[Ans, B., Rousset, S., French, R. M., & Musca, S. (2002). Preventing Catastrophic Interference in Multiple-Sequence Learning Using Coupled Reverberating Elman Networks. *Proceedings of the 24th Annual Conference of the Cognitive Science Society.* NJ:LEA]

Preventing Catastrophic Interference in Multiple-Sequence Learning Using Coupled Reverberating Elman Networks

Bernard Ans*, Stéphane Rousset*, Robert M. French† & Serban Musca*

**Experimental Psychology Laboratory*
Université Pierre Mendès-France, Grenoble 2 – CNRS UMR 5105
BP 47, 38040 Grenoble cedex 09, France
email: Bernard.Ans@upmf-grenoble.fr

† Quantitative Psychology and Cognitive Science
Université de Liège (Bât B32), Sart Tilman
4000 Liège, Belgique
email: rfrench@ulg.ac.be

Abstract

Everyone agrees that real cognition requires much more than static pattern recognition. In particular, it requires the ability to learn *sequences* of patterns (or actions) But learning sequences really means being able to learn *multiple* sequences, one after the other, without the most recently learned ones erasing the previously learned ones. But if catastrophic interference is a problem for the sequential learning of individual patterns, the problem is amplified many times over when multiple *sequences* of patterns have to be learned consecutively, because each new sequence consists of many linked patterns. In this paper we will present a connectionist architecture that would seem to solve the problem of multiple sequence learning using pseudopatterns.

Introduction

Building a robot that could unfailingly recognize and respond to hundreds of objects in the world – apples, mice, telephones and paper napkins, among them – would unquestionably constitute a major artificial intelligence *tour de force*. But everyone agrees that real cognition requires much more than static pattern recognition. In particular, it requires the ability to learn *sequences* of patterns (or actions). This was the primary reason for the development of the simple recurrent network (SRN, Elman, 1990) and the many variants of this architecture.

But learning sequences means more than being able to learn a single, isolated sequence of patterns: it means being able to learn *multiple sequences*, one after the other, without the most recently learned ones erasing the previously learned ones. But if catastrophic interference – the phenomenon whereby new learning completely erases old learning – is a problem with static pattern learning (McCloskey & Cohen, 1989; Ratcliff, 1990), the problem is amplified many times over when multiple sequences of patterns have to be learned consecutively, because each sequence consists of many new linked patterns. What hope is there for a previously learned sequence of patterns to survive after the network has learned a new sequence consisting of many individual patterns?

In this paper, we will present a connectionist architecture that solves the problem of multiple sequence learning.

Catastrophic interference

The problem of catastrophic interference (or forgetting) has been with the connectionist community for well over a decade now (McCloskey & Cohen, 1989; Ratcliff, 1990; for a review see Sharkey & Sharkey, 1995). Catastrophic forgetting occurs when newly learned information suddenly and completely erases information that was previously learned by the network, a phenomenon that is not only implausible cognitively, but disastrous for most practical applications. The problem has been studied by numerous authors over the past decade (see French, 1999 for a review). The problem is that the very property – a single set of weights to encode information – that gives connectionist networks their remarkable abilities of generalization and graceful degradation in the presence of incomplete information are also the root cause of catastrophic interference (see, for example, French, 1992).

Various authors (Ans & Rousset, 1997, 2000; French, 1997; Robins, 1995) have developed systems that rehearse on *pseudo*-episodes (or pseudopatterns), rather than on the real items that were previously learned. The basic principle of this mechanism is when learning new external patterns to interleave them with *internally-generated* pseudopatterns. These latter patterns, self-generated by the network from *random* activation, reflect (but are not identical to) the previously learned information. It has now been established that this pseudopattern rehearsal method effectively eliminates catastrophic forgetting.

A serious problem remains, however, and that is this: cognition involves more than being able to sequentially learn a series of "static" (non-temporal) patterns without interference. It is of equal importance to be able to serially learn many of *temporal sequences of patterns*. We will propose an pseudopattern-based architecture that can effectively learn multiple temporal patterns consecutively.

The key insight of this paper is this:

> *Once an SRN has learned a particular sequence, each pseudopattern generated by that network reflects the entire sequence (or set of sequences) that has been learned.*

From which our key result follows:

> When learning a new sequence, simple rehearsal with these sequence-encoding pseudopatterns will prevent catastrophic forgetting of the previously learned sequence(s).

We will use a connectionist architecture using two coupled "auto-associative recurrent networks (AARN)" (Maskara & Noetzel, 1992, 1993; Cleeremans & Destrebecqz, 1997) that pass information back and forth to each other by means of pseudopatterns. We will refer to auto-associative recurrent networks as Reverberating SRNs (RSRN), in order to emphasize the manner in which they use pseudopatterns to eliminate catastrophic interference in multiple sequence learning.

The remainder of this paper is organized as follows. We will briefly review the standard dual-network pseudopattern solution to the problem of catastrophic forgetting in static pattern learning. We will then show (Simulation 1) that multiple-sequence learning is particularly susceptible to catastrophic forgetting. We will then show how our pseudopatterns-based dual-network architecture can be used to effectively overcome catastrophic interference in multiple sequence learning.

Overcoming catastrophic interference with pseudopatterns

Before discussing catastrophic interference in multiple-sequence learning, we need to briefly describe what pseudopatterns are and how they can be used to reduce catastrophic interference in the simpler case of static pattern learning.

Assume we have a three-layer feedforward network that learns a number of binary patterns drawn from some distribution. Assume, thereafter, that these patterns are no longer available. How can one determine, even approximately, what the network has learned? Answer: By "bombarding" the input of the network with *random* binary vectors and collecting the associated output vectors. Each input-output pair of vectors produced in this way will constitute a *pseudopattern* that will be a reflection of the function previously learned by the network.

The basic idea to use pseudopatterns to reduce catastrophic interference is due to Robins (1995). It works as follows. The network learns a first set of patterns, $\{P_i\}$. Then before it begins to learn a second set of patterns, $\{Q_i\}$, noise is fed through the network to produce a set of pseudopatterns, $\{y_i\}$, as above. These pseudopatterns are added to the new patterns to be learned and the network trains on this larger set until all of the new patterns, $\{Q_i\}$, are learned to criterion. When the network is tested on the originally learned patterns, $\{P_i\}$, it has not forgotten them catastrophically. Had there been no pseudopatterns mixed in with the new patterns that were learned, the network would have completely forgotten the originally learned patterns.

Dual-Network Architectures

But where are these internally generated pseudopatterns stored so that they can be interleaved with the new patterns? One answer is to generate them on the fly in a separate network (French, 1997; Ans & Rousset, 1997, 2000). Let us consider one way that a single new pattern, *Q*, might be learned in this dual-network model. Assume that new patterns are learned by NET 1 (this can be considered to be the "Performance Network"), while previously learned patterns are stored in NET 2 (which could be considered to be the "Storage Network"). NET 1 learns *Q* as follows:

> i) pattern *Q* is input to NET 1, which modifies its weights once; ii) noise is input to NET 2, which generates a pseudopattern; iii) this pseudopattern is presented to NET 1, which modifies its weights once; iv) if *Q* has been learned to criterion, stop; otherwise go to i).

We call this the "awake state". Once the output error for pattern *Q* has dropped below criterion, we transfer the information in NET1 to NET 2 as follows:

> Loop *N* times: i) noise is input to NET 1, which generates a pseudopattern; ii) this pseudopattern is input to NET 2, which modifies its weights once.

We call this phase, when information is transferred to NET 2, the "sleep state." This will be the basis of all learning in the dual-network framework described in this paper.

"Reverberating" backpropagation

A reverberating backpropagation (RBP) network is a standard three-layer network that has a built-in autoassociator ("reverberator") for the input of the patterns to be learned (Ans & Rousset, 1997, 2000). We have shown this network in an "unfolded" manner (Figure 1). In this visualization, the output layer is divided into the "autoassociative" nodes for the input component of the patterns to be learned (on the left in Figure 1) and the "target" nodes for the targets of the patterns to be learned (on the right in Figure 1).

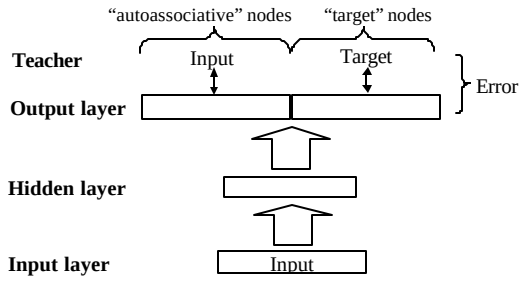

Figure 1. An RBP network

Assume the network is to learn a pattern *P: I ? T*, consisting of an input, *I*, and a target, *T*. *I* is presented on input and is sent through the network to the output layer. For all of the nodes in the output layer, an error is calculated. For the "autoassociative" output nodes,

the error is based on the difference between the network output and the original input, *I*, whereas for the "target" nodes, this error is based on the difference between the "heteroassociative" output and the desired target *T*. The errors associated with both the autoassociator and the target output nodes are then backpropagated through the network in order to change the network's weights.

"Attractor" pseudopatterns in an RBP Network

To generate pseudopatterns in a reverberating network, a random input i_y is presented to the input layer of the network and fed through to the output layer. The activation values of the "autoassociative" nodes in the output layer (nodes on the left of the output layer in Fig. 1) constitute a new input, i'_y, which is then sent through the network (the activation values on the "target" nodes in the output layer are ignored). This produces a pattern of activation on the autoassociative output nodes, i''_y, which is then presented to the input nodes of the network, and so on. After a number of reverberating cycles through the network, a final "reverberated" input, i_y^R, is sent through the network and the activation vector of all the output nodes (the "autoassociative" and the "target" output nodes), o_y, is used to produce a pseudopattern $y : i_y^R \rightarrow o_y$. The significant advantages of using an input autoassociator with a feedforward backpropagation network have been shown elsewhere (Ans & Rousset, 1997, 2000). Suffice it to say that the reverberation process transforms a pure random input, i_y, into an *attractor* of the system, i_y^R. An "attractor" pseudopattern produced provides a much better reflection of the old patterns than a pseudopattern produced from simple random noise on input. It is this reverberation technique that is largely responsible for the power of this technique.

Reverberating Simple Recurrent Networks (RSRN)

We will assume that the reader is familiar with the design of a Simple Recurrent Network (SRN, Elman, 1990). An RSRN (Figure 2) works very much like a standard SRN (Maskara & Noetzel, 1992, 1993; Cleeremans & Destrebecqz, 1997). Just as the RBP network involves adding "autoassociative" nodes to the output layer of a BP network, a reverberating SRN involves adding "autoassociative" nodes to the output layer of an SRN. The full input to the network consists of the "standard" input, i.e., the input from the sequence item, *S*(*t*), and the "context" input, *H*(*t* -1), which is the activation vector of the hidden layer associated with the previous item in the sequence, *S*(*t* -1).

"Attractor" pseudopatterns in an RSRN

The principle is identical to pseudopattern generation in an RBP network. Crucially, once the RSRN has learned a sequence of items, each "reverberated" pseudopattern generated by it reflects *the entire sequence learned by the recurrent network*. Each pseudopattern can be thought of as a very compact representation of the entire previously learned sequence.

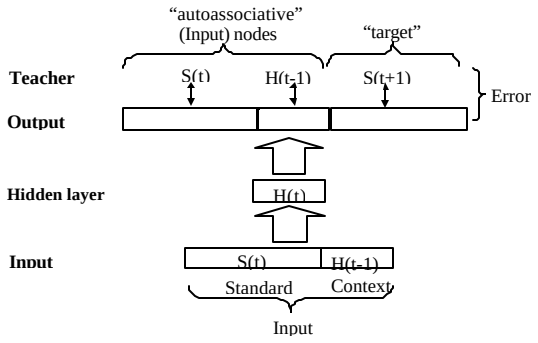

Figure 2. A Reverberating SRN (RSRN)

A dual-network architecture with self-refreshing memory to overcome catastrophic forgetting in multiple sequence learning

Ans & Rousset (1997, 2000) proposed a reverberating dual-network architecture with a self-refreshing memory to avoid catastrophic forgetting in static pattern learning. The basic dual-network architecture consists of two coupled RBP networks, denoted NET 1 and NET 2. NET 1 is the primary network that interfaces with the environment and that learns the new patterns. The secondary network, NET 2, is a "storage" network because information initially learned in NET 1 will ultimately be transferred to NET 2.

The basic principles of RSRN dual-network learning, where each of the networks is an RSRN, are virtually identical to those underlying dual-networks composed of RBP networks. Dual-network RSRNs are designed to learn multiple sequences of patterns.

Assume we have two identical RSRNs, NET 1 and NET 2. New sequences will be learned only by NET 1, while NET 2 will store the previously learned information. The learning procedure is similar to that of the basic RBP dual-network. A sequence, $S = S(0), S(1), ..., S(n)$ is presented to NET 1. The network makes a single pass through the entire sequence, updating its weights once for each item in the sequence. This defines one learning "epoch" corresponding to one presentation of all items in the sequence in order. Next, NET 2 generates a number of pseudopatterns (e.g., 10 per learning epoch). These pseudopatterns are *close to attractor states* of NET 2,

which makes them particularly good vehicles for information transfer from NET 2 to NET 1. For each NET 2 pseudopattern, NET 1 performs one feedforward-backpropagation learning pass. Once this is completed, a new learning epoch starts and NET 1 makes another pass through the sequence *S*. NET 2 generates new pseudopatterns, each of which is learned for one feedforward-backpropagation pass by NET 1. And so on. This is the awake state for an RSRN dual-network.

It is extremely important to notice that each pseudopattern generated by NET 2 is a *static input-output pattern that represents a dynamic state* (i.e., the previously learned sequence or sequences). This is what gives this system its power: there is no need to attempt to reproduce an entire pseudo-sequence that will then be interleaved with the new sequence being learned. Rather, we only need to interleave with the new sequence to be learned *non-temporal pseudopatterns*, each of which reflects (at least partially) the information in the entire previously learned temporal sequence (or sequences).

To transfer the sequence newly learned by NET 1 to NET 2, we again make use of pseudopatterns. This time the pseudopatterns are generated by NET 1 and learned by NET 2. For each pseudopattern generated by NET 1, NET 2 performs a single feedforward-backpropagation learning pass.

Overview of Simulations

We will present two simulations. The first will demonstrate the severity of catastrophic interference in multiple sequence learning in standard SRNs. The second will demonstrate that interleaving pseudo*patterns* (reflecting the whole previously learned sequence) with the new sequence effectively eliminates catastrophic interference.

A standard SRN network was used for the first simulation demonstrating the severity of catastrophic interference in multiple sequence learning. In the second simulation, a dual-network architecture consisting of two coupled RSRNs will be used. Each RSRN has an input layer with 100 "standard" input units (corresponding to the size of the items, $S(t)$, in the sequence) and 50 "context" units. The hidden layer consists of 50 units. The output layer consists of 150 "autoassociative" inputs that are identical to the input layer plus 100 "target" units (Figure 2).

Learning a given sequence consists of presenting it repeatedly to the network until each item in the sequence can predict the subsequent item with a pre-defined degree of precision. The network weights are updated by backpropagation once per presentation of each sequence item. A cross-entropy error function is used (Hinton, 1989; Plaut, McClelland, Seidenberg & Patterson, 1996) with a learning rate of 0.01, a momentum of 0.5 and a 1.0 bias term. All weights are randomly initialized between –0.5 and 0.5.

To create the "attractor" pseudopatterns, noise on input is "reverberated" 5 times before the actual pseudopattern that will be used is created. Ten pseudopatterns from NET 2 are interleaved with each epoch of the sequence learned by NET 1. During transfer of the information from NET 1 to NET 2, 10^4 pseudopatterns are used.

Measuring learning and forgetting

For each item, $S(t)$, of the sequence fed forward through the network, we calculate the difference between the activation values of "target" units in the output layer and the desired target item in the sequence, $S(t+1)$. We calculate this difference for each of the 100 "target" output units and count the number of output units for which the absolute value of this difference is above the learning criterion of 0.1. So, for example, assume a given item, $S(t)$, in the sequence is sent through the network. If, on the output layer, 14 of the "target" output units differ from the corresponding units of $S(t+1)$ by more than 0.1, we say that there are 14 "incorrect" units. A sequence is considered to have been learned if, for each of its elements, $S(t)$, the network produces a vector of "target" output values, each of which is within 0.1 of the corresponding element of $S(t+1)$. The overall measure of how well the network has learned (or forgotten) a sequence after a given number of learning epochs will be the total number of incorrect units over all items of the sequence.

Simulation 1: Catastrophic forgetting in multiple sequence learning

To illustrate the severity of catastrophic forgetting in multiple sequence learning, we will consider two sequences, *A* and *B*, and have an SRN attempt to learn them sequentially. The sequences are constructed as follows. Twenty-two distinct random binary vectors of length 100 are created. Half of these vectors are used to produce the first ordered sequence of items, *A*, denoted by $A(0)$, $A(1)$, …, $A(10)$. The remaining 11 vectors are used to create a second sequence of items, *B*, denoted by $B(0)$, $B(1)$, …, $B(10)$. In order to introduce a degree of ambiguity into each sequence (so that a simple BP network would not be able to learn them), we modify each sequence so that $A(5) = A(8)$ and $B(1) = B(5)$. First, sequence *A* is completely learned by the network. Then sequence *B* is learned and, during the course of learning, we monitor at regular intervals how much of sequence *A* has been forgotten by the network.

Fig. 3a shows the progression of the network's learning of sequence *B*. The number of incorrect units for each serial position of the sequence, as defined above, is shown. As expected, it is harder for the network to learn $B(2)$ and $B(6)$ since their immediate predecessors are identical and, in order to distinguish them, the network needs to additionally take into consideration the context of the preceding items in the sequence. After 450 epochs, the network has completely learned the entire sequence.

Then, during learning of sequence *B*, we

[Ans, B., Rousset, S., French, R. M., & Musca, S. (2002). Preventing Catastrophic Interference in Multiple-Sequence Learning Using Coupled Reverberating Elman Networks. *Proceedings of the 24th Annual Conference of the Cognitive Science Society.* NJ:LEA]

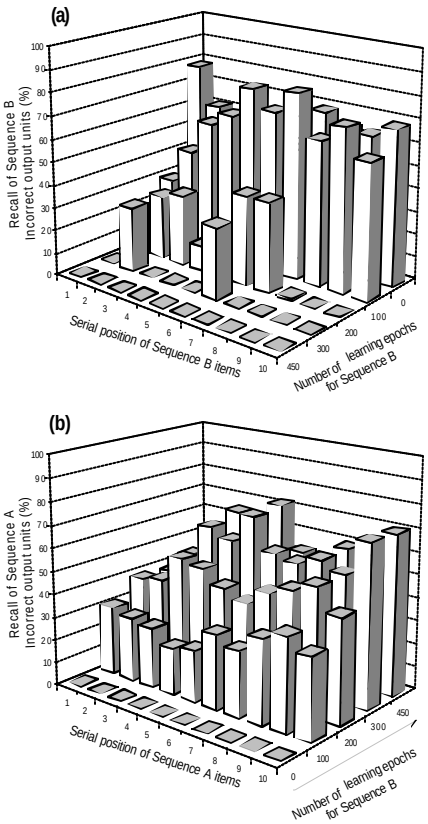

Figure 3. Catastrophic forgetting in an SRN during multiple sequence learning. **(a)**: Learning of sequence *B* (after having previously learned sequence *A*). By 450 epochs (an epoch corresponds to one pass through the entire sequence), sequence *B* has been completely learned. **(b)**: The number of incorrect units for sequence *A* during learning of sequence *B*. After 450 epochs, the SRN has, for all intents and purposes, completely forgotten the previously learned sequence *A*.

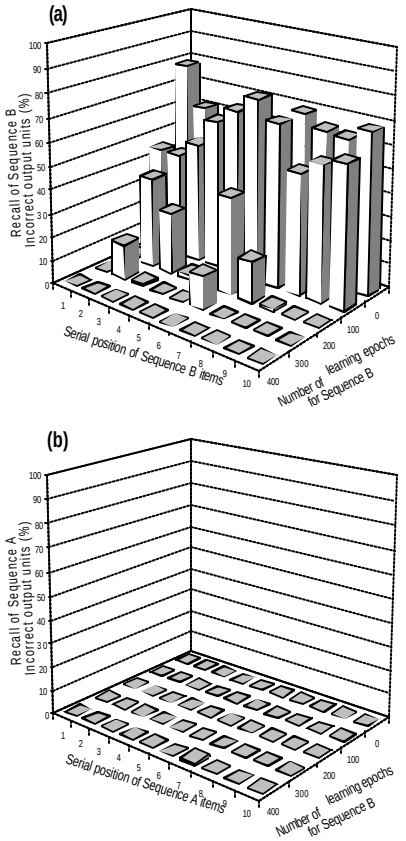

Figure 4. Recall performance for sequences *B* and *A* during learning of sequence *B* by a dual-network RSRN. **(a)**: By 400 epochs, the second sequence *B* has been completely learned. **(b)**: The previously learned sequence *A* shows virtually no forgetting. Catastrophic forgetting of the previously learned sequence *A* has been completely overcome.

monitored the forgetting of the previously learned sequence, *A* (Fig. 3b). Initially (i.e., before the network began learning sequence *B*), it can be seen that sequence *A* was completely learned. But very early on, as sequence *B* is being learned, the network's memory of sequence *A* is overwritten. By 5 epochs after beginning to learn the sequence *B* (not shown in Fig. 3b), the network gets an average of 40% of the units of the items of sequence *A* wrong. By 250 epochs, the network's performance on sequence *A* is essentially no better than chance and, by 450 epochs, sequence *A* is completely forgotten. In short, learning sequence B causes severe catastrophic forgetting of sequence A.

Simulation 2: Catastrophic forgetting is overcome with pseudopatterns

An RSRN dual-network architecture was used with the parameters indicated above. Both networks are initialized to random weight settings between −0.5 and 0.5. NET 1 then completely learns sequence *A* and then generates 10^4 pseudopatterns in order to transfer this learning to NET 2. (There are 2^{150} possible distinct states for the input layer of each network, and hence, there is a very little possibility that the random binary vectors used to produce the "attractor pseudo-input" would be actual input patterns already seen by the network.)

Now, NET 1 begins to learn sequence *B*. After each learning epoch (consisting of the entire sequence of items in *B*), NET 1 receives 10 pseudopatterns from NET 2 and does one feedforward-backpropagation pass for each of them. (The number of pseudopatterns is not related to the length of the previously learned sequences and can be varied.)

Fig. 4a shows that the NET 1 does, in fact, learn sequence *B* completely by 400 epochs. In other words, for all items in sequence *B*, all of the units in the network output are within 0.1 of the desired target

output. Notice that the sequence items $B(2)$ and $B(6)$ are learned more slowly by the network. This was, of course, expected since these two items are preceded by identical items, hence creating ambiguity having to be solved by the temporal context. During learning of sequence *B*, we tested the performance of the network on all of the items of sequence *A*. Fig. 4b shows that *there is virtually no forgetting of sequence A as the network learns sequence B.* In short, catastrophic forgetting has been completely overcome by the coupled system of RSRNs using pseudopattern information transfer.

Concluding remarks

We have shown that the reverberating dual-network architecture, originally proposed earlier to overcome catastrophic forgetting in non-temporal pattern learning (Ans & Rousset, 1997, 2000) can be generalized to sequential learning of multiple temporal sequences. The basic principle of interleaving internally-generated pseudopatterns from a long-term storage network with patterns from the environment being learned by a second network has been developed elsewhere (Ans & Rousset, 1997, 2000; French, 1997; Robins, 1995).

When learning multiple sequences of patterns, it turns out that interleaving simple input-output pseudopatterns, each of which reflect *the entire previously learned sequence(s)*, reduces (or eliminates entirely) forgetting of the initially learned sequence(s).

Further, we demonstrate the power of a network architecture that allows us to produce "reverberated" pseudopatterns that are, in reality, attractors of the entire network and therefore reflect, in a highly compressed manner, the previously learned sequences.

We have demonstrated a technique that effectively allows multiple sequences to be learned consecutively. Of course, these networks can be made to forget gradually. This gradual forgetting depends on the size of the learning network and on the number, the overlap, the length and the complexity of the successively learned sequences. But the problem of sudden, "catastrophic" forgetting of previously learned sequences caused by learning a new sequence of patterns would seem to have been overcome.

Acknowledgments

This research was supported in part by a research grant from the European Commission (HPRN-CT-1999-00065) and by the French government (CNRS UMR 5105).

References

Ans, B. & Rousset, S. (1997) Avoiding catastrophic forgetting by coupling two reverberating neural networks. *C.R. Acad. Sci. Paris, Life Sciences*, 320, 989–997.

Ans, B. & Rousset, S. (2000) Neural networks with a self-refreshing memory: knowledge transfer in sequential learning tasks without catastrophic forgetting. *Connection Science*, 12, 1–19.

Cleeremans, A. & Destrebecqz, A. (1997). Incremental Sequence learning. In *Proceedings of the Nineteenth Annual Meeting of the Cognitive Science Society*, NJ: LEA, 119-124

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

French, R.M. (1992) Semi-distributed representations and catastrophic forgetting in connectionist networks. *Connection Science*, 4, 365–377.

French, R.M. (1997) Pseudo-recurrent connectionist networks: An approach to the 'sensitivity-stability' dilemma. *Connection Science*, 9, 353–379.

French, R.M. (1999) Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3, 128–135.

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

Maskara, A., & Noetzel, A. (1992). Forced simple recurrent neural network and grammatical inference. In *Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society*, 420-425, NJ: LEA.

Maskara, A. & Noetzel, A. (1993). Sequence learning with recurrent neural networks. *Connection Science, 5*, 139-152.

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

McCloskey, M. & Cohen, N.J. (1989) Catastrophic interference in connectionist networks: The sequential learning problem. In G.H. Bower (Ed.), *The Psychology of Learning and Motivation*, Vol. 24. New York: Academic Press, pp. 109–165.

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

Ratcliff, R. (1990) Connectionist models of recognition memory: Constraints imposed by learning and forgetting functions. *Psychological Review*, 97, 285–308.

Robins, A.V. (1995) Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science*, 7, 123–146.

Sharkey, N.E. & Sharkey, A.J.C. (1995) An analysis of catastrophic interference. *Connection Science*, 7, 301–329.