On *instrumentalism* and *the reality of patterns*, try D. Dennett, "Real patterns" in *Journal of Philosophy*, 88, 1991, 27–51 (the current flagship statement of the intentional stance). See also the previous flagship, "True believers: The intentional strategy and why it works" in D. Dennett, *The Intentional Stance* (Cambridge, MA: MIT Press, 1987). For some excellent discussion of these issues, and a *revealing treatment of Conway's 'Game of Life'*, see J. Bermúdez, "Rationality, mental causation and commonsense psychology," chapter 6 of his *Philosophy of Psychology: A Contemporary Introduction* (New York: Routledge, 2005). See also T. Crane "Understanding thinkers and their thoughts" (chapter 2 of *The Mechanical Mind: A philosophical introduction to minds, machines, and mental representation* (London: Penguin, 2003)

For some criticism, see L. Rudder-Baker, "Instrumental intentionality" in S. Stich and T. Warfield (eds.), *Mental Representation: A Reader* (Oxford, England: Blackwell, 1994, pp. 332–344). A clear and direct response to the "true believers" argument.

To continue the debate, see D. Dennett, "Back from the drawing board" in B. Dahlbom (ed.), *Dennett and His Critics* (Oxford, England: Blackwell, 1993, pp. 203–235). This is Dennett's response to a wide variety of critiques, all of which appear in the same volume. See especially the sections called "Labels: Am I a behaviorist? An ontologist?" (pp. 210–214), "Intentional laws and computational psychology" (pp. 217–222), and the reply to Millikan (pp. 222–227).

For a wonderful extended analysis, see B. McLaughlin and J. O'Leary-Hawthorn, "Dennett's logical behaviorism" in *Philosophical Topics*, 22, 189–259, 1994. A very thorough and useful critical appraisal of Dennett's problematic "behaviorism." See also Dennett's response in the same issue, pp. 517–522.

A difficult but rewarding engagement with the "real patterns" ideas is to be found in J. Haugeland, "Pattern and being" in B. Dahlbom (ed.), *Dennett and His Critics* (Oxford, England: Blackwell, 1993, pp. 53–69).

For a taste of something different, see R. G. Millikan, "On mentalese orthography." In B. Dahlbom (ed.), *Dennett and His Critics* (Oxford, England: Blackwell, 1993, pp. 97–123). A different kind of approach to all the issues discussed above. Not easy, but your efforts will be rewarded.

Finally, for a nice account of the *'manipulationist' approach to causal explanation*, try C. Craver, "Causal relevance and manipulation", chapter 3 of his *Explaining the Brain: Mechanisms and the Mosaic Unity of Neuroscience* (Oxford, England: Oxford University Press, 2007).

4 CONNECTIONISM

4.1 Sketches
4.2 Discussion
(A) Connectionism and Mental Causation
(B) Systematicity
(C) Biological Reality?
4.3 Suggested Readings

4.1 Sketches

The computational view of mind comes in two basic varieties[1]. The classical physical symbol system variety, already encountered in Chapter 2, stresses the role of symbolic atoms, (usually) serial processing, and expressive resources whose combinational forms closely parallel those of language and logic. The other main variety differs along all three of these dimensions and is known variously as connectionism, parallel distributed processing, and artificial neural networks.

These latter models, as the last name suggests, bear some (admittedly rather distant) relation to the architecture and workings of the biological brain. Like the brain, an artificial neural network is composed of many simple processors linked in parallel by a daunting mass of wiring and connectivity. In the brain, the "simple processors" are neurons (note the quotes: neurons are much more complex than connectionist units) and the connections are axons and synapses. In connectionist networks, the simple processors are called "units" and the connections are numerically weighted links between these units—links known, unimaginatively but with pinpoint accuracy, as *connections*.

[1]A third variety may be the 'structured probabilistic' approaches that we will meet in chapter 11

In both cases, the simple processing elements (neurons, units) are generally sensitive only to local influences. Each element takes inputs from a small group of "neighbors" and passes outputs to a small (sometimes overlapping) group of neighbors.

The differences between simple connectionist models and real neural architectures remain immense, and we will review some of them later in this chapter. Nonetheless, something of a common flavor does prevail. The essence of the common flavor lies mostly in the use of large-scale parallelism combined with local computation and in the (related) use of a means of coding known as *distributed representation*. To illustrate these ideas, consider the now-classic example of NETtalk.

NETtalk (Sejnowski and Rosenberg, 1986, 1987) is an artificial neural network, created in the mid-1980s, whose task was to take written input and turn it into coding for speech, in other words to do grapheme-to-phoneme conversion. A successful classical program, called DECtalk, was already in existence and performed the same task courtesy of a large database of rules and exceptions, hand-coded by a team of human programmers. NETtalk, by contrast, instead of being explicitly programmed, *learned* to solve the problem using a learning algorithm and a substantial corpus of example cases—actual instances of good text-to-phoneme pairings. The output of the network was then fed to a fairly standard speech synthesizer that took the phonetic coding and transformed it into real speech. During learning, the speech output could be heard to progress from initial babble to semirecognizable words and syllable structure, to (ultimately) a fair simulacrum of human speech. The network, it should be emphasized, was not intended as a model of language understanding but only of the text-to-speech transition—as such, there was no semantic database tied to the linguistic structures. Despite this lack of semantic depth, the network stands as an impressive demonstration of the power of the connectionist approach. Here, in briefest outline, is how it worked.

The system, as mentioned above, comprises a set of simple processing units. Each unit receives inputs from its neighbors (or from the world, in the case of so-called *input units*) and yields an output according to a simple mathematical function. Such functions are often nonlinear. This means that the numerical value of the output is not directly proportional to the sum of the inputs. It may be, for example, that a unit gives a proportional output for an intermediate range of total input values but gives a constant output above and below that range, or that the unit will not "fire" until the inputs sum to a certain value and thereafter will give proportional outputs. The point, in any case, is that a unit becomes activated to whatever degree (if any) the inputs from its local neighbors dictate, and that it will pass on a signal accordingly. If unit *A* sends a signal to unit *B*, the strength of the signal arriving at *B* is a joint function of the level of activation of the "sender" unit and the numerical weighting assigned to the connection linking *A* to *B*. Such weights can be positive (excitatory) or negative (inhibitory). The signals arriving at the receiving units may thus vary, being determined by the product of the numerical weight on a specific connection and the output of the "sender" unit.

NETtalk (see Box 4.1) was a fairly large network involving seven groups of input units, each group comprising some 29 individual units whose overall activation

Box 4.1

The NETtalk Architecture

The specific architecture of NETtalk (see Figure 4.1) involved three layers of units (a typical "first-generation" layout, but by no means obligatory). The first layer comprised a set of "input" units, whose task was to encode the data to be processed (information about letter sequences). The second layer consisted of a group of so-called "hidden" units whose job was to effect a partial recording of the input data. The third layer consisted of "output" units whose activation patterns determined the system's overall response to the original input. This response was specified as a vector of numerical activation values, one value for each output unit. The knowledge that the system used to guide the input–output transitions was thus encoded to a large extent in the weights on the various interunit connections. An important feature of the connectionist approach lies in the use of a variety of potent (though by no means omnipotent!) learning algorithms. These algorithms (see text and Box 4.2) adjust the weights on the interunit connections so as to gradually bring the overall performance into line with the target input–output function implicit in a body of training cases.

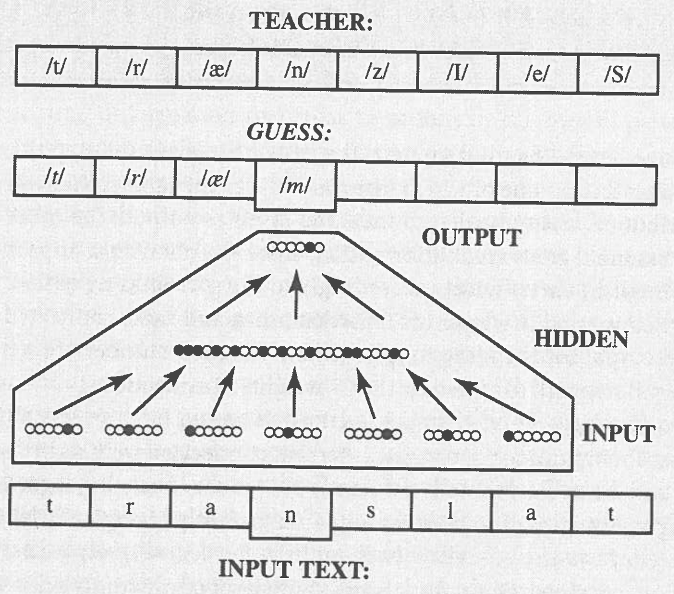

Figure 4.1 Schematic of NETtalk architecture showing only some units and connectivity. Each of 29 input units represents a letter. The 7 groups of input units were transformed by 80 hidden units. These hidden units then projected to 26 output units, which represented phonemes. There were a total of 18,629 weights in the network.

(From Sejnowski and Rosenberg, 1987, by permission)

Box 4.2

Gradient Descent Learning

The learning routine involves what is known as gradient descent (or hill climbing, since the image can be systematically inverted!). Imagine you are standing somewhere on the inner slopes of a giant pudding basin. Your task is to find the bottom—the point corresponding to the lowest error and hence the best available solution. But you are blindfolded and cannot see the bottom and cannot run directly to it. Instead, you take a single step and determine whether you went up or down. If you went up (a local error), you go back and try again in the opposite direction. If you went down, you stay where you are. By repeating this procedure of small steps and local feedback, you slowly snake toward the bottom and there you halt (since no further step can take you any lower). The local feedback, in the case of the neural network, is provided by the supervisory system that determines whether a slight increase or decrease in given weight would improve performance (assuming the other weights remain fixed). This procedure, repeated weight by weight and layer by layer, effectively pushes the system down a slope of decreasing error. If the landscape is a nice pudding-basin shape with no nasty trenches or gorges, the point at which no further change can yield a lower error signal will correspond to a good solution to the problem.

specified one letter. The total input to the system at each time step thus specified seven distinct letters, one of which (the fourth) was the target letter whose phonemic contribution was to be determined and given as output. The other six letters provided essential contextual information, since the phonemic impact of a given letter (in English) varies widely accordingly to the surrounding letters. The input units were connected to a layer of 80 hidden units, and these connected in turn to a set of 26 output units coding for phonemes. The total number of interunit links in the overall network summed to 18,829 weighted connections.

Given this large number of connections, it would be impractical (to say the least) to set about finding appropriate interunit connection weights by hand coding and trial and error! Fortunately, automatic procedures (learning algorithms) exist for tuning the weights. The most famous (but probably biologically least realistic) such procedure is the so-called *back-propagation learning* algorithm. In back-propagation learning, the network begins with a set of randomly selected connection weights (the layout, number of units, etc., being fixed by the designer). This network is then exposed to a large number of input patterns. For each input pattern, some (initially incorrect) output is produced. An automatic supervisory system monitors the output, compares it to the target (correct) output, and calculates small adjustments to the connection weights—adjustments that would cause slightly improved performance were the network to be reexposed to the very same input pattern. This procedure (see Box 4.2) is repeated again and again for a large (and

cycling) corpus of training cases. After sufficient such training, the network often (though not always) learns an assignment of weights that effectively solves the problem—one that reduces the error signal and yields the desired input–output profile.

Such learning algorithms can discover solutions that we had not imagined. Researcher bias is thus somewhat decreased. Moreover, and perhaps more importantly, the way the trained-up network *encodes* the problem-solving information is quite unlike the more traditional forms of symbol-string encoding characteristic of the work discussed in Chapter 2. The connectionist system's long-term knowledge base does not consist in a body of declarative statements written out in a formal notation based on the structure of language or logic. Instead, the knowledge inheres in the set of connection weights and the unit architecture. Many of these weighted connections participate in a large number of the system's problem-solving activities. Occurrent knowledge—the information active during the processing of a specific input—may usefully be equated with the transient activation patterns occurring in the hidden unit layer. Such patterns often involve *distributed* and *superpositional* coding schemes. These are powerful features, so let's pause to unpack the jargon.

An item of information is here said to have a distributed representation if it is expressed by the simultaneous activity of a number of units. But what makes distributed representation computationally potent is not this simple fact alone, but the systematic use of the distributions to encode further information concerning subtle similarities and differences. A distributed pattern of activity can encode "microstructural" information such that variations in the overall pattern reflect variations in the content. For example, a certain pattern might represent the presence of a black cat in the visual field, whereas small variations in the pattern may carry information about the cat's orientation (facing ahead, side-on, etc.). Similarly, the activation pattern for a black panther may share some of the substructure of the cat activation pattern, whereas that for a white fox may share none. The notion of superpositional storage is precisely the notion of such partially overlapping use of distributed resources, in which the overlap is informationally significant in the kinds of ways just outlined. The upshot is that semantically related items are represented by syntactically related (partially overlapping) patterns of activation. The public-language words *cat* and *panther* display no such overlap (though *phrases* such as "black panther" and "black cat" do). Distributed superpositional coding may thus be thought of as a trick for forcing still more information into a system of encodings by exploiting even more highly structured syntactic vehicles than words. This trick yields a number of additional benefits, including economical use of representational resources, "free" generalization, and graceful degradation. Generalization occurs because a new input pattern, if it resembles an old one in some aspects, will yield a response rooted in that partial overlap. "Sensible" responses to new inputs are thus possible. "Graceful degradation," alluring as it sounds, is just the ability to produce sensible responses given some systemic damage. This is possible because the overall system now acts as a kind of pattern completer—given a large enough fragment of a familiar pattern, it will recall the whole thing. Generalization, pattern completion, and damage tolerance are thus all reflections of the same powerful computational strategy: the use of distributed, superpositional storage schemes and partial cue-based recall.

Two further properties of such coding schemes demand our attention. The first is the capacity to develop and exploit what Paul Smolensky (1988) has termed *dimension shifted* representations. The second is the capacity to display fine-grained context sensitivity. Both properties are implied by the popular but opaque gloss on connectionism that depicts it as a *subsymbolic paradigm*. The essential idea is that whereas basic physical symbol system approaches displayed a kind of semantic transparency (see Chapter 2), such that familiar words and ideas were rendered as simple inner symbols, connectionist approaches introduced a much greater distance between daily talk and the contents manipulated by the computational system. By describing connectionist representation schemes as dimension-shifted and subsymbolic, Smolensky (and others) means to suggest that the features that the system uncovers are finer grained and more subtle than those picked out by single words in public language. The claim is that the contentful elements in a subsymbolic program do not directly recapitulate the concepts we use "to consciously conceptualize the task domain" (Smolensky, 1988, p. 5) and that "the units do not have the same semantics as words of natural language" (p. 6). The activation of a given unit (in a given context) thus signals a semantic fact: but it may be a fact that defies easy description using the words and phrases of daily language. The semantic structure represented by a large pattern of unit activity may be very rich and subtle indeed, and minor differences in such patterns may mark equally subtle differences in contextual nuance. Unit-level activation differences may, thus, reflect minute details of the visual, tactile, functional, or even emotive dimensions of our responses to the same stimuli in varying real-world contexts. The pioneer connectionists McClelland and Kawamoto (1986) once described this capacity to represent "a huge palette of shades of meaning" as being "perhaps . . . the paramount reason why the distributed approach appeals to us" (p. 314).

This capacity to discover and exploit rich, subtle, and nonobvious schemes of distributed representation raises an immediate methodological difficulty: how to achieve, after training, some understanding of the knowledge and strategies that the network is actually using to drive its behavior? One clue, obviously, lies in the training data. But networks do not simply learn to repeat the training corpus. Instead they learn (as we saw) general strategies that enable them to group the training instances into property-sharing sets, to generalize to new and unseen cases, and so forth. Some kind of knowledge organization is thus at work. Yet it is impossible (for a net of any size or complexity) to simply read off this organization by, for example, inspecting a trace of all the connection weights. All you see is numerical spaghetti!

The solution to this problem of "posttraining analysis" lies in the use of a variety of tools and techniques, including statistical analysis and systematic interference. Systematic interference involves the deliberate damaging or destruction of groups of units, sets of weights, or interunit connections. Observation of the network's "post-lesion" behavior can then provide useful clues to its normal operating

strategies. It can also provide a further dimension (in addition to brute performance) along which to assess the "psychological reality" of a model, by comparing the way the network reacts to damage to the behavior patterns seen in humans suffering from various forms of local brain damage and abnormality (see, e.g., Patterson, Seidenberg, and McClelland, 1989; Hinton and Shallice, 1989). In practice, however, the most revealing forms of posttraining analysis have involved not artificial lesion studies but the use of statistical tools (see Box 4.3) to generate a picture of the way the network has learned to negotiate the problem space.

Box 4.3

Cluster Analysis

Cluster analysis is an example of an analytic technique addressing the crucial question, "what kinds of representation has the network acquired?" A typical three-layer network, such as NETtalk, uses the hidden unit layer to partition the inputs so as to compress and dilate the input representation space in ways suited to the particular target function implied by the training data. Thus, in text-to-phoneme conversion, we want the rather different written inputs "sale" and "sail" to yield the same phonetic output. The hidden units should thus compress these two input patterns into some common intermediate form. Inputs such as "shape" and "sail" should receive different, though not unrelated, codings, whereas "pint" and "hint," despite substantial written overlap, involve widely variant phonemic response and should be dilated—pulled further apart. To perform these tricks of pulling together and pushing apart, NETtalk developed 79 different patterns of hidden unit activity. Cluster analysis then involved taking each such pattern and matching it with its nearest neighbor (e.g., the four-unit activation pattern 1010 is nearer to 1110 than to 0101, since the second differs from the first in only one place whereas the third differs in four). The most closely matched pairs are then rendered (by a process of vector averaging) as new single patterns, and the comparison process is repeated. The procedure continues until the final two clusters are generated, representing the grossest division of the hidden unit space learned by the system. The result is an unlabeled, hierarchical tree of hidden unit activity. The next task is to label the nodes. This is done as follows. For each of the original 79 activation patterns, the analyst retains a trace of the input pattern that prompted that specific response. She then looks at the pairs (or pairs of pairs, etc.) of inputs that the network "chose" to associate with these similar hidden unit activation patterns so as to discern what those inputs had in common that made it *useful* for the network to group them together. The result, in the case of NETtalk, is a branching hierarchical tree (see Figure 4.2) whose grossest division is into the familiar

(continued)

groupings of vowels and consonants and whose subdivisions include groupings of different ways of sounding certain input letters such as *i*, *o*, and so on. In fact, nearly all the phonetic groupings learned by NETtalk turned out to correspond closely to divisions in existing phonetic theory. One further feature discussed in Section 4.2 is that various versions of NETtalk (maintaining the same architecture and learning routine and training data but beginning with different assignments of random weights) exhibited, after training, very different sets of interunit weightings. Yet these superficially different solutions yield almost identical cluster-analytic profiles, Such nets use different numerical schemes to encode what is essentially the *same* solution to the text-to-phoneme conversion problem.

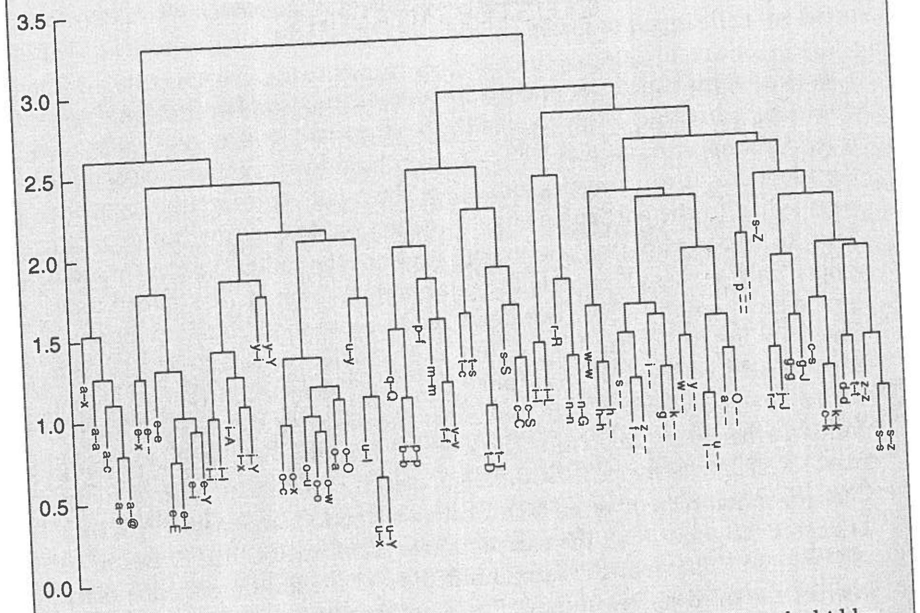

Figure 4.2 Hierarchical cluster analysis of the average activity levels on the hidden units for each letter-to-sound correspondence (*l–p* for letter l and phoneme *p*). The closest branches correspond to the most nearly similar activation vectors of the hidden units.

(From Sejnowski and Rosenberg, 1987. Parallel networks that learn to pronounce English text. *Complex Systems*, 1, 145–168, Reproduced by kind permission of T. Sejnowski.)

So far, then, we have concentrated our attention on what might be termed "first-generation" connectionist networks. It would be misleading to conclude, however, without offering at least a rough sketch of the shape of more recent developments and the gradual emergence of what may be an important new class of approaches (more on this in Chapter 11) that share key features of both connectionist and classical models.

Second-generation connectionism was marked by an increasing emphasis on temporal structure. First-generation networks, it is fair to say, displayed no real capacity to deal with time or order. Inputs that designated an ordered sequence of letters had to be rendered using special coding schemes that artificially disambiguated the various possible orderings. Nor were such networks geared to the production of output patterns extended in time (e.g., the sequence of commands needed to produce a running motion)[1] or to the recognition of temporally extended patterns, such as the sequences of facial motion that distinguish a wry smile from a grimace. Instead, the networks displayed a kind of "snapshot reasoning" in which a frozen temporal instant (e.g., coding for a picture of a smiling person) yields a single output response (e.g., a judgment that the person is happy). Such networks could not identify an instance of pleasant surprise by perceiving the gradual transformation of puzzlement into pleasure (see, e.g., Churchland, 1995).

To deal with such temporally extended data and phenomena, second-generation connectionist researchers have deployed so-called *recurrent neural networks*. These networks share much of the structure of a simple three-layer "snapshot" network, but incorporate an additional feedback loop. This loop (see Figure 4.3) recycles some aspect of the network's activity at time $t_1$alongside the new inputs arriving at time t_2. Elman nets (see Elman, 1991b) recycle the hidden unit activation pattern from the previous time slice, whereas Jordan (1986) describes a net that recycles its previous output pattern. Either way, what is preserved is some kind of ongoing trace of the network's last activity. Such traces act as a kind of short-term memory, enabling the network to generate new responses that depend both on the current input and on the previous activity of the network. Such a set-up also allows output activity to continue in the complete absence of new inputs, since the network can continue to recycle its previous states and respond to them.

For example, Elman (1991b) describes a simple recurrent network whose goal is to categorize words according to lexical role (noun, verb, etc.). The network was exposed to grammatically proper sequences of words (such as "the boy broke the window"). Its immediate task was to predict the next word in the ongoing sequence. Such a task, it should be clear, has no unique solution insofar as many continuations will be perfectly acceptable grammatically. Nonetheless, there are whole classes of words that cannot be allowed to follow. For example, the input sequence "the boy who" cannot be followed by "cat" or "tree." These constraints on acceptable successor words reflect grammatical role, and the training regime thus provides data germane to the larger goal of learning about lexical categories.

Elman's network proved fairly adept at its task. It "discovered" categories such as verb and noun and also evolved groupings for animate and inanimate objects, foods, and breakable objects—properties that were good clues to grammatical role in the training corpus used. To determine exactly what the network learned, Elman used another kind of posttraining analysis (one better suited to the special

[1]These issues are usefully discussed in Churchland and Sejnowski (1992, pp. 119–120). For a more radical discussion, see Port, Cummins, and McCauley (1995).

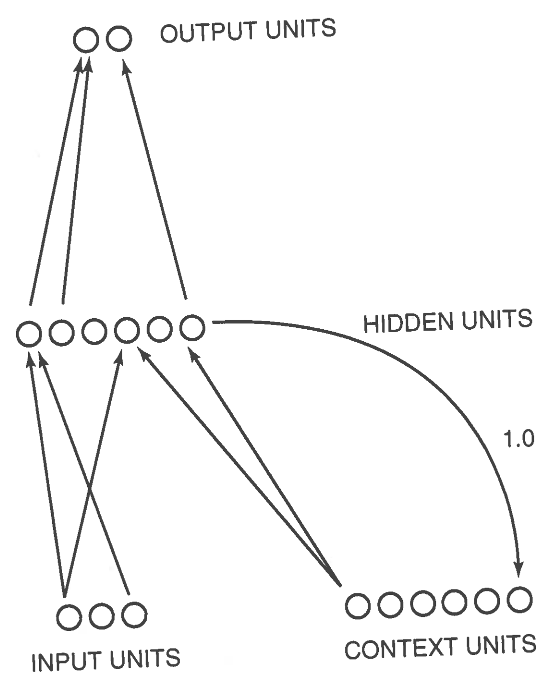

Figure 4.3 A three-layer recurrent network. The context units are activated, one by one, by the corresponding hidden units. For simplicity, not all the activation is shown. (After Elman, 1991b, with permission.)

case of recurrent nets) called *principal component analysis* (PCA). The details are given in Clark (1993, pp. 60–67) and need not detain us here. It is worth noting, however, that whereas cluster analysis can make it seem as if a network has merely learned a set of static distributed symbols and is thus little more than a novel implementation of the classical approach, principal component analysis reveals the role of even deeper dynamics in promoting successful behavior. The key idea is that whereas cluster analysis stresses relations of similarity and difference between static states ("snapshots"), PCA reflects, in addition, the ways in which being in one state (in a recurrent network) can promote or impede movement into future states. Standard cluster analysis would not reveal these constraints on processing trajectories. Yet the grammatical knowledge acquired by the recurrent net inheres quite profoundly in such temporally rich information-processing detail.[2]

The more such temporal dynamics matter, the farther we move (I contend) from the guiding image of the basic physical symbol system hypothesis. For at the heart of that image lies the notion of essentially static symbol structures that retain stable meanings while being manipulated by some kind of central processor. Such a picture, however, does not usefully describe the operation of even the simple

[2]See Elman (1991b, p. 106).

recurrent networks previously discussed. For the hidden unit activation patterns (the nearest analogue to static symbols) do not function as fixed representations of word-role. This is because each such pattern reflects something of the prior context,[3] so that, in a sense, "every occurrence of a lexical item has a separate internal representation" (Elman, 1991b, p. 353). Elman's model thus uses so-called *dynamic representations*. Unlike the classical image in which the linguistic agent, on hearing a word, retrieves a kind of general-purpose lexical representation, Elman is suggesting a dynamic picture in which

> There is no separate stage of lexical retrieval. There are no representations of words in isolation. The representations of words (the internal states following input of a word) always reflect the input taken together with the prior state . . . the representations are not propositional and their information content changes constantly over time . . . words serve as guideposts which help establish mental states that support (desired) behavior. (Elman, 1991b, p. 378)

Elman thus invites us to see beyond the classical image of static symbols that persist as stored syntactic items and that are "retrieved" and "manipulated" during processing. Instead, we confront an image of a fluid inner economy in which representations are constructed on the spot and in light of the prevailing context, and in which much of the information-processing power resides in the way current states constrain the future temporal unfolding of the system.

Work in third-generation connectionism (and related approaches that might best be dubbed simply *artificial neural networks*) continues this flight from the (static) inner symbol by laying even greater stress on a much wider range of dynamic and time-involving properties. For this reason it is sometimes known as *dynamical connectionism*. Dynamical connectionism (see Wheeler, 1994, p. 38; Port and van Gelder, 1995, pp. 32–34) introduces a number of new and more neurobiologically realistic features to the basic units and weights paradigm, including special-purpose units (units whose activation function is tailored to a task or domain), more complex connectivity (multiple recurrent pathways and special-purpose wiring), computationally salient time delays in the processing cycles, continuous-time processing, analog signaling, and the deliberate use of noise. Artificial neural networks exhibiting such nonstandard features support "far richer intrinsic dynamics than those produced by mainstream connectionist systems" (Wheeler, 1994, p. 38). We shall have more to say about the potential role of such richer and temporally loaded dynamics in future chapters. For the moment, it will suffice to note that second- and third-generation connectionist research is becoming progressively more dynamic: it is paying more heed to the temporal dimension and it is exploiting a wider variety of types of units and connectivity. In so doing, it is moving ever farther from the old notion of intelligence as the manipulation of static, atemporal, spatially localizable inner symbols.

[3]Even the first word in a sentence incorporates a kind of "null" context that is reflected in the network state.

Finally, it is worth flagging a somewhat different, yet in many ways deeply related, class of approaches. The origins of these approaches go back to work on the aptly named (to see why, see Chapter 11) "Helmholz Machine" (Dayan et al., 1995; Dayan and Hinton, 1996; see also Hinton and Zemel, 1994). At their heart lies the use of prediction-driven learning (the power of one form of which we saw in the discussion of Elman, 1991b, above) within a multilayered structure rich in both "top-down" and "bottom-up" connections. The task of such a network, in learning, is to find out how to use the downward connections to generate (for itself, from the top down) the sensory data as it manifests at each level. Such networks do not need to be "supervised" by being shown labeled (i.e., preclassified by the experimenter) examples of some desired input–output mapping. Applied in the special context of a multilevel (hierarchical) architecture, such approaches begin to address the question of how to learn to represent and process complex nested structures of the kind we find in language, vision, motor control, and many other cognitive domains (see Hinton, 2007 and Box 4.4). We pick up this important thread in Chapter 11.

The connectionist movement and its various successors remains, it is fair to conclude, the leading expression of a certain kind of "inner symbol flight." Not, it seems to me, flight from the very idea of an inner symbol, where this can be quite liberally construed (see 6.2 [A] and 7.2 [C] following) but flight from the idea

Box 4.4

Dealing with Structure

We have already met one of the key puzzles to which these approaches may offer a solution. It is the need, as expressed by Hinton (1990, p. 47) to represent and process "complex, articulated structures" such as part–whole hierarchies: structures in which elements form wholes that can themselves be elements of one or more larger wholes (for some examples see the After-Sketch to Chapter 2). This is the very same need, it is worth noticing, that drove much early skepticism about the connectionist alternative to the use of classical, sentence-like internal representational forms (see discussion in 4.2 [B] below). But jump to the year 2007 and we find Hinton, a theorist not given to overstatement, writing that:

"The limitations of backpropagation learning can now be overcome by using multilayer neural networks that contain top-down connections and training them to generate sensory data rather than to classify it."

(Hinton, 2007a, p. 428)

Such approaches come in many varieties, one of which we shall meet in Chapter 11.

of static, chunky, user-friendly, semantically transparent (see Chapter 2) inner symbols. Those kinds of symbols, at least in simulations of motor control and perceptual processing, are now something of an endangered species. They are being replaced by subtler, often highly distributed and increasingly dynamic (time-involving) inner states. This is, I believe, a basically laudable transition. Connectionist models profit from (increasing) contact with real neuroscientific theorizing. And they exhibit a profile of strengths (motor control, pattern recognition) and weaknesses (planning and sequential logical derivation) that seems reassuringly familiar and evolutionarily plausible. They look to avoid, in large measure, the uncomfortable back-projection of our experiences with text and words onto the more basic biological canvas of the brain. But the new landscape brings new challenges, problems, and uncertainties. Time to meet the bugbears.

4.2 Discussion

(A) CONNECTIONISM AND MENTAL CAUSATION

Connectionism, according to some philosophers, offers a rather concrete challenge to the folk psychological image of mind. The leading idea, once again, is that folk psychology is committed to the causal efficacy of the mental states named in ordinary discourse, and that there is now a tension between such imagined causal efficacy and the specific connectionist vision of inner processing and storage.

The key move in this argument is the insistence that the folk framework is indeed committed to a strong and direct notion of causal efficacy. In this vein, Ramsey, Stich, and Garon (1991b) insisted that the commonsense understanding of mind involves a crucial commitment to what they term "propositional modularity." This is the claim that the folk use of propositional attitude talk (talk of Andy's believing that the wine is chilled and so on) implies a commitment to "*functionally discrete, semantically interpretable* states that play a *causal role* in the production of other propositional attitudes and ultimately in the production of behavior" (Ramsey, Stich, and Garon, 1991, p. 204, original emphasis). Ramsey, Stich, and Garon argue that distributed connectionist processing does not support such propositional modularity and hence that if human minds work like such devices, then the folk vision is fundamentally inaccurate.

Why suppose that the folk are committed to propositional modularity anyway? The evidence is in part anecdotal (we do talk of people gaining or losing beliefs one at a time, and in that sense we seem to depict the beliefs, etc., as discrete items—Ramsey, Stich, and Garon, 1991, p. 205) and in part substantive. The substantive evidence is that the very usefulness of the folk framework seems to depend on our being able to cite specific beliefs as explanatory of specific actions. Andy may believe that the cat wants feeding, that Rome is pretty, and that the wine is chilled, but we reserve the right to explain his going into the kitchen as a direct result of his belief about the wine. The cat belief, though real and capable of prompting the same behavior, may be imagined to be inactive at that moment.