CHAPTER 1

The Appeal of Parallel Distributed Processing

J. L. McCLELLAND, D. E. RUMELHART, and G. E. HINTON

What makes people smarter than machines? They certainly are not quicker or more precise. Yet people are far better at perceiving objects in natural scenes and noting their relations, at understanding language and retrieving contextually appropriate information from memory, at making plans and carrying out contextually appropriate actions, and at a wide range of other natural cognitive tasks. People are also far better at learning to do these things more accurately and fluently through processing experience.

What is the basis for these differences? One answer, perhaps the classic one we might expect from artificial intelligence, is "software." If we only had the right computer program, the argument goes, we might be able to capture the fluidity and adaptability of human information processing.

Certainly this answer is partially correct. There have been great breakthroughs in our understanding of cognition as a result of the development of expressive high-level computer languages and powerful algorithms. No doubt there will be more such breakthroughs in the future. However, we do not think that software is the whole story.

In our view, people are smarter than today's computers because the brain employs a basic computational architecture that is more suited to deal with a central aspect of the natural information processing tasks that people are so good at. In this chapter, we will show through examples that these tasks generally require the simultaneous consideration of many pieces of information or constraints. Each constraint may be imperfectly specified and ambiguous, yet each can play a potentially

decisive role in determining the outcome of processing. After examining these points, we will introduce a computational framework for modeling cognitive processes that seems well suited to exploiting these constaints and that seems closer than other frameworks to the style of computation as it might be done by the brain. We will review several early examples of models developed in this framework, and we will show that the mechanisms these models employ can give rise to powerful emergent properties that begin to suggest attractive alternatives to traditional accounts of various aspects of cognition. We will also show that models of this class provide a basis for understanding how learning can occur spontaneously, as a by-product of processing activity.

Multiple Simultaneous Constraints

Reaching and grasping. Hundreds of times each day we reach for things. We nearly never think about these acts of reaching. And yet, each time, a large number of different considerations appear to jointly determine exactly how we will reach for the object. The position of the object, our posture at the time, what else we may also be holding, the size, shape, and anticipated weight of the object, any obstacles that may be in the way—all of these factors jointly determine the exact method we will use for reaching and grasping.

Consider the situation shown in Figure 1. Figure 1A shows Jay McClelland's hand, in typing position at his terminal. Figure 1B indicates the position his hand assumed in reaching for a small knob on the desk beside the terminal. We will let him describe what happened in the first person:

> On the desk next to my terminal are several objects—a chipped coffee mug, the end of a computer cable, a knob from a clock radio. I decide to pick the knob up. At first I hesitate, because it doesn't seem possible. Then I just reach for it, and find myself grasping the knob in what would normally be considered a very awkward position—but it solves all of the constraints. I'm not sure what all the details of the movement were, so I let myself try it a few times more. I observe that my right hand is carried up off the keyboard, bent at the elbow, until my forearm is at about a 30° angle to the desk top and parallel to the side of the terminal. The palm is facing downward through most of this. Then, my arm extends and lowers down more or less parallel to the edge of the desk and parallel to the side of the terminal and, as it drops, it turns about 90° so that the

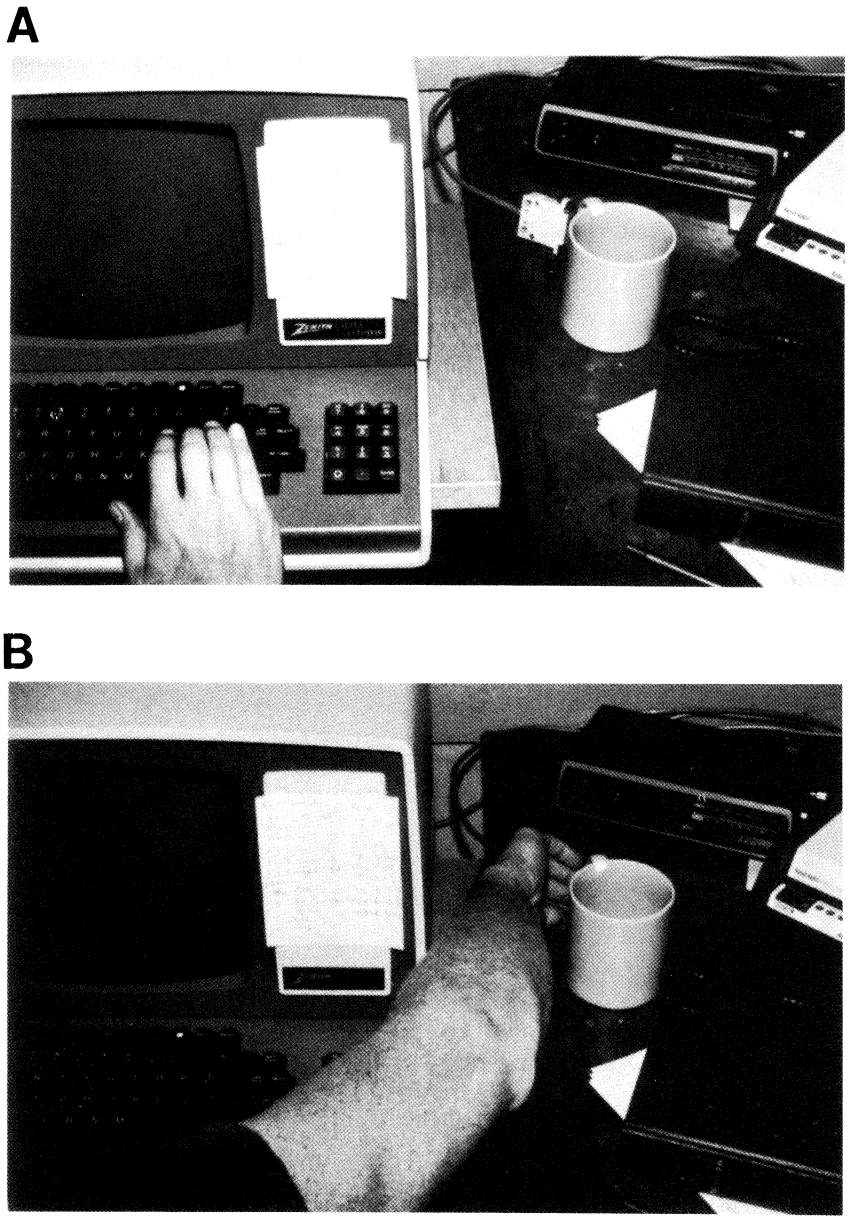

FIGURE 1. *A:* An everyday situation in which it is necessary to take into account a large number of constraints to grasp a desired object. In this case the target object is the small knob to the left of the cup. *B:* The posture the arm arrives at in meeting these constraints.

palm is facing the cup and the thumb and index finger are below. The turning motion occurs just in time, as my hand drops, to avoid hitting the coffee cup. My index finger and thumb close in on the knob and grasp it, with my hand completely upside down.

Though the details of what happened here might be quibbled with, the broad outlines are apparent. The shape of the knob and its position on the table; the starting position of the hand on the keyboard; the positions of the terminal, the cup, and the knob; and the constraints imposed by the structure of the arm and the musculature used to control it—all these things conspired to lead to a solution which exactly suits the problem. If any of these constraints had not been included, the movement would have failed. The hand would have hit the cup or the terminal—or it would have missed the knob.

The mutual influence of syntax and semantics. Multiple constraints operate just as strongly in language processing as they do in reaching and grasping. Rumelhart (1977) has documented many of these multiple constraints. Rather than catalog them here, we will use a few examples from language to illustrate the fact that the constraints tend to be reciprocal: The example shows that they do not run only from syntax to semantics—they also run the other way.

It is clear, of course, that syntax constrains the assignment of meaning. Without the syntactic rules of English to guide us, we cannot correctly understand who has done what to whom in the following sentence:

The boy the man chased kissed the girl.

But consider these examples (Rumelhart, 1977; Schank, 1973):

I saw the grand canyon flying to New York.
I saw the sheep grazing in the field.

Our knowledge of syntactic rules alone does not tell us what grammatical role is played by the prepositional phrases in these two cases. In the first, "flying to New York" is taken as describing the context in which the speaker saw the Grand Canyon—while he was flying to New York. In the second, "grazing in the field" could syntactically describe an analogous situation, in which the speaker is grazing in the field, but this possibility does not typically become available on first reading. Instead we assign "grazing in the field" as a modifier of the sheep (roughly, "who were grazing in the field"). The syntactic structure of each of

these sentences, then, is determined in part by the semantic relations that the constituents of the sentence might plausibly bear to one another. Thus, the influences appear to run both ways, from the syntax to the semantics and from the semantics to the syntax.

In these examples, we see how syntactic considerations influence semantic ones and how semantic ones influence syntactic ones. We cannot say that one kind of constraint is primary.

Mutual constraints operate, not only between syntactic and semantic processing, but also within each of these domains as well. Here we consider an example from syntactic processing, namely, the assignment of words to syntactic categories. Consider the sentences:

I like the joke.
I like the drive.
I like to joke.
I like to drive.

In this case it looks as though the words *the* and *to* serve to determine whether the following word will be read as a noun or a verb. This, of course, is a very strong constraint in English and can serve to force a verb interpretation of a word that is not ordinarily used this way:

I like to mud.

On the other hand, if the information specifying whether the function word preceding the final word is *to* or *the* is ambiguous, then the typical reading of the word that follows it will determine which way the function word is heard. This was shown in an experiment by Isenberg, Walker, Ryder, and Schweikert (1980). They presented sounds halfway between *to* (actually /tˆ/) and *the* (actually /dˆ/) and found that words like *joke*, which we tend to think of first as nouns, made subjects hear the marginal stimuli as *the,* while words like *drive*, which we tend to think of first as verbs, made subjects hear the marginal stimuli as *to.* Generally, then, it would appear that each word can help constrain the syntactic role, and even the identity, of every other word.

Simultaneous mutual constraints in word recognition. Just as the syntactic role of one word can influence the role assigned to another in analyzing sentences, so the identity of one letter can influence the identity assigned to another in reading. A famous example of this, from Selfridge, is shown in Figure 2. Along with this is a second example in which none of the letters, considered separately, can be identified unambiguously, but in which the possibilities that the visual

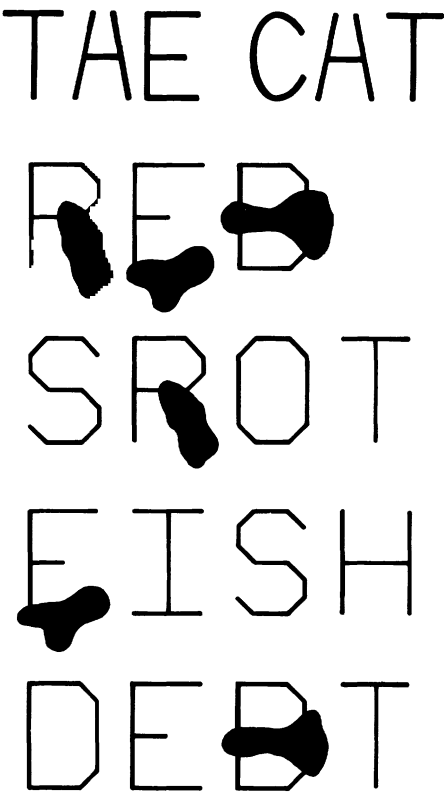

FIGURE 2. Some ambiguous displays. The first one is from Selfridge, 1955. The second line shows that three ambiguous characters can each constrain the identity of the others. The third, fourth, and fifth lines show that these characters are indeed ambiguous in that they assume other identities in other contexts. (The ink-blot technique of making letters ambiguous is due to Lindsay and Norman, 1972).

information leaves open for each so constrain the possible identities of the others that we are capable of identifying all of them.

At first glance, the situation here must seem paradoxical: The identity of each letter is constrained by the identities of each of the others. But since in general we cannot know the identities of any of the letters

until we have established the identities of the others, how can we get the process started?

The resolution of the paradox, of course, is simple. One of the different possible letters in each position fits together with the others. It appears then that our perceptual system is capable of exploring all these possibilities without committing itself to one until all of the constraints are taken into account.

Understanding through the interplay of multiple sources of knowledge. It is clear that we know a good deal about a large number of different standard situations. Several theorists have suggested that we store this knowledge in terms of structures called variously: *scripts* (Schank, 1976), *frames* (Minsky, 1975), or *schemata* (Norman & Bobrow, 1976; Rumelhart, 1975). Such knowledge structures are assumed to be the basis of comprehension. A great deal of progress has been made within the context of this view.

However, it is important to bear in mind that most everyday situations cannot be rigidly assigned to just a single script. They generally involve an interplay between a number of different sources of information. Consider, for example, a child's birthday party at a restaurant. We know things about birthday parties, and we know things about restaurants, but we would not want to assume that we have explicit knowledge (at least, not in advance of our first restaurant birthday party) about the conjunction of the two. Yet we can imagine what such a party might be like. The fact that the party was being held in a restaurant would modify certain aspects of our expectations for birthday parties (we would not expect a game of Pin-the-Tail-on-the-Donkey, for example), while the fact that the event was a birthday party would inform our expectations for what would be ordered and who would pay the bill.

Representations like scripts, frames, and schemata are useful structures for encoding knowledge, although we believe they only approximate the underlying structure of knowledge representation that emerges from the class of models we consider in this book, as explained in Chapter 14. Our main point here is that any theory that tries to account for human knowledge using script-like knowledge structures will have to allow them to interact with each other to capture the generative capacity of human understanding in novel situations. Achieving such interactions has been one of the greatest difficulties associated with implementing models that really think generatively using script- or frame-like representations.

PARALLEL DISTRIBUTED PROCESSING

In the examples we have considered, a number of different pieces of information must be kept in mind at once. Each plays a part, constraining others and being constrained by them. What kinds of mechanisms seem well suited to these task demands? Intuitively, these tasks seem to require mechanisms in which each aspect of the information in the situation can act on other aspects, simultaneously influencing other aspects and being influenced by them. To articulate these intuitions, we and others have turned to a class of models we call *Parallel Distributed Processing* (PDP) models. These models assume that information processing takes place through the interactions of a large number of simple processing elements called units, each sending excitatory and inhibitory signals to other units. In some cases, the units stand for possible hypotheses about such things as the letters in a particular display or the syntactic roles of the words in a particular sentence. In these cases, the activations stand roughly for the strengths associated with the different possible hypotheses, and the interconnections among the units stand for the constraints the system knows to exist between the hypotheses. In other cases, the units stand for possible goals and actions, such as the goal of typing a particular letter, or the action of moving the left index finger, and the connections relate goals to subgoals, subgoals to actions, and actions to muscle movements. In still other cases, units stand not for particular hypotheses or goals, but for aspects of these things. Thus a hypothesis about the identity of a word, for example, is itself distributed in the activations of a large number of units.

PDP Models: Cognitive Science or Neuroscience?

One reason for the appeal of PDP models is their obvious "physiological" flavor: They seem so much more closely tied to the physiology of the brain than are other kinds of information-processing models. The brain consists of a large number of highly interconnected elements (Figure 3) which apparently send very simple excitatory and inhibitory messages to each other and update their excitations on the basis of these simple messages. The properties of the units in many of the PDP models we will be exploring were inspired by basic properties of the neural hardware. In a later section of this book, we will examine in some detail the relation between PDP models and the brain.

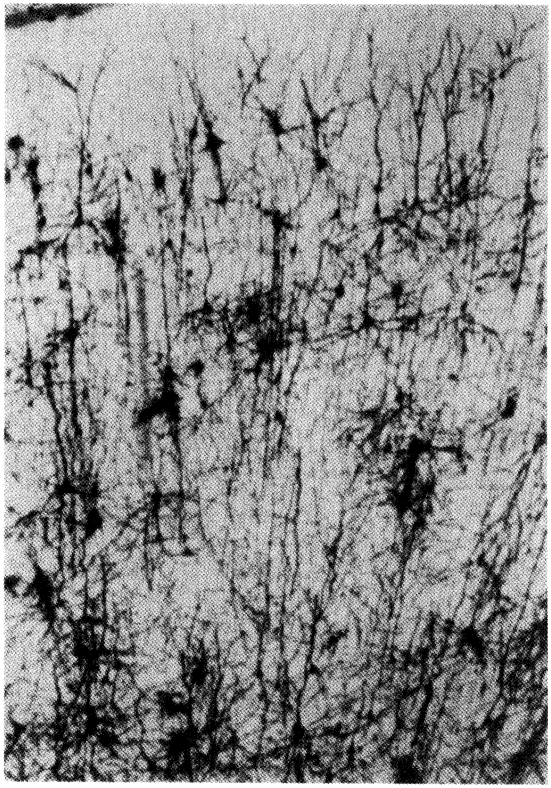

FIGURE 3. The arborizations of about 1 percent of the neurons near a vertica' slice through the cerebral cortex. The full height of the figure corresponds to the thickness of the cortex, which is in this instance about 2 mm. (From *Mechanics of the Mind*, p. 84, by C. Blakemore, 1977, Cambridge, England: Cambridge University Press. Copyright 1977 by Cambridge University Press. Reprinted by permission.)

Though the appeal of PDP models is definitely enhanced by their physiological plausibility and neural inspiration, these are not the primary bases for their appeal to us. We are, after all, cognitive scientists, and PDP models appeal to us for psychological and computational reasons. They hold out the hope of offering computationally sufficient and psychologically accurate mechanistic accounts of the phenomena of human cognition which have eluded successful explication in conventional computational formalisms; and they have radically altered the way we think about the time-course of processing, the nature of representation, and the mechanisms of learning.

The Microstructure of Cognition

The process of human cognition, examined on a time scale of seconds and minutes, has a distinctly sequential character to it. Ideas come, seem promising, and then are rejected; leads in the solution to a problem are taken up, then abandoned and replaced with new ideas. Though the process may not be discrete, it has a decidedly sequential character, with transitions from state-to-state occurring, say, two or three times a second. Clearly, any useful description of the overall organization of this sequential flow of thought will necessarily describe a sequence of states.

But what is the internal structure of each of the states in the sequence, and how do they come about? Serious attempts to model even the simplest macrosteps of cognition—say, recognition of single words—require vast numbers of microsteps if they are implemented sequentially. As Feldman and Ballard (1982) have pointed out, the biological hardware is just too sluggish for sequential models of the microstructure to provide a plausible account, at least of the microstructure of *human* thought. And the time limitation only gets worse, not better, when sequential mechanisms try to take large numbers of constraints into account. Each additional constraint requires more time in a sequential machine, and, if the constraints are imprecise, the constraints can lead to a computational explosion. Yet people get faster, not slower, when they are able to exploit additional constraints.

Parallel distributed processing models offer alternatives to serial models of the microstructure of cognition. They do not deny that there is a macrostructure, just as the study of subatomic particles does not deny the existence of interactions between atoms. What PDP models do is describe the internal structure of the larger units, just as subatomic physics describes the internal structure of the atoms that form the constituents of larger units of chemical structure.

We shall show as we proceed through this book that the analysis of the microstructure of cognition has important implications for most of the central issues in cognitive science. In general, from the PDP point of view, the objects referred to in macrostructural models of cognitive processing are seen as approximate descriptions of emergent properties of the microstructure. Sometimes these approximate descriptions may be sufficiently accurate to capture a process or mechanism well enough; but many times, we will argue, they fail to provide sufficiently elegant or tractable accounts that capture the very flexibility and open-endedness of cognition that their inventors had originally intended to capture. We hope that our analysis of PDP models will show how an

examination of the microstructure of cognition can lead us closer to an adequate description of the real extent of human processing and learning capacities.

The development of PDP models is still in its infancy. Thus far the models which have been proposed capture simplified versions of the kinds of phenomena we have been describing rather than the full elaboration that these phenomena display in real settings. But we think there have been enough steps forward in recent years to warrant a concerted effort at describing where the approach has gotten and where it is going now, and to point out some directions for the future.

The first section of the book represents an introductory course in parallel distributed processing. The rest of this chapter attempts to describe in informal terms a number of the models which have been proposed in previous work and to show that the approach is indeed a fruitful one. It also contains a brief description of the major sources of the inspiration we have obtained from the work of other researchers. This chapter is followed, in Chapter 2, by a description of the quantitative framework within which these models can be described and examined. Chapter 3 explicates one of the central concepts of the book: *distributed representation*. The final chapter in this section, Chapter 4, returns to the question of demonstrating the appeal of parallel distributed processing models and gives an overview of our explorations in the microstructure of cognition as they are laid out in the remainder of this book.

EXAMPLES OF PDP MODELS

In what follows, we review a number of recent applications of PDP models to problems in motor control, perception, memory, and language. In many cases, as we shall see, parallel distributed processing mechanisms are used to provide natural accounts of the exploitation of multiple, simultaneous, and often mutual constraints. We will also see that these same mechanisms exhibit emergent properties which lead to novel interpretations of phenomena which have traditionally been interpreted in other ways.

Motor Control

Having started with an example of how multiple constraints appear to operate in motor programming, it seems appropriate to mention two

models in this domain. These models have not developed far enough to capture the full details of obstacle avoidance and multiple constraints on reaching and grasping, but there have been applications to two problems with some of these characteristics.

Finger movements in skilled typing. One might imagine, at first glance, that typists carry out keystrokes successively, first programming one stroke and then, when it is completed, programming the next. However, this is not the case. For skilled typists, the fingers are continually anticipating upcoming keystrokes. Consider the word *vacuum.* In this word, the *v, a,* and *c* are all typed with the left hand, leaving the right hand nothing to do until it is time to type the first *u.* However, a high speed film of a good typist shows that the right hand moves up to anticipate the typing of the *u*, even as the left hand is just beginning to type the *v.* By the time the *c* is typed the right index finger is in position over the *u* and ready to strike it.

When two successive key strokes are to be typed with the fingers of the same hand, concurrent preparation to type both can result in similar or conflicting instructions to the fingers and/or the hand. Consider, in this light, the difference between the sequence *ev* and the sequence *er.* The first sequence requires the typist to move up from home row to type the *e* and to move down from the home row to type the *v,* while in the second sequence, both the *e* and the *r* are above the home row.

The hands take very different positions in these two cases. In the first case, the hand as a whole stays fairly stationary over the home row. The middle finger moves up to type the *e*, and the index finger moves down to type the *v.* In the second case, the hand as a whole moves up, bringing the middle finger over the *e* and the index finger over the *r.* Thus, we can see that several letters can simultaneously influence the positioning of the fingers and the hands.

From the point of view of optimizing the efficiency of the typing motion, these different patterns seem very sensible. In the first case, the hand as a whole is maintained in a good compromise position to allow the typist to strike both letters reasonably efficiently by extending the fingers up or down. In the second case, the need to extend the fingers is reduced by moving the whole hand up, putting it in a near-optimal position to strike either key.

Rumelhart and Norman (1982) have simulated these effects using PDP mechanisms. Figure 4 illustrates aspects of the model as they are illustrated in typing the word *very.* In brief, Rumelhart and Norman assumed that the decision to type a word caused activation of a unit for that word. That unit, in turn, activated units corresponding to each of the letters in the word. The unit for the first letter to be typed was made to inhibit the units for the second and following letters, the unit

for the second to inhibit the third and following letters, and so on. As a result of the interplay of activation and inhibition among these units, the unit for the first letter was at first the most strongly active, and the units for the other letters were partially activated.

Each letter unit exerts influences on the hand and finger involved in typing the letter. The *v* unit, for example, tends to cause the index finger to move down and to cause the whole hand to move down with it. The *e* unit, on the other hand, tends to cause the middle finger on the left hand to move up and to cause the whole hand to move up also. The *r* unit also causes the left index finger to move up and the left hand to move up with it.

The extent of the influences of each letter on the hand and finger it directs depends on the extent of the activation of the letter. Therefore, at first, in typing the word *very*, the *v* exerts the greatest control.

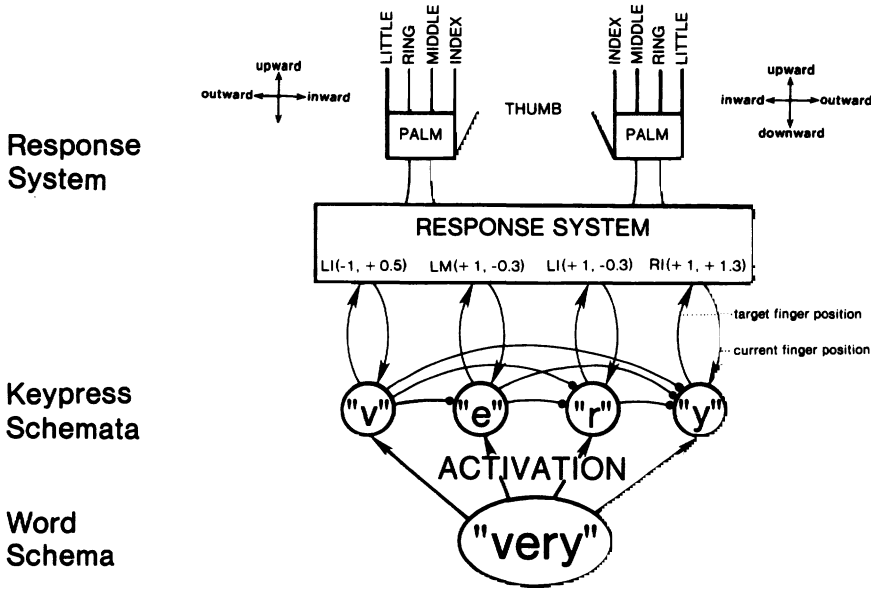

FIGURE 4. The interaction of activations in typing the word *very*. The *very* unit is activated from outside the model. It in turn activates the units for each of the component letters. Each letter unit specifies the target finger positions, specified in a keyboard coordinate system. L and R stand for the left and right hands, and I and M for the index and middle fingers. The letter units receive information about the current finger position from the response system. Each letter unit inhibits the activation of all letter units that follow it in the word: inhibitory connections are indicated by the lines with solid dots at their terminations. (From "Simulating a Skilled Typist: A Study of Skilled Motor Performance" by D. E. Rumelhart and D. A. Norman, 1982, *Cognitive Science, 6,* p. 12. Copyright 1982 by Ablex Publishing. Reprinted by permission.)

Because the *e* and *r* are simultaneously pulling the hand up, though, the *v* is typed primarily by moving the index finger, and there is little movement on the whole hand.

Once a finger is within a certain striking distance of the key to be typed, the actual pressing movement is triggered, and the keypress occurs. The keypress itself causes a strong inhibitory signal to be sent to the unit for the letter just typed, thereby removing this unit from the picture and allowing the unit for the next letter in the word to become the most strongly activated.

This mechanism provides a simple way for all of the letters to jointly determine the successive configurations the hand will enter into in the process of typing a word. This model has shown considerable success predicting the time between successive keystrokes as a function of the different keys involved. Given a little noise in the activation process, it can also account for some of the different kinds of errors that have been observed in transcription typing.

The typing model represents an illustration of the fact that serial behavior—a succession of key strokes—is not necessarily the result of an inherently serial processing mechanism. In this model, the sequential structure of typing emerges from the interaction of the excitatory and inhibitory influences among the processing units.

Reaching for an object without falling over. Similar mechanisms can be used to model the process of reaching for an object without losing one's balance while standing, as Hinton (1984) has shown. He considered a simple version of this task using a two-dimensional "person" with a foot, a lower leg, an upper leg, a trunk, an upper arm, and a lower arm. Each of these limbs is joined to the next at a joint which has a single degree of rotational freedom. The task posed to this person is to reach a target placed somewhere in front of it, without taking any steps and without falling down. This is a simplified version of the situation in which a real person has to reach out in front for an object placed somewhere in the plane that vertically bisects the body. The task is not as simple as it looks, since if we just swing an arm out in front of ourselves, it may shift our center of gravity so far forward that we will lose our balance. The problem, then, is to find a set of joint angles that simultaneously solves the two constraints on the task. First, the tip of the forearm must touch the object. Second, to keep from falling down, the person must keep its center of gravity over the foot.

To do this, Hinton assigned a single processor to each joint. On each computational cycle, each processor received information about how far the tip of the hand was from the target and where the center of gravity was with respect to the foot. Using these two pieces of information, each joint adjusted its angle so as to approach the goals of maintaining

balance and bringing the tip closer to the target. After a number of iterations, the stick-person settled on postures that satisfied the goal of reaching the target and the goal of maintaining the center of gravity over the "feet."

Though the simulation was able to perform the task, eventually satisfying both goals at once, it had a number of inadequacies stemming from the fact that each joint processor attempted to achieve a solution in ignorance of what the other joints were attempting to do. This problem was overcome by using additional processors responsible for setting combinations of joint angles. Thus, a processor for flexion and extension of the leg would adjust the knee, hip, and ankle joints synergistically, while a processor for flexion and extension of the arm would adjust the shoulder and elbow together. With the addition of processors of this form, the number of iterations required to reach a solution was greatly reduced, and the form of the approach to the solution looked very natural. The sequence of configurations attained in one processing run is shown in Figure 5.

Explicit attempts to program a robot to cope with the problem of maintaining balance as it reaches for a desired target have revealed the difficulty of deriving explicitly the right combinations of actions for each possible starting state and goal state. This simple model illustrates that we may be wrong to seek such an explicit solution. We see here that a solution to the problem can emerge from the action of a number of simple processors each attempting to honor the constraints independently.

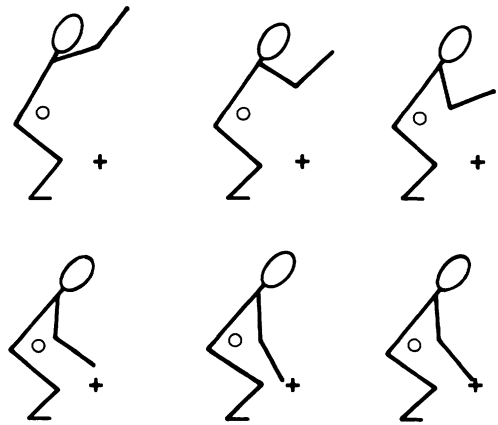

FIGURE 5. A sequence of configurations assumed by the stick "person" performing the reaching task described in the text, from Hinton (1984). The small circle represents the center of gravity of the whole stick-figure, and the cross represents the goal to be reached. The configuration is shown on every second iteration.

Perception

Stereoscopic vision. One early model using parallel distributed processing was the model of stereoscopic depth perception proposed by Marr and Poggio (1976). Their theory proposed to explain the perception of depth in random-dot stereograms (Julesz, 1971; see Figure 6) in terms of a simple distributed processing mechanism.

Julesz's random-dot stereograms present interesting challenges to mechanisms of depth perception. A stereogram consists of two random-dot patterns. In a simple stereogram such as the one shown here, one pattern is an exact copy of the other except that the pattern of dots in a region of one of the patterns is shifted horizontally with respect to the rest of the pattern. Each of the two patterns—corresponding to two retinal images—consists entirely of a pattern of random dots, so there is no information in either of the two views considered alone that can indicate the presence of different surfaces, let alone depth relations among those surfaces. Yet, when one of these dot patterns is projected to the left eye and the other to the right eye, an observer sees each region as a surface, with the shifted region hovering in front of or behind the other, depending on the direction of the shift.

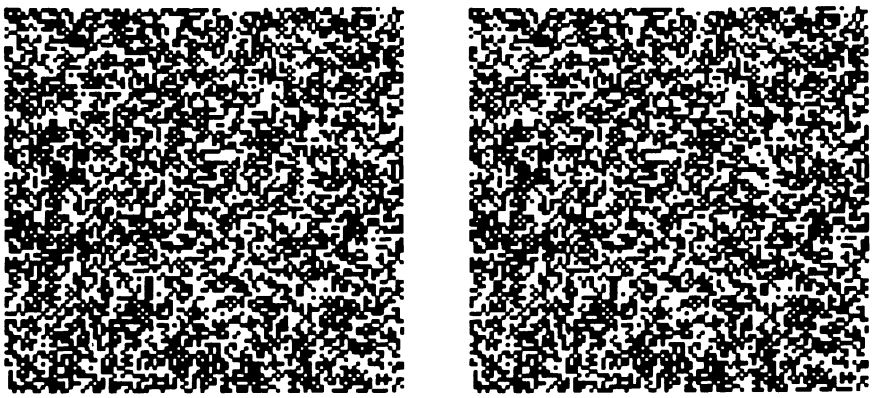

FIGURE 6. Random-dot stereograms. The two patterns are identical except that the pattern of dots in the central region of the left pattern are shifted over with respect to those in the right. When viewed stereoscopically such that the left pattern projects to the left eye and the right pattern to the right eye, the shifted area appears to hover above the page. Some readers may be able to achieve this by converging to a distant point (e.g., a far wall) and then interposing the figure into the line of sight. (From *Foundations of Cyclopean Perception*, p. 21, by B. Julesz, 1971, Chicago: University of Chicago Press. Copyright 1971 by Bell Telephone Laboratories, Inc. Reprinted by permission.)

What kind of a mechanism might we propose to account for these facts? Marr and Poggio (1976) began by explicitly representing the two views in two arrays, as human observers might in two different retinal images. They noted that corresponding black dots at different perceived distances from the observer will be offset from each other by different amounts in the two views. The job of the model is to determine which points correspond. This task is, of course, made difficult by the fact that there will be a very large number of spurious correspondences of individual dots. The goal of the mechanism, then, is to find those correspondences that represent real correspondences in depth and suppress those that represent spurious correspondences.

To carry out this task, Marr and Poggio assigned a processing unit to each possible conjunction of a point in one image and a point in the other. Since the eyes are offset horizontally, the possible conjunctions occur at various offsets or disparities along the horizontal dimension. Thus, for each point in one eye, there was a set of processing units with one unit assigned to the conjunction of that point and the point at each horizontal offset from it in the other eye.

Each processing unit received activation whenever both of the points the unit stood for contained dots. So far, then, units for both real and spurious correspondences would be equally activated. To allow the mechanism to find the right correspondences, they pointed out two general principles about the visual world: (a) Each point in each view generally corresponds to one and only one point in the other view, and (b) neighboring points in space tend to be at nearly the same depth and therefore at about the same disparity in the two images. While there are discontinuities at the edges of things, over most of a two-dimensional view of the world there will be continuity. These principles are called the *uniqueness* and *continuity* constraints, respectively.

Marr and Poggio incorporated these principles into the interconnections between the processing units. The uniqueness constraint was captured by inhibitory connections among the units that stand for alternative correspondences of the same dot. The continuity principle was captured by excitatory connections among the units that stand for similar offsets of adjacent dots.

These additional connections allow the Marr and Poggio model to "solve" stereograms like the one shown in the figure. At first, when a pair of patterns is presented, the units for all possible correspondences of a dot in one eye with a dot in the other will be equally excited. However, the excitatory connections cause the units for the correct conjunctions to receive more excitation than units for spurious conjunctions, and the inhibitory connections allow the units for the correct conjunctions to turn off the units for the spurious connections. Thus,

the model tends to settle down into a stable state in which only the correct correspondence of each dot remains active.

There are a number of reasons why Marr and Poggio (1979) modified this model (see Marr, 1982, for a discussion), but the basic mechanisms of mutual excitation between units that are mutually consistent and mutual inhibition between units that are mutually incompatible provide a natural mechanism for settling on the right conjunctions of points and rejecting spurious ones. The model also illustrates how general principles or rules such as the uniqueness and continuity principles may be embodied in the connections between processing units, and how behavior in accordance with these principles can emerge from the interactions determined by the pattern of these interconnections.

Perceptual completion of familiar patterns. Perception, of course, is influenced by familiarity. It is a well-known fact that we often misperceive unfamiliar objects as more familiar ones and that we can get by with less time or with lower-quality information in perceiving familiar items than we need for perceiving unfamiliar items. Not only does familiarity help us determine what the higher-level structures are when the lower-level information is ambiguous; it also allows us to fill in missing lower-level information within familiar higher-order patterns. The well-known *phonemic restoration effect* is a case in point. In this phenomenon, perceivers hear sounds that have been cut out of words as if they had actually been present. For example, Warren (1970) presented *legi#lature* to subjects, with a click in the location marked by the #. Not only did subjects correctly identify the word legislature; they also heard the missing /s/ just as though it had been presented. They had great difficulty localizing the click, which they tended to hear as a disembodied sound. Similar phenomena have been observed in visual perception of words since the work of Pillsbury (1897).

Two of us have proposed a model describing the role of familiarity in perception based on excitatory and inhibitory interactions among units standing for various hypotheses about the input at different levels of abstraction (McClelland & Rumelhart, 1981; Rumelhart & McClelland, 1982). The model has been applied in detail to the role of familiarity in the perception of letters in visually presented words, and has proved to provide a very close account of the results of a large number of experiments.

The model assumes that there are units that act as detectors for the visual features which distinguish letters, with one set of units assigned to detect the features in each of the different letter-positions in the word. For four-letter words, then, there are four such sets of detectors. There are also four sets of detectors for the letters themselves and a set of detectors for the words.

In the model, each unit has an activation value, corresponding roughly to the strength of the hypothesis that what that unit stands for is present in the perceptual input. The model honors the following important relations which hold between these "hypotheses" or activations: First, to the extent that two hypotheses are mutually consistent, they should support each other. Thus, units that are mutually consistent, in the way that the letter *T* in the first position is consistent with the word *TAKE*, tend to excite each other. Second, to the extent that two hypotheses are mutually inconsistent, they should weaken each other. Actually, we can distinguish two kinds of inconsistency: The first kind might be called between-level inconsistency. For example, the hypothesis that a word begins with a *T* is inconsistent with the hypothesis that the word is *MOVE*. The second might be called mutual exclusion. For example, the hypothesis that a word begins with *T* excludes the hypothesis that it begins with *R* since a word can only begin with one letter. Both kinds of inconsistencies operate in the word perception model to reduce the activations of units. Thus, the letter units in each position compete with all other letter units in the same position, and the word units compete with each other. This type of inhibitory interaction is often called *competitive inhibition*. In addition, there are inhibitory interactions between incompatible units on different levels. This type of inhibitory interaction is simply called *between-level inhibition*.

The set of excitatory and inhibitory interactions between units can be diagrammed by drawing excitatory and inhibitory links between them. The whole picture is too complex to draw, so we illustrate only with a fragment: Some of the interactions between some of the units in this model are illustrated in Figure 7.

Let us consider what happens in a system like this when a familiar stimulus is presented under degraded conditions. For example, consider the display shown in Figure 8. This display consists of the letters *W*, *O*, and *R*, completely visible, and enough of a fourth letter to rule out all letters other than *R* and *K*. Before onset of the display, the activations of the units are set at or below 0. When the display is presented, detectors for the features present in each position become active (i.e., their activations grow above 0). At this point, they begin to excite and inhibit the corresponding detectors for letters. In the first three positions, *W*, *O*, and *R* are unambiguously activated, so we will focus our attention on the fourth position where *R* and *K* are both equally consistent with the active features. Here, the activations of the detectors for *R* and *K* start out growing together, as the feature detectors below them become activated. As these detectors become active, they and the active letter detectors for *W*, *O*, and *R* in the other positions start to activate detectors for words which have these letters in

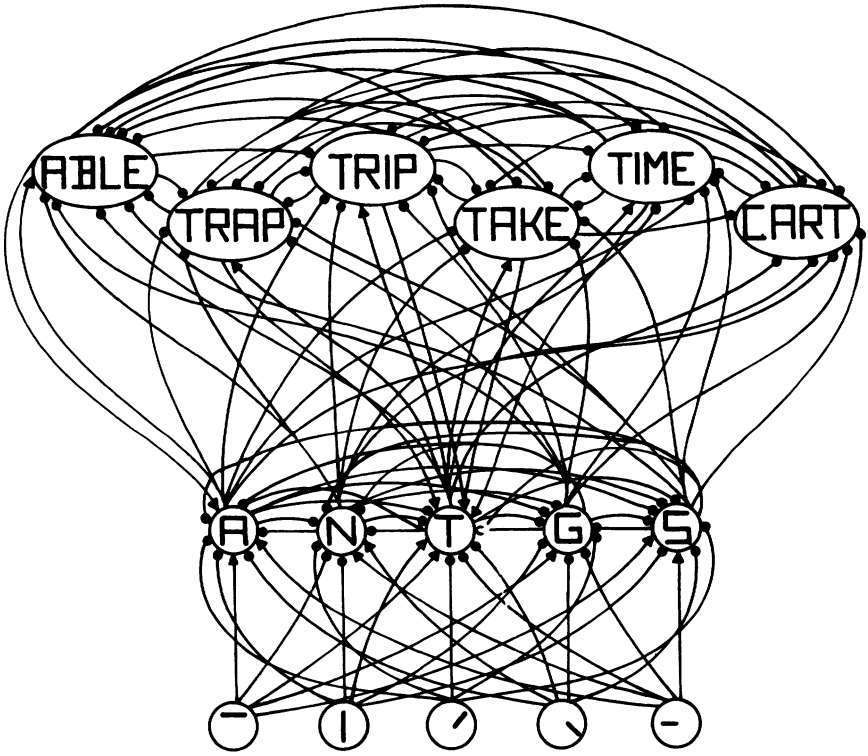

FIGURE 7. The unit for the letter *T* in the first position of a four-letter array and some of its neighbors. Note that the feature and letter units stand only for the first position; in a complete picture of the units needed from processing four-letter displays, there would be four full sets of feature detectors and four full sets of letter detectors. (From "An Interactive Activation Model of Context Effects in Letter Perception: Part 1. An Account of Basic Findings" by J. L. McClelland and D. E. Rumelhart, 1981, *Psychological Review, 88,* p. 380. Copyright 1981 by the American Psychological Association. Reprinted by permission.)

them and to inhibit detectors for words which do not have these letters. A number of words are partially consistent with the active letters, and receive some net excitation from the letter level, but only the word *WORK* matches one of the active letters in all four positions. As a result, *WORK* becomes more active than any other word and inhibits the other words, thereby successfully dominating the pattern of activation among the word units. As it grows in strength, it sends feedback to the letter level, reinforcing the activations of the *W, O, R,* and *K* in the corresponding positions. In the fourth position, this feedback gives *K* the upper hand over *R*, and eventually the stronger activation of the

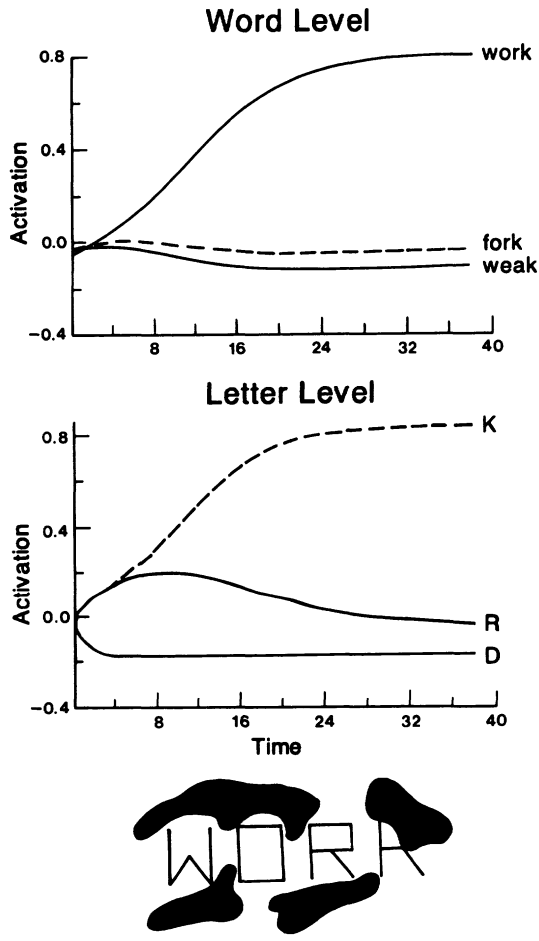

FIGURE 8. A possible display which might be presented to the interactive activation model of word recognition, and the resulting activations of selected letter and word units. The letter units are for the letters indicated in the fourth position of a four-letter display.

K detector allows it to dominate the pattern of activation, suppressing the *R* detector completely.

This example illustrates how PDP models can allow knowledge about what letters go together to form words to work together with natural constraints on the task (i.e., that there should only be one letter in one place at one time), to produce perceptual completion in a simple and direct way.

Completion of novel patterns. However, the perceptual intelligence of human perceivers far exceeds the ability to recognize familiar patterns and fill in missing portions. We also show facilitation in the

perception of letters in unfamiliar letter strings which are word-like but not themselves actually familiar.

One way of accounting for such performances is to imagine that the perceiver possesses, in addition to detectors for familiar words, sets of detectors for regular subword units such as familiar letter clusters, or that they use abstract rules, specifying which classes of letters can go with which others in different contexts. It turns out, however, that the model we have already described needs no such additional structure to produce perceptual facilitation for word-like letter strings; to this extent it acts as if it "knows" the orthographic structure of English. We illustrate this feature of the model with the example shown in Figure 9, where the nonword *YEAD* is shown in degraded form so that the second letter is incompletely visible. Given the information about this letter, considered alone, either *E* or *F* would be possible in the second position. Yet our model will tend to complete this letter as an *E*.

The reason for this behavior is that, when *YEAD* is shown, a number of words are partially activated. There is no word consistent with *Y, E* or *F, A,* and *D*, but there are words which match *YEA_* (*YEAR*, for example) and others which match *_EAD* (*BEAD, DEAD, HEAD,* and *READ*, for example). These and other near misses are partially activated as a result of the pattern of activation at the letter level. While they compete with each other, none of these words gets strongly enough activated to completely suppress all the others. Instead, these units act as a group to reinforce particularly the letters *E* and *A*. There are no close partial matches which include the letter *F* in the second position, so this letter receives no feedback support. As a result, *E* comes to dominate, and eventually suppress, the *F* in the second position.

The fact that the word perception model exhibits perceptual facilitation to pronounceable nonwords as well as words illustrates once again how behavior in accordance with general principles or rules can emerge from the interactions of simple processing elements. Of course, the behavior of the word perception model does not implement exactly any of the systems of orthographic rules that have been proposed by linguists (Chomsky & Halle, 1968; Venesky, 1970) or psychologists (Spoehr & Smith, 1975). In this regard, it only approximates such rule-based descriptions of perceptual processing. However, rule systems such as Chomsky and Halle's or Venesky's appear to be only approximately honored in human performance as well (Smith & Baker, 1976). Indeed, some of the discrepancies between human performance data and rule systems occur in exactly the ways that we would predict from the word perception model (Rumelhart & McClelland, 1982). This illustrates the possibility that PDP models may provide more accurate accounts of the details of human performance than models

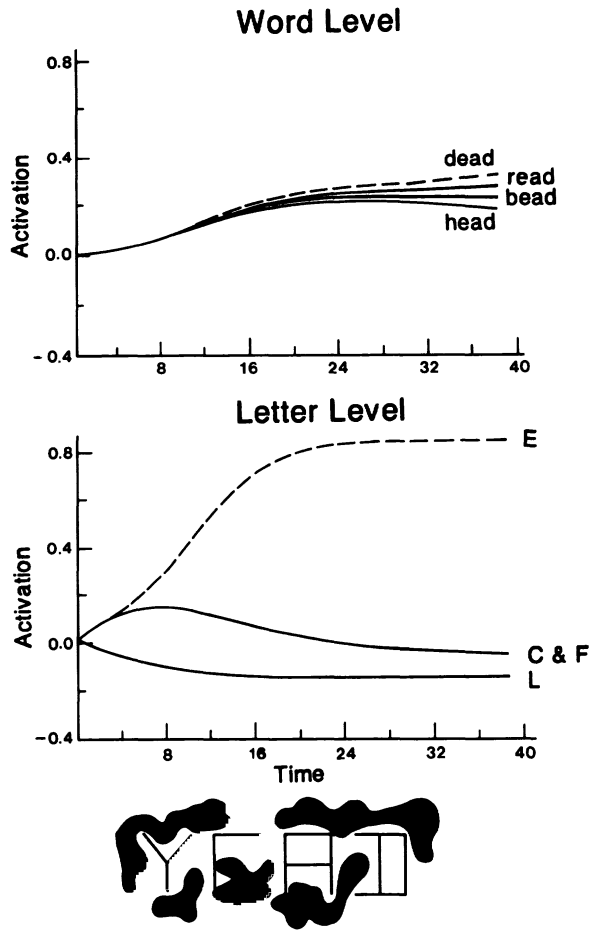

FIGURE 9. An example of a nonword display that might be presented to the interactive activation model of word recognition and the response of selected units at the letter and word levels. The letter units illustrated are detectors for letters in the second input position.

based on a set of rules representing human competence—at least in some domains.

Retrieving Information From Memory

Content addressability. One very prominent feature of human memory is that it is content addressable. It seems fairly clear that we

can access information in memory based on nearly any attribute of the representation we are trying to retrieve.

Of course, some cues are much better than others. An attribute which is shared by a very large number of things we know about is not a very effective retrieval cue, since it does not accurately pick out a particular memory representation. But, several such cues, in conjunction, can do the job. Thus, if we ask a friend who goes out with several women, "Who was that woman I saw you with?", he may not know which one we mean—but if we specify something else about her—say the color of her hair, what she was wearing (in so far as he remembers this at all), where we saw him with her—he will likely be able to hit upon the right one.

It is, of course, possible to implement some kind of content addressability of memory on a standard computer in a variety of different ways. One way is to search sequentially, examining each memory in the system to find the memory or the set of memories which has the particular content specified in the cue. An alternative, somewhat more efficient, scheme involves some form of indexing—keeping a list, for every content a memory might have, of which memories have that content.

Such an indexing scheme can be made to work with error-free probes, but it will break down if there is an error in the specification of the retrieval cue. There are possible ways of recovering from such errors, but they lead to the kind of combinatorial explosions which plague this kind of computer implementation.

But suppose that we imagine that each memory is represented by a unit which has mutually excitatory interactions with units standing for each of its properties. Then, whenever any property of the memory became active, the memory would tend to be activated, and whenever the memory was activated, all of its contents would tend to become activated. Such a scheme would automatically produce content addressability for us. Though it would not be immune to errors, it would not be devastated by an error in the probe if the remaining properties specified the correct memory.

As described thus far, whenever a property that is a part of a number of different memories is activated, it will tend to activate all of the memories it is in. To keep these other activities from swamping the "correct" memory unit, we simply need to add initial inhibitory connections among the memory units. An additional desirable feature would be mutually inhibitory interactions among mutually incompatible property units. For example, a person cannot both be single and married at the same time, so the units for different marital states would be mutually inhibitory.

McClelland (1981) developed a simulation model that illustrates how a system with these properties would act as a content addressable memory. The model is obviously oversimplified, but it illustrates many of the characteristics of the more complex models that will be considered in later chapters.

Consider the information represented in Figure 10, which lists a number of people we might meet if we went to live in an unsavory neighborhood, and some of their hypothetical characteristics. A subset

The Jets and The Sharks

Name	Gang	Age	Edu	Mar	Occupation
Art	Jets	40's	J.H.	Sing.	Pusher
Al	Jets	30's	J.H.	Mar.	Burglar
Sam	Jets	20's	COL.	Sing.	Bookie
Clyde	Jets	40's	J.H.	Sing.	Bookie
Mike	Jets	30's	J.H.	Sing.	Bookie
Jim	Jets	20's	J.H.	Div.	Burglar
Greg	Jets	20's	H.S.	Mar.	Pusher
John	Jets	20's	J.H.	Mar.	Burglar
Doug	Jets	30's	H.S.	Sing.	Bookie
Lance	Jets	20's	J.H.	Mar.	Burglar
George	Jets	20's	J.H.	Div.	Burglar
Pete	Jets	20's	H.S.	Sing.	Bookie
Fred	Jets	20's	H.S.	Sing.	Pusher
Gene	Jets	20's	COL.	Sing.	Pusher
Ralph	Jets	30's	J.H.	Sing.	Pusher
Phil	Sharks	30's	COL.	Mar.	Pusher
Ike	Sharks	30's	J.H.	Sing.	Bookie
Nick	Sharks	30's	H.S.	Sing.	Pusher
Don	Sharks	30's	COL.	Mar.	Burglar
Ned	Sharks	30's	COL.	Mar.	Bookie
Karl	Sharks	40's	H.S.	Mar.	Bookie
Ken	Sharks	20's	H.S.	Sing.	Burglar
Earl	Sharks	40's	H.S.	Mar.	Burglar
Rick	Sharks	30's	H.S.	Div.	Burglar
Ol	Sharks	30's	COL.	Mar.	Pusher
Neal	Sharks	30's	H.S.	Sing.	Bookie
Dave	Sharks	30's	H.S.	Div.	Pusher

FIGURE 10. Characteristics of a number of individuals belonging to two gangs, the Jets and the Sharks. (From "Retrieving General and Specific Knowledge From Stored Knowledge of Specifics" by J. L. McClelland, 1981, *Proceedings of the Third Annual Conference of the Cognitive Science Society*, Berkeley, CA. Copyright 1981 by J. L. McClelland. Reprinted by permission.)

of the units needed to represent this information is shown in Figure 11. In this network, there is an "instance unit" for each of the characters described in Figure 10, and that unit is linked by mutually excitatory connections to all of the units for the fellow's properties. Note that we have included property units for the names of the characters, as well as units for their other properties.

Now, suppose we wish to retrieve the properties of a particular individual, say Lance. And suppose that we know Lance's name. Then we can probe the network by activating Lance's name unit, and we can see what pattern of activation arises as a result. Assuming that we know of no one else named Lance, we can expect the Lance name unit to be hooked up only to the instance unit for Lance. This will in turn activate the property units for Lance, thereby creating the pattern of

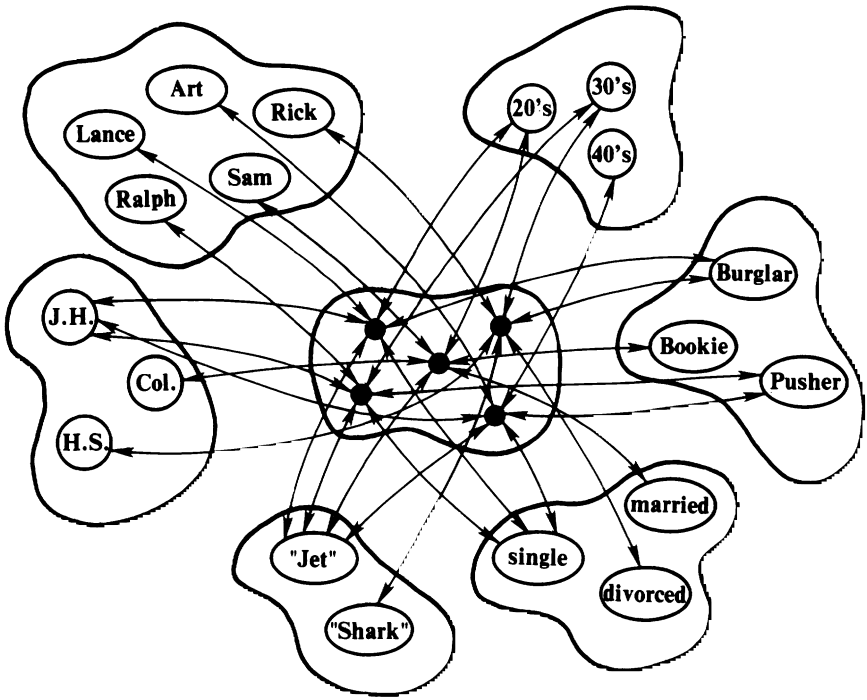

FIGURE 11. Some of the units and interconnections needed to represent the individuals shown in Figure 10. The units connected with double-headed arrows are mutually excitatory. All the units within the same cloud are mutually inhibitory. (From "Retrieving General and Specific Knowledge From Stored Knowledge of Specifics" by J. L. McClelland, 1981, *Proceedings of the Third Annual Conference of the Cognitive Science Society*, Berkeley, CA. Copyright 1981 by J. L. McClelland. Reprinted by permission.)

activation corresponding to Lance. In effect, we have retrieved a representation of Lance. More will happen than just what we have described so far, but for the moment let us stop here.

Of course, sometimes we may wish to retrieve a name, given other information. In this case, we might start with some of Lance's properties, effectively asking the system, say "Who do you know who is a Shark and in his 20s?" by activating the Shark and 20s units. In this case it turns out that there is a single individual, Ken, who fits the description. So, when we activate these two properties, we will activate the instance unit for Ken, and this in turn will activate his name unit, and fill in his other properties as well.

Graceful degradation. A few of the desirable properties of this kind of model are visible from considering what happens as we vary the set of features we use to probe the memory in an attempt to retrieve a particular individual's name. Any set of features which is sufficient to uniquely characterize a particular item will activate the instance node for that item more strongly than any other instance node. A probe which contains misleading features will most strongly activate the node that it matches best. This will clearly be a poorer cue than one which contains no misleading information—but it will still be sufficient to activate the "right answer" more strongly than any other, as long as the introduction of misleading information does not make the probe closer to some other item. In general, though the degree of activation of a particular instance node and of the corresponding name nodes varies in this model as a function of the exact content of the probe, errors in the probe will not be fatal unless they make the probe point to the wrong memory. This kind of model's handling of incomplete or partial probes also requires no special error-recovery scheme to work—it is a natural by-product of the nature of the retrieval mechanism that it is capable of graceful degradation.

These aspects of the behavior of the Jets and Sharks model deserve more detailed consideration than the present space allows. One reason we do not go into them is that we view this model as a stepping stone in the development of other models, such as the models using more distributed representations, that occur in other parts of this book. We do, however, have more to say about this simple model, for like some of the other models we have already examined, this model exhibits some useful properties which emerge from the interactions of the processing units.

Default assignment. It probably will have occurred to the reader that in many of the situations we have been examining, there will be other

activations occurring which may influence the pattern of activation which is retrieved. So, in the case where we retrieved the properties of Lance, those properties, once they become active, can begin to activate the units for other individuals with those same properties. The memory unit for Lance will be in competition with these units and will tend to keep their activation down, but to the extent that they do become active, they will tend to activate their own properties and therefore fill them in. In this way, the model can fill in properties of individuals based on what it knows about other, similar instances.

To illustrate how this might work we have simulated the case in which we do not know that Lance is a Burglar as opposed to a Bookie or a Pusher. It turns out that there are a group of individuals in the set who are very similar to Lance in many respects. When Lance's properties become activated, these other units become partially activated, and they start activating their properties. Since they all share the same "occupation," they work together to fill in that property for Lance. Of course, there is no reason why this should necessarily be the right answer, but generally speaking, the more similar two things are in respects that we know about, the more likely they are to be similar in respects that we do not, and the model implements this heuristic.

Spontaneous generalization. The model we have been describing has another valuable property as well—it tends to retrieve what is common to those memories which match a retrieval cue which is too general to capture any one memory. Thus, for example, we could probe the system by activating the unit corresponding to membership in the Jets. This unit will partially activate all the instances of the Jets, thereby causing each to send activations to its properties. In this way the model can retrieve the typical values that the members of the Jets have on each dimension—even though there is no one Jet that has these typical values. In the example, 9 of 15 Jets are single, 9 of 15 are in their 20s, and 9 of 15 have only a Junior High School education; when we probe by activating the Jet unit, all three of these properties dominate. The Jets are evenly divided between the three occupations, so each of these units becomes partially activated. Each has a different name, so that each name unit is very weakly activated, nearly cancelling each other out.

In the example just given of spontaneous generalization, it would not be unreasonable to suppose that someone might have explicitly stored a generalization about the members of a gang. The account just given would be an alternative to "explicit storage" of the generalization. It has two advantages, though, over such an account. First, it does not require any special generalization formation mechanism. Second, it can provide us with generalizations on unanticipated lines, on demand.

Thus, if we want to know, for example, what people in their 20s with a junior high school education are like, we can probe the model by activating these two units. Since all such people are Jets and Burglars, these two units are strongly activated by the model in this case; two of them are divorced and two are married, so both of these units are partially activated.[1]

The sort of model we are considering, then, is considerably more than a content addressable memory. In addition, it performs default assignment, and it can spontaneously retrieve a general concept of the individuals that match any specifiable probe. These properties must be explicitly implemented as complicated computational extensions of other models of knowledge retrieval, but in PDP models they are natural by-products of the retrieval process itself.

REPRESENTATION AND LEARNING IN PDP MODELS

In the Jets and Sharks model, we can speak of the model's *active representation* at a particular time, and associate this with the pattern of activation over the units in the system. We can also ask: What is the stored knowledge that gives rise to that pattern of activation? In considering this question, we see immediately an important difference between PDP models and other models of cognitive processes. In most models, knowledge is stored as a static copy of a pattern. Retrieval amounts to finding the pattern in long-term memory and copying it into a buffer or working memory. There is no real difference between the stored representation in long-term memory and the active representation in working memory. In PDP models, though, this is not the case. In these models, the patterns themselves are not stored. Rather, what is stored is the *connection strengths* between units that allow these patterns to be re-created. In the Jets and Sharks model, there is an instance unit assigned to each individual, but that unit does not contain a copy of the representation of that individual. Instead, it is simply the case that the connections between it and the other units in the system are such that activation of the unit will cause the pattern for the individual to be reinstated on the property units.

---

[1] In this and all other cases, there is a tendency for the pattern of activation to be influenced by partially activated, near neighbors, which do not quite match the probe. Thus, in this case, there is a Jet Al, who is a Married Burglar. The unit for Al gets slightly activated, giving Married a slight edge over Divorced in the simulation.

This difference between PDP models and conventional models has enormous implications, both for processing and for learning. We have already seen some of the implications for processing. The representation of the knowledge is set up in such a way that the knowledge necessarily influences the course of processing. Using knowledge in processing is no longer a matter of finding the relevant information in memory and bringing it to bear; it is part and parcel of the processing itself.

For learning, the implications are equally profound. For if the knowledge is the strengths of the connections, learning must be a matter of finding the right connection strengths so that the right patterns of activation will be produced under the right circumstances. This is an extremely important property of this class of models, for it opens up the possibility that an information processing mechanism could learn, as a result of tuning its connections, to capture the interdependencies between activations that it is exposed to in the course of processing.

In recent years, there has been quite a lot of interest in learning in cognitive science. Computational approaches to learning fall predominantly into what might be called the "explicit rule formulation" tradition, as represented by the work of Winston (1975), the suggestions of Chomsky, and the ACT* model of J. R. Anderson (1983). All of this work shares the assumption that the goal of learning is to formulate explicit rules (propositions, productions, etc.) which capture powerful generalizations in a succinct way. Fairly powerful mechanisms, usually with considerable innate knowledge about a domain, and/or some starting set of primitive propositional representations, then formulate hypothetical general rules, e.g., by comparing particular cases and formulating explicit generalizations.

The approach that we take in developing PDP models is completely different. First, we do not assume that the goal of learning is the formulation of explicit rules. Rather, we assume it is the acquisition of connection strengths which allow a network of simple units to act *as though* it knew the rules. Second, we do not attribute powerful computational capabilities to the learning mechanism. Rather, we assume very simple connection strength modulation mechanisms which adjust the strength of connections between units based on information locally available at the connection.

These issues will be addressed at length in later sections of this book. For now, our purpose is to give a simple, illustrative example of the connection strength modulation process, and how it can produce networks which exhibit some interesting behavior.

Local vs. distributed representation. Before we turn to an explicit consideration of this issue, we raise a basic question about

representation. Once we have achieved the insight that the knowledge is stored in the strengths of the interconnections between units, a question arises. Is there any reason to assign one unit to each pattern that we wish to learn? Another possibility—one that we explore extensively in this book—is the possibility that the knowledge about any individual pattern is not stored in the connections of a special unit reserved for that pattern, but is distributed over the connections among a large number of processing units. On this view, the Jets and Sharks model represents a special case in which separate units are reserved for each instance.

Models in which connection information is explicitly thought of as distributed have been proposed by a number of investigators. The units in these collections may themselves correspond to conceptual primitives, or they may have no particular meaning as individuals. In either case, the focus shifts to patterns of activation over these units and to mechanisms whose explicit purpose is to learn the right connection strengths to allow the right patterns of activation to become activated under the right circumstances.

In the rest of this section, we will give a simple example of a PDP model in which the knowledge is distributed. We will first explain how the model would work, given pre-existing connections, and we will then describe how it could come to acquire the right connection strengths through a very simple learning mechanism. A number of models which have taken this distributed approach have been discussed in this book's predecessor, Hinton and J. A. Anderson's (1981) *Parallel Models of Associative Memory.* We will consider a simple version of a common type of distributed model, a *pattern associator.*

Pattern associators are models in which a pattern of activation over one set of units can cause a pattern of activation over another set of units without any intervening units to stand for either pattern as a whole. Pattern associators would, for example, be capable of associating a pattern of activation on one set of units corresponding to the appearance of an object with a pattern on another set corresponding to the aroma of the object, so that, when an object is presented visually, causing its visual pattern to become active, the model produces the pattern corresponding to its aroma.

How a pattern associator works. For purposes of illustration, we present a very simple pattern associator in Figure 12. In this model, there are four units in each of two pools. The first pool, the A units, will be the pool in which patterns corresponding to the sight of various objects might be represented. The second pool, the B units, will be the pool in which the pattern corresponding to the aroma will be represented. We can pretend that alternative patterns of activation on

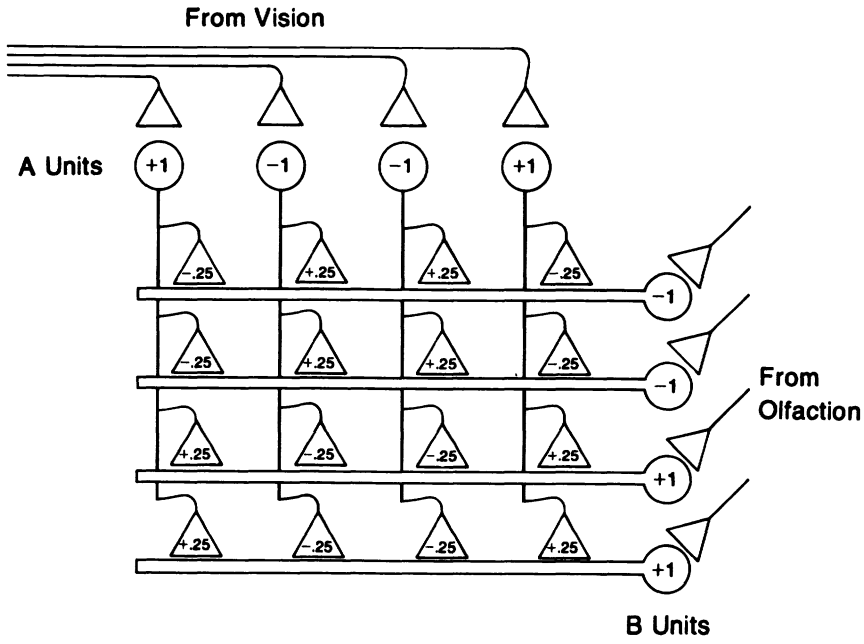

FIGURE 12. A simple pattern associator. The example assumes that patterns of activation in the A units can be produced by the visual system and patterns in the B units can be produced by the olfactory system. The synaptic connections allow the outputs of the A units to influence the activations of the B units. The synaptic weights linking the A units to the B units were selected so as to allow the pattern of activation shown on the A units to reproduce the pattern of activation shown on the B units without the need for any olfactory input.

the A units are produced upon viewing a rose or a grilled steak, and alternative patterns on the B units are produced upon sniffing the same objects. Figure 13 shows two pairs of patterns, as well as sets of interconnections necessary to allow the A member of each pair to reproduce the B member.

The details of the behavior of the individual units vary among different versions of pattern associators. For present purposes, we'll assume that the units can take on positive or negative activation values, with 0 representing a kind of neutral intermediate value. The strengths of the interconnections between the units can be positive or negative real numbers.

The effect of an A unit on a B unit is determined by multiplying the activation of the A unit times the strength of its synaptic connection with the B unit. For example, if the connection from a particular A unit to a particular B unit has a positive sign, when the A unit is

+1	−1	−1	+1	
−.25	+.25	+.25	−.25	−1
−.25	+.25	+.25	−.25	−1
+.25	−.25	−.25	+.25	+1
+.25	−.25	−.25	+.25	+1

−1	+1	−1	+1	
+.25	−.25	+.25	−.25	−1
−.25	+.25	−.25	+.25	+1
−.25	+.25	−.25	+.25	+1
+.25	−.25	+.25	−.25	−1

FIGURE 13. Two simple associators represented as matrices. The weights in the first two matrices allow the A pattern shown above the matrix to produce the B pattern shown to the right of it. Note that the weights in the first matrix are the same as those shown in the diagram in Figure 12.

excited (activation greater than 0), it will excite the B unit. For this example, we'll simply assume that the activation of each unit is set to the sum of the excitatory and inhibitory effects operating on it. This is one of the simplest possible cases.

Suppose, now, that we have created on the A units the pattern corresponding to the first visual pattern shown in Figure 13, the rose. How should we arrange the strengths of the interconnections between the A units and the B units to reproduce the pattern corresponding to the aroma of a rose? We simply need to arrange for each A unit to tend to excite each B unit which has a positive activation in the aroma pattern and to inhibit each B unit which has a negative activation in the aroma pattern. It turns out that this goal is achieved by setting the strength of the connection between a given A unit and a given B unit to a value proportional to the product of the activation of the two units. In Figure 12, the weights on the connections were chosen to allow the A pattern illustrated there to produce the illustrated B pattern according to this principle. The actual strengths of the connections were set to $\pm .25$, rather than ± 1, so that the A pattern will produce the right magnitude, as well as the right sign, for the activations of the units in the B pattern. The same connections are reproduced in matrix form in Figure 13A.

Pattern associators like the one in Figure 12 have a number of nice properties. One is that they do not require a perfect copy of the input to produce the correct output, though its strength will be weaker in this case. For example, suppose that the associator shown in Figure 12 were presented with an A pattern of $(1,-1,0,1)$. This is the A pattern shown in the figure, with the activation of one of its elements set to 0. The B pattern produced in response will have the activations of all of the B units in the right direction; however, they will be somewhat weaker than they would be, had the complete A pattern been shown. Similar

effects are produced if an element of the pattern is distorted—or if the model is damaged, either by removing whole units, or random sets of connections, etc. Thus, their pattern retrieval performance of the model degrades gracefully both under degraded input and under damage.

How a pattern associator learns. So far, we have seen how we as model builders can construct the right set of weights to allow one pattern to cause another. The interesting thing, though, is that we do not need to build these interconnection strengths in by hand. Instead, the pattern associator can teach itself the right set of interconnections through experience processing the patterns in conjunction with each other.

A number of different rules for adjusting connection strengths have been proposed. One of the first—and definitely the best known—is due to D. O. Hebb (1949). Hebb's actual proposal was not sufficiently quantitative to build into an explicit model. However, a number of different variants can trace their ancestry back to Hebb. Perhaps the simplest version is:

> When unit A and unit B are simultaneously excited, increase the strength of the connection between them.

A natural extension of this rule to cover the positive and negative activation values allowed in our example is:

> Adjust the strength of the connection between units A and B in proportion to the product of their simultaneous activation.

In this formulation, if the product is positive, the change makes the connection more excitatory, and if the product is negative, the change makes the connection more inhibitory. For simplicity of reference, we will call this the *Hebb rule*, although it is not exactly Hebb's original formulation.

With this simple learning rule, we could train a "blank copy" of the pattern associator shown in Figure 12 to produce the B pattern for rose when the A pattern is shown, simply by presenting the A and B patterns together and modulating the connection strengths according to the Hebb rule. The size of the change made on every trial would, of course, be a parameter. We generally assume that the changes made on each instance are rather small, and that connection strengths build up gradually. The values shown in Figure 13A, then, would be acquired as a result of a number of experiences with the A and B pattern pair.

It is very important to note that the information needed to use the Hebb rule to determine the value each connection should have is *locally available* at the connection. All a given connection needs to consider is the activation of the units on both sides of it. Thus, it would be possible to actually implement such a connection modulation scheme locally, in each connection, without requiring any programmer to reach into each connection and set it to just the right value.

It turns out that the Hebb rule as stated here has some serious limitations, and, to our knowledge, no theorists continue to use it in this simple form. More sophisticated connection modulation schemes have been proposed by other workers; most important among these are the delta rule, discussed extensively in Chapters 8 and 11; the competitive learning rule, discussed in Chapter 5; and the rules for learning in stochastic parallel models, described in Chapters 6 and 7. All of these learning rules have the property that they adjust the strengths of connections between units on the basis of information that can be assumed to be locally available to the unit. Learning, then, in all of these cases, amounts to a very simple process that can be implemented locally at each connection without the need for any overall supervision. Thus, models which incorporate these learning rules train themselves to have the right interconnections in the course of processing the members of an ensemble of patterns.

Learning multiple patterns in the same set of interconnections. Up to now, we have considered how we might teach our pattern associator to associate the visual pattern for one object with a pattern for the aroma of the same object. Obviously, different patterns of interconnections between the A and B units are appropriate for causing the visual pattern for a different object to give rise to the pattern for its aroma. The same principles apply, however, and if we presented our pattern associator with the A and B patterns for steak, it would learn the right set of interconnections for that case instead (these are shown in Figure 13B). In fact, it turns out that we can actually teach the same pattern associator a number of different associations. The matrix representing the set of interconnections that would be learned if we taught the same pattern associator both the rose association and the steak association is shown in Figure 14. The reader can verify this by adding the two matrices for the individual patterns together. The reader can also verify that this set of connections will allow the rose A pattern to produce the rose B pattern, and the steak A pattern to produce the steak B pattern: when either input pattern is presented, the correct corresponding output is produced.

The examples used here have the property that the two different visual patterns are completely uncorrelated with each other. This being

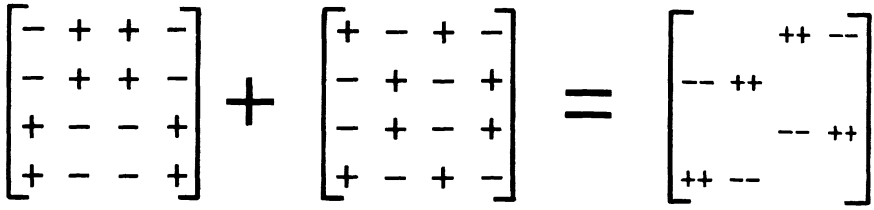

FIGURE 14. The weights in the third matrix allow either A pattern shown in Figure 13 to recreate the corresponding B pattern. Each weight in this case is equal to the sum of the weight for the A pattern and the weight for the B pattern, as illustrated.

the case, the rose pattern produces no effect when the interconnections for the steak have been established, and the steak pattern produces no effect when the interconnections for the rose association are in effect. For this reason, it is possible to add together the pattern of interconnections for the rose association and the pattern for the steak association, and still be able to associate the sight of the steak with the smell of a steak and the sight of a rose with the smell of a rose. The two sets of interconnections do not interact at all.

One of the limitations of the Hebbian learning rule is that it can learn the connection strengths appropriate to an entire ensemble of patterns only when all the patterns are completely uncorrelated. This restriction does not, however, apply to pattern associators which use more sophisticated learning schemes.

Attractive properties of pattern associator models. Pattern associator models have the property that uncorrelated patterns do not interact with each other, but more similar ones do. Thus, to the extent that a new pattern of activation on the A units is similar to one of the old ones, it will tend to have similar effects. Furthermore, if we assume that learning the interconnections occurs in small increments, similar patterns will essentially reinforce the strengths of the links they share in common with other patterns. Thus, if we present the same pair of patterns over and over, but each time we add a little random noise to each element of each member of the pair, the system will automatically learn to associate the central tendency of the two patterns and will learn to ignore the noise. What will be stored will be an average of the similar patterns with the slight variations removed. On the other hand, when we present the system with completely uncorrelated patterns, they will not interact with each other in this way. Thus, the same pool of units can extract the central tendency of each of a number of pairs of unrelated patterns. This aspect of distributed models is exploited extensively in Chapters 17 and 25 on distributed memory and amnesia.

Extracting the structure of an ensemble of patterns. The fact that similar patterns tend to produce similar effects allows distributed models to exhibit a kind of spontaneous generalization, extending behavior appropriate for one pattern to other similar patterns. This property is shared by other PDP models, such as the word perception model and the Jets and Sharks model described above; the main difference here is in the existence of simple, local, learning mechanisms that can allow the acquisition of the connection strengths needed to produce these generalizations through experience with members of the ensemble of patterns. Distributed models have another interesting property as well: If there are regularities in the correspondences between pairs of patterns, the model will naturally extract these regularities. This property allows distributed models to acquire patterns of interconnections that lead them to behave in ways we ordinarily take as evidence for the use of linguistic rules.

A detailed example of such a model is described in Chapter 18. Here, we describe the model very briefly. The model is a mechanism that learns how to construct the past tenses of words from their root forms through repeated presentations of examples of root forms paired with the corresponding past-tense form. The model consists of two pools of units. In one pool, patterns of activation representing the phonological structure of the root form of the verb can be represented, and, in the other, patterns representing the phonological structure of the past tense can be represented. The goal of the model is simply to learn the right connection strengths between the root units and the past-tense units, so that whenever the root form of a verb is presented the model will construct the corresponding past-tense form. The model is trained by presenting the root form of the verb as a pattern of activation over the root units, and then using a simple, local, learning rule to adjust the connection strengths so that this root form will tend to produce the correct pattern of activation over the past-tense units. The model is tested by simply presenting the root form as a pattern of activation over the root units and examining the pattern of activation produced over the past-tense units.

The model is trained initially with a small number of verbs children learn early in the acquisition process. At this point in learning, it can only produce appropriate outputs for inputs that it has explicitly been shown. But as it learns more and more verbs, it exhibits two interesting behaviors. First, it produces the standard *ed* past tense when tested with pseudo-verbs or verbs it has never seen. Second, it "overregularizes" the past tense of irregular words it previously completed correctly. Often, the model will blend the irregular past tense of the word with the regular *ed* ending, and produce errors like *CAMED* as the past of

COME. These phenomena mirror those observed in the early phases of acquisition of control over past tenses in young children.

The generativity of the child's responses—the creation of regular past tenses of new verbs and the overregularization of the irregular verbs—has been taken as strong evidence that the child has induced the rule which states that the regular correspondence for the past tense in English is to add a final *ed* (Berko, 1958). On the evidence of its performance, then, the model can be said to have acquired the rule. However, no special rule-induction mechanism is used, and no special language-acquisition device is required. The model learns to behave in accordance with the rule, not by explicitly noting that most words take *ed* in the past tense in English and storing this rule away explicitly, but simply by building up a set of connections in a pattern associator through a long series of simple learning experiences. The same mechanisms of parallel distributed processing and connection modification which are used in a number of domains serve, in this case, to produce implicit knowledge tantamount to a linguistic rule. The model also provides a fairly detailed account of a number of the specific aspects of the error patterns children make in learning the rule. In this sense, it provides a richer and more detailed description of the acquisition process than any that falls out naturally from the assumption that the child is building up a repertoire of explicit but inaccessible rules.

There is a lot more to be said about distributed models of learning, about their strengths and their weaknesses, than we have space for in this preliminary consideration. For now we hope mainly to have suggested that they provide dramatically different accounts of learning and acquisition than are offered by traditional models of these processes. We saw in earlier sections of this chapter that performance in accordance with rules can emerge from the interactions of simple, interconnected units. Now we can see how the aquisition of performance that conforms to linguistic rules can emerge from a simple, local, connection strength modulation process.

We have seen what the properties of PDP models are in informal terms, and we have seen how these properties operate to make the models do many of the kinds of things that they do. The business of the next chapter is to lay out these properties more formally, and to introduce some formal tools for their description and analysis. Before we turn to this, however, we wish to describe some of the major sources of inspiration for the PDP approach.

ORIGINS OF PARALLEL DISTRIBUTED PROCESSING

The ideas behind the PDP approach have a history that stretches back indefinitely. In this section, we mention briefly some of the people who have thought in these terms, particularly those whose work has had an impact on our own thinking. This section should not been seen as an authoritative review of the history, but only as a description of our own sources of inspiration.

Some of the earliest roots of the PDP approach can be found in the work of the unique neurologists, Jackson (1869/1958) and Luria (1966). Jackson was a forceful and persuasive critic of the simplistic localizationist doctrines of late nineteenth century neurology, and he argued convincingly for distributed, multilevel conceptions of processing systems. Luria, the Russian psychologist and neurologist, put forward the notion of the *dynamic functional system.* On this view, every behavioral or cognitive process resulted from the coordination of a large number of different components, each roughly localized in different regions of the brain, but all working together in dynamic interaction. Neither Hughlings-Jackson nor Luria is noted for the clarity of his views, but we have seen in their ideas a rough characterization of the kind of parallel distributed processing system we envision.

Two other contributors to the deep background of PDP were Hebb (1949) and Lashley (1950). We already have noted Hebb's contribution of the Hebb rule of synaptic modification; he also introduced the concept of cell assemblies—a concrete example of a limited form of distributed processing—and discussed the idea of reverberation of activation within neural networks. Hebb's ideas were cast more in the form of speculations about neural functioning than in the form of concrete processing models, but his thinking captures some of the flavor of parallel distributed processing mechanisms. Lashley's contribution was to insist upon the idea of distributed representation. Lashley may have been too radical and too vague, and his doctrine of equipotentiality of broad regions of cortex clearly overstated the case. Yet many of his insights into the difficulties of storing the "engram" locally in the brain are telling, and he seemed to capture quite precisely the essence of distributed representation in insisting that "there are no special cells reserved for special memories" (Lashley, 1950, p. 500).

In the 1950s, there were two major figures whose ideas have contributed to the development of our approach. One was Rosenblatt (1959, 1962) and the other was Selfridge (1955). In his *Principles of Neurodynamics* (1962), Rosenblatt articulated clearly the promise of a neurally inspired approach to computation, and he developed the *perceptron convergence procedure*, an important advance over the Hebb rule for

changing synaptic connections. Rosenblatt's work was very controversial at the time, and the specific models he proposed were not up to all the hopes he had for them. But his vision of the human information processing system as a dynamic, interactive, self-organizing system lies at the core of the PDP approach. Selfridge's contribution was his insistence on the importance of interactive processing, and the development of *Pandemonium*, an explicitly computational example of a dynamic, interactive mechanism applied to computational problems in perception.

In the late 60s and early 70s, serial processing and the von Neumann computer dominated both psychology and artificial intelligence, but there were a number of researchers who proposed neural mechanisms which capture much of the flavor of PDP models. Among these figures, the most influential in our work have been J. A. Anderson, Grossberg, and Longuet-Higgins. Grossberg's mathematical analysis of the properties of neural networks led him to many insights we have only come to appreciate through extensive experience with computer simulation, and he deserves credit for seeing the relevance of neurally inspired mechanisms in many areas of perception and memory well before the field was ready for these kinds of ideas (Grossberg, 1978). Grossberg (1976) was also one of the first to analyze some of the properties of the competitive learning mechanism explored in Chapter 5. Anderson's work differs from Grossberg's in insisting upon distributed representation, and in showing the relevance of neurally inspired models for theories of concept learning (Anderson, 1973, 1977); the work in Chapters 17 and 25 on distributed memory and amnesia owes a great deal to Anderson's inspiration. Anderson's work also played a crucial role in the formulation of the *cascade* model (McClelland, 1979), a step away from serial processing down the road to PDP. Longuet-Higgins and his group at Edinburgh were also pursuing distributed memory models during the same period, and David Willshaw, a member of the Edinburgh group, provided some very elegant mathematical analyses of the properties of various distributed representation schemes (Willshaw, 1981). His insights provide one of the sources of the idea of coarse coding described at length in Chapter 3. Many of the contributions of Anderson, Willshaw, and others distributed modelers may be found in Hinton and Anderson (1981). Others who have made important contributions to learning in PDP models include Amari (1977a), Bienenstock, Cooper, and Munro (1982), Fukushima (1975), Kohonen (1977, 1984), and von der Malsburg (1973).

Toward the middle of the 1970s, the idea of parallel processing began to have something of a renaissance in computational circles. We have already mentioned the Marr and Poggio (1976) model of stereoscopic

depth perception. Another model from this period, the *HEARSAY* model of speech understanding, played a prominent role in the development of our thinking. Unfortunately, HEARSAY's computational architecture was too demanding for the available computational resources, and so the model was not a computational success. But its basically parallel, interactive character inspired the interactive model of reading (Rumelhart, 1977), and the interactive activation model of word recognition (McClelland & Rumelhart, 1981; Rumelhart & McClelland, 1982).

The ideas represented in the interactive activation model had other precursors as well. Morton's *logogen* model (Morton, 1969) was one of the first models to capture concretely the principle of interaction of different sources of information, and Marslen-Wilson (e.g., Marslen-Wilson & Welsh, 1978) provided important empirical demonstrations of interaction between different levels of language processing. Levin's (1976) *Proteus* model demonstrated the virtues of activation-competition mechanisms, and Glushko (1979) helped us see how conspiracies of partial activations could account for certain aspects of apparently rule-guided behavior.

Our work also owes a great deal to a number of colleagues who have been working on related ideas in recent years. Many of these colleagues appear as authors or coauthors of chapters in this book. But there are others as well. Several of these people have been very influential in the development of the ideas in this book. Feldman and Ballard (1982) laid out many of the computational principles of the PDP approach (under the name of *connectionism*), and stressed the biological implausibility of most of the prevailing computational models in artificial intelligence. Hofstadter (1979, 1985) deserves credit for stressing the existence of a subcognitive—what we call microstructural—level, and pointing out how important it can be to delve into the microstructure to gain insight. A sand dune, he has said, is not a grain of sand. Others have contributed crucial technical insights. Sutton and Barto (1981) provided an insightful analysis of the connection modification scheme we call the *delta rule* and illustrated the power of the rule to account for some of the subtler properties of classical conditioning. And Hopfield's (1982) contribution of the idea that network models can be seen as seeking minima in energy landscapes played a prominent role in the development of the Boltzmann machine (Chapter 7), and in the crystallization of the ideas presented in Chapters 7 and 14 on harmony theory and schemata.

The power of parallel distributed processing is becoming more and more apparent, and many others have recently joined in the exploration of the capabilities of these mechanisms. We hope this book represents

the nature of the enterprise we are all involved in, and that it does justice to the potential of the PDP approach.

ACKNOWLEDGMENTS

This research was supported by Contract N00014-79-C-0323, NR 667-437 with the Personnel and Training Research Programs of the Office of Naval Research, by grants from the System Development Foundation, and By a NIMH Career Development Award (MH00385) to the first author.