VIEW Communicated by Gary Cottrell

Generalization in Interactive Networks: The Benefits of Inhibitory Competition and Hebbian Learning

Randall C. O'Reilly
Department of Psychology, University of Colorado at Boulder, Boulder, CO 80309, U.S.A.

Computational models in cognitive neuroscience should ideally use biological properties and powerful computational principles to produce behavior consistent with psychological findings. Error-driven backpropagation is computationally powerful and has proven useful for modeling a range of psychological data but is not biologically plausible. Several approaches to implementing backpropagation in a biologically plausible fashion converge on the idea of using bidirectional activation propagation in interactive networks to convey error signals. This article demonstrates two main points about these error-driven interactive networks: (1) they generalize poorly due to attractor dynamics that interfere with the network's ability to produce novel combinatorial representations systematically in response to novel inputs, and (2) this generalization problem can be remedied by adding two widely used mechanistic principles, inhibitory competition and Hebbian learning, that can be independently motivated for a variety of biological, psychological, and computational reasons. Simulations using the Leabra algorithm, which combines the generalized recirculation (GeneRec), biologically plausible, error-driven learning algorithm with inhibitory competition and Hebbian learning, show that these mechanisms can result in good generalization in interactive networks. These results support the general conclusion that cognitive neuroscience models that incorporate the core mechanistic principles of interactivity, inhibitory competition, and error-driven and Hebbian learning satisfy a wider range of biological, psychological, and computational constraints than models employing a subset of these principles.

1 Introduction

A long-standing goal of neural network modeling is to develop a formalism that is consistent with the known biological properties of the neural networks of the brain, uses powerful computational principles, and produces behavior consistent with findings from psychology. This goal has often been thwarted by incompatibilities among these different levels of constraints. This article examines the ability of networks to process novel information, termed *generalization*, and how this ability is affected by network

properties motivated by attempts to produce a formalism that achieves this long-standing goal. We will see that interactive (bidirectional, recurrent) networks, which are appealing for a variety of biological, psychological, and computational reasons, nevertheless tend to generalize much worse than their feedforward counterparts. However, the inclusion of two other principled mechanisms, inhibitory competition and Hebbian learning (which are similarly appealing for a variety of biological, psychological, and computational reasons), results in good generalization in an interactive network. These results support the use of formalisms that include all of these principles.

1.1 The Importance of Interactivity. Interactive networks provide a number of advantages from the biological, computational, and psychological perspectives. Biologically, bidirectional connectivity is ubiquitous in the cortex (Felleman & Van Essen, 1991; Levitt, Lewis, Yoshioka, & Lund, 1993; White, 1989; Douglas & Martin, 1990). Computationally, interactivity enables iterative, constraint-satisfaction-style processing (Smolensky, 1986; Ackley, Hinton, & Sejnowski, 1985; Hopfield, 1982, 1984), and generally allows information processing to flow in many different directions within one network, providing a richer and more flexible system. Psychologically, interactivity is important for understanding phenomena like the word superiority effect, where higher-level word representations both depend on and yet exert top-down influence over lower-level letter representations (McClelland & Rumelhart, 1981) and other kinds of top-down processing effects (Vecera & O'Reilly, 1998).

Another major motivation for interactive networks is that they enable a more biologically plausible form of error-driven backpropagation learning. Backpropagation learning (Rumelhart, Hinton, & Williams, 1986) provides the starting point for the algorithms discussed in this article and has been used in a wide range of psychological models, but is unfortunately inconsistent with known biological properties (Crick, 1989; Zipser & Andersen, 1988). Specifically, a literal biological interpretation of backpropagation would require that an error derivative term be (1) propagated backward across the synapse and down the axon of the sending neuron, (2) summed and multiplied by a derivative term, and (3) propagated down the dendrites of this neuron and across its synapses, and so on. No evidence for these mechanisms exists.

It was recently shown that backpropagation can be implemented in a more biologically plausible fashion using bidirectional activation propagation in an interactive network using the GeneRec algorithm (O'Reilly, 1996a), which is a generalization of the recirculation algorithm (Hinton & McClelland, 1988). In GeneRec, error information is propagated as two separate terms by standard activation propagation mechanisms in interactive networks, and the difference between these terms (which is the error signal) can be plausibly computed using the synaptic modification mechanisms un-

derlying long-term potentiation and depression (LTP / LTD). Versions of the GeneRec algorithm are equivalent to the other known ways of implementing powerful error-driven learning using interactive activation propagation instead of direct error propagation (e.g., the deterministic Boltzmann machine, Hinton, 1989b; and contrastive Hebbian learning, Movellan, 1990). Thus, several different approaches converge on the idea that the way to perform error-driven learning in a more biologically plausible manner is to use interactive networks, where error signals are communicated by top-down activation propagation.

1.2 Interactivity Impairs Generalization. Despite the numerous advantages of interactivity, it can also impart some significant limitations: it tends to impair the generalization performance of networks. Generalization is important from both computational and psychological perspectives. Computationally, generalization has been a major focus in the study of machine learning, where it enables limited training data to be applied more broadly (Weigend, Rumelhart, & Huberman, 1991; Wolpert, 1992; Vapnick & Chervonenkis, 1971). Psychologically, generalization is important for modeling human nonword reading performance (e.g., the fact that people generally pronounce nonwords like *nust* according to the regularities of the English language; Seidenberg & McClelland, 1989; Plaut, McClelland, Seidenberg, & Patterson, 1996), and more generally for understanding how human and animal cognition can exhibit flexibility in ever-changing environments. Thus, the fact that interactivity impairs generalization, often to a significant degree, is problematic for any attempt to develop a biologically, computationally, and psychologically satisfying neural network algorithm.

To understand how an interactive network can interfere with generalization, one must understand how networks typically generalize. Networks can form distributed internal representations that encode the compositional features of the environment in a combinatorial fashion, which enables novel stimuli to be processed successfully by activating the appropriate novel combination of representational (hidden) units. Although the combination is novel, the constituent features are familiar and have been trained to produce or influence appropriate outputs, such that the novel combination of features should also produce a reasonable result. In the domain of reading, a network can pronounce nonwords correctly when it represents the pronunciation consequences of each letter using different units that can easily be recombined in novel ways for nonwords.

In this combinatorial view of generalization, interactivity impairs generalization because an interactive network is a dynamic system, whereas a feedforward network is not. Under certain conditions, the interactive activation dynamics (e.g., attractors) can interfere with the ability to form novel combinatorial representations; the units interact with each other too much to retain the kind of independence necessary for novel recombination. Consider a very simple example (see Figure 1): train that a red and

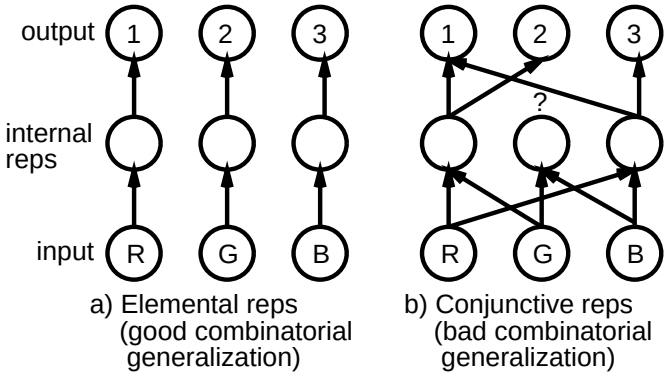

Figure 1: Illustration of why independent (elemental) internal representations are important for good combinatorial generalization. Inputs are different colored lights (red, green, and blue), and outputs are different buttons that could be pressed. Training consists of R,G $\rightarrow$ 1,2 and R,B $\rightarrow$ 1,3, and testing is on G,B, which, combinatorially, should produce 2,3. (a) the internal representations encode the input-output mapping in an independent or elemental fashion, enabling full combinatorial generalization. (b) the representations are conjunctive, meaning that inputs interact or conjoin in determining the internal representations (via attractor dynamics), producing different attractors for each pair of inputs. This can produce bad combinatorial generalization because the weights for the novel G,B attractor have not been associated with anything meaningful during training.

green light means push buttons 1 and 2, and a red and blue light means push buttons 1 and 3. Then test with green and blue lights—buttons 2 and 3 should be pushed. If the training is encoded using independent internal representations for each elemental input-output mapping, then the system will naturally generalize to novel combinations (see Figure 1a). However, if there are interactive attractor dynamics that cause different pairs of inputs to be encoded by very different sets of internal units, then combinatorial generalization will not work, because the green-blue test input will activate a novel internal representation that has not been systematically associated with the correct outputs (see Figure 1b).

In other words, combinatorial generalization suffers to the extent that inputs interact conjunctively in determining the internal representation. The trade-off between independent, elemental representations versus conjunctive or configural representations has been analyzed in the context of hippocampal representations, which are thought to be conjunctive, while cortical representations are thought to be more elemental or independent (Marr,1971; O'Reilly & McClelland, 1994; McClelland, McNaughton, & O'Reilly, 1995; O'Reilly & Rudy, 2000, in press). According to this analysis,

conjunctive representations are important for capturing specifics, while elemental and independent representations are important for capturing generalities of the environment and using these representations for generalization to novel situations.

Thus, as will be demonstrated, generalization suffers when there are conjunctive attractor dynamics, where multiple input features interact and give rise to different internal representations, instead of each feature contributing independently (combinatorially) to the internal representations. Specifically, these conjunctive attractor dynamics can arise when learning and activation dynamics are underconstrained, such that individual units tend to be activated by a large and unsystematic collection of input features, and the network is highly sensitive to small differences between different inputs. In addition to demonstrating a substantial impairment of combinatorial generalization in underconstrained interactive networks, the simulations in this article provide more direct evidence that conjunctive attractor dynamics are the source of the problem.

However, some other reports in the literature have demonstrated successful generalization in interactive networks (e.g., Plaut, & McClelland, 1993; Plaut et al., 1996). Indeed, Plaut and colleagues attributed the generalization success of their networks to their ability to form componential attractors, where attractor dynamics operated within, but not between, the representations of the compositional features of letters and phonemes of words (just like the independent representations of Figure 1a). Thus, the components could be effectively recombined to subserve good generalization, instead of suffering from the problems of conjunctive representations. However, the algorithm that Plaut and colleagues used did not significantly develop the recurrent feedback weights from the output layer to the hidden layer over the course of learning (Plaut, personal communication, 1996), and the networks were specifically constrained to settle very rapidly. These facts minimize the extent to which those networks can be considered interactive; instead, they are effectively feedforward networks that have only token feedback connections, and their generalization performance can thus be attributed to the lack of interactivity rather than to the existence of componential attractors.

Another case showed that recurrent connections actually improved generalization (Harm & Seidenberg, 1999), but here the attractor dynamics were only in the output layer and served to clean up representations to the nearest valid output. The arguments in this article apply instead to fully interactive networks where the input-output mapping itself is mediated by an interactive network via one or more hidden layers. The importance of the distinction between output-only attractors and those developed over hidden layers was clearly demonstrated by Noelle and Cottrell (1996), who found that networks with only local recurrent connectivity at the directly trained output layer generalized well, but networks with recurrent connectivity at the hidden layer that received a distal error signal from an output

layer generalized poorly. More recent work by Noelle and Zimdars (1999) showed improved generalization with distal error signals by incorporating additional biasing mechanisms.

Thus, the idea that generalization performance is impaired in fully interactive networks is consistent with the findings in the literature, as well as the simulations reported below. Therefore, the improvement in biological plausibility imparted by the GeneRec algorithm and variations of it is offset by a decrement in psychological plausibility and computational efficacy, leaving us perhaps not that much closer to the eventual goal of a biologically, psychologically, and computationally satisfying algorithm.

1.3 Biases for Improving Generalization in Interactive Networks. A very different conclusion can be reached if one retains the advantages of interactive networks and augments them with additional biases that favor the development of systematic, truly componential attractor structures. The use of such biases in neural networks has been discussed in the context of the fundamental bias-variance trade-off (Geman, Bienenstock, & Doursat, 1992). This trade-off emphasizes the fact that biases that are appropriate for the task can greatly facilitate learning and generalization by reducing the level of variance, where variance reflects the extent to which parameters are underconstrained by learning, and thus free to vary, causing random errors in generalization. These biases are also known as regularizers (e.g., Poggio & Girosi, 1990). However, inappropriate biases can obviously hurt performance by introducing systematic errors, such that there is no such thing as a single universally beneficial set of biases (Wolpert, 1996a, 1996b). It is nevertheless possible that mammalian cortical learning mechanisms have a set of biases that facilitate learning and generalization in the small subset of all possible environments that characterize the natural world (and what an animal needs to learn about the natural world to survive). An important goal of the work described here is to make progress in identifying such biases.

It is well known that purely error-driven learning mechanisms are weakly biased and are therefore typically underconstrained by the learning task, causing them to suffer from too much variance (Vapnick & Chervonenkis, 1971; Weigend et al., 1991; Geman et al., 1992; Wolpert, 1992). For example, this means that the weights in a trained network tend to reflect a large contribution from their random initial values, which prevents the units from systematically carving up the input-output mapping into separable subsets that can be independently combined for the novel testing items. Instead, each unit participates haphazardly in many different aspects of the mapping. This haphazardness is often not that detrimental to generalization in a purely feedforward network (e.g., as demonstrated in a componential generalization task; Brousse, 1993), but it proves far more damaging in the context of the attractor dynamics in an interactive network.

Therefore, it seems likely that appropriate additional biases could signifi-

cantly increase the network's tendency to produce componential attractors. The specific choices of biases explored in this article are biologically and psychologically motivated, with the hope of producing an algorithm that is at once biologically plausible and capable of effective generalization. Specifically, two important principled mechanisms, inhibitory competition and Hebbian learning, are incorporated as biases in the network. Both of these mechanisms have been widely used for biological and psychological models, but they have not generally been combined with error-driven learning in an interactive network. Biologically, there is good evidence for inhibitory competition in the cortex in the form of the effects of pervasive inhibitory interneurons (Gabbot & Somogyi, 1986), and Hebbian learning characterizes the properties of LTP/LTD in the cortex (Bear, 1996). Psychologically, inhibition is important for modeling attentional and other phenomena, and Hebbian learning has been used in modeling a number of different learning phenomena (Miller, Keller, & Stryker, 1989).

In the context of generalization in interactive networks, the specific biases of inhibitory competition and Hebbian learning constrain both the interactive activation dynamics (in the case of inhibitory competition) and the learned weight patterns such that the network actually produces componential attractors.

The simulations presented in this article, which are implemented using the Leabra algorithm that combines GeneRec with inhibitory competition and Hebbian learning mechanisms (O'Reilly, 1996b, 1998; O'Reilly, & Munakata, 2000; O'Reilly & Rudy, in press), demonstrate that these biases outperform other standard biases such as weight decay on a combinatorial generalization task. Furthermore, these biases have proven useful for a wide range of naturalistic learning environments, as demonstrated by modeling a range of cognitive phenomena in perception, memory, language, and higher-level cognition using the same algorithm, often with the same parameters (O'Reilly & Munakata, 2000). Thus, by incorporating the additional mechanistic principles of inhibitory competition and Hebbian learning together with interactivity in a biologically plausible form of backpropagation via the GeneRec algorithm, it is possible to have a neural network model that satisfies a wider range of biological, psychological, and computational constraints than models using only subsets of these principles.

The article is organized as follows. First, a combinatorial generalization task is introduced, and feedforward backpropagation and GeneRec are compared, demonstrating the generalization problem with interactive networks. Corroborating evidence from recurrent backpropagation is then presented. Then inhibitory competition and Hebbian learning are discussed and the Leabra implementation summarized. Then generalization results for Leabra are presented, with a variety of analyses of its performance, followed by an exploration of a version of the combinatorial generalization task with exceptions and an entirely different generalization task involving handwritten digit recognition.

2 Combinatorial Generalization and Interactivity

The task used to explore combinatorial generalization was constructed with several different desiderata in mind. First, it has a simple combinatorial structure, such that novel inputs can be composed from combinations of a basic vocabulary of features. This combinatorial structure is implemented by having four different input-output slots, where the output mapping for a given slot depends only on the corresponding input pattern for that slot (similar to the tasks studied by Brousse, 1993, and Noelle & Cottrell, 1996, and the example shown in Figure 1). One could think of this as the letters in a word mapping to corresponding phonemes, except that unlike English, this mapping is completely regular (a combination of regular and irregular cases is explored later). Each slot has a vocabulary of input-output mappings. The second desiderata is that there is some interesting substructure to the vocabulary mapping within each slot, which establishes that a slot is a coherent, interdependent collection of units. Specifically, the input vocabulary consists of all 45 combinations of 5 horizontal and 5 vertical bars in a 5×5 grid, and the output mapping is a localist identification of the two input bars (this is similar to the bars tasks used by Földiák, 1990; Saund, 1995; Zemel, 1993; Dayan & Zemel, 1995). The third desiderata is that the structure of the task should be visibly evident in the weight patterns, which is accomplished by the use of the bars, so that it should be easy to examine the trained weight patterns for evidence of having represented the basic elements of the task. The resulting network with an example input-output pattern is shown in Figure 2.

The total possible number of distinct input patterns is approximately 4.1 million (45^4), but the networks are trained on only 100 randomly constructed examples. Thus, to the extent that any significant level of generalization is observed, this task serves as a demonstration of how neural networks can leverage a relatively small amount of experience to produce a huge range of generalized behavior. Five hundred randomly constructed testing patterns (all different from the 100 training patterns) were used to assess generalization performance, with generalization reported as a proportion of testing items with errors. Error was scored using the criterion that each output unit had to be on the right side of .5 according to the correct target pattern (i.e., using a unit-wise tolerance of .5). Ten different randomly initialized networks were run for 500 epochs each, and the results are averages over the best generalization performance (assessed every 25 epochs during training) of these 10 networks.

2.1 Basic Results. To assess the effect of interactivity, two different networks were compared on the combinatorial generalization task: a standard feedforward backpropagation network and an interactive GeneRec network using the symmetric, midpoint variation of the learning rule, which is equivalent to contrastive Hebbian learning (CHL) or a deterministic Boltzmann

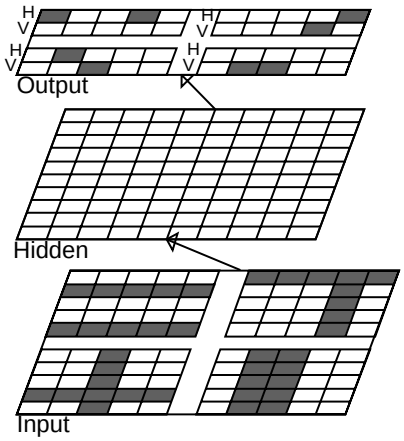

Figure 2: Architecture of the combinatorial generalization network. The input and output are composed of four slots, with the input pattern for each slot being one of 45 different possible combinations of two horizontal and / or vertical bars in the 5×5 slot grid, and the output pattern being a localist identification of each of the two lines (the first row of five output units for each slot representing the vertical lines and the second row representing the horizontal lines). Darkened units show an example input-output pattern.

machine (DBM):

$$\Delta w_{ij} = \epsilon[(x_i^+ y_j^+) - (x_i^- y_j^-)] \tag{2.1}$$

for sending and receiving unit activations x_i and y_j, respectively, in the minus (target unclamped) and plus (target clamped) phases (see O'Reilly, 1996a, for details). A learning rate (ϵ) of .01 was used for both, without any momentum or other factors. The basic generalization results for networks with 100 hidden units are shown in Figure 3, which clearly shows that the interactivity of GeneRec impairs generalization considerably. The interactive GeneRec network makes roughly 90% errors, while the backpropagation network makes only 40% errors.

2.2 Hidden Layer Size. It is generally thought that fewer degrees of freedom should produce more constrained learning and thus better generalization. The results for different numbers of hidden units, shown in Figure 4, indicate that in this task, more hidden units produce better generalization, with the 100 hidden-unit case (shown in Figure 3) producing the best performance (and larger numbers of hidden units produce somewhat better performance but with decreasing gains). This result goes against the standard idea that more hidden units results in overfitting, but overfitting

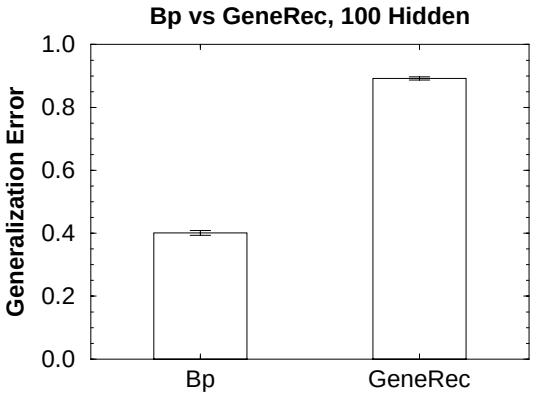

Figure 3: Generalization results (proportion error on the test set) comparing standard feedforward backpropagation (Bp) with interactive GeneRec. GeneRec generalizes very poorly.

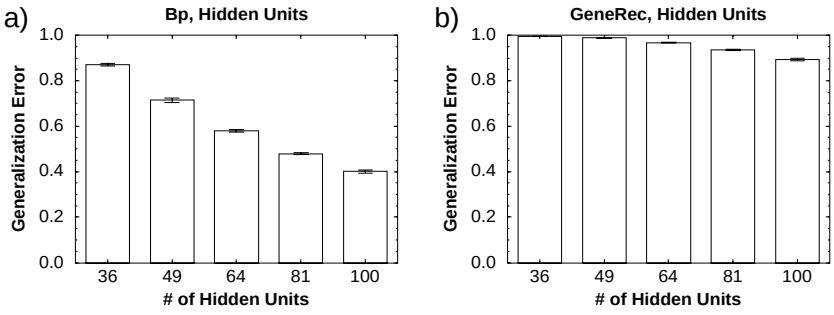

Figure 4: Effect of number of hidden units on generalization performance in (a) standard feedforward backpropagation (Bp) and (b) interactive GeneRec. Larger hidden layers perform better.

may not be as much of a problem here because the task is not noisy; the input-output mapping is completely deterministic. A likely explanation is that more hidden units allow for greater averaging over multiple partially redundant but idiosyncratic hidden units, producing more systematic overall behavior. Similar results were reported by Weigend (1994) and in the version of this task with exceptions discussed later. However, the handwritten digit recognition task has noisy input patterns, and evidence of overfitting with larger hidden layers is observed.

2.3 Weight Decay. A commonly used bias or regularizing function is weight decay (Hinton, 1989a; Weigend et al., 1991). We implemented two

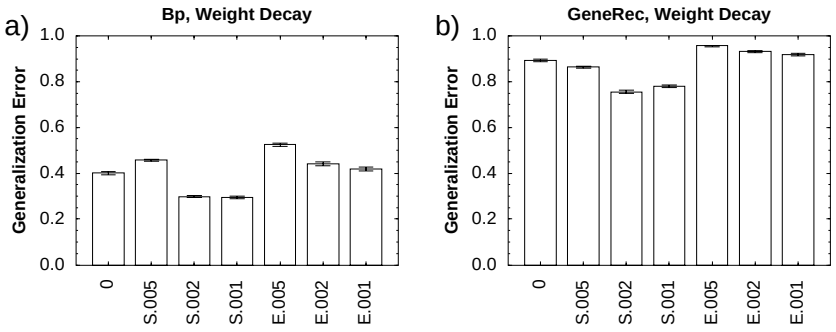

Figure 5: Effect of weight decay on generalization performance in (a) standard feedforward backpropagation (Bp) and (b) interactive GeneRec. First bar is 0 weight decay. Bars labeled S are for the given level of simple weight decay, while E bars are for the given level of weight-elimination weight decay. Simple weight decay at the .002 level produces the best results.

commonly used forms of weight decay in the Bp and GeneRec networks: simple weight decay and weight-elimination weight decay (Weigend et al., 1991). In simple weight decay, a constant fraction of the weight value is subtracted at each weight update, and weight elimination is similar except that the rate of decay is a more complex function of the weight such that larger weights suffer relatively less decay than smaller ones (supporting a prior assumption of a bimodal distribution of weight values—one population of larger weights that are actually useful and another of near-zero weights that are not useful; see Weigend et al., 1991, for details).

The results with these forms of weight decay for the 100-hidden-unit network are shown in Figure 5. Although a small amount (.002; smaller amounts had progressively smaller effects) of simple weight decay appears to improve generalization performance in both Bp and GeneRec reliably, the difference is not substantial. The weight-elimination version of weight decay always appears to impair, rather than improve, performance. Although the specific forms of weight decay explored here were not overly successful in this task, it is possible that other forms might perform better. Nevertheless, most forms of weight decay are problematic from a psychological perspective because they predict a level of forgetting that is likely much stronger than what is observed in humans and animals, given the weight decay levels that produce computational benefits.

2.4 Weight Patterns. An examination of the weights in the trained networks clearly shows why generalization is impaired in the interactive network (see Figure 6). The units have not carved the input-output mapping into separable subsets that can be independently combined for the novel

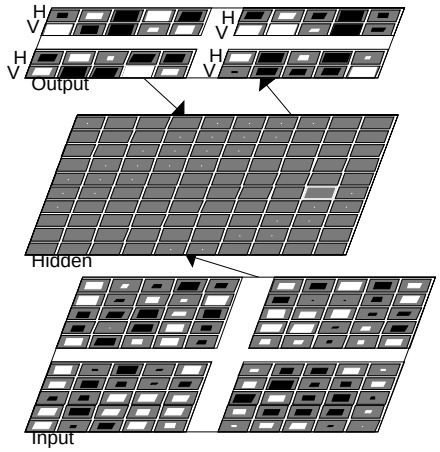

Figure 6: Weights for a typical hidden unit in the GeneRec network after training, with the size of the square indicating magnitude and color indicating sign (white = positive, black = negative). Both feedforward weights from the input layer and feedback weights from the output layer are shown (the GeneRec weights are symmetric so these feedback weights also indicate the nature of the hidden-to-output weights; also note that there are no within-layer connections, as indicated by the zero-value gray squares). It is clear that the units are not dividing up the task into separable mappings for each slot; instead, each unit participates in multiple slot mappings.

testing items; instead, each unit participates in the input-output mapping for multiple slots. Although this is true for both the backpropagation and GeneRec networks, it is particularly damaging for the interactive GeneRec network because of its attractor dynamics. In contrast, the feedforward backpropagation network is still capable of producing a roughly linear combination of hidden unit activations that yields reasonable (though far from perfect) levels of generalization.

2.5 Combinatorial Response Analysis: Direct Evidence for Conjunctive Attractor Dynamics. One way of more directly assessing the extent of conjunctive attractor dynamics in the interactive network is to measure its internal hidden representations in response to input patterns that differ systematically from each other by changes in one or more of the input slots. Because there is no relationship between the slots, the hidden layer should exhibit a combinatorial response as a function of the number of slots that are different. This is what would happen if the representations for each slot were independent of each other (as in Figure 1a). If, however, we see that the hidden layer representations are more different than the input patterns (i.e.,

exhibiting pattern separation; O'Reilly & McClelland, 1994), this suggests that conjunctive activation dynamics are causing the network to overreact to differences in the input. In other words, the hidden units are encoding specific conjunctions of input features across different slots, and therefore very different hidden representations are activated when the input changes (as in Figure 1b).

In short, if we find that the hidden representations in the interactive network are more different than the corresponding differences in input patterns, then this supports the idea that conjunctive attractor dynamics are behind the poor generalization observed in these networks.

To address this important question, two sets of 250 input pattern pairs were generated—one set that differed in only one slot (pairs overlapped by 75%) or by two slots (pairs overlapped by 50%). Thus, to the extent the average pairwise hidden unit overlap falls below 75% and 50% for the first and second sets, respectively, the hidden patterns are exhibiting conjunctivity. We presented these patterns to the Bp and GeneRec networks and computed the normalized dot product (cosine) of the corresponding hidden representations (after settling in GeneRec) for each set of pairs. To control for the baseline activation levels of the units, which would otherwise distort the similarity measure, the average activation values over all the test input patterns were subtracted from the pattern-wise activation values before computing the cosine.

Figure 7 shows the results of this analysis, where it is clear that the interactive GeneRec network is representing the input patterns using consistently more different hidden representations than the feedforward network. It is particularly instructive that the random initial networks exhibit a much more proportional response, as would be expected when the weights are small and the units are in the linear range of the sigmoidal activation function. However, when the GeneRec network's weights get larger, it exhibits a much more nonlinear response, indicative of the complex, conjunctive attractors that have developed over training.

To substantiate further that the conjunctive attractors formed in the GeneRec network are really spurious conjunctions across different slots and not the result of simply activating a trained attractor that was close to the novel input pattern, the output patterns need to be analyzed. If the GeneRec network is indeed producing spurious conjunctive attractors, one would expect that the outputs would be largely uninterpretable noise (e.g., based on the random-looking weight patterns in Figure 6), and not well formed outputs that correspond to trained patterns. To test this, the output activation patterns for each of the four output slots were compared to the entire vocabulary of possible output patterns within each slot. The output was counted as well formed if each of the slots matched (to within .5 unit-wise tolerance) one of these valid output patterns, and was considered malformed otherwise. Note that this analysis ignores whether the actual combination of patterns across slots was trained; it just treats each slot individually (i.e.,

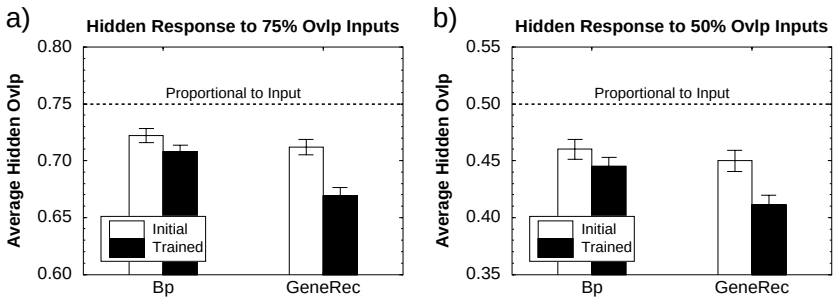

Figure 7: Average pairwise overlap (normalized dot product or cosine) between hidden patterns corresponding to inputs that differ by (a) 75% (one out of four slots different) and (b) 50% (2 out of 4 slots different). Feedforward backpropagation (Bp) remains much closer to a linear response level (75% and 50% hidden similarity, respectively) after training compared to interactive GeneRec, which shows evidence of attractor dynamics pulling the network away from a linear, combinatorial response to the inputs.

allowing any of the approximately 4.1 million possible patterns to match). The number (out of 500 total patterns) of malformed outputs for the 75% and 50% overlap testing patterns is shown in Figure 8. These results, showing that 80% of the GeneRec outputs are malformed compared to only 40% for backprop, clearly support the idea that the interactive attractor dynamics and underconstrained weights in GeneRec lead to spurious, malformed conjunctive attractors.

2.6 Summary. An interactive network that is otherwise identical to a feedforward one generalizes much worse on this combinatorial generalization task. Manipulations of hidden units and weight decay had qualitatively similar effects on both types of networks, such that the interactive network appears to behave just like the feedforward one, but with a greatly elevated offset in generalization errors. This result, together with the analysis of the hidden layer responses and well-formedness of the outputs, clearly supports the idea that the spurious, conjunctive attractor dynamics of the interactive network interfere with its ability to produce systematic combinatorial representations of novel stimuli.

3 Comparison with Recurrent Backpropagation

Although GeneRec has been shown to compute essentially the same error gradients as backpropagation, even in networks with several hidden layers (O'Reilly, 1996a), one might argue that the bad generalization results from GeneRec are somehow an artifact of this particular algorithm. To ad-

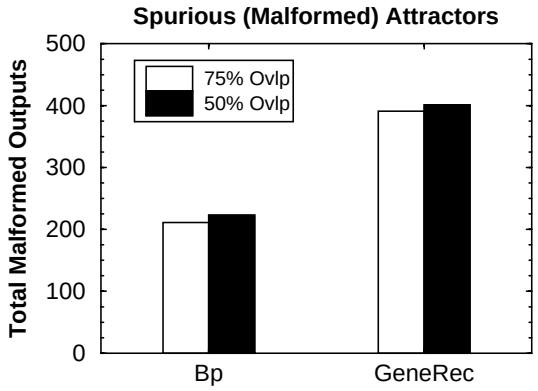

Figure 8: Number of malformed outputs out of 500 total for the 75% and 50% overlap inputs. An output is well-formed if each of the slot outputs corresponds to one of the vocabulary of possible slot outputs, and malformed otherwise. The results support the idea that conjunctive attractor dynamics and underconstrained weights cause GeneRec to produce spurious (malformed) outputs.

dress this concern and provide additional evidence for the idea that interactivity impairs generalization, networks using the Almeida-Pineda (AP) fixed-point recurrent backpropagation algorithm (Almeida, 1987; Pineda, 1987a, 1987b, 1988) were also run. This algorithm is a mathematically direct extension of feedforward error backpropagation to a recurrent attractor network having the same basic architecture as the GeneRec network. To the extent that this AP network also suffers from poor generalization, this substantiates the GeneRec results and shows that they are not artifactual.

One important difference between AP and the CHL version of GeneRec is that GeneRec maintains the weight symmetry between feedforward and feedback weights, whereas AP does not. The net result is that GeneRec develops the feedback weights over the course of learning, whereas AP leaves them virtually untouched. Because it is the magnitude of these feedback weights that largely determines the extent of the interactive activation dynamics in the network, this is an important difference. By adjusting the random weight initialization in the AP network, we can control the average magnitude of the feedback weights to achieve three objectives: (1) show that these weights influence the level of generalization, such that the more interactive networks with stronger feedback weights generalize worse; (2) show that GeneRec's generalization is roughly what would be expected given the magnitude of its feedback weights; and (3) show that the Plaut et al. (1996) generalization results with a recurrent backpropagation network are as expected from a recurrent network with small, undeveloped feedback weights.

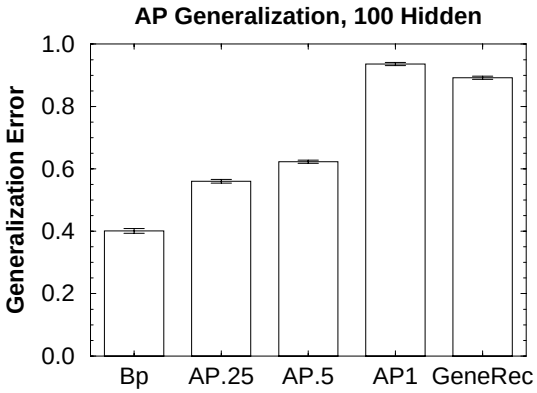

Figure 9: Generalization results for Almeida-Pineda (AP) with various levels of feedback weight variance (.25, .5, and 1.0), as compared with the Bp and GeneRec results shown in Figure 3. As the strength of these feedback weights increases, causing increased attractor dynamics, generalization performance decreases. Figure 10 shows that GeneRec's generalization is commensurate with AP's given the strength of its feedback weights.

Figure 9 shows the generalization performance of three different AP networks having .25, .5, and 1.0 uniform random initial weight variance (centered on 0), resulting in an average feedback weight magnitude (mean of absolute values of all output-to-hidden weights) of roughly .125, .25, and .5, respectively. Clearly, generalization performance is a function of the strength of these weights, with the 1.0 variance case resulting in generalization performance roughly equivalent to that of GeneRec, while the .25 variance case produces generalization performance roughly comparable to that of the feedforward network, as was the case with the Plaut et al. (1996) model.

Figure 10a compares the average feedback weight magnitudes from before and after training with those values from the GeneRec network. Commensurate with their generalization performance, the GeneRec network has slightly smaller feedback weights compared to the 1.0 variance AP network. Thus, GeneRec's generalization performance is as expected based on its level of interactivity as determined by these feedback weights. The correlation between the generalization scores and the trained feedback weight magnitude measure was very strong ($r = .9705$). Also evident from Figure 10 is the fact that GeneRec develops the feedback weights over training much more than AP does.

Another way of measuring the level of interactivity in a network is in terms of settling time—the number of activation update cycles required to achieve a stable activation state (stopping criterion was when maximum

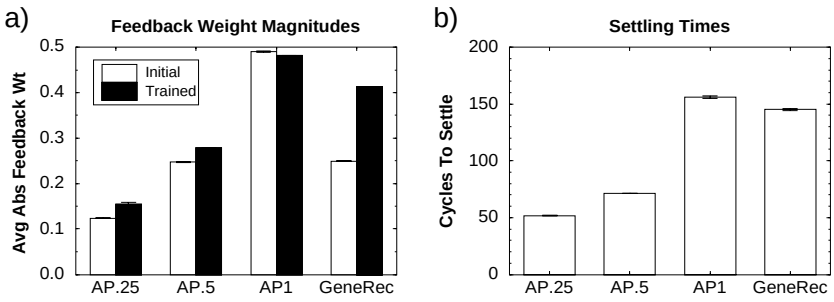

Figure 10: Measures of the level of interactivity in the AP and GeneRec networks: (a) average feedback weight magnitudes before and after training and (b) settling times (average number of cycles to achieve a stable activation state). These measures are strongly correlated ($r = .9705$ and $r = .9998$, respectively) with the generalization performance shown in Figure 9, as well as with each other ($r = .9733$). Also, GeneRec develops the feedback weights much more than AP does.

activation change of any unit in the network in one cycle went below .01). A longer settling time is indicative of a more complex activation trajectory and thus of more extensive attractor dynamics. As shown in Figure 10b, settling time is well correlated with both the strength of the feedback weights ($r = .9733$) and the generalization performance ($r = .9998$). To summarize, the AP results strongly corroborate the conclusion reached previously that the conjunctive attractor dynamics produced by interactive networks directly impair generalization performance.

4 Inhibitory Competition and Hebbian Learning

Although it is possible that any of a number of different biases could be added to the interactive GeneRec network to improve generalization performance, this article focuses on two such biases that have been widely used and can be independently motivated for a variety of biological, psychological, and computational reasons. These biases and their implementation in the Leabra algorithm are discussed in the subsequent sections.

4.1 Inhibitory Competition. Roughly 15 to 20% of the neurons in the cortex are GABAergic inhibitory interneurons (White, 1989; Gabbot & Somogyi, 1986). The importance of these neurons for controlling the positive feedback loops between excitatory cortical pyramidal neurons is revealed for example in the epileptic-like effects of GABA antagonists (Grinvald, Frostig, & Lieke, 1988). Thus, any realistic model of the cortex should include a role for inhibitory competition. Many neural network formalisms

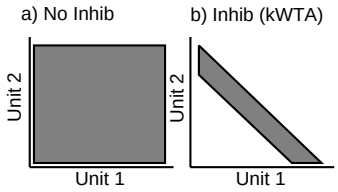

Figure 11: Illustration in two dimensions (units) of how kWTA inhibition can restrict the activation dynamics of an interactive network. (a) A network without inhibition, which can explore all possible combinations of activation states. (b) The effect of inhibition in restricting the activation states that can be explored to those having a roughly constant level of activity (e.g., under the kWTA inhibition function described in the text), leading to faster and more constrained attractor dynamics.

show the computational advantages of inhibitory competition (Grossberg, 1976; Kohonen, 1984; McClelland & Rumelhart, 1981; Rumelhart & Zipser, 1986). One such advantage is that inhibitory competition allows only the most strongly excited representations to prevail, with this selection process identifying the most appropriate representations for subsequent processing. Furthermore, learning mechanisms are affected by this selection process such that only the selected representations are refined over time through learning, resulting in an effective differentiation and distribution of representations.

In the context of the generalization problem, the selection and distribution effects of inhibitory competition can result in individual units specializing on the input-output mapping for separable components (e.g., the slots in the combinatorial problem explored in this article). When the units are specialized in this way, combinatorial representations are facilitated. Furthermore, inhibition has the effect of greatly constraining the activation dynamics of an interactive network, as illustrated in Figure 11. Thus, a network with inhibitory competition should settle more rapidly and with less influence from the kinds of attractor dynamics that can impair generalization.

Another way of thinking about the benefits of inhibitory competition comes from the idea that given the general structure of the natural environment, sparse distributed representations (i.e., with relatively few units active at a time) are particularly useful (Barlow, 1989; Field, 1994). For example, in visual processing, a given object can be defined along a set of feature dimensions (e.g., shape, size, color, texture), with a large number of different values along each dimension (e.g., many different possible shapes, sizes, colors, textures). Assuming that these feature values are encoded individual units (or small groups thereof) in a distributed representation, the overall representation of a given object will activate only a relatively small subset

of the entire set of feature units (i.e., the representations will be sparse). More generally, it seems as though the world can be usefully represented in terms of a large number of categories with a large number of exemplars per category (animals, furniture, trees, etc.). If we again assume that only a relatively few such exemplars are processed at a given time, a bias favoring sparse representations is appropriate. The combinatorial generalization problem has this structure, with only a few of the possible different bar stimuli present at any given time.

To substantiate the argument further in favor of using sparse distributed representations, Olshausen and Field (1996) developed a simulation that showed that imposing a bias for developing this type of representation can result in the development of realistic early visual representations (oriented edge detectors) of natural visual scenes. The close fit between the simulated representations and those in the early visual system suggests that the visual system also uses a sparse distributed representational system.

Although seemingly straightforward, achieving a sparse distributed representation is technically challenging, primarily because this case is difficult to analyze mathematically within a probabilistic learning framework. The problem is one of combinatorial explosion; one needs to take into account all the different possible combinations of active and inactive units to analyze a sparse distributed representation based on true inhibitory competition. Thus, sparse distributed representations fall in a complex intermediate zone between two easily analyzed frameworks (Hinton & Ghahramani, 1997): (1) the winner-take-all (*WTA*) framework (Rumelhart & Zipser, 1986; Grossberg, 1976; Nowlan, 1990), where only one unit is allowed to be active at a time—having a single active unit eliminates the combinatorial problems, but also does not allow for distributed representations; and (2) the independent units framework, where the units are considered to be (conditionally) independent of each other (e.g., a standard backpropagation network)—this allows the combined probability of an activation pattern to be represented as a simple product of the individual unit probabilities (and for distributed representations), but there is no inhibitory competition.

There have been a number of attempts to remedy the limitations of these two analytical frameworks, by introducing distributed representations within a basically WTA framework, or by introducing sparseness constraints within the independent units framework. However, the limitations of these frameworks are difficult to overcome. Basically, any use of WTA prevents the cooperativity and combinatoriality of true distributed representations, and the need to preserve independence among the units in the independent units framework prevents the introduction of any true activation-based competition (see the discussion in O'Reilly, 1998).

Leabra achieves sparse distributed representations with true competition by using a *k*-winners-take-all (kWTA) mechanism, which generalizes the WTA approach to *k* winners (Majani, Erlarson, & Abu-Mostafa, 1989). A kWTA mechanism can enforce true competition among the units, while al-

lowing for a (sparse) distributed representation across the subset of *k* units. kWTA mechanisms have been analyzed for factors such as stability and convergence onto *k* units and can be implemented with biologically plausible lateral inhibition mechanisms (Majani et al., 1989; Fukai & Tanaka, 1997). However, they have not been analytically treated within a probabilistic learning framework, due to the combinatorial explosion problems. Nevertheless, the simple form of kWTA used in Leabra (described in detail in the appendix) works well in bidirectionally connected networks and has been shown to be useful for modeling a wide range of cognitive phenomena (O'Reilly & Munakata, 2000).

4.2 Hebbian Learning. Although Hebbian learning has typically been used as the primary form of learning in a range of models, it is used here as a supplemental form of learning to complement a network learning primarily from error-driven GeneRec. Hebbian learning by itself is simply not powerful enough to enable the learning of even moderately complex input-output mappings and so cannot be considered a self-sufficient learning mechanism (as we will see below, Hebbian learning alone cannot learn the combinatorial generalization task explored in this article). However, there are a number of ways of understanding why the use of Hebbian learning, computed on the plus phase activations to reinforce the correct patterns, is beneficial in the context of error-driven learning.

First, error-driven and Hebbian learning have two complementary learning objectives, which can be referred to as task learning and model learning, respectively. Error-driven learning accomplishes task learning because it is specifically (and exclusively) driven by the demands of the task (i.e., the discrepancies between actual and desired outputs). In contrast, model learning has the objective of learning an internal model of the environment irrespective of specific tasks. From a machine learning perspective, task learning amounts to learning the conditional probability distribution of the output on the input, while model learning amounts to learning the entire (joint) probability distribution (e.g., Rubinstein & Hastie, 1997).[1]. Given that Hebbian and error-driven learning have complementary objectives, combining them makes sense and can generally lead to a more constrained, strongly biased learning algorithm.

At a more detailed level, the benefits and limitations of Hebbian and error-driven learning can be understood in terms of how locally they operate. Hebbian learning is limited in what it can learn because each unit is driven only by the local correlational structure present in its inputs; there is no overall objective function guiding these local weight changes to achieve

[1] As implemented in Hebbian learning, model learning really amounts to representing only the correlational aspects of the environmental structure, not the entire joint probability distribution.

a more distal output pattern somewhere else in the network. In contrast, error-driven learning is ultimately driven by just such a distal error function, which is why it can learn complex problems using intermediate hidden layers.

One of the benefits of combining Hebbian with error-driven learning can thus be understood as enabling learning to take place even when the distal error signals are weak or inconsistent across trials. Furthermore, units in a purely error-driven network are typically underconstrained by the task at hand and are also highly dependent on the responses of other units for determining what they end up representing. This extreme interdependence and lack of local constraint can lead to slow and ineffective learning in large, many-layered networks. To the extent that Hebbian learning provides a useful bias that is applied locally to each unit, it can shape learning locally and apply individual constraints on unit representations, limiting the codependence problems.

From a biological perspective, the combination of error-driven GeneRec learning and Hebbian learning actually provides a better fit to the known properties of synaptic modification (LTP/LTD) than does GeneRec alone (O'Reilly, 1996a). The qualitative signs of GeneRec and Hebbian weight changes are consistent except in two cases: (1) when the sending and receiving units are persistently active in both phases, Hebbian predicts a weight increase (LTP) while GeneRec predicts no weight change; and (2) GeneRec predicts a weight decrease (LTD) when the units are erroneously active (i.e., when the plus-phase activation coproduct is greater than that of the minus phase in the GeneRec learning rule, equation 2.1), while Hebbian predicts no weight change (assuming the the plus phase coproduct is zero). Therefore, the combination of Hebbian and GeneRec is essentially just like Hebbian, except that LTD should occur when the units are erroneously active, meaning that the known Hebbian mechanisms are largely sufficient.

Hebbian synaptic modification appears to operate on the concentration of postsynaptic calcium ions: lower levels of calcium produce LTD, and higher levels produce LTP (Artola, Brocher, & Singer, 1990; Lisman, 1989, 1994; Bear & Malenka, 1994). Under this mechanism, Hebbian LTP results from the calcium influx produced by persistent activation of sending and receiving neurons. This mechanism can also produce the LTD required by GeneRec for erroneously active units. Because erroneous activation means that there is initial activation (in the minus or expectation phase, where the actual network output is produced) followed by inactivation (in the plus or outcome phase, where the target network output is experienced), the initial activation will allow calcium to enter the cell, but the subsequent inactivation prevents the concentration from reaching LTP levels, resulting in LTD.

In the context of the generalization problem, Hebbian learning can facilitate good generalization by encouraging units to represent the strongest aspects of the correlational structure in the input, which are the correla-

tions present in the activations of individual lines within a given slot and in the input-output mapping between lines and line identity units. Thus, Hebbian learning should encourage units to specialize on representing intraslot information, producing a more componential representation that can be recombined for novel inputs to produce good generalization.

The implementation of Hebbian learning in Leabra is essentially the same as in competitive learning (Rumelhart & Zipser, 1986; Nowlan, 1990:

$$\Delta w_{ij} = \epsilon(x_i^+ y_j^+ - y_j^+ w_{ij}) = \epsilon y_j^+ (x_i^+ - w_{ij}) \tag{4.1}$$

for sending unit activation x_i^+ and receiving unit activation y_j^+ (both in the plus phase) with learning rate ϵ. Rumelhart and Zipser (1986) showed that this equation can be interpreted as producing weight values that converge on the conditional probability that the sending unit is active given that the receiving unit is active, $P(x_i = 1|y_i = 1)$, assuming that the activation states reflect the probability that the unit is active (a detailed derivation is provided in the appendix). In the context of a competitive activation function, a given receiving unit is active only when a relatively large number of inputs align with its weights, meaning that this conditional probability learning comes to reflect the strong correlations in the inputs. As explained in detail in the appendix, this Hebbian learning term is combined additively with the GeneRec error-driven learning term at every connection in the network.

One limitation of the Hebbian learning algorithm is that the weights linearly reflect the strength of the conditional probability. This linearity can limit the network's ability to focus on only the strongest correlations, while ignoring weaker ones. In the combinatorial generalization task, for example, any "spurious" correlations between input patterns across different slots will be faithfully encoded by linear-Hebbian units and prevent good generalization. Thus, it is useful to include a further bias on the learning system to encode only the strongest correlations. One way of achieving this goal is to use subtractive normalization instead of the multiplicative normalization used in the competitive learning rule, because subtractive normalization moves the weights toward extrema, while multiplicative normalization retains a linear proportionality to the correlation strength (Miller & MacKay, 1994; Goodhill & Barrow, 1994). However, the completely binary weights produced by subtractive normalization are not likely to be generally useful, especially in more complex tasks and in the context of error-driven learning, where graded weight values are required to balance the contribution of individual units to multiple different aspects of problems.

The approach taken here is to introduce a contrast enhancement function that magnifies the stronger weights and shrinks the smaller ones in a parametric, continuous fashion. This contrast enhancement is achieved by passing the linear weight values computed by the learning rule through a

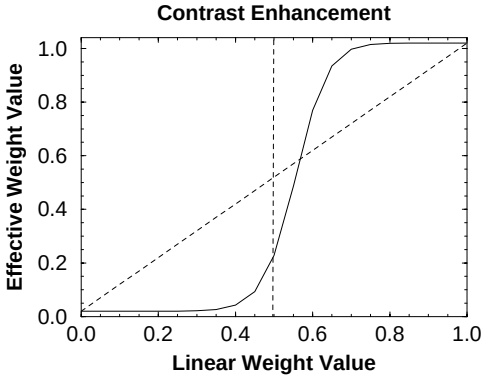

Figure 12: Effective weight value as a function of underlying linear weight value, showing contrast enhancement of correlations around the middle values of the conditional probability as represented by the linear weight value. The gain parameter, $\gamma = 6$, controls the sharpness of the sigmoid, and the midpoint of the function is shifted upward by the offset parameter $\theta = 1.25$.

sigmoidal nonlinearity of the following form (see Figure 12):

$$\hat{w}_{ij} = \frac{1}{1 + \left(\theta \frac{w_{ij}}{1 - w_{ij}}\right)^{-\gamma}}, \tag{4.2}$$

where $\hat{w}_{ij}$ is the contrast-enhanced weight value, and the sigmoidal function is parameterized by an offset θ and a gain γ. The offset (which is multiplicative but moves the middle of the sigmoid up or down in an offset-like fashion) determines where the threshold for the nonlinearity is located, and the gain determines how sharp the nonlinearity is. A gain value of 6 and an offset of 1.25 works well for a wide range of problems (O'Reilly & Munakata, 2000). In the combinatorial generalization problem, a higher threshold (i.e., a higher θ) for correlations to be reflected in the weights should produce even better isolation of the slot representations in the hidden layer; indeed, a threshold of 1.5 worked better than 1.25, and was used for the results reported here.

5 Generalization with Inhibitory Competition and Hebbian Learning

To test the importance of inhibitory competition and Hebbian learning for improving the generalization performance of an interactive error-driven network (GeneRec), a Leabra network was run on the combinatorial generalization task. The network had 100 hidden units with the kWTA k parameter set to 25 units active (25% activity is the default), the k was set to 8 for

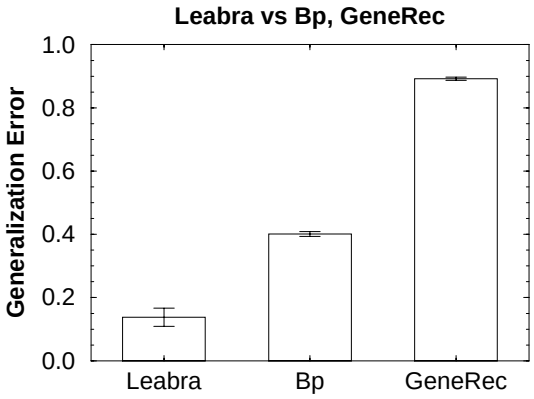

Figure 13: Generalization results comparing Leabra (GeneRec plus inhibitory competition and Hebbian learning) with standard feedforward backpropagation (Bp) and interactive GeneRec. The additional biases in Leabra result in much better generalization than even feedforward backpropagation.

the output layer, and the learning rate was .01 (see the appendix for other standard parameters and equations). The results clearly indicate that these biases greatly facilitate generalization in an interactive network (see Figure 13, specifically the comparison between the two interactive networks using exactly the same error-driven learning rule, GeneRec and Leabra). In the best case of the 10 Leabra networks, generalization error was only .008. If we extrapolate this error on the 500 test cases to the entire space of test items, the network has achieved nearly perfect generalization to over 4 million different patterns based on learning only 100. The average generalization performance corresponds in the extrapolation to getting over 3.5 million problems correct based on the 100 training patterns.

One indication that Hebbian learning is specifically important for the generalization performance of Leabra comes from the time course of generalization over training (see Figure 14). Although the network learns the task after only 10 epochs or fewer of training, generalization performance does not start to improve significantly until roughly 200 or more epochs. In contrast, the backpropagation network starts to generalize much earlier, and this generalization generally parallels learning performance on the task. Bp takes roughly 50 epochs to learn the task, during which time most of the generalization is achieved.

An obvious way to determine the importance of Hebbian learning is to run a Leabra network without it; this is then just a GeneRec network with inhibitory competition. Such a network performs quite badly on this task, having an average generalization error of .992 ($\pm$.00109 standard error of the mean[SEM]). Thus, it is clear that Hebbian learning is essential.

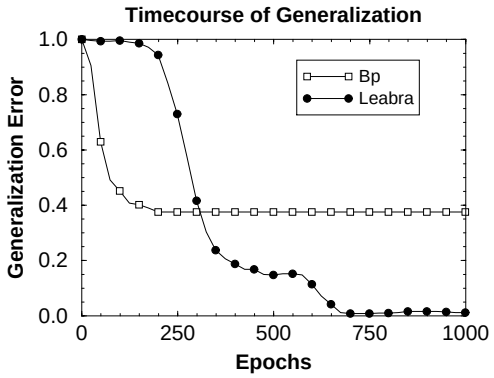

Figure 14: Time course of generalization in Leabra (best generalizing network) and Bp, showing that generalization performance in Leabra starts to improve well only after the task has been learned by error-driven learning (which occurs within the first 10 epochs). This identifies Hebbian learning as specifically important. In contrast, generalization in Bp begins much more rapidly, and more closely parallels the time course of learning (Bp takes roughly 50 epochs to learn the task).

However, Hebbian learning cannot even begin to learn this task on its own, as a run of Leabra with only Hebbian learning (together with the kWTA inhibitory competition) demonstrates (see Figure 15). Note that Hebbian learning without inhibitory competition does noxt typically work very well, because the positive feedback nature of Hebbian learning requires the selection and differentiation pressure of inhibitory competition. Thus, although it is insufficient on its own in this task, inhibitory competition is nevertheless playing an important role. Other tasks have shown that inhibitory learning can make an important contribution on its own (O'Reilly & Munakata, 2000).

An examination of the weights of a trained Leabra network (see Figure 16) shows that individual hidden units are parceling the task up into the logical separable components of individual lines within a given slot. Having represented the task in such a separable fashion, the units in the network can easily be recombined in novel ways to process the novel testing items, thereby achieving good generalization. As a useful simplification, one can understand the interaction between error-driven and Hebbian learning in forming these representations as follows: error-driven learning (together with inhibitory competition) ensures that different units take account of different aspects of the problem and that all aspects are covered such that the problem is actually solved. Hebbian learning operates slowly and refines the hidden unit representations so that they encode only the strongest correlations; because these are the individual line elements, the weights

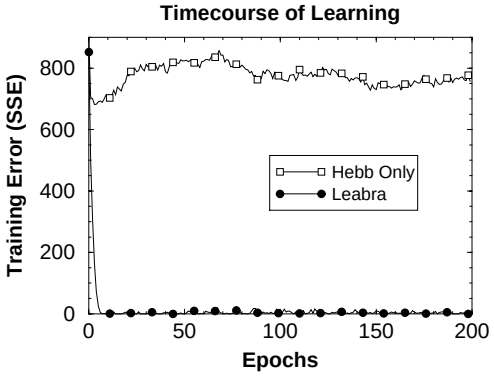

Figure 15: Time course of learning for a version of Leabra with only Hebbian learning (Hebb only), compared to the standard case with both error-driven and Hebbian learning (Leabra). The kWTA inhibitory competition is present in both cases.

come to emphasize these and exclude inputs from other slots, which are more weakly correlated.

Given the kind of combinatorial representations learned by the Leabra network, it is not too surprising that it exhibits an almost exactly proportional response to the 75% and 50% overlapping test items used to analyze the Bp and GeneRec networks previously (see Figure 17). Furthermore, Leabra produces only a small proportion of spurious (malformed) outputs compared to Bp or GeneRec (see Figure 18). Interestingly, the random initial weights in the untrained Leabra network produce a significant amount of pattern separation, which then disappears after training. This is consistent with the idea that sparser levels of activation, with random weights, produce more pattern separation (O'Reilly & McClelland, 1994). Thus, the benefits of inhibitory competition for damping attractor dynamics come at the cost of increasing pattern separation, with random weight patterns. Over learning, the inhibitory competition provides a beneficial pressure for shaping the weights (as discussed previously) and overcomes the initial conjunctivity bias.

6 Generalization with Exceptions

One potential objection to the above results showing an advantage for inhibitory competition and Hebbian learning is that these biases might be so specific to the purely regular combinatorial case that the network would break down in a task that also had irregularities or exceptions to the general rule. The ability to handle both regularities and exceptions is important for dealing with many language phenomena, including the well-studied cases

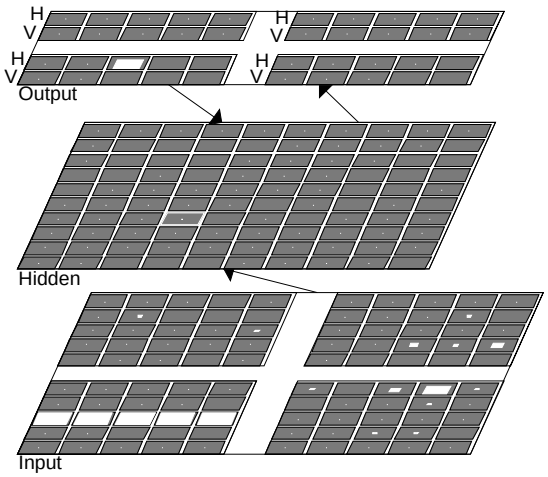

Figure 16: Weights for a typical hidden unit in the Leabra network after training (see Figure 6 for details). Both feedforward weights from the input layer and feedback weights from the output layer are shown. (The Leabra weights are symmetric, so these feedback weights also indicate the nature of the hidden-to-output weights. Also note that there are no within-layer connections, as indicated by the zero-value gray squares; the kWTA inhibition in Leabra is computed directly, not via inhibitory weights.) It is clear that the units are dividing up the task into separable mappings for each slot — each unit participates in mapping a single line within a slot (in this case, the middle horizontal line in the lower left-hand slot) to its corresponding identification output. This division enables combinatorial generalization, explaining the good generalization results. Note that this unit, like many others, also had a weak representation of inputs in other slots, presumably due to relatively strong spurious correlations across slots.

of the English past tense and spelling-to-sound mappings (Rumelhart & McClelland, 1986; Seidenberg & McClelland, 1989; Plaut et al., 1996; Plunkett & Marchman, 1996). We can explore this issue by introducing exceptions into the combinatorial generalization task explored here.

Exceptions were generated for any pattern having a vertical bar in the last (right-most) position in the final (fourth) slot (like the past tense inflectional suffix that appears at the end of a word), with the consequence that the output pattern for the third slot was repeated in the fourth slot, instead of this fourth output slot reflecting the actual inputs from the fourth slot. Thus, these exceptions required there to be interactions between slots, instead of their being completely independent, as in the regular mapping (which was exactly as in the previous task). To equalize the opportunity for generalization compared to the fully regular task, the networks were trained on 100 randomly generated regular cases (as before) plus 27 randomly generated

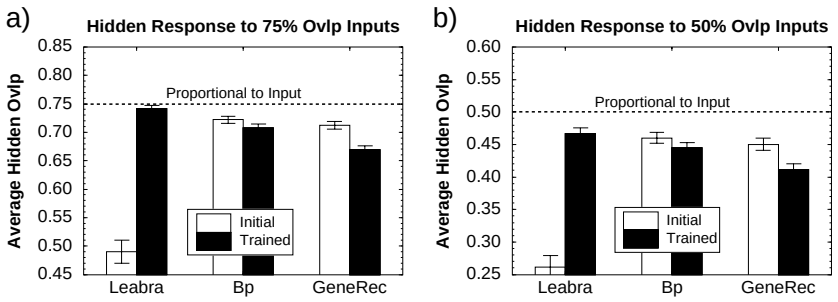

Figure 17: Average pairwise overlap (normalized dot product or cosine) between hidden patterns corresponding to inputs that differ by (a) 75% (one out of four slots different) and (b) 50% (two out of four slots different). Although with random weights the inhibitory competition in Leabra results in significant pattern separation (very different hidden representations for similar inputs), training produces a network with proportional responding to input differences.

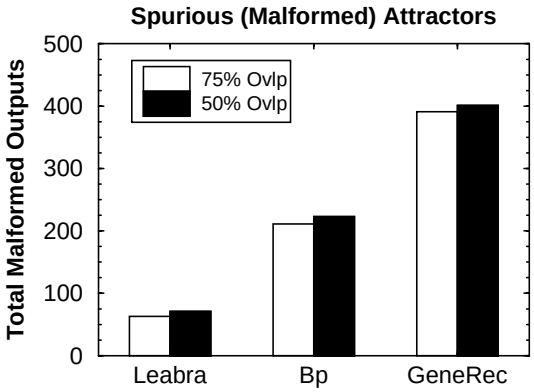

Figure 18: Number of malformed outputs out of 500 total for the 75% and 50% overlap inputs. An output is wellformed if each of the slot outputs corresponds to one of the vocabulary of possible slot outputs, and malformed otherwise. The Leabra network produces largely well-formed outputs, consistent with the generalization results.

irregulars (patterns were generated at random until 100 regulars had accumulated; the proportion of irregulars is close to the expected 20% having a vertical line in one out of five positions).

The generalization results for the three main algorithms (Leabra, Bp, GeneRec) are shown in Figure 19, where it is clear that the biases in Leabra still enable better generalization even in a task with exceptions. The overall level of generalization was worse than in the fully regular case, as one would

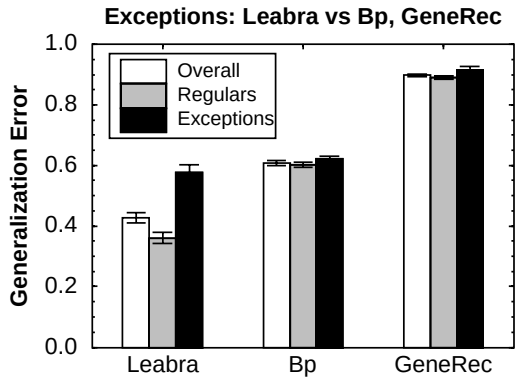

Figure 19: Generalization results (proportion error on test set of 500 items, and broken down into proportion error for regulars and exceptions separately) for the three algorithms (Leabra, Bp, and GeneRec) on the problem with exceptions. Although generalization performance is naturally worse than in the purely regular case, Leabra still performs better than the other algorithms (substantially better on regulars and marginally better on exceptions), indicating the advantages of the inhibitory competition and Hebbian learning biases.

expect given the more complex task being trained. However, the Leabra network is generalizing on the exceptional mapping better than either of the other networks (though not by much in the case of the Bp network), meaning that it is capable of systematically encoding more complex interactions between slots and exhibiting substantial generalization on that basis. Because the regular performance is better than the exception performance, it is clear that the biases in Leabra are specifically advantageous for the regular mapping, but this does not come at a cost to the exceptions.

The training error curves (not shown) were basically the same as the fully regular case, with just one epoch more required on average to reach a 0 error criterion (7.7 epochs in the fully regular case, 8.7 in the exception case). In examining the weights of the Leabra network (not shown), one could find units that encoded the exceptional cases by having input weights from the last position of the final slot, plus weights from the third slot, and a mapping to the fourth slot output appropriate for the third slot weights.

To follow up on the issue of the effect of number of hidden units on generalization, Figure 20 shows that even in the task with exceptions, larger hidden layers produce better generalization. Again, these results can be understood in terms of the benefits of a larger sample of idiosyncratic units that appear to outweigh the overfitting costs associated with this task, which is still deterministic, even if somewhat more complex.

In other work, the Leabra algorithm has also been shown to exhibit systematic encodings of complexly interacting stimuli, resulting in good gen-

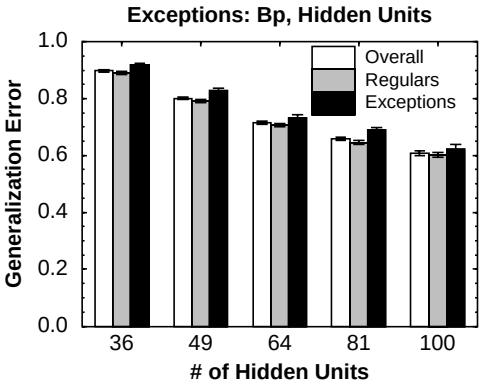

Figure 20: Generalization results for Bp with different hidden layer sizes on the problem with exceptions. Even with exceptions, a larger hidden layer improves generalization.

eralization. In particular, the spelling-to-sound model presented in O'Reilly and Munakata (2000) replicated the generalization performance of the backprop networks reported in Plaut et al. (1996) , but did so using simpler and more surface-valid input-output representations than the carefully crafted representations used by Plaut et al. (1996). Table 1 shows the nonword generalization data from this model compared with Plaut et al. (1996) and human data on a range of different nonword lists. To pronounce the nonwords correctly, the model has to encode complex combinations of both independent and conjunctive (i.e., taking into account multiple letters) letter-phoneme mappings. Thus, it should be clear that the benefits of the biases in Leabra are not restricted to simple, purely regular mappings.

7 Generalization in Handwritten Digit Recognition

To substantiate further the generality of the results on the tasks studied above, an additional task with very different characteristics was explored. This task is handwritten digit recognition, which has been widely used in the literature. The specific digit set employed, from Nowlan (1990), has 48 different samples of each of the 10 digits (0–9) encoded as thresholded binary intensities in a 16×16 input grid. Because the digits are handwritten, they are highly variable (noisy) within a category and overlap considerably with those in other categories. The challenge is to form category boundaries that are sufficiently general to provide reasonable generalization performance to novel instances, based on these noisy and overlapping training exemplars. Because of the noisy inputs, there is a strong possibility for overfitting in

Table 1: Summary of Nonword Reading Performance in the Leabra Model Compared to the PMSP (Plaut et al., 1996) and Human Data.

Nonword Set	Leabra	PMSP	People
Glushko (1979) regulars	95.3	97.7	93.8
Glushko (1979) exceptions	97.6	100.0	95.9
McCann & Besner (1987) controls	85.9	85.0	88.6
McCann & Besner (1987) homophones	92.3	N.A.	94.3
Taraban & McClelland (1987)	97.9	N.A.	100.0[a]

Notes: The Glushko data show the results when alternative outputs that are consistent with the training corpus are allowed.
[a]Human accuracy for the Taraban & McClelland set was not reported but was presumably near 100 percent (the focus of the study was on reaction times).
Source: Data from O'Reilly & Munakata (2000).

this domain. Training was performed on 32 instances per digit, with generalization testing on the remaining 16.

The generalization results for the three basic algorithms with 64 hidden units each are shown in Table 2a. This replicates the same pattern seen previously, where GeneRec generalizes considerably worse than feedforward backpropagation (Bp), and Almeida-Pineda (AP) performs intermediately (weight initialization variance was .5, producing moderate amounts of feedback weights that are not increased with training). Critically, the extra biases in Leabra produce much better generalization in a fully recurrent network, with generalization error slightly better than the Bp network with an equivalent number of hidden units.

The effect of number of hidden units, shown in Table 2b, shows an opposite pattern from previous results, with more hidden units leading to worse generalization in Bp and to a lesser extent in GeneRec, but not in Leabra (which actually shows the opposite pattern). This is to be expected because this task has noisy inputs, which allow for the units to overfit the specific, idiosyncratic features of the training inputs and thus not generalize as well to the novel testing inputs. However, the extra biases of the Leabra algorithm prevent it from this overfitting (e.g., the Hebbian learning forces the units to extract the central tendency of the input patterns corresponding to a given digit category), and it appears to benefit from a richer, more redundant distributed representation that comes from having a larger hidden layer. Finally, the fact that the best Bp network (with 15 hidden units) performs slightly better than Leabra (with 64 hiddens) is not of central relevance to the point of this article. The critical thing is that Leabra performs *much* better than GeneRec.

Table 2: Generalization in Digit Recognition Networks.

a.

Algorithm	Generalization Error	Standard Error of the Mean
Leabra	.131	.00456
Bp	.151	.00482
AP	.202	.0126
GeneRec	.279	.0323

b.

	Leabra		Bp		GeneRec	
Hiddens	Generalization Error	Standard Error of the Mean	Generalization Error	Standard Error of the Mean	Generalization Error	Standard Error of the Mean
15	.221	.0127	.12	.00867	.236	.00809
30	.164	.00878	.131	.0106	.231	.0106
64	.131	.00456	.151	.00482	.279	.0323

Notes: (a) Generalization results for digit recognition for the different algorithms (64 hidden units), shown as proportion generalization error on the 160 testing items. (b) Effects of number of hidden units, with an indication of overfitting with larger hidden layers in Bp and to a lesser extent in GeneRec, but not in Leabra (which actually shows the opposite pattern).

8 Discussion

This article has shown that an attempt to make error-driven backpropagation learning more biologically plausible by using bidirectional activation propagation to communicate error signals has the unfortunate consequence of producing bad generalization. This impaired generalization can be attributed to the conjunctive attractor dynamics of an otherwise unconstrained interactive network, which interfere with the ability to systematically form novel combinatorial representations. However, by adding inhibitory competition and Hebbian learning to the interactive network, good generalization is restored, and the resulting algorithm satisfies a variety of constraints from the biological, psychological, and computational perspectives.

The generalization results presented here demonstrate that neural networks are capable of substantial levels of generalization based on a relatively small sample of the environment (e.g., 100 training patterns out of 4.1 million possible, resulting in an average of 3.5 million correct responses). Such results should help to counter the persistent claims that neural networks are

incapable of producing systematic behavior based on statistical learning of the environment (Marcus, 1998; Pinker & Prince, 1988).

There is reason to believe that the particular combination of mechanistic principles embodied by the Leabra algorithm has a wide range of applicability. One can identify six core such principles: interactivity, inhibitory competition, distributed representations, error-driven task learning, Hebbian model learning, and an overarching concern for biological plausibility. These principles reflect important themes that have played a central role in a large number of neural network models and theories over the years (see O'Reilly, 1998, for a review). This article adds to this existing body of research by showing that there can be important computational advantages to combining these mechanisms together into one coherent framework. In addition to the specific issue of generalization that was the focus of this article, this framework can be used to understand a wide range of cognitive phenomena and provides a unified way of implementing a number of existing cognitive models (O'Reilly & Munakata, 2000).

Appendix: Implementational Details

A.1 Pseudocode. The pseudocode for Leabra is given here, showing exactly how the pieces of the algorithm described in more detail in the subsequent sections fit together.

Outer loop: Iterate over events (trials) within an epoch. For each event:

1. Iterate over minus and plus phases of settling for each event.
   - (a) At start of settling, for all units:
     - i. Initialize all state variables (activation, v_m, etc.).
     - ii. Apply external patterns (clamp input in minus, input and output in plus).
   - (b) During each cycle of settling, for all nonclamped units:
     - i. Compute excitatory netinput ($g_e(t)$ or η_j—equation A.3).
     - ii. Compute kWTA inhibition for each layer, based on g_i^Θ (equation A.7):
       - A. Sort units into two groups based on g_i^Θ: top k and remaining $k+1$ to n.
       - B. If basic, find k and $k+1$th highest; if average based, compute average of $1 \rightarrow k$ and $k+1 \rightarrow n$.
       - C. Set inhibitory conductance g_i from g_k^Θ and g_{k+1}^Θ (equation A.6).

iii. Compute point-neuron activation combining excitatory input and inhibition (equation A.1).

(c) After settling, for all units:

i. Record final settling activations as either minus or plus phase (y_j^- or y_j^+).

2. After both phases, update the weights (based on linear current weight values) for all connections:

(a) Compute error-driven weight changes (equation A.8) with soft weight bounding (equation A.13).

(b) Compute Hebbian weight changes from plus-phase activations (equation 4.1).

(c) Compute net weight change as weighted sum of error driven and Hebbian (equation A.14).

(d) Increment the weights according to net weight change, and apply contrast enhancement (equation 4.2).

A.2 Point Neuron Activation Function. Leabra uses a point neuron activation function that models the electrophysiological properties of real neurons, while simplifying their geometry to a single point. This function is nearly as simple computationally as the standard sigmoidal activation function, but the more biologically-based implementation makes it considerably easier to model inhibitory competition, as described below. Further, using this function enables cognitive models to be related to more physiologically detailed simulations more easily, thereby facilitating bridge building between biology and cognition.

The membrane potential V_m is updated as a function of ionic conductances g with reversal (driving) potentials E as follows:

$$\frac{dV_m(t)}{dt} = \tau \sum_c g_c(t)\overline{g_c}(E_c - V_m(t)) \tag{A.1}$$

with 3 channels (c) corresponding to e excitatory input, l leak current, and i inhibitory input. Following electrophysiological convention, the overall conductance is decomposed into a time-varying component $g_c(t)$ computed as a function of the dynamic state of the network, and a constant $\overline{g_c}$ that controls the relative influence of the different conductances. The equilibrium potential can be written in a simplified form by setting the excitatory driving potential (E_e) to 1 and the leak and inhibitory driving potentials (E_l and E_i) of 0:

$$V_m^\infty = \frac{g_e\overline{g_e}}{g_e\overline{g_e} + g_l\overline{g_l} + g_i\overline{g_i}}\,, \tag{A.2}$$

which shows that the neuron is computing a balance between excitation and the opposing forces of leak and inhibition. This equilibrium form of the equation can be understood in terms of a Bayesian decision-making framework (O'Reilly & Munakata, 2000).

The excitatory net input-conductance $g_e(t)$ or η_j is computed as the proportion of open excitatory channels as a function of sending activations times the weight values:

$$\eta_j = g_e(t) = \langle x_i w_{ij} \rangle = \frac{1}{n} \sum_i x_i w_{ij}. \tag{A.3}$$

The inhibitory conductance is computed via the kWTA function described in the next section, and leak is a constant.

Activation communicated to other cells (y_j) is a thresholded (Θ) sigmoidal function of the membrane potential with gain parameter γ,

$$y_j(t) = \frac{1}{\left(1 + \frac{1}{\gamma[V_m(t) - \Theta]_+}\right)}, \tag{A.4}$$

where $[x]_+$ is a threshold function that returns 0 if $x < 0$ and x if $x > 0$. Note that if it returns 0, we assume $y_j(t) = 0$, to avoid dividing by 0. As it is, this function has a very sharp threshold, which interferes with graded learning mechanisms (e.g., gradient descent). To produce a less discontinuous deterministic function with a softer threshold, the function is convolved with a gaussian noise kernel, which reflects the intrinsic processing noise of biological neurons,

$$y_j^*(x) = \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}\sigma} e^{-z^2/(2\sigma^2)} y_j(z - x) dz, \tag{A.5}$$

where x represents the $[V_m(t) - \Theta]_+$ value, and $y_j^*(x)$ is the noise-convolved activation for that value. In the simulation, this function is implemented using a numerical lookup table, as an analytical solution is not possible.

A.3 k-Winners-Take-All Inhibition. Leabra uses a kWTA function to achieve sparse distributed representations, with two different versions having different levels of flexibility around the k out of n active units constraint. Both versions compute a uniform level of inhibitory current for all units in the layer as follows,

$$g_i = g_{k+1}^{\Theta} + q(g_k^{\Theta} - g_{k+1}^{\Theta}), \tag{A.6}$$

where $0 < q < 1$ is a parameter for setting the inhibition between the upper bound of g_k^{Θ} and the lower bound of g_{k+1}^{Θ}. These boundary inhibition values

are computed as a function of the level of inhibition necessary to keep a unit right at threshold,

$$g_i^\Theta = \frac{g_e^* \bar{g}_e (E_e - \Theta) + g_l \bar{g}_l (E_l - \Theta)}{\Theta - E_i}, \tag{A.7}$$

where g_e^* is the excitatory net input without the bias weight contribution. This allows the bias weights to override the kWTA constraint.

In the basic version of the kWTA function, which is relatively rigid about the kWTA constraint, g_k^Θ and g_{k+1}^Θ are set to the threshold inhibition value for the kth and $k+1$th most excited units, respectively. Thus, the inhibition is placed exactly to allow k units to be above threshold and the remainder below threshold. For this version, the q parameter is almost always .25, allowing the kth unit to be sufficiently above the inhibitory threshold.

In the average-based kWTA version, g_k^Θ is the average g_i^Θ value for the top k most excited units, and g_{k+1}^Θ is the average of g_i^Θ for the remaining $n-k$ units. This version allows for more flexibility in the actual number of units active depending on the nature of the activation distribution in the layer and the value of the q parameter (typically between .5 and .7 depending on the level of sparseness in the layer, with a default value of .6). The simulations used the average-based version for the hidden layer (which can take advantage of the flexibility) with $q = .6$ (default), and the basic version for the output layer (which does not need the flexibility), with $q = .25$ (default). Performance using just the basic version for all layers was significantly worse (generalization error of 0.239) than the more appropriately biased case with average-based in the hidden layer (0.138) but was still better than backpropagation (0.401).

Activation dynamics similar to those produced by the kWTA function have been shown to result from simulated inhibitory interneurons that project both feedforward and feedback inhibition (O'Reilly & Munakata, 2000). Thus, although the kWTA function is somewhat biologically implausible in its implementation (e.g., requiring global information about activation states and using sorting mechanisms), it provides a computationally effective approximation to biologically plausible inhibitory dynamics.

A.4 Error-Driven Learning. As indicated in the text, Leabra uses the symmetric midpoint version of the GeneRec algorithm (O'Reilly, 1996a), which is functionally equivalent to the deterministic Boltzmann machine and contrastive Hebbian learning (CHL) (Hinton, 1989b; Movellan, 1990). The network settles in two phases—an expectation (minus) phase, where the network's actual output is produced and an outcome (plus) phase, where the target output is experienced—and then computes a simple difference of a pre- and postsynaptic activation product across these two phases (as

shown in equation 2.1 and reproduced here),

$$\Delta_{err} w_{ij} = (x_i^+ y_j^+) - (x_i^- y_j^-) \tag{A.8}$$

for sending unit x_i and receiving unit y_j in the two phases.

A.5 Hebbian Learning. The simplest form of Hebbian learning adjusts the weights in proportion to the product of the sending (x_i) and receiving (y_j) unit activations: $\Delta w_{ij} = x_i y_j$. The weight vector is dominated by the principal eigenvector of the pairwise correlation matrix of the input, but it also grows without bound. Leabra uses essentially the same learning rule used in competitive learning or mixtures-of-gaussians (Rumelhart & Zipser, 1986; Nowlan, 1990), which can be seen as a variant of the Oja normalization (Oja, 1982). This learning rule was shown in equation 4.1, and is reproduced here:

$$\Delta_{hebb} w_{ij} = x_i^+ y_j^+ - y_j^+ w_{ij} = y_j^+ (x_i^+ - w_{ij}) \tag{A.9}$$

After Rumelhart and Zipser (1986) and O'Reilly and Munakata (2000), we can see that this equation converges on the conditional probability that the sender is active given that the receiver is active. First, we write the activations as encoding the probability that the units are active for the current input pattern, indexed by t, and sum the weight changes over all patterns:

$$\begin{aligned} \Delta w_{ij} &= \epsilon \sum_t [P(y_j|t)P(x_i|t) - P(y_j|t)w_{ij}]P(t) \\ &= \epsilon \left(\sum_t P(y_j|t)P(x_i|t)P(t) - \right. \\ &\quad \left. \sum_t P(y_j|t)P(t)w_{ij} \right). \end{aligned} \tag{A.10}$$

Then we set Δw_{ij} to zero and solve to find the equilibrium weight value, resulting in:

$$\begin{aligned} 0 &= \epsilon \left(\sum_t P(y_j|t)P(x_i|t)P(t) - \right. \\ &\quad \left. \sum_t P(y_j|t)P(t)w_{ij} \right) \\ w_{ij} &= \frac{\sum_t P(y_j|t)P(x_i|t)P(t)}{\sum_t P(y_j|t)P(t)}. \end{aligned} \tag{A.11}$$

The interesting thing to note here is that the numerator $\sum_t P(y_j|t)P(x_i|t)P(t)$ is actually the definition of the joint probability of the sending and receiving units both being active together across all the patterns t, which is just

$P(y_j, x_i)$. Similarly, the denominator $\sum_t P(y_j|t)P(t)$ gives the probability of the receiving unit being active over all the patterns, or $P(y_j)$. Thus, we can rewrite the above equation as

$$w_{ij} = \frac{P(y_j, x_i)}{P(y_j)} = P(x_i|y_j), \tag{A.12}$$

at which point it becomes clear that this fraction of the joint probability over the probability of the receiver is just the definition of the conditional probability of the sender given the receiver.

A.6 Combining Error-Driven and Hebbian Learning. Error-driven and Hebbian learning are combined additively at each connection to produce a net weight change. Two equations are needed: a soft weight bounding equation to to keep the error-driven component within the same 0–1 range of the Hebbian term and the combination equation.

Soft weight bounding with exponential approach to the 0–1 extremes is implemented using

$$\Delta_{sberr} w_{ik} = [\Delta_{err}]_+ (1 - w_{ik}) + [\Delta_{err}]_- w_{ik}, \tag{A.13}$$

where Δ_{err} is the error-driven weight change, and the $[x]_+$ operator returns x if $x > 0$ and 0 otherwise, while $[x]_-$ does the opposite, returning x if $x < 0$, and 0 otherwise.

The net weight change equation combining error-driven and Hebbian learning (which also includes the learning rate parameter ϵ) uses a normalized mixing constant k_{hebb}:

$$\Delta w_{ij} = \epsilon[k_{hebb}(\Delta_{hebb}) + (1 - k_{hebb})(\Delta_{sberr})]. \tag{A.14}$$

Three different values of k_{hebb} were used to explore the relative contributions of error-driven and Hebbian learning: $k_{hebb} = 0$ gives purely error-driven learning (plus the kWTA inhibition constraint), $k_{hebb} = .02$ is the value used for combining Hebbian and error driven, producing the main results, and $k_{hebb} = 1$ gives purely Hebbian learning (plus the kWTA inhibition constraint). Note that only a relatively small amount of Hebbian learning is required to achieve the benefits; this is in part because the Hebbian learning factor is so persistently present on each trial of learning and shaping the learning so reliably according to the local correlations present.

Acknowledgments

I thank Yuko Munakata and Gary Cottrell for helpful comments. This work was supported in part by NIH program project grant MH47566 and NSF grant IBN-9873492.

References

Ackley, D. H., Hinton, G. E., & Sejnowski, T. J. (1985). A learning algorithm for Boltzmann machines. *Cognitive Science*, *9*, 147–169.

Almeida, L. B. (1987). A learning rule for asynchronous perceptrons with feedback in a combinatorial environment. In M. Caudil & C. Butler (Eds.), *Proceedings of the IEEE First International Conference on Neural Networks* (pp. 609–618). San Diego, CA.

Artola, A., Brocher, S., & Singer, W. (1990). Different voltage-dependent thresholds for inducing long-term depression and long-term potentiation in slices of rat visual cortex. *Nature*, *347*, 69–72.

Barlow, H. B. (1989). Unsupervised learning. *Neural Computation*, *1*, 295–311.

Bear, M. F. (1996). A synaptic basis for memory storage in the cerebral cortex. *Proceedings of the National Academy of Sciences*, *93*, 3453.

Bear, M. F., & Malenka, R. C. (1994). Synaptic plasticity: LTP and LTD. *Current Opinion in Neurobiology*, *4*, 389–399.

Brousse, O. (1993). *Generativity and systematicity in neural network combinatorial learning* (Tech. Rep. No. CU-CS-676-93). Boulder, CO: University of Colorado at Boulder, Department of Computer Science.

Crick, F. H. C. (1989). The recent excitement about neural networks. *Nature*, *337*, 129–132.

Dayan, P., & Zemel, R. S. (1995). Competition and multiple cause models. *Neural Computation*, *7*, 565–579.

Douglas, R. J., & Martin, K. A. C. (1990). Neocortex. In G. M. Shepherd (Ed.), *The synaptic organization of the brain* (pp. 389–438). Oxford: Oxford University Press.

Felleman, D. J., & Van Essen, D. C. (1991). Distributed hierarchical processing in the primate cerebral cortex. *Cerebral Cortex*, *1*, 1–47.

Field, D. J. (1994). What is the goal of sensory coding? *Neural Computation*, *6*(4), 559–601.

Földiák, P. (1990). Forming sparse representations by local anti-Hebbian learning. *Biological Cybernetics*, *64*(2), 165–170.

Fukai, T., & Tanaka, S. (1997). A simple neural network exhibiting selective activation of neuronal ensembles: From winner-take-all to winners-share-all. *Neural Computation*, *9*, 77–97.

Gabbot, P. L. A., & Somogyi, P. (1986). Quantitative distribution of GABA-immunoreactive neurons in the visual cortex (area 17) of the cat. *Experimental Brain Research*, *61*, 323–331.

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

Glushko, R. J. (1979). The organization and activation of orthographic knowledge in reading aloud. *Journal fo Experimental Psychology*, *5*, 674–691.

Goodhill, G. J., & Barrow, H. G. (1994). The role of weight normalization in competitive learning. *Neural Computation*, *6*, 255–269.

Grinvald, A., Frostig, R. D., & Lieke, E. (1988). Optical imaging of neuronal activity. *Physiological Review*, *68*, 1285–1366.

Grossberg, S. (1976). Adaptive pattern classification and universal recoding I: Parallel development and coding of neural feature detectors. *Biological Cybernetics*, *23*, 121–134.

Harm, M. W., & Seidenberg, M. S. (1999). Phonology, reading acquisition, and dyslexia: Insights from connectionist models. *Psychological Review*, *106*, 491–528.

Hinton, G. E. (1989a). Connectionist learning procedures. *Artificial Intelligence*, *40*, 185–234.

Hinton, G. E. (1989b). Deterministic Boltzmann learning performs steepest descent in weight-space. *Neural Computation*, *1*, 143–150.

Hinton, G. E., & Ghahramani, Z. (1997). Generative models for discovering sparse distributed representations. *Philosophical Transactions of the Royal Society (London) B*, *352*, 1177–1190.

Hinton, G. E., & McClelland, J. L. (1988). Learning representations by recirculation. In D. Z. Anderson (Ed.), *Neural information processing systems, 1987* (pp. 358–366). New York: American Institute of Physics.

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, *79*, 2554–2558.

Hopfield, J. J. (1984). Neurons with graded response have collective computational properties like those of two-state neurons. *Proceedings of the National Academy of Sciences*, *81*, 3088–3092.

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

Levitt, J. B., Lewis, D. A., Yoshioka, T., & Lund, J. S. (1993). Topography of pyramidal neuron intrinsic connections in macaque monkey prefrontal cortex (areas 9 and 46). *Journal of Comparative Neurology*, *338*, 360–376.

Lisman, J. E. (1989). A mechanism for the Hebb and the anti-Hebb processes underlying learning and memory. *Proceedings of the National Academy of Sciences*, *86*, 9574–9578.

Lisman, J. (1994). The CaM Kinase II hypothesis for the storage of synaptic memory. *Trends in Neurosciences*, *17*, 406.

Majani, E., Erlarson, R., & Abu-Mostafa, Y. (1989). The induction of multiscale temporal structure. In D. S. Touretzky (Ed.), *Advances in neural information processing systems, 1* (pp. 634–642). San Mateo, CA: Morgan Kaufmann.

Marcus, G. F. (1998). Rethinking eliminative connectionism. *Cognitive Psychology*, *37*, 243.

Marr, D. (1971). Simple memory: A theory for archicortex. *Philosophical Transactions of the Royal Society (London) B*, *262*, 23–81.

McCann, R. S., & Besner, D. (1987). Reading pseudohomophones: Implications for models of pronunciation and the locus of word-frequency effects in word naming. *Journal of Experimental Psychology*, *13*, 14–24.

McClelland, J. L., McNaughton, B. L., & O'Reilly, R. C. (1995). Why there are complementary learning systems in the hippocampus and neocortex: Insights from the successes and failures of connectionist models of learning and memory. *Psychological Review*, *102*, 419–457.

McClelland, J. L., & Rumelhart, D. E. (1981). An interactive activation model of context effects in letter perception: Part 1. An account of basic findings. *Psychological Review*, *88*(5), 375–407.

Miller, K. D., Keller, J. B., & Stryker, M. P. (1989). Ocular dominance column development: Analysis and simulation. *Science*, *245*, 605–615.

Miller, K. D., & MacKay, D. J. C. (1994). The role of constraints in Hebbian learning. *Neural Computation*, *6*, 100–126.

Movellan, J. R. (1990). Contrastive Hebbian learning in the continuous Hopfield model. In D. S. Touretzky, G. E. Hinton, & T. J. Sejnowski (Eds.), *Proceedings of the 1989 Connectionist Models Summer School* (pp. 10–17). San Mateo, CA: Morgan Kaufman.

Noelle, D. C., & Cottrell, G. W. (1996). In search of articulated attractors. In G. W. Cottrell (Ed.), *Proceedings of the 18th Annual Conference of the Cognitive Science Society* (pp. 329–334). Hillsdale, NJ: Erlbaum.

Noelle, D. C., & Zimdars, A. L. (1999). Methods for learning articulated attractors over internal representations. In M. Hahn, & S. C. Stoness (Eds.), *Proceedings of the 21st Annual Conference of the Cognitive Science Society* (pp. 480–485). Hillsdale, NJ: Erlbaum.

Nowlan, S. J. (1990). Maximum likelihood competitive learning. In D. S. Touretzky (Ed.), *Advances in neural information processing systems, 2* (pp. 574–582). San Mateo, CA: Morgan Kaufmann.

Oja, E. (1982). A simplified neuron model as a principal component analyzer. *Journal of Mathematical Biology*, *15*, 267–273.

Olshausen, B. A., & Field, D. J. (1996). Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, *381*, 607.

O'Reilly, R. C. (1996a). Biologically plausible error-driven learning using local activation differences: The generalized recirculation algorithm. *Neural Computation*, *8*(5), 895–938.

O'Reilly, R. C. (1996b). *The Leabra model of neural interactions and learning in the neocortex*. Unpublished doctoral dissertation, Carnegie Mellon University, Pittsburgh, PA.

O'Reilly, R. C. (1998). Six principles for biologically-based computational models of cortical cognition. *Trends in Cognitive Sciences*, *2*(11), 455–462.

O'Reilly, R. C., & McClelland, J. L. (1994). Hippocampal conjunctive encoding, storage, and recall: Avoiding a tradeoff. *Hippocampus*, *4*(6), 661–682.

O'Reilly, R. C., & Munakata, Y. (2000). *Computational explorations in cognitive neuroscience: Understanding the mind by simulating the brain*. Cambridge, MA: MIT Press.

O'Reilly, R. C., & Rudy, J. W. (2000). Computational principles of learning in the neocortex and hippocampus. *Hippocampus*, *10*, 389–397.

O'Reilly, R. C., & Rudy, J. W. (in press). Conjunctive representations in learning and memory: Principles of cortical and hippocampal function. *Psychological Review*.

Pineda, F. J. (1987a). Generalization of backpropagation to recurrent and higher order neural networks. In D. Z. Anderson (Ed.), *Proceedings of the IEEE Conference on Neural Information Processing Systems* (pp. 602–611). New York: IEEE.

Pineda, F. J. (1987b). Generalization of backpropagation to recurrent neural networks. *Physical Review Letters*, *18*, 2229–2232.

Pineda, F. J. (1988). Dynamics and architecture for neural computation. *Journal of Complexity*, *4*, 216–245.

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. (1993). Generalization with componential attractors: Word and nonword reading in an attractor network. *Proceedings of the 15th Annual Conference of the Cognitive Science Society* (pp. 824–829). Hillsdale, NJ: Erlbaum.

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.

Plunkett, K., & Marchman, V. A. (1996). Learning from a connectionist model of the acquisition of the English past tense. *Cognition*, *61*, 299.

Poggio, T., & Girosi, F. (1990). Regularization algorithms for learning that are equivalent to multilayer networks. *Science*, *247*, 978–982.

Rubinstein, Y. D., & Hastie, T. (1997). Discriminative vs. informative learning. In D. Heckerman, H. Mannila, & D. Pregibon (Eds.), *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining* (pp. 49–53). Menlo Park, CA: AAAI Press.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning internal representations by error propagation. In D. E. Rumelhart, J. L. McClelland, & PDP Research Group (Eds.), *Parallel distributed processing. Vol. 1: Foundations* (pp. 318–362). Cambridge, MA: MIT Press.

Rumelhart, D. E., & McClelland, J. L. (1986). On learning the past tenses of English verbs. In J. L. McClelland, D. E. Rumelhart, & PDP Research Group (Eds.), *Parallel distributed processing. Vol. 2: Psychological and biological models* (pp. 216–271). Cambridge, MA: MIT Press.

Rumelhart, D. E., & Zipser, D. (1986). Feature discovery by competitive learning. In D. E. Rumelhart, J. L. McClelland, & PDP Research Group (Eds.), *Parallel distributed processing. Vol. 1: Foundations* (pp. 151–193). Cambridge, MA: MIT Press.

Saund, E. (1995). A multiple cause mixture model for unsupervised learning. *Neural Computation*, *7*, 51–71.

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

Smolensky, P. (1986). Information processing in dynamical systems: Foundations of harmony theory. In D. E. Rumelhart, J. L. McClelland, & PDP Research Group (Eds.), *Parallel distributed processing. Vol. 1: Foundations* (pp. 282–317). Cambridge, MA: MIT Press.

Taraban, R., & McClelland, J. L. (1987). Conspiracy effects in word pronunciation. *Journal of Memory and language*, *26*, 608–631.

Vapnik, V. N., & Chervonenkis, A. (1971). On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and Its Applications*, *16*, 264–280.

Vecera, S. P., & O'Reilly, R. C. (1998). Figure-ground organization and object recognition processes: An interactive account. *Journal of Experimental Psychology: Human Perception and Performance*, *24*, 441–462.

Weigend, A. (1994). On overfitting and the effective number of hidden units. In M. C. Mozer, P. Smolensky, & A. S. Weigend (Eds.), *Proceedings of the 1993 Connectionist Models Summer School* (pp. 335–342). Hillsdale, NJ: Erlbaum.

Weigend, A. S., Rumelhart, D. E., & Huberman, B. A. (1991). Generalization by weight-elimination with application to forecasting. In R. P. Lippmann, J. E. Moody, & D. S. Touretzky (Eds.), *Advances in neural information processing systems, 3* (pp. 875–882). San Mateo, CA: Morgan Kaufmann.

White, E. L. (1989). *Cortical circuits: Synaptic organization of the cerebral cortex, structure, function, and theory*. Boston: Birkhäuser.

Wolpert, D. H. (1992). On the connection between in-sample testing and generalization error. *Complex Systems*, *6*, 47–94.

Wolpert, D. H. (1996a). The existence of a priori distinctions between learning algorithms. *Neural Computation*, *8*, 1391–1420.

Wolpert, D. H. (1996b). The lack of a priori distinctions between learning algorithms. *Neural Computation*, *8*, 1341–1390.

Zemel, R. S. (1993). *A minimum description length framework for unsupervised learning*. Unpublished doctoral dissertation, University of Toronto, Canada.

Zipser, D., & Andersen, R. A. (1988). A backpropagation programmed network that simulates response properties of a subset of posterior parietal neurons. *Nature*, *331*, 679–684.

Received March 11, 1999; accepted September 22, 2000.