COGNITIVE PSYCHOLOGY **37,** 243–282 (1998)
ARTICLE NO. CG980694

Rethinking Eliminative Connectionism

Gary F. Marcus

New York University

Humans routinely generalize universal relationships to unfamiliar instances. If we are told ''if glork then frum,'' and ''glork,'' we can infer ''frum''; any name that serves as the subject of a sentence can appear as the object of a sentence. These universals are pervasive in language and reasoning. One account of how they are generalized holds that humans possess mechanisms that manipulate symbols and variables; an alternative account holds that symbol-manipulation can be eliminated from scientific theories in favor of descriptions couched in terms of networks of interconnected nodes. Can these ''eliminative'' connectionist models offer a genuine alternative? This article shows that eliminative connectionist models cannot account for how we extend universals to arbitrary items. The argument runs as follows. First, if these models, as currently conceived, were to extend universals to arbitrary instances, they would have to generalize outside the space of training examples. Next, it is shown that the class of eliminative connectionist models that is currently popular cannot learn to extend universals outside the training space. This limitation might be avoided through the use of an architecture that implements symbol manipulation.

1. INTRODUCTION

Humans routinely generalize universal relationships to unfamiliar instances. If we are told ''if glork then frum,'' and ''glork,'' we can infer

For comments on earlier drafts, I thank Dan Anderson, Neil Berthier, Ned Block, Richard Bogartz, Michael Brent, Mike Casey, Noam Chomsky, Chuck Clifton, Zoubin Ghahramani, Graeme Halford, John Hummel, Jay McClelland, Denis Mareschal, Randy O'Reilly, Steve Pinker, Zenon Pylyshyn, Erik Reichle, Ed Stein, Neil Stillings, Whitney Tabor, Zsofia Zvolenszky, and several anonymous reviewers. For helpful discussion, I thank Andy Barto, Bob Berwick, Susan Carey, Gary Dell, Dan Dennett, Jerry Fodor, Lee Giles, Keith Holyoak, Ray Jackendoff, Robbie Jacobs, Art Markman, Mike McCloskey, Elissa Newport, Neal Pearlmutter, Steven Phillips, Terry Regier, Mike Tanenhaus, David Touretzky, and audiences at Birkbeck College, Johns Hopkins University, NYU, Oxford University, Rutgers University, UCLA, University of Essex, University of Massachusetts, University of Rochester, the April 1996 Edinburgh Conference on Evolution and Language, and the November 1996 Annual Meeting of the Psychonomics Society. This research was partially supported by a Faculty Research Grant from the University of Massachusetts. Address correspondence and reprint requests to Gary Marcus, Department of Psychology, New York University, 6 Washington Place, New York, NY 10003. E-mail: gary.marcus@nyu.edu.

0010-0285/98 $25.00

''frum'' (Smith, Langston, & Nisbett, 1992). If all gronks are bleems, and all bleems are blickets, we can infer that all gronks are blickets. If we hear a new name, *Dweezil,* used as the subject of a sentence, we can automatically use it as the object of another sentence (Chomsky, 1957). If *blicket* is the stem of a verb, *blicketing* is the progressive form of that verb (Prasada & Pinker, 1993). As Fodor and Pylyshyn (1988, p. 3) put it, ''the ability to entertain a given thought implies the ability to entertain thoughts with semantically related contents''; that is, the mind is systematic in its ability to generalize abstract relationships.

1.1. Accounting for Universals

Accounting for how the mind extends these universals is an important project for cognitive science. One popular view, advocated by Fodor (1975) and Newell (1980), assumes that universals are extended through the action of symbol-manipulating machinery. Advocates of symbol-manipulation suppose that there are mentally represented *rules* that describe relationships between *variables,* that those variables may be instantiated with particular *instances,* and that there are *operations* such as copying and concatenation that perform computations on variables. For instance, the process of forming the regular past tense of an English verb might involve a mechanism that instantiates the variable **verb stem** with an instance, say *fax,* and an operation that combines that instance with the *-ed* morpheme, yielding *faxed* (e.g., Marcus et al., 1995).

While the view that the mind manipulates variables is widespread in linguistics and artificial intelligence, it is not a view that is uniformly accepted. For example, Plaut, McClelland, Seidenberg, and Patterson (1996) wrote that

> A rule-based approach has considerable intuitive appeal [but. . .] An alternative comes out of research on connectionist or parallel distributed processing networks, in which computation takes the form of cooperative and competitive interactions among large numbers of simple, neuron-like processing units. (p. 56)

This alternative view, which Pinker and Prince (1988) called *eliminative connectionism,*[1] has its roots in work such as the connectionist model of Rumelhart and McClelland (1986) describing how children learn to inflect (English) verbs for the past tense. Their aim was to provide

> a distinct alternative to the view that children learn the rule of English past-tense formation in any explicit sense [. . . by showing] that a reasonable account of the acquisition of past tense can be provided without recourse, if according to style, to the notion of a 'rule' as anything more than a *description* of the language.

[1] Some eliminative connectionist researchers acknowledge that rules play some rule in cognition, but suggest that rules are restricted to ''conscious rule use'' or ''deliberate, serial reasoning.''

Although eliminative connectionist models differ in their details, they share a common design philosophy (sometimes ascribed to the framework of Parallel Distributed Processing) that is probably familiar to most readers. The task of each model is to learn a mapping from an input vector (a set of nodes with activation values) to an output vector (a second set of nodes with activation values), on the basis of a set of training examples, with feedback provided by an external teacher.[2] Input and output nodes typically encode features, such as the presence or absence of a word, a sound, or an object. Input and output encoding schemes either can be *localist* or *distributed.* In localist representations, each input node corresponds to a specific word or concept, and only one input node is activated at a given time. In distributed representations, the inputs are encoded by sets of nodes, with each input node corresponding to a feature; an individual entity corresponds to a set of simultaneously activated features that typically represent subcomponents such as phonological or semantic units. For example, in the past tense model of Hare, Elman, and Daugherty (1995), the word *bid* would be represented by the simultaneous activation of three nodes, the nodes corresponding to *b* in the onset position, *i* in the nucleus position, and *d* in the coda position.

Two kinds of network architectures that are commonly used in arguments for eliminative connectionism are feedforward networks (e.g., Rumelhart, Hinton, & Williams, 1986b) and simple recurrent networks (Elman, 1990). *Feedforward networks* (see Fig. 1) are networks that contain a layer of input units, zero or more layers of hidden units (i.e., units that are neither input nor output units), and a layer of output units; the term *feedforward* indicates the fact that activation percolates in only one direction. The weights in these models are usually initially set to random values and then adjusted through the application of an error-correction algorithm, such as back-propagation (Rumelhart et al., 1986b), that computes the difference between the actual output and some target output.

Simple recurrent networks (see Fig. 2) resemble feedforward networks, but are enhanced with a ''recurrent'' layer of context units. These context units receive input from the hidden units and feed back into the hidden units, allowing simple recurrent networks to keep track of temporal sequences. In both types of models, the response to a novel item depends in part on the similarity of that input to those input–output pairs on which the model has been trained.

1.2. Preliminary Evidence for Eliminative Connectionism

Although there have been some ''a priori'' arguments for eliminative connectionism, none are convincing. For example, some investigators have sug-

[2] There are other types of connectionist models such as self-organizing maps (Kohonen, 1984), and reinforcement learning models in which the model is given some input and a feedback signal but not detailed information about the intended output; but since these have not been used to account for the generalization of universals, I will not discuss them further.

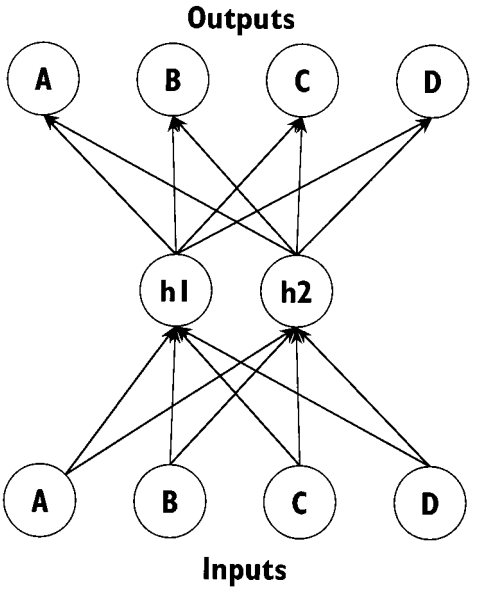

FIG. 1. Feedforward network. Input units are drawn at the bottom; output units are drawn at the top. Connections between nodes are drawn as arrows.

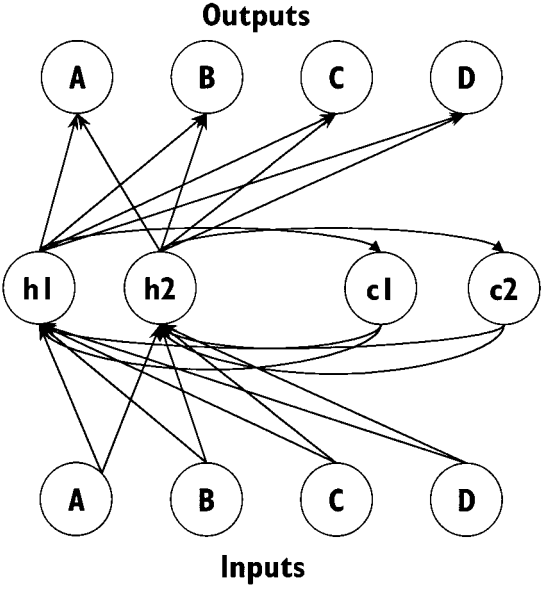

FIG. 2. Simple recurrent network. The context layer consists of the units marked c1 and c2. Connections from h1 to c1 and from h2 to c2 are fixed at 1.0; all other weights are modifiable. See text for further details.

gested that symbolic models are incompatible with learning and cannot represent quantitative information, but in fact canonically symbolic architectures such as ACT-R (Anderson, 1983) and SOAR (Newell, 1990) routinely do both. Others have suggested that eliminative connectionist models are more biologically plausible than symbolic models, but the back-propagation algorithm that eliminative connectionist models typically rely on is clearly not biologically plausible (Crick & Asunama, 1986; Smolensky, 1988).

Others have emphasized the ability of connectionist networks to approximate any function, but while it is true that in principle certain classes of networks are universal function approximators, such proofs may have little relevance to psychology. Such proofs do not show that a particular network with fixed resources (say a three-layer network with 10 input nodes, 10 hidden units, and 10 output nodes) can approximate any given function.[3] Instead, these proofs merely establish that for any given function, *some* network with some set of weights and connections can approximate that function. Furthermore, these proofs only pertain to what networks can represent, not what they can learn.

Still others have argued that eliminative connectionist models are more parsimonious than symbolic models, but these models have many free parameters, including the number of nodes, the ways in which those nodes are interconnected, and the learning algorithm. Hence it is hardly clear that they provide a more parsimonious account (McCloskey, 1991). Moreover, especially because biological systems are clearly complex, constraining ourselves a priori to just a few mechanisms is probably not wise. As Francis Crick (1988, p. 138) put it, ‘‘While Occam’s razor is a useful tool in physics, it can be a very dangerous implement in biology.’’

Because such prior considerations do not militate in favor of (or against) eliminative connectionism, we must turn to other sorts of evidence, especially to detailed consideration of eliminative connectionist models. Advocates and critics of eliminative connectionism agree that it is crucial to determine precisely what sorts of problems lie within the scope of eliminative connectionist models. This paper is a first step in that direction.

1.3. Preview

Any problem-solver that cannot entertain an infinite number of possibilities simultaneously must order some hypotheses before others (for a related point, see Goodman, 1955). The learner’s tendency to choose some hypotheses over others is sometimes referred to as the learner’s ‘‘hypothesis space

[3] The cascade-correlation algorithm (Fahlman & Lebiere, 1990) is an algorithm which dynamically adds hidden nodes as necessary. In the limit, given unbounded resources, a network using that algorithm is guaranteed to model any function (within a restricted but broad class), but there is no guarantee that such a network can find an adequate solution given a plausible number of training examples or a plausible number of hidden units.

bias''; no learner is entirely free of such a bias. For example, consider the input–output pairs [2,2], [4,4], [6,6], and [8,8]. What output would correspond to the input 7? This is a matter of induction, not deduction; in principle, any output is possible. A human would make the induction that the output *7* is most plausible (perhaps corresponding to the function $f(x) = x$), but a truly general problem solver would also have to be capable of learning functions such as *if* x *is even then* $f(x) = x$, *otherwise* $f(x) = (x - 1)$, in which case the output corresponding to the input *7* would be *6*.

Eliminative connectionist models could only provide plausible accounts of human cognition if the inductions that the models made matched the inductions humans made. Earlier research by connectionists working outside the eliminative connectionist approach suggests that (at least in some domains) unconstrained connectionist networks are not likely to draw humanlike generalizations. For example, Denker et al. (1987, p. 877) wrote that

> Since antiquity, man has dreamed of building a device that would ''learn from examples'', ''form generalizations'', and ''discover the rules'' behind patterns in the data. Recent work has shown that a highly connected, layered network of simple analog processing elements can be astonishingly successful at this, in some cases. [But] . . . the symmetric, low-order, local solutions that humans seem to prefer are not the ones that the network chooses from the vast number of solutions available; indeed, the generalized delta method [used in most eliminative connectionist models] and similar learning procedures do not usually hold the ''human'' solutions stable against perturbations.

Concern about whether eliminative connectionist networks can adequately generalize has been central to most critiques of eliminative connectionism, including Fodor and Pylsyhyn (1988), Hadley (1994), Marcus et al. (1995), Prasada and Pinker (1993), and Pinker and Prince (1988). In a suggestion that this article will concur with, Pinker and Prince (p. 176) argued that an important limit on eliminative connectionist models is that they

> . . . do not easily provide *variables* that stand for sets of individuals regardless of their featural decomposition, and over which quantified generalizations can be made.

Similar arguments have been made within less radical quarters of the connectionist community (Barnden, 1984, 1992; Dyer, 1995; Shastri & Ajjanagadde, 1993; Sun, 1992; Touretzky, 1991; Touretzky & Hinton, 1985). For instance, Touretzky (1991, p. 21) argued that

> The problem with pattern transformers is that they require an unreasonable amount of training in order to generalize correctly. When the transformation function to be induced involves very high-order predicates [in the (Minsky & Papert, 1969) *Perceptrons* sense], so that inputs nearby in Hamming space do not necessarily result in nearby outputs, the training set must include almost every possible input/output pair.

Simulations by Geman, Bienstock, and Doursat (1992), Pavel, Gluck, and Henkle (1988), Pazzani and Dyer (1987), Prasada and Pinker (1993), De-

Losh, Busemeyer, and McDaniel (1997), and Busemeyer, McDaniel, and Byun (1997) are consistent with the suggestion of Denker and others that eliminative connectionist networks often face difficulty in drawing human-like generalizations.

A principal goal of the current paper is to characterize more precisely one class of problems that pose special difficulties for contemporary eliminative connectionist networks. In particular, this paper extends previous findings about the ability of networks to generalize by proving that a large class of models does not generalize a large class of universals in the ways that humans do. Importantly, this argument will be about learning, not representation; thus, although the class of models that I describe will be able to *represent* these universals, they will not be able to *learn* them.[4] The argument that will be presented here thus takes a different form from the argument of Minsky and Papert (1969) that two-layer perceptrons could not even represent functions like exclusive-or.

1.4. Road Map

The remainder of the paper is structured as follows. First, as preliminaries, I briefly sketch the framework of symbol-manipulation as it applies to variable manipulation (Section 2; for a longer discussion, see Marcus, 1999), and then discuss the architecture of contemporary eliminative connectionist models (Section 3).

Section 4 defines the notion of a *training space* and shows that to generalize universals to untrained items, contemporary eliminative connectionist networks would need to generalize outside the training space. Section 5 shows that neither feedforward networks nor simple recurrent networks can generalize outside the training space. Section 6 considers the role of experience, Section 7 considers alternative connectionist models, and Section 8 concludes.

[4] Based on an earlier version of this manuscript, Holyoak and Hummel (in press) argue that the functions that I describe could not even be represented in an eliminative connectionist network. Their argument is that the eliminative connectionist network could only represent the function with respect to some finite set of entities. The difficulty with this argument is that the same argument could be applied to any finite implementation of any algorithm. For example, a Turing machine with a finite tape can extend the identity function only to a finite number of inputs; likewise, a computer program that can only represent numbers up to 2^{64} cannot extend identity to $2^{64} + 1$. As such, these sorts of limitations of finiteness cannot choose between plausible models, symbolic or not, and must apply to humans as well, since humans have finite resources. (Still, I would agree with Holyoak and Hummel that there is an important difference between representing a function by providing a complete list of possible input features and their corresponding output features and representing a function more parsimoniously via a placeholder, as in a symbol-manipulating system.)

2. SYMBOL MANIPULATION

The extension of universals is straightforward in a system that permits operations over symbolic variables. *Symbols* are mental encodings of equivalence classes (Abler, 1989; Newell & Simon, 1975; Pylyshyn, 1984, 1986; Vera & Simon, 1994), which is to say that all members encoded via a particular symbol are treated equally by some higher level operation (Pylyshyn, 1984, 1986). Symbols can encode either individuals (e.g., *Donald Duck*) or categories (e.g., **duck** or **cartoon character**), and can encode either atomic elements (e.g. *Superman*) or complex combinations (e.g., *the tattered but well-read Mark Twain novels*).

Fundamental to the view of symbol-manipulation is a distinction between *instances* (*tokens*) and *classes* (*types*) (Fodor, 1975; Jackendoff, 1983; for discussions of connectionist approaches to types and tokens, see Mozer, 1991; Marcus, 1999). For instance, *Daffy* is an instance of the class **duck;** at the same time, the class **duck** is not equivalent to the instance *Donald.* The class **duck** is also not equivalent to the set of all actual ducks, since the class **duck** also includes fictional ducks, no longer existing ducks, and so forth.

Given a distinction between instances and classes, it is natural to express generalizations that hold with respect to classes of entities, such as *all ducks can swim.* Rather than specifying individually that *Daffy* likes to swim, *Donald* likes to swim, and so forth, one can describe a generalization that does not make reference to any specific **duck,** by using a variable. The English sentence *all ducks can swim,* for instance, might be translated into the predicate calculus formulation, *for all* x*, if* x *is a duck, then* x *can swim.* Variables such as these allow us to express generalizations compactly (Barnden, 1992; Kirsh, 1987).

The machinery of variables and operations over variables also enables generalization to novel instances of a class. As soon as an item is assimilated into a class, that item automatically inherits the privileges of that category. If *Daffy* is a member of the class **duck,** it can be inferred that *Daffy* can swim.

In contemporary computers, variables typically refer to registers; these registers contain either the values of instances of those variables, or pointers to other registers that contain the values of those instances. Registers themselves can be implemented in a variety of physical media, ranging from Tinkertoys to vacuum tubes to rewritable optical disks.

Although *eliminative* connectionist models do not incorporate variable-manipulation, variable-manipulation is not intrinsically incompatible with connectionism. A branch of connectionism that Pinker and Prince (1988) dubbed *implementational connectionism* seeks to understand how symbol-manipulation and connectionism could be reconciled. The goal of this approach is to use connectionism as a tool to understand how symbol-manipula-

tion could be implemented in a neural substrate (e.g., Barnden, 1984, 1992; Fahlman, 1979; Feldman & Ballard, 1982; Hinton, 1990; Holyoak, 1991; Holyoak and Hummel, in press; Shastri & Ajjanagadde, 1993; Smolensky, 1995; Touretzky, 1991; Touretzky & Hinton, 1985).

Building a connectionist implementation of variable-manipulation would entail finding a way to keep variables distinct from their instantiations. One way to do this is to assign a different node to each variable and to each instance. For instance, one node might correspond to the variable **agent-of-loving,** while another node might correspond to a possible instance of that variable, such as *Peter* or *John.* The binding between an instance and a variable (i.e., the mechanism that indicates how a given variable is currently instantiated) can then be encoded either by activating the variable and its current instantiation simultaneously in a common rhythm (Hummel & Biederman, 1992; Hummel & Holyoak, 1997; Shastri & Ajjanagadde, 1993), or, more generally, by attaching a shared identification code to both the instance and the variable (Lange & Dyer, 1996). A second way of keeping variables distinct from instances—analogous to the way in which a computer uses a memory register to indicate a variable and voltage levels to indicate the instantiation of that variable—is to use a separate bank of units for each variable, with the activation values of a given variable's bank of unit then representing that variable's instantiation. For example, one bank of units might encode the **agent-of-loving,** another the **patient-of-loving.**

A complete connectionist implementation of variable-manipulation would also have to include a mechanism for performing one or more *operations* over those instances, such as copying the contents (i.e., the current instantiation) of one variable into the contents of another. Computers do this with *instructions,* bits of code that tell the computer what to do with the contents of some memory register, such as ''place them into another memory register'' or ''compare the contents of register A with the contents of register B.'' Crucially, these operations are defined to work over all possible instantiations of those registers, and their performance is indifferent as to whether the current instantiation is an instantiation that has previously been encountered. For instance, the copy operation in an 8-bit microprocessor could copy the string of binary bits [1 1 1 1 1 1 1 1] even if it happened to have never before encountered a string containing a 1 bit in the rightmost position of an input string. A connectionist implementation of variable-implementation would provide a way of performing operations over variables in a connectionist substrate.

3. ELIMINATIVE CONNECTIONISM

3.1. Burden of Proof

An implementational connectionist must not only show that some connectionist model is adequate, but also that the adequate model serves as an im-

plementation of some reasonable symbol-manipulating algorithm; likewise, an eliminative connectionist must show not only that some model is adequate, but also that the adequate model is not a covert implementation of the very symbol-manipulating models that it aims to eliminate.

A full-fledged eliminative connectionist research program would thus include systematic comparisons between eliminative connectionist models and symbolic models, showing why the two genuinely differ—a task that is by no means easy. Part of the reason that the task is not easy is that there are so many different ways of implementing the algorithm that underlies a given symbol-manipulating model. For example, as Marr (1982) famously pointed out, one can implement a (symbol-manipulating) tic-tac-toe-playing model with a carefully structured set of Tinkertoys. The crucial notion is one of mapping or correspondence: to show that two models are in some sense equivalent, one must show that there is a systematic mapping between the two, both in the predictions they make and in their internal states.

Because mappings can take on so many different forms, showing that a given device A does not implement the algorithm that underlies some particular model B is difficult: one can never simply itemize all the possible mappings and show that none apply. Still there are ways to show that two models (or algorithms) differ. One way to establish that model A does not implement the algorithm underlying model B is by showing that A and B do not make the same predictions. Another possibility is to show that the intermediate states of A do not map onto the intermediate states of B. (Strictly speaking, any given connectionist model can be implemented by some symbol-manipulation model, such as the computer program that simulates the model. The question is thus not really whether a given model can be implemented by *any* symbol-manipulating algorithm, but rather whether the proposed connectionist model systematically maps onto some reasonable symbol-manipulating alternative. A carefully stated argument for an eliminative connectionist model of some domain should thus specify some set of reasonable symbol-manipulating models and show why that connectionist model does not map onto any of that set of models.)

At this point, one might wonder whether there really are any eliminative connectionist models; scholars such as Lachter and Bever (1988) have argued that some apparent eliminative connectionist models covertly implement symbol-manipulation. Indeed, even the most radical connectionist models incorporate some elements of symbol-manipulation, including the use of symbols. A given node, for instance, might be activated if and only if the word ''cat'' appears in the input stream; such a node thus defines an equivalence class of utterances of the word ''cat,'' hence it serves as a symbol (Marcus, 1999; Vera & Simon, 1994). Likewise, it could be argued that many apparent eliminative connectionist models incorporate distinctions between variables (banks of nodes) and instances (represented by their activity levels). Still, there are at least two important ways in which standard parallel distrib-

uted processing models clearly differ from the standard picture of symbol-manipulation. First, these models differ with respect to their treatment of compositionality (a topic that is outside the scope of this paper; for discussion see Fodor & Pylyshyn, 1988; Marcus, 1999). Second, these models differ with respect to their treatment of operations over variables, the subject of the current article.

3.2. How Eliminative Connectionist Models Work

Whereas variable-manipulating systems include a facility for representing abstract relationships between variables, such as **x** = **y** + 2, or **past** = **stem** + *ed*, eliminative connectionist models such as feedforward networks and simple recurrent networks do not include any explicit representation of a relationship between variables.

Instead, the mapping between input and output is represented through the set of connection weights. As mentioned in the Introduction, the connection weights in these models are usually initialized to random values, rather than prespecified in advance. These weights are subsequently adjusted through training with the back-propagation algorithm or one of its variants (e.g., Fahlman & Lebiere, 1990). Such algorithms compute error with respect to some target pattern, and adjust connection weights according to that measure and a mechanism for ''blame-assignment'' that distributes the error among the units feeding a given node. The motivation for these rules, as McClelland and Rumelhart (1986, p. 214) put it, is to

> provide very simple mechanisms for extracting regularities from an ensemble of inputs without the aid of sophisticated generalization or rule-formulating mechanisms that oversee the performance of the processing system.

Moreover, as McClelland and Rumelhart note in the same paragraph

> These learning rules are completely local, in the sense that they change the connection between one unit and another on the basis of information that is locally available to the connection rather than on the basis of global information about overall performance.

In other words, eliminative connectionist models replace operations that work over variables with local learning, changing connections between individual nodes without using ''global information.''

Dozens of models that follow this overall strategy can be found in journal articles, books, and conference proceedings. To my knowledge, virtually all current eliminative connectionist models adopt this strategy. (Section 7 considers whether eliminative connectionism could be supported through other kinds of models.) The remaining discussion will concentrate on three running examples, chosen for their representativeness and influence. Nothing rests on these particular examples, however; many similar models could be used to illustrate the same points.

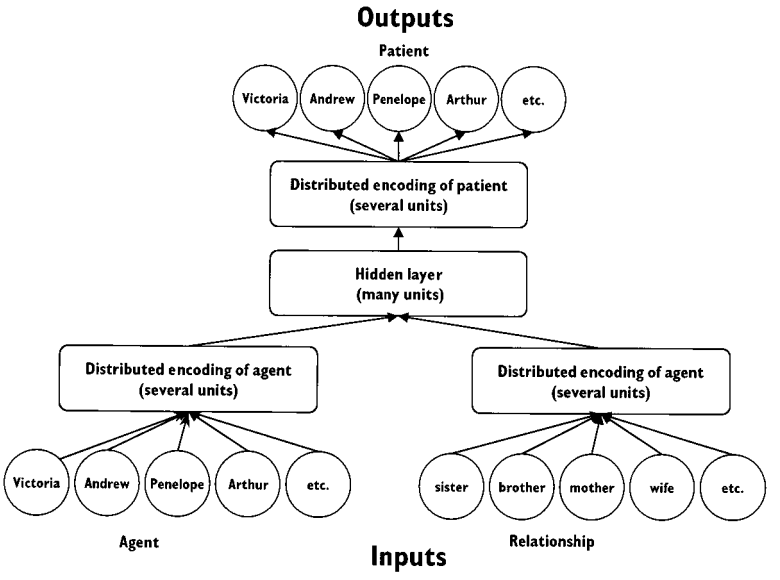

FIG. 3. Sketch of the family tree model of Hinton (1986). Not all units or all connections are shown. Circles indicate units; rectangles indicate hidden layers consisting of multiple units. The input to the model is indicated by activating one agent unit and one relationship unit; the set of patients corresponding to that agent and relationship are activated within the output bank. For example, to encode the fact that Penny is the mother of Arthur and Victoria, the input units corresponding to the patient *Penny* and the relationship *mother*, and the output units *Arthur* and *Victoria* would be activated.

3.2.1. The family tree model. One important and influential early eliminative connectionist model was the ''family tree'' model of Hinton (1986), which aimed to learn abstract relationships like **sister** and **mother.** The model, sketched in Fig. 3, was trained on facts of the form ''X is the Y of Z,'' such as *Penny is the Mother of Victoria and Arthur.* These facts were drawn from two isomorphic family trees (e.g., the second tree contained the same number of people and the same number of relationships in each generation as the first). In all, there were 104 possible facts; in one test run, the model was trained on 100 of these 104 facts and was able to generalize to 3 of the remaining 4 facts; in the other test run (with a different set of randomized initial weights), the model was trained on the same 100 facts and was able to generalize to all four of the remaining facts.

3.2.2. The balance-beam model. Another important, influential model is the model of the development of children's understanding of the balance-beam by McClelland (1989), which Shultz et al. (1995) described as ''the pioneering attempt to apply modern connectionist techniques to developmental problem solving tasks.'' In this study, a network is confronted with

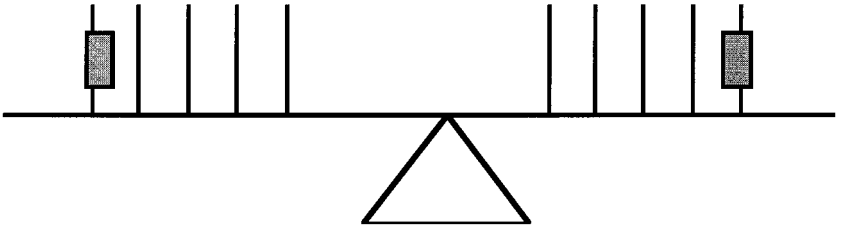

FIG. 4. The balance beam task.

a version of the balance beam problem that contains five equally spaced pegs on either side of a fulcrum, depicted here in Fig. 4.

The network, depicted in Fig. 5, is a feedforward network with a bank of 20 input units, 4 hidden units, and 2 output units. The output units represent the relative weight on the left and right sides of the balance beam. The 20 input units are divided into two banks of 10, one for the left side of the beam, one for the right. Each bank of 10 is in turn subdivided into a bank of 5 units representing the number of weights on some peg and a bank of 5 units representing the distance that the weight-bearing peg is from the fulcrum.

McClelland trained this network on all 625 possible inputs. (625 = 5 possible positions for the left object * 5 possible weights for the left object *

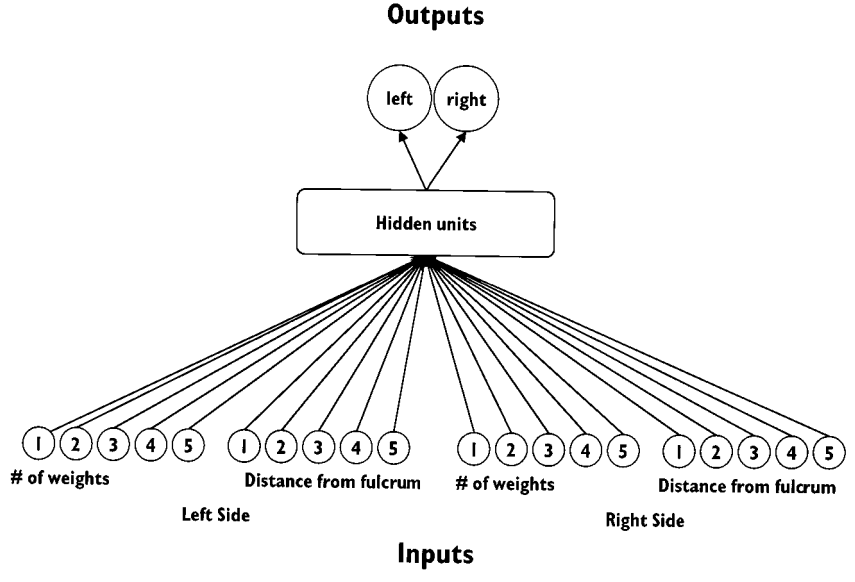

FIG. 5. McClelland's balance beam model. See text for explanation.

5 possible positions for the right object ∗ 5 possible weights for the right object). After sufficient training, the model was able to produce accurately outputs corresponding to whether the balance beam will tip.

3.2.3. The sentence-prediction model. A third example is the simple recurrent network (henceforth, SRN) account of learning aspects of language by Elman (1990, 1991, 1993). In a recent survey of articles published from 1990 to 1994 (Pendlebury, 1996), the paper that introduced the model (Elman, 1990) was the most widely cited paper in psycholinguistics, and the 11th most cited paper in psychology.

The simple recurrent network is especially important because it has been used to motivate an argument that the problems of variable binding may simply be irrelevant to adequate accounts of cognition. Garson (1993) used the SRN to propose

> an alternative connectionist paradigm that takes the project of understanding [variable] binding much less literally . . . solutions to the ''binding problem'' emerge from weight selection in a general purpose architecture . . . This line of research . . . shows at least that some implicit binding can be handled without special architecture.

The task of this network is prediction: given a string of input items such as words, the model tries to predict what might come next. These sequences, produced by external symbolic grammar, are presented to the model in a word-by-word fashion, one word per sweep through the network. At each time step, the model is presented with a given word in a sentence; the target (i.e., the pattern on which the network is trained) is the subsequent word in that sentence. (The sequential device that governs which word is presented to the model at each time step is currently implemented in a symbol-manipulating algorithm, one that would ultimately have to be replaced by some device that does not manipulate symbols.) The network is not trained on all possible continuations simultaneously, but rather on only one continuation in any given training step. A somewhat simplified sketch of the model is given in Fig. 6.

The network contains an input layer, a hidden layer, a context layer, and an output layer (some but not all versions of the model also include two additional ''transducer'' layers). At the conclusion of each time step, the contents of the hidden layer are copied (using fixed rather than trained connections) to a context layer; the contents of the context layer then feed back into the hidden layer at the next time step, providing the network with access to temporal information. Input words are encoded locally; each word is represented by a single unit; the input vector is a string of 0's with a single 1 corresponding to the node that is activated by that word. Output nodes correspond to individual words. The network is, at any given time step, trained on an output vector that contains a string of 0's with a single 1 encoding the actual continuation word. At any given point, the network will tend to activate more than one output. Although one could plausibly interpret such

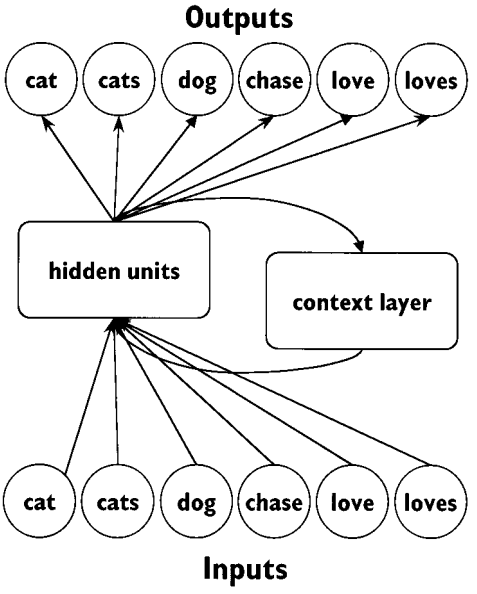

FIG. 6. Sketch of a simple recurrent network used to predict sequences of words. The actual model had more input and output nodes. Rounded rectangles indicate sets of units. Weights from hidden units to context units are fixed at 1.0; all other weights are modifiable. See text for further details.

an output as a blend between different words, Elman draws a different interpretation that is designed to capture the fact that more than one continuation is possible. In particular, Elman takes the output vector to be a ''likelihood vector'' according to which the activation of each node corresponds to the relative probability of that given word appearing as a continuation.

This network has been applied to a variety of temporal learning tasks. For example, Elman trained the network on a ''semi-realistic artificial grammar'' that included a variety of dependencies such as subject–verb agreement (*cats love* vs *cat loves*). Elman argued that dependencies (as in the relationship between the subject and the verb) were particularly important because they figured prominently in earlier arguments against statistical models that lacked explicit grammatical rules (e.g., Miller & Chomsky, 1963). To the degree that the model genuinely captured the underlying abstract relationships, it might undermine those earlier arguments.

Once the model was trained, it was often able to predict plausible continuations for strings such as *cats chase* ____, and even more complicated strings such as *boys who chase dogs* ____—just the sort of cases that Miller and Chomsky used to argue against earlier statistical models.

It is difficult, however, to evaluate the model's performance. The primary quantitative measure of the SRN's performance that Elman provided was to

compare which continuations the model predicts with what actually comes next within a test corpus. There are two problems with this measure: first, since the test corpus was ''generated in the same way as the . . . training corpus'' (1991, p. 204), it is not clear to what degree the test corpus actually examined the model's ability to generalize as opposed to its ability to memorize (Hadley, 1994). Second, Elman provided no formal way of comparing how well the model did on this task with how well a human might do on a comparable task. What I will show in Section 5 is that the SRN network does not derive the same abstraction as people do and that what it can learn is severely restricted in its generality.

4. TRAINING SPACE

4.1. Definitions

In what follows, it is necessary to distinguish two kinds of generalization, generalization that is within a training space, and generalization that is outside that training space. What counts as being within the training space will depend on two things: the set of training examples and the representational scheme.

The following discussion assumes that each input to a network is composed of a set of *n* Boolean features, such as [+/−animate]. This set of *n* features can be used to define an *n*-dimensional space that I will call the *input space.* It is clear that any possible input item corresponds to a point somewhere in the input space. The *training set* is the set of points in the input space on which the model is trained.

Let us call each binary value of a feature a *feature value.* For example, let us treat [+animate] and [−animate] as two distinct feature values; each input will be composed of exactly *n* feature values. An input that is not in the training set but that is composed entirely of feature values that appeared within the training set lies within the *training space.* Any input that includes a feature value that did not appear within the training set lies *outside the training space* (e.g., if all the model's inputs are [+animate], any item containing the feature value [−animate] would lie outside the training space). The training space is thus a subspace contained inside the input space, delimited by the values of the features that appeared in the training set.[5]

Given these definitions, one can distinguish between test items that lie *within the training space* (that is, those items that are comprised purely of feature values that a model has been trained on), and those test items that

[5] While many items within the training space will be either items from the training set or linear combinations of items from the training set, the training space also includes items that are not linear combinations of input items. For example, if the training set were the vectors [101] and [011], the item [111] would not be a linear combination of the input items but would lie within the training space.

lie *outside the training space* (that is, those items that include feature values that the model has not been trained on). Informally, generalizations within the training space can be thought of as interpolations, while generalizations outside the training space can be thought of as extrapolations.

4.2. Problems That May Not Require Going beyond the Training Set

Not every problem requires a learner to go outside the training space, or even outside the training set. Some cognitive tasks (e.g., memorizing telephone numbers) may be appropriately cast as problems in which a learner need not generalize at all. In such tasks, it may suffice to merely memorize the training set. (Eliminative connectionist networks with sufficient resources can, given sufficient training, memorize any training set in which each input corresponds to exactly one output. Among the open questions is whether such networks can master such training sets in a plausible number of trials, and whether they can do so without excessive interference, e.g., McCloskey & Cohen, 1989.)

Likewise, tasks in which there is a relatively small number of possible inputs, e.g., the game of tic-tac-toe, can, in principle, be mastered by mere rote memorization. (In a standard 3 by 3 tic-tac-toe board, there are 3^9 possible positions, including rotations, reflections, and impossible positions.)

4.3. Problems That May Not Require Generalization beyond the Training Space

Other cognitive tasks clearly involve some degree of generalization. Among such tasks, some may demand only that a learner may generalize to items that are wholly comprised of features on which the learner has task-relevant experience. Such tasks would not require generalization outside the training space. For example, in some areas of motor control and skill learning, it seems plausible that a learner might not be able to generalize outside the training space (for one possible example of such a domain, see Ghahramani, Wolpert, & Jordan, 1996).

Reading is a task that may or may not require generalization outside the training space. In all current models, some possible words would lie outside the plausible training space but are nonetheless readable. For example, in the Seidenberg and McClelland (1989) model, words that contain untrained phoneme triples (i.e., untrained Wickelphones) would lie outside the training space. Other models of reading typically treat each syllable with a distinct set of units. In such models, generalizing to a word that contained more syllables than any training word would require that the model generalize outside the training space. Still, the possibility remains that researchers could eventually discover a way of representing all readable words within a space in which a network can plausibly be trained. If such a representation existed, reading might be adequately modeled by an architecture that cannot generalize outside the training space.

4.4. Problems That Do Require a Model to Generalize beyond the Training Space

In a variety of domains, people can freely generalize. For example, people can generalize kinship terms (*sister, uncle,* etc.) to new families or new family members, people can understand balance-beam problems with arbitrary numbers of weights or pegs, and people can generalize grammatical relationships to arbitrary words.[6]

The ability to freely generalize relationships to arbitrary items is not restricted to conscious rule use (cf., Smolensky, 1988). For example, in carefully controlled experiments Tomasello and Olguin (1993) showed that children less than 23 months old can take a nonsense noun that they heard used only as a subject and use it as an object; in this instance, a child is generalizing a grammatical rule to an arbitrary item, even though it is unlikely that the child could articulate the relevant rule explicitly. Similarly, my colleagues and I (Marcus, Vijayan, Bandi Rao, & Vishton, in press) have found that even 7-month-old infants appear to be able to generalize abstract language-like rules to new words.

In order to capture cases of free generalization to arbitrary items, current eliminative connectionist models would need to generalize outside the training space. For example, in the family tree model, each new person is represented by a new node (and thus a new feature value); generalizing family relationships to new people would thus depend on generalization outside the training space. In the balance beam model, each new number of weights is represented by a new node; generalizing to a balance beam that contained a new number of weights would thus depend on the ability to go outside the training space. In the sentence-prediction model, each new word is represented by a new node (and thus a new feature value); generalizing to a new word would thus depend on the ability to go outside the training space. Since humans can freely generalize in these domains, the viability of a given model depends on whether it can generalize outside the training space.

5. GENERALIZATION OUTSIDE THE TRAINING SPACE

5.1. Evidence from Simulations

Suppose that you were trying to learn the function that is illustrated in Table 1.

What is the appropriate response to the input pattern [1 1 1 1 1]? Although as an inductive problem, there can be no single correct answer to this question, in informal testing, I have found that human adults consistently

[6] Another, perhaps more subtle linguistic mapping that appears to be depend on the ability to generalize outside the training space is the relationship between the underlying phonological form of a word and its surface form (Berent, Everett, & Shimron, 1998).

TABLE 1
A Sample Function

Training cases										
Input					Output					
0	0	0	1	0	0	0	0	1	0
0	0	1	0	0	0	0	1	0	0
0	0	1	1	0	0	0	1	1	0
0	1	0	0	0	0	1	0	0	0
0	1	0	1	0	0	1	0	1	0
0	1	1	0	0	0	1	1	0	0
0	1	1	1	0	0	1	1	1	0
1	0	0	0	0	1	0	0	0	0
1	0	0	1	0	1	0	0	1	0
1	0	1	0	0	1	0	1	0	0
1	0	1	1	0	1	0	1	1	0
1	1	0	0	0	1	1	0	0	0
1	1	0	1	0	1	1	0	1	0
1	1	1	0	0	1	1	1	0	0
1	1	1	1	0	1	1	1	1	0

Note. See text for details.

predict that the output corresponding to the input [1 1 1 1 1] is [1 1 1 1 1]. Humans generalize this *identity* or *sameness* relation freely, both to cases within and outside the training space. (The test pattern [1 1 1 1 1] lies outside the training space, because the feature value *1-in-the-rightmost-position* did not appear in the training set.)

Standard eliminative connectionist models generalize this function in a different way. For example, in a series of simulations, I trained feedforward networks with 10 input units, 10 output units, and 10 hidden units on an extended version of the problem shown in Table 1, in which each input string consisted of 10 binary input digits. In one condition, I trained the network on all $2^{10} = 1024$ possible input patterns; the network readily mastered this training set. In the other condition, I trained the network only on the even numbers, leaving the feature-value 1-in-the-rightmost-position—hence all odd numbers—outside the training space. In this condition, the network did not generalize identity to the odd numbers; instead, for example, the network responded to the input [1 1 1 1 1 1 1 1 1 1] with the output [1 1 1 1 1 1 1 1 1 0].

The network did not generalize the identity function to odd numbers even when I varied the learning rate, number of hidden units, the number of hidden layers, and the sequence of training examples.

It is important to realize that the network's response is a perfectly reasonable induction, mathematically consistent with the input. (Note, for instance, that within the training set, the conditional probability of the rightmost digit

of the output's being a ''1'' is zero.) Thus the network is not ''wrong'' in any absolute sense; rather, what is important for present purposes is that the inductions of the model sharply differ from those made by humans.

In a further experiment, I tested a version of the simple recurrent network with 13 input units, 13 output units, a layer of 40 hidden units, and a layer of 40 context units on a problem of identity-over-time. In this task, I presented the network with a series of sentences such as *a rose is a rose, a tulip is a tulip,* and so forth, and then tested the model on another sequence of the same general form that contained a novel word, *a blicket is a* ____. Whereas humans tend to complete that sequence with the word *blicket,* the simple recurrent network did not activate the output unit corresponding to *blicket.* In some replications, the network activated other words such as *rose* or *tulip;* in other replications, no word was strongly activated. Which word (if any) is activated strongly depends on the set of random weights that are initially assigned.

Other recent experiments have yielded similar results.[7] For example, DeLosh et al. (1997) tested the ability of an eliminative connectionist model to extrapolate linear, exponential, and quadratic functions, and compared that result with the extrapolation abilities of human subjects. They found that although humans were able to extrapolate these functions beyond the range of trained responses, the eliminative connectionist model that they studied was not able to extrapolate adequately beyond the range of trained responses. (A related failure to extrapolate is described in Busemeyer et al., 1997).

Similarly, a person who is trained on examples from the finite state grammar that permits sequences like GFFFFFQLLLLL and GFFFQLLL can use the abstract structure encoded there to facilitate the learning of a grammar that shares the same formal structure but has a different lexicon (e.g., a grammar containing strings like PXXXXRTTTT and PXXRTTTT). Such transfers by definition would require going outside the training space in a simple recurrent network that represented each letter by a unique node. Frank Hong and I (Hong & Marcus, 1996) found that this transfer effect cannot be modeled by the simple recurrent network, because the network cannot generalize

[7] A failure to generalize outside the training space may also have been involved in a somewhat earlier finding, by Prasada and Pinker (1993). Humans appear to be able to extend the *-ed* suffixation process to novel words regardless of a word's similarity to stored examples, even to words that contain sounds unfamiliar in English, like *Jelsin out-gorbacheved Gorbachev.* In simulations, Prasada and Pinker found that although the Rumelhart and McClelland (1986) model can in some (though not all) circumstances apply generalizations to novel items that strongly resemble training items, it encountered difficulty when inflecting input words that lacked resemblance to trained examples. The network tended to produce weird responses like *fraced* as the past tense of the novel word *slace, imin* as the past tense of *smeeb, bro* as the past tense of *ploanth,* and *freezled* as the past tense of *frilg.* Although Prasada and Pinker did not distinguish between items that were and were not in the training space, it is likely that the unfamiliar sounding words were outside the training space.

outside the training space. (Dominey, 1997, appears to have found a similar result.)

Likewise, an analogical reasoning problem called the Klein 4-group task requires generalization of geometric relations from one set of items to another set of items, hence generalization outside the training space. While humans routinely transfer in this task, Phillips and Halford (1997) found that neither a feedforward network model of this task nor a simple recurrent network model was able to capture this transfer.

Each of these examples illustrates the inability of feedforward networks and simple recurrent networks to generalize outside the training space. The next section explains the mathematics that underlie this limitation.

5.2. Training Independence

In hindsight, the fact that feedforward networks and simple recurrent networks that are trained through localist error-correction algorithms are unable to generalize outside the training space should be unsurprising. Informally, a unit that never sees a given feature-value is akin to a node that is freshly added to a network after training. A node that did not participate in training obviously will not behave in the same way as a node that did participate in training.

More formally, the reason that these networks cannot generalize outside the training space can be understood in terms of the equations of the back-propagation algorithm that adjusts connection weights. First, the equations of the back-propagation algorithm are such that whenever an input node is activated at the level of zero, its connections to other nodes remains unchanged—regardless of what happens with all the connections that feed from other input nodes. The amount in which a given connection weight that emanates from unit *i* changes is determined by an equation that includes as a multiplicative factor the activation level of unit *i;* thus whenever the activation level of *i* is zero, the weight change must be zero, regardless of the activity levels of other input units. This aspect of the back-propagation algorithm can be called *input independence.* (An as-yet unproven corollary that appears to be true is that if during training any given input does not, alone or in combination with other units, predict the output, that unit will effectively be trained independently of any other input unit.)

Output units are also trained independently from one another. This, too, follows directly from the equations that define back-propagation. For a given input–output pair, the weight on the connection from a given input/hidden unit *a* to output unit *j* is adjusted according to the following equations, derived in Rumelhart et al. (1986a),[8]

[8] These equations assume that the output units are activated according to the logistic function that is used in virtually all multilayer networks.

$$\begin{aligned}\text{change in weight of connection from unit } a \text{ to output unit } j \\ = \text{learning rate} * \text{error signal} \\ * \text{activation of input/hidden unit } a,\end{aligned} \tag{1}$$

where

$$\begin{aligned}\text{error signal} = (&\text{target for unit } j \\ &- \text{observed activation of output unit } j) \\ &* [\text{activation of unit } j \\ &* (1 - \text{activation of unit } j)].\end{aligned} \tag{2}$$

Crucially, these equations, which change the weights that feed a given output unit, do not make any reference to the activation levels of the other outputs, the targets of other outputs, or the weights feeding the other outputs. Consequently, the set of weights connecting one output unit to its input units is adjusted entirely independently of the set of weights feeding all other output units, a limitation that can be called *output independence.*

Generalizations are therefore not transferred from output unit to output unit. For example, consider the auto-associator model of identity described earlier in Subsection 5.1: the weights that feed into the output unit corresponding to a novel item are trained independently of the weights that feed into the other output units. Thus, on any given trial, the target for output unit A has no impact on the adjustment of the weights the feeds output unit B; hence there is no transfer between nodes.[9]

In light of the output independence limitation, one way to understand contemporary eliminative connectionist networks is as a set of *independent* classifiers (see Touretzky, 1991, for a similar suggestion). That is, each output unit computes its own classification function. In this way, multilayer networks are quite similar to two-layer networks. In a two-layer network, each output unit computes a (positively or negatively) weighted combination of the input features; modulo the differences introduced by nonlinear activation functions in a three-layer network, each output unit computes a positively or negatively weighted combination of the combinations constructed by the units that feed it. Thus although adding additional hidden units allows the network to add additional combinations of features, and adding additional hidden layers allows the network to exploit combinations of combinations of features, no such addition allows the network to exploit abstract universals. Each categorizer (i.e., output unit), from the network's perspective, is an entirely separate problem. To learn a general function across all output

[9] A somewhat different argument for a similar conclusion is given in Phillips (1994).

units, the network must have relevant experience on each possible output unit, a limitation that is entirely unaffected by the introduction of hidden layers.

Though the training space itself is defined by the nature of the input and output representations, the limitation applies to a wide variety of possible representations. Regardless of the input representation chosen (so long as units represent a finite number of distinct values), there will always be features that lie outside the training space. Any feature that lies outside the training space will not be generalized properly—regardless of the learning rate, the number of hidden units, or the number of the hidden layers.

5.3. Scope

Training independence applies to simple recurrent networks and feedforward networks; more generally, it would apply to any model in which each connection weight was trained independently.

The training independence limitations do not undermine the use of these architectures in tasks that would not require generalization outside the training space,[10] but do undermine their use in tasks that would require generalization outside the training space. For example, consider the family-tree model. What the model learns about family members that it is trained on will not generalize to new sets of family members, because the latter cases will be by definition outside the training space. A human who is told that ''the sibling of Cain is Abel'' can immediately infer that ''the sibling of Abel is Cain,'' but the family-tree model cannot; it never genuinely abstracts the symmetric relationship underlying ''sibling.'' Instead, because of training independence, the family-tree model can only simulate that relationship with respect to a set of heavily trained items.

Similarly, the balance-beam model cannot generalize to balance beams of a width greater than the balance beams in training or to balance beams that contain more weights than any of the beams that appeared in training. For example, if the model is trained only on balance beams of width four, it cannot reliably distinguish a problem containing a weight on the 5th point left of center balanced against a weight on the 6th point from the right. Training independence guarantees that the same problem would hold if the model were trained on balance beams 100 units wide and tested on balance beams 104 units wide.

[10] Although I have shown that current eliminative connectionist models cannot generalize outside the training space, I have left open the extent to which eliminative connectionist models can generalize within the training space. Sometimes eliminative connectionist models can generalize adequately within the training space, sometimes, as in the parity test, conducted by Clark and Thornton (1997), a given model cannot generalize adequately within the training space. The ability to draw such generalizations (unlike out-of-training-space generalizations) depends on the details of how many hidden units are used, what the learning rate is, what the nature of the function to be learned is, and so forth.

Likewise, the sentence-prediction model cannot generalize to novel words any linking relationship in which two items must be identical. While the ''rose is a rose'' sentences tested above are just a tiny, somewhat artificial part of language, such linking relationships are pervasive. For example, such a linking relationship is implicit every time a referential item (e.g., *himself*) is linked to an antecedent. Presumably, when we read or hear the word *himself* in a string such as *Peter loves himself,* we mentally reactivate some mental representation of Peter. The simple recurrent network cannot account for how we do this with novel words (e.g., how we reconstruct the antecedent *himself* in the sentence *Dweezil loved himself*).[11] Likewise, in order to be able to answer a question about some discourse, we must draw a link between a question word (say, *who* in the query *Who does John love?*) and some entity (say, *Mary*). The simple recurrent network provides no direct way of answering questions about the sentences it is exposed to, but one can test this ability by training the network on sentences such *John loves Mary. Who loves Mary? John does.* As I confirmed in further simulations, training independence keeps the simple recurrent network from being able to answer questions about novel entities. The simple recurrent network is thus inherently unsuited to modeling those aspects of language that involve genuine linking dependencies, including question-answering and antecedent-resolution.

5.4. Other Kinds of Generalization

Although training independence guarantees that each output unit is trained independently, and that each input unit is trained independently, multilayer networks can still generalize in interesting ways. Suppose for instance that a simple recurrent network is trained on a series of sentences in which two items *John* and *Bill* appear in virtually identical circumstances, say in the sentences *John loves Mary, John loves Susan, Bill loves Mary,* and *Bill loves Susan;* the weights leading from those two inputs into the hidden layer will be nearly identical. In such a case, the two input units that encode John and Bill, respectively, would elicit nearly identical patterns of hidden unit activity. (Another way of putting this is that the items *John* and *Bill* would cluster together in hidden unit space.) Similarly, given the same set of sentences, the output units corresponding to *Susan* and *Mary* would be connected to the hidden units in nearly identical ways, despite the fact that each change

[11] Elman's experiments with subject-verb agreement obscured this point, because he did not test whether the model could retrieve the specific subject at the point at which it predicted verb agreement. In fact, all the network had to do was keep track of which class of words the subject belonged to (e.g., noun or verb, singular or plural, animate or inanimate). To do that, it sufficed to have all words in a class elicit a common pattern of hidden unit activation. But if (say) *cats* and *dogs* elicit identical hidden unit activation patterns, their individual identity has been lost; hence there is no way for the network to recover whether the subject was *cats* or *dogs.*

of weight leading to the *Susan* unit would occur independently of each change of a weight leading to the *Mary* unit.

Given that the *Susan* and *Mary* units are connected to the hidden units in essentially the same way, if the network were now exposed to a sentence fragment containing a novel word, say *John blickets* ____, it would be equally likely to activate the continuations *Susan* and *Mary.* In fact, I implemented precisely this example and found that the network activated both *Susan* and *Mary* to levels of about 0.70. Both units were activated far more than any other units, so it is reasonable to say that the network correctly predicted both as possible continuations, just as a human might in similar circumstances.

This turns out, however, to be a case of getting the right prediction for the wrong reason. In a second stage of training, I trained the network on *John blickets Susan,* but not on *John blickets Mary.* In this second stage, the network gradually increased the activation of *Susan* as a continuation for *John blickets* ____, but gradually *decreased* the activation of *Mary* as a possible continuation. Whereas continued experience with the sentence *John blickets Susan,* would make a person more likely (or at least equally likely) to accept as grammatical the sentence *John blickets Mary,* greater experience causes the network to be *less* likely to predict (i.e., accept as grammatical) *John blickets Mary.*

Throughout this example the units representing *Mary* and *Susan* have always been trained independently. What happens in this example is that in the initial stage of training, the two units are trained in nearly identical circumstances. At the conclusion of this initial stage, the *Mary* unit and the *Susan* unit tend to respond in identical ways to novel stimuli. (They are strongly activated by *John blickets* ____ despite the novelty of *blicket* in part because of regularities such as the fact that John is always followed two words later by either *Mary* or *Susan.*) To the extent that *Mary* and *Susan* appear in different training sentences in the second stage of training, the output units that represent them tend to diverge. (Unless the network is stuck in a local minimum, this divergence must occur, because each time that the network predicts *Mary* as a continuation to *John blickets* ____ when the actual continuation was *Susan,* back-propagation adjusts the connection weights leading into the *Mary* unit by a function of the difference between the activation of *Mary* and the (zero) target for *Mary,* in a way that tends to reduce that difference.)

Training independence does not show that a network like the simple recurrent network can never generalize. Such networks can (given the right parameters and training regimen) often generalize within the training space. A model that can account for some but not all the data may well be worth pursuing—provided it is not incompatible with the remaining data. The simple recurrent network, however, is incompatible with generalizing a particular class of universals, *universally quantified one-to-one mappings.* (Univer-

sally quantified relations are relations that hold for all elements in some class; one-to-one relations are those in which every input has a unique output.) When the simple recurrent network succeeds in generalizing to a new input, it succeeds by predicting which previously learned category (or categories) a new input belongs to. After the first stage of training in the *John blickets Mary* example, the network can predict that *Mary* is a possible continuation to *John blickets* ____, because that sentence fragment resembles other sentence fragments (*John loves* ____, *Bill loves* ____) that the Mary node has been trained to respond to. In universally quantified one-to-one functions, it does not suffice to assimilate each new response to an already known category, because each new input maps to a new output. The right answer for a new input in the functions will thus not be a category that the network has seen before. Since the network learns how to treat each category independently, prior experience on other categories does not enable the network to generalize to the new input–output pair. The localism that underlies backpropagation is incompatible with generalizing universally quantified one-to-one mappings to novel items.

Were universally-quantified one-to-one mappings of little importance, such a limitation might be of little consequence. In fact, universally quantified one-to-one mappings are pervasive. For example, each stem form of a verb corresponds to a different past tense form; the past tense of the stem *out-Gorbachev* is *out-Gorbacheved.* Each paternal name corresponds to a different child last name (the child of Mr. Jones bears the last name Jones; the child of Mr. Bfltspk bears the last name Bfltspk). The playing card that forms a pair with 2 is 2; the card that would form a pair with a novel card, say, a Duke is a Duke. These sorts of mappings can be captured in a system that has operations that manipulate variables that are instantiated with instances, but they cannot be captured by simple recurrent networks or feedforward networks.

6. THE ROLE OF EXPERIENCE

An obvious concern is that humans have more experience than the models described here. Nonetheless, two considerations suggest that the limitations of training independence would also undermine feedforward networks or simple recurrent networks with a great deal more experience.

First, training independence is not alleviated by additional training examples of the same kind. The simple recurrent network, for example, could be trained on the linking relationship underlying ‘‘an X is an X’’ for 999 different words, and it would still fail to generalize to the thousandth word.

Second, the inability to generalize as humans do is not necessarily alleviated by additional training examples of different kinds. For example, I conducted an experiment in which I trained the ‘‘identity’’ network (from Subsection 5.1) on the sameness relationship for all ten inputs. This initial

background phase allows the network to ''know what each unit stands for.'' Next, I confirmed that this network mastered the training set, even for the tenth (rightmost) input node (the one that distinguishes odd and even numbers). Next, I trained this pre-trained network, using all of the even numbers as inputs, on a second function, which can be called ''string reversal,'' a function that transforms, e.g., [1 1 0 0 1] to [1 0 0 1 1]. Even with the additional background training on the identity function, the network did not generalize string reversal to the odd numbers. Although the tenth bit lies inside the training space for the identity function, it *lies outside the training space for the string reversal function.* Because the tenth bit lies outside the training space of the string reversal function, the network does not generalize the string reversal function to the tenth bit. Similarly, I found that training the simple recurrent network on the sentences such as *the bee sniffs the blicket* (as well as sentences such as *the bee sniffs the flower, the bee sniffs the tulip,* and so forth), does not help the network infer that the continuation to a *blicket is a* ____ is *blicket.* What matters is the training space *with respect to some particular function.* Any item that is outside the training space with respect to that function—even if it is within the training space of some other function—will not be generalized properly.

While simple recurrent networks and feedforward networks can not generalize to items that are familiar, but new to some particular function, humans can. For example, consider the following two examples that consist of a novel function entirely of well-practiced letters:

W X H → H X W
[read as ''given the input W X H, the output is H X W'']

H K X → X K H.

What output corresponds to the input ''S O C''? On virtually any account of how inputs are represented, the input ''S O C'' lies outside the training space of the function that is illustrated. For example, ''S'' would be outside the training space of the function that is illustrated, whether ''S'' were encoded locally as +**S,** through sets of high-level distributed features such as +**curved-line,** or through sets of low-level distributed features such as +**pixel-in-the-center-of-the-bottom-row.**

Despite the fact that the input ''S O C'' lies outside the training space of the function that is illustrated, people readily infer that the corresponding output is ''C O S.'' (People can extend the reversal relation to novel squiggles that are not even letters, transforming the input ''✺ ✉ ➔'' into the output ''➔ ✉ ✺.'') Indeed, human generalizations of the reversal pattern seem to be indifferent as to whether they are applied to items that are within the function's training space (more W's, X's, and so forth) or to items that outside the function's training space (S's, O's, C's, etc.).

In sum, humans can generalize outside the training space of a particular function on the basis of limited experience, whereas neither feedforward networks nor simple recurrent networks can generalize outside the training space, regardless of how large the training space is. Although humans do have more learning experience than any particular network, the difference in how they generalize outside the training space is not due to differences in experience, but rather to differences in the computations that they perform.

7. ALTERNATIVE MODELS

7.1. Alternative Architectures

As already discussed, obvious modifications like adding hidden layers or additional hidden units do not affect the training independence limitations. Such modifications therefore would not allow models to generalize universals beyond the training space. Likewise, dividing these models into modules (Jacobs, Jordan, & Barto, 1991) would not by itself help with problems posed here, since the problem does not lie in the internal topology of the network, but in the training of the connections that run from the internal layer(s) to the output units.[12]

Critics of earlier drafts of this article proposed several alternative connectionist accounts, but each alternative account ignored the distinction between implementational and eliminative connectionism. For example, one suggested that the identity task could be solved by a model that used a ''moving-window'' technique borrowed from Rosenberg and Sejnowski (1987), along the lines of the model illustrated in Fig. 7. The connectionist part is a single input node and single output node, which must be combined with an external control mechanism. This external control mechanism would move (say) left-to-right across the string of input bits, taking one bit at a time and passing that bit to the simple net that connects with the input with weight one to the output node (which would have a linear activation function).

Such a model can indeed extend identity to arbitrary items, but such a model is essentially an implementation of a patently symbolic algorithm like the following:

[12] Modular networks that incorporate ways of binding variables could generalize outside the training space. For example, an extension of the RAAM networks of Pollack (1990) that was proposed by Chalmers (1990) and Niklasson and Gelder (1994) uses two separate networks, one to encode (and decode) all possible instances of variable in a fixed input bank, and another to transform the encoded representations in a systematic way; the structure of these models precisely parallels the division between encoding and computation in standard symbolic models. Such networks are best seen as implementations of those models, not as genuine, nonsymbolic alternatives.

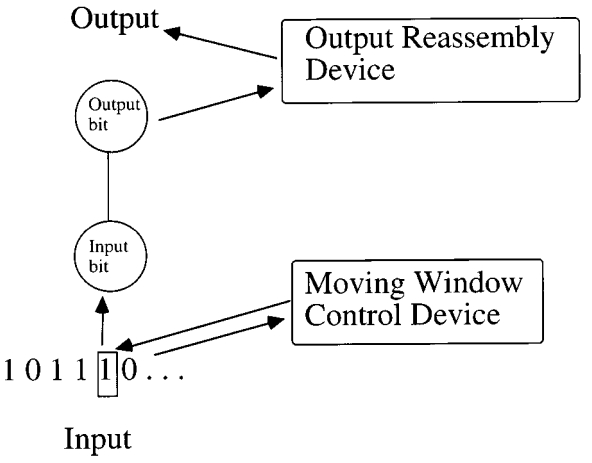

FIG. 7. A model of the identity function. Rectangles indicate devices whose internal structure is not illustrated.

repeat for each bit X in a string

copy bit X

until string is empty.

Since the model makes the same predictions as such an algorithm and has the same intermediate states, it is hard to see how it could be construed as an alternative to symbol-manipulation.

Indeed, the ''connectionist'' part of this model is only a tiny part of a larger system; most of the real burden lies with the serial ''moving window control device'' and the corresponding ''output reassembly device.'' This does not mean that eliminative connectionists must disavow sequentiality,[13] but rather that researchers who claim that symbol-manipulation is incorrect or provides ''a mere approximation'' must provide an account of sequentiality that somehow differs from the account provided by symbol-manipulation.

Another possibility is to build a hybrid model that includes some symbol-manipulating components and some non-symbol-manipulating components.[14] The model that Pinker, Prince, and I have proposed for the English

[13] Although no connectionist to my knowledge has specifically disavowed sequentiality, the possibility that Parallel Distributed Processing might ultimately obviate the need for sequential mechanisms seems implicit in many discussions of the appeal of connectionism, such as an early claim that ''parallel distributed processing models offer alternatives to serial models of the microstructure of cognition'' (McClelland, Rumelhart, & Hinton, 1986, p. 12).

[14] Not all purported ''hybrid'' models are really hybrids between symbol-manipulating mechanisms and non-symbol-manipulating models. Some putative hybrids are really pure sym-

past tense is of this sort (Marcus, 1996; Marcus et al., 1995; Pinker, 1991; Pinker & Prince, 1988). Irregular verbs (*sing-sang*) are inflected by pattern-associator while regular verbs are inflected by a rule (concatenate the *-ed* morpheme to any instantiation of the variable verb stem). Since such a model includes both symbol-manipulating machinery and a pattern associator that does not appear to depend on computations over variables and a rule, it would not *eliminate* symbol-manipulation; rather the pattern associator would be used to *supplement* a symbol-manipulating system.

None of which is to say that it is impossible to build some connectionist model that could generalize outside the training space. Models such as those of Shastri and Ajjanagadde (1993), Hummel and Holyoak (1997), Holyoak (1997), and Halford et al. (1997) implement explicit variables and explicit relationships between those variables, and hence can readily generalize outside the training space.[15] (For one demonstration of a connectionist model that implements variable-manipulation and solves the identity task, see Holyoak and Hummel, in press.)

7.2. Alternative Learning Algorithms

Most variants on back-propagation retain the limitations of training independence, as do other popular learning algorithms such as the Hebbian rule. One could change the learning algorithm such that units were not trained independently. Recall, though, McClelland and Rumelhart's claim that what distinguishes their models is that the ''learning rules are completely local.'' Algorithms that treat sets of nodes globally as a group would abandon the core assumption of eliminative connectionism in favor of models that might well be implementations of systems that manipulate variables.

7.3. Alternative Ways to Set Up the Problem Space

Another possibility is to set up the problem in a different way.

7.3.1. Distributed representations. The most obvious way of changing the nature of the problem would be to use distributed representations. Recall that distributed representations are representations in which a given input item is encoded as a collection of input features; a word, for instance, could be represented by some set of semantic or phonological features (Hinton, McClelland, & Rumelhart, 1986).

The way that distributed representations might in principle help is that they might force all possible inputs into the same space, such that a novel input would be guaranteed to appear within the training space. The idea is

bol-manipulating models in which some but not all components are implemented at a connectionist level. Here the term hybrid really refers to a hybrid level of analysis rather than a hybrid cognitive architecture.

[15] For a critique of the tensor representation of variables used by Halford et al. (1997), see Holyoak and Hummel (in press).

that if you choose the right distributed representation, no input will be entirely new. Unfortunately, this approach cannot work as a general solution, for two reasons.

First, distributed representations could work as general solution only if the representations of all possible novel items only contained features that appear in words that appeared in the training set. In fact, realistic novel items would often contain untrained features. For instance, in the balance beam model, the input representations already are distributed (each input stimulus entails activating 4 nodes), yet there will always be some novel input that does not overlap in feature space with the inputs that appeared in training. If the model is trained on problems containing no more than n weights per peg, a problem containing $n + 1$ weights on a peg will lie outside the training space.

Second, models that use many types of distributed representations cannot distinctly represent combinations of possible items, a problem that von der Malsburg (1981) dubbed the ''superposition catastrophe'' (see also Hummel & Holyoak, 1993; Hinton et al., 1986). If models like the sentence-prediction model and the family-tree model were modified to use distributed output representations, they could not do the very tasks for which they were intended. For example, the goal of Elman's model is to activate all and only the plausible continuations to a given sentence fragment; encoded words as distributed representations would interfere with this goal. For example, suppose that words were encoded by a set of orthographic features, such as **letter ''a'' in the first position of a word**, **letter ''a'' in the second position**, and so forth. Because virtually any letter can appear in either a noun or a verb, activating the set of all features that appear in nouns would be tantamount to activating the set of features that can appear in verbs. In short, it would be impossible to activate the nouns without simultaneously activating the verbs.

Likewise, in any reasonable semantic encoding scheme, there would be some overlap between the features of nouns and the features of verbs, so activating all the nouns would lead to activating inadvertently at least some of the verbs. Indeed, the only way around this is to presuppose the categories noun and verb, by having input representations that contain some features that would appear only in nouns and other features that would be unique to verbs—thereby presupposing the very categories that the model is supposed to acquire.[16]

Indeed, while the published versions of the simple recurrent network all

[16] An alternative is encoding schemes in which distributed features are correlated (albeit imperfectly) with grammatical category. For example, if some input feature like ''animate'' occurs with 75% of nouns and 25% of verbs, in positions in which a noun should occur, the network tends to activate that feature to a level of 0.75. This is, to be sure, a correct reflection of the conditional probability of that feature appearing in that position. But because some verbs carry that feature, and because some nouns do not carry it, the network's response once again cannot separate the nouns from the verbs.

encode words locally, an earlier, unpublished version of the SRN (Elman, 1988) encoded words with distributed representations; given the superposition catastrophe, it is unsurprising that this ''network's performance at the end of training . . . was not very good.'' After five passes through 10,000 sentences, ''the network was still making many mistakes'' (p. 17).[17] In short, the simple recurrent network cannot be used to predict sequences of word classes drawn from a grammar if the representations of words from different classes overlap.

A similar problem of superposition would affect a version of Hinton's family-tree model that used distributed output representations, because many agent-relationship combinations must allow more than one filler for the patient role. For example, given a query such as ''Penny is the mother of X,'' the response should be ''Arthur AND Victoria.'' In the localist output version of the family-tree model (i.e., the one Hinton actually proposed), this problem is solved naturally: the model simply activates simultaneously both the *Arthur* node and the *Victoria* node. In a distributed output model, this becomes more difficult. To take a concrete example, imagine that Arthur is encoded by activating only input nodes 1 and 2, Victoria by nodes 3 and 4, Penny by nodes 1 and 3, and Mike by nodes 2 and 4. To indicate that Arthur and Victoria were both sons of Penny, this distributed input/output version of Hinton's model would need to activate the nodes 1, 2, 3, and 4: exactly the same set of nodes as it would have used to activate Penny and Mike, another example of the superposition catastrophe.

Mixed representations in which some features are localist identifiers of individuals (+**John,** +**Peter,** +**Mary**) and in which other features form distributed representations (+**young,** +**male,** +**red-haired**) do not solve these problems. In the family-tree model, the localist identifiers of individuals would still lie outside the training space and hence not be generalized to,

[17] Although the network's performance was poor on the task of predicting new words, groups of words from common categories (e.g., transitive verbs) elicited common patterns of hidden unit activation. Elman argued that these common patterns of hidden unit activation show that ''the network has discovered that there are several major categories of words. One large category corresponds to *verbs;* another category corresponds to *nouns.*'' In fact, these common patterns of hidden unit activation do not genuinely represent categories such as noun or verb. In the randomly generated training grammar, words of a common category appear in essentially the same set of structures. In real language, words are not distributed uniformly in this way; some nouns may be more likely to occur in one structure than another. Moreover, we can accord nouns that have appeared in only one sentence structure the same grammatical privileges as a word that appeared in many structures (e.g., if we read the sentence the *gerenuk eluded the lion,* we can infer the grammaticality of the sentence *the lion chased the gerenuk*). In contrast, in the simple recurrent network, a word that appears in a single sentence elicits a very different hidden unit activation pattern than does a word that appears in many different sentences. Thus, rather than reflecting genuine grammatical categories, the patterns of hidden unit activation reflect only the similarity of contexts in which a given set of words appeared.

while the distributed feature would still be subject to the superposition catastrophe. Similarly, in the case of the simple recurrent network, if *rose* were encoded as [+**rose,** +**flower**], and *lilac* were encoded as [+**lilac,** +**flower**], an SRN that was trained *on a rose is a rose* would predict that the continuation of *a lilac is a . . .* would be [+**flower**], rather than the correct [+**flower,** +**lilac**]. As before, whichever features were outside the training space would not be generalized.

Note furthermore that if it were to turn out that eliminative connectionist networks could account for how universals are freely generalized, but only by choosing the representation vector very carefully in a particular problem domain so as to avoid the extra-training-space problem, the answer to the key question, ''What makes the model work?'' would lie in factors that have nothing to do with network architecture. Rather, in such a case, the explanatory power would lie in the choice of innate input representation hardware and the usually hidden pre-processor that delivers the input vector to a model.

7.3.2. Encoding inputs as analog values. While most networks represent inputs by pattern of activation across sets of nodes, in principle one could use a single node to represent all possible inputs, assigning each possible input to some real number. One could then implement the identity function by connecting a single input node with a connection weight of 1.0 to an output node with a linear activation function. Such an approach would implement the identity function, but only by incorporating what is a transparent implementation of a register and a copy instruction. The node in question would represent a variable; its value would represent the instantiation of that variable. The operation of copying would be implemented by forcing the activation of the output node to equal the activation of the input node. Such a model makes the same predictions and has the same intermediate state as a symbol-manipulating operation that copies the contents of one register into another register, hence it would not provide an alternative to symbol-manipulation.

8. GENERAL DISCUSSION

This paper has presented the following argument:

- Humans can generalize a wide range of universals to arbitrary novel instances. They appear to do so in many areas of language (including syntax, morphology, and discourse) and thought (including transitive inference, entailments, and class-inclusion relationships).
- Advocates of symbol-manipulation assume that the mind instantiates symbol-manipulating mechanisms including symbols, categories, and variables, and mechanisms for assigning instances to categories and representing and extending relationships between variables. This account

provides a straightforward framework for understanding how universals are extended to arbitrary novel instances.

- Eliminative connectionism proposes that the mind does not instantiate the mechanisms of symbol-manipulation.
- Current eliminative connectionist models map input vectors to output vectors using the back-propagation algorithm (or one of its variants).
- To generalize universals to arbitrary novel instances, these models would need to generalize outside the training space.
- These models cannot generalize outside the training space.
- Therefore, current eliminative connectionist models cannot account for those cognitive phenomena that involve universals that can be freely extended to arbitrary cases.

The continued plausibility of the eliminative connectionist account thus rests on discovering an alternative that can generalize these universals without (perhaps covertly) implementing the very symbol-manipulation accounts that eliminative connectionism aims to eliminate. At least for now, eliminative connectionism does not offer a viable alternative account of how humans extend universals, whether they be universals in language or in cognition. In contrast, the framework of symbol-manipulation can provide an account of how humans generalize universals.

This is not, of course, a claim that *all* aspects of cognition are computed by the manipulation of symbols. Rather, the claim is only that some pervasive aspects of cognition (e.g., the generalization of universals) do depend on symbol-manipulation. It seems likely that there are other aspects of cognition (e.g., aspects of motor control, memory, and skill-learning) that do not involve symbol-manipulation; feedforward networks and simple recurrent networks might provide an adequate account of those domains.

Since the problems of extrapolation that are identified in this paper are so straightforward and so pervasive, it is surprising that they have not been emphasized earlier. This may be because advocates of eliminative connectionism tend to focus on what their models can explain, while ignoring the limitations of their models. For example, reports of eliminative connectionist models by advocates of eliminative connectionism typically report the abilities of models to generalize within the training space, but rarely if ever report tests of the abilities of those models to generalize outside the training space. Similarly, advocates of eliminative connectionism have often attempted to rebut criticism by noting that multilayer feedforward networks represent a substantial advance over earlier two-layer networks. To be sure, multilayer feedforward networks can represent some functions that are altogether unrepresentable in two-layer networks, but hidden layers are not a panacea. Instead, what I have shown is that multilayer feedforward networks inherit—unchanged—the Achilles heel of their two-layer predecessors (perceptrons), described over 35 years ago by Rosenblatt (1962):

> In a simple perceptron, patterns are recognized before ''relations''; indeed, abstract relations, such as ''A above B'' or ''the triangle is inside the circle'' are never abstracted as such, but can only be acquired by means of a sort of exhaustive rote-learning procedure, in which every case in which the relation holds is taught to the perceptron individually. (p. 73)

The argument presented in this article is not a tautology—it is not a claim that variables can be handled only by systems that can handle variables. Every function that has been described in this paper could be *represented* in some connectionist network. Rather, the problem is that, given realistic, finite input, current eliminative connectionist models lack sufficient innate structure to *learn* these functions. The argument is that universals can only be extended by a system that is innately endowed with machinery that instantiates the machinery of symbol-manipulation. In the simulations reported here, the linking relationships between variables were never violated, but the feedforward/simple recurrent network architecture, whatever its other virtues as a general-purpose problem solver, could not extract a regular relationship that held between variables. Because of intrinsic limitations, the back-propagation algorithm can detect correlations between features, but not between variables; no matter what the structure of the environment is, the innate structure of the back-propagation algorithm guarantees that it can never bootstrap its way into detecting correlations between variables. The limitations of back-propagation do not by themselves rule out the possibility that some other algorithm might discern correlations between variables, but there is no evidence that there is way of constructing such an algorithm that would not itself presuppose variables.

Although the arguments presented in this paper undermine current versions of eliminative connectionism, connectionism itself should not be abandoned. Rather, my hope is that this paper will renew interest in *implementational* connectionism. If the basic entities presupposed by symbol-manipulation are essential to language and cognition, it is an important project to discover how the machinery of symbol manipulation could be implemented in the brain. The field of connectionism might well play a crucial role in that project.

REFERENCES

Abler, W. L. (1989). On the particulate principle of self-diversifying systems. *Journal of Social and Biological Structures,* **12,** 1–13.

Anderson, J. R. (1983). *The architecture of cognition.* Cambridge, MA: Harvard Univ. Press.

Barnden, J. A. (1984). On short-term information processing in connectionist theories. *Cognition and Brain Theory,* **7,** 285–328.

Barnden, J. A. (1992). Connectionism, generalization, and propositional attitudes: A catalogue of challenging issues. In J. Dinsmore (Ed.), *The symbolic and connectionist paradigms: Closing the gap* (pp. 149–178). Hillsdale, NJ: Erlbaum.

Barto, A. G. (1992). Reinforcement learning and adaptive critic methods. In D. A. White & D. A. Sofge (Eds.), *Handbook of intelligent control* (pp. 469–491). New York: Van Nostrand–Reinhold.

Berent, I., Everett, D. L., & Shimron, J. (1998). *Do phonological representations specify variables? Evidence from the Obligatory Contour Principle.* Manuscript submitted for publication.

Busemeyer, J., McDaniel, M. A., & Byun, E. (1997). The abstraction of intervening concepts from experience with multiple input-multiple output causal environments. *Cognitive Psychology,* **32,** 1–48.

Chalmers, D. J. (1990). Syntactic transformations on distributed representations. *Connection Science,* **2,** 53–62.

Chomsky, N. (1957). *Syntactic structures.* The Hague: Mouton.

Clark, A., & Thornton, C. (1997). Trading spaces: Computation, representation, and the limits of uninformed learning. *Behavioral and Brain Sciences,* **20,** 57–90.

Crick, F. (1988). *What mad pursuit.* New York: Basic Books.

Crick, F., & Asunama, C. (1986). Certain aspects of anatomy and physiology of the cerebral cortex. In J. L. McClelland, D. E. Rumelhart, & The PDP Research Group (Eds.), *Parallel distributed processing: Explorations in the microstructure of cognition: Vol. 2. Psychological and biological models.* Cambridge, MA: MIT Press.

DeLosh, E. L., Busemeyer, J. R., & McDaniel, M. (1997). Extrapolation: The sine qua non of abstraction in function learning. *Journal of Experimental Psychology: Learning, Memory, and Cognition,* **23,** 968–986.

Denker, J., Schwartz, D., Wittner, B., Solla, S., Howard, R., Jackel, L., & Hopfield, J. (1987). Large automatic learning, rule extraction, and generalization. *Complex Systems,* **1,** 877–892.

Dominey, P. F. (1997). An anatomically structured sensory-motor sequence learning system displays some general linguistic capacities. *Brain and Language,* **59,** 50–75.

Dyer, M. G. (1995). Connectionist natural language processing: A status report. In R. Sun & L. A. Bookman (Eds.), *Computational architectures integrating neural and symbolic processes.* Dordrecht: Kluwer Academic.

Elman, J. L. (1988). *Finding structure in time* (CRL Tech. Rep. No. 8801). La Jolla, CA: UCSD.

Elman, J. L. (1990). Finding structure in time. *Cognitive Science,* **14,** 179–211.

Elman, J. L. (1991). Distributed representations, simple recurrent networks, and grammatical structure. *Machine Learning,* **7,** 195–224.

Elman, J. L. (1993). Learning and development in neural networks: The importance of starting small. *Cognition,* **48,** 71–99.

Fahlman, S. E. (1979). *NETL: A system for representing and using real-word knowledge.* Cambridge, MA: MIT Press.

Fahlman, S. E., & Lebiere, C. (1990). The cascade-correlation learning architecture. In D. S. Touretzky (Ed.), *Advances in neural information processing systems 2* (pp. 38–51). Los Angeles: Morgan Kaufmann.

Feldman, J. A., & Ballard, D. H. (1982). Connectionist models and their properties. *Cognitive Science,* **6,** 205–254.

Fodor, J. A. (1975). *The language of thought.* New York: Crowell.

Fodor, J. A., & Pylyshyn, Z. (1988). Connectionism and cognitive architecture: A critical analysis. *Cognition,* **28,** 3–71.

Garson, J. W. (1993). Must we solve the binding problem in neural hardware? *Behavioral and Brain Sciences,* **16,** 459–460.

Geman, S., Bienstock, E., & Doursat, R. (1992). Neural networks and the bias/variance dilemma. *Neural Computation,* **4,** 1–58.

Ghahramani, Z., Wolpert, D. M., & Jordan, M. I. (1996). Generalization to local remappings of the visuomotor coordinate transformation. *Journal of Neuroscience,* **16,** 7085–7096.

Goodman, N. (1955). *Fact, fiction, and forecast.* Indianapolis: Bobbs-Merrill.

Hadley, R. F. (1994). Systematicity in connectionist language learning. *Mind & Language,* **9,** 247–272.

Halford, G. S., Wilson, W. H., Gray, B., & Phillips, S. (1997). *A neural net model for mapping hierarchically structured analogs.* Paper presented at the Proceedings of the Fourth Conference of the Australasian Cognitive Science Society, Newcastle, Australia.

Hare, M., Elman, J., & Daugherty, K. (1995). Default generalisation in connectionist networks. *Language and Cognitive Processes,* **10,** 601–630.

Hinton, G. E. (1986). Learning distributed representations of concepts. In *Proceedings of the Eighth Annual Conference of the Cognitive Science Society* (pp. 1–12). Hillsdale, NJ: Erlbaum.

Hinton, G. E. (Ed.). (1990). *Connectionist symbol processing.* Cambridge: MIT Press.

Hinton, G. E., McClelland, J. L., & Rumelhart, D. E. (1986). Distributed representations. In D. E. Rumelhart, J. L. McClelland, & The PDP Research Group (Eds.), *Parallel distributed processing: Explorations in the microstructure of cognition: Vol. 1. Foundations.* Cambridge, MA: MIT Press.

Holyoak, K. (1991). Symbolic connectionism. In K. A. Ericsson & J. Smith (Eds.), *Toward a general theory of expertise.* Cambridge: Cambridge Univ. Press.

Holyoak, K. J., & Hummel, J. E. (in press). The proper treatment of symbols in a connectionist architecture. In E. Deitrich & A. Markman (Eds.), *Cognitive dynamics: Conceptual change in humans and machines.* Cambridge, MA: MIT Press.

Hong, F. S., & Marcus, G. F. (1996). *Neural networks and finite state grammars.* Unpublished manuscript, University of Massachusetts.

Hummel, J. E., & Biederman, I. (1992). Dynamic binding in a neural network for shape recognition. *Psychological Review,* **99,** 480–517.

Hummel, J. E., & Holyoak, K. J. (1993). Distributing structure over time. *Brain and Behavioral Sciences,* **16,** 464.

Hummel, J. E., & Holyoak, K. J. (1997). Distributed representations of structure: A theory of analogical access and mapping. *Psychological Review,* **104,** 427–466.

Jackendoff, R. (1983). *Semantics and cognition.* Cambridge, MA: MIT Press.

Jacobs, R. A., Jordan, M. a. I., & Barto, A. G. (1991). Task decomposition through competition in a modular connectionist architecture: The what and where vision tasks. *Cognitive Science,* **15,** 219–250.

Kirsh, D. (1987). Putting a price on cognition. *The Southern Journal of Philosophy,* **26** (Suppl.), 119–135. (Reprinted, 1991, in T. Horgan & J. Tienson (Eds.), *Connectionism and the philosophy of mind.* Dordrecht: Kluwer).

Kohonen, T. (1984). *Self-organization and associative memory.* Berlin: Springer-Verlag.

Lachter, J., & Bever, T. G. (1988). The relation between linguistic structure and associative theories of language learning: A constructive critique of some connectionist learning models. *Cognition,* **28,** 195–247.

Lange, T. E., & Dyer, M. G. (1996). Parallel reasoning in structured connectionist networks: Signatures versus temporal synchrony. *Behavioral and Brain Sciences,* **19,** 328–331.

Marcus, G. F. (1996). Why do children say ''breaked''? *Current Directions in Psychological Science,* **5,** 81–85.

Marcus, G. F. (1999). *The algebraic mind: Integrating connectionism and cognitive science.* Cambridge, MA: MIT Press.

Marcus, G. F., Brinkmann, U., Clahsen, H., Wiese, R., & Pinker, S. (1995). German inflection: The exception that proves the rule. *Cognitive Psychology,* **29,** 186–256.

Marcus, G. F., Vijayan, S., Bandi Rao, S., & Vishton, P. M. (in press). *7-month-old infants can learn rules. Science.*

Marr, D. (1982). *Vision.* San Francisco: Freeman.

McClelland, J. L. (1989). Parallel distributed processing: Implications for cognition and development. In R. G. M. Morris (Ed.), *Parallel distributed processing: Implications for psychology and neurobiology* (pp. 9–45). Oxford: Oxford Univ. Press.

McClelland, J. L., & Rumelhart, D. E. (1986). A distributed model of human learning and memory. In J. L. McClelland, D. E. Rumelhart, & The PDP Research Group (Eds.), *Parallel distributed processing: Explorations in the microstructure of cognition: Vol. 2. Psychological and biological models* (pp. 170–215). Cambridge, MA: MIT Press.

McClelland, J. L., Rumelhart, D. E., & Hinton, G. E. (1986). The appeal of parallel distributed processing. In J. L. McClelland, D. E. Rumelhart, & The PDP Research Group (Eds.), *Parallel distributed processing: Explorations in the microstructure of cognition: Vol. 1. Foundations* (pp. 365–422). Cambridge, MA: MIT Press.

McCloskey, M. (1991). Networks and theories: The place of connectionism in cognitive science. *Psychological Science,* **2,** 387–395.

McCloskey, M., & Cohen, N. J. (1989). Catastrophic interference in connectionist networks: The sequential learning problem. In G. H. Bower (Ed.), *The Psychology of learning and motivation: Advances in research and theory* (pp. 109–165). San Diego: Academic Press.

Miller, G., & Chomsky, N. A. (1963). Finitary models of language users. In R. D. Luce, R. R. Bush, & E. Galanter (Eds.), *Handbook of mathematical psychology* (Vol. II). New York: Wiley.

Minsky, M. L., & Papert, S. A. (1969). *Perceptions.* Cambridge, MA: MIT Press.

Mozer, M. C. (1991). *The perception of multiple objects: A connectionist approach.* Cambridge, MA: MIT Press.

Newell, A. (1980). Physical symbol systems. *Cognitive Science,* **4,** 135–183.

Newell, A. (1990). *Unified theories of cognition.* Cambridge, MA: Harvard Univ. Press.

Newell, A., & Simon, H. A. (1975). Computer science as empirical inquiry: Symbols and search. *Communications of the Association for Computing Machinery,* **19,** 113–136.

Niklasson, L. F., & Gelder, T. V. (1994). On being systematically connectionist. *Mind & Language,* **9,** 288–302.

Pavel, M., Gluck, M. A., & Henkle, V. (1988). Generalization by humans and multi-layer adaptive networks. *Proceedings of the Cognitive Science Society* (pp. 680–687). Hillsdale, NJ: Erlbaum.

Pazzani, M., & Dyer, M. (1987). A comparison of concept identification in human learning and network learning with the generalized delta rule. *Proceedings of the Tenth International Joint Conference on Artificial Intelligence* (pp. 147–150). Los Altos, CA: Kaufmann.

Pendlebury, D. A. (1996). Which psychology papers, places, and people have made a mark. *APS Observer,* **9,** 14–18.

Phillips, S. (1994). *Connectionism and systematicity.* Paper presented at the Proceedings of the Fifth Australian Conference on Neural Networks, Brisbane, Australia.

Phillips, S., & Halford, G. S. (1997). *Systematicity: Psychological evidence with connectionist implications.* Paper presented at the Proceedings of the Nineteenth Annual Conference of the Cognitive Science Society, Stanford University.

Pinker, S. (1991). Rules of language. *Science,* **253,** 530–555.

Pinker, S., & Prince, A. (1988). On language and connectionism: Analysis of a parallel distributed processing model of language acquisition. *Cognition,* **28,** 73–193.

Plaut, D. C., McClelland, J. L., Seidenberg, M. S., & Patterson, K. E. (1996). Understanding normal and impaired word reading: Computational principles in quasi-regular domains. *Psychological Review,* **103,** 56–115.

Pollack, J. B. (1990). Recursive distributed representations. *Artificial Intelligence,* **46,** 77–105.

Prasada, S., & Pinker, S. (1993). Similarity-based and rule-based generalizations in inflectional morphology. *Language and Cognitive Processes,* **8,** 1–56.

Pylyshyn, Z. (1984). *Computation and cognition: Toward a foundation for cognitive science.* Cambridge, MA: MIT Press.

Pylyshyn, Z. (1986). Cognitive science and the study of cognition and language. In E. C. Schwab & H. C. Nusbaum (Eds.), *Pattern recognition by humans and machines* (pp. 295–314). Orlando: Academic Press.

Rosenberg, C. R., & Sejnowski, T. J. (1987). Parallel networks that learn to pronounce English text. *Complex Systems,* **1,** 145–168.

Rosenblatt, F. (1962). *Principles of neurodynamics.* New York: Spartan.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986a). Learning internal representations by error propagation. In J. L. McClelland, D. E. Rumelhart, & The PDP Research Group (Eds.), *Parallel distributed processing: Explorations in the microstructure of cognition: Vol. 1. Foundations.* Cambridge, MA: MIT Press.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986b). Learning representations by back-propagating errors. *Nature,* **323,** 533–536.

Rumelhart, D. E., & McClelland, J. L. (1986). On learning the past tenses of English verbs. In J. L. McClelland, D. E. Rumelhart, & The PDP Research Group (Eds.), *Parallel distributed processing: Explorations in the microstructure of cognition: Vol. 2. Psychological and biological models.* Cambridge, MA: MIT Press.

Seidenberg, M. S., & McClelland, J. L. (1989). A distributed, developmental model of word recognition and naming. *Psychological Review,* **96,** 523–568.

Shastri, L., & Ajjanagadde, V. (1993). From simple associations to systematic reasoning: A connectionist representation of rules, variables, and dynamic bindings using temporal synchrony. *Behavioral and Brain Sciences,* **16,** 417–494.

Shultz, T. R., Schmidt, W. C., Buckingham, D., & Mareschal, D. (1995). Modeling cognitive development with a generative connectionist algorithm. In T. J. Simon & G. S. Halford (Eds.), *Developing cognitive competence: New approaches to process modeling* (pp. 205–261): Hillsdale, NJ: Erlbaum.

Smith, E. E., Langston, C., & Nisbett, R. E. (1992). The case for rules in reasoning. *Cognitive Science,* **16,** 1–40.

Smolensky, P. (1988). On the proper treatment of connectionism. *Behavioral and Brain Sciences,* **11,** 1–74.

Smolensky, P. (1995). Reply: Constituent structure and explanation in an integrated connectionist/symbolic cognitive architecture. In C. Macdonald & G. Macdonald (Eds.), *Connectionism: Debates on psychological explanation.* Oxford: Blackwell Sci.

Sun, R. (1992). On variable binding in connectionist networks. *Connection Science,* **4,** 93–124.

Tomasello, M., & Olguin, R. (1993). Twenty-three-month-old children have a grammatical category of noun. *Cognitive Development,* **8,** 451–464.

Touretzky, D. S. (1991). Connectionism and compositional semantics. In J. A. Barnden & J. B. Pollack (Eds.), *High-level connectionist models* (pp. 17–31). Hillsdale, NJ: Erlbaum.

Touretzky, D. S., & Hinton, G. E. (1985). Symbols among the neurons. *Proceedings IJCAI-85, Los Angeles.*

Vera, A. H., & Simon, H. A. (1994). Reply to Touretzky and Pomerleau. *Cognitive Science,* **18,** 355–360.

von der Malsburg, C. (1981). *The correlation theory of brain function* (Internal Report No. 81-2). Max-Planck-Insittut for Biophysical Chemistry, Department of Neurobiology.

Accepted August 3, 1998