

**Exploring a simple model of distributed memory: What neuropsychologists can learn from connectionism.**

John L. Stricker  
Gregory G. Brown

## **Abstract**

It is often suggested that the use of connectionist models would encourage neuropsychologists to think of cognition in terms of dynamic interactions rather than an assembly line of independent components. However, connectionist models, and the use of computational modeling in general has yet to establish itself in the mainstream of cognitive neuropsychology. A number of factors have likely contributed to this lack of acceptance, including: 1) the complexity involved in understanding and building these models, 2) the difficulty of evaluating these models (direct comparisons between connectionist models are rare in the literature), and 3) the specificity of these models (e.g. generally, different models are created for different problems rather than having a general model which solves multiple problems). In this paper, we demonstrate how a simple connectionist model can be used as a tool for developing conceptualizations of how a distributed memory system might operate and illustrate how conceptual assumptions and the specifics of the model can produce misleading results. By focusing on making the model learn multiple problems as efficiently as possible, we will examine how properties of short term memory, long term memory, and working memory can be understood in a distributed system without requiring separate, modular memory systems. We will focus on how this model reflects and informs recent developments concerning the role of the medial temporal lobe in memory formation.

## **Introduction**

Since their resurgence in the 1980s, researchers have argued that the use of connectionist models could assist neuropsychologists in the development and testing of theories of brain function and dysfunction (Farah, 1994; Seidenberg, 1988). It is often suggested that the use of such models would encourage researchers to think of cognition in terms of dynamic interactions rather than an assembly line of independent components. However, connectionist models, and the use of computational modeling in general has yet to establish itself in the mainstream of neuropsychology, and more modular conceptualizations of neuropsychology remain prominent (Harley, 2004; Ranganath & Blumenfeld, 2005; Rorden & Karnath, 2004). In this paper, we will explore how connectionist simulations can shed light on past and current explanations of learning deficits associated with medial temporal brain dysfunction.

### **Dissociations and the Two-Stage Model of Memory**

Dysfunction in the medial temporal lobe has been strongly associated with anterograde amnesia: an inability to recall newly experienced events. This deficit is classically illustrated by the patient H.M. who was unable to learn new episodic information following surgical resection of his medial temporal lobe (Scoville & Milner, 1957). Similar deficits also exist in Alzheimer's disease, where the medial temporal lobe appears most sensitive to the disease process (Braak & Braak, 1991; Hyman et al., 1986). The presence of semantic dementia, where damage to the temporal cortex causes deficits in well learned knowledge while episodic information remains intact, although rare, has also been well documented, indicating a double dissociation between well learned-information and episodic information (Hodges & Patterson, 2007). These striking

dissociations have prompted researchers to focus intensely on the medial temporal lobe, particularly the hippocampus, and the role it plays in memory.

### The Two Stage Memory Model

From the basic deficits illustrated by anterograde amnesia, it is intuitively appealing to conceptualize memory as at least two separate systems: a short term memory system that learns things quickly but does not maintain that information for long, and a long term memory system that learns things slowly but is able to maintain that information for a long period of time. According to this model, information in the long term memory system is slowly built up over time by interactions with the short term memory system, and permanent memories are formed through a process of consolidation. This two-stage conceptualization continues to be very popular in both the neuropsychological and the computational literature, and the medial temporal lobes are associated with the functions of the short-term memory system (Norman et al., 2005; Squire, 2004).

The two-stage memory model also makes conceptualizing memory in terms of separate memory systems appealing because it resolves a conflict of functions: how can a memory system learn new things without interfering with what is already known? This conflict is illustrated (and thus is further supported) by catastrophic interference in distributed memory models. In a distributed memory model, knowledge thought to be stored in a distributed fashion and is represented by differing strengths of connections (referred to as weights) between simulated neurons (referred to as nodes). When new information is learned, there is nothing protecting the previously learned information from being overwritten, thus producing catastrophic interference: the new problem is learned but the old problem is forgotten (McCloskey & Cohen, 1989). Consequently, it

appears that if memory is stored in a distributed fashion, then it is necessary for there to be a separate short term and long term memory system (McClelland et al., 1995; Norman et al., 2005).

#### Doubts about the two stage model

In light of more recent findings, some researchers have begun to question the idea that common disassociations in the literature prove that a clearly defined modular role for the medial temporal system exists (Gaffan, 2002; Ranganath & Blumenfeld, 2005; Squire et al., 2004). However, if the medial temporal cortex is not exclusively related to, or is incapable of containing complete representations of memory, it remains unclear how a distributed memory system would be able to overcome the conflict of functions that is solved by the two stage model. We now turn to the development of a connectionist model that addresses this conflict.

### **Developing a Connectionist Model that Solves Multiple Problems**

#### A Brief Introduction to Neural Networks

Neural networks have a long and colorful history (Anderson & Rosenfeld, 1988). Many of the neural networks used by researchers today are based upon the Parallel Distributed Processing framework (Rumelhart et al., 1986). The process of training in neural networks consists of providing the network with a set of inputs and expected outputs, allowing the network to feed the activation forward to the output units, and then adjusting the weights of the network connections and each node's bias value (each node other than input nodes have a trainable resting state, or bias level) based upon the amount of error that the network produces. Error in the network is determined by the amount to which the output activation of the network differs from the output expected in the training set.

Overall network error for a given training set is usually calculated by taking the sum of squares of the differences between desired output and the networks actual output, dividing this number by the number of training trials, and taking the square root of that value. This value is usually abbreviated as RMS.

Adjustments are made based upon whether or not the network's output corresponds with the output specified by the training data. The adjustments to be made are determined through a process called back-propagation. Essentially, back-propagation consists of taking the error of the output node (the difference between the output node's activation and the expected response) and working back through the network and determining how much each node is responsible for the amount of error that was produced. Connections and bias values are then adjusted, taking into account the amount of error for which each is responsible. The process of back-propagation makes only small adjustments each time the network is presented with inputs and an expected output. Consequently, the training data set must be presented to the network many times until eventually the network's error for all of the training data is reduced. For an easy to understand description of the mathematical details behind back-propagation, see chapter 1 of (Plunkett & Elman, 1997) and chapter 4 of Smith (M. Smith, 1993). The specific training procedures used in this work are described in Appendix A.

### The problem domain

The simple problem domain of Boolean logic expressions will be used to demonstrate different properties of the distributed memory system. Boolean expressions evaluate at least two arguments that can be true or false and return a value of true or false. These sorts of logic expressions describe a relationship between the presence and/or

absence of different kinds of information, and are immensely useful in every day logic and in programming languages. These expressions are also easily adapted into artificial neural networks (ANNs) because they involve binary inputs, generate a single binary output, and describe a simple rule. Table 1 illustrates how two different Boolean expressions would be coded into a neural network. The inputs represent the information provided to the network, and the output indicates whether or not the Boolean expression is true for each of the inputs provided (a 1 indicates that the relationship is true, and a 0 indicates that the relationship is not true). For example, for the expression OR to be true, at least one of the inputs or both has to be present (with 0 indicating the absence of the input and 1 indicating the presence of the input). As the middle of Table 1 shows, the exclusive-OR (XOR) expression is a little more specific than OR: in order for XOR to be true, one, and only one of the inputs must be present.

#### Multiple Problems in a Feed-Forward Network

Figure 1 illustrates a feed forward network that can solve XOR (as initially described by Rumelhart et al., 1986). In training this network to learn XOR using the training methods described in Appendix A, the network is able to successfully solve XOR about 90% of the time. The network does not consistently solve XOR all the time because its solution to the problem can be influenced by the random values that are used to initialize its weights prior to training. Of those networks that successfully solved XOR, it took on average 498 epochs (each epoch is a training cycle consisting of presentation of an input/output pair and then weight adjustment based on output error), indicating that on average, the entire problem had to be presented to the network 124 times before the problem was solved.

Demonstrating Interference. To demonstrate the process of interference, networks were first trained on OR and then trained on XOR. In this case, about 97% of the networks were able to solve XOR (about a 7% increase over learning XOR alone). The networks that were able to solve XOR took a total 256 epochs on average. Of note, this is about twice as fast as the networks that learned XOR without learning OR first in the previous study. However, all of the networks suffered from catastrophic interference: even though the OR information was able to constructively assist the networks in learning XOR, the full knowledge of OR was interfered with by the XOR knowledge. These results could easily lend themselves to a two-stage model of thinking: somehow previously learned knowledge needs to be protected. Yet this example also makes something else very clear: as the network is learning these problems, *it has absolutely no way to distinguish one problem from the next*. This is starkly unlike the manner in which humans operate, as we are able to separate not only one problem from the next, but one moment in time from the next.

Introducing context. If a cognitive system is going to learn and remember multiple problems, it must have some access to information that separates one problem from another. This can be most simply represented in the current network architecture by adding an additional input node that distinguishes between the two problems (e.g. -1 for OR and 1 for XOR). Although the use of a contextual input makes it at least possible for the network to distinguish between two problems, when the networks are trained on OR and then XOR with context values, the current architecture is still unable to successfully solve OR after being trained on XOR. Thus, the mere presence of contextual values is not enough to overcome catastrophic interference, and it remains possible that a distributed

memory system is incapable of solving multiple problems, even when provided with a means to do so.

Introducing retraining. Another element which is also missing from the current architecture is that it is only allowed a single training session to learn each of the two problems. That is, it must learn OR and then after learning OR it must learn XOR. Very often, humans require multiple exposures to a problem, even after comprehending the problem in order to learn it. To simulate this aspect of multiple problem solving, we will allow the network to retrain itself on problems that have been interfered with. For example, after learning or then XOR, if the model has forgotten OR it will be allowed to retrain itself on OR. If this retraining causes it to forget XOR, then the model will then be allowed to train again on XOR, and so on, until the model can maintain both problems or until a specified number of repetitions have elapsed. In this study, 93% of the networks were able to learn both problems if retraining was allowed, and took anywhere from 4 to 29 problem presentations, with an average of 16 (meaning OR and XOR had to be trained 8 times each).

At this point, the use of context and retraining methods have helped improve resistance to catastrophic interference, but depending upon their initial state, some of these networks remain unable to solve XOR. Furthermore, these networks appear to require extensive training before learning both problems successfully. Again, the case for a two stage system remains persuasive, even with these embellishments. In the next section, we will examine if improving the learning efficiency of these neural networks will allow a distributed memory system to effectively solve multiple problems.

Cascaded networks

As an alternative to the feed forward architecture, Jordan Pollack (Pollack, 1987) described a cascaded architecture which was able to learn more efficiently. In Pollack's architecture, one network sets the activation weights for the other network, while the other network produces the output. Pollack's architecture is illustrated in Figure 2. Pollack referred to the two networks in the architecture as the Context network and the Function network. The Context network receives inputs and sets the weight values for the Function network. The Function network then uses those weight values to produce an output.

Using batch training (weights are only updated after the entire problem is presented), a learning rate of 0.5, and an initialization range of  $\pm 0.5$ , Pollack reported that it took about 30 batch epochs (meaning 120 total presentations of input/output pairs) to solve XOR to a minimum RMS of 0.2. Using the training methods and parameters described in Appendix A, we were able to improve the performance of this network so that on average, the networks took 5 epochs to learn XOR --almost 100 times faster than the traditional feed forward network. Furthermore, the network was able to solve XOR regardless of its initialization values (i.e. out of 1,000 simulations run, 100% of them were able to solve XOR, compared with 93%).

Mirrored cascaded networks. This improved learning efficiency is promising, but when approaching multiple problems it becomes difficult to determine where the contextual input should be placed (i.e. either in the Function Network, or the Context Network, or both). We found that regardless of where the contextual input was placed within this cascaded architecture, the cascaded networks performed worse than their

feedforward counterparts at learning multiple problems and avoiding catastrophic interference.

However, even though Pollack attributed the success of his networks as likely due to the separation of inputs, we found that by allowing inputs to be duplicated across the two subnetworks, the networks were able to learn efficiently and suffer from a minimal amount of catastrophic interference. This architecture is illustrated in Figure 3. The inputs in these networks are mirrored across the two subnetworks, hence we will refer to this architecture as a mirrored cascaded network (MCN). On average it took about 4 epochs for these networks to learn OR, then an additional 2 epochs for the networks to learn XOR. Finally, about 99 percent of the networks that were trained were able to maintain OR information after learning XOR. When retraining was allowed for each of the 11 (out of 1,000) networks that failed to retain OR, these networks required only one more presentation of OR in order to successfully learn both problems, taking a total average of about 10 epochs.

Extending MCN capacity. Currently, the networks that we have been working with have only a single layer of trainable weights. One way to extend the capacity of these networks is to allow an additional layer of units in the weight setting side (the right side) of the network as shown in Figure 4. In order to maximize the learning efficiency of this network, several changes were made to the learning parameters of the network.

These changes are detailed in Appendix A.

Given only two inputs, there are a maximum of 16 different Boolean expressions that can be calculated. These expressions are illustrated in Table 2. In order to uniquely identify each of the 16 expressions, 4 context nodes will be required. We trained the

network illustrated in Figure 5 to learn all 16 problems presented individually with retraining.

Out of 1000 attempts, all of the networks were able to accurately learn all 16 expressions, all 64 input/output pairs, within 181 epochs on average. On average, the networks were exposed to 52 different problems with a minimum of 37 and a maximum of 72 exposures. Since each of the problems had 4 input/output pairs, for a total of 64 input output pairs, and only 180 epochs were required on average, the constructive networks were exposed to each of the problems 3-4 times.

### **Discussion**

The models presented suggest several requirements of a distributed system. First, in order for a single distributed system (i.e. without positing separate or complementary learning systems) to store multiple items of information, *it is necessary for there to be some distinguishing factor between those items and that distinguishing factor must be part of the learning process*. Second, an exhaustive consolidation process is not necessary to produce knowledge that is resistant to interference, rather, if the system is able to learn information efficiently, that information can continue to exist even in the presence of new learning. Finally, given contextual information and efficient learning, when information is interfered with only minimal retraining is required.

Additionally, these models illustrate how a process of memory consolidation does not have to occur independently of learning, but rather, consolidation can be conceptualized as the learning process being repeated, either by internal rehearsal or external retraining. This conceptualization is supported by recent evidence indicating that

the process of memory consolidation does not occur independently from learning (Eichenbaum, 2006; Morris et al., 2006).

The complexity of computational architectures combined with their inability to fully represent human cognition may easily be the reason why such models have not been commonly adopted by neuropsychologists. However, it is also important to acknowledge that these types of models allow us to overcome our limited ability to conceptualize how distributed interactive systems work. Without such conceptualizations, we may be reduced to making artificial dichotomous categorizations (Newell, 1973; Van Orden et al., 2001). However, as these studies have demonstrated, computational models themselves can be built around intuitive assumptions about a division of functions, as is the case in the two stage memory model. Furthermore, limitations in the architectures that are being studied can easily lead to false conclusions about the function being simulated. At any point along the way in this small series of studies, we could have abandoned the architecture and presumed that its current state represents the limitation of a distributed memory system, thus requiring separate modules in order to function. In evaluating a computational architecture, it is important to ascertain if the architecture is provided with similar resources and capabilities that humans have at their disposal. However, this is often not the case as computational architectures generally do not start solving a new problem with years of previously acquired knowledge and are built solely to perform a single experimental task. Furthermore, architectures do not have a sophisticated representation of the environment, and most are not allowed to retrain themselves on a problem once it has been presented. As humans we make extensive use of our environment as we learn and remember information (Clark, 1997; Hutchins, 1995).

Without having these capabilities, it is premature to describe the internal workings of the architecture itself as a failure.

Re-conceptualizing the role of the medial temporal area

The medial temporal area of the brain, particularly the hippocampus, has long been championed as central to learning and memory, with the default conceptualization being one of storage. The intuitive explanation is that since new episodic memories are not formed after the hippocampus has been damaged, then the hippocampus is where new episodic memories are stored, at least temporarily. Thus, when the system breaks down, the conceptualization is that a storage unit has failed. This basic intuition is maintained today even in some of the most advanced computational models (Burgess & Hitch, 2005; Meeter & Murre, 2005; Norman et al., 2005; Norman & O'Reilly, 2003).

However, a more specific role is emerging in the literature for the hippocampus. Rather than a separate memory system, the hippocampus can also be conceptualized as a contextual generator, not representing full memories, but acting as an episodic tagging system that helps to organize new information as it is being learned (Eichenbaum & Fortin, 2003; Frankland & Bontempi, 2005; D. M. Smith & Mizumori, 2006). The model developed in this paper illustrates the benefits of having such a system: *if memories are stored in a distributed fashion, the introduction of contextual information during the process of learning allows for an automatic organization of separate sets of knowledge and a reduction of interference in that knowledge.* Additionally, as illustrated with the OR/XOR decision task, the contextual system allows a network to constructively learn new information (i.e. utilize previously learned information facilitate new learning). Intuitively, such constructive learning might be thought to require a complex working

memory system (Baddeley, 1986). Even in a more connectionist framework, such constructive learning is thought to require at least the addition of learning resources from one problem to the next (Quartz & Sejnowski, 1997). However, in the case of the MCNs, this type of constructive learning happens without requiring any specialized working memory system or the gradual addition of learning resources.

Several researchers have also demonstrated that contextual learning, and not simply declarative knowledge, is impaired in amnesics with hippocampal dysfunction (Chun & Phelps, 1999; Shanks et al., 2006). Given the importance of contextual information in learning as demonstrated by the models developed here, disruption of this contextual system could easily be conceptualized as leading to deficits in learning new information, while allowing older information to be retained. For example, if the contextual system was disrupted, or the contextual information became fixed, then each piece of newly learned information would simply override any other newly learned information. However, previously learned information that was encoded with an working contextual system could still be easily retrieved because the context that distinguished that information was successfully produced and encoded.

#### Developing conceptualizations in neuropsychology

Explanations of loss of functioning can easily confuse symptoms with the localization of function (e.g. inability to form new episodic memories is associated with damage to the hippocampus, therefore the hippocampus is the place where episodic memories are stored). For neuropsychologists, the benefits of honing in on true functional localization would be the development of more sensitive tests. For example, if the idea of the hippocampus as a contextual system holds true, then tests that more carefully

manipulate the degree of contextual information available may prove to be more sensitive to medial temporal damage. Such tests would focus not so much on the learning and recall of items, but upon manipulating the richness of the contextual information that is available as the information is being learned. Developing more sensitive tests would ultimately test the utility of more sophisticated conceptualizations. For example, given the prevalence of Alzheimer's disease, and the importance of early detection for treatment purposes, developing more sensitive instruments would be of immense clinical value. Learning tasks that emphasize contextual information over content are present in the experimental literature, but are currently not in clinical use. However, a recent review of the literature indicates that paired associate tasks utilizing associate recognition, where the context of the words is emphasized more than the words themselves may be most sensitive to early detection of Alzheimer's disease (Lowndes & Savage, 2007).

In describing the localization of functions in cognition, Luria notes that "In order to progress from establishment of the *symptom* (loss of a given function) to the *localization* of the corresponding mental activity, a long road has to be traveled." (Luria, 1973). The progression occurs not just via the adoption of the correct methodology, but with converging evidence over time across multiple disciplines and methods, including animal models, computational simulations, and research data. For the clinical neuropsychologists, the understanding of the conceptualizations provided by computational simulations may ultimately lead to the development of better methods of detecting and measuring dysfunction.

## References

- Anderson, J.A., & Rosenfeld, E. (1988). *Neurocomputing : foundations of research*.  
Cambridge, Mass.: MIT Press.
- Arbib, M.A. (2003). *The handbook of brain theory and neural networks* (2nd ed.).  
Cambridge, Mass.: MIT Press.
- Baddeley, A.D. (1986). *Working memory*. New York: Clarendon Press.
- Braak, H., & Braak, E. (1991). Neuropathological Staging of Alzheimer-Related  
Changes. *Acta Neuropathologica*, 82(4), 239-259.
- Burgess, N., & Hitch, G. (2005). Computational models of working memory: putting  
long-term memory into context. *Trends in Cognitive Sciences*, 9(11), 535-541.
- Chun, M.M., & Phelps, E.A. (1999). Memory deficits for implicit contextual information  
in amnesic subjects with hippocampal damage. *Nature Neuroscience*, 2(9), 844-  
847.
- Clark, A. (1997). *Being There : Putting Brain, Body, and World Together Again*.  
Cambridge, Mass.: MIT Press.
- Eichenbaum, H. (2006). The secret life of memories. *Neuron*, 50(3), 350-352.
- Eichenbaum, H., & Fortin, N. (2003). Episodic memory and the hippocampus: It's about  
time. *Current Directions in Psychological Science*, 12(2), 53-57.
- Farah, M.J. (1994). Neuropsychological Inference with an Interactive Brain - a Critique  
of the Locality Assumption. *Behavioral and Brain Sciences*, 17(1), 43-61.
- Frankland, P.W., & Bontempi, B. (2005). The organization of recent and remote  
memories. *Nature Reviews Neuroscience*, 6(2), 119-130.

- Gaffan, D. (2002). Against memory systems. *Philosophical Transactions of the Royal Society of London Series B-Biological Sciences*, 357(1424), 1111-1121.
- Harley, T.A. (2004). Does cognitive neuropsychology have a future? *Cognitive Neuropsychology*, 21(1), 3-16.
- Hodges, J.R., & Patterson, K. (2007). Semantic dementia: a unique clinicopathological syndrome. *Lancet Neurol*, 6(11), 1004-1014.
- Hutchins, E. (1995). *Cognition in the Wild*. Cambridge, Mass.: MIT Press.
- Hyman, B.T., Vanhoesen, G.W., Kromer, L.J., & Damasio, A.R. (1986). Perforant Pathway Changes and the Memory Impairment of Alzheimers-Disease. *Annals of Neurology*, 20(4), 472-481.
- Lecun, Y., Bottou, L., Orr, G.B., & Müller, K.R. (1998). Efficient backprop. In G.B. Orr & K.R. Müller (Eds.), *Neural Networks: Tricks of the Trade*. Berlin: Springer-Verlag.
- Lowndes, G., & Savage, G. (2007). Early detection of memory impairment in Alzheimer's disease: A neurocognitive perspective on assessment. *Neuropsychology Review*, 17(3), 193-202.
- Luria, A.R. (1973). *The working brain: an introduction to neuropsychology*. London: Allen Lane.
- 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(3), 419-457.

- McCloskey, M., & Cohen, N. (1989). Catastrophic interference in connectionist networks: the sequential learning problem. In G.H. Bower (Ed.), *The psychology of learning and motivation*, Vol. 24 (pp. 109-164). New York: Academic Press.
- Meeter, M., & Murre, J.M.J. (2005). TraceLink: A model of consolidation and amnesia. *Cognitive Neuropsychology*, 22(5), 559-587.
- Morris, R.G.M., Inglis, J., Ainge, J.A., Olverman, H.J., Tulloch, J., Dudai, Y., & Kelly, P.A.T. (2006). Memory reconsolidation: Sensitivity of spatial memory to inhibition of protein synthesis in dorsal hippocampus during encoding and retrieval. *Neuron*, 50(3), 479-489.
- Newell, A. (1973). You can't play 20 questions with nature and win: Projective comments on the papers of this symposium. In W.G. Chase (Ed.), *Visual information processing*, (pp. 283-308). New York: Academic Press.
- Norman, K.A., Newman, E.L., & Perotte, A.J. (2005). Methods for reducing interference in the Complementary Learning Systems model: Oscillating inhibition and autonomous memory rehearsal. *Neural Networks*, 18(9), 1212-1228.
- Norman, K.A., & O'Reilly, R.C. (2003). Modeling hippocampal and neocortical contributions to recognition memory: A complementary-learning-systems approach. *Psychological Review*, 110(4), 611-646.
- Plunkett, K., & Elman, J.L. (1997). *Exercises in rethinking innateness a handbook for connectionist simulations*. Cambridge, Mass.: MIT Press.
- Quartz, S.R., & Sejnowski, T.J. (1997). The neural basis of cognitive development: a constructivist manifesto. *Behav Brain Sci*, 20(4), 537-556; discussion 556-596.

- Ranganath, C., & Blumenfeld, R.S. (2005). Doubts about double dissociations between short- and long-term memory. *Trends in Cognitive Sciences*, 9(8), 374-380.
- Rorden, C., & Karnath, H.O. (2004). Using human brain lesions to infer function: a relic from a past era in the fMRI age? *Nature Reviews Neuroscience*, 5(10), 813-819.
- Rumelhart, D.E., Hinton, G., & Williams, R. (1986). Learning internal representations by error propagation. In D.E. Rumelhart & J.L. McClelland (Eds.), *Parallel Distributed Processing, Vol 1: Foundations*. Cambridge, MA: The MIT Press.
- Scoville, W.B., & Milner, B. (1957). Loss of Recent Memory after Bilateral Hippocampal Lesions. *Journal of Neurology Neurosurgery and Psychiatry*, 20(1), 11-21.
- Seidenberg, M.S. (1988). The Cognitive Neuropsychology of Language - Coltheart, M, Sartori, G, Job, R. *Cognitive Neuropsychology*, 5(4), 403-426.
- Shanks, D.R., Channon, S., Wilkinson, L., & Curran, H.V. (2006). Disruption of sequential priming in organic and pharmacological amnesia: A role for the medial temporal lobes in implicit contextual learning. *Neuropsychopharmacology*, 31(8), 1768-1776.
- Smith, D.M., & Mizumori, S.J.Y. (2006). Hippocampal place cells, context, and episodic memory. *Hippocampus*, 16(9), 716-729.
- Smith, M. (1993). *Neural networks for statistical modeling*. New York: Van Nostrand Reinhold.
- Squire, L.R. (2004). Memory systems of the brain: A brief history and current perspective. *Neurobiology of Learning and Memory*, 82(3), 171-177.

Squire, L.R., Stark, C.E.L., & Clark, R.E. (2004). The medial temporal lobe. *Annual Review of Neuroscience*, 27, 279-306.

Van Orden, G.C., Pennington, B.F., & Stone, G.O. (2001). What do double dissociations prove? *Cognitive Science*, 25(1), 111-172.

Wilson, D.R., & Martinez, T.R. (2003). The general inefficiency of batch training for gradient descent learning. *Neural Networks*, 16(10), 1429-1451.

## **Appendix A: Training Techniques**

There are many ways of tuning the training process of neural networks. Because of the large number of interactive parameters involved, including the architecture that is used, the question of interest, and the problem to be solved, there is no single agreed upon way in which to code problems, train the network, and discontinue training (see (Arbib, 2003), for an example of the wide variety of network applications). In the present work, a specific training regime has been developed in order to enable the networks to solve problems faster and in an informative manner.

Stochastic Training. In stochastic training, the network is shown one of the input/output pairs randomly, and then the weights are adjusted. The network is then shown a randomly chosen input/output pair from the remaining pairs, and so on, until all of the pairs for a problem have been presented, then the process repeats. Strong arguments have been made for the use of stochastic learning procedures when the problem data is large, redundant, or noisy (Lecun et al., 1998; Wilson & Martinez, 2003). There are several reasons stochastic training rather than batch training is used in the current work. First of all, it more realistic to conceptualize learning as something that is

continually occurring. Consequently, it is more appropriate that the learning process (i.e. a weight adjustment) should be occurring with each input/output pair rather than waiting until all input/output pairs for a problem have been presented. An additional benefit to the use of stochastic training is that the term learning trial, often referred to as an *epoch*, now refers to how many times a single input/output pair was presented to the network rather than how many times the entire problem was presented. This results in a more precise measure of performance and also makes comparisons between problems that may have a different number of input/output pairs easier to interpret. Finally, batch training does not allow a problem to be solved until all of the input/output pairs have been presented at least once. Some of the networks that will be presented can find solutions to a problem without training on all of the input/output pairs.

Stochastic training is often used with replacement. That is, after the network is trained on a randomly selected input/output pair, that pair is made available for selection again rather than removing that pair from selection until all pairs in the problem have been presented. Replacing each pair after training adds a small amount of noise to the training process, but this small amount of noise can sometimes be useful in pushing a network out of local minima. In the present work, the networks use stochastic learning *without* replacement because this is more reflective of approaching each problem as a whole while still learning after each input/output pair is presented. Additionally, the use of stochastic training with replacement makes epoch interpretation more difficult (e.g. if a network learns a problem with 4 input/output pairs after 8 epochs with replacement, it is unknown if the network saw each of the pairs twice, or if the network was exposed to all of the input/output pairs only after 8 epochs).

Input coding. Neural network inputs for simplified problems like the logic problems used in the present work are often coded with values of 1 or 0 (e.g. Rumelhart et al., 1986). However, the use of 0 values reduces the ability of a network to utilize the absence of information as it attempts to solve a problem. When an input value is zero, all signals from that input node will be zero because a connection's weight is multiplied by the output of the sending node. However, if 0 inputs are replaced with -1's, the network is still able to computationally take advantage of the absence of information. For the networks explored in the present work, the use of a negative value (-1) instead of 0 allows both feed-forward and cascaded networks to solve problems faster. For all networks in the present work, inputs will be coded with -1 and 1 unless otherwise specified.

Output coding. It is common practice to place the output values at points before asymptotes of the transformation function (this is also recommended by Rumelhart and McClelland, 1986, p. 329). For example, the logarithmic sigmoid function has asymptotes of 0 and 1. This means that as the input value to the function approaches infinity the output value of the function will become closer to 1 and as the input value approaches negative infinity the output value of the function will become closer to 0. However, since in practice infinity is never an input value the output of the function can never precisely equal the target values. This can lead to some very large weights as the network struggles in vain to reach these target values. Large weight values can lead a network into a local minimum that may take a large number of epochs to leave, or in which it may become stuck. For all networks in the present work, target output values are 0.1 and 0.9 for networks with logarithmic sigmoid transformation functions and -0.9 and

0.9 for networks with hyperbolic tangent transformation functions unless otherwise specified.

Criteria for stopping training. In order to ensure that comparisons across networks and across problems with different numbers of input/output pairs can be made and that the networks themselves are still producing a low overall amount of error, unless otherwise specified, networks were trained until all outputs were correct *and* the square-root of the means squared error (RMS) falls below 0.25 for networks with logarithmic sigmoid output functions and 0.50 for networks with hyperbolic tangent functions.

Learning rate, momentum, and initialization values. The value of the learning rate has a critical impact upon the performance of networks that are attempting to solve multiple problems. For smaller networks the learning rate is typically set at 0.6. However, the more connections that a network has, the more likely it is that a large learning rate will make learning the problem more difficult. Additionally, if a network has a large number of connections it is also often beneficial to use smaller learning rates for connections in higher levels of the network. This is because connections that are closer to the output levels of the network will receive larger error values and thus produce larger weight adjustments (Lecun et al., 1998). In the present work, the learning rates for the networks are set at 0.6 unless otherwise specified.

The momentum value for all simulations was set at 0.9, and the effects of manipulating momentum are not explored within the present work (changes in the momentum constant does not seem to have a large impact on network performance and some of the effects of momentum can be accounted for by changing the learning rate).

The feed forward networks that are described were much more sensitive to their initial weight values than the cascaded networks, and were more successful when a larger initialization range was used. For these studies, an initialization range of  $\pm 0.5$  was used for the feed forward networks. Cascaded networks were less sensitive to their initialization state but benefited slightly from smaller initialization values. Hence for these networks an initialization range of  $\pm 0.1$  was used.

Training multiple networks. Neural networks using back-propagation will arrive at exactly the same solution if they are provided with the same initialization values. Because initialization values are chosen randomly each time a network is run, some initialization values may produce networks that solve the problem quickly while others may produce networks that are unable to solve the problem at all. In order to determine the extent to which the networks that are used in the present work are able to solve the problem(s) regardless of their starting initialization values, multiple networks are run for each condition of each study. To ensure that the results of each condition in each study are not due to a chance combination of starting values, each condition for each study is simulated with 1,000 networks.

Maximizing the learning efficiency of the 2 layer MCN. Instead of logarithmic sigmoid functions (with 0 to 1 outputs) which are typical for hidden units in feed forward networks, hyperbolic tangent (hTan) transformation functions (with a -1 to 1 range) for hidden and output units were used. The use of hTan functions changes several other aspects of network training. Outputs are now coded as near the asymptotes of the hTan function (instead of 0.1 and 0.9, they are now -0.9 and 0.9). Additionally, networks with hTan functions (and networks with a large number of hidden units) appear to be more

sensitive to large learning rate values. Furthermore, because of this sensitivity, and because of the large number of hidden units (10) being used, smaller learning rates were used for connections to output units than for connections to hidden units. This is to adjust for the fact that the connections to the output units are receiving larger error values and thus are subject to change more quickly as opposed to the connections to hidden units (Lecun et al., 1998). In the Cascaded Networks, if a trainable weight was connected to a weight setting node that set the weight for a connection to an output unit, then the learning rate for connections to output units was used for that weight. If the trainable weight was connected to a weight setting node that set the weight for a connection to a hidden unit, then the learning rate for connections to hidden units was used for a connection to a hidden unit, then the learning rate for connections to hidden units was used for that weight. Finally, the use of hTan functions also changes the minimum error value that is used. Because the network's output has twice the range of output than a network with logSig functions (-1 to 1 vs. 0 to 1), a minimum error of 0.50 instead of 0.25 allowed.

TABLE 1: Arguments and results for Boolean OR and exclusive-or (XOR)

| Input 1 | Input 2 | OR output | XOR output |
|---------|---------|-----------|------------|
| 0       | 0       | 0         | 0          |
| 0       | 1       | 1         | 1          |
| 1       | 0       | 1         | 1          |
| 1       | 1       | 1         | 0          |

TABLE 2: The 16 Boolean expressions that are possible with 2 arguments

| Inputs:      | 0,0 | 0,1 | 1,0 | 1,1 |
|--------------|-----|-----|-----|-----|
| FALSE        | 0   | 0   | 0   | 0   |
| AND          | 0   | 0   | 0   | 1   |
| X AND NOT Y  | 0   | 0   | 1   | 0   |
| Y AND NOT X  | 0   | 1   | 0   | 0   |
| X            | 0   | 0   | 1   | 1   |
| Y            | 0   | 1   | 0   | 1   |
| XOR          | 0   | 1   | 1   | 0   |
| OR           | 0   | 1   | 1   | 1   |
| NOT (X OR Y) | 1   | 0   | 0   | 0   |
| EQUALS       | 1   | 0   | 0   | 1   |
| NOT X        | 1   | 1   | 0   | 0   |
| NOT Y        | 0   | 0   | 1   | 1   |
| $X \geq Y$   | 1   | 0   | 1   | 1   |
| $Y \geq X$   | 1   | 1   | 0   | 1   |
| NOT(X AND Y) | 1   | 1   | 1   | 0   |
| TRUE         | 1   | 1   | 1   | 1   |

FIGURE 1: a feed-forward network that can solve XOR

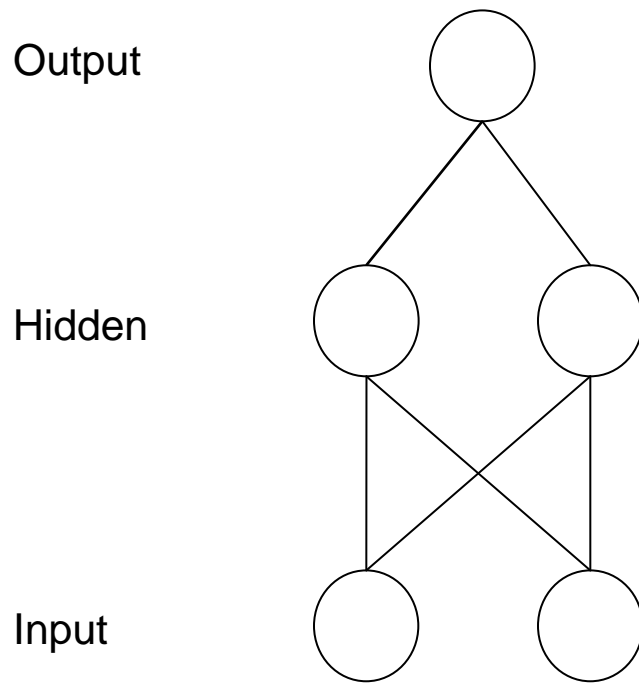


FIGURE 2: Pollack's architecture

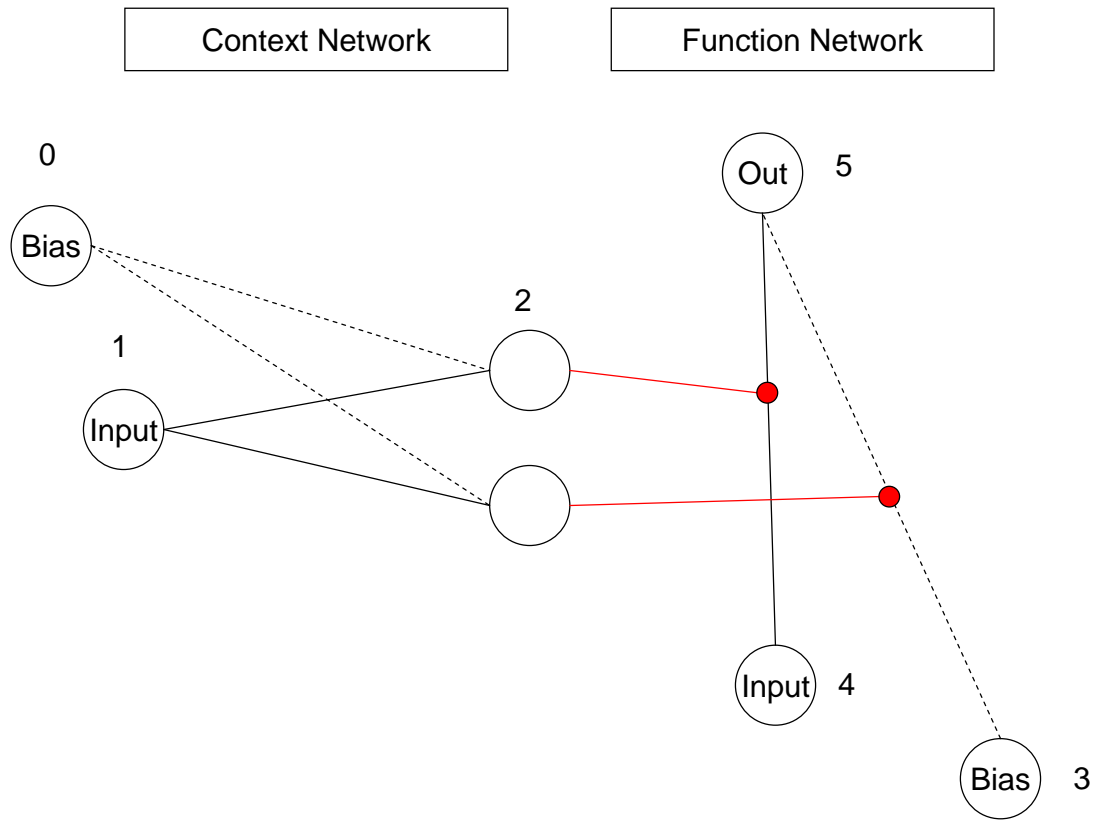


FIGURE 3: A mirrored cascaded architecture

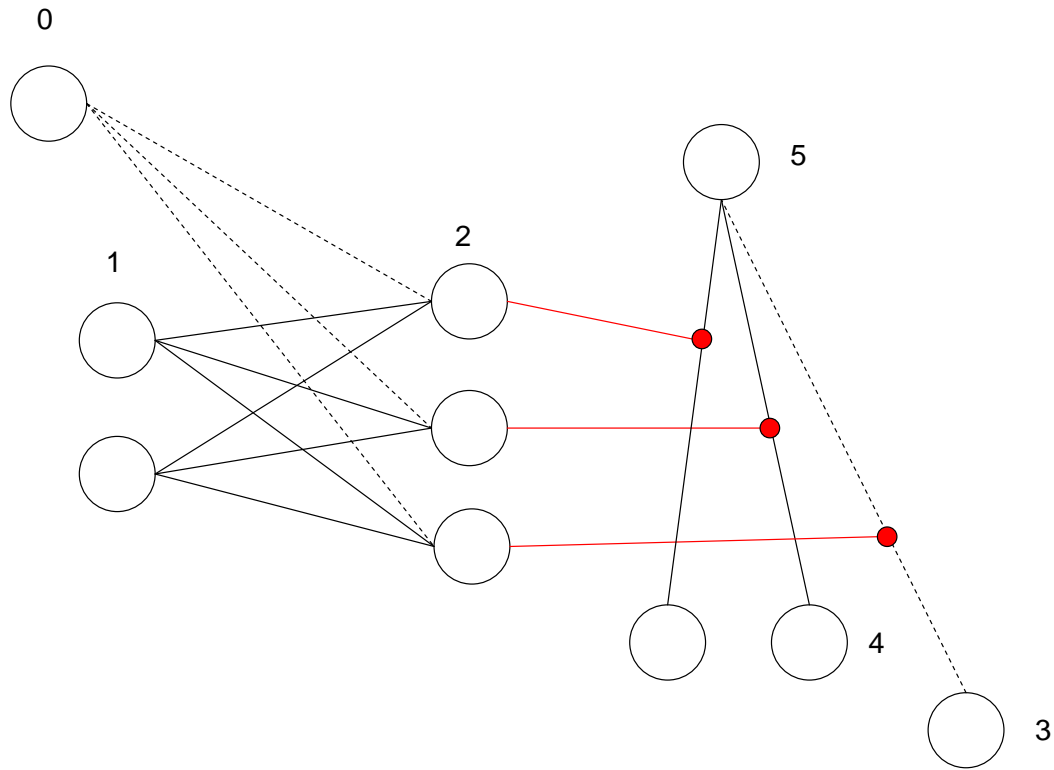


FIGURE 4: A mirrored cascaded architecture with hidden units.

