Is There "Catastrophic Interference" in Connectionist Networks?

Phil A. Hetherington & Mark S. Seidenberg

Department of Psychology, McGill University

Concern has recently developed regarding the possibility that parallel distributed processing models will exhibit massive amounts of retroactive interference. McCloskey & Cohen (in press) have suggested that such models exhibit "catastrophic interference" under realistic training conditions; they conclude that PDP models may not be able to simulate basic aspects of human performance. In this paper we report replications and extensions of simulations on which these claims were based. The new simulations suggest that "catastrophic interference" may be less of a problem than McCloskey & Cohen suggest; specifically, it is related to the use of a rigid training scheme that bears little resemblance to how children actually learn.

Learning in parallel distributed processing models involves changes to the weights on connections between units as a consequence of feedback or "experience." One of the main properties of these models is that the effects of learning are superimposed on one another; a model's performance is determined by the aggregate effects of the ensemble of training experiences. This property of PDP models is thought to be theoretically important; for example, it enables Seidenberg and McClelland's (in press) model of word recognition to simulate the effects of inconsistent spelling-sound correspondences on tasks such as naming and lexical decision, and Rumelhart and McClelland (1986) have argued that it is critical to an account of facts about the child's acquisition of past tense morphology. Recently, however, it has been noted that this property of PDP models may have some negative side effects. Two issues have arisen. First there is the problem of retroactive interference: events later in the training regime may result in poorer performance on previously-learned items. For example, a word pronunciation model (e.g., Sejnowski & Rosenberg, 1986; Seidenberg & McClelland, in press) might be trained to generate the correct pronunciation of a word such as GAVE; subsequent training on a word such as HAVE might result in changes to the weights that have a negative impact on performance on GAVE, yielding incorrect output or "unlearning".

A second, related issue concerns the regimes used in training PDP models. Most training schemes to date have involved what McCloskey and Cohen (in press; hereafter MC) have termed "concurrent" schedules: there is a set of target patterns to be learned, and training proceeds by sampling from the entire set. This contrasts with what MC term "sequential" regimes, in which target patterns are introduced at different times. Sequential regimes are thought to be more realistic in terms of peoples' actual experience. In learning to read, for example, children are exposed to different words at different times, whereas in the Sejnowski and Rosenberg (1986) and Seidenberg and McClelland (in press) models, all words are available for training at all times.

Given the fact that learning can produce retroactive interference, it is clear that the performance of PDP models will be highly dependent on the type of training scheme that is used. MC have conjectured that PDP models may be incapable of simulating human learning under realistic training conditions. Specifically, their claim is that there is a retroactive interference problem that can only be overcome by using concurrent training schemes. When the more realistic sequential schemes are utilized, such models exhibit "catastrophic" interference: learning on later trials results in grossly impaired performance on previously-learned items. MC illustrated this problem by analyzing some simulations of the task of learning simple arithmetic.

The MC paper raises important issues concerning learning in PDP models; Pinker and Prince (1988) and Lachter and Bever (1988) present similar concerns. MC's simulations demonstrate that there are conditions under which PDP models exhibit behavior that does not relate well to human performance. However, the scope of the "catastrophic" interference problem is unclear; in this paper we report simulations that examine the issue further. Our main conclusion is that catastrophic interference is not as general a problem for PDP models as MC suggest; in fact, replications of their simulations with slight changes in the training procedure yield very different results than they reported. Our simulations provide a

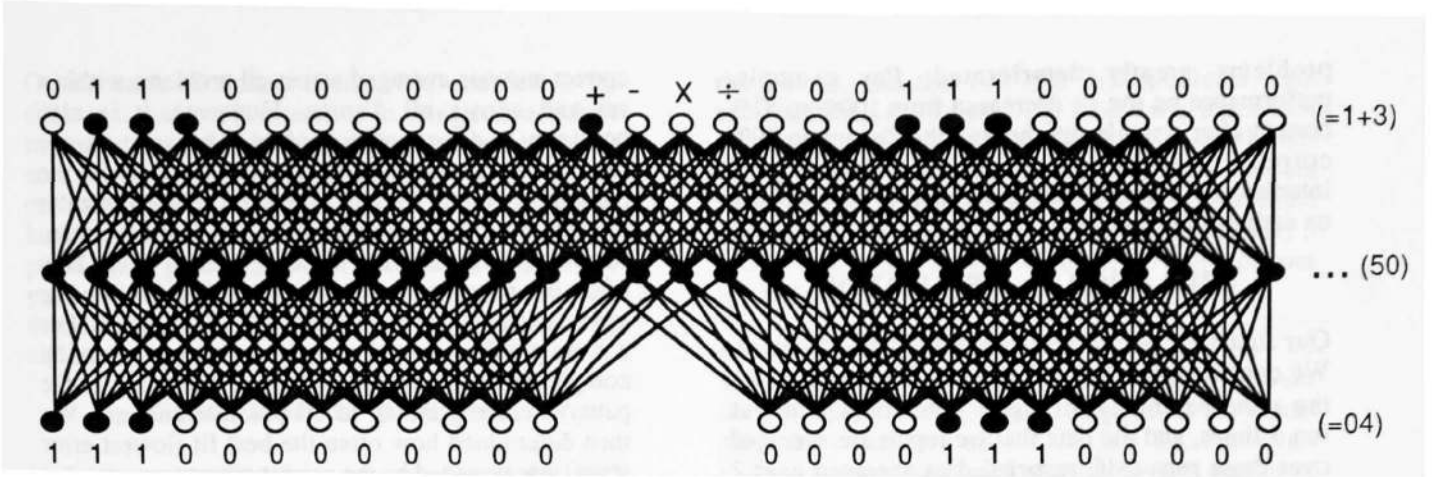

Figure 1: The Model (Input on Top; Output on Bottom)

broader perspective on the conditions that do and do not yield excessive retroactive interference, and why.

BACKGROUND: MODEL AND TASK

MC report several simulations using a two-layer model (i.e., a model with two layers of connections), trained using the backpropagation algorithm (Rumelhart, Hinton, & Williams, 1986). The model consisted of 28 input units, 50 hidden or association (Rosenblatt, 1962) units, and 24 output units (Figure 1), with full connectivity between adjacent layers. The model was trained to perform simple addition and multiplication problems; for example, given the input [3+2], the model was to produce the output [5]. Each of the two digits in an equation was encoded by 12 input units. The remaining four input units encoded which operation was to be performed (+, -, x, +). The first 12 output units represented the tens column of the answer; the second 12 coded the digits column. Figure 1 illustrates the problem [1 + 3 = 04]. The input and output representations were distributed; each of the numbers from 0 to 9 was encoded by three units. This method is similar to the thermometer coding scheme used by Anderson (1983) and by Viscuso, Anderson, and Spoehr (in press) to code continuous values in qualitative physics and mathematics. This distributed representation can represent any pair of operands and their sums or products, yet the individual units and connections do not represent the numbers themselves.[1]

MC's simulations were concerned with the task of learning simple arithmetic problems. Consider, for example, the set of simple addition problems involving the digits 1-9. MC show that the model is able to learn the target set of patterns when it is trained using a concurrent method. During the training phase, the model was presented with problems from the target set. Problems were randomly sampled from the set; all problems were available to be sampled at all times. Thus, the model might be trained on [1+3], then [2+9], then [8+7], etc. Under these conditions, the network learned to successfully map all pairs of operands to their respective sums; it also learned the mapping from the operands to their products.

Very different results were obtained using a sequential training method, however. The model was initially trained on addition problems involving 1's (see "Simulation 1: Replication" below); training continued until the model performed without error on these items. The model was then trained on problems involving 2's. The primary motivation for this training scheme was the intuition that it more closely resembles the experience of children learning arithmetic. Children are not exposed to all problems in a random order; they learn the simpler problems and then move to more complex ones. With the sequential method, the model learned to compute the 2's problems; however, performance on the 1's

1. The scheme used to encode digits was not entirely arbitrary. Each digit was encoded by 3 consecutive input units. The first 3 units were used to encode 0, the second 3 units encoded 1, the third 3 units, 2. Thus, digits that differ by one shared 2 encoding units; digits that differ by 2 shared 1 encoding unit, and digits that differ by more than 2 shared no units in common.

problems greatly deteriorated. For example, performance on the 1's decreased from 100% to 57% correct after a single run through the 2's, and to 30% correct after two such runs. "Catastrophic interference" refers to this decrement in performance on earlier-trained items.

SIMULATION 1: REPLICATION

Our first step was to replicate MC's basic findings. We constructed a network exactly like theirs, using the same parameter settings.[2] The simulation was run 5 times, and the data that we report are averaged over these runs (MC reported data averaged over 2 runs). The model was trained on the 1's and 2's problems in sequence. The 1's set included 16 problems: 1+1, 1+2, 1+3, ... 1+9; 3+1, 4+1, ... 9+1. There were no problems containing 0's (as in the MC simulations), and the 2+1 problem was excluded because it occurred in the 2's set. Similarly, the 2's set included 16 problems; 1+2 was excluded because it occurred in the 1's set. Hence the two problem sets were mutually exclusive.

Training involved a series of epochs, where each epoch refers to the presentation of all problems within a set in random order. For example, 40 epochs of training on the 1's set involved presenting 40 sets of the 16 1's problems, each in a different random order. Performance was evaluated in two ways. First, for each problem we calculated an error sum of squares (**E**); this was the sum of the squared differences between computed and target values over all output units:

$$\mathbf{E} = \sum_i (o_i - e_i)^2$$

This score provides a general quantitative measure of performance. Below we report the error scores for the correct answers averaged across all problems within a set and across all 5 runs. However, it is also necessary to determine how often the correct answer to a given problem provided the best fit to the computed output. That is, the error score indicates how closely the computed output matched the pattern for the *correct* answer; we also need to know how often the correct answer produced the lowest error score (what MC term the "best match" criterion). For a given problem, we calculated a set of error scores by comparing the computed pattern of activation to the patterns corresponding to all possible answers. We then determined how often the best fit (lowest error score) was provided by the correct answer.

The training procedure followed MC's sequential method. The model was trained on the 1's problems for 40 epochs. (MC trained their network until all of the output units had activations within .1 of the target activation levels, which took approximately 35 epochs.) The model was then trained on the 2's problems for 40 epochs. We tested the model's performance on the 1's during the training on the 2's. These test trials did not involve additional learning on the 1's; thus, we could examine how training on the 2's affected performance on the 1's. The 1's were tested after each of the first 5 epochs of training on the 2's; thereafter they were tested after every 5 epochs of training until the 40th epoch. Starting at epoch 40, the 1's were tested after each of 5 additional epochs, and then at 5 epoch intervals until the 80th epoch. All that changed across simulations was the order in which the problems were presented within an epoch and the initial random values assigned to the weights.

Results and Discussion

The model learned the 1's set very quickly. After 15 epochs of training, the average error score was .144, and no errors were made (i.e., for all problems, the correct answer provided the best fit to the computed output). Error scores continued to decrease with additional training. However, performance on the 1's decreased drastically once training on the 2's began. In only one epoch, the mean error for the 1's increased from .038 to .734, more than an order of magnitude. After five epochs, the mean error reached 1.41. The best match criterion yielded similar results: the mean number of correct responses fell from 16 to 8.6 in one epoch. By five epochs, the mean number of correct responses was a catastrophic 2.8. Thus, learning the 2's problems interfered with performance on the 1's.

2. The simulations were implemented using the McClelland and Rumelhart (1988) software running on an IBM PS/2 Model 80 computer. Except where noted in the text, the simulations followed MC's procedure exactly. The learning rate was set to .25. MC consider this to be a conservative rate although McClelland & Rumelhart's (1988, p. 107) recommendation to use a rate equal to the inverse of the number of input units would result in a much smaller value (.036). Weights on connections between units were assigned initial random values between +/- .3. Target activation values were set to .9 for units that should be on, and .1 for units that should be off. Finally, all hidden and output units were given a random bias.

On the basis of similar results, MC concluded, "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" (p. 14). The question to be addressed is this: how serious is the "catastrophic interference" problem? In particular, how closely is it related to the particular conditions studied by MC, and how do these conditions relate to the ones experienced by children in learning arithmetic and other skills?

SIMULATION 2: SAVINGS

Under the sequential training procedure studied by MC and in Simulation 1, performance on the 1's deteriorates drastically during the learning of the 2's. The decrement in performance is seen in the increasing error scores for the 1's, and the decrease in the proportion of correct answers. Hence it appears that the solutions to the 1's problems were unlearned. If this is correct, the model's performance differs greatly from that of humans; as MC note, unlearning (e.g., in verbal learning experiments) is virtually never complete (see, e.g., Postman & Underwood, 1973). It is possible, however, that the solutions to the 1's problems were not completely unlearned; the weights on connections between units could still have encoded information relevant to these problems despite the seemingly poor level of performance. This issue can be examined by determining whether there is any *savings* (Ebbinghaus, 1885) when the 1's problems are relearned. Consider the following procedure: we train the model as in Simulation 1, producing poor performance on the 1's once the 2's are introduced. We then retrain the model on the 1's, and introduce a new set of problems, the 3's. If the 1's have been completely unlearned due to "catastrophic interference," they should be relearned at the same rate as the entirely new problem set. Faster relearning on the 1's would indicate memory savings, because the 1's problems had not been completely unlearned. Savings would indicate that the network had retained information relevant to computing the correct answers, facilitating relearning.

In the second simulation, we examined whether savings would occur. We first replicated Simulation 1. The 1's were trained for 40 epochs, followed by the 2's for 40 epochs. This procedure results in poor performance on the 1's. We then trained the model on a set of problems involving 1's and 3's. This set of 30 problems contained all of the unique 1's and 3's problems (i.e., the 3+1 problem was excluded from the 1's set, and the 1+3 and 2+3 problems were excluded from the 3's set). The model was trained for 40 epochs on this larger set. This period, epochs 80-120, will be termed the retraining phase. The model's performance on the 1's and 3's problems was tested after each of the first 5 retraining epochs, and every 5 epochs thereafter. The data we report are averaged over the 10 independent simulation runs.

Results and Discussion

The primary results are presented in Figure 2. Over the first 5 epochs of the retraining phase, the model performed similarly on the 1's and 3's. However, looking at the longer trend over the first 25 epochs of retraining, learning of the 3's was slower than relearning of the 1's. Using the mean error scores as the dependent measure in an analysis of variance, there was a main effect of problem set, $\mathbf{F}(1,18) = 43.42$, $\mathbf{p}<.001$. The same effect was found using the best match criterion: the 1's produced significantly fewer errors over the first 25 epochs of training, $\mathbf{F}(1,18) = 46.77$, $\mathbf{p}<.001$. Hence there was savings in the relearning of the 1's, indicating that they had not been completely unlearned.

Since the 1's and 3's problems were similar in terms of complexity, they should have been equally easy to learn. Hence, the improved performance on the 1's appears to have been due to savings—prior experience with the 1's that was not completely erased by exposure to the 2's. To be certain that both problem sets were equally easy to learn, we ran a control

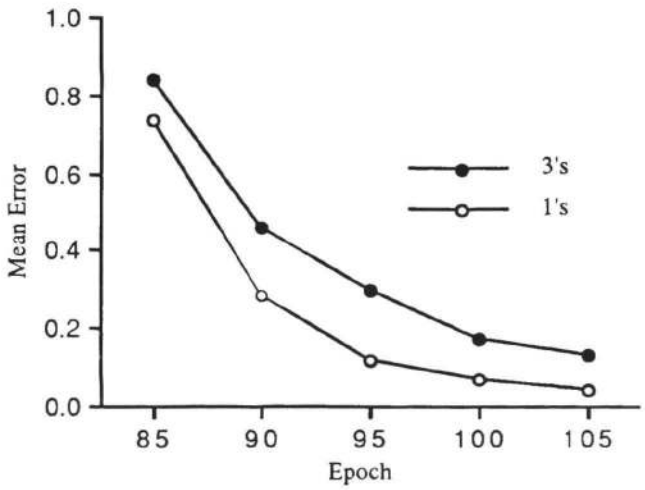

Figure 2: Learning 3's vs Relearning 1's

simulation in which the model was trained from the first epoch on the 1's-and-3's set. The model was then tested on both the 1's and 3's problems at 5 epoch intervals; there were no statistically significant differences between the two problem sets, in terms of either error scores or percentage of correct responses.

This simulation demonstrates that under the "catastrophic" interference conditions studied by MC, the 1's are not entirely unlearned. Because the network retains information relevant to these problems, they are relearned more quickly than a novel set of problems. The simulation illustrates that global measures such as mean squared error or number of correct answers may not fully capture all that a model has learned. The existence of savings is especially important because it bears on the scope of the "catastrophic interference" problem. If there is significant savings, then the "catastrophic" performance of the 1's might be dramatically improved by a small number of relearning trials. That is, catastrophic interference may critically depend on the *blocking* of training trials. When the model is trained on a block of 1's problems, and then on a block of 2's problems, performance on the 1's declines. If, instead of following this strict blocking scheme, there is some minimal retraining on the 1's, performance will rapidly improve due to savings. In the present case, we retrained the model on the 1's after exposure to the 2's (starting at epoch 80). After only 3 epochs of retraining, performance improved from a mean error of 1.73 and 12.5% correct to a mean error of 0.23 and 86.3% correct. After 5 epochs, the error was .11 and 97.5% were correct.

Rapid relearning can also be illustrated in the context of Seidenberg and McClelland's (in press) model of word naming. The model was trained on a set of 2897 monosyllabic words. The model takes a spelling pattern as input and produces a phonological code as output. After 250 epochs of training, the model performs this task with a high degree of accuracy. For a word such as TINT, for example, the best fit to the computed output is provided by the correct phonological code /tint/. Consider now what would happen if we trained the model on a block of trials involving the word PINT, which is spelled like TINT but pronounced differently. Training on PINT will affect the weights in a way that has a negative impact on TINT, producing retroactive interference. Figure 3 illustrates this effect. After 250 epochs of training, TINT produced an error score of 8.92. The model was then trained on 20 PINT trials, with TINT retested after each trial. As the figure illustrates, training on PINT increases the error score for TINT, indicating poorer performance or "unlearning." However, the figure also shows the effects of additional learning trials on TINT. With only 2 additional trials, the error score falls below the level that had been achieved prior to training on PINT. In sum, a small number of retraining or "reminding" trials is sufficient to overcome the interfering effects of prior learning.

It is clear, then, that retroactive interference in simple PDP nets depends on the properties of the training regime. MC's main point is that the concurrent regime used in most simulations is unrealistic. However, the scheme they introduced is equally unrealistic. Their scheme is not merely sequential; it involves strictly blocking trials by type. Consider how this blocking scheme relates to the child's experience in learning arithmetic. It can be seen from any arithmetic primer that children are not taught 1's problems, then 2's, then others in strict blocks. In fact, children's problems are typically ordered in terms of the magnitudes of sums, not operands, with considerable overlap across problem sets. In learning multiplication tables, new problems are typically embedded in written practice sheets along with problems introduced earlier (e.g., Campbell & Graham, 1985). The problem sets are indeed ordered—small-number problems are usually taught earlier—but these problems are also drilled and practiced when new ones are introduced.

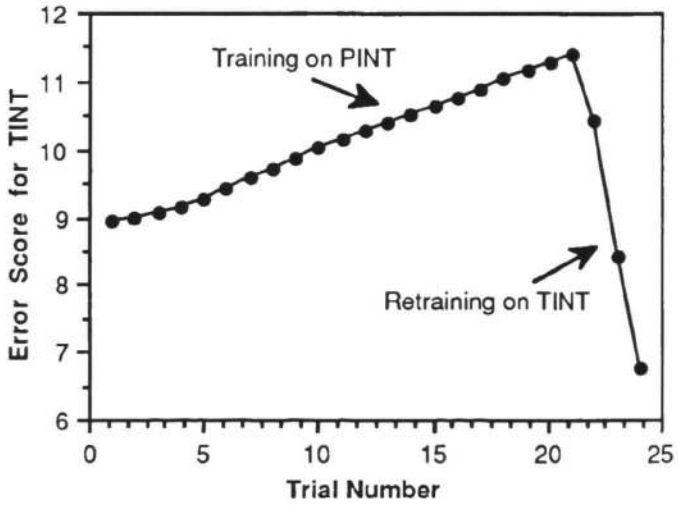

Figure 3: Retraining on TINT after Training on PINT.

Our main point is that a more realistic training regime—one that does not involve strict blocking by type—would take advantage of the savings illustrated in Simulation 2. As long as the child (or model) experiences a small number of relearning trials, the learning of new problems should not result in massive interference. The next simulation examined this issue empirically.

SIMULATION 3: A MORE REALISTIC TRAINING REGIME

As we have noted, addition problems are not taught using mutually exclusive sets of problems. If the 1's are taught first, followed by the 2's, the set of 2's usually contains some of the 1's as reminder or refresher trials. From the teacher's intuitive perspective, the purpose of these trials is to consolidate or reinforce prior learning. The simulation models provide a computational way to construe this "consolidation" process: the reminding trials are necessary in order to reduce the interfering effects of new learning. We examined this process in a new simulation involving 5 stages. The main idea was to use a sequential training regime in which we used overlapping problem sets. The effect of this regime was to slowly introduce new problems while slowly phasing out old ones. Each of the five stages was 10 epochs long. The first stage involved training the model on two sets of 1's problems. During each epoch in the second stage, the model was trained on two sets of 1's and one set of 2's. In the third stage, the model was trained on one set of 1's, two sets of 2's, and one set of 3's. In the fourth stage, the model was trained on one set of 1's, two sets of 2's, two sets of 3's, and one set of 4's. Finally, the fifth stage included one set of 2's, two sets of 3's, two sets of 4's, and one set of 5's. No 1's were presented in the final stage.

As can be seen from this description, the training procedure involved fading in new problems while fading out old ones. Thus, the training regime was not strictly concurrent (all problems were not available for training simultaneously) but it was not as rigidly sequential as the MC procedure. All sets of problems were defined as before; they were constructed so as to contain 13 problems that did not occur in any other set (e.g., 1+3 occurred in the 1's set, not the 3's set). The simulation was replicated 5 times; the data are averaged across all 5 runs.

Results and Discussion

The primary data concern performance on the 1's as a function of exposure to other problems (Figure 4). The data in the figure were averaged over two consecutive epochs. During the first stage, the model learned the 1's problems. Introduction of the 2's during stage two initially caused a small decrement in performance (epochs 12-14), but there was rapid recovery (epochs 16-18). Similar effects were obtained at stages three and four, with a notable decrease in magnitude in stage four. Thus, training on other problems produced diminishing amounts of interference on the 1's. Data concerning the average number of correct responses showed a similar pattern. During stage 5, when there was no additional training on the 1's, the error scores for these problems began to increase again. Note, however, that the increase was still relatively small, and the model still averaged less than one error per problem set. After 75 epochs—35 epochs after the network was last trained on the 1's set and following 2730 trials on other problems—the mean error score for the 1's was .32 and the mean number of correct responses was 11.8/13 (91%). In sum, the model did not exhibit catastrophic interference.

GENERAL DISCUSSION

Our findings can be summarized as follows. MC are correct in observing that there is massive retroactive interference in a simple PDP model of arithmetic learning when the problem sets are strictly blocked (Simulation 1). Earlier problems are not completely unlearned, however, as evidenced by the savings

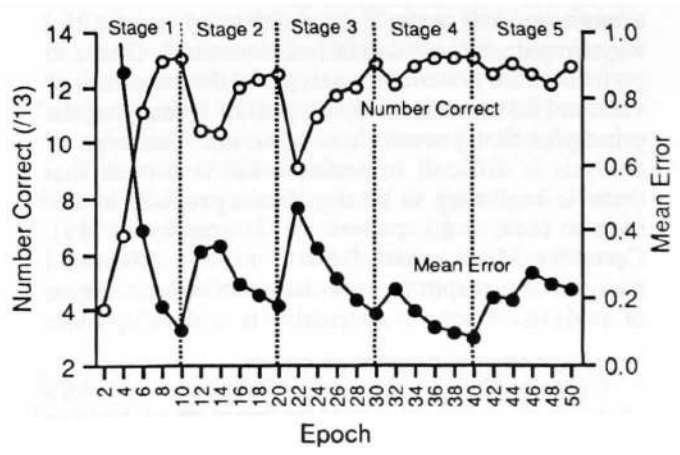

Figure 4: Performance on the 1's Problems During Five Stages of Training

observed in Simulation 2. Taking advantage of this savings merely requires relaxing the strict blocking of training trials by type (Simulation 3). This does not involve the "concurrent" procedure that MC consider unrealistic; rather, it involves a training sequence more like the ones used in the actual teaching of arithmetic. Thus, in Simulation 3, the model was able to learn the 1's problems and this knowledge was not eliminated by a large amount of training on subsequent problems.

The main point of our simulations has been to suggest that it would be a mistake to overinterpret MC's results, since very small changes to their procedures yield very different results. We should stress, however, that our simulations by no means resolve any of the important questions concerning retroactive interference in PDP models. Our simulations—as well as MC's—provide empirical data concerning a relatively small subset of cases. These simulations represent individual points in a very large multidimensional space of possible models. This space of possibilities is defined by the range of possible architectures (e.g., number of units, patterns of connectivity, encoding schemes), learning procedures, and training regimes. Empirical demonstrations such as ours and MC's can be useful in identifying potential problems and solutions. However, they do not provide a definitive basis for identifying principled limitations of the PDP approach.[3]

It will be important to understand the scope of retroactive interference problems in PDP networks in a more rigorous way. There seem to be two fruitful ways to pursue this issue in future research. One is to perform more systematic analyses of the properties of various PDP models, with the goal of identifying the principles that govern their behavior. This type of analysis is difficult to perform, but it is clear that there is beginning to be significant progress in this regard (see, e.g., papers in Touretzky, 1989). Certainly Minsky and Papert's (1969) celebrated analysis of perceptrons provides a model for this type of analysis. A second alternative is to develop more realistic models that provide a systematic account of a broad range of behavioral data. The problem with demonstrations such as MC's (and our own) is that they do not attempt to simulate a realistic learning task or account for detailed aspects of human performance. In the area of arithmetic learning, for example, there is a large amount of behavioral data, several accounts of which have already been proposed (e.g., Groen & Parkman, 1972; Siegler & Shrager, 1984). A reasonable goal would be to attempt to develop simulation models that address such nontrivial phenomena in detail. Again, examples of PDP models with broad scope and coverage of the data are beginning to appear (e.g., Seidenberg & McClelland, in press; Dell, 1986).

Ratcliff (1989) presents an impressive example of the second approach. He explored whether a connectionist model could simulate an extensive set of findings concerning recognition memory performance, and systematically explored several modelling variables (MC also report simulations of some of these phenomena). Interestingly, all of Ratcliff's models produced behaviors unlike humans'. Analyses such as Ratcliff's contribute to understanding where there is and is not a good match between the properties of connectionist models and those of human behavior. The failure of Ratcliff's simulation models suggests that this type of recognition memory performance cannot be construed in terms of learning in multilayer nets via backpropagation. Several characteristics of these recognition memory phenomena appear to be critical to understanding why the simulations failed. Unlike most learning, the typical recognition memory experiment does involve strict sequencing of trials, as well as rapid stimulus presentation that limits the use of rehearsal or other learning strategies, and very simple, unrelated stimuli, such as lists of letters or words. The question then is whether other types of learning exhibit the characteristics that apparently make the connectionist approach so inapplicable in this case.

Consider in this light the question of retroactive interference in learning simple arithmetic. Our simulations suggest that the seriousness of this problem depends in part on questions concerning the learning regime: is it strictly concurrent, is it strictly blocked, or is it neither of these extremes? We suggest that it is more concurrent than MC recognize, and less sequential than in the case of recognition memory experiments. This is simply an empirical

3. We did explore one other factor, the number of hidden units, which we thought would be important on the basis of previous research (e.g., Seidenberg & McClelland, in press) and a reviewer's comments. However, essentially similar results were obtained using 13, 25, and 50 hidden units.

question, however. With a more realistic characterization of the task and the learning environment, it should then be possible to determine whether, in fact, there is a serious retroactive interference problem or not. It is doubtful, however, whether this substantive issue can be decided on the basis of demonstration programs like MC's. One of the main lessons of research in traditional, symbol-processing artificial intelligence was that general principles cannot be uncovered by studying toy problems. There is no reason to think that anything different should obtain in the case of PDP.

REFERENCES

Anderson, J.A. (1983). Cognitive and psychological computations with neural networks. *IEEE Transactions on Systems, Man, and Cybernetics, SMC-13*, 799-815.

Campbell, J.I.D., & Graham, D.J. (1985). Mental multiplication skill: Structure, process, and acquisition. *Canadian Journal of Psychology, 39*(2), 338-366.

Dell, G., (1986). A spreading activation theory of retrieval in sentence production. *Psychological Review, 93*, 283-321.

Ebbinghaus, H. (1885). *Ueber das gedächtnis: Untersuchen zur experimentellen psychologie* ("On memory") (H.A. Ruger & C.E. Bussenius, trans.). New York: Dover, 1964.

Groen, G.J., & Parkman, J.M. (1972). A chronometric analysis of simple addition. *Psychological Review, 79*, 329-343.

Lachter, J., & Bever, T.G. (1988). The relationship 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. (1988). *Parallel distributed processing: A handbook of models, programs, and exercises. Volume 3.* Cambridge MA.: MIT Press.

McCloskey, M., & Cohen, N.J. (in press). Catastrophic interference in connectionist networks: The sequential learning problem. Paper to appear in G.H. Bower (Ed.), *The psychology of learning and motivation: Volume 23*. New York: Academic Press.

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

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

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

Ratcliff, R. (1989). *Connectionist models of recognition memory: Constraints imposed by learning and forgetting functions.* Unpublished manuscript.

Rosenblatt, F. (1962). *Principles of neurodynamics.* Washington: Spartan Books.

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. Volume 1* (pp. 318-362). Cambridge MA.: MIT Press.

Rumelhart, D.E., & McClelland, J.L. (1986). On learning the past tenses of English verbs. In J.L. McClelland & D.E. Rumelhart (Eds.), *Parallel distributed processing: Explorations in the microstructure of cognition. Volume 2* (pp. 216-271). 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., & Rosenberg, C. (1986). *NETtalk: A parallel network that learns to read aloud.* Baltimore, MD: Johns Hopkins University EE and CS Technical Report JHU/EECS-86/01.

Siegler, R.S., & Shrager, J. (1984). Strategy choices in addition and subtraction: How do children know what to do? In C. Sophian (Ed.), *Origins of Cognitive skills.* Hillsdale, NJ. Lawrence Erlbaum Associates.

Touretzky, D.S., Ed. (1989). *Advances in neural information processing systems 1.* San Mateo, CA: Morgan Kaufmann.

Viscuso, S.R., Anderson, J.A., & Spoehr, K.T. (in press). Representing simple arithmetic in neural networks. In *Advanced cognitive science: Theory and applications.*

This research was supported by NSERC grant A7924 and a grant from the Quebec Ministry of Education. MSS is also affiliated with the Canadian Institute for Advanced Research Artificial Intelligence and Robotics Program.
E-mail: INMK@MUSICB.MCGILL.CA