Modeling the Brain

A neural-network approach to natural-language processing and similar problems may be the key to building systems that "learn"

Matthew Zeidenberg

THE IDEA OF simulating the brain formed the foundation for much of the early work in artificial intelligence. The brain was seen as a "neural network," that is, a set of nodes, or neurons, connected by communication lines. Lately, there has been a substantial revival in the use of neural-network models, or connectionism, as the field is often called (see reference 1). Connectionist models are applicable to a variety of cognitive-science problems, including natural-language processing, speech processing, and vision.

One major advocate of connectionism is Daniel W. Hillis. His Connection Machine is more brain-like than a traditional computer. Hillis points out that in a conventional computer, most of the silicon lies inactive most of the time. At any given time, only the CPU and a very small part—a few bytes—of the memory are active (see reference 2). The Connection Machine is composed of many processor/memory units, most of which are active at the same time. Douglas R. Hofstadter (see references 3 and 4) has long been an advocate of a similar view of cognition, with interacting actors in a cognitive process exchanging messages.

On the simplest level, the brain functions as follows: Neurons activate or inhibit the firing of other neurons. Whether or not a particular neuron fires depends on the inhibitory or excitatory inputs from all the neurons connected to it. Somehow, the activations of all the neurons, how they communicate with one another, and the nervous system's interactions with the environment determine your memories and thoughts—at least as far as philosophical materialists are concerned.

Of course, neurophysiologists, while still largely in the dark as to the operation of higher cognitive functions, have learned a great deal more about the brain than is evident in this simple model. Nevertheless, scientists in the 1950s were amazed at what simple systems of nodes with excitatory and inhibitory connections could do. (See the text box "The Perceptron Controversy" on page 240.)

In 1943, Warren S. McCulloch and Walter Pitts proved that any neural-network model in which a finite amount of information could define the state of an individual neuron could be modeled on a standard computer. The "finite amount of information" assumption is a big one: The number of bits needed to describe the state of a given neuron may be so large that it makes the simulation slow and impractical.

Is the Brain Relevant?

A strong school in AI feels that studying the brain is not the most fruitful road to understanding thought. The brain represents just one way of making a thinking machine, and certainly not the optimal way. Traditional AI sees thought as a series of problems to solve, and believes strongly that there is no philosophical reason why a computer can't solve them. The basis for this belief is the Church-Turing thesis, which roughly states that if a function is computable, you can compute it with a conventional computer—formally, a Turing machine. This thesis cannot be proved, but it is widely accepted because no one can think of a counterexample.

Daniel Dennett of Tufts University argues that neurophysiologists, working from the "bottom up"—that is, from the minute details up to the overall problem to be solved—in an attempt to understand human cognition, and computer scientists, working from the "top down," may ultimately both reach their goal. But he thinks the computer scientists will reach it first (see reference 5). The argument is: If I take a computer running a spreadsheet, and try to figure out what the program is doing by looking at the electrical currents inside, I won't progress as quickly as I would if I tried to write another program that also runs a spreadsheet.

In an approach that is in between those of the neurophysiologists and the more traditional AI researchers, David E. Rumelhart, James L. McClelland, and their colleagues don't dismiss the brain as irrelevant to the functioning of the mind. Rather, they feel that, by experimenting with neural networks, they can gain insight into how the brain copes with the problems it has to solve.

Parallelism

Another reason for studying brain-like models is their parallelism. The "circuit-

continued

Matthew Zeidenberg (939 East Gorham St., Madison, WI 53703) is a graduate student in computer science at the University of Wisconsin.

Until we build effective parallel-processing hardware, connectionist models are unlikely to provide computationally efficient solutions to AI problems.

ry" of the brain is much slower than a computer's. In order for the brain to work as fast as it does—psychologists have shown that we can recognize objects in a split second—many neurons must work in parallel. In contrast, many AI programs run very slowly. The hope is: If we can find ways to run AI programs in parallel, they will run in a reasonable amount of time.

Parallel computation has been a busy area in computer science over the past 10 years. Most mainstream research on parallel computing is quite different from the neural-network approach. Researchers have studied algorithms for mesh-connected arrays of processors, pipelines, processors arranged in a tree-like fashion, and distributed systems of interconnected processors, to cite just a few.

Neural networks represent only one line of research in parallel computation. Basically, you must answer two fundamental questions in designing a parallel computer system: How do you connect the processors for communication purposes? And how much computing power and memory do you put in each processor? Many researchers find no reason to restrict themselves to neural-network models, which represent a very small subset of the possible parallel-computing models.

Nevertheless, neural-network researchers think that their models, by being most faithful to what we know about the brain, will show the most success. Unfortunately, neural networks have seldom been built in hardware; normally, they must be simulated in software. These simulations have typically been very slow, since one processor had to do the work of many. Until we build effective parallel-processing hardware, connectionist models are unlikely to provide computationally efficient solutions to AI problems.

The Connection Machine

One attempt to build a parallel computer is Hillis's Connection Machine, which has many small processors, each containing a small amount of memory. The machine has a fixed architecture—that is, certain processors are physically connected to certain others. Any pair of processors not physically connected can communicate in software via special processors called "routers," which exist to forward messages. It is critical that connections between processors be programmable so that the machine is not limited in the types of networks it can realize.

Hillis's machine can realize a wide variety of network models. It can adapt itself to the mesh-like architecture used in image processing, as well as to conventional semantic networks for knowledge representation. Recently, the Connection Machine has shown that it is well suited to database tasks, which are closely related to semantic networks. Hillis's company, Thinking Machines Corporation, has built a prototype of the machine, which it is marketing. The company provides a version of the popular AI programming language, LISP, that allows you access to the power of parallelism without having to know the details of the machine. On the Connection Machine, you can implement a neural network in hardware rather than in software, so the network will run much faster.

Varieties of Neural Networks

Most neural-network models owe something to perceptrons but are more general. The typical neural-network model consists of a set of nodes, or neurons, and connections (see figure 1). Each node contains a real number, which is its *activation*. Each connection also contains a real number, its *weight*. These numbers are usually positive and usually have a maximum value. Some of the units are connected to input and output. The weights represent the strength of the connection between two neurons.

Generally, a neural network is a dynamic system, moving from one state to the next. As such, it has a mathematical rule that takes the system from one state to the next. An infinite number of such rules are possible. However, we usually want to constrain our models to those that influence the activation of a given node based only on the activations of the nodes connected to it and the weights of the connections to those nodes.

Neural networks are not explicitly programmed like a conventional computer. Rather, they obey laws, or rules, like a physical system. You must program a conventional computer, but a neural network simply behaves. Neural-network designers view this as an advantage, since it provides a mechanism whereby intelligence can arise from physical law.

One of the simplest of these rules is a linear rule. You compute the activation of a given node as the sum of the products of the weight of each node it is connected to and the strength of that connection. Often such a rule is *thresholded*: Values that go above a certain threshold are cut off, to avoid arbitrarily large activation values. There are many variants of linear rules.

Another rule, suggested by D. O. Hebb (see reference 6) strengthens the connection between two nodes that are highly activated at the same time. Some versions of the Hebbian learning rule allow inputs, called teaching inputs, to in-

continued

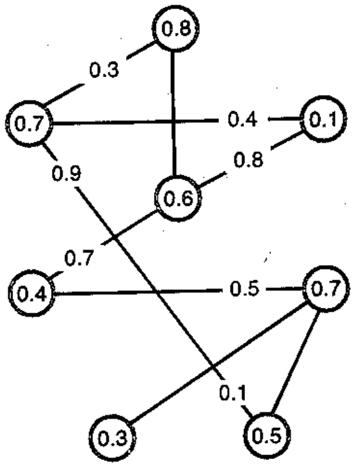


Figure 1: *In a typical neural network, every node has an activation, and every connection between nodes has a weight. A rule, which varies from network to network, governs the way in which weights and activations change over time.*

The Perceptron Controversy

The 1950s brought substantial interest in what neural networks could do. After all, if we understood their behavior, we would be able to understand the brain and the mind. That was the hope. One of the most popular variants of the neural network was the "perceptron," invented by Frank Rosenblatt. The perceptron was a "perception machine."

Perception has always been the most difficult area in AI, and the area in which the least progress has been made. The ability to decipher the world, to break it up into meaningful parts, is a human ability of astonishing complexity.

Ironically, areas that people view as difficult—like playing chess or solving a chemical structure—are areas in which AI programs have had the most success, although they seldom equal the abilities of the best human experts.

Yet, the computer has not been programmed that can learn a language the way any infant does, or given a scene of any room, can recognize all the objects in the room. Thus, machines that can perceive have always held special interest.

What Is a Perceptron?

A perceptron is a neural-network model with an array of input units, each of which can take on the value 0 or 1. This array is called the *retina*, in analogy to human vision.

The perceptron also has another array of units called the *predicates*. Each predicate can be connected to any subset of the units in the retina and can compute any linear function of the values of these units.

Finally, the predicate units are connected to one or more decision units, which return a single answer—yes or no—depending on the values of the units in the retina. Thus, a perceptron can perform an elementary classification task—that is, it can classify input patterns by some property. Since perception is basically a classification problem (i.e., classifying objects as chairs, tables, or whatever), the hope was that the perceptron model, properly elaborated, could account for complex perception.

The Great Debate

The controversy over perceptrons continued for some time, and in 1969, Marvin Minsky and Seymour Papert wrote a formal analysis of perceptrons (see reference 7), or, more precisely, the single-layer perceptron, which squelched interest in them.

Minsky and Papert proved, mathematically, that there were certain functions of input that the single-layer perceptron could not compute. One of the simplest was the parity function, which tells if the number of ones in the input is even or odd. If the perceptron could not compute such a simple function, they reasoned, it could hardly perform the complex tasks required for perception and intelligence.

At the time, Minsky and Papert's work appeared to have destroyed perceptrons and perceptron-like models as viable lines of AI research. Little attention was paid to the fact that they directed their criticism at a very simple system, the single-layer perceptron.

If you add one more layer of units between the input units and the predicates, the computational power of the machine rises abruptly, and Minsky and Papert's critique no longer applies. And if you add multiple layers, it is difficult to characterize formally the network's behavior.

This difference was not well understood at the time, and Minsky and Papert's work put a strong damper on research. It didn't discourage everyone, however: Throughout this time, Steven Grossberg of Boston University continued detailed studies of brain-like systems.

fluence the change in weight. This type of rule is a formalization of associationist psychology, which holds that associations are built up between things that occur together.

Competitive Learning

Learning is, perhaps, the most important phenomenon in psychology. Early neural-network researchers were anxious to show how networks could learn patterns in the input presented to them—that is, how they could come to perceive these patterns *on their own*.

One of the methods that various researchers have devised over the years is *competitive learning*. This method has a bottom level of input units that contains the pattern to be input to the system. The level above the input units consists of clusters of units. Each unit in a cluster competes with the other units in the cluster for the right to recognize an input pattern. Over a learning period, each unit in a cluster comes to recognize a subset of the patterns presented to it. Thus, each cluster represents a classification, or group, of input patterns.

In competitive learning, each unit in each cluster is connected to all the input units. The weights of the connections are initially set to random values. The random weights cause certain units in clusters to start responding more to particular input patterns, since the weights of the connections to particular input units are stronger to some than to others.

As the learning proceeds, the weights change. As particular units in the cluster become sensitive to particular units in the input pattern, the weights connecting the associated pairs of units increase, at the expense of unassociated pairs of units. Different units in the same cluster inhibit each other, so that only one unit in a cluster "wins" the right to recognize a given pattern.

Thus, over time, different units in a cluster come to "recognize" different properties of input patterns. For instance, a cluster of two units might separate all the input patterns into those that are mostly on (i.e., have most of their units highly activated) and those that are mostly off. Larger clusters would make more discriminating classifications.

There may be an additional level of clusters that uses the first level of clusters as its input pattern. This level could extract more complex features from the bottom-level input pattern.

Rumelhart and David Zipser applied the competitive-learning paradigm to letter and word recognition. Letters were represented by bits on a grid, which was the input pattern for the competitive-learning system. The system came to spontaneously recognize an "A" and a "B" in a fixed position on the grid.

This is very interesting, for it illustrates a potential mechanism by which people may have learned to recognize letters. This mechanism is completely general, since it presupposes nothing about the letters except that they can be distinguished from one another.

Boltzmann Machines

An important class of neural networks simulates the behavior of physical systems. Physical systems have a tendency to move into states of minimum potential energy. A simple example of this is a ball rolling into the valley between two hills. At the top of the hill, potential energy is high; in the valley, it is low.

continued

In a distributed network, nodes don't have a simple meaning; rather, an individual concept is represented by a pattern over all the nodes.

This process is called *relaxation*. John Hopfield has shown that a certain simple evolutionary rule for a neural network will lead to relaxation. Systems such as Hopfield's, which resemble thermodynamic systems like the atoms in a room, are called Boltzmann Machines, after Ludwig Boltzmann, a physicist who made major contributions to thermodynamics. Boltzmann Machines are widely used in a variety of neural-network applications.

In vision and in playing games, you can often formulate solutions to problems, such as recognizing a set of objects or discovering the best move, as constraint-satisfaction problems. For instance, in chess, the constraints are the possible ways that a piece can move, and the total "goodness" of the move, as measured by some formula, taking into account pieces captured, board position, and so on. Relaxation can correspond closely to constraint-satisfaction—a Boltzmann Machine can satisfy constraints automatically.

Distributed Representations

One important feature of many neural-network models is their *distributed* nature. A standard semantic network, like those used in early knowledge-representation schemes, consists of a set of nodes connected in some fashion. Each node represents a single word or concept. If the network is "thinking" of the word "cat," the node for "cat" is activated, and all other nodes are not. This is a *local* representation.

In contrast, in a distributed network, nodes don't have a simple meaning; rather, an individual concept is represented by a pattern over all the nodes. For instance, if there are 10 nodes, activating nodes 1, 3, 4, and 7 might represent the concept "gorilla," while activating nodes 2, 4, 5, and 7 might represent the closely related concept "chimp." Concepts that are closely related have similar representations.

A parallel distributed-processing (PDP) network, a neural network that uses distributed representation, offers the advantage of *automatic generalization*. If I want to represent the concept "gorillas are hairy," I strengthen the connection between all the nodes composing the concept "gorilla" and all the nodes composing the concept "hairy." As a result, since most of the nodes in "gorilla" are also used in "chimp," an association is also made between "chimp" and "hairy." This is how automatic generalization works. In a local representation, where "gorilla" and "chimp" are represented by separate nodes, a connection between "gorilla" and "hairy" would not imply a connection between "chimp" and "hairy."

Another advantage of a distributed representation is its insensitivity to damage. In a local representation, if the system loses the node representing "grandmother," it loses its concept of grandmother. People don't display disorders like this; there are no people who are completely normal except that they have lost their concept of grandmother. This has led to the opinion that the brain doesn't use local representation.

In a distributed representation, in order to lose a concept, you must lose *all* the nodes representing it. If you lose only one or two of the nodes, the concept may be degraded, but it's still there. This is closer to the type of memory loss seen in older adults: Memory is degraded in a uniform fashion.

Schemata

One criticism of neural-network models is that they're not as flexible at representing knowledge as standard methods are. The standard methods include the local semantic network, of which Marvin Minsky's *frame* and Roger Schank's *script* are varieties. For instance, a frame description of a bedroom would contain information about all the objects in that room and how they relate to one another. The relations between objects are represented by labeled links.

Cognitive psychologists, notably developmental psychologists like Jean Piaget, use the concept of a *schema*. A schema is a mirror—in the mind—of a real situation. As children, and as adults, we learn new associations and relations between objects and integrate them into our schemata.

It's not immediately clear how a neural-network model can account for knowledge represented in a schema; however, Rumelhart, Paul Smolensky, McClelland, and Geoffrey Hinton have shown that it's possible. They first gathered data from subjects about rooms—kitchens, bedrooms, offices, living rooms, and bathrooms. They took 40 words associated with rooms and asked each subject whether each word was associated with each room. Then, they set up a network that had each of the 40 words represented by a single node. They set the weight of a connection between two nodes to correspond to the extent to which the two tended to be used together when a single subject described a single room.

The network uses Hopfield's energy-minimization rule. When a single descriptor is "clamped on" (i.e., when its activation is permanently set to its maximum value), the system relaxes into one of five states, or rooms, since each room implies a constraint as to which words can occur together.

In the network, you don't explicitly define the schemata; you only set the associations between pairs of descriptors. The schema emerges out of the network as a natural consequence of its behavior. Thus, the schemata are not explicitly represented in the network, but rather are simply patterns of activation across a set of descriptors.

This system has several nice properties. First, it explains how schema are activated when you have incomplete information—that is, why you think "kitchen" when you see "refrigerator." This corresponds to the "clamping on" of a single descriptor.

Since schemata are patterns rather than single units, this system allows for more flexibility in representing things. A slightly different version of a particular object can correspond to a slight change in the weights. And closely related schemata, such as "woman" and "girl," can overlap. In a more elaborate scheme, each descriptor in a schema can itself be a schema. The number of connections you need, however, rises quickly.

Cognitive Hierarchies

Often neural-network models are ordered into hierarchies. Several levels exist in such a hierarchy, each composed of a set of units. Typically, units that receive input are at the bottom of the system, and units that give output are at the top. In a bottom-up system, units at each level connect to other units on their own level and influence units on levels above them. In a top-down system, units again connect to units on their own level but influence units on levels below.

Top down and bottom up are familiar concepts in cognitive science. For instance, in sentence perception, these terms refer to how different-size linguistic elements, the phoneme (sound), morpheme (word element), word, phrase, and sentence, interact with one other.

continued

Thanks to you it works for all of us.
United Way
notice to workers about unemployment insurance

This ad is for all those who ever wonder why your company runs a United Way campaign.

When it comes right down to it, you're probably the best reason your company has for getting involved with the United Way.

You see, they know almost all of the money given to the United Way goes back out into the community to help people.

So if you, or the people you work with, should ever need any of our services, like day care, family counseling or health care, we'll be right there to help. In fact, there are tens of thousands of United Way-supported programs and services in cities and towns across the country. That means help is nearby wherever you are.

And your company knows that could mean the difference between keeping or losing a valuable employee.

That's why they give. And that's why they ask you to give. Because there may come a day when you need help yourself.

United Way
Thanks to you, it works, for ALL OF US.

Ad Council A Public Service of This Magazine & The Advertising Council

The HEARSAY-II speech-recognition program from Carnegie Mellon was one of the first AI programs to integrate knowledge from several levels, storing its results in a global data structure called the blackboard.

Does the overall perception of a word help you to perceive all the letters in it, individually, in a top-down fashion? Most psychologists would say yes. For instance, psychologists have done experiments in which they show subjects nonwords like BCAK and PLAM; the subjects interpret these words as BACK and PALM. The theoretical explanation is that the units representing letters activate the units representing words in a manner that is somewhat insensitive to the letter's position in the word. The unit for the whole word actually influences the perception of the individual sound. Neural-network models exist that model this process and others like it.

For example, in McClelland's programmable blackboard model of reading, units for letters and units for words are connected by a grid. A connection in the grid between a letter and a word is set to a positive value if the letter is in the word, and to zero if it isn't. The letter units reinforce the word units in a bottom-up fashion, and the word units influence the activation of the letter units in a top-down fashion. Thus, the network converges to the perception of a single word at a time.

The programmable blackboard model does not handle the perception of individual letters, but you could readily add a third level to the system, a level of letter subfeatures. Information would pass up and down in the network, from letter subfeature to letter to word, and back down again.

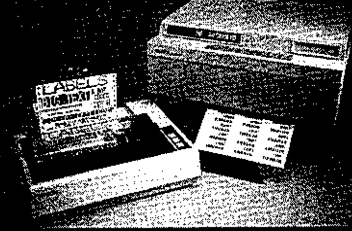
A Parallel Reading Network

One problem in creating a reading network is that people tend to read more than one word at a time. Since a single network reads only one word, it can't handle this. If the network tries to read more than one word, you get "crosstalk"; that is, if the input words are "bank" and "lane," the network will perceive both the two inputs and "lank" and "bane" as well. As a solution, McClelland proposes duplicate copies of networks. Duplicate individual word-recognition networks would have programmable connections instead of hard-wired connections between letters and words.

In addition to programmable networks, you could have a hard-wired network that represents the relationships between letters and words. This network programs all the programmable networks via connections to them. Thus, you could represent knowledge centrally instead of

continued

LABELING SOFTWARE



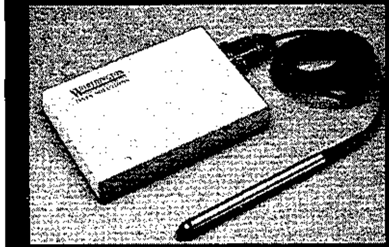
For DOT MATRIX and Laser Printers (Epson/IBM/Okidata/LaserJet)

- Labels for shelves, bins, inventory
- Text readable up to 50 ft.
- Bar Codes: I 2 of 5, UPC/ EAN MIL-STD, AIAG, Code 39
- Any size label • Flexible format
- Color options • Reversals • Fast
- File input • Menu driven • $279
- Other bar code programs from $49.
- Not Copy Protected!

30 Day Money Back Guarantee!

WORTHINGTON
DATA SOLUTIONS
417-A Ingalls Street
Santa Cruz, California 95060
408/458-9938

Bar Code Readers for PC, XT, AT, PS/2



RS-232 and PS/2 model........ $399
PC/XT/AT Internal/External ... $385

- Rugged Metal Pen
- Reads I 2 of 5, UPC/EAN Codabar, Code 39, etc.
- Attaches as 2nd Keyboard
- No software needed to add bar code reading to your system

30 Day Money Back Guarantee!

WORTHINGTON
DATA SOLUTIONS
417-A Ingalls Street,
Santa Cruz, California 95060
408/458-9938

having to duplicate it several times. You can save memory space by loading a programmable network with only knowledge relevant to processing the word it currently encounters. McClelland has worked out the details of his model thoroughly.

The programmable blackboard model accounts for psychological data concerning such things as parallelism in reading and word misperception. It shows how useful the psychological modeling approach to AI is: In explaining a good deal of psychological data with a model, we get a system that is quite good at the job at hand. McClelland has constructed a similar model of speech recognition and built a model of a higher-level process.

Processing Sentences

One important aspect of sentence understanding involves determining the various roles that the different parts of a sentence play. For instance, consider the following two sentences:

The house rented for $2000.
The man rented the car.

In the first sentence, the house is the thing rented; in the second, the man is the agent of the rental. Yet in the two sentences, the nouns "man" and "house" are in the same position. Somehow, the model must discern their different roles.

McClelland and Alan Kawamoto have developed a connectionist system to do this role assignment. Words are described by "semantic microfeatures"—basic dimensions that describe many objects and actions. For instance, two of the microfeatures describing nouns are *human* and *softness*, which have the values "human, nonhuman" and "soft, hard," respectively. Words are not directly represented in the system's networks, but in terms of the activations of units representing microfeatures.

The model has a group of units for each of the major roles that different nouns can play in an action. These roles are Agent (actor), Patient (acted upon), Instrument (thing used), and Modifier (adverbial word or clause). For instance, the sentence "The man ate the sandwich" would activate the microfeatures of "ate" and "man" in the set of units that corresponds to the Agent; this represents the fact that the Agent for the verb "ate" is "man."

The system is trained on a series of sentences. The correct role assignments for the training sentences are shown to the system. These assignments correspond to the activations of particular nodes. The system adjusts the connections between these nodes so that they reinforce one another.

After being trained on a sufficient number of sentences, the system can make correct role assignments for new sentences. It can even make accurate role assignments for sentences with some syntactic ambiguity. For instance, in the sentence "The man hit the boy with the mallet," the system figures out that "mallet" is the Instrument of "hit" instead of belonging to "boy," since "mallet" has microfeatures that fit in well with it being an Instrument.

The system also handles a number of other problems well, and generally does a good job in assigning roles. McClelland and Kawamoto are currently considering ways of expanding their system into a more complete language-understanding model—for instance, one that includes a network to parse sentences.

The Promise for the Future

Neural networks are good for a variety of natural-language processing tasks, including letter recognition, reading, and sentence understanding. They are also useful in storing knowledge in schemata and in retrieving items from memory. They are not a cure-all for what ails AI and cognitive psychology, but they do bring a strong and biologically plausible new direction to many important problems.

Eventually, a connectionist model will probably be built of the natural-language-understanding process, since, as psychologists have shown, it involves integrating knowledge from many domains, including phonetics, morphology, syntax, and semantics. Connectionist models are particularly good at integrating these types of knowledge. ■

ACKNOWLEDGMENT
I would like to thank Gregg Oden of the psychology department at the University of Wisconsin for introducing me to the subject of connectionism.

REFERENCES
1. Rumelhart, David E., James L. McClelland, et al. *Parallel Distributed Processing: Explorations in the Microstructures of Cognition*, vols. 1 and 2. Cambridge, MA: MIT Press, 1986.
2. Hillis, W. Daniel. *The Connection Machine*. Cambridge, MA: MIT Press, 1986.
3. Hofstadter, Douglas R. *Godel, Escher, Bach: An Eternal Golden Braid*. New York, NY: Basic Books Inc., 1979.
4. Hofstadter, Douglas R. *Metamagical Themas*. New York: Basic Books Inc., 1985.
5. Dennett, Daniel. *Brainstorms*. Cambridge, MA: MIT Press, 1981.
6. Hebb, D. O. *The Organization of Behavior*. New York: John Wiley and Sons, 1949.
7. Minsky, Marvin, and Seymour Papert. *Perceptrons*. Cambridge, MA: MIT Press, 1969.