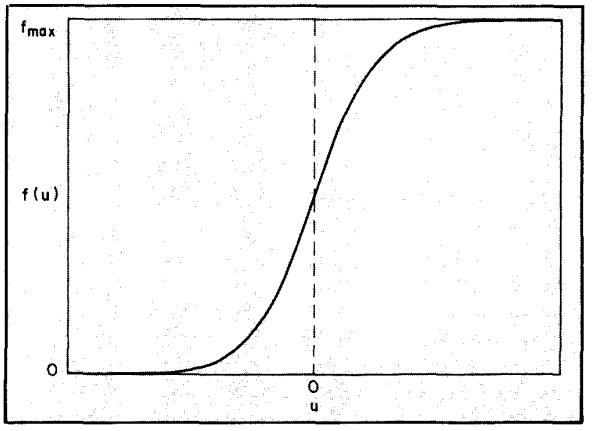

Fig. 3 The pulse code firing rate, or output, of a neuron as a function of the input current to the cell body.

its hardware, and biological information flow is very much two-way.

Examination of the VLSI mode of operation reveals a system clock, which drives and synchronizes logic circuits and logical states. Neurobiology behaves much more like an analytical, dynamic system. Neural system "computation" is described by an equation of motion, like those found in classical electromagnetic theory or classical mechanics.

The question of timing leads to the issue of time delays. Time delays are the enemy in digital VLSI. The digital approach is to make the delays shorter and shorter so that the chip can be flogged to go faster and faster. Neurobiology is intrinsically slow to begin with and has its own set of time delays. It makes a virtue of those delays, using the system's intrinsic time delays as part of its processing. One last point: while the VLSI electronic circuit is thought of as a fixed circuit, in neurobiology the structure of the circuit changes every time learning occurs.

All of the above-described features of neurobiology could be built into a VLSI. It would result in a very different system, however. Consider the differences on decision making due to just two of the aspects, large connectivity and feedback. The digital VLSI approach is like making a decision in a business that is organized like a pyramid. The neural approach is like an organization that is an interactive and communicating democracy without hierarchy. Decisions can be made by either, but the nature of the decisions can be qualitatively different. The neural systems tend to a single holistic decision on all the information rather than sequential logical decisions based on partial information.

Understanding Neurobiological Computations

To understand how neural networks compute, we will examine the electronic circuit of an artificial neural network. The elements of the hardware (Fig. 4) are a set of amplifiers with sigmoid I/O characteristics, input capacitance, output resistance, and a set of resistive connections ("synapses") connecting outputs of some amplifiers with inputs of others. Neurobiology has "amplifiers" of two types: normal ("excitatory") and inverted ("inhibitory"). Thus, each amplifier in Fig. 4 has been drawn with both possibilities and has both a normal and an inverted output.

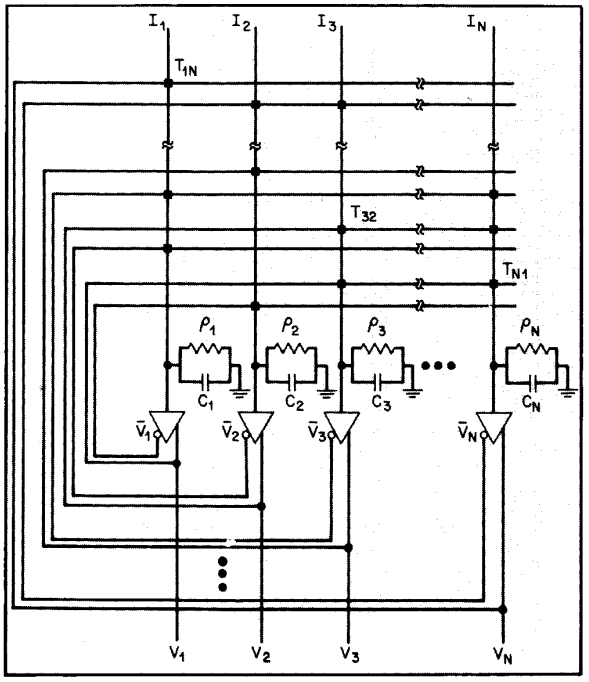

Fig. 4 The electrical circuit diagram of an artificial neural network; the dots are resistive interconnects describing T_{ij}. The input capacitance and resistance of the amplifiers are not shown.

Inputs can be delivered at the top in Fig. 3, and external outputs can be taken from some of the amplifiers. This completes our artificial neural network model. It is a parody of neurobiology but already encompasses many of the features that differentiate digital-logical circuits from neurobiological ones.

$$C_i \frac{du_i}{dt} = -\frac{u_i}{\tau_i} + \sum_j T_{ij} V_j + I_i$$

$$V_i = g(u_i)$$

This terse set of circuit equations describes the operation of this system. They look trivial but are not, because of the nonlinearity of the input/output characteristic. The computational program and knowledge of the system are all embodied in the set of conductive interconnects T_{ij} between the output of some amplifiers and the input of others. It is very important that these connections are of this finite impedance; they are not "wire" connects. In short, all of neurobiology's power comes from the fact that the T_{ij} are weak connections. The *analog* strength of the connections matters. A general circuit of this type is so complex that if appropriate special parameters are chosen, it can mimic any digital machine. Clearly, this is *not* the way neurobiology uses these equations. It is preferable to see how they compute in a more neurobiological way and to understand how such circuits compute in a way that does not depend on simply trying to make a VAX computer from them.

What Is Computation?

Think about computation from the standpoint of a digital circuit. A digital machine has logical states. As the system clock moves on, the machine moves from one logical state to another logical state. A computation is begun by starting at a particular logical state in the machine. That initial state is determined by the program and the input data. The machine moves along, following its intrinsic trajectory in its logical space. When it gets to the answer, it stops, and the answer can be read out.

Having described computation as a path in a logical space, let's look at the *real* digital machine, which is, in fact, an analog machine. If you think of the trajectory in analog space, then there is a lot of jitter due to noise and machine imperfection, but the trajectory remains constrained close to a logical line. There is a restoration, so that if the trajectory starts to deviate from the prescribed path, it gets pushed back onto it. That is really what a digital computer is all about and why it generally is not possible to make long computations on analog machines. To make a computation that goes on for a long time, there must be a restoration process. Otherwise, since the circuit is a collection of high-gain amplifiers, noise and imperfections are amplified, and the trajectory ends up nowhere near the right answer (Fig. 5).

Consider a dynamic system that starts somewhere in its own dynamic state space, moves along, comes to a stable state, and stops. Such a system could also be the basis for doing computations. This description of computation, which I introduced for neural networks, directly maps onto the earlier digital description of computation. It is very important to understand that in order to do an appreciable computation in an analog system, there must still be restoration so that the system gets back onto the appropriate path track, even though the circuit is not perfectly made.

The computational dynamics of typical neural networks are characterized by the existence of several stable states. Those states correspond to answers, or possible answers, of problems. The appropriate circuits (in the simplest cases) have stable states. Once the initial inputs are given, the time evolution of the dynamics leads to a stable state, whose voltages describe the answer.

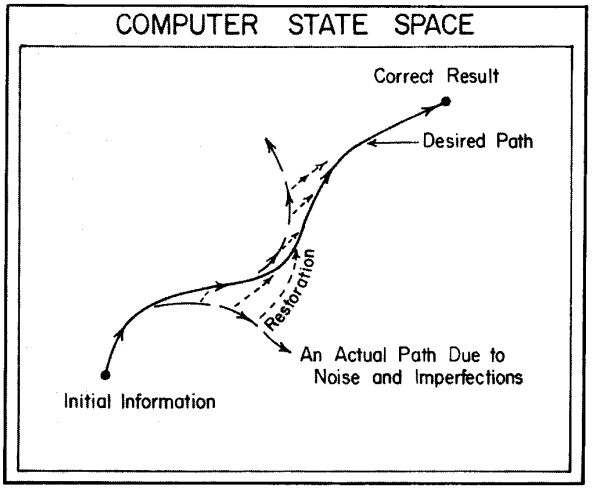

Fig. 5 The state-space view of computation. Batch mode computation is performed by choosing an initial state; the "answer" is the final point reached by following the trajectory in time. Restoration is required to keep the system on the correct path in spite of noise and imprecisions of manufacture.

There are two quite different circuits consisting of amplifiers and interconnects between the amplifiers, which can be made to show stable states. Figure 6a shows a trivial class. It is simply a feed-forward circuit—amplifier output goes to the input of following amplifiers, and so on. The network is trivially stable because there is no feedback. This is not a powerful system from the point of view of computation, but it turns out to be quite powerful from the standpoint of being able to program such a network through a learning procedure to do a useful task.

Figure 6b shows a system that is intensely fed back: the outputs of a set of amplifiers return to become inputs with an interconnection matrix T_{ij}. This is another kind of system that can be made stable, for example, if the connection matrix is symmetrically designed. This kind of circuit is much more powerful, and has richer dynamics, but is harder to program. Both networks are interesting from the point of view of artificial neural computation. Technologically, they emphasize the same things, a set of sigmoid I/O amplifiers and a far larger set of interconnects that describe the relationships between the output of some amplifiers and input to others. Building useful circuits will in the long run require combining aspects of both classes.

Why does a class of circuits have stable states? Imagine the trajectory dynamics or flow map drawn in only two dimensions—as in Fig. 7a—of a network with stable states. Starting anywhere, the path moves to a stable state at which the neighborhood motions are all inward.

The existence of stable states and flow patterns in state space can be understood once through a mathematical quantity behind the "map" that explains why points are stable. Stable states are like moving downhill on a surface and stopping at a valley bottom. Symmetrically connected

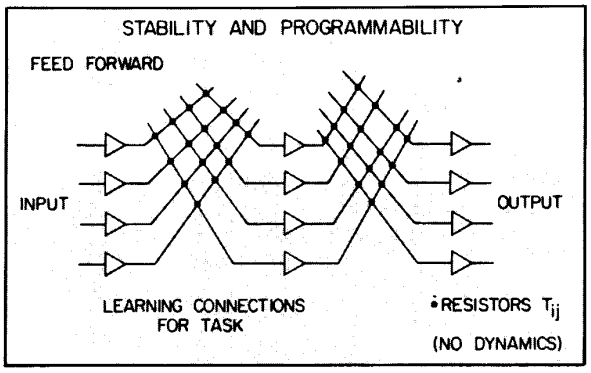

Fig. 6a A feedforward network automatically has stable states in its dynamics.

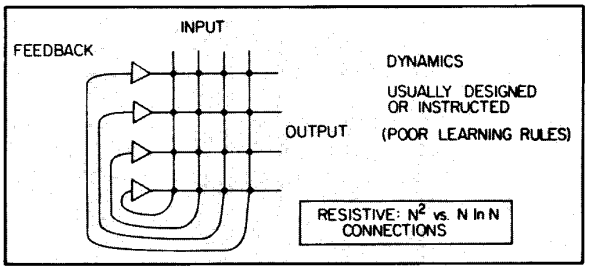

Fig. 6b A highly fed-back network. Some classes of feedback networks also have dynamics described by stable-state attractors.

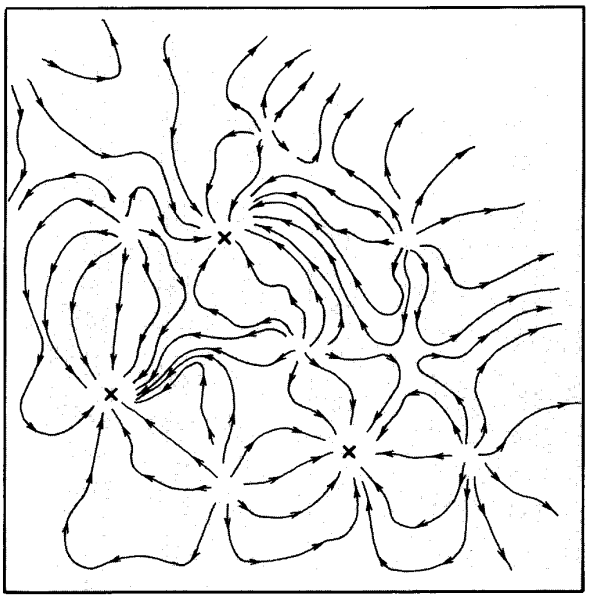

Fig. 7a The trajectories of the state evolution with time, or state-space flow, for a stable feedback network.

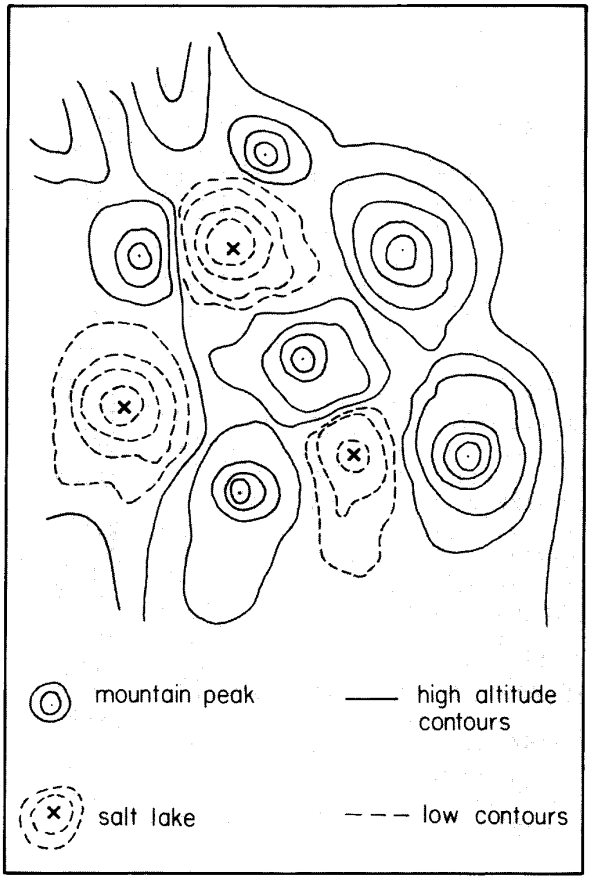

Fig. 7b The energy function that allows us to understand why the state-space flow has stable-state characteristics. The circuit dynamics always move the state of the circuit downward on this contour map of the energy terrain.

networks automatically have this kind of Lyapunov function or "energy function" (Fig. 7b) behind their dynamics and automatically have a stable state. Much larger classes of feedback circuits can also be shown to have E-function behavior as their basis of stability.

What computations can be done with these stable states of symmetric networks? In the simplest instance, a system with many stable states is a kind of memory. The coordinates of the stable state determine the actual thing known, a list of numbers. How is the memory state accessed? It is clear that if the initial state for a computation is in the general vicinity of a particular stable state, the neural network circuit will settle down to exactly that nearby stable state and stop moving.

There are two kinds of memory. In ordinary digital machines, information is put in a particular memory location or address and the hardware looks at this location to retrieve the information. If the address is unknown, but part of the information stored in the particular memory is known, the partial information cannot be used to access the memory. The only way to access the information is by actually knowing a specific address.

Human memory, on the other hand, is very different: Suppose you went to a party last Saturday night, met a group of people and got into a heated political debate. Three weeks pass. When someone reminds you of a few facts about the party, you easily recall many details from that evening. There is no notion of an address to that memory—it is an *associative* memory. Any reasonable subset of the information about the event is sufficient to enable access. A great deal of what passes for intelligence in the way of human behavior turns out, on closer inspection, to be a (massive) associative memory at work.

A reasonable subset of information about a memory can be viewed as a location in information space that is nearer to the desired memory than it is to any other memory. A circuit with dynamics like that shown in Fig. 7a automatically has the capability of being an associative memory. This explains how a difficult-seeming problem of making an associative memory can be readily done in biology. The first massive artificial neural network hardware has been associative memories.

We will illustrate an associative memory at work by simulating an artificial neural network. In this example, 28 neurons were used to remember four seven-digit telephone numbers. The knowledge of the memories is embedded in the strength of the T_{ij} connections. This embedding is nonlocal, in that each connection contains information about all four telephone numbers, and reconstructing a particular telephone number involves all connections. Figure 8 illustrates the operation of this associative memory.

How can artificial neural networks be programmed to solve more general problems? David Tank of AT&T Bell Laboratories and I have used *the existence of energy functions* as a programming tool for problem solving. Many problems can be described as *optimization* problems. For example, consider the problem of recognizing someone. What is really being asked is, of all my acquaintances, which one of those faces most closely resembles the face I am looking at? What maximizes the "overlap" between what I see and the people I know? Recognition can thus be described as a maximum (or minimum) problem. Questions like "What route should I take to drive home?" or "Whom should I

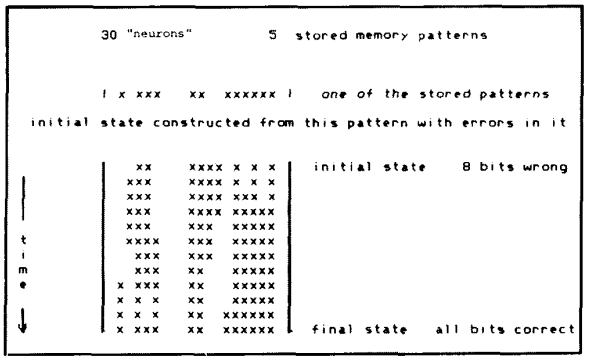

Fig. 8 The operation of a 28-neuron associative memory operated in the high-gain limit. The connections have been chosen to store four telephone numbers (above). A starting state is chosen with three digits (in bit representation) taken from one of the numbers and the others chosen at random. As time goes on, the dynamics change the incorrect bits into correct ones, retrieving the correct number as the final stable state.

vote for?" are intrinsically optimization problems, in which some definable quantity is being maximized or minimized. Since networks with E-functions intrinsically try to minimize a mathematical function, if you can map the item to be minimized onto the network in a direct way, then it can find the minimum by obeying its own dynamics. Its spontaneous behavior will do the computation for you.

David Tank and I used the old computational chestnut, the "traveling salesman" problem, to demonstrate that problems of this difficult class could be put on the network. The problem is a familiar one: there are many cities to visit. The salesman must visit each and return to the starting point. The task is to choose the shortest overall path, a so-called minimum problem, which might be put on the network. As with any computer, there must be some kind of syntactical agreement about how computer output is to be interpreted. The neural network to solve a "traveling salesman problem" involving the 10 cities requires 100 amplifiers. The network was constructed in such a way that in any stable state each of the amplifiers was either strongly "on" or all the way "off." The possible *final* states of the system are thus actually digital, with each amplifier having an output of 1 or 0. Each of the 100 amplifiers in a 10×10 array represents one proposition. For example, amplifier D1 represents the proposition that you should visit city D first. If in the final stable state the output of that amplifier is a 1, it means that the proposition "visit D1 first" is true. If output D1 is a 0, the corresponding proposition is false. The final state shown in Fig. 9 takes the salesman first to A, then E, then G, and so on.

We designed a set of resistive connections so that the set of amplifiers had stable states that represented closed paths that visit each city once. An additional set of connections with conductances whose numerical values were equal to the distances between the cities provided the network with the information about the locations of the cities. Then, not knowing what the answer was, we started the system in a random unbiased state and allowed it to progress to an answer.

This kind of computation does not guarantee finding the best answer. What the network does is to find a very good answer very quickly. If the problem is easy, the arrangement of the cities simple, then the best answer can readily

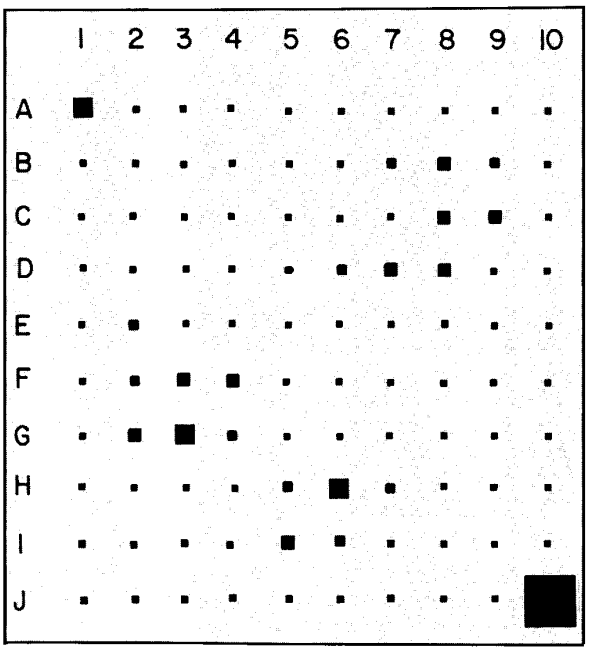

Fig. 9a A state of the network during its convergence to a good answer. Many amplifiers are partially turned "on." The interior of the analog domain is used in finding a good solution, though the ultimate solution found will be binary, with each amplifier either strongly "on" or strongly "off."

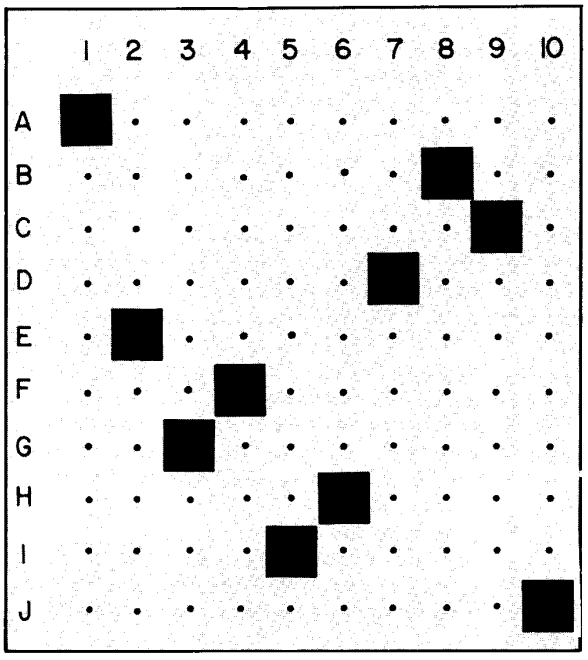

Fig. 9b A 10 × 10 array of amplifiers for solving the traveling-salesman problem. The large squares represent amplifiers with an output of 1; the small squares 0. This state represents the path A E G . . . (after Hopfield and Tank).

be found. If the case is subtle, and very different paths are almost equal in length, then only a good path will be found. In most parts of neurobiological computation, a similar situation is believed to prevail in real-world situations. The usual problem in face recognition is that there is one answer that is overwhelmingly good. An idiot would be able to find it. The difficulty is trying to get a machine computation to weigh the massive evidence and to come to a single holistic conclusion on the basis of all the data.

It is instructive to see how the "traveling-salesman-problem" answer is found by the network. In Fig. 9a, the size of the square represents the degree to which an amplifier is turned on. During the convergence process, an analog state is reached that cannot be described either as "City F should be visited third" or "City F should be visited fourth," since amplifiers have outputs that are between 0 and 1. The state of Fig. 9b could be described as partial belief in a set of similar but contradictory propositions. This analog dimension is necessary to good computation in this problem. If the gain is too large, the circuit is pushed into a digital domain and paths no better than random are found. If you let the circuit move through the *interior* of the logical space, it finds good solutions.

Turning a real-world problem, which is where one would like to apply such networks, Figs. 10a and 10b illustrate one of the difficulties in identifying words in spoken speech. The pressure wave and sonogram of the spoken phrase "six seven nine" is shown. The idea is to recognize those digits as they are spoken. Recognizing a particular word presents various problems. One is the fact that different speakers say words differently, and even a given speaker will say a word differently each time the word is uttered. A second problem arises in determining what constitutes a word. Words aren't spoken separately. Worse, they don't just run up against each other, they overlap. It is, therefore, computationally difficult to pick words out of a natural stream of speech. There are many ways to approach this problem. Tank, Unnikrishnan, and I have found that it is possible to make a neural network that is capable of taking a data stream and recognizing "679" with high reliability in continuous speech. To do so, it is necessary to use all the tricks I have described so far, and one more.

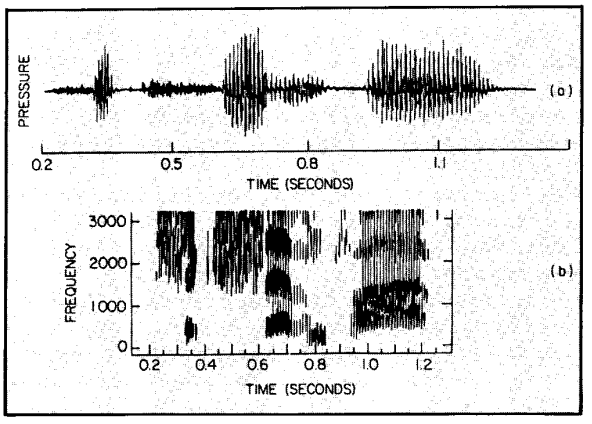

Fig. 10 The pressure waveform (a) and power spectrum versus time (sonogram) (b) of the spoken phrase "six seven four." (Courtesy of K. Unnikrishnan.)

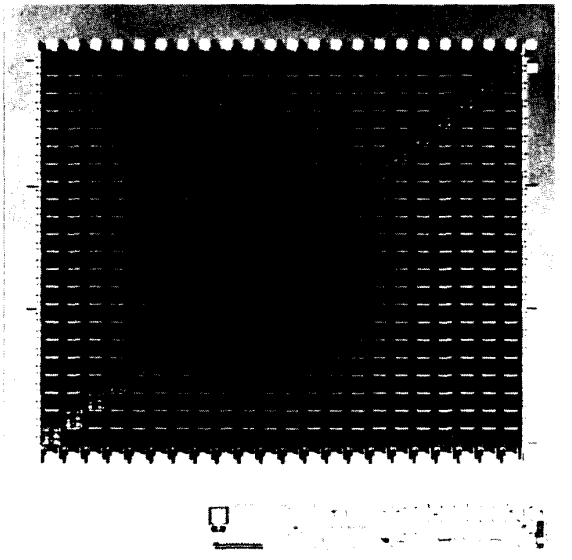

Fig. 11 The VLSI associative memory chip designed by Emmerling, Sivilotti, and Mead. It is approximately 6 mm square, and contains 22 "neurons" and analog connections between them which can be set to connection strengths 1, 0, or −1.

Time delays can help organize information. To recognize an entire word, you must somehow remember at the end of the word what the beginning of the word was like 0.7 sec. ago. One of the easiest ways of doing that is to use organized time delays, which is rather like going back to old-fashioned mercury delay lines to store information. Since there are time delays in the "hardware" of neurobiology, evolution is free to make use of them for functional purposes. This is the one significant additional feature that we have added to other neural ideas for word recognition.

The neural circuit for word recognition of a small vocabulary circuit is completely analog in concept and operation. It has no clock; the computation of a recognition is driven along in the processing of the input signal itself. While most of the speech work was done in simulation, a small electronic network of op-amps, resistors, and capacitors was also constructed and functions as expected.

Finally, let me turn to the state of artificial neural network technology. The first circuit that looked at the stability of intensely fed-back networks was made from lumped circuit components by John Lambe at the Jet Propulsion Laboratory. The circuit was physically very large, but it demonstrated that highly fed-back "neural" circuits could be stable. Stability is not an artifice of pure mathematics. The realities of circuits do not bring up delicate issues, which lead to instability.

The first microfabrication of an associative memory neural network by Carver Mead, Michael Emmerling, and Massimo Sivilotti at California Institute of Technology is pictured in Fig. 11. The chip they made transferred the whole neural net idiom into elements that were easy to build from a VLSI point of view. Resistors and capacitive summations in our earlier discussion were replaced by current sinks, current sources, current mirrors, and so on. The VLSI circuit was then an analogy to what I have described, not a faithful copy.

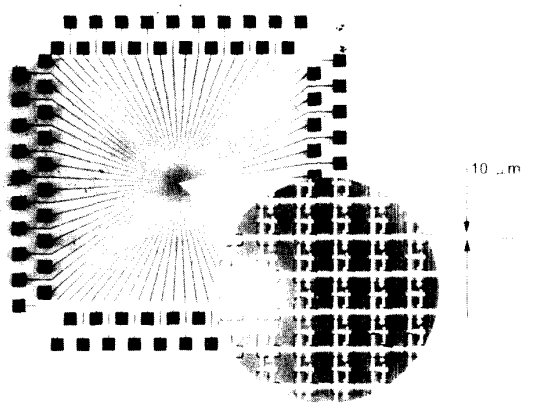

Fig. 12 A network of resistive connections with antifuse connections made from α-silicon. The insert is a magnified image of a piece of the chip. (Courtesy of Thakoor, Lambe, Moopen, Lamb, Khanna.)

The Future

L. Jackel, R. Howard, and H. P. Grof at AT&T Bell Laboratories have made similar but larger chips. By using a technology in which resistive elements can be made, they have placed 256K of connections and 512 neurons on a single chip with 2-μm line rules. There has been difficulty in practical fabrication in this case because the usually available VLSI fabrication does not include the resistive elements. However, the difficulties are not insuperable.

We would eventually like to have a programmable set of connections, not just a set of connections for a ROM or for a fixed processor. Figure 12 shows one way to go about this as carried out at the Jet Propulsion Laboratory by Anil Thakoor, John Lambe, Alex Moopenn, and Satish Khanna. A thin-film matrix of 1,600 synapses for a programmable read-only memory neural network based on memory switching is fabricated by using hydrogenated amorphous silicon. The wires in one direction are for inputs, and in the other direction for outputs. At each connection there is a resistive link put in series with a link of amorphous hydrogenated silicon. Amorphous hydrogenated silicon is normally an insulator, but if high voltage is applied across one of the intersections, then that particular bit of amorphous hydrogenated silicon breaks down and becomes a quasimetallic connection. The resistor is then written into the circuit. It is important in this regard to realize that these circuits intrinsically involve high-impedance, weak connections. It is desirable to have connections that have impedances on the scale of 1 to 100 mΩ.

While time does not permit detailed discussion here, floating gate technologies represent a way to reach high synapse densities. But what we would really most like to have is a technology in which you can put down an array of wires in one direction, covered by a layer of material, and then an array of wires in the other direction, a self-registering technology. The intervening layers should have resistive properties so that if large voltages were placed across it, the resistive connection at that point is altered. The ideal set of connections should also have continuously adjustable conductivities. The attractiveness of this approach is the potential for very high densities of connec-

tions. Some major breakthroughs in materials technologies will be necessary to achieve this goal.

Conclusion

In summary, the field of artificial neural networks tries to simulate and to fabricate networks and devices in the spirit of neurobiology, to solve useful computational problems of the kind that biology does effortlessly. In considering neurobiology, John von Neumann, the father of the stored program digital computer, wrote "All this will lead to theories of computation, which are much less rigidly of an all-or-none nature, than is formal logic. They will be of a much less combinatorial, and much more analytical, character." For neurobiology, I am certain he is right. Those of us who are working on artificial networks hope that there are direct lessons from neurobiology that can be usefully realized in electronics.

John J. Hopfield received his A.B. from Swarthmore College in 1954 and his Ph.D. in physics from Cornell University in 1958. Following two years at AT&T Bell Laboratories, he held various positions at the Ecole Normale Superieure, Paris; University of California, Berkeley; Princeton University; Bohr Institute, Copenhagen; Cavendish Lab, Cambridge, England; and is presently dividing his time as a Professor of chemistry and biology at California Institute of Technology, Pasadena, California, and a Member of the Technical Staff at AT&T Bell Laboratories in Murray Hill, New Jersey. He is a Member, American Academy of Arts and Sciences; Member of National Academy of Sciences; Fellow, American Association for the Advancement of Science; Fellow, American Physical Society and has received numerous awards for his work in solid-state physics and neural networking. These awards include the Buckley Prize in 1969 and the Biological Physics Prize in 1985, both from the American Physical Society; the John and Catherine T. Mac Arthur Award in 1983; and the Michelson-Morley Award in 1988.

His present research interests include studies of the physics of biomolecular functions and neural networks.