CATASTROPHIC INTERFERENCE IN CONNECTIONIST NETWORKS: THE SEQUENTIAL LEARNING PROBLEM

Michael McCloskey
Neal J. Cohen

I. Introduction

Connectionist networks in which information is stored in weights on connections between simple processing units have attracted considerable interest in cognitive science (e.g., Rumelhart, McClelland, & the PDP Research Group, 1986; McClelland, Rumelhart, & the PDP Research Group, 1986). Much of the interest centers around two characteristics of these networks. First, the weights on connections between units need not be prewired by the model builder but rather may be established through training in which items to be learned are presented repeatedly to the network and the connection weights are adjusted in small increments according to a learning algorithm (e.g., Ackley, Hinton, & Sejnowski, 1985; Rumelhart, Hinton, & Williams, 1986; Hinton & Sejnowski, 1986). Second, the networks may represent information in a distributed fashion. That is, the representation of an item may be spread, or distributed, across many different processing units and connections, and each unit and connection may be involved in representing many different items.

Distributed representations established through the application of learning algorithms have several properties that are claimed to be desirable from the standpoint of modeling human cognition (e.g., Hinton, McClelland, & Rumelhart, 1986; McClelland, Rumelhart, & Hinton, 1986; but see Prince & Pinker, 1988; Fodor & Pylyshyn, 1988; Lachter &

THE PSYCHOLOGY OF LEARNING
AND MOTIVATION, VOL. 24

Bever, 1988). These properties include content-addressable memory and so-called automatic generalization, in which a network trained on a set of items responds correctly (i.e., generalizes) to other untrained items within the same domain. The present chapter focuses on another, less desirable, property of distributed representations: New learning may interfere catastrophically with old learning when networks are trained sequentially.

II. Sequential Learning and Interference

Typically, connectionist networks are trained by repeatedly presenting a single set of training items that includes all of the items to be learned, or at least incorporates all of the regularities to be captured by the network. This training method, which we will refer to as *concurrent training*, may in some circumstances accurately reflect the way in which a human learner encounters material to be learned. Often, however, human learning is more sequential. For example, children learning basic arithmetic facts such as $4 + 2 = 6$ are usually trained on addition facts before multiplication facts, and within each operation they usually encounter "small" facts such as $2 + 3$ before large facts such as $8 + 9$. Autobiographical memories provide an especially clear example of sequential learning: Memories for one's life experiences are obviously acquired sequentially over a lifetime.

Therefore, if connectionist networks are to be used to model human learning, then the networks must be able to learn sequentially. However, when many networks are trained sequentially, the problem of interference arises: Training on a new set of items may drastically disrupt performance on previously learned items.

Disruption of old knowledge by new learning is a recognized feature of connectionist models with distributed representations (e.g., Carpenter & Grossberg, 1986; Hinton *et al.*, 1986; Hinton & Plaut, 1987; Ratcliff, in press; Sutton, 1986). However, the interference is sometimes described as if it were mild and/or readily avoided (see, e.g., Hinton *et al.*, 1986, pp. 81–82). Perhaps for this reason, the interference phenomenon has received surprisingly little attention, and its implications for connectionist modeling of human cognition have not been systematically explored.

In this chapter we illustrate the phenomenon and assess its implications. After describing briefly the basic properties of the networks used in our modeling, we present two examples of catastrophic interference occurring under sequential learning conditions. We then explore the gen-

erality of the interference phenomenon, consider why it occurs, and discuss its ramifications.

III. Network Architecture and Learning Procedures

A. Network Architecture

The modeling reported in this chapter was carried out with layered feed-forward networks of deterministic units that could take on activation values ranging continuously from 0 (fully off) to 1.0 (fully on). Each of our networks included a layer of input units, a layer of output units, and (in most instances) an intervening layer of hidden units. All input units were connected to all hidden units, and all hidden units were connected to all output units. Connections between units had weights that could be positive or negative. Figure 1 presents for purposes of illustration a very simple "2–1–2" network with two input units (A and B), one hidden unit (C), and two output units (D and E).

When an input is presented to a network of this sort (by setting each input unit to a particular activation state), the input units send signals to the hidden units, thereby activating the hidden units to varying degrees. The hidden units then send signals to the output units, which in turn adopt activation states on the basis of these signals. Thus, activation is propagated forward from the input units to the hidden units and then to the output units. In this way the network maps a pattern of activation across the input units onto a pattern of activation across the output units.

The signal sent from a unit in one layer to a unit in the next layer is the

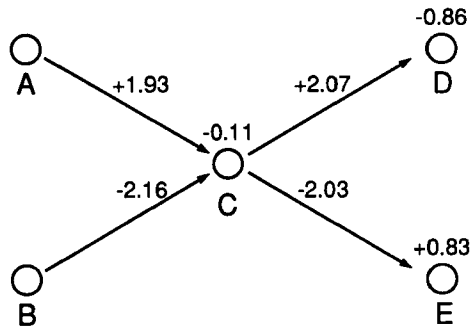


Fig. 1. A simple 2–1–2 network with two input units (A and B), one hidden unit (C), and two output units (D and E).

product of the activation of the sending unit and the weight on the connection from the sending unit to the receiving unit. Suppose, for example, that the pattern [1 0] (i.e., unit A on and unit B off) were presented to the 2–1–2 network in Fig. 1. The hidden unit C would receive a signal of 1.93 from unit A (i.e., 1, the activation of unit A, multiplied by 1.93, the weight on the A–C connection). The signal to unit C from unit B would be 0, because unit B has an activation of 0.

In addition to receiving signals from units in preceding layers, each hidden unit and output unit has a bias (indicated in Fig. 1 by the number above each unit) they may be positive or negative. Biases function precisely like connection weights, and in fact a unit's bias may be thought of as the weight on a connection to that unit from a unit that is always on.

The activation level of a unit is determined by summing the signals it receives as input, and transforming the sum nonlinearly by means of the logistic function $a = 1/(1 + e^{-I})$, where a is the activation state of the unit, and I is the sum of the unit's inputs.

The logistic activation function has several interesting properties. If a unit's inputs sum to 0, the unit will take on an activation value of .5. Predominantly negative inputs drive the activation value below .5, and predominantly positive inputs raise the activation above .5. The function approaches an asymptote of 0 as inputs become increasingly negative, and reaches an asymptote at 1.0 as the inputs become increasingly positive. Thus, units can take on activation values ranging continuously from 0 to 1.0. In the case of the pattern [1 0] presented to the 2–1–2 network, the total input to the hidden unit would be 1.82 (i.e., the sum of 1.93, the input from unit A; 0, the input from unit B; and −0.11, the unit's bias), resulting in an activation level of .86. This level of activation in the hidden unit would in turn lead to an activation level of .71 for output unit D, and .29 for unit E.

B. Training Procedure

Training a connectionist network on a set of items to be learned involves a series of learning trials. On each trial the items in the training set are presented to the network, and the connection weights are altered in accordance with a learning algorithm. The goal of the training is to configure the weights in such a way that the network will map each input pattern in the training set onto the desired output pattern. The weights are adjusted in small steps, so that over learning trials the outputs produced by the network gradually come to approximate the target outputs more and more closely.

In the modeling reported in this chapter we used the back-propagation learning algorithm (Rumelhart, Hinton, & Williams, 1986). On each learning trial the items in the training set were presented in a random order. For each item the input pattern was presented, and the network was allowed to generate an output in the manner described in the preceding section. The network's output was then compared to the target output, and on the basis of this comparison the connection weights were altered.

In concurrent training all of the items to be learned were included in a single training set, which was presented repeatedly over a series of learning trials until good performance was achieved. In sequential training, however, the network was first trained on one set of items and then on a second set, with the aim of assessing retention of the originally trained items during learning of the new items.

C. The Learning Algorithm

The back-propagation learning algorithm aims to minimize the squared difference between target activation levels for output units and the activation levels generated by the network. After the network generates an output in response to an input pattern, an error signal d is computed for each output unit as

$$d = (t - a)(a)(1 - a)$$

where a is the activation level of the output unit, and t is the target activation level. The term $(a)(1 - a)$ is the derivative of the logistic function at the point corresponding to the activation of the output unit. Following Rumelhart, Hinton, and Williams (1986), we computed error signals using .9 rather than 1.0 as the target activation level for a unit that should be on, and .1 rather than 0 as the target level for a unit that should be off. (The use of these target values improves the performance of the learning algorithm under certain circumstances. See pp. 141–142 and especially footnote 3, for further discussion of this point.)

Suppose, for example, that we want the 2–1–2 network to generate the output pattern [1 0] (i.e., unit D on, unit E off) in response to the input [1 0]. As we have seen, the network in Fig. 1 generates the output [.71 .29] when presented with this input pattern. Thus, the error signal for unit D would be computed as $(.9 - .71)(.71)(1 - .71)$, or .039. For unit E, the error signal would be $-.039$.

The error signals for the output units are used to compute error signals for the hidden units. Specifically, the error signal for a hidden unit is given by

$$d_i = a_i(1 - a_i) \sum_j w_{ji} d_j$$

where w_{ji} is the weight on the connection to output unit j from hidden unit i. The term $a_i(1 - a_i)$ is the derivative of the logistic function at the point corresponding to the activation of the hidden unit, and the other term in the equation is the sum across all output units of the product of the output unit's error signal and the weight on the connection to the output unit from the hidden unit. Carrying out this computation for the hidden unit in the 2–1–2 example yields an error signal of .019 for this unit.

Finally, the error signals are used to compute changes in connection weights:

$$\Delta w_{ji} = r d_j a_i$$

where r is a constant that is referred to as the learning rate parameter. The learning rate determines the size of the weight adjustments during learning. With a high learning rate, weights may be charged by a relatively large amount at each weight adjustment, whereas with low learning rates the weights are changed in smaller steps.

Rumelhart, Hinton, and Williams (1986) note that the performance of the learning algorithm is often improved by including *momentum* in the calculation of weight changes. Momentum is computed as a proportion of the weight change made for the connection on the immediately preceding cycle of weight adjustments. Thus, weight changes are calculated as follows:

$$\Delta w_{ji}[n] = r d_j a_i + m \Delta w_{ji}[n - 1]$$

where $\Delta w_{ji}[x]$ indicates the weight change for weight adjustment cycle x, and m is a constant referred to as the momentum parameter. With the inclusion of momentum, the change in a connection's weight tends to be similar to prior weight changes for that connection. In the simulations we report, a momentum parameter of .9 was used.

Consider, for example, the connection from the hidden unit C to output unit D in the 2–1–2 network. Assuming that the learning rate is set at .5, and that the weight change for the connection was .071 on the preceding cycle of weight adjustments, the weight change for the connection would be [(.5)(.039)(.86)] + [(.9)(.071)], or .081. Thus, the weight on the connection would be changed by the learning algorithm from +2.07 to +2.15.

The back-propagation learning algorithm has one additional property that requires discussion. If a network begins with equal weights on all connections (e.g., weights of 0), then for any given input to the network

every hidden unit will have the same activation level and the same error signal. Therefore, the weight change computed for the connection from a hidden unit to any given output unit will be the same for all hidden units, and the weight change for the connection from an input unit to a hidden unit will be the same for all hidden units. As a result, the hidden units will fail to differentiate over learning trials. That is, weight configurations that differ from hidden unit to hidden unit will not develop; rather, every hidden unit will have the same configuration of weights on connections with input and output units. Because learning in most situations requires differentiation of hidden units, this constitutes a problem. However, the problem is easily resolved by initializing connection weights to small random values (Rumelhart, Hinton, & Williams, 1986). In the modeling discussed in this chapter the connection weights were initially set to values sampled randomly from a uniform distribution ranging from $-.3$ to $+.3$.

We turn now to our first example of catastrophic interference occurring under sequential learning conditions.

IV. Arithmetic Facts

Figure 2 depicts a three-layer network for learning basic arithmetic facts, such as $2 + 3 = 5$. The network included 28 input units, 50 hidden units, and 24 output units. We assume that encoding of a problem (e.g., $6 + 1$) corresponds to the creation of a pattern of activation across the input units, as illustrated in the figure. (Shading indicates level of activation: the darker the shading, the higher the activation.) The first 12 input units represent the first number in the problem (e.g., 6), the next 12 units represent the second number (e.g., 1), and the final 4 units represent the arithmetic operation (e.g., addition). Similarly, the first 12 output units represent the tens digit of the answer, and the remaining 12 units represent the ones digit.

We used a "coarse-coded" representation for numbers, as shown in Table I. Each of the numbers 0–9 was represented by activation of three units, and for numbers close in magnitude some of the same units were activated. Thus, the network represents numbers and arithmetic facts in a distributed fashion. The individual units in the network do not represent numbers, and individual connections in the network do not represent relations between numbers. Rather, numbers are represented by patterns of activation across several units, and each unit is involved in representing several different numbers. Further, an arithmetic fact such as $6 + 1 = 7$ cannot be localized to a small set of interconnected units that represent only that fact. Rather, the representation of a fact is distributed across

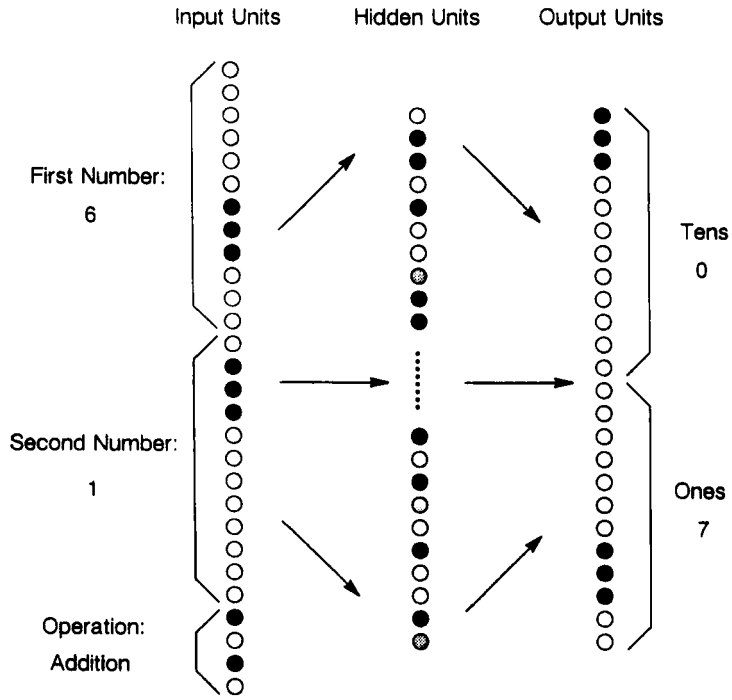


Fig. 2. The arithmetic network.

TABLE I

Coarse-Coded Representations for Numbers in the Arithmetic Network

	Unit											
Number	1	2	3	4	5	6	7	8	9	10	11	12
0	1	1	1	0	0	0	0	0	0	0	0	0
1	0	1	1	1	0	0	0	0	0	0	0	0
2	0	0	1	1	1	0	0	0	0	0	0	0
3	0	0	0	1	1	1	0	0	0	0	0	0
4	0	0	0	0	1	1	1	0	0	0	0	0
5	0	0	0	0	0	1	1	1	0	0	0	0
6	0	0	0	0	0	0	1	1	1	0	0	0
7	0	0	0	0	0	0	0	1	1	1	0	0
8	0	0	0	0	0	0	0	0	1	1	1	0
9	0	0	0	0	0	0	0	0	0	1	1	1

many different connections between many different units, and each unit and connection is involved in representing many different facts.

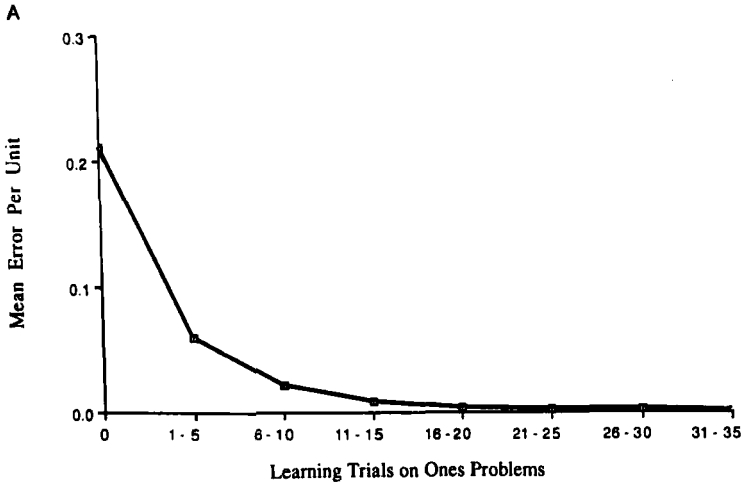
When the network is trained concurrently on the 200 single-digit addition and multiplication problems (i.e., 0 + 0 through 9 + 9, and 0 × 0 through 9 × 9), excellent performance is readily achieved. As we have noted, however, concurrent training does not conform well to the way in which children learn arithmetic facts. Hence, we decided to explore the performance of the network with sequential training. In the simulation we will discuss, the network was first trained on the ones addition facts, and then on the twos facts. In this simulation, and in fact in all of the simulations we report in this chapter, two independent runs were carried out, and results were averaged over the two runs. In no instance were there substantial differences in network performance between runs.

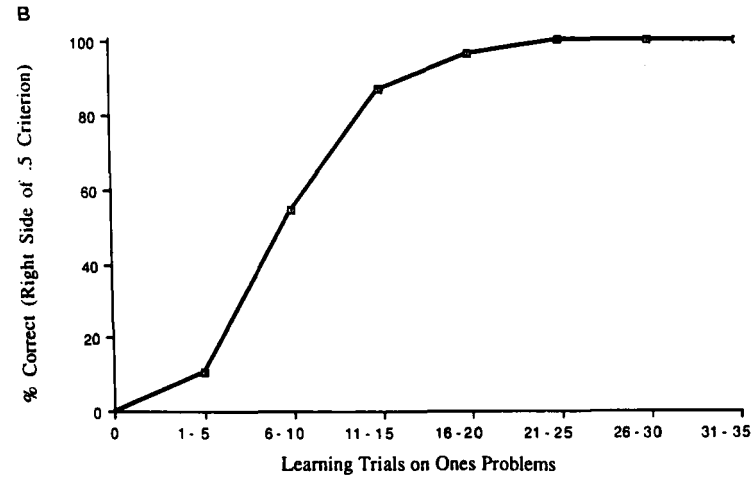
A. Training on the Ones Addition Facts

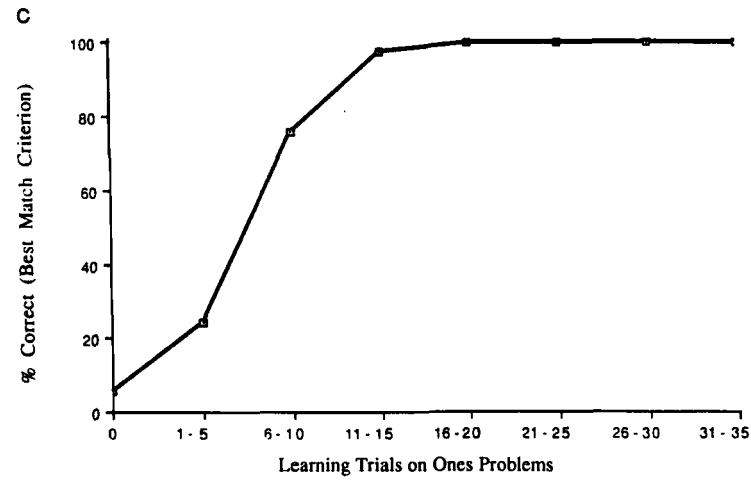
A training set consisting of the 17 single-digit ones problems (i.e., 1 + 1 through 9 + 1, and 1 + 2 through 1 + 9) was presented repeatedly until the network responded correctly to all of these problems. The learning rate parameter was set to .25, which is conservative in the sense of ensuring that weight changes will be relatively small.

Because there are significant questions concerning how the performance of connectionist models should be evaluated, we considered four different performance measures. An *error* measure was defined as the squared difference between the target activation level for an output unit and the activation level produced by the network, averaged over output units and over training problems. Figure 3A plots error across learning trials; each data point in the figure is an average over five consecutive trials. (A learning trial consisted of one presentation of each of the 17 training problems.) It is apparent that error declines steadily over training, reflecting the fact that the network's outputs come to approximate the target outputs more and more closely.

The other three performance measures concern the percentage of problems the network answers correctly. According to a stringent *within .1* criterion a response was considered correct only if all of the output units had activation levels within .1 of the target activation levels. A less stringent criterion required each output unit to be closer to the target on/off state than to the alternative state. In other words, all units that were supposed to be on were required to have activation levels of greater than .5, and all units that were supposed to be off were required to have activation levels of less than .5. We refer to this criterion as the *right side of .5* criterion. Figure 3B presents the performance of the network according to the right side of .5 criterion.


A
Mean Error Per Unit
0.3
0.2
0.1
0.0
0
1 - 5
6 - 10
11 - 15
16 - 20
21 - 25
26 - 30
31 - 35
Learning Trials on Ones Problems


B
% Correct (Right Side of .5 Criterion)
100
80
60
40
20
0
0
1 - 5
6 - 10
11 - 15
16 - 20
21 - 25
26 - 30
31 - 35
Learning Trials on Ones Problems


C
% Correct (Best Match Criterion)
100
80
60
40
20
0
0
1 - 5
6 - 10
11 - 15
16 - 20
21 - 25
26 - 30
31 - 35
Learning Trials on Ones Problems

Finally, network performance was also assessed with a *best match* criterion. By this relatively lax criterion a response was counted correct if it was closer to the correct response than to any of the alternative responses; closeness was defined by the error measure. For example, a response to the problem 4 + 1 was counted correct by the best match criterion if the network's output more closely matched the output pattern for the number 5 than the pattern for any other answer to a single-digit addition problem. Figure 3C shows the network's performance according to the best match criterion. By any of the performance measures the network performed very well after about 15–35 learning trials.

Figure 4 illustrates how the network's output changes over training. Panel A shows the target output pattern for the problem 6 + 1; this pattern represents the number 7. Each bar on the graph represents the activation level of one of the output units. For the problem 6 + 1 output units 1–3 should be on, representing zero tens. Specifically, the desired activation level for these units is .9. Units 20–22 should also be on, representing seven ones. Finally, the other output units should be off; that is, these units should have activation levels of about .1.

Panel B shows the network's output for the problem 6 + 1 prior to training. Panel C shows the output after five learning trials on the ones problems, and panel D shows performance at the completion of training on these problems. It is apparent that over learning trials the network's output comes to approximate the correct output more and more closely.

B. Training on the Twos Addition Facts

Training on the ones problems continued until all 17 problems were correct by the stringent within .1 criterion. The network was then trained on the 17 twos facts (i.e., 2 + 1 through 2 + 9, and 1 + 2 through 9 + 2), much as a child might learn these facts after learning the ones facts. (Note that 2 + 1 and 1 + 2 were included in both the ones training set and the twos training set.)

Following each learning trial on the twos problems, the network was tested on both the ones and two facts. Because no weight adjustments were made during the test trials, the testing in no way affected the network's performance on any of the facts. In reporting results for ones and twos problems during training on the twos, we considered 2 + 1 and 1

Fig. 3. A, Error as a function of number of learning trials during training on the ones addition facts. B, Performance of the arithmetic network according to the right side of .5 criterion during training on the ones addition facts. C, Performance of the arithmetic network according to the best match criterion during training on the ones addition facts.

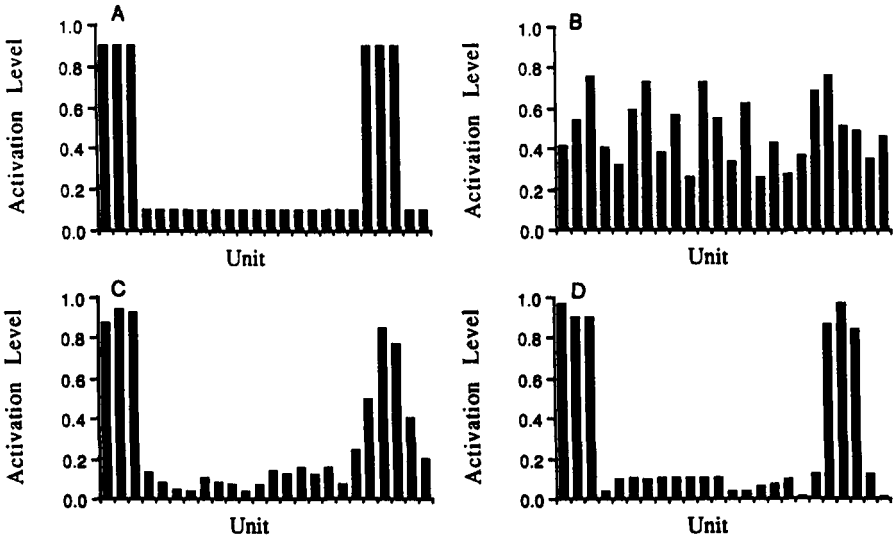


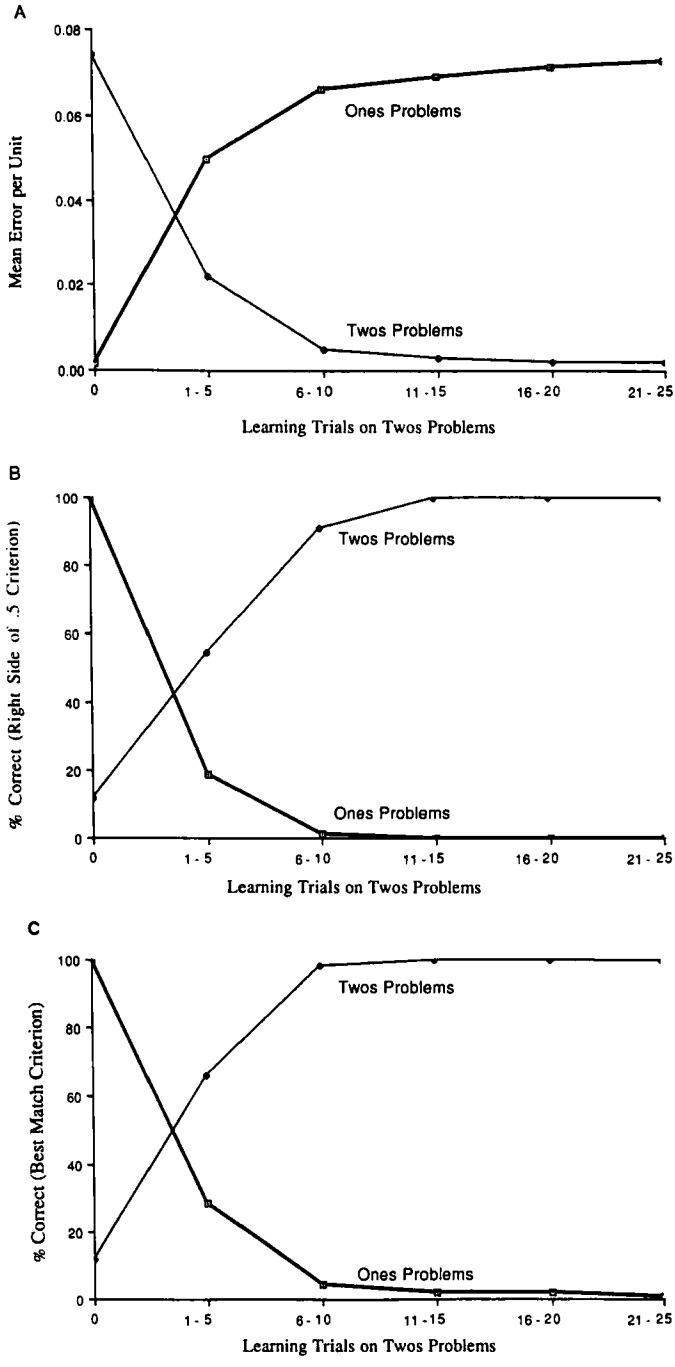
Fig. 4. Output of the arithmetic network for the problem 6 + 1 as a function of amount of training. A, Target output pattern; B, Network output prior to training; C, Network output after five learning trials on the ones addition facts; D, Network output at the completion of ones training.

+ 2 to be twos problems, because these problems were included in the twos training set. Thus, the results reported for the ones problems come from the 15 problems that were not trained during learning of the twos.

Like the ones facts, the twos facts were readily learned. However, once the network had learned the twos facts it no longer responded correctly to the ones problems. Figure 5A shows the error measure for the ones and twos facts during training on the twos. It is evident that training on the twos facts drastically increases error on the ones facts. In fact, the average squared error for the ones problems increases by more than an order of magnitude after a single learning trial on the twos facts, from .0015 to .0453.

Performance on the ones facts is also drastically disrupted when assessed with the right side of .5 and within .1 criteria, as illustrated for the

Fig. 5. A, Error measure for ones and twos addition facts as a function of number of learning trials on the twos facts. B, Performance of the arithmetic network on the ones and twos addition facts as a function of number of learning trials on the twos facts, assessed according to the right side of .5 criterion. C, Performance of the arithmetic network on the ones and twos addition facts as a function of number of learning trials on the twos facts, assessed according to the best match criterion.


A
Mean Error per Unit
0.08
0.06
0.04
0.02
0.00
Ones Problems
Twos Problems
0
1 - 5
6 - 10
11 - 15
16 - 20
21 - 25
Learning Trials on Twos Problems
B
% Correct (Right Side of .5 Criterion)
100
80
60
40
20
0
Twos Problems
Ones Problems
0
1 - 5
6 - 10
11 - 15
16 - 20
21 - 25
Learning Trials on Twos Problems
C
% Correct (Best Match Criterion)
100
80
60
40
20
0
Twos Problems
Ones Problems
0
1 - 5
6 - 10
11 - 15
16 - 20
21 - 25
Learning Trials on Twos Problems

former criterion in Fig. 5B. Finally, Figure 5C presents results tabulated according to the best match criterion. Even by this lax criterion, degradation of the ones learning is quite severe: performance on the ones decreases from 100% to 57% correct after a single learning trial on the twos facts, and to 30% correct after two trials.

The poor best match performance is particularly noteworthy, because it is something of an open question how an output pattern generated by a network such as our arithmetic network would be transformed into an overt response. It is conceivable that even if learning the twos facts substantially increased the squared error for the ones facts, or even led to outputs in which some units were on the wrong side of .5, the outputs might still be close enough to the target outputs to allow a correct response to be generated. However, it is difficult to argue that an output pattern that resembles the pattern for an incorrect number more closely than the pattern for the correct number could provide the basis for a correct response.

1. Errors

The network's errors on the ones facts after training on the twos facts take an interesting form. After training on the twos facts the network responds to the vast majority of the ones problems as if they were twos problems; for example, 5 + 1 is 7, and 6 + 1 is 8. Figure 6 shows the

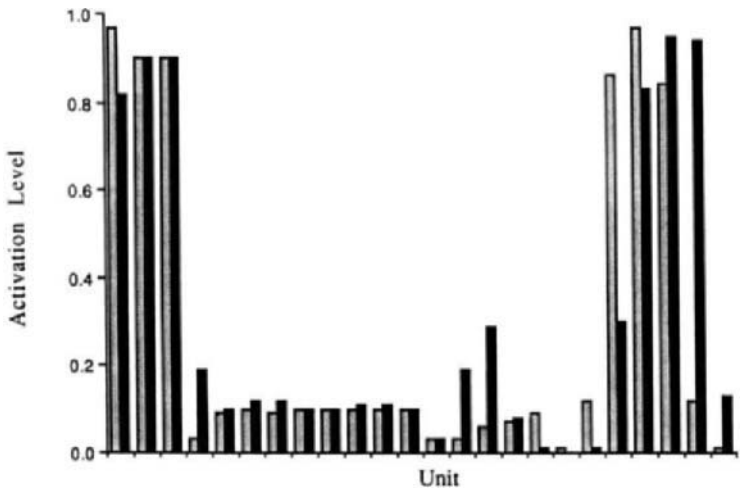


Fig. 6. Comparison of network output for the problem 6 + 1 at the completion of ones training (stippled bars), and after one learning trial on the twos facts (solid bars).

output for the problem 6 + 1 at the completion of training on the ones facts (stippled bars), and then after one learning trial on the twos facts (solid bars). It is apparent that after a single trial on the twos facts the network's response to 6 + 1 is no longer correct. The activation level for two of the output units has changed dramatically, with the result that the new output closely resembles the pattern for the number 8.

2. *Performance on 2 + 1 and 1 + 2*

The problems 2 + 1 and 1 + 2 were included in both the ones training set and the twos training set. At the completion of ones training, then, 2 + 1 and 1 + 2 were correct by the stringent within .1 criterion, and training on these problems continued uninterrupted during learning of the twos facts. Thus, it seems patently obvious that excellent performance on 2 + 1 and 1 + 2 would be maintained during training on the twos facts.

Remarkably, however, this was not the case. Performance on 2 + 1 and 1 + 2 showed brief but dramatic disruption during the first few learning trials on the twos facts. After a single learning trial on the twos facts the average squared error for 2 + 1 and 1 + 2 increased from 0021 to .0363, and both facts were incorrect not only by the within .1 and right side of .5 criteria, but even by the best match criterion. Additional training quickly reestablished correct responses to both problems according to the best match and right side of .5 criteria (after the second twos learning trial in one run of the simulation, and after the third trial in the other run). However, the facts were not correct by the within .1 criterion (which both of the facts met at the completion of training on the ones) until after about the eighth trial.

C. Implications of the Arithmetic Results

In simulating sequential learning of arithmetic facts, we found that the learning of new facts profoundly disrupted performance on previously learned facts. To the extent that one is interested in using connectionist networks to model human learning and memory, this sort of disruption would appear to be a significant problem.

It is certainly possible that the sequential training used in our simulation was too sequential; children do not fully learn new facts without practicing previously learned facts. However, after even a single learning trial on the twos problems, performance on the ones problems was substantially disrupted. Furthermore, training on the twos facts briefly but drastically impaired performance on 2 + 1 and 1 + 2, even though training on these facts continued during learning of the twos. It seems unlikely that human learners, after having learned the facts 2 + 1 = 3 and 1 + 2

= 3 in the context of the other ones problems, would experience serious difficulty with these facts when they were studied in the context of the twos facts. Thus, the network's performance for 2 + 1 and 1 + 2 departs in a particularly striking fashion from the performance expected of human learners.

We should make clear that the mere occurrence of interference in connectionist models is not a problem. Disruption of old knowledge by new learning is seen not only in connectionist networks but also in humans; the phenomenon of retroactive interference is well documented (e.g., Barnes & Underwood, 1959). Accordingly, interference comparable in severity to that observed in human learners would be a desirable feature in connectionist networks; such interference would suggest that the networks can capture the sorts of limitations that people encounter as they learn and remember.

Unfortunately, the results we have reported cannot readily be characterized as retroactive interference effects of a sort we would expect to observe in human learners. In the first place, it is by no means a foregone conclusion that retroactive interference occurs at all in the learning of arithmetic facts. It is conceivable that in some circumstances, especially those involving acquisition of knowledge in highly structured domains, the learning of new material would have no effect, or even a facilitating effect, on previously learned material. Even if it were assumed that human learners were subject to retroactive interference in acquisition of arithmetic facts, the interference we observed in our network would appear to be much more severe than one would expect from a human learner. To put it somewhat flippantly, the magnitude of the observed interference makes it seem more like retrograde amnesia than retroactive interference.

V. Retroactive Interference

To explore further the relative severity of retroactive interference (RI) effects in human learners and sequentially trained connectionist networks, we simulated a classic retroactive interference experiment (Barnes & Underwood, 1959).

A. The Barnes and Underwood Study

In the Barnes and Underwood (1959) experiment subjects learned lists of eight paired associates in an A–B, A–C design. Stimuli were nonsense syllables (e.g., *dax*), and responses were adjectives (e.g., *regal*). The sub-

jects were first trained on the A–B list. On each learning trial the A stimuli were presented one at a time, the subject attempted for each stimulus to recall the appropriate B response, and finally the correct response was presented for the subject to study. Training ended after the first trial on which the subject correctly recalled all eight responses. Subjects required an average of 10.36 trials to reach this learning criterion.

After completion of A–B training, subjects were given 1, 5, 10, or 20 learning trials on the A–C list. Following this A–C training, the subjects received a final test in which each stimulus (A) item was presented, and the subject was asked to recall both the first list (B) and the second list (C) response. Barnes and Underwood argue that subjects did not expect to be tested on the A–B list after the completion of A–B learning, and hence had no reason to try to remember the B responses during A–C training. Further, when Barnes and Underwood asked subjects whether they had used the first list (B) responses to mediate learning of the second list (C) responses, only 2 of the 96 subjects reported that they had. (These two subjects reported that the use of the first-list responses had only confused them.)

Barnes and Underwood's results for the final recall test are presented in Fig. 7. As the number of A–C learning trials increased, recall of the C responses improved steadily, from 3.46 out of 8 (43%) after one A–C trial,

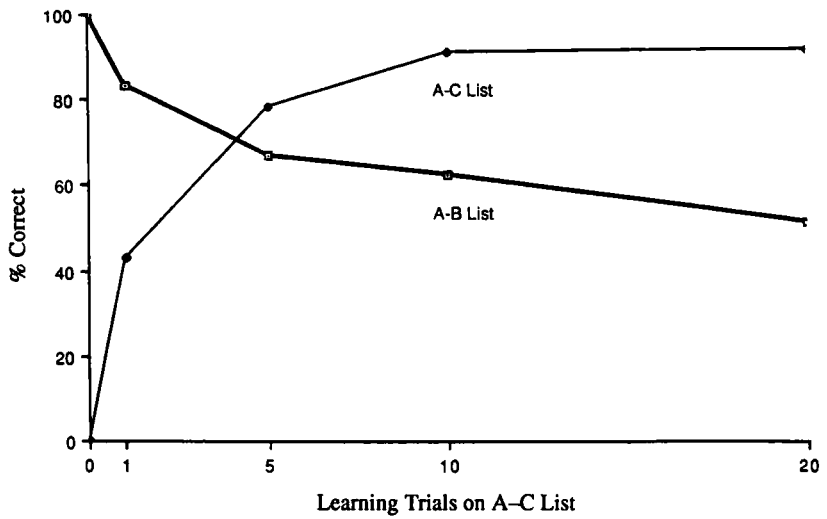


Fig. 7. Percentage correct on the A–B and A–C list in the Barnes and Underwood (1959) study as a function of number of trials on the A–C list.

to 7.33 (92%) after 20 trials. At the same time recall of the B responses steadily declined, from 6.67 (83%) after one trial on the A–C list, to 4.12 (52%) after 20 trials. Thus, learning of the C responses interfered with recall of the B responses, and this retroactive interference effect was larger the greater the amount of training on the A–C list.

B. The RI Simulation

1. The RI Network

We simulated the Barnes and Underwood study roughly, with the three-layer network shown in Fig. 8. The input on each trial consisted of a 10-unit pattern representing the A stimulus and a 10-unit context pattern. The output was a 10-unit pattern representing the response. The patterns used to represent stimuli, responses, and contexts were sampled randomly without replacement from the set of 1024 possible patterns of 10 zeroes and ones (e.g., [0 0 1 0 1 1 1 0 0 0]).

The context pattern presented as part of the input vector served to differentiate the A–B and A–C lists. The context pattern was constant during A–B training, and was changed to a different pattern during A–C train-

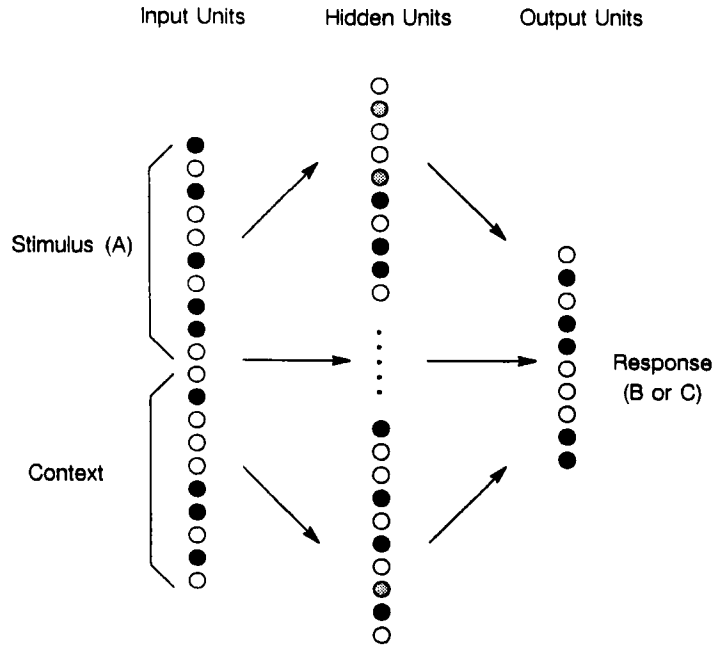


Fig. 8. The RI network.

ing. Thus, the goal was to train the network to respond with the appropriate B response when presented with an A stimulus and the A–B context pattern and to respond with the appropriate C response when a stimulus was presented with the A–C context pattern.

2. *Concurrent Training*

Testing with concurrent training methods revealed that when the A–B and the A–C items were included in a single training set that was presented repeatedly, the network could readily learn all 16 associations. Manipulation of the number of hidden units indicated that the network could learn all of the items with as few as 10 hidden units, but that larger sets of hidden units led to faster learning, up to about 40 hidden units. Further increases in the number of hidden units led to only slight increases in rate of learning. Our initial sequential learning runs used 50 hidden units, 10 more than the number at which the rate of concurrent learning began to reach an asymptote.

We also manipulated the learning rate parameter over the range .1–1.0, and found that the fastest concurrent learning was achieved with rates of .25 and .50; at both higher and lower parameter values, learning was slower. In the sequential learning runs we initially used a learning rate of .25, which is the more conservative of the two rates that yielded rapid concurrent learning (in the sense of yielding smaller weight changes on each cycle of weight adjustments).

3. *Sequential Training*

a. A–B Training. The network was trained on the A–B list until perfect performance was achieved according to the stringent .1 criterion. The mean number of learning trials required to reach this criterion was 42.5.

b. A–C Training. After completion of A–B training, the network was trained on the A–C list. Following each A–C learning trial, performance on both the A–B and A–C lists was assessed, by presenting each A stimulus with the A–B context pattern, and with the A–C context pattern.

Figure 9 depicts the effect of A–C training on the error measure for the B and C responses. It is apparent from the figure that with the onset of A–C training, error on the A–B list immediately increased dramatically.

Catastrophic interference was also apparent when the network's performance was assessed in terms of percentage of correct responses. Figure 10A,B presents performance assessed with the right side of .5 and within .1 criteria, respectively. By either criterion performance on the A–B list was reduced to zero by an amount of A–C training insufficient to yield *any* correct C responses.

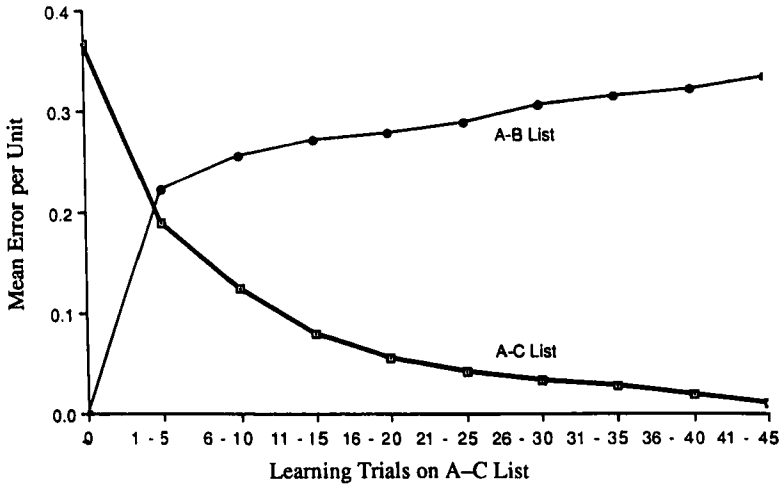


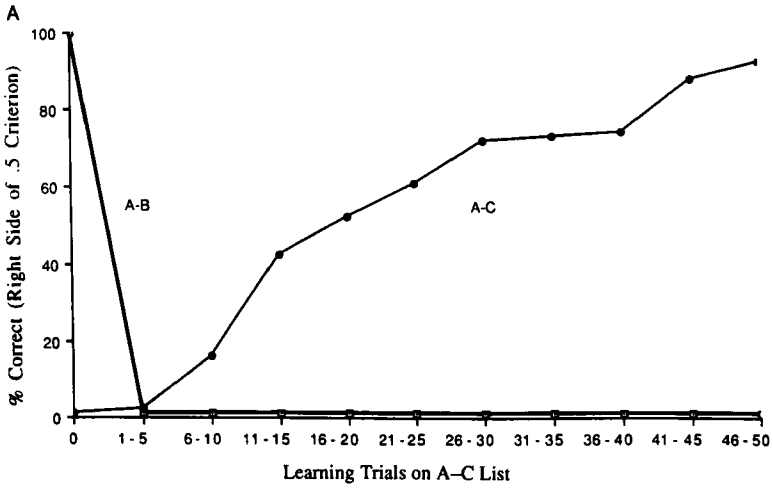
Fig. 9. Error measure for the A–B and A–C list as a function of number of learning trials on the A–C list.

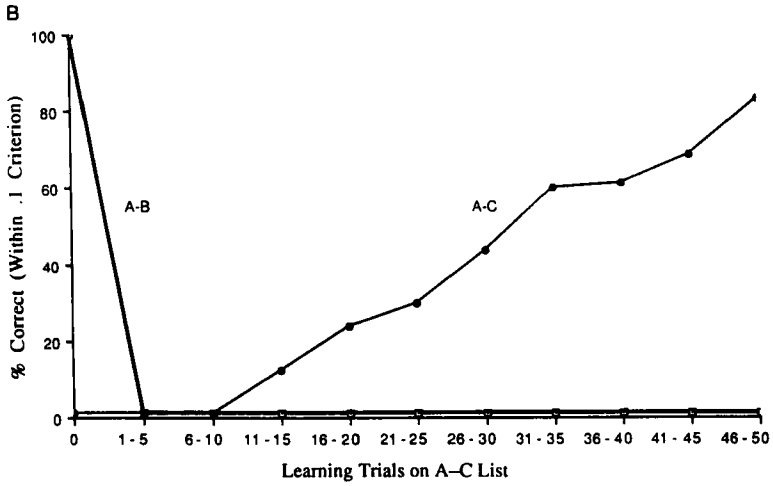
Even by the lax best match criterion, performance on the A–B list was drastically disrupted by A–C training (Figure 10C). After three learning trials on the A–C list, which is sufficient to produce only about 20% correct responses on this list, performance on the A–B list fell from 100 to 0% correct, and remained at or near 0% thereafter.

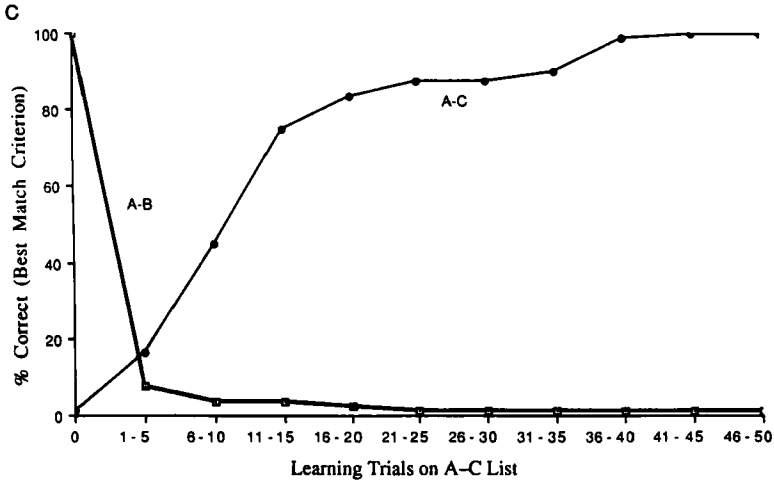
In contrast, Barnes and Underwood found that even after an amount of A–C training sufficient to yield over 90% correct responses on the A–C list, subjects still performed at a level of better than 50% correct on the A–B list. Of course, our simulation is at best a rough one; nevertheless, it is striking that A–C training insufficient to produce significant A–C learning reduces performance on the A–B list virtually to zero.

The network's errors are similar to those observed for the arithmetic model. Specifically, after training on the A–C list the network's response to each A stimulus closely resembles the corresponding C response, not only in the presence of the A–C context pattern, but also in the presence of the A–B context.

Fig. 10. A, Performance of the RI network on the A–B and A–C lists as a function of number of learning trials on the A–C list, assessed according to the right side of .5 criterion. B, Performance of the RI network on the A–B and A–C lists as a function of number of learning trials on the A–C list, assessed according to the within .1 criterion. C, Performance of the RI network on the A–B and A–C lists as a function of number of learning trials on the A–C list, assessed according to the best match criterion.


A
% Correct (Right Side of .5 Criterion)
100
80
60
40
20
0
A-B
A-C
0
1-5
6-10
11-15
16-20
21-25
26-30
31-35
36-40
41-45
46-50
Learning Trials on A–C List


B
% Correct (Within .1 Criterion)
100
80
60
40
20
0
A-B
A-C
0
1-5
6-10
11-15
16-20
21-25
26-30
31-35
36-40
41-45
46-50
Learning Trials on A–C List


C
% Correct (Best Match Criterion)
100
80
60
40
20
0
A-C
A-B
0
1-5
6-10
11-15
16-20
21-25
26-30
31-35
36-40
41-45
46-50
Learning Trials on A–C List

VI. Generality of the Interference Problem

Our arithmetic and retroactive interference results raise a variety of questions. Would the disruption of old knowledge by new learning be reduced or eliminated if the networks had more hidden units, or if we used a different representation for the stimuli and responses? Does the effect occur in learning of other sorts of information? In other words, under what sets of conditions does new learning catastrophically disrupt old learning?

At present we have no definitive answer to this question. In fact, as we discuss in a later section, it may not be possible to arrive at a general answer, given the current level of development of the connectionist framework. Hence, the results presented in the following section are intended not as a systematic analysis of the generality of the interference phenomenon, but rather as a series of demonstrations that the catastrophic interference phenomenon is not limited to the particular parameter settings, and so forth, used in the simulations we have reported thus far.

A. Generality across Simulation Parameters

We have carried out a variety of manipulations within the context of the RI simulation, in an attempt to determine whether the severity of the interference could be reduced. In reporting the results of these manipulations we will present the percentage of correct responses on the A–B list after amounts of A–C training sufficient to yield 25, 50, and 75% correct responses on the A–C list. For purposes of comparison, we summarize the Barnes and Underwood (1959) results: For example, according to the tabulation, after an amount of A–C training sufficient to yield 50% correct responses on the A–C list, performance on the A–B list was about 80% correct.

No A–C Training	25% Correct	50% Correct	75% Correct
100	>83	80	70

The figures are approximate, because Barnes and Underwood did not assess A–B performance when A–C performance was at precisely 25, 50, or 75% correct. One A–C learning trial yielded 43% correct responses on the A–C list (and 83% correct on the A–B list), whereas five trials resulted in 78.6% correct responses on the A–C list (and 67% correct on the A–B list). Thus, we estimated A–B performance at the 50 and 75% A–C

TABLE II

PERCENTAGE CORRECT ON THE A–B LIST BY THE RIGHT SIDE OF .5 CRITERION AFTER VARIOUS AMOUNTS OF A–C LEARNING, AS A FUNCTION OF THE NUMBER OF HIDDEN UNITS

Number of hidden units	Amount of A–C learning			
	No A–C training	25% Correct	50% Correct	75% Correct
10	100	0	0	0
20	100	0	0	0
50	100	0	0	0
70	100	0	0	0
100	100	0	0	0

performance levels by interpolation. No specific estimate of A–B performance could readily be made for the 25% A–C performance level, because a single A–C learning trial led to A–C performance of considerably better than 25% correct. Hence, we note in the table only that A–B performance at this level of A–C learning would presumably be greater than the 83% correct obtained after one A–C learning trial.

1. Number of Hidden Units

We first manipulated the number of hidden units in the network to determine whether the interference effect might be smaller with either more or fewer hidden units than the 50 used in the initial runs. With the learning rate held constant at .25, networks with 10, 20, 70, and 100 hidden units were trained and tested in the manner described above.

Table II summarizes the performance of the networks assessed with the right side of .5 criterion, and Table III presents the results obtained with the best match criterion.[1] Results for the within .1 criterion are the same as those for the right side of .5 criterion, and so are not presented separately. Data from the initial runs with 50 hidden units are included in the tables for purposes of comparison.

It is evident from the tables that manipulating the number of hidden units had no discernible effect on the extent to which A–C learning disrupted performance on the A–B list. Even with 100 hidden units, ten

[1]For these results, and for the other findings we will present, percentage correct on the A–C list was computed according to the same criterion used to assess performance on the A–B list. Thus, for the data shown in Table II, the percentage correct on the A–C list was calculated according to the right side of .5 criterion, and for the results in Table III A–C performance was assessed with the best match criterion.

TABLE III

PERCENTAGE CORRECT ON THE A–B LIST BY THE BEST MATCH CRITERION AFTER VARIOUS AMOUNTS OF A–C LEARNING, AS A FUNCTION OF THE NUMBER OF HIDDEN UNITS

Number of hidden units	Amount of A–C learning			
	No A–C training	25% Correct	50% Correct	75% Correct
10	100	6	13	0
20	100	6	0	6
50	100	13	0	0
70	100	0	0	0
100	100	6	0	0

times the number needed to learn the lists concurrently, performance on the A–B list was, by any criterion, at or near zero after A–C training sufficient to yield performance of 25% correct or better on the A–C list. These results, like those from the initial runs, stand in stark contrast to the findings of Barnes and Underwood with human subjects.

Because varying the number of hidden units did not affect the severity of interference, unless otherwise indicated we used networks with 50 hidden units (as in our initial RI simulation) to explore the effects of other manipulations.

2. *The Learning Rate Parameter*

We next manipulated the value of the learning rate parameter, which determines the magnitude of weight changes during learning. To supplement the initial results obtained with a learning rate of .25, we trained networks at learning rates of 1.00, .75, .50, .10, .05, .01, and .001. Results for the right side of .5 criterion are presented in Table IV. At all learning rates, performance on the A–B list was reduced to 0% correct by the time a performance level of 25% correct was achieved on the A–C list. Results for the more stringent within .1 criterion were of course the same.

The results for the best match criterion, presented in Table V, seem more encouraging. Increasing the learning rate above the original value of .25 did not lessen the severity of interference: At learning rates above .25, A–B performance was reduced virtually to zero by A–C training sufficient to yield performance of 25% correct or better on the A–C list. However, the severity of interference was lessened somewhat by reducing the learning rate to levels below .25. After A–C training sufficient to produce 25% correct on the A–C list, A–B performance was only 13%

TABLE IV

PERCENTAGE CORRECT ON THE A–B LIST BY THE RIGHT SIDE OF .5 CRITERION AFTER VARIOUS AMOUNTS OF A–C LEARNING, AS A FUNCTION OF THE LEARNING RATE

	Amount of A–C learning			
Learning rate	No A–C training	25% Correct	50% Correct	75% Correct
1.00	100	0	0	0
.75	100	0	0	0
.50	100	0	0	0
.25	100	0	0	0
.10	100	0	0	0
.05	100	0	0	0
.01	100	0	0	0
.001	100	0	0	0

correct with a learning rate of .25, but 38% correct with the .001 rate. Although these results for low learning rates appear encouraging at first, several considerations suggest that in fact they are something less than promising.

a. Laxness of the Best Match Criterion. In the first place, the disruption of A–B performance is less than total only by the best match criterion. However, the best match criterion may well be too lax. By this

TABLE V

PERCENTAGE CORRECT ON THE A–B LIST BY THE BEST MATCH CRITERION AFTER VARIOUS AMOUNTS OF A–C LEARNING, AS A FUNCTION OF THE LEARNING RATE

	Amount of A–C learning			
Learning rate	No A–C training	25% Correct	50% Correct	75% Correct
1.00	100	6	0	0
.75	100	0	0	0
.50	100	0	6	6
.25	100	13	0	0
.10	100	13	0	6
.05	100	25	25	19
.01	100	31	31	19
.001	100	38	31	25

TABLE VI

Output Pattern Example Counted Correct by the Best Match Criterion, the Target Pattern, and a Different, Better-Matching Pattern

Activation pattern	Output unit									
	1	2	3	4	5	6	7	8	9	10
Target pattern	.10	.90	.90	.10	.10	.10	.90	.90	.90	.90
Actual pattern	.02	.95	.45	.22	.66	.10	.82	.24	.88	.75
Example of a better match	.10	.90	.10	.10	.90	.10	.90	.10	.90	.90

criterion an output pattern generated by the network is considered correct if it matches the target output pattern more closely than it matches the pattern for any of the other B or C responses. Patterns that do not happen to occur as responses in the task are not considered.

Consider, for example, the output pattern presented along with the target pattern in Table VI. In this output pattern, which was generated by one of our networks in response to an A–B test item after some A–C training, two units that should be on (i.e, units 3 and 8) have activation values less than .5, and one unit that should be off (i.e., unit 5) has an activation level greater than .5. Nevertheless, the output was counted correct by the best match criterion, because it conformed to the target output pattern better than to any of the other patterns occurring as responses in the task. However, among the possible output patterns that did *not* happen to occur as responses in the task, there are 17 that match the network's output better than the target pattern does. (One such pattern is presented as an example in Table VI.) In using the best match criterion we assume that these better matches could somehow be ruled out as possible responses.

It is difficult to imagine how the better matches could be excluded unless there were a postprocessor external to the network that knew the set of responses that occurred in the task and selected the one that best matched the network's output. However, it is not at all clear that this sort of postprocessor could be implemented straightforwardly, because the postprocessor would itself presumably have to learn the B and C responses (sequentially). Therefore, if the postprocessor is assumed to be a network of the sort under consideration here, the interference problem would presumably arise with respect to its learning of the responses. In addition of course, if we were to postulate that the postprocessor is a different sort of mechanism that is not subject to severe interference dur-

ing sequential learning, we would have to specify what sort of mechanism the postprocessor is, and why this sort of mechanism is used only for postprocessing and not to perform the task as a whole. At best, then, the use of the best match criterion carries with it a promise to specify, at some point in the future, how the correct response could be generated from the network's output. The above discussion suggests that this promise may prove difficult to fulfill.[2]

b. Severity of Interference. The second reason that the results with low learning rates cannot be considered particularly encouraging is that even if we adopt the lax best match criterion, the amount of interference observed in the simulations is still much greater than that obtained by Barnes and Underwood. For example, whereas Barnes and Underwood's subjects were approximately 70% correct on the A–B list after A–C training sufficient to yield performance of 75% correct on the A–C list, the best performance achieved in the simulations at this level of A–C learning was 25% correct (with the .001 learning rate).

c. Amount of Training on the A–B List. Third, if we adopt the best match criterion as the appropriate means of assessing the network's performance, then it must be concluded that we overtrained the networks on the A–B list. Barnes and Underwood trained subjects on the A–B list to a criterion of one perfect recall trial. Therefore, in simulating the Barnes and Underwood study we should terminate A–B training when a network first reaches a performance level of 100% correct. If we are using the best match criterion to assess the network's performance during A–C training, then we should also use this criterion to decide when to terminate the initial A–B training. In other words, if we assume that a correct response can be generated whenever the network's output matches the target out-

[2]It might be suggested that in some instances a postprocessor that had learned the specific responses occurring in the task would not be necessary to clean up output patterns generated by the initial network. For example, in our arithmetic network each number in the range 0–9 was represented by turning on three output units (see Table I). Hence, an output such as [0 1 0 0 0 0 1 1 1 0 0 0] could presumably be cleaned up (by turning off the second output unit) by a postprocessor that knew about the form of numerical representations but did not know the specific numbers occurring as responses in any particular task. Similarly, in the case of the RI simulation it might be suggested that the set of possible meaningful output patterns is somehow constrained so that certain patterns could be eliminated as potential responses even in the absence of knowledge of the responses that occurred in the task. However, given that the network must presumably be capable of representing whatever response items could be presented in a task, the set of meaningful network outputs would then be much larger than the set of responses that happened to occur in any particular task. Hence, it is not clear how a postprocessor could rule out all output patterns that did not happen to occur in the task without learning the specific patterns that did occur.

put pattern better than the pattern for any of the alternative responses in the task, then we should stop A–B training once the network reaches perfect performance by this criterion. However, the data presented thus far have come from simulations in which the networks were trained on the A–B list until performance was perfect according to the much more stringent within .1 criterion. Typically, training to perfect performance by the within .1 criterion requires many more learning trials than are needed to achieve 100% correct responses by the best match criterion. For example, at a learning rate of .01, the network took an average of 110 learning trials to reach 100% correct by the best match criterion, but 720 trials to achieve perfect performance by the within .1 criterion. Thus, from the perspective of the best match criterion, we gave the networks many more A–B learning trials than was appropriate.

To determine whether this A–B overtraining might have affected the amount of interference observed with the best match criterion, we trained a network to perfect performance on the A–B list according to the best match criterion and then looked at the effects of A–C training, again using the best match criterion. The network was trained with a learning rate of .001, the rate that yielded the best retention of A–B learning in our initial runs (i.e., 38, 31, and 25% correct after A–C training sufficient to yield A–C performance of 25, 50, and 75% correct, respectively). The results were quite clear: When the network was trained only until it first reached perfect performance by the best match criterion, A–B performance was reduced to zero by A–C training sufficient to yield A–C performance of 25% correct or better.

d. Number of Trials Needed to Learn the Lists. The final reason that low values of the learning rate parameter do not hold significant promise for resolving the interference problem is that at low learning rates the networks require an unreasonably large number of trials to learn the A–B and A–C lists. The Barnes and Underwood interference results were obtained under conditions in which subjects required an average of 10.36 trials to learn the A–B list. Therefore, a simulation of the subjects' performance can be considered successful only if the simulation shows a comparable amount of interference under conditions that allow learning to occur in a comparable number of trials.

For learning rates in the range .25–.75, the number of trials needed to learn the list can be considered at least roughly comparable to the 10.36 required by Barnes and Underwood's subjects (see Table VII). However, at learning rates of .01 and .001, which were the values that yielded the least severe interference, the number of trials needed to reach criterion was clearly incommensurate with the number required by human learn-

TABLE VII

TRIALS TO CRITERION ON THE A–B LIST AS A FUNCTION OF CRITERION AND LEARNING RATE

Learning rate	Criterion		
	Best match	Right side of .5	Within .1
1.00	43	43	48
.75	15	23	28
.50	15	20	28
.25	18	28	43
.10	28	48	78
.05	35	73	150
.01	110	308	720
.001	638	2,925	7,225

ers. For example, at a learning rate of .001, the network took an average of 638 trials to reach 100% correct by the best match criterion, 2,925 trials by the right side of .5 criterion, and 7,225 trials by the within .1 criterion. The same point applies to the learning of the A–C list: At low learning rates the number of A–C learning trials required by our networks to achieve any given level of performance was far greater than the number needed by Barnes and Underwood's subjects.

Thus, values of the learning rate parameter sufficiently high to allow lists to be learned in a reasonable number of trials result in total disruption of A–B performance by A–C learning, and at parameter settings at which the interference is less severe (although still too severe to be considered comparable to that obtained by Barnes and Underwood), the number of trials required to learn the lists is one or more orders of magnitude greater than the number needed by Barnes and Underwood's subjects. In other words, a single setting of the learning rate parameter cannot satisfy both of the constraints imposed by the Barnes and Underwood results.

This problem is particularly discouraging in that it forecloses what might appear to be a promising approach toward reducing the interference to levels comparable to those reported by Barnes and Underwood. In particular, one might have imagined that although interference was still too severe in the networks at the .001 learning rate, the use of even lower learning rates might reduce the interference to acceptable levels. However, whether or not learning rates below .001 would in fact yield substantially reduced interference, it is clear that at these extremely low learning rates the number of trials required to learn the lists would be even more unreasonable than at the .001 rate.

e. A Multiple Rehearsals Account. It might be suggested that the large numbers of trials our networks require at learning rates of .001 or below may not be as unreasonable as we have supposed. In particular, it might be argued that subjects in the Barnes and Underwood study may have rehearsed each stimulus–response pair many times on each learning trial. If each rehearsal of an item is considered to be a separate presentation, the number of presentations of each item may have been far greater than 10.36, the average number of learning trials.

This argument suffers from a number of problems; here we mention just one. At the potentially promising learning rates below .001 (as well as at the .001 rate and perhaps even the .01 rate), the number of learning trials the network would require to learn the lists is too great to be explained in terms of rehearsal. Even when performance was assessed by the lax best match criterion, the network required 638 presentations of each A–B pair to learn the A–B list at the .001 learning rate. In order to suggest that through rehearsal, Barnes and Underwood's subjects experienced an equivalent number of presentations in the course of 10.36 learning trials, we would have to assume that the subjects rehearsed each stimulus–response pair approximately 62 times on each trial, for a total over the eight pairs of almost 500 rehearsals per trial. Given that a trial in the Barnes and Underwood study took about 36 secs, subjects would have to have rehearsed at a rate of about 14 items per second, which would not appear to be reasonable. If we assess performance by the right side of .5 criterion, or the within .1 criterion, the number of rehearsals we must posit becomes even more unreasonable (approximately 280 rehearsals of each item on each trial for the right side of .5 criterion, and about 700 for the within .1 criterion). And, of course, it must be borne in mind that by any account the interference was still too great with the .001 learning rate; if lower learning rates were used in an attempt to reduce the interference, still more rehearsals would have to be posited.

Given, then, that very low learning rates do not appear to be the answer to the interference problem, we continued to use the original learning rate of .25 when manipulating other network variables.

3. Overtraining on the A–B List

In exploring the effects of manipulating the learning rate parameter, we found that disruption of A–B performance due to A–C learning was less severe when training on the A–B list was continued until perfect performance was achieved according to the within .1 criterion than when A–B training was terminated when performance was perfect by the best match criterion. Hence, we next attempted to determine whether still further training on the A–B list might substantially reduce or perhaps even elimi-

nate the disruption of A–B performance resulting from A–C learning. In human learners overtraining on the A–B list (i.e., training beyond the first perfect recall trial) leads to improved performance on tests of A–B retention after interpolated A–C training (e.g., Postman, 1962), and it seems intuitively likely that with sufficient overtraining, little or no decrement in A–B performance would be observed following moderate amounts of A–C learning.

We first trained a network on the A–B list until performance was perfect according to the within .1 criterion, which required an average of 42.5 trials. The network then underwent 1,000 additional A–B learning trials. This massive overtraining reduced the average squared error per output unit from .0013 to less than .000001. The network was then trained on the A–C list, as in our previous simulations.

The results are presented in Table VIII for each of the three criteria. By any of the criteria, A–B performance was reduced virtually to zero by A–C training sufficient to yield performance of 25% correct or better on the A–C list. Thus, extensive overtraining involving over 20 times the number of learning trials required to reach the initial learning criterion failed to prevent the drastic disruption of A–B performance by small amounts of A–C training.

4. *Freezing of Weights*

We next attempted to determine whether the disruption of A–B performance by A–C learning could be reduced to reasonable proportions if the connection weights established during A–B training were not allowed to change during A–C training.

A network with 100 hidden units was used. As in all of the simulations, each connection weight was initialized to a small random value (Rumelhart, Hinton, & Williams, 1986). During A–B training, the learning algorithm was allowed to alter the weights on the connections to and from 50

TABLE VIII

PERCENTAGE CORRECT BY EACH CRITERION ON AN OVERTRAINED A–B LIST AFTER VARIOUS AMOUNTS OF A–C LEARNING

	Amount of A–C learning			
Criterion	No A–C traning	25% Correct	50% Correct	75% Correct
Best match	100	6	0	0
Right side of .5	100	0	0	0
Within .1	100	0	0	0

of the hidden units. The connection weights for the other 50 hidden units were fixed at their initial values.

The network was trained on the A–B list until performance was perfect according to the within .1 criterion. The connection weights for the 50 hidden units involved in the A–B learning were then frozen. That is, the weights were fixed at the values established by the A–B training and were not allowed to change during A–C training. Rather, A–C training was allowed to modify the connection weights for the 50 hidden units that were not involved in A–B learning (i.e., the weights that were fixed at their initial values during A–B learning). All weights, whether frozen or not, contributed to the generation of outputs when inputs were presented to the network.

The results are shown in Table IX. It is obvious from the table that freezing the weights failed to prevent the catastrophic disruption of A–B performance by A–C training: By any criterion, percentage correct on the A–B list was at zero after A–C training sufficient to yield A–C performance of 25% correct or better.

At first glance, this result might seem nonsensical. If the weights established by A–B learning were fixed during A–C training, how could performance on the A–B list have been disrupted? The answer is that the weights established by the A–C training, as well as those resulting from A–B training, come into play whenever an input is presented to the network. Thus, the combined effect of the weights established by A–B training and the weights established during A–C training was to produce incorrect patterns of activation on the output units when the A–B list was tested after some A–C training.

Freezing the weights established by A–B learning not only failed to reduce the disruption of A–B performance by A–C training, but also substantially impaired learning of the A–C list. When weights were not fro-

TABLE IX

PERCENTAGE CORRECT BY EACH CRITERION ON THE A–B LIST AFTER VARIOUS AMOUNTS OF A–C LEARNING WHEN WEIGHTS ESTABLISHED DURING A–B LEARNING WERE FROZEN DURING A–C TRAINING

	Amount of A–C learning			
Criterion	No A–C training	25% Correct	50% Correct	75% Correct
Best match	100	0	0	0
Right side of .5	100	0	0	0
Within .1	100	0	0	0

zen, the average number of A–C learning trials needed to achieve perfect A–C performance according to the within .1 criterion was 74.5 for networks with 50 hidden units, and 69 for networks with 100 hidden units. In contrast, when the weights established by A–B training were frozen, the average number of trials to criterion on the A–C list was 151. Under all of these conditions, training the A–B list to the same criterion required roughly 40 trials. Thus, whereas Barnes and Underwood (1959) found that rate of learning was approximately the same for the A–B and A–C lists, freezing the connection weights established by A–B training resulted in much slower A–C than A–B learning.

5. *Target Activation Values of 0 and 1*

In the work reported thus far, we have followed Rumelhart, Hinton, and Williams (1986) in using .1 rather than 0 as the target activation value for an output unit that should be off, and .9 rather than 1.0 for an output unit that should be on. That is, we have computed the error for an output unit by subtracting the unit's activation value from .1 if the unit should be off, and from .9 if the unit should be on. These error values (among other factors) determine the direction and magnitude of the weight changes that occur during training.

The use of .1 and .9 as the target activation values has the effect of preventing the connection weights from becoming very large as training progresses, and hence of preventing output activation levels from approaching 0 or 1.0. In fact, if presentation of an input produces an activation level of greater than .9 in an output unit that should be on, weight changes that tend to decrease the activation level will be made. Similarly, if an input produces an activation level of below .1 in an output unit that should be off, weight changes that tend to increase the activation level will occur.

TABLE X

PERCENTAGE CORRECT BY EACH CRITERION ON THE A–B LIST AFTER VARIOUS AMOUNTS OF A–C LEARNING AT TARGET ACTIVATION LEVELS OF 0 AND 1.0

	Amount of A–C learning			
Criterion	No A–C training	25% Correct	50% Correct	75% Correct
Best match	100	0	6	0
Right side of .5	100	0	0	0
Within .1	100	0	0	0

It might be suggested, therefore, that the use of .1 and .9 as the target activation levels actually prevents the A–B list from being thoroughly learned. Hence, we explored the effects of A–C training on A–B performance in simulations using 0 and 1.0 as the target activation levels.

The results are presented in Table X. It is apparent that the use of 0 and 1.0 as the target output activation levels did nothing to reduce the disruption of A–B performance by A–C training.

Further, this manipulation drastically impaired learning of the A–C list. In both runs with target activation levels of 0 and 1.0, the A–C list failed to reach perfect performance by the within .1 criterion in 1,000 A–C learning trials; rather, the networks became stuck in local minima at which some items were correct but others were incorrect.[3]

6. *Local Representations for Stimuli and Responses*

Finally, we sought to determine whether the catastrophic interference occurring in our RI network might be specific to the particular representations we chose for stimuli, responses, and list contexts. In the simulations

[3]This failure to learn the A–C list may reflect the nature of the back-propagation learning algorithm. When the activation of an output unit is very close 0 or 1.0, the learning algorithm leaves the weights on connections to that unit virtually unchanged, regardless of whether the output activation level is near the correct or the incorrect extreme. For example, if an input pattern produces an activation level of .999 in an output unit, the weights on connections to that unit will be changed very little, regardless of whether the target activation level for that unit is 1.0 or 0. In calculating weight changes, the difference between the target and actual output values is multiplied by the derivative of the logistic activation function at the point corresponding to the activation level of the output unit. This derivative is given by $a(1 - a)$, where a is the activation of the output unit. As the activation level approaches 0 or 1.0, the derivative approaches 0. Thus, regardless of the difference between an output unit's target and actual activation levels, the weight changes for connections to that unit will approach 0 as the unit's activation level approaches 0 or 1.0. If, then, the activation of an output unit is near the incorrect extreme (i.e., 0 for a unit that should be on, and 1.0 for a unit that should be off), the learning algorithm may fail to correct the error. (The use of .1 and .9 as the target activation values usually avoids this problem, because output activation levels are actively prevented from approaching the extreme values of 0 and 1.0.) With target activation values of 0 and 1.0, training on the A–B list created a situation in which output activation levels for some A–C items were very close to the incorrect extreme. That is, at the beginning of A–C training the network generated, for some A–C items, outputs in which one or more units with target activation levels of 1.0 had activation levels very close to 0, and/or one or more units with target activation levels of 0 had activation levels very close to 1.0. Extensive training failed to alter some of these incorrect outputs, with the result that good performance on the A–C list was not achieved. (This phenomenon does not occur at the beginning of A–B training, because the weights in the network are small and random, so that an input pattern is very unlikely to generate either a very high or very low activation level in an output unit.)

reported thus far, we used distributed representations for stimuli and responses. Each stimulus item was represented by a random pattern of activation across 10 input units, so that the various stimulus items overlapped with respect to the units that were activated, or on. For example, for both of the stimuli shown below, the second, sixth, and ninth input units were on:

[0 1 0 0 1 1 0 0 1 1]

[1 1 1 0 0 1 1 1 1 0]

Thus, when either of these stimuli was presented, the second, sixth and ninth input units sent activation to the hidden unit layer. (Overlap across input patterns with respect to units that are off is less significant, because units that are off do not send activation to other units.) Similarly, there was overlap across response terms and between the A–B and A–C context patterns.

It might be imagined that interference could be reduced by the use of representations in which the various stimuli, responses, and list contexts are more distinct from one another. To explore this possibility we stimulated the Barnes and Underwood (1959) study using "local" representations for stimuli, responses, and list contexts. Each of the eight stimulus terms was represented by turning on a single unit in an eight-element stimulus vector. For example, one stimulus term was represented by the pattern [1 0 0 0 0 0 0 0], a second by the pattern [0 1 0 0 0 0 0 0], and so forth. Thus, there was no overlap between stimulus patterns with respect to the units that were on. Similarly, each of the 16 responses (i.e., eight B and eight C responses) was represented by turning on a single unit in a 16-element output vector. List context was represented by a two-element vector with one unit turned on for the A–B context (i.e., [1 0]), and the other unit turned on for the A–C context (i.e., [0 1]). Note that even with local representations for stimuli, responses, and list contexts, the representations of stimulus–response associations are nevertheless distributed across many different weighted connections between input and hidden units and between hidden and output units.

With the local representations all 16 items were readily learned under concurrent training conditions, although learning was somewhat slower than with distributed representations.

In the sequential training runs the network was first trained on the A–B list until performance was perfect by the within .1 criterion. As in concurrent training, learning proved to be slower with local than with distributed representations: The average number of trials required to learn the list

TABLE XI

PERCENTAGE CORRECT BY EACH CRITERION ON THE A–B LIST AFTER VARIOUS AMOUNTS OF A–C LEARNING, WITH LOCAL REPRESENTATIONS FOR STIMULI, RESPONSES, AND CONTEXT

Criterion	Amount of A–C learning			
	No A–C training	25% Correct	50% Correct	75% Correct
Best match	100	13	0	0
Right side of .5	100	0	0	0
Within .1	100	0	0	0

was 140 (compared to 42.5 for an otherwise comparable network with distributed representations).[4]

After completion of A–B training, the network was trained on the A–C list. The impact of this training on A–B performance is shown in Table XI. It is apparent that A–B performance is once again catastrophically disrupted by A–C training sufficient to yield performance of 25% correct or better on the A–C list. Thus, the severe interference we have observed repeatedly with the distributed representations is not specific to these representations.

It is important to point out that in using local representations we did not eliminate overlap between A–B and A–C input patterns with respect to the units that were on. In the A–B, A–C paradigm used in the Barnes and Underwood study, the stimulus terms (e.g., *dax*) are the same in the two lists. Consequently, in simulating this study we used the same eight stimulus patterns in our A–B and A–C lists. For example, one of the A–B input patterns was constructed by pairing the stimulus pattern [1 0 0 0 0 0 0] with the A–B context pattern [1 0], yielding the full input

[4]The slower learning with local representations may reflect the fact that with distributed but not local representations there are regularities across items in relationships between input and output unit states. Even when input and output patterns are randomly generated, as in our RI simulations with distributed representations, there are likely to be such regularities by chance, as long as the number of patterns in the training set is considerably less than the total number of possible patterns. In our A–B list with distributed representations, for example, when the first and second input units were on, the second output unit was always on, and the sixth was always off. Regularities of this sort may be exploited by the network in learning the items. (Indeed, the capturing of such regularities is the basis for so-called automatic generalization.) With the local representations in our RI network, however, there are no regularities across items to be captured (with the exception of those following from the fact that all of the output units were usually off).

pattern [1 0 0 0 0 0 0 0 1 0]. The input pattern for the A–C item with the same stimulus term was generated by pairing the same stimulus pattern with the A–C context pattern [0 1], yielding [1 0 0 0 0 0 0 0 0 1]. Thus, for both the A–B item and the corresponding A–C item, the first input unit was turned on.

A representational format that might have greater potential for reducing interference is one in which corresponding A–B and A–C input patterns are completely nonoverlapping. For example, we might represent a particular stimulus term in the A–B list context by pattern 1 shown below, and the same stimulus term in the A–C context by the nonoverlapping pattern 2:

1. [1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
2. [0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

In a representation of this sort a single activated unit represents the combination of a particular stimulus term with a particular context (e.g., activation of unit 1 represents *dax* in the A–B list context, and activation of unit 2 represents *dax* in the A–C context). Unfortunately, this sort of representation cannot be considered adequate, because it entirely fails to capture the fact that the same stimulus terms are used in the two lists. Hence, the representations cannot distinguish an A–B, A–C procedure from an A–B, C–D procedure in which the stimulus terms are completely different in the two lists. Given that the performance of human learners varies substantially as a function of whether the stimulus terms are the same or different in the two lists (e.g., retroactive interference is far greater with the A–B, A–C procedure than with the A–B, C–D procedure), the inability to distinguish the two procedures constitutes a fatal shortcoming of the completely nonoverlapping representations.

7. *Summary of RI Results*

The catastrophic disruption of A–B performance by A–C training observed in our original RI simulation was not eliminated by varying the number of hidden units or the learning rate parameter, by massive overtraining on the A–B list, by freezing the weights established by A–B learning, by using 0 and 1 rather than .1 and .9 as the target output activation values, or by using an alternative representation for the stimuli, responses, and list contexts. In fact, we have failed to find any set of circumstances in which a network's performance on the A–B list after A–C training even approached the levels observed by Barnes and Underwood (1959) with human learners. For example, whereas Barnes and Under-

wood obtained A–B performance of about 80% correct after A–C training sufficient to yield 50% correct responses on the A–C list, the best A–B performance we were able to obtain with comparable amounts of A–C training was 31% correct. Furthermore, to achieve 31% correct we had to use a learning rate of .001, which results in unacceptably slow learning; we had to assess performance according to the best match criterion, which is probably too lax; and we had to overtrain the network massively on the A–B list.

Barnes and Underwood (1959), discussing the function relating A–B performance to amount of A–C training in their study, state that

> A hyperbolic equation fitted to this curve predicts an asymptote at 3.46 [correct out of 8 on the A–B list]. Thus, it does not seem that all items would be extinguished, even with an extremely large number of trials on A–C. This conforms to the fact that forgetting in the RI paradigm has not been shown to be complete even with very high degrees of interpolated learning (Barnes & Underwood, 1959, p. 102).

Similarly, in a review of interference research, Postman and Underwood (1973, p. 20) state that "regardless of the degree of IL [i.e., interpolated learning of the second list], unlearning [of first-list responses] is virtually never complete and in fact rarely exceeds about 50%." In contrast, we have been unable to achieve reductions in A–B performance *as small as* 50%. In fact, in the vast majority of our simulations A–B performance was reduced to virtually 0% correct by A–C training sufficient to yield only two out of eight correct responses on the A–C list.

Of course, it is conceivable that some untried combination of parameter settings, representational formats, and so forth would yield simulation results comparable to those reported by Barnes and Underwood (1959). However, we have been unable to find any such combination in spite of significant efforts to do so.

B. Generality of the Interference Problem across Domains

We have not yet systematically explored the generality of the interference problem across types of information to be learned. However, it appears that the problem may occur in a variety of domains. In the first place, we have observed catastrophic interference not only in the RI simulation, but also in the simulations of arithmetic fact learning. Furthermore, other examples of the phenomenon have been reported. For example, Ratcliff (1988) has found severe interference in sequential training of "encoder" networks; Hinton and Plaut (1987) have reported interference effects resulting from sequential training of associations between random vectors in a situation somewhat different from our RI simulation; and

Sejnowski and Rosenberg (1987) have interpreted massed versus distributed practice effects in their NetTalk model in terms of disruption of previously acquired information by new learning. Indeed, Sutton (1986) argues that steepest-descent learning algorithms, such as the back-propagation algorithm, are especially prone to disruption of previous learning by new learning.

Finally, if we consider why the interference phenomenon occurs, it becomes clear that there is no reason to expect the phenomenon to be limited to a few specific situations or parameter settings. This point is discussed in the following section.

VII. Why Does New Learning Disrupt Old Knowledge?

In traditional cognitive models sequential learning does not pose any special problems. Consider, for example, a propositional network model of memory for arithmetic facts. In a model of this sort concepts are represented by individual nodes, and facts are represented by connecting nodes with labeled links that represent relationships among concepts (e.g., Anderson & Bower, 1973; Anderson, 1983).

Figure 11A presents (in a simplified form) an arbitrary portion of a propositional network representing addition facts. The fact 5 + 4 = 9, for example, is represented by the network structure connecting the nodes representing 5, 4, the addition operation, and 9. Also shown is the fact 6 + 4 = 10.

The learning of new facts involves the building of new propositional structures in the memory network. Thus, Fig. 11B shows the propositional network after the learning of a new fact: 7 + 4 = 11. Because each fact has a representation separate from the representations of other facts, the storing of the new fact representation does not in any way disrupt the representations of the previously learned facts (although ease of retrieval may be affected).

In a connectionist model with distributed representations the situation is quite different, because each connection weight is involved in responding to many different inputs. Thus, adjustment of weights to encode the desired response to a new input pattern will necessarily alter the network's response to other inputs as well. In many respects this is a desirable feature. It is, for example, the basis for so-called automatic generalization, in which a network, through training on some patterns, comes to respond appropriately to other (untrained) patterns. The disadvantage is that changing weights to encode a new piece of information may alter previously learned responses to other input patterns. This is what hap-

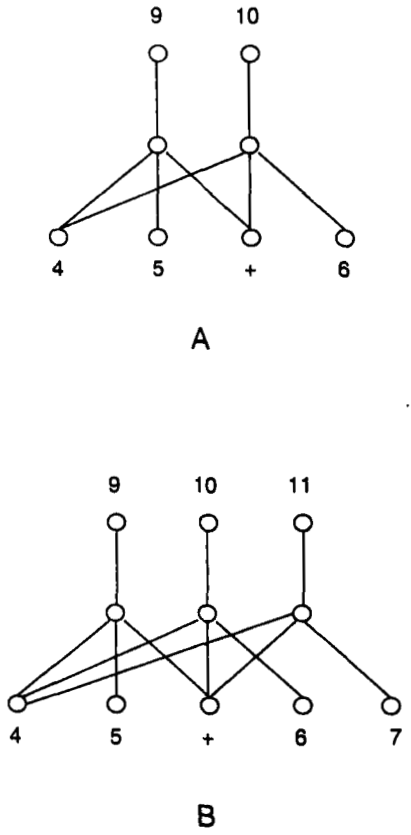
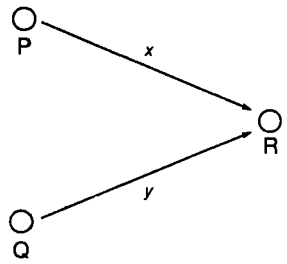


Fig. 11. A, Propositional network representation for some addition facts. B, The network after learning the new fact 2 + 3 = 5.

pened in our arithmetic and RI simulations: Weight adjustments during learning of the twos addition facts altered the previously learned responses to the ones facts, and weight adjustments during learning of the A–C list altered previously learned responses to the A–B items.

An example may help to clarify just why this interference effect occurs. Figure 12 depicts a very simple network consisting of two input units, P and Q, connected to a single output unit R. We will be concerned with training the network on the two patterns shown in the figure. In pattern 1, both input units are on, and the output unit is also on. In pattern 2, unit P is on, unit Q is off, and the output unit R is off.

For any network, the configuration of connection weights may be thought of as a point in a multidimensional space with a number of dimen-



	Input		Output
	P	Q	R
Pattern 1	1	1	1
Pattern 2	1	0	0

Fig. 12. A simple three-unit network, and two training patterns.

sions equal to the number of weights, and the position on each dimension representing the value of the corresponding weight. With our two-connection network, the weight configuration may be depicted as a point on a plane. In illustrating the performance of the network we will use the x axis to represent the weight on the P–R connection, and the y axis to represent the Q–R weight.

Consider pattern 1, with both input units on and the output unit also on. Where in weight space must the weight configuration be to produce the correct output for this pattern? Suppose that we want the output unit, which should be on for pattern 1, to have an activation of .9 or greater. The logistic function that transforms a unit's inputs into an activation level is such that a unit needs inputs summing to approximately 2.20 in order to achieve an activation of .9. Thus, we want the inputs to unit R to sum to at least 2.20.

The inputs to unit R are the activation of unit P times x (the weight on the P–R connection), and the activation of unit Q times y (the weight on the Q–R connection). For pattern 1 the input to unit R will be $(1x) + (1y)$, or simply $x + y$. Thus, for the network to respond correctly to pattern 1, the sum of x and y must be at least 2.20. Figure 13A shows the region in weight space satisfying this constraint. Any weight configuration in the dark shaded region will yield a correct response to pattern 1.

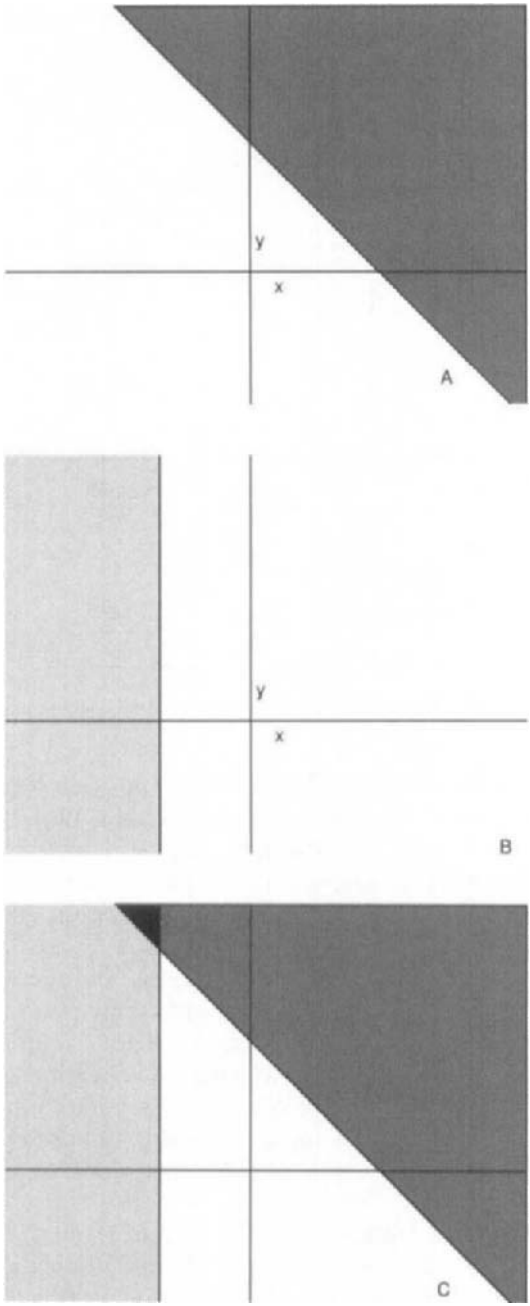


Fig. 13. A, Solution space for pattern 1. (The boundary of the solution region intersects the x and y axes at 2.20.) B, Solution space for pattern 2. (The boundary of the solution region intersects the x axis at -2.20.) C, Solution space for pattern 1 (dark shading), solution space for pattern 2 (light shading), and overall solution space (solid region).

Now consider pattern 2, in which P is on, Q is off, and R is off. Assuming that we want the activation level of R to be .1 or less for this pattern, the inputs to unit R must sum to -2.20 or less. For pattern 2, the input to R will be simply x, the weight from P to R. No input will be received from unit Q, because this unit is off in the input pattern. Thus, the solution space for pattern 2 is defined by the expression $x \leq -2.20$, indicated by the light shaded region in Fig. 13B.

It is easy to see that if we want the network to respond correctly to both pattern 1 and pattern 2, the weight configuration must be somewhere in the intersection of the solution spaces for the two patterns, as shown by the solid triangular area in Fig. 13C. If the weight configuration is in this region, which we will call the overall solution space, the network will respond correctly to both pattern 1 and pattern 2.

Figure 14A presents the results of concurrent training on patterns 1 and 2. The line in the figure shows the movement of the weight configuration over learning trials.[5] It is evident from the figure that the weight configuration moves rather directly to the overall solution space. The box at the bottom of the figure shows the network's output for the two patterns at the completion of training.

Each pattern to be learned may be thought of as pulling the weight configuration toward the solution space for that pattern. With concurrent training, both patterns are pulling at the same time; as a result, the weight configuration moves toward a region of weight space that is good for both patterns. More specifically, pattern 1 pulls diagonally upward and to the right, and pattern 2 pulls horizontally to the left; the weight configuration moves in the direction of the resultant of these two forces.

But what if we train sequentially? The line labeled 1 in Fig. 14B shows the movement of the weight configuration during training on pattern 1. The configuration moves directly toward the solution space for pattern 1. If the network is then trained on pattern 2, the weight configuration turns and moves directly toward the solution region for pattern 2, as shown by the line labeled 2 in the figure. At the completion of training, the network responds appropriately to pattern 2 but no longer gives a correct response to pattern 1. In fact, the network is farther from the solution space for

[5]In training the three-unit network we used a momentum term of zero, which has the effect of exaggerating the effects of sequential training without altering their fundamental character. Also, for purposes of expository convenience the units in the network did not have biases; 0 and 1 were used as the target activation values; connection weights were updated after every learning trial rather than after every item; and the weights were initially set to zero, rather than to small random values. None of these simplifications affects the points we illustrate with the network.

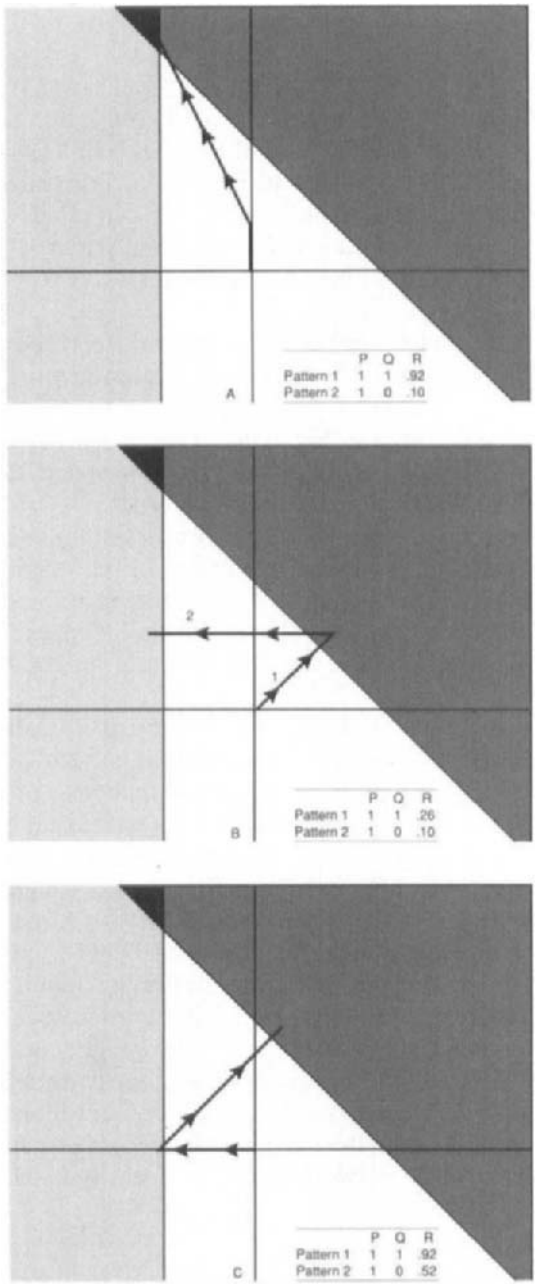


Fig. 14. A, Movement of the weight configuration over learning trials with concurrent training on patterns 1 and 2. B, Movement of the weight configuration over learning trials with sequential training on pattern 1 (segment labeled 1) and then on pattern 2 (segment labeled 2). C, Movement of the weight configuration over learning trials with sequential training on pattern 2 and then on pattern 1.

pattern 1 than when training began. The output for pattern 1, which should be about .9, and at the beginning of training was .5, is now less than .3.

With sequential training, the weight configuration is first pulled directly toward the solution space for pattern 1, and then directly toward the solution space for pattern 2. Once training on pattern 1 stops, this pattern is no longer pulling toward its solution space. There is nothing to prevent the weight configuration from being pulled out of the pattern 1 solution space by the training on pattern 2, and there is nothing to cause the weight configuration to move toward the overall solution space.

Similarly, if the network is trained first on pattern 2 and then on pattern 1, the weight configuration moves directly to the pattern 2 solution space, and from there directly to the pattern 1 solution region, as shown in Fig. 14C.

If training alternates between the two patterns—pattern 2, then pattern 1, then pattern 2, and so forth—the weight configuration will zigzag toward the overall solution space. (In fact in concurrent training the weight configuration zigzags in exactly this manner, although in very small steps, as long as the weights are adjusted after every pattern.) If training is strictly sequential, however, the resulting weight configuration is unlikely to be appropriate for the initially learned pattern.

The point of this example is not that new learning will always severely disrupt previously acquired information. The extent of the disruption will depend upon the shapes and relative positions of the solution spaces for the old and new patterns. In some situations new learning might conceivably pull the initially established weight configuration to a point in weight space that still permits good performance on initially learned material.

Rather, the point is that in sequential training there is nothing to prevent new learning from pulling the weight configuration out of the solution space for previously learned material, and there is nothing to ensure that the configuration moves to a region of weight space that is good for both old and new information. To put it another way, gradient-descent learning algorithms, such as the back-propagation algorithm used in the present simulations or the Boltzmann machine algorithm (Ackley *et al.*, 1985), are simply not designed to deal with situations in which the set of items to be learned changes over time.

VIII. Possible Solutions

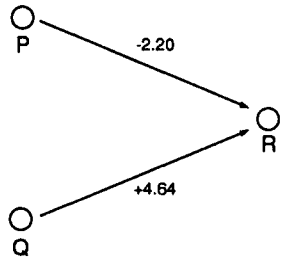
How might the interference problem be resolved? In this section we consider two possible approaches, suggesting that neither holds substantial promise.

A. Modification of Learning Algorithms

One might imagine that the learning algorithms could be modified in such a way that previously learned input–output mappings would be protected from alteration while new mappings are being encoded. However, this would probably be difficult to accomplish, for the following reason.

If a learning algorithm is to protect previously learned patterns, it must presumably have some way of identifying those patterns. If the algorithm cannot determine which particular input–output mappings have been learned, it is not clear how it could protect these mappings from disruption.

Unfortunately, there is no straightforward way of identifying the particular input–output mappings on which a network has previously been trained. A connectionist network with distributed representations is best thought of not as storing the specific patterns on which it has been trained, but rather as inducing from training on some patterns a function that will map any input pattern onto an output pattern. In other words, it is not the case that a network will produce an output only in response to a previously trained input pattern. Rather, the network will produce an



	Input		Output
	P	Q	R
Pattern 1	1	1	.92
Pattern 2	1	0	.10
Untrained	0	1	.99

Fig. 15. Weights established by training the PQR network concurrently on patterns 1 and 2, and the network's output to the two training patterns and one untrained pattern.

output for any arbitrary input pattern. And there is no straightforward way to determine from an output whether it represents (1) a learned response to a previously presented input, (2) a valid generalization from the trained patterns, or (3) a meaningless or incorrect response to a pattern that has not been trained. As a consequence, there is no way of knowing which particular input–output mappings should be preserved when new learning occurs and which can be changed without cost. Another way of saying this is that the network has no representation of a pattern as a whole.

This point may be clarified by referring once more to our PQR network. Figure 15 shows the connection weights established by training concurrently on patterns 1 and 2. These weights encode regularities at the level of individual units. For example, the weight from Q to R indicates that when unit Q is on, there is a strong tendency for unit R to be on. However, the weights do not encode information about which particular input patterns have been trained. That is, there is nothing in the weights to indicate that the network was trained on the input patterns [1 1] and [1 0], but not on, say, [0 1]. The network will produce an output for any pattern of activation across the input units. For example, presentation of the untrained input pattern [0 1] will produce an activation level of greater than .9 in the output unit, and in fact will activate the output unit more strongly than the trained pattern [1 1]. Thus, there is no way to tell by looking at the weight configuration that [1 1] has been trained, and [0 1] has not.[6]

The thrust of this example is that as long as information is represented in an exclusively distributed fashion, such that information at the level of whole concepts and relations among them is not explicitly represented, it is unclear how a learning algorithm could be devised that would protect against disruption just those patterns that have been previously learned.

B. Restricting New Learning to Adjustment of Context Weights

David Rumelhart (personal communication, November 1988) has suggested that the interference problem might be resolved by assuming that episodic memories are established primarily by modifying weights on connections from context units. Imagine a hypothetical A–B, A–C retroactive interference experiment in which subjects learn arbitrary associa-

[6]We would expect therefore that networks of the sort under consideration in this chapter would not perform well on recognition tasks. And indeed Ratcliff (1988) has found that multilayer encoder networks and McClelland and Rumelhart's (1985) autoassociative model perform poorly on recognition tasks.

tions between words (e.g., *perspective–banana* in the A–B list, and *perspective–quality* in the A–C list). Rumelhart's proposal assumes that for any two words that might be paired in the experiment, an association has previously been established. With respect to the above examples, the proposal would assume that in some previous context *perspective* and *banana* had been associated, and that in some other context *perspective* and *quality* had been associated.

These preexperimental associations could be established in a network that maps context plus word input patterns onto word output patterns (see Fig. 16). Specifically, the preexperimental associations would be established by training the network to map particular word inputs onto particular word outputs in particular contexts (e.g., map *perspective* in context *x* onto *abacus*), such that across the various contexts all possible word–word mappings are learned. The connection weights created by this

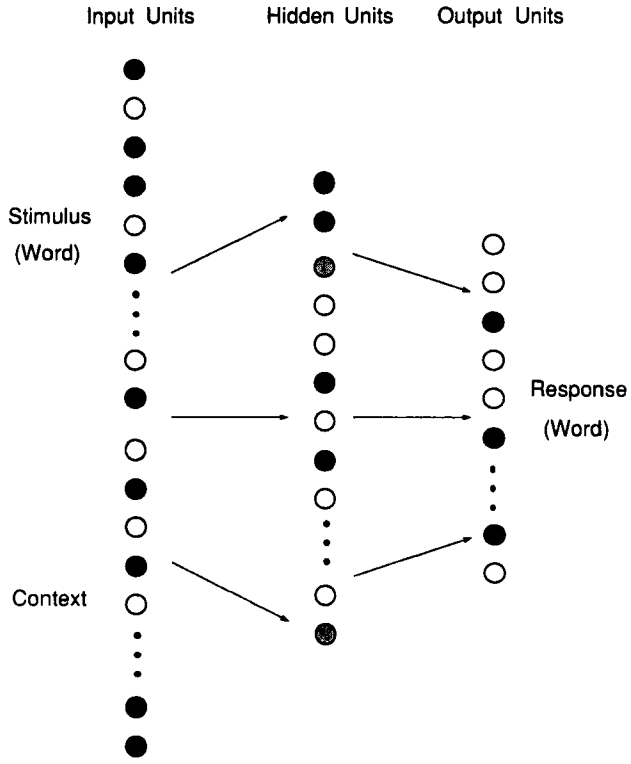


Fig. 16. A network for mapping word plus context inputs onto word outputs.

training would allow any word input to be mapped onto any word output, given appropriate signals from the context units.

Once all word–word associations have been established through preexperimental training, the learning of A–B and A–C lists in a simulated RI experiment could be accomplished solely by modifying the context-unit–hidden-unit weights according to the learning algorithm; no changes need be made for other weights (i.e., weights on connections from stimulus input units to hidden units, or the weights on connections from hidden to output units).[7] The modifications of context weights serve in essence to select from the large number of previously learned responses to a stimulus word the particular response required in the context of an experimental list. For example, in the learning of the A–B list, context weights would be modified in such a way as to select the response *banana* from among all of the responses previously associated with the stimulus *perspective,* so that the network would generate the output *banana* in response to the input *perspective in the A–B context.* Thus, whereas learning A–B and A–C lists in our RI simulation involved establishing the A–B and A–C associations in the network, learning such lists in the Rumelhart scheme involves selecting from among previously established associations.

If the A–B and A–C context patterns are nonoverlapping (i.e., if units that are on in the A–B context pattern are off in the A–C pattern, and vice versa), then none of the weights involved in representing the A–B associations will be modified during A–C learning, and no interference will occur. A simplified example may help to clarify these points. Suppose that the A–B context pattern is [1 1 0 0], and the A–C pattern is [0 0 1 1]. A–B training will modify weights on connections from the first and second context units, but not the weights on connections from the third and fourth context units. (The learning algorithm does not modify weights on connections from an input unit if that unit is off in the input pattern.) A–C learning, on the other hand, will modify the weights on connections from the third and fourth context units, but not the weights on connections from the first and second units. Thus, A–C learning will not modify weights established by A–B learning. Nor will the weightsestablished by

[7]D. E. Rumelhart (personal communication, November 1988) suggests that the weights on connections from stimulus input units to hidden units, and from hidden to output units, might be allowed to change during the learning of A–B and A–C lists, but at a much slower rate than weights on connections from context units. However, this assumption does not contribute to a reduction of interference in the Rumelhart scheme; in fact, allowing the noncontext weights to change would increase interference. In any event, the points we develop in the following discussion remain the same whether the noncontext weights remain fixed or change slowly.

A–C learning come into play when A–B associations are tested, because all of the context units that are on in the A–C context pattern are off in the A–B pattern, and therefore will not send signals to the hidden units when the A–B pattern is presented. Thus, A–C learning will leave the A–B learning intact.

Of course, if some units are on in both the A–B and A–C context patterns, then A–C learning will modify some weights involved in representing the A–B associations, and some interference will occur. However, if the overlap between context patterns is slight, then the interference would presumably not be catastrophic.

Although Rumelhart's proposal is an interesting one, it does not constitute a solution to the interference problem. In the first place, the Rumelhart scheme applies only to the learning of episodic as opposed to semantic information. In other words, the scheme applies to the establishment of context-specific associations (i.e., input A maps onto output B in context C). However, the scheme does not apply to the acquisition of general knowledge, that is, knowledge that is context independent, such as knowledge of arithmetic facts.

Even within the realm of episodic memory, the range of application of the Rumelhart scheme is limited. In particular, the scheme cannot be applied to the learning of associations between items that did not have preexisting representations in memory, because these items could not have been preexperimentally associated. Thus, for example, learning associations between novel sentences would not fall within the scope of the Rumelhart proposal. Also, unless preexperimental representations and associations are posited for nonsense syllables, it is not clear how the scheme would be applied to paired-associated learning with nonsense syllable stimuli and/or responses (as in the bulk of the published research on retroactive interference).

Even if we focus on items that clearly have preexisting representations (e.g., words), the Rumelhart scheme encounters major problems. Consider a college student serving as a subject in our hypothetical retroactive interference experiment. Assume that the student has a vocabulary of 50,000 words, any two of which might serve as a stimulus–response pair in the experiment. In order to apply Rumelhart's scheme to this situation, one must assume that prior to the experiment the student had learned 2,500,000,000 associations between words (i.e., 50,000 × 50,000), or about 350,000 per day for his or her entire life. In our view this is not a reasonable assumption.

Of course, one could assume that only some word–word associations (e.g., *dog–cat*, but not *perspective–banana*) are learned preexperimen-

tally. However, taking this tack would serve only to limit further the scheme's range of applicability, and in particular would presumably place arbitrary associations such as *perspective–banana* beyond the scope of the scheme.

A further difficulty with the Rumelhart scheme's preexperimental learning assumptions is that a network would require concurrent training to acquire the preexperimental associations. That is, the full set of associations to be learned (e.g., *perspective* in context $x \rightarrow$ *banana, perspective* in context $y \rightarrow$ *quality, perspective* in context $z \rightarrow$ *abacus*) would have to be presented repeatedly to the network, with small weight adjustments occurring on each presentation. Otherwise, the interference problem would arise in the acquisition of the preexperimental associations. If it is assumed that human learning of preexperimental associations similarly requires repeated encounters with all possible pairs of items, then the scheme's already implausible learning assumptions are further strained. If, however, the concurrent training required by the network is not taken to reflect the process whereby humans acquire preexperimental associations, then the scheme begs exactly the question it was designed to answer (i.e., how can sequential learning be modeled in connectionist networks?).

A final difficulty with the Rumelhart scheme concerns the nature of the context patterns. The scheme yields little or no interference only given the assumption that context patterns are essentially nonoverlapping. However, it is not clear that this assumption is reasonable. To the extent that context representations specify such information as the time at which learning occurred, the nature of the surroundings, and so forth, then context patterns for two learning episodes that are similar with respect to these factors, such as the patterns for A–B and A–C lists in an RI experiment, would presumably have considerable overlap. If various context patterns have substantial overlap, however, then learning in one context may alter many of the weights established during learning in earlier contexts, and so may disrupt the earlier learning.

For these reasons, we suggest that at least in its present form Rumelhart's tentative proposal does not constitute a solution to the interference problem.[8]

[8]D. E. Rumelhart (personal communication, November 1988) emphasizes that his proposal for reducing interference has not been finalized, but rather continues to evolve. Thus, the points we have made should be interpreted as applying to a particular version of the Rumelhart scheme; subsequent versions might conceivably resolve some of the problems we have discussed.

IX. Implications and Future Directions

Our analysis of the causes of interference in connectionist networks implies that in any connectionist model in which new learning may alter connection weights involved in representing previously learned information, there is a potential for interference; the new learning may disrupt the old learning. Further, the simulations we have presented suggest that in at least some circumstances the interference is catastrophic, far more severe than we would expect from human learners.

These findings suggest that maintenance of previously acquired information during new learning may pose a serious challenge for current forms of connectionist models and should be a major consideration in work aimed at further developing the connectionist framework. At this point it is an open question whether the interference problem can be resolved without substantially altering the essential characteristics of current forms of connectionist models, and without giving up attractive features such as the ability to generalize.

A more specific implication of our results is that in the development of particular models careful attention must be given to the way in which learning is modeled. Because network performance varies drastically as a function of how training is conducted, serious efforts must be made to reproduce the manner in which human learners encounter material to be learned.

The present findings also point to several issues that need to be examined further in subsequent work on interference in connectionist networks. Two of these issues are discussed briefly in the following sections.

A. Rehearsal of Previously Learned Information

First, it will be important to consider the extent to which people actually learn sequentially in various situations, and the extent to which previously learned material may be rehearsed in the course of learning new material. In this context there are a number of interesting issues to explore. For example, work by Hinton and Sejnowski (1986) and Hinton and Plaut (1987) suggests that rehearsing some previously learned information during new learning may to some extent protect from disruption not only the rehearsed items, but other previously learned items as well. We have found with our arithmetic model, however, that the problems 2 + 1 and 1 + 2, which were learned during training on the ones problems, showed temporary but drastic disruption when training on twos problems began, even though training on 2 + 1 and 1 + 2 continued uninterrupted when the twos problems were introduced. Additional results from both

the arithmetic and RI models indicate that even if training on all of the previously learned information continues during acquisition of new information, the old information nevertheless shows this temporary disruption.

The temporary disruption phenomenon may be seen clearly in our PQR network. Figure 17 shows what happens when the network is first trained on pattern 1 alone, and then on patterns 1 and 2 concurrently. When pattern 2 is introduced, the weight configuration is quickly pulled out of the pattern 1 solution space, and then moves toward the overall solution space. Once pattern 1 is well learned, it no longer pulls the weight configuration very strongly. When pattern 2 is introduced it pulls strongly, and as a result the configuration is pulled out of the pattern 1 solution space.

B. Interference under Concurrent Training Conditions

We have discussed the interference issue in the context of situations involving sequential learning, because these situations present the issue in its clearest form. However, the points we have developed, and the conclusions we have drawn, may also apply to many situations in which learning is more concurrent.

Consider as an example Seidenberg and McClelland's (in press) model of single-word reading, in which a connectionist network maps orthographic representations of words onto phonological representations. Seidenberg and McClelland present results from simulations in which a

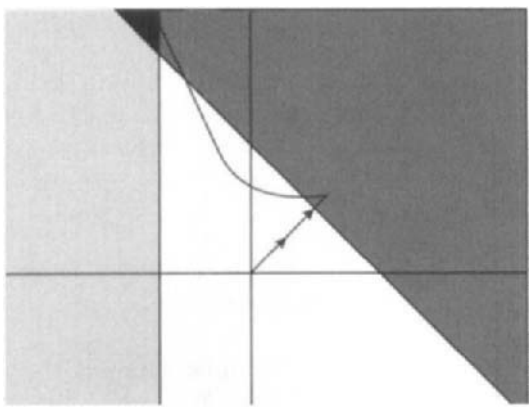


Fig. 17. Movement of the weight configuration over learning trials with training on pattern 1, followed by concurrent training on patterns 1 and 2.

network was trained concurrently on a corpus of 2,897 words, using the back-propagation learning algorithm. Words were presented to the network according to their frequency in the language, such that high-frequency words were presented more often than low-frequency words. However, as is inevitable with a limited corpus of words, the training procedure did not fully reflect the range of frequencies with which words occur in the experience of a human reader. For example, words with a frequency of 1 per million were presented to the network far more often than once every million words. On the basis of the simulation results, Seidenberg and McClelland (in press) argue that their model captures a variety of phenomena.

The interference issue arises with respect to the Seidenberg and McClelland model in the following way: After a human reader encounters a word, many other words may be encountered before the word is seen again. This is especially true of low-frequency words like *yacht*, which human readers encounter on average less than once in every hundred thousand words.

In the Seidenberg and McClelland model the weight adjustments that occur whenever a word is presented to the network have the potential to disrupt the representations of other words. Thus, one may ask whether in a network of this sort a low-frequency irregular word such as *yacht* could ever be learned and retained when so many weight adjustments occur between successive presentations of the word. Seidenberg and McClelland's simulation with a 2,897-word corpus cannot answer this question, because the number of different words the network encountered and the number of words intervening between two successive presentations of a low-frequency word were vastly smaller than for human learners.

Thus, the interference phenomena we have discussed in this chapter raise concerns about the conclusions that can be drawn from simulations involving toy versions of the tasks humans perform. Minsky and Papert (1988) raise similar concerns on different grounds in the new edition of their classic book *Perceptrons*.

X. Concluding Remarks

In concluding, we must acknowledge a limitation of the work we have presented. In order to determine the extent to which interference represents a problem for connectionist models, we need to be able to specify the conditions under which interference will be more severe than expected from human learners, and the conditions in which interference

would take on more reasonable proportions. However, at present we are not in a position to do this. Our analysis of the causes of interference implies only that *at least some* interference will occur whenever new learning may alter weights involved in representing old learning, and our simulation results demonstrate only that interference was catastrophic in some specific networks.

We could offer some rough generalizations concerning factors affecting the severity of interference: For example, other things being equal, the greater the amount of new learning, the greater the disruption of old learning. However, we are far from being able to delineate systematically the factors that determine the severity of interference, or the ways in which these factors interact to produce a particular level of interference in a particular network. Perhaps this somewhat unsatisfying state of affairs reflects our own limitations. Perhaps however, it has something to say about the current level of development of the connectionist framework.

In its present form connectionist modeling attempts to explain human cognitive functions in terms of networks that are themselves poorly understood. Thus, when a network behaves in a particular way the reasons may not be entirely clear. One often cannot be sure of the extent to which the network's performance crucially depends upon particular parameter settings, or on some more or less arbitrary choices that had to be made in developing a simulation. Similarly, it may be impossible to predict what effects particular modifications of a network would have. Thus, in the extreme one may be limited to drawing conclusions that apply only to particular networks with particular parameter settings, particular numbers of hidden units, and so forth.

Progress has also been impeded by a relative lack of attention within the connectionist framework to questions concerning the nature of the overall cognitive architectures within which particular networks fit. In the absence of clearly articulated claims about a network's place in a larger system one cannot readily answer such questions as, What computations must the network be able to carry out?; How accurate must the network's output be to allow a correct overt response to be generated?; and, What constraints can be placed on the form that inputs to the network may take? Even in the Seidenberg and McClelland (in press) reading model, which is perhaps the most explicit connectionist model with respect to specifying the overall architecture within which the implemented network fits, some significant questions remain unanswered. For example, the mechanisms that transform the network's outputs into overt responses are not merely unimplemented, but in fact entirely unspecified. As a result, it is unclear just how closely and in what ways the network's outputs must correspond to the correct outputs to allow a correct overt

response to be generated. Thus, for example, if the network generates a not entirely correct output for a low-frequency word, there may be no basis for deciding whether the output is sufficiently accurate to allow generation of the correct overt response. Similarly, in our own simulations the absence of a framework within which we could motivate specific claims about the generation of overt responses led to our use of four different, and essentially arbitrary, performance measures.

In our view, prospects for progress in connectionist modeling hinge critically upon elaboration of the connectionist framework along at least two fronts. First, as Minsky and Papert (1988) have argued, empirical exploration must be accompanied by formal analysis of the behavior of connectionist networks. Second, attention must be directed toward articulating overall cognitive architectures within which particular networks fit. Some encouraging preliminary steps have been taken, but it remains to be seen whether connectionism will emerge as a productive framework for modeling human cognition.

ACKNOWLEDGMENTS

The research reported in this chapter was supported by NIH grant NS21047 to Michael McCloskey, and by a grant from the Sloan Foundation to Neal Cohen. We thank Sean Purcell and Andrew Olson for assistance in generating the figures, and Alfonso Caramazza, Walter Harley, Paul Macaruso, Jay McClelland, Andrew Olson, Brenda Rapp, Roger Ratcliff, David Rumelhart, and Terry Sejnowski for helpful discussions.

REFERENCES

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

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

Anderson, J. R., & Bower, G. H. (1973). *Human associative memory*. Washington, DC: Winston.

Barnes, J. M., & Underwood, B. J. (1959). "Fate" of first-list associations in transfer theory. *Journal of Experimental Psychology*, **58**, 97–105.

Carpenter, G. A., & Grossberg, S. (1986). Adaptive resonance theory: Stable self-organization of neural recognition codes in response to arbitrary lists of input patterns. In *Program of the Eighth Annual Conference of the Cognitive Science Society* (pp. 45-62). Hillsdale, NJ: Erlbaum.

Fodor, J. A., & Pylyshyn, Z. W. (1988). Connectionism and cognitive architecture: A critical analysis. *Cognition*, **28**, 3–72.

Hinton, G. E., McClelland, J. L., & Rumelhart, D. E. (1986). Distributed representations. In D. E. Rumelhart & J. L. McClelland (Eds.), *Parallel distributed processing: Explora-*

tions in the microstructure of cognition: Vol. 1. Foundations (pp. 77–109). Cambridge, MA: MIT Press.

Hinton, G. E., & Plaut, D. C. (1987). Using fast weights to deblur old memories. In *Program of the Ninth Annual Conference of the Cognitive Science Society* (pp. 177–186). Hillsdale, NJ: Erlbaum.

Hinton, G. E., & Sejnowski, T. J. (1986). Learning and relearning in Boltzmann machines. In D. E. Rumelhart & J. L. McClelland (Eds.), *Parallel distributed processing: Explorations in the microstructure of cognition: Vol. 1. Foundations* (pp. 282–317). Cambridge, MA: MIT Press.

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

McClelland, J. L., Rumelhart, D. E., & Hinton, G. E. (1986). The appeal of parallel distributed processing. In D. E. Rumelhart & J. L. McClelland (Eds.), *Parallel distributed processing: Explorations in the microstructure of cognition: Vol. 1. Foundations* (pp. 3-44). Cambridge, MA: MIT Press.

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

Minsky, M. L., & Papert, S. A. (1988). *Perceptrons: An introduction to computational geometry (Expanded edition)*. Cambridge, MA: MIT Press.

Postman, L. (1962). Transfer of training as a function of experimental paradigm and degree of first-list learning. *Journal of Verbal Learning and Verbal Behavior*, **1**, 109–118.

Postman, L., & Underwood, B. J. (1973). Critical issues in interference theory. *Memory & Cognition*, **1**, 19–40.

Prince, A., & Pinker, S. (1988). On language and connectionism: Analysis of a parallel distributed processing model of language acquisition. *Cognition*, **28**, 73–194.

Ratcliff, R. (in press). *Connectionist models of memory: Constraints imposed by learning and forgetting functions. Psychological Review*.

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

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

Seidenberg, M. S., & McClelland, J. L. (in press). A distributed, developmental model of word recognition and naming. *Psychological Review*.

Sejnowski, T. J., & Rosenberg, C. R. (1987). Learning and representation in connectionist models. In M. S. Gazzaniga (Ed.), *Perspectives in memory research and training*. Cambridge, MA: MIT Press.

Sutton, R. S. (1986). Two problems with backpropagation and other steepest-descent learning procedures for networks. In *Program of the Eighth Annual Conference of the Cognitive Science Society* (pp. 823–831). Hillsdale, NJ: Erlbaum.