I, John R. Riesenberg, hereby submit this as part of the requirements for the degree of: __Master of Arts__

in: __Psychology__

It is entitled: __Catastrophic Forgetting in Neural Networks__

Approved by: Richard Honeck, Ph.D.

Peter Chiu, Ph.D.
CATASTROPHIC FORGETTING IN NEURAL NETWORKS

A thesis submitted to the
Division of Research and Advanced Studies
of the University of Cincinnati

in partial fulfillment of the
requirements for the degree of

MASTER OF ARTS

in the department of Psychology
of the College of Arts and Sciences

2000

by

John R. Riesenber

B.S., University of Cincinnati, 1985
B.A., University of Cincinnati, 1985

Committee Chair: Richard Honeck, Ph.D.
ABSTRACT

This paper explores the phenomenon of ‘catastrophic forgetting’ in artificial neural networks (ANNs). Catastrophic forgetting simply is the inability of an ANN to learn a second set of information without forgetting what it previously learned. For the past decade, catastrophic forgetting or, as it is sometimes called ‘catastrophic interference’, has baffled many researchers trying to solve this problem.

McCloskey and Cohen (1989) attempted to recreate the Barnes and Underwood (1959) study of retroactive inhibition with an ANN simulation, but quickly found that their backpropagation ANN model did not simulate human learning very well. They found that because knowledge is highly distributed throughout the network, learning a second set of items causes the network to create a new solution space based on the newly learned information (McCloskey and Cohen, 1989). This approach mistakenly led many to believe that solving the catastrophic forgetting problem involved eliminating overlapping hidden representations in the network.

Robins (1995) developed the sweep pseudorehearsal procedure by viewing a trained network as a kind of function approximator. This approach unfortunately lacked plausibility as a cognitive model. French (1997) and Ans and Rousset (1997) each built dual-network architectures based on the evidence gathered by McClelland, McNaughton, and O'Reilly (1995) that the brain solved the catastrophic forgetting problem by evolving the hippocampus and neocortex into complementary learning systems. These dual-network architectures used the
sweep pseudorehearsal technique to pass a network’s knowledge to another network to overcome the catastrophic forgetting in backpropagation networks.

To prepare the reader for the technical discussions about catastrophic forgetting, three popular ANN models are presented (backpropagation, Hopfield, and SDM) to give a flavor of how ANN models work.
ACKNOWLEDGMENTS

I would like to thank:

Dr. Honeck for his guidance and support
in helping me complete this program,

Dr. Chiu for his contributions on this subject,

Lynn Yosua, my wife, who with love encouraged me to complete this project
knowing that she would carry the majority of the burden raising our child, Kevin,
during this period,

Mary Gontero and Bruce Hemmerich for reading my initial drafts
ensuring their readability,

Marc Siemer, my manager, and Ken Roth, Director of Information Resources,
both at Cinergy Corporation, for encouraging me to reach this goal and providing
me with the work schedule flexibility necessary to complete this project,

and the People at Cinergy’s Energy Commodities Business Unit who through
their patience allowed me to concentrate on this work.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>5</td>
</tr>
<tr>
<td>Human Sequential Learning and Catastrophic Forgetting</td>
<td>7</td>
</tr>
<tr>
<td>The Barnes and Underwood Study</td>
<td>8</td>
</tr>
<tr>
<td>Catastrophic Interference in ANNs</td>
<td>9</td>
</tr>
<tr>
<td>Why does new learning disrupt old knowledge?</td>
<td>11</td>
</tr>
<tr>
<td>Catastrophic forgetting results from overlapping representations</td>
<td>17</td>
</tr>
<tr>
<td>Simulating the Brain – a Connectionist Approach</td>
<td>19</td>
</tr>
<tr>
<td>Biological Neural Nets and Their Simulations</td>
<td>19</td>
</tr>
<tr>
<td>The Learning Rule</td>
<td>23</td>
</tr>
<tr>
<td>ANN Models</td>
<td>25</td>
</tr>
<tr>
<td>Backpropagation Model</td>
<td>25</td>
</tr>
<tr>
<td>Hopfield Model</td>
<td>27</td>
</tr>
<tr>
<td>Semi-distributed Network Models</td>
<td>35</td>
</tr>
<tr>
<td>Attempts at Solving Catastrophic Forgetting</td>
<td>39</td>
</tr>
<tr>
<td>Reducing Overlapping Distributed Representations</td>
<td>40</td>
</tr>
<tr>
<td>Rehearsal and Pseudorehearsal Network Training</td>
<td>42</td>
</tr>
<tr>
<td>Complementary Learning Systems – A Dual-Network System</td>
<td>48</td>
</tr>
<tr>
<td>Conclusions</td>
<td>58</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>1</td>
<td>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.</td>
</tr>
<tr>
<td>2</td>
<td>Percentage correct on the A-B and A-C list in the McCloskey and Cohen (1989) study as a function of number of trials on the A-C list.</td>
</tr>
<tr>
<td>3</td>
<td>A simple three-unit network and two training pattern (McCloskey and Cohen, 1989, p. 149).</td>
</tr>
<tr>
<td>4A</td>
<td>Solution space for pattern 1 (McCloskey and Cohen, 1989, p. 150)</td>
</tr>
<tr>
<td>4C</td>
<td>Solution space for pattern 1 (dark shading), solution space for pattern 2 (light shading), and overall solution space (solid region) (McCloskey and Cohen, 1989, p. 150).</td>
</tr>
<tr>
<td>5A</td>
<td>Movement of the weight configuration over learning trials with concurrent training on pattern 1 and 2 (McCloskey and Cohen, 1989, p. 152).</td>
</tr>
<tr>
<td>5B</td>
<td>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) (McCloskey and Cohen, 1989, p. 152).</td>
</tr>
<tr>
<td>5C</td>
<td>Movement of the weight configuration over learning trials with sequential training on pattern 2 and then on pattern 1 (McCloskey and Cohen, 1989, p. 152).</td>
</tr>
<tr>
<td>6</td>
<td>Schematic drawing of a typical neuron (Hertz, Krough, and Palmer, 1991, p. 2).</td>
</tr>
<tr>
<td>7</td>
<td>Anatomy of backpropagation neurons used in catastrophic forgetting experiments.</td>
</tr>
<tr>
<td>8</td>
<td>Backpropagation model topology.</td>
</tr>
</tbody>
</table>
Schematic configuration space of a Hopfield model with three attractors. This picture is very idealized and in particular the space should really be a discrete set of points (on a hypercube), not a continuous region. Nevertheless, it is a very useful image to keep in mind (Hertz et al., 1991, p. 13).

This figure shows the Hopfield model. Notice there are no feedback connections to each neuron. Eliminating these connections explicitly forces the weight matrix to have zeros on the diagonal. I have also added external input signals, $I_i$, to each neuron (Freeman and Skapura, 1991, p. 143).

A three neuron Hopfield network (Perry, 1998) and its associated connection weight matrix.

Connection weight matrix for the Parry (1998) example after training the network on the patterns $[0 \ 1 \ 1 \ 0 \ 1]$ and $[1 \ 0 \ 1 \ 0 \ 1]$. The example calculates the dot product of the column out of the connection weight matrix representing the neuron we are updating and the current state to calculate the new weight. For instance, if we are updating the third neuron, then we use the third column of the weight matrix for the dot product calculation.


Interpreting learning in a neural net as function approximation. Learning fits a function to the training population items, with a range of functions are possible, including ‘compact’ and noisy functions (Robins, 1995, p. 136).

New items are learned without rehearsal, the new learned function may not be similar to the old function describing the original population (Robins, 1995, p. 136).

New items are learned with recency rehearsal and sweep rehearsal, the new learned function preserves much of the shape of the old function (Robins, 1995, p. 136).

New items are learned with pseudorehearsal using pseudoitems. The new learned function preserves much of the shape of the old function (Robins, 1995, p. 136).
Summary of the results presented for base population goodness over 10 trials. Conditions shown are sweep rehearsal, sweep pseudorehearsal, recency rehearsal, and no rehearsal (Robins, 1995, p. 133). “The ‘Goodness’ measure is the network’s ability to correctly reproduce the approximate outputs for a given set of inputs. It is calculated by averaging for each input-output pair the normalized dot product of the target vector and the actual output vector observed (see Ratcliff, 1990, p. 288). Vectors are transformed so that a goodness of 1 indicates a perfect match and a value of 0 indicates chance performance (50% match)” (Robins, 1995, p. 141).

French’s (1997) dual-network architecture. Stage (I): the NET 1 network is learning new items along with pseudoitems generated by NET 2. Stage (II): the NET 2 network is learning pseudoitems generated by NET 1 (transport of NET 1 memory towards NET 2).

Improved recognition of the original data with respect to standard backpropagation (measured n per cent of the originally learned patterns that were exactly recognized after the new patterns had been learned) (French, 1997, p. 374).

The reverberating architecture. Stage (I): the NET 2 network is learning pseudoitems generated by the reverberating process in NET 1 (transport of NET1 memory towards NET 2). Stage (II): the NET 1 network is learning external items along with pseudoitems generated by the reverberating process in NET 2 (Ans and Rousset, 2000, p. 4).

Mean goodness of the old base as a function of the number of training cycles of the new base. Graphs (plotted for each set of ten cycles) stop when the new base learning is completed and the full square (the ■ on the y-axis) refers to the initial goodness of the old base before the new base training starts (Ans and Rousset, 1997, p. 996).
Introduction

In astronomy, researchers use computers to simulate fusion and the consumption of nuclear fuel to understand the process stars undergo when they supernova. Because researchers cannot physically manipulate and control the atoms that are in a star, they apply their understanding of physics to computer simulations to test their theories. By comparing the simulation results to what they observe, researchers can determine how well they understand the dying process of a star, how stars supernova, and how stars explode gases into space allowing the formation of new stars (Benningfield, 2000).

Similarly, in the psychological, cognitive science and neuro-psychological communities, researchers are using the computer to simulate both brain processes and the interaction between neurons. Researchers compare the results from their simulations to the results of more empirical research to understand how different parts of the brain may function, how learning occurs, and how our brains create memories. To date, researchers have been able to use these neuronal simulations, known as Artificial Neural Networks or ANNs, to create artificial visual systems and artificial noses, understand how the brain “sees” patterns, and observe a computer “learn.” These triumphs, unfortunately, are modest compared to many of the skills found in humans. Humans, for example, can learn serially a list of paired words and then later, learn a second list of paired words. After learning the second list, humans forget only a small fraction of the pairs in the first list (Barnes and Underwood, 1959). ANNs, on the other hand, learn quite well the first list of pairs, but after learning the second list
of pairs, completely forget the first list. Researchers have coined the term “catastrophic forgetting“ or “catastrophic interference “ to describe this phenomenon in ANNs (McCloskey and Cohen, 1989; Ratcliff, 1990).

Many believe that the brain is capable of sequential learning because it has solved the catastrophic forgetting problem. The best framework for modeling cognition lies in the best combination of ANN architecture and memory consolidation algorithms (Robins and McCallun, 1999). ANN architectures vary widely in how computer simulated neurons connect to each other, how the strength of neuron connections are determined and how the simulated neuronal networks go about “learning.” Memory consolidation algorithms, in general, specifically try to solve the catastrophic forgetting problem directly by reducing the representational overlap found inside the network. Most ANN researchers, however, believe that to truly solve the catastrophic forgetting problem, it will take both empirical and simulation research. In this essay, I will examine the work done on both fronts and explore the work that seems to show the most promise.

I will begin with a review of the topics of human sequential learning and catastrophic forgetting. A review of Barnes’ and Underwood’s (1959) classic study of human sequential learning leads off, establishing the benchmark against which many computer simulations of serial learning are compared. In then discuss McClosky and Cohen’s (1989) revelation that ANNs suffer from catastrophic forgetting and explain what causes catastrophic forgetting in ANNs.

Next, I describe how ANNs are simple models of neuronal interactions occurring inside the brain. I continue by examining some of the models used in
research, such as backpropagation, Hopfield, and SDM models. This will provide a flavor of how ANN models work and show that not all ANN models suffer from catastrophic forgetting. Finally, I explore attempts to solve the catastrophic forgetting problem, outlining the successes and failures for each.

The paper concludes by summarizing the more promising ANN architectures and learning algorithms for solving the catastrophic forgetting problem. Progress is assessed and suggestions are made for moving our understanding to the next level.

Human Sequential Learning and Catastrophic Forgetting

One characteristic innate in humans and animals is their ability for cumulative learning over time and events. As Sharkey and Sharkey (1995, p. 302) note, “In terms of adult learning, it is clearly possible to continually refine and update knowledge without forgetting all that was previously known about it.” By testing what knowledge can transfer when a new task is confronted, changes from its original task, researchers can determine how humans attain some insight into relevant relationships. In terms of long-term memory, researchers have devised theories to try to explain how memory works or fails. One such theory of long-term memory is interference theory which describes the apparent loss of a memory from among a number of other memories that interfere with the recovery of the memory that is sought. By studying both transfer of knowledge and memory interference, researchers can gain greater knowledge about how humans learn sequentially.
The Barnes and Underwood Study

In a 1959 study, Barnes and Underwood (1959) examined retroactive inhibition in serial learning. Retroactive inhibition (RI) is the learning of new material that hampers the recall of old material. Barnes and Underwood wanted to determine what contributed to RI. They expected to find that the extinction or unlearning of previously learned material contributed to RI because earlier studies suggested that memory for previously learned material gradually decreased as new material was learned.

To test their hypotheses, Barnes and Underwood used four groups. These groups were given two lists of eight nonsense word paired-associates to learn, an A–B list and an A–C list. List 1, the A–B list, was presented to the subjects until all word pairs were learned to one perfect trial. Then, list 2, the A–C list, was presented for a specified number of trials, namely 1, 5, 10 or 20 trials, respectively, for the four groups. After the specified number of trials, learning of list 2 stopped. Each subject was then presented a sheet of paper containing the eight “A” words. The subjects were asked to write down two responses, one from list 1 and one from list 2 (the “B” and “C” words), which they thought were associated with each “A” word.

The results are presented in Figure 1. As the number of learning trials on list 2 increases, the number of correct responses for this list increases, whereas list 1 responses gradually decrease. As Barnes and Underwood state, “It is as if the A–B associations [list 1] are weakened or extinguished during the learning of A–C [list 2]” (Barnes and Underwood, 1959, pp. 100 - 101).
To see if this was simple forgetting, Barnes and Underwood used a control group of 12 subjects. This group learned list 1 to one perfect trial and then rested for a period of time equal to that spent in learning list 2 by the group given 20 trials. After their rest, the control subjects were given a sheet of paper where the eight “A” words were presented and asked to write down a response for its corresponding “B” word. The mean recall was 7.75 out of the eight word pairs. Thus, the decrement in recall of list 1 as a function of trials on list 2 is not simple forgetting, but must result from the learning of list 2.

**Catastrophic Interference in ANNs**

Some 30 years later, McCloskey and Cohen (1989) used an ANN to roughly simulate Barnes and Underwood’s RI study. Instead of using nonsense words, however, McCloskey and Cohen substituted “words” represented by 10 binary digits (e.g. [0 0 1 0 1 1 1 0 0 0 ]). These “words” were generated randomly.
and, just like Barnes and Underwood, divided into two groups (list 1 and list 2) of eight paired-associates. To distinguish between the groups, a 10-digit binary context word allowed the ANN model to determine whether it was learning list 1 or list 2.

Now, in order for an ANN to demonstrate what it has learned, or in this case, not learned, it needs training. Training an ANN on a set of items involves a series of learning trials. On each trial, the items in the training set (in our case, the “A” words and their corresponding context word) are presented to the network. The goal of the training is to get the network to produce the corresponding “B” or “C” word as the ANN’s output. The network’s output is compared to the target output, and on the basis of this comparison the network adjusts itself internally. On each successive trial, the network adjusts itself until it produces a close approximation to the targeted output, the corresponding “B” or “C” word.

McCloskey and Cohen trained the ANN on list 1 as a single training set, which was presented repeatedly over a series of learning trials until good performance was achieved. Afterwards, the ANN began training on list 2. Between list 2 training sets, however, performance on list 1 and list 2 recall was assessed by presenting the ANN with the “A” word and list context word, and comparing the output to the corresponding “B” or “C” word. “After three learning trials on the A–C list [list 2], which is sufficient to produce only about 20% correct responses on this list, performance on the A–B list [list 1] fell from 100% to 0%
correct, and remained at or near 0% thereafter” (McCloskey and Cohen, 1989, p. 128). See Figure 2.

Figure 2. Percentage correct on the A-B and A-C list in the McCloskey and Cohen (1989) study as a function of number of trials on the A-C list.

Why does new learning disrupt old knowledge?

As you can see by comparing figures 1 and 2, the evidence clearly shows the difference between the gradual forgetting of list 1 found by Barnes and Underwood, and the catastrophic forgetting of list 1 found by McCloskey and Cohen, as list 2 is learned. Ratcliff (1990), Lewandowsky (1991), and French (1992) each confirmed McCloskey and Cohen’s results. Nevertheless, what causes ANN’s to exhibit this problem? What is happening inside the neural network to cause new learning to replace the old knowledge? Several ANN based kinds of explanations will now be considered.
The problem results from how the knowledge is stored. One reason why ANNs show catastrophic forgetting involves the way they store knowledge. During training, ANNs store their knowledge according to the strength of the connections between the neurons. The mechanism inside the ANN that determines the strength of a connection between the neurons is the connection weight. As the network trains on each pattern, the connection weights adjust to produce the desired pattern. In McCloskey and Cohen’s experiment, as the network trains on each “A” word, the network adjusts itself by adjusting its connection weights to produce the corresponding “B” word. Thus, each connection weight is involved in responding to many different inputs.

When additional training occurs for an already trained network, the connection weights adjust again. These further adjustments not only encode the desired response to the new input pattern, but alter the network’s response to other inputs as well. “In many respects this is a desired 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” (McCloskey and Cohen, 1989, p. 147).

Ok, so how does this cause ANNs to forget? To further clarify the explanation, I will use McCloskey’s and Cohen’s (1989) example. Figure 3 presents a very simple network consisting of two input nodes. Nodes P and Q
connect to a single output node $R$. The network is trained on the two patterns depicted in the figure. In pattern 1, both input patterns are on, and the output node is on. For pattern 2, node $P$ is on, node $Q$ is off, and output node $R$ is off.

$$\begin{array}{ccc}
\text{Input} & \text{Output} \\
\hline
P & Q & R \\
\hline
\text{Pattern 1} & 1 & 1 & 1 \\
\text{Pattern 2} & 1 & 0 & 0 \\
\end{array}$$

Figure 3. A simple three-unit network and two training patterns (McCloskey and Cohen, 1989, p. 149).

Now, the configuration of connection weights can be thought of as a point in a multidimensional space. The number of weights equals the number of dimensions. The position on each dimension represents the value of the corresponding weight. In McCloskey’s and Cohen’s two-connection network, the weight configuration is a point on a plane with the $x$ axis representing the $P-R$ weight and the $y$ axis representing the $Q-R$ weight.

Without getting into the details for the specific calculations (see McCloskey and Cohen, 1989, pp. 148-153 for mathematical details), Figure 4A depicts any weight configuration in the dark shaded region that will yield a correct response for pattern 1. For pattern 2, Figure 4B shows any weight configuration in the light shaded region that will yield a correct response. It is easy to see that if
we want the network to respond correctly to both patterns, the weight configuration must be somewhere in the intersection of the two solution spaces, as shown in the solid triangular area in Figure 4C. We will call this intersection of the two solution spaces the overall solution space.

Concurrently training the network on both pattern 1 and pattern 2, Figure 5A presents the movement of the weight configuration over learning trials. As you can see, the weight configuration moves rather directly to the overall solution space. As McCloskey and Cohen (1989) put it, “each pattern to be learned may be thought of as pulling the weight configuration toward the solution space for that pattern” (McCloskey and Cohen, 1989, p. 151). When both patterns are trained concurrently, both patterns pull simultaneously, causing the weight configuration to move toward the region of space that is good for both patterns, the overall solution space.

What happens when pattern 1 is trained first and pattern 2 is trained second? The weight configuration moves toward the solution space for pattern 1, as the line labeled 1 shows in Figure 5B. Then when training pattern 2, the weight configuration moves toward the solution space for pattern 2. When training is complete, the network responds correctly for pattern 2 but no longer yields a correct response for pattern 1. During this sequential training, there is nothing to prevent pattern 2 from pulling the weight configuration out of the solution space for pattern 1. And, there is nothing to prevent the weight configuration from moving away from the overall solution space.
Figure 4A. Solution space for pattern 1 (McCloskey and Cohen, 1989, p. 150).

Figure 4B. Solution space for pattern 2 (McCloskey and Cohen, 1989, p. 150).

Figure 4C. Solution space for pattern 1 (dark shading), solution space for pattern 2 (light shading), and overall solution space (solid region) (McCloskey and Cohen, 1989, p. 150).
Figure 5A. Movement of the weight configuration over learning trials with concurrent training on pattern 1 and 2 (McCloskey and Cohen, 1989, p. 152).

Figure 5B. 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) (McCloskey and Cohen, 1989, p. 152).

Figure 5C. Movement of the weight configuration over learning trials with sequential training on pattern 2 and then on pattern 1 (McCloskey and Cohen, 1989, p. 152).
Similarly, when training pattern 2 before pattern 1, the weight configuration moves toward the solution space for pattern 2. Then, when training for pattern 1 begins, the weight configuration moves completely out of the solution space for pattern 2 into pattern 1’s solution space, as shown in Figure 5C.

To conclude, if training alternates between the two patterns, the weight configuration zigzags toward the overall solution space. However, if training is sequential, the resulting weight configuration is unlikely to be appropriate for the initially learned pattern.

Catastrophic forgetting results from overlapping representations

Kortge (1990) suggested that the problem of catastrophic forgetting is not the result of distributing the knowledge throughout the network. Rather it is due to the ANN’s rules for learning. In other words, can we do something that prevents the learning of a second pattern from moving the weight configuration away from the overall solution space? Kortge devised a solution such that when the ANN adjusts itself to make the output closer to the target, it adjusts only those connection weights that need adjusting. Kortge believed that adjusting all connection weights leads to interference with other patterns’ output and increases the likelihood that the weight configuration moves farther away from the overall solution space. By changing the rules for learning to compute some measure of novelty to the target and using this novelty measure to limit the number of connection weight changes throughout the network, the network could demonstrate a reduction of catastrophic forgetting. The bigger the novelty measure, the larger the number of connection weight changes. The effect
reduces the amount of overlap between the input representations of the new patterns and the previously learned patterns. The novelty vector algorithm, as Kortge predicted, produced a moderate decrease in catastrophic forgetting.

This suggested to French (1992) that, in general, the key to solving the catastrophic forgetting problem involves reducing the overlap of internal distributed representations. He figured that reducing the degree of representational overlap reduces catastrophic forgetting. French demonstrated his theory by using a method called “activation sharpening”, where the most active node’s activation level is increased slightly and the other nodes’ activation level is decreased slightly. This, unfortunately, yielded limited success. The method showed promise for sharpening of one node. However, sharpening more than one node produced little or no improvement, contrary to what he expected. Still, French believed that knowledge should be somewhat localized, enough to limit the overlap of internal distributed representations, yet not so much that the network could not generalize. French, therefore, advocated the notion of semi-distributed ANN architectures.

Although semi-distributed ANN architectures do not exhibit catastrophic forgetting and support French’s opinion, questions were left unanswered. In particular, are there highly distributed representations not affected by catastrophic forgetting (French, 1992)? In the following sections, I describe the components of an ANN model; present the backpropagation, Hopfield, and semi-distributed class of ANN architectures; and describe the strategies deployed in trying to answer French’s question.
Simulating the Brain – a Connectionist Approach

Up to this point, I have described catastrophic interference in ANNs while sidestepping the internal working details of ANNs. ANNs are just one attempt at modeling how the brain may function. As researchers discover new facts about brain functioning, other researchers develop programs and algorithms to test what was learned. There are many theories that attempt to explain the inner workings of the brain, all with varying degrees of success. In this section, I first, discuss the evidence that supports a connectionist view of the brain, followed by, the definition of terms and components used in most ANN models. These terms show their relationship to their corresponding biological counterpart. This allows for an easier discussion when I describe the ANN models themselves.

Biological Neural Nets and Their Simulations

Hopfield (1982) suggested that a network of interconnected, simple processors, as a whole, creates a stable memory and that collective computational properties spontaneously arise. This may explain how nerve cells or neurons in the brain perform many cognitive operations faster than a conventional computer although the neuron is slower than computer components by a factor of $10^6$ (Rumelhart, 1989). This also means that human processes that take on an order of a second or less (e.g. perception, memory retrieval, speech processing) can only involve a hundred or so time steps. Feldman (1985) called this the “100-step-program” constraint. Feldman believes that explanations for mental phenomena should not involve more than a hundred sequential operations. Since these processes tend to be quite complex, our algorithms must
involve considerable parallelism. ANNs are an attempt to simulate this parallelism.

To understand how an ANN works, let us first look at the mechanics of passing a message through an animal's nervous system. The nerve cell is the simplest element in an animal's nervous system. A nerve cell or neuron has three subdivisions: the dendrites, the cell body, and the axon (Figure 6). Dendrites branch out, connect to other neurons and receive impulses from these other neurons. The axon is a very long fiber that forks out into several branches at the end, connecting to yet other neurons. The axon transmits an impulse to the neurons that connect to it. There are an estimated 100 billion neurons in the human brain each interconnecting to each other with multiple connections at the ends of the dendrites and at the end of the axon endings.

Neurons send impulses to each other as electrical voltages. Dendrites sum up the voltages they receive and compare them to a threshold or activation point. If the total input voltage exceeds the activation point, the neuron “fires”, sending a voltage out its axon to the neurons that connect to it. Therefore, as neurons pass voltages to each other, a message can pass through the network. Of course, activation points and neuron voltage outputs vary from neuron to neuron – allowing a message to evolve as it moves through the network.
ANN programs simply model this neuronal process. Just like nerve cells, an ANN program has a collection of simple processing units called neurons or units. A matrix of connection weights represents the axon and dendrites of each nerve cell, describing both the strength of the connections and the pattern of connectivity between the neurons. Each neuron has an activation rule that describes how the inputs to the neuron combine and an output function that describes the output to pass onto other neurons. Using this activation rule, the inputs to the neuron compare against the activation point to determine the output to pass along to the other neurons (Hertz, Krough & Palmer, 1991).

The matrix of connection weights, as mentioned earlier, is where the knowledge is stored. It determines the amount of input to a neuron and the connections between the neurons. A positive connection weight represents an excitatory input; a negative weight represents an inhibitory input. A connection
weight of zero means no connection. The absolute value of a connection weight ($|w_{ij}|$) represents the connection strength.

The activation rule is usually a deterministic function that defines the total input to the neuron. It normally is the weighted sum of the separate inputs ($\sum w_{ij}o_j$, where $o_j$ is the separate inputs and $w_{ij}$ is the connection weight) compared against the activation point. Because different models have different assumptions, the activation rule can have different mathematical properties. For instance, it can also be stochastic or random, obeying the laws of probability. Or, it can be continuous such as a sigmoid (s-shaped) function, allowing the individual neuron to reach a minimum or maximum value of activation. As a final alternative, the activation rule could simply decay slowly over time, so that without external input, activation does not directly go to zero (Rumelhart, 1989). In the ANN models described so far, the researchers used the deterministic activation rule.

The result of the activation rule is the activation state or activation level for the neuron at time $t$. It represents the pattern of activation over the set of neurons. This gives us a mechanism to understand the behavior of the network as it evolves over time.

The output function simply maps the current state of activation to the output signal. Most of the time, the output function is the activation state. (In mathematical terms, the output function is the identity function, $f(x) = x$, where a function takes any value $x$ and returns $x$.) Sometimes the output function is some kind of threshold function, so that a neuron has no effect on another neuron.
unless its activation exceeds a certain value. Still other times, the output function
is a stochastic function where the activation state probabilistically determines the
neuron’s output (Rumelhart, 1989). Again, the ANN models described so far
used an output function that is the same as the activation state. Figure 7
summarizes the anatomy for an artificial neuron used in most catastrophic
forgetting experiments.

![Figure 7. Anatomy of backpropagation neurons used in catastrophic forgetting
experiments.](image)

**The Learning Rule**

To this point, I have mapped the similarities between the biological neural
net and the artificial neural net. Unfortunately, researchers do not understand
how a biological neural net learns. Researchers therefore devise mechanisms
with their theories to allow ANNs to learn – the *learning rule*.

Learning rules are methods for modifying the connection weight strength
through experience. Typically, if a neuron receives input from another neuron
when both neurons are highly active, then the connection between the neurons is
strengthened (Rumelhart, 1989). This kind of learning rule is the *Hebbian*
learning rule. Simply, this is a function using a “teacher” (or target) and an activation level to decide how much to change the connection weight. To help the ANN avoid oscillations around a solution space or slow its convergence toward an adequate solution, a learning rate is employed to control the amount of weight modification (Ratcliff, 1990, p. 287).

A simplification of the Hebbian learning rule is the Widrow-Hoff learning rule or delta learning rule. It uses the difference (or delta) between the actual activation achieved and the target activation provided by the teacher to decide how much to change the connection weight. (In mathematical terms, $\Delta w_{ij} = \eta \cdot (\tau_i - a_i) \cdot o_j$, where $\eta$ is the learning rate, $\tau_i$ is the teacher, $a_i$ is the activation level, and $o_j$ is the output of the neuron) (Rumelhart, 1989). The catastrophic forgetting experiments discussed so far used the delta learning rule.

Finally, the last learning rule is the perceptron learning rule. Here, neurons with the strongest activation learn. Typically, researchers use the perceptron learning rule in the study of competitive learning. Grossberg (1987) used the perceptron learning rule in his competitive learning model ART.

To this point, I have discussed those components that are common to ANN models. Missing, however, are two vital pieces that differentiate ANN models – network topology and learning rule implementation. Let us use these to describe the various ANN models.
ANN Models

I present in this section the two most prominent ANN models, Backpropagation and Hopfield. I also look at the class of semi-distributed architectures advocated by French (1992) as a solution for the catastrophic forgetting problem and give an example of one model called Sparse Distributed Memory (SDM).

Furthermore, my goal is to introduce the reader to some of the better-known ANN models. I discuss these models in their most basic terms. I spare the reader the underlying mathematics (except where it helps the reader’s understanding) so no special or extensive mathematical training is required to understand each model. If the reader wants more details, I supply sufficient references for further exploration.

Backpropagation Model

The backpropagation topology arranges the neurons into layers and each layer contains any number of neurons. There are three kinds of layers. (1) The *input layer* receives only input into the network. (2) The *output layer* returns the result out of the network. (3) The *hidden layers* are sandwiched between the input and output layers. There can be any number of hidden layers, whereas there is only one layer for both the input and output layers. Connections between neurons exist only between neurons in adjacent layers. Thus, neurons in the input layer connect only to neurons in the first hidden layer and neurons in the last hidden layer connect only to neurons in the output layer. Neurons in other hidden layers connect only to neurons in adjacent hidden layers. This implies that
the flow of the network begins at the input layer and moves through the various hidden layers to the output layer ("feed-forward") to produce an output (see Figure 8).

This, of course, is how McCloskey and Cohen tested how well the ANN remembered the associations in list 1 and list 2. By presenting an “A” word for a list to the input layer and letting it pass through the network to the output layer, the ANN produced what it “thinks” is the appropriate “B” word. Yet, to get the ANN to store the correct association, McCloskey and Cohen (1989) adjusted the connection weights by training the network using the learning rule.

![Network Flow](image)

**Figure 8.** Backpropagation model topology.

Training is a two-step process in backpropagation and this process repeats for each presented input pattern. First, the presented pattern is assigned to the input layer and it feeds-forward through the network to the output layer as a result. Next, the output layer result is compared against the target value. If the output layer attains the desired result, then the weights to the output layer are
unchanged. If, however, the output layer result differs from the target value, then
the weights to the output layer are changed slightly, so as to reduce the
differences between the actual value attained and the target value. Of course,
since the basis for the “error” originates from those weights upstream, these
weights also must be adjusted in the same way until the weight changes
backpropagate to the input layer. This process is called the backpropagation of
error and it uses a form of the delta rule called the generalized delta learning rule
to calculate the difference between the actual output and the target value
(Rumelhart 1989).

Because backpropagation uses a teaching algorithm, Grossberg (1987, p.
48) believes that backpropagation is clearly not a plausible model for human
cognition, because people learn easily without an external teacher. However, it
certainly resembles the brain more closely than conventional computers.
Backpropagation demonstrates the ability of a neural network to store
associations and to generalize. Backpropagation is just one approach though.
Other ANN models work with different assumptions. We will next look at another
type of connectionist network, the Hopfield model, and see how it handles the
catastrophic forgetting problem.

**Hopfield Model**

Hopfield (1982) devised a simple model of associative memory. It is a
simple ANN, capable of storing memories or patterns in a manner similar to the
brain. If we have partial information about a memory we want to retrieve, our
brains usually can find the memory we want. Likewise, the Hopfield network has
the same ability, in that a full pattern is retrievable even if the network has only partial information about the memory. Moreover, the Hopfield network has a degree of stability. If just a few connections between the neurons are severed, the recall ability of the memory is not too badly corrupted and the network can respond with a “best guess.” The brain, of course, has a similar phenomenon that during an average lifetime many neurons die, yet there is no catastrophic loss of individual memories. For this reason, the Hopfield network is an intriguing alternative to backpropagation as a model for cognitive memory, although it also suffers from similar problems with catastrophic forgetting as you see later.

The model. The Hopfield network model is an iterative autoassociative (or content addressable) memory. Patterns are recalled from storage in a Hopfield network by presenting an input pattern, letting the network iterate through the neurons and changing them until the neurons reach a stable state – returning the stored pattern. During this retrieval process, a presented pattern moves through the total solution space towards the stored pattern, better known as an attractor. A Hopfield network can store many attractors in its total solution space with the total solution space divided into basins of attraction for each attractor ξ (see Figure 9). As patterns (or partial patterns) are presented to the network for recall, the dynamics of the network carry the input pattern into one of the attractors as shown by the trajectories sketched. This requires the network to “feedback cycle” a few times to find the correct attractor and settle the network into a stable state.
Figure 9. Schematic configuration space of a Hopfield model with three attractors. This picture is very idealized and in particular the space should really be a discrete set of points (on a hypercube), not a continuous region. Nevertheless, it is a very useful image to keep in mind (Hertz et al., 1991, p. 13).

The neurons in the discrete Hopfield network are a vast simplification of real neurons. The discrete Hopfield neurons can only exist in one of two possible “states” – active (1) or not active (–1). Every neuron connects both input and output connections to every other neuron with some degree of strength (see Figure 10 and Figure 11). The number of neurons needed for the network is the number of bits in the patterns you are storing. Hopfield networks simulate biological neuronal operation in that they have varying propagation delays, varying firing times, etc. Thus, at any instant of time, a Hopfield neuron changes its state depending on the inputs it receives from the other neurons (Hopfield, 1982; Hertz et al., 1991, pp. 13-25; Hinton, 1989, pp. 191-193; Perry, 1998).
Figure 10. This figure shows the Hopfield model. Notice there are no feedback connections to each neuron. Eliminating these connections explicitly forces the weight matrix to have zeros on the diagonal. I have also added external input signals, $l_i$, to each neuron (Freeman and Skapura, 1991, p. 143).

\[
W = \begin{bmatrix}
0 & 1 & -2 \\
1 & 0 & 1 \\
-2 & 1 & 0 \\
\end{bmatrix}
\]

Figure 11. A three neuron Hopfield network (Perry, 1998) and its associated connection weight matrix.
In practice, the neuron update order is in a semi-random order to simulate the parallel processing nature of biological neural operation. All neurons update in one step, but within that step, neurons update in a predetermined randomized order to avoid a bad pseudo-random generator. An example of such a neuron update ordering within a step might be 3, 1, 5, 2, 4.

An example of retrieving a memory from a Hopfield network. Up to this point, I have described how the Hopfield network model is a simple model of associative memory, capable of storing memories or patterns in a manner similar to the brain. And to recall a stored memory, we only need to provide partial information to retrieve it. Perry’s (1998) five-neuron example can serve as a demonstration of this memory retrieval process. Since the example uses only five neurons, a 5x5-weight matrix is set up and trained (see Figure 12), storing the patterns [0 1 1 0 1] and [1 0 1 0 1]. (I discuss how to train the Hopfield network later in the section.) To cue the network to recall an appropriate attractor, a five-bit input value of [1 1 1 1 1] is assigned to the five neurons. (To help visualize this assignment, each bit is depicted in Figure 10 as \(I_1\) through \(I_5\).) To find the attractor, the Hopfield network iterates through each neuron, updating each of the five neurons, applying the following algorithm until all neurons stop changing.

\[ V_{in} = \sum_{j \neq i} w_{ij} V_j \]

\[ v_i = 1 \text{ if } v_{in} \geq 0 \]

\[ \text{else } v_i = 0 \]

where \(w_{ij}\) is an element of the connection weight matrix and \(v_i\) represents a neuron.
Figure 12. Connection weight matrix for the Parry (1998) example after training the network on the patterns [0 1 1 0 1] and [1 0 1 0 1]. The example calculates the dot product of the column out of the connection weight matrix representing the neuron we are updating and the current state to calculate the new weight. For instance, if we are updating the third neuron, then we use the third column of the weight matrix for the dot product calculation.

Before I start the example, notice that the example steps through updating each neuron one at a time using a pseudo-random update order of 3, 1, 5, 2, 4. Therefore, I begin by updating neuron 3. (For clarity, I bold the neuron that is being updated.)

1. Update neuron 3: \( V_{3in} = w_{3j}v_j = (0 \ 0 \ 0 \ -2 \ 2) \cdot (1 \ 1 \ 1 \ 1 \ 1) = 0 + 0 + 0 - 2 + 2 = 0 \)
   Since 0 \( \geq \) 0, \( V_3 = 1 \). Neuron \( V_3 \) does not change. Let us update the next neuron.

2. Update neuron 1: \( V_{1in} = w_{1j}v_j = (0 \ -2 \ 0 \ 0 \ 0) \cdot (1 \ 1 \ 1 \ 1 \ 1) = -2 \)
   Since -2 < 0, \( V_1 = 0 \). Neuron \( V_1 \) changes. The neuron pattern becomes \[0 \ 1 \ 1 \ 1 \ 1\].

3. Update neuron 5: \( V_{5in} = w_{5j}v_j = (0 \ 0 \ 2 \ -2 \ 0) \cdot (0 \ 1 \ 1 \ 1 \ 1) = 0 \)
   Since 0 \( \geq \) 0, \( V_5 = 1 \). Neuron \( V_5 \) does not change.

\[
W = \begin{bmatrix}
0 & -2 & 0 & 0 & 0 \\
-2 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -2 & -2 \\
0 & 0 & -2 & 0 & -2 \\
-0 & 0 & 2 & -2 & 0
\end{bmatrix}
\]
4. Update neuron 2: $V_2^{in} = w_2^j v_j = (-2 0 0 0 0) \cdot (0 1 1 1 1) = 0$

Since $0 \geq 0$, $V_2 = 1$. Neuron $V_2$ does not change.

5. Update neuron 4: $V_4^{in} = w_4^j v_j = (0 0 -2 0 -2) \cdot (0 1 1 1 1) = -4$

Since $-4 < 0$, $V_4 = 0$. Neuron $V_4$ changes. The neuron pattern becomes $[0 1 1 0 1]$.

6. Update neuron 3: $V_3^{in} = w_3^j v_j = (0 0 0 -2 2) \cdot (0 1 1 0 1) = 2$

Since $2 \geq 0$, $V_3 = 1$. Neuron $V_3$ does not change.

7. Update neuron 1: $V_1^{in} = w_1^j v_j = (0 -2 0 0 0) \cdot (0 1 1 0 1) = -2$

Since $-2 < 0$, $V_1 = 0$. Neuron $V_1$ does not change.

8. Update neuron 5: $V_5^{in} = w_5^j v_j = (0 0 2 -2 0) \cdot (0 1 1 0 1) = 2$

Since $2 \geq 0$, $V_5 = 1$. Neuron $V_5$ does not change.

9. Update neuron 2: $V_2^{in} = w_2^j v_j = (-2 0 0 0 0) \cdot (0 1 1 0 1) = 0$

Since $0 \geq 0$, $V_2 = 1$. Neuron $V_2$ does not change.

10. Update neuron 4: $V_4^{in} = w_4^j v_j = (0 0 -2 0 -2) \cdot (0 1 1 0 1) = -4$

Since $-4 < 0$, $V_4 = 0$. Neuron $V_4$ does not change.

Now that all of the neurons stopped changing, reaching a stable state, the process stops. The output of the network is the pattern $[0 1 1 0 1]$, one of the training patterns the network learned. This demonstrates the amazing power of the Hopfield network. But, how do you train one?

**Training a Hopfield network.** Training of the connection weights is not really training as we learned for backpropagation. It is simply a calculation for the connection weight matrix for each attractor pattern that we want the network to
learn. This method described by Hopfield (1982) assigns the connection weight matrix with the following equation:

$$w_{ij} = (2v_i - 1) (2v_j - 1), \ i \neq j \text{ and } w_{ij} = 0, \ i=j; \text{ with } V \text{ as the input vector and } W \text{ is a symmetric matrix of size } n. \ n \text{ is also the number of neurons in the network.}$$

To store multiple attractors, simply sum the above equation for each of the attractor patterns. The following equation summarizes it.

$$w_{ij} = \sum_{s=1}^{n} (2v^s_i - 1) (2v^s_j - 1), \ i \neq j \text{ and } w_{ij} = 0, \ i=j; \text{ with } s \text{ indicating the input vector.}$$

Unfortunately, the Hopfield model has its drawbacks. Using the described learning rule, the Hopfield model can only store at best roughly $0.138n$ arbitrary orthogonal patterns (where $n$ is the number of neurons in the network) (Hertz et al., 1991, pp. 17-20). Robins and McCallum (1998) used a simple variant of the delta rule called the ‘thermal’ perceptron learning rule (Frean, 1992) to train a Hopfield network and increase the storage capacity to $2n$. In either case, this small capacity does not come remotely close to the capacity of the human brain.

Moreover, Robins and McCallum (1998) showed that the Hopfield network suffers from catastrophic forgetting much like backpropagation in that input patterns to be learned are not mutually orthogonal. They trained a network on a base population of 44 items. After training, the network learned an additional 20 items. Between learning of each new item, the base population error was tested. The test consisted of presenting each base population item to the network and letting it iterate until it reached a stable state. If the input item did not change, they considered the item stable. If, however, the network changed the input item,
they considered the item unstable. Figure 13 demonstrates how interference to
the base items increased as the number of new items learned by the network
grew.

![Graph showing catastrophic forgetting in a Hopfield network model.](image)

Figure 13. Catastrophic forgetting in a Hopfield network model (Robins and

**Semi-distributed Network Models**

As mentioned earlier, French (1992) advocated the notion of semi-
distributed ANN architectures as the means for solving the catastrophic forgetting
problem. His rationale was that ANN models suffer from the ‘stability/plasticity
dilemma’ (see, for example Grossberg, 1987, which incidentally also describes
an effective solution to the catastrophic forgetting problem with his competitive
learning model, ART, but is outside the scope of this paper). Ideally, the
representations in an ANN need to be pliable enough to learn new things and
adapt to a changing environment. It also needs to be stable enough to retain
important information over time. “The dilemma is that while both are desirable
properties, the requirements of stability and plasticity are in conflict. Stability depends on preserving the structure of representations, plasticity depends on altering it. An appropriate balance is difficult to achieve” (Robins, 1995, p. 124).

ANNs and look-up tables lie at opposite ends of the stability/plasticity spectrum. ANNs are fully distributed representations with all knowledge stored in the connection weights. The advantage ANNs have, as we already discovered, is their ability to generalize; the disadvantage is excessive plasticity or catastrophic forgetting of newly learned items. Look-up tables, on the other hand, separate knowledge into compartments preventing new information from disrupting old information. Unfortunately, a system of look-up tables cannot generalize about unfamiliar information like ANNs can. The middle ground is a system that spreads information around to prevent new knowledge from disrupting old knowledge, yet the system can still generalize. This is the class of ANN models called semi-distributed.

To give you the flavor of the semi-distributed class of ANNs, I will briefly describe SDM, a model that falls into this class. I will not get into the details for SDM as I did for the backpropagation and Hopfield models. I only want to illustrate how a neural system can generalize even while its representations are not distributed over the entire memory. With their representations semi-distributed, they exhibit limited representational overlap (assuming they are not saturated), experiencing little catastrophic forgetting (French, 1992).

**Sparse Distributed Memory (SDM).** SDM (Kanerva, 1993; Keeler, 1988) is a math model of long-term memory; a memory management approach for
handling storage and retrieval of information with a very large storage capacity (as large as \(2^{1000}\) addresses). It can function as an autoassociative memory (like Hopfield), a heteroassociative memory (like backpropagation), or a sequential-access memory (like a conventional computer).

Since it is easier to describe SDM as a parallel to conventional computer memory, I will present a description first of a computer’s random-access memory (RAM) then describe SDM. In RAM, a number or address identifies each storage location. This address specifies the position of the location in the array of memory and the information is stored at that location. Reading data from memory involves specifying the address and the contents at that address are returned as data. Similarly, writing data to memory involves specifying the address and replacing the data at that location with the new data. The address finding mechanism is typically sequential (or some variation of sequential) and since the hardware is very fast, finding an address is quick and straightforward.

SDM works similarly to RAM, except it is more complex. Since SDM deals with a very large address space, it, therefore, is impossible to build a hardware location mechanism for finding each of the addresses in such a very large address space (Kanerva, 1993). Kanerva, overcame this issue by using statistical probability. If data was distributed randomly and uniformly, then a set of \(M\) hardware addresses can be picked at random, across the address space, to find the statistical probable value of the contents of the address for the desired data. In other words, a much smaller set of hardware addresses is selected randomly throughout the possible set of all addresses. To write data to a specific
address, the data simply is written to all addresses that fall within a Hamming distance D of the specified address. The Hamming distance is analogous to drawing a circle with radius D from a point. In our case, every storage location within the Hamming radius of the specified address is written with the data we are storing. This is why SDM is a semi-distributed model. Data is “distributed” all over the storage locations within the Hamming radius, yet it is sparsely stored relative to other hardware addresses (Keeler, 1988).

To read from SDM, all the storage locations within the Hamming distance D of the specified address are processed. Processing the data from these selected storage locations consists of summing each \(i^{th}\) bit together to yield \(i\) sums. These sums are thresholded at zero, so sums less than or equal to 0 are set to 0 and sums greater than 0 are set to 1 to produce the retrieved value for the specified address. Thus, if we had the bit sums of [-445, 379, -201, … , 77, 611, -517] would yield the value of [0, 1, 0, … , 1, 1, 0] as the value at the location of the address we specified.

Now, retrieval from SDM is a statistical reconstruction of the original value stored at that memory location. It should be the same value as originally stored at that memory location as long as not too many other values are stored in the area. This saturation, of course, would cause considerable representational overlap and new information would interfere with old information. Saturation of memory, unfortunately, causes not just forgetting of old information but an inability to store the new information as well (French, 1992).
This section, as a whole, tried to uncover the mystery as to how ANNs work by describing two of the most widely used ANN models, backpropagation and Hopfield. I also indicated that backpropagation is not the only ANN model to suffer from catastrophic forgetting, as the Hopfield model suffers from it as well. I finally presented the class of semi-distributed models and described one member of that class, SDM, as an example to show how it does not suffer from catastrophic forgetting. By this point, the reader should have a flavor for ANN modeling and be ready to explore the many attempts at solving the catastrophic forgetting problem.

Attempts at Solving Catastrophic Forgetting

In the early part of this paper, I explained the catastrophic forgetting problem and gave some ideas researchers have as to what causes it. I also presented a few of the ANN architectures available, specifically focusing on the two most popular models – backpropagation and Hopfield. The vast majority of the literature has focused on solving the catastrophic forgetting problem for backpropagation since these two models rely on a target for learning and their knowledge is fully distributed throughout the network.

I discuss in this section the proposed methods for solving the catastrophic forgetting problem in the backpropagation model. These fall into four general categories based on reducing overlapping distributed representations, rehearsal, pseudorehearsal and dual-network systems.
Reducing Overlapping Distributed Representations

As discussed earlier, French (1992) suggests that catastrophic forgetting occur as a result of the overlap of distributed representations. He thought that simply reducing this overlap reduces the problem. Several studies explored this approach all with limited success. I will summarize them briefly.

The novelty rule (Kortge, 1990) tried to prevent the learning of a new item from moving the weight configuration away from the overall solution space. By adjusting only those connection weights that need adjusting, the amount of overlap between the input representations of the new item and the previously learned items would be reduced. As we learned, this had a moderate effect on catastrophic forgetting.

French (1992) experimented with activation sharpening. Unlike the novelty rule, which attempted to reduce overlap of the input values, activation sharpening attempted to reduce overlap in the hidden layers. Neurons with the most activation increase their activation level slightly, while other surrounding neurons’ activation levels decreases slightly. This again achieved limited success.

Context biasing (French, 1994) attempted to produce hidden layer representations that had increased separation (orthogonality) and “sparser” hidden neuron representations with a smaller number of active neurons. French’s technique requires the network to remember both the previous target pattern and the previous hidden layer representation as a new pattern trains on the network. Context biasing reduced catastrophic forgetting by 50% compared to standard backpropagation, a mild improvement.
Murre (1992) tried a method providing “reminders” to the network of items earlier learned. These reminders enabled the network to keep the hidden layer representations orthogonal by presenting new items in “blocks” with randomly selected previously learned items for 10 training cycles. Construction of these blocks were such that items did not overlap and that presenting two blocks of one type corresponded to presenting each element of the block twice, in random order. Murre’s results were limited, on par with the previously mentioned studies.

McRae and Hetherington (1993) demonstrated that pre-training a network on a set of patterns did indeed reduce catastrophic forgetting. Their rationale was that subjects in memory experiments began with a large body of knowledge about words and their properties. Learning of the first item (or list of items) in a sequential learning task must be learned within the constraints resulting from prior knowledge. Thus, “many of the hidden layer [neurons] are already committed to representing a limited aspect of the input space because of strong weights from a limited number of units. When a second item (or short list) is trained, interference with the first is decreased because the probability of overlap at the hidden layer is reduced” (McRae and Hetherington, 1993, p. 724). This is an intriguing assumption, but it does not address the ANN’s early learning stage.

These methods, in general, do not prevent catastrophic forgetting from occurring. Although, catastrophic forgetting did improve such that retraining of the prior learned items was always quicker than training a network from the beginning (Robins and McCallum, 1999, p. 1193).
Rehearsal and Pseudorehearsal Network Training

Ratcliff (1990) not only replicated the work by McCloskey and Cohen (1989), he also went onto to see if he could tweak the backpropagation model in any way to produce any reduction in catastrophic forgetting. Ratcliff, in one instance, tried a training procedure based on a ‘simplified model’ of human rehearsal in list learning (Ratcliff, 1990, p. 293). His training procedure went like this. The network trained on a list of items in the usual fashion. Afterwards, training on a new item began. Unlike previous training, training for the new item included three items from the previous list, forming a buffer of four items taking turns presenting themselves to the network. When the network learned all items to criterion then another new item began training with another group of three previously learned items forming their own buffer. This procedure continued until the network learned all new items. The recency rehearsal training, as Ratcliff called it, had a very modest impact on the catastrophic forgetting effect. Previously learned list retention still dropped off to the same degree as seen previously. However, the previously learned list retention for the recency rehearsal training method improved significantly compared to the no rehearsal training method. See Figure 18 later in this section.

Sweep rehearsal training. Robins (1995) looked at Ratcliff’s results as a possibility that could be useful in engineering and industrial applications. Since his discipline is computer science, Robins was not trying to create a cognitive memory model or explore an alternative that directly relates to human performance. His goal, on the other hand, was simply to find an effective
practical procedure. Robins examined several possible alternative procedures and came up with one he called sweep rehearsal. Sweep rehearsal uses a dynamic training buffer as opposed to the fixed buffer used in recency rehearsal. The dynamic buffer randomly chooses three previously learned items for the buffer each time the new item takes its turn training the network. The training for the new item progresses until the new item trains to criterion, after which another new item begins training in the buffer. The advantage sweep rehearsal has over other methods examined by Robins is that it exposes more previously learned items to one or more training cycles, but does not actually retrain any previously learned items to criterion (Robins, 1995). The procedure yielded excellent results (see Figure 18, later in this section) with the previously learned items maintaining a very high level of accuracy. The sweep rehearsal procedure was so effective that as intervening trials increased, the retention of the previously learned items slightly improved!

**Backpropagation models as function approximators.** Robins’ sweep rehearsal results were stunning. The practicality of retaining the previously learned items, unfortunately, makes this procedure of limited interest. Robins, therefore, needed to refine his method so that training new items with the help of the previously learned items was not necessary. This meant that he had to rethink the cause of catastrophic forgetting from the explanation provided by McCloskey and Cohen (1989) and French (1992). Instead of viewing the catastrophic forgetting problem in complex mathematical terms as done previously, Robins viewed it in simple algebraic terms by conceptualizing the
backpropagation network as a function approximator. He abstracted the behavior of the backpropagation network to a ‘toy’ two-dimensional graph with the x-axis representing the possible inputs to the network and the y-axis representing the outputs of the network as \( F(x) \) of the inputs. For a given training population of actual inputs to the network and the actual outputs that they generate, Robins viewed the process of learning as a process of fitting the training population to some function as shown in Figure 14.

![Graph showing possible inputs and outputs](image)

**Figure 14.** Interpreting learning in a neural net as function approximation.

Learning fits a function to the training population items, with a range of functions are possible, including ‘compact’ and noisy functions (Robins, 1995, p. 136).

Continuing with Robins’ analogy, teaching the network with a new set of items after the network had already learned a base set of items yields Figure 15. “This is the cause of catastrophic forgetting, the base population inputs [previous learned items] will no longer generate the correct outputs using the new function” (Robins, 1995, p. 137).
Figure 15. New items are learned without rehearsal, the new learned function may not be similar to the old function describing the original population (Robins, 1995, p. 136).

By comparison, the recency rehearsal and the sweep rehearsal procedures include the previously learned items in the training process as new items are learned. This constrains the new function learned by the network to fit both old and new learned items (see Figure 16).

Figure 16. New items are learned with recency rehearsal and sweep rehearsal, the new learned function preserves much of the shape of the old function (Robins, 1995, p. 136).
Sweep pseudorehearsal training. To continue constraining the function to fit both old and new items without retraining on the old items, Robins simply constructed ‘pseudoitems’ to serve the same purpose as the previously learned items. By randomly generating input items and presenting them to the network, the network produced output items that fit the old function. These pseudoitems combined with the new items using the sweep rehearsal procedure to train the network. This again constrains the new function learned by the network to fit both the pseudoitems and the new items (see Figure 17). Robins called this training procedure, sweep pseudorehearsal.

Figure 17. New items are learned with pseudorehearsal using pseudoitems. The new learned function preserves much of the shape of the old function (Robins, 1995, p. 136).

The sweep pseudorehearsal procedure proved to be reasonably effective at allowing the network to re-approximate a function with the new items. Figure 18 shows the performance of sweep pseudorehearsal training compared to the other rehearsal and no rehearsal training procedures. Robins felt he had potential solutions for the catastrophic forgetting problem with the rehearsal and pseudorehearsal training procedures allowing new information to be integrated
into existing practical ANN applications. For some, having an omni-gnostic teacher know when an input pattern is new or learned, was troubling as a cognitive model. Nonetheless, Robins was not concerned with issues of ‘psychological plausibility’ (even though pseudorehearsal can treat each type of pattern the same), he may have provided the missing “key” to creating a viable ANN cognitive model.

Figure 18. Summary of the results presented for base population goodness over 10 trials. Conditions shown are sweep rehearsal, sweep pseudorehearsal, recency rehearsal, and no rehearsal (Robins, 1995, p. 133). “The ‘Goodness’ measure is the network’s ability to correctly reproduce the approximate outputs for a given set of inputs. It is calculated by averaging for each input-output pair the normalized dot product of the target vector and the actual output vector observed (see Ratcliff, 1990, p. 288). Vectors are transformed so that a goodness of 1 indicates a perfect match and a value of 0 indicates chance performance (50% match)” (Robins, 1995, p. 141).
Complementary Learning Systems – A Dual-Network System

The approach developed by Robins (1995) has wonderful potential. However, how could his pseudoitem notion be “neurally” implemented in the framework of a biological connectionist architecture. McClelland, McNaughton, and O’Reilly (1995) argued that radical separation of representations might have been the approach nature arrived at in the development of the hippocampus and the neocortex. McClelland et al. believe that the hippocampus became a place for the initial storage of memories in a form that avoids interference with the knowledge already acquired in the neocortical system. The hippocampus rapidly acquires knowledge and gradually trains the neocortex on this new knowledge, allowing the neocortex to integrate the new knowledge with the old.

Examples supporting a complementary learning system in the brain. McClelland et al. presented many examples from the literature to support their claim. Here I briefly summarize some of the main points they cite to provide a flavor of the research they discuss.

1. Lesions to the hippocampal system produce profound deficits in new learning where as knowledge acquired well before the lesion is unaffected. This condition was reported for a patient who had a large bilateral portion of his hippocampus and other temporal lobe removed. The patient had profound deficits in memory for events that occurred either after the lesion or during the months and weeks before it. This supports the idea that the hippocampus initially stores memories before passing those memories to the neocortex for long-term storage.
2. Lesions to the hippocampal system effect selective forms of learning. The hippocampus system appears to be essential for rapid formation of comprehensive associations. One such example of the types of associations that the hippocampus handles is memory for paired-associates like those studied by Barnes and Underwood (1959). When lesions are found on the hippocampal system, the person has difficulty learning arbitrary paired associates. This also supports the idea that the hippocampus quickly stores initial memories.

3. Studies of rats with hippocampal lesions show that the hippocampus is essential to the formation of memories involving places or locations in the environment. This supports the increasing evidence that the hippocampus is required for tasks that depend on learning to navigate in a previously unfamiliar spatial environment. This further reinforces the role of the hippocampus.

4. After patients have a hippocampal lesion, performance on recent material can actually be worse than performance on somewhat older material. This suggests “… that some real consolidation takes place because it rules out the alternative interpretation that memories are initially stored in two forms whose effects are additive: a relatively transient, hippocampal-system-dependent form and a more persistent, hippocampal-system-independent form” (McClelland et al., 1995, pp. 421-422). Furthermore, McClelland et al. cite three additional studies that reinforce the dual-store system hypothesis.

5. Looking at the brain’s anatomy, the perirhinal and parahippocampal cortices are considered part of the neocortex. They, however, appear functionally to belong at least in part to the hippocampal memory system. McClelland et al.
consider these structures as borderline areas in which the neocortical processing system and the hippocampal memory systems overlap. Moreover, McClelland et al. say that bi-directional pathways interconnect the hippocampal and neocortical systems such that patterns can be passed between the two systems. The researchers do not believe that direct copies of patterns pass between the systems. Instead, they believe that representations in one system are re-represented in a compressed format over a smaller number of neurons in the other system. If there is redundancy in the neocortical representations, then such compression can occur without loss of essential information. McClelland et al. further their point by saying that ANN data compression schemes already exploit such redundancy to achieve high compression rates. These ANN compression models compress patterns as a pattern enters the system and decompress the pattern as it exits.

Although I only touch on a few of their points, in general, McClelland et al. make a strong case that the hippocampus and the neocortex are complementary learning systems. And that the hippocampus receives new knowledge and acts as teacher for the new knowledge so the neocortex can integrate it with prior knowledge.

A Dual-Network ANN architecture. Armed with insight of the roles that the hippocampus and the neocortex play inside the brain from McClelland et al., French (1997) and Ans and Rousset (1997) independently set out to create a connectionist model of these two brain structures cooperating with each other. They assumed that if McClelland et al. were right that evolution solved the
catastrophic forgetting problem with two complementary learning systems, then a
dual-network ANN model should learn serially without disrupting prior knowledge.

The ANN models developed by French and Ans and Rousset sought to
separate the previously learned representations from those the network was
currently learning. To do this, they had to develop a method to extract
approximations of previously learned patterns from one network to mix with new
pseudorehearsal training procedure, these researchers created a dual ANN
architecture that could pass a network’s memory to another network. I will first
describe French’s architecture and show its effectiveness in serial learning.

French’s dual-network architecture consists of two backpropagation
networks. The first network, called NET 1, works like the hippocampus, collecting
new knowledge and teaching this new knowledge to NET 2, the second network.
NET 2, acting similar to the neocortex, stores all final or long-term knowledge.
When the dual-network system starts to learn a new item pair, consisting of an
input pattern and a target pattern, NET 2 first generates a set of pseudoitem
pairs. Net 2 creates these pseudoitem pairs by randomly generating input
patterns and feeding them into NET 2 to produce associated target patterns. NET
1 then learns the set of item pairs consisting of the new item pair and the
pseudoitem pairs following the pseudorehearsal training procedure (see stage (I)
in Figure 19). Training continues until each individual item (or pseudoitem) pair
learns to criterion. Once NET 1 finishes its training, NET 1 generates a number of
pseudoitem pairs that are then learned by NET 2 for final storage of the
combined knowledge, again following the pseudorehearsal training procedure (see stage (II) in Figure 19). So every time a new item pair is to be learned, this two-stage process begins, allowing for the integration of new items with the old. Figure 20 shows the 29% improvement in recognition of the previously learned item pairs for the dual-network architecture compared to standard backpropagation.

![Diagram](https://via.placeholder.com/150)

**Figure 19.** French’s (1997) dual-network architecture. Stage (I): the NET 1 network is learning new items along with pseudoitems generated by NET 2. Stage (II): the NET 2 network is learning pseudoitems generated by NET 1 (transport of NET 1 memory towards NET 2).
Reverberating dual-networks. Ans and Rousset (1997) also used the pseudorehearsal training procedure as French did to pass memory between two networks. Ans and Rousset, however, implemented an innovation of their own into their dual-network architecture. They thought “...that only a single pass of each random input pattern through a feedforward network is largely insufficient to correctly extract the information structure from its connection weights” (Ans and Rousset, 1997, p. 994). Ans and Rousset thought that by combining the autoassociative power of the Hopfield network with backpropagation, a new network architecture could optimally capture the deep structure of previously learned knowledge for transfer to another network. Their new “hybrid” architecture used a single hidden unit layer and allowed the hidden unit neurons to link back to the input layer neurons. So, when the network generates a pseudoitem pair by passing a randomized input pattern through the network to
generate an associated target pattern, the network also ‘reverberates’ (iterates) between the input layer and the hidden layer a number of times to converge close to a network attractor. This method allows for a better approximation of a valid input pattern while simultaneously generating a corresponding target pattern as pseudoitem pairs for training another network. In other words, when one network is learning from another network, the difference between the pseudo-target pattern and the output pattern generated by the pseudo-input pattern is minimized – increasing the accuracy of transferring the network’s memory to the second network.

Because this dual-network system consisted of two fundamentally different network architectures than backpropagation, the serial learning process was also adapted accordingly. To illustrate this new process, assume that NET 1 already learned a base set of items and that NET 2 is still ‘empty’ (i.e., with random connection weights). The neural system enters the first processing stage (Stage (I) in Figure 21), where NET 1 cannot learn new item pairs. Rather it is continuously receiving random generated input (or noise) into its input layer. NET 1 generates a pseudo-target pattern as usual and ‘reverberates’ (iterates) between the input layer and the hidden layer a set number of times to return a pseudo-input pattern that is close to one of the network’s attractors. These pseudoitem pairs (the pseudo-output pattern and the newly generated pseudo-input pattern) are constantly presented to NET 2 to learn. When it is time for the network to learn a new item pair, Stage (II) (see Figure 21) starts. Here, NET 1 stops receiving noise as input; NET 2, using the same pseudoitem pair
generation process as NET 1 used in Stage (I), starts receiving noise to generate pseudoitem pairs for transferring its memory back to NET 1 for pseudorehearsal training with the new item pair. During pseudorehearsal training, the reverberating mechanism turns off, allowing the network to train as a standard backpropagation network would normally. After the system learns the new item pair, the learning process reverts to Stage (I) until the system starts learning another new item pair, at which Stage (II) begins again.

Figure 21. The reverberating architecture. Stage (I): the NET 2 network is learning pseudoitems generated by the reverberating process in NET 1 (transport of NET1 memory towards NET 2). Stage (II): the NET 1 network is learning external items along with pseudoitems generated by the reverberating process in NET 2 (Ans and Rousset, 2000, p. 4).
Results demonstrate that, first, the reverberating dual-network avoid the catastrophic forgetting found in standard backpropagation networks. The network also showed good retention of NET 1’s memory as it was transferred to NET 2. Using $R = 10$ iterations and $N = 4$ pseudoitem pairs, the upper graph in Figure 22 shows that NET 2 reached exactly the same mean goodness value of 0.948 as NET 1 when testing the retention of the base items that the system learned prior to beginning Stage (I). This “…means that information previously learned in the first network can be very well captured and transferred to the other [network]” (Ans and Rousset, 1997, p. 995). The lower graph in Figure 22 shows catastrophic forgetting with $R = 0$ iterations and $N = 0$ pseudoitem pairs. The middle graphs in Figure 22 demonstrate the effect of changing the $R$ and $N$ parameters. Without reverberation ($R = 0, N = 1$), retroactive interference is still relatively high. Learning new item pairs, in addition, proves difficult since the final number of cycles needed to reach criterion is high. Comparing graphs ($R = 10, N = 1$) and ($R = 0, N = 1$) in Figure 22 examines the effect of reverberation also. Learning of new item pairs is much faster. “This comparison demonstrates clearly the crucial role of neuron-like processes, reverberating from random stimulations, in discovering the deep structure of information distributively represented within network connectivity” (Ans and Rousset, 1997, p. 996).
Figure 22. Mean goodness of the old base as a function of the number of training cycles of the new base. Graphs (plotted for each set of ten cycles) stop when the new base learning is completed and the full square (the \( I \) on the y-axis) refers to the initial goodness of the old base before the new base training starts (Ans and Rousset, 1997, p. 996).

**Upper plot:** goodness variation with \( R = 10 \) iterations and \( N = 4 \) pseudoitem pairs per one new item pair trained in NET 1. Note that retroactive interference is dramatically reduced since the final mean goodness of the old base is close to its initial level.

**Middle plots:** note the important enhancement of the old base goodness between simulations performed without \( (R = 0) \) and with \( (R = 10) \) reverberating networks, for the same \( N \) parameter value \( (N = 1) \).

**Lower plot:** catastrophic forgetting with no reverberating or pseudorehearsal precesses.
Thus, not only did the dual-network architectures of French and Ans and Rousset have the ability to copy a network’s memory to another network, but catastrophic forgetting was also reduced. This suggests that McClelland et al. may be right that evolution solved the catastrophic forgetting problem in the brain by implementing complementary learning systems, the hippocampus and the neocortex.

Conclusions

We started this paper by examining the effects of serial learning on human populations and on ANN computer simulations. We quickly found that the backpropagation ANN model used by McCloskey and Cohen (1989) did not simulate the learning process of the brain very well. And that learning a second set of information by an ANN caused the network to ‘catastrophically forget’ the previously learned information. We learned that because the knowledge is highly distributed throughout the network, that when a second item is learned by the network, the system creates a new solution space based on the new information learned (McCloskey and Cohen, 1989). This led many to believe that the catastrophic forgetting problem was due to overlapping representations in the network. Solving the problem thus meant creating a semi-distributed architecture that spreads out the representations far enough so that new information did not disrupt the old information (French, 1992).

We also learned that ANNs are simple models of neuronal interactions occurring inside the brain and that the organization and learning approaches these various ANN models represent are attempts at what researchers think the
brain is doing. We examined some of the models used in research such as backpropagation, Hopfield, and SDM to give the reader a flavor of how ANN models work. We learned in addition that not all ANN models suffer from catastrophic forgetting, such as the model SDM.

Finally, we explored the attempts made at solving the catastrophic forgetting problem. We found that attempts to reduce overlapping representations in the hidden layer of backpropagation models generally did not prevent catastrophic forgetting from occurring. Although we did learn that pre-training a network on a set of patterns does reduce catastrophic forgetting and presents some interesting possibilities for ANN research as the field further matures (McRae and Hetherington, 1993).

We found that a subtle breakthrough on the catastrophic forgetting problem appeared when Robbins (1995) attacked the problem for practical engineering and computer science purposes rather than as a cognitive model of the brain. Robins found that rehearsing the previously learned items with a new item in a learning procedure he called sweep rehearsal obliterated the catastrophic forgetting problem. Unfortunately, maintaining such a set of items learned by a network loses the practicality that Robins sought. Persevering, Robins developed the sweep pseudorehearsal procedure by viewing a trained network as a kind of function approximator. By passing into the network randomly generated input data, the network could generate targets, forming a pseudoitem comprising the input and the target. Using these pseudoitems in place of the
originally learned data, Robins successfully taught a network to learn sequentially with a retention rate of 70%, a level comparable to human sequential learning.

From a cognitive modeling prospective, Robins’ pseudorehearsal approach lacked plausibility. French (1997) and Ans and Rousset (1997) each built dual-network architectures based on the evidence gathered by McClelland, McNaughton, and O’Reilly (1995) that the brain solved the catastrophic forgetting problem by implementing the hippocampus and neocortex as complementary learning systems. As we learned, these dual-network architectures used the pseudorehearsal technique to pass a network’s knowledge to another network, thus solving the catastrophic forgetting problem for backpropagation networks. French (1992) asked the question, “Is there highly distributed representations that are not affected by catastrophic forgetting.” To answer French, the answer is yes, however, it probably was not in a form he anticipated. Yes, all the knowledge is stored inside a single highly distributed neural net, except that to integrate the old knowledge with the new, two networks were used to separate the old from the new to prevent the new knowledge from disrupting the representations of the old knowledge, as French (1992) advocated.

From here, where does the ANN research go? Nobel Prize winner Gerald Edelman (1992) suggests that the brain is made up of neuronal groups. These neurons engage in a topobiological competition to form groups through neuron movement, neuron process extension and neuron death during the brain’s development process. The entire process is a selectional one where the neurons explore looking for matching connections to form small independent networks. As
the animal engages in some form of behavior, additional synaptic connections develop by biochemical process to selectively strengthen and weaken the connections inside the groups of networks. Eventually, the groups of neurons begin to connect to other groups with re-entrant signaling capability. Local maps form between the re-entrant groups, allowing for the development of psychological processes and perceptual categorization. Eventually, global maps form by linking local maps and unmapped brain structures, such as the hippocampus, together. The “… global mapping allows selectional events occurring in its local maps to be connected to the animal’s motor behavior, to new sensory samplings of the world, and to further successive reentry events” (Edelman, 1992, p. 89). Simply stated, these global maps allow the animal to give particular categorical responses.

Edelman’s theory of neuronal group selection (TNGS) is an interesting proposition. Although further details of TNGS are well beyond the scope of this paper, it is safe to say that now that there is a technology to pass network knowledge to other networks, researchers can explore the processes that comprise TNGS with ANN simulations.

To conclude this paper, I hope I have exposed the insides of the “black box” called artificial neural nets and opened other avenues for their use whether for practical purposes or research. Furthermore, I hope I have lifted the bounds that exist in many researcher’s mindsets about using ANNs as a way to model cognition.
References


