Virtual Memories and Massive Generalization in Connectionist Combinatorial Learning

Olivier Brousse and Paul Smolensky

Department of Computer Science &
Institute of Cognitive Science
University of Colorado at Boulder
Boulder, CO 80309-0430
olivier@boulder.colorado.edu
smolensky@boulder.colorado.edu

Abstract

We report a series of experiments on connectionist learning that addresses a particularly pressing set of objections to the plausibility of connectionist learning as a model of human learning. Connectionist models have typically suffered from rather severe problems of inadequate generalization (where generalizations are significantly fewer than training inputs) and interference of newly learned items with previously learned items. Taking a cue from the domains in which human learning dramatically overcomes such problems, we see that indeed connectionist learning can escape these problems in *combinatorially structured domains*. In the simple combinatorial domain of letter sequences, we find that a basic connectionist learning model trained on 50 6-letter sequences can correctly generalize to about 10,000 novel sequences. We also discover that the model exhibits over 1,000,000 *virtual memories*: new items which, although not correctly generalized, can be learned in a few presentations while leaving performance on the previously learned items intact. We conclude that connectionist learning is not as harmful to the empiricist position as previously reported experiments might suggest.

1. Introduction

Two important capabilities of human learning that connectionist models have until now seemingly failed to share are these: Acquiring competence from a small set of examples, as children do when they learn their native language by being exposed to only a very small fraction of what they are ultimately competent with, and fast learning of new items with no interference, as when we learn new fact from a single presentation. More specifically, connectionism has until now suffered from the following two problems:

> *The connectionist generalization problem:* Generalizations are few, and do not outnumber training examples. Current models suggests that in order to obtain correct performance on a target set of inputs, a network needs to be trained on a sizable fraction (between 25% and 75%) of the learning set. (While the amount of information that is available during training is still an open issue in the ongoing debate between empiricists and nativists, it would be hard to find *anyone* even remotely comfortable with the idea that children are exposed to 25%-75% of their ultimate competence).

> *The connectionist interference problem:* New items to be learned are in current practice intermingled with all the previously trained inputs and subjected again to the lengthy and rather laborious training algorithm. Several experiments have shown that if a network successfully trained on one set of items is then trained on another, the network will unlearn the first set. In McCloskey & Cohen (1988), this is referred as "catastrophic interference."

There is no doubt that connectionism would fail in its claim to be a plausible model for human learning if catastrophic interference and weak generalization were invariably prevailing in all domains. We have hypothesized, however, that connectionist networks will not suffer from the two problems mentioned above in *combinatorial domains*. In this paper we will further define this hypothesis, and report on a series of experiments supporting it.

2. Hypothesis

Taking a cue from the domains in which human learning dramatically overcomes the problems of slow and interferential learning, like language and facts, we have hypothesized that connectionist models will not suffer from the generalization and interference problems in *combinatorial domains*. More specifically, we hypothesize that in such domains, we will see:

> *Massive generalization*: To learn a set of inputs, only a fraction of the set will need to be trained upon.
>
> *Fast, interference-free learning*: Once a network has learned a subset of inputs, to learn a new input which shares in the structure of previously trained inputs, the new input will only need to be presented a few times and there will be no interference with the previously trained inputs although they are not be trained again.

We will take a domain to be combinatorial if the elements in the domain are constructed by combining smaller elements, and if the correct processing of larger elements can be generalized from correct processing of smaller elements of which they are composed, taking due consideration of the means of composition.

3. Experiments

To test our hypothesis, we have chosen one of the simplest combinatorial domains possible: Cartesian products of sets. X_i are sets, for $i = 1, \ldots, n$, and our combinatorial domain is

$$X = X_1 \times X_2 \times \cdots \times X_n = \{(x_1, x_2, \ldots, x_n) \,/\, x_i \in X_i\}$$

X is the domain consisting of sequences of n elements, the ith element in each sequence being some member of X_i. In our experiments, all X_i have the same number of elements. We train an auto-associative network on a randomly selected subset of the chosen domain, and then test and train for correct association each of the remaining items of the domain, one by one. The auto-associative network, associating each item with itself, can be interpreted as a graded *recognizer* of whether an input shares in the regularities of the training set.

In our connectionist experiments, each element of each set was represented in the network as some pattern of activities. It is these patterns, of course, that matter, and not whatever name we find convenient to give to the elements. In this paper we will use letters as convenient labels for these elements. As a further convenience we will use the same set of labels for each set X_i in the Cartesian product, but again this is of no consequence. (In particular the network is not charged in any way with discovering that we like to use the same letter to label an element in X_1 and an element in X_2.) Thus if $n = 4$, a typical element of X could be written (A, B, A, C) or, more simply, as the string ABAC. We call the number of elements in X_i A, the alphabet size.

Architecture and training technique

The connectionist learning technique we used is a standard one: auto-association using back-propagation learning.

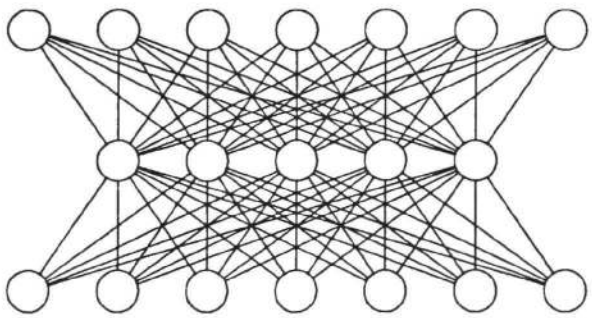

Figure 1: Architecture of the networks used in these experiments

The network is a three-layer feed-forward network in which the input layer and output layer have N units

and the hidden layer has H units. The architecture is shown in Figure 1. Each training input (one of the elements of X) is represented as a pattern of activity on the N input units according to a mapping described below. The target output on the N output units is identical to the input pattern: the network must associate each element of X with itself. The network was trained with standard back-propagation, the units of the network being semi-linear units as in Rumelhart, Hinton and Williams (1986).

Representation

Each of the n elements in X were coded using tensor product representation as defined in Smolensky (1987). Random binary vectors were generated to represent each $x_i \in X_i$ (fillers) while the associated roles were a vector representing the set X_i it belonged to. In all the experiments we will report on, the role vectors were simply the vectors of null activities with the exception of the ith coordinate of X_i which had activity 1. The representations were thus semi-local, as defined in Smolensky (1987), and amounted to a simple concatenation of the representations of the x_i's. Thus, in the case $n = 3$, if the random binary vectors of activities representing $x_1 \in X_1$, $x_2 \in X_2$ and $x_3 \in X_3$ were (1, 0, 1, 1, 0), (0, 0, 1, 0, 0) and (1, 1, 1, 0, 0), respectively, then the vector of activities representing (x_1, x_2, x_3) was simply (1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0) = (1, 0, 1, 1, 0) * (1, 0, 0) + (0, 0, 1, 0, 0) * (0, 1, 0) + (1, 1, 1, 0, 0) * (0, 0, 1), where * denotes the tensor product operation.

Performance measures

Our basic measure of the network's performance on a particular input was the number of output units that were "correct": within a certain error criterion ε of the correct value. In all experiments reported in this paper, we used $\varepsilon = 0.4$. For each experiment, the network was initialized with a random set of small weights, and the back-propagation algorithm was applied to each pattern of the training set, in a random order, weight updates being performed after each pattern presentation. Application of the learning algorithm to the training set was repeated until all inputs were correctly associated according to the performance measure mentioned above. The reader can refer to the appendix for further information on the training procedure and the values of the experimental parameters. In many experiments the "control group" against which performance was tested was the set of all possible inputs with activities in {0, 1}. We called patterns belonging to this set "random bit patterns".

4. Results

4.1. Regularity detection: English 4-letter words

We report here on early experiments designed to test the basic assumption that a feed-forward auto-associator is capable of learning through back-propagation to recognize whether an unfamiliar sequence shares in the combinatorial regularities characterizing some domain.

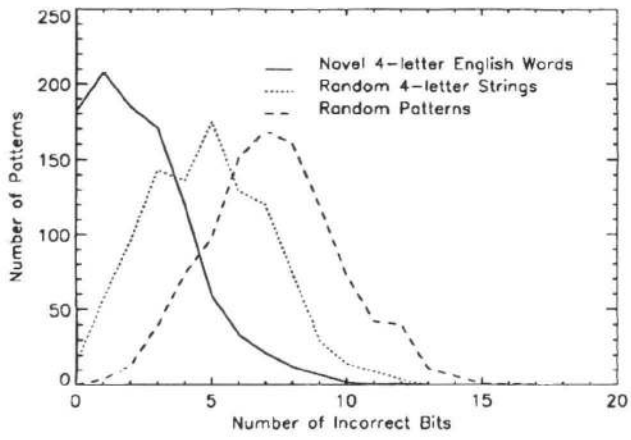

Figure 2: Generalizations; Network trained on 100 English 4-letter words. The network generalizes best on novel English words, then on 4-letter strings, then on random bit patterns.

We took the domain X to be a set of 1100 4-letter English words, and trained a network (here and henceforth, a back-propagation feed-forward auto-associator) on 100 randomly selected such words. We then tested its generalization ability on the 1000 remaining untrained words, on 1000 randomly selected 4-letter strings, and 1000 random bit patterns. If the network can recognize the degree to which new patterns share in the regularities with the training set, it should generalize best with English words, then 4-letter strings, then random-bit patterns. This is confirmed experimentally in Figure 2, where 87% English words have less that 5 incorrect bits, versus 45% for random strings and 22% for random bit patterns.

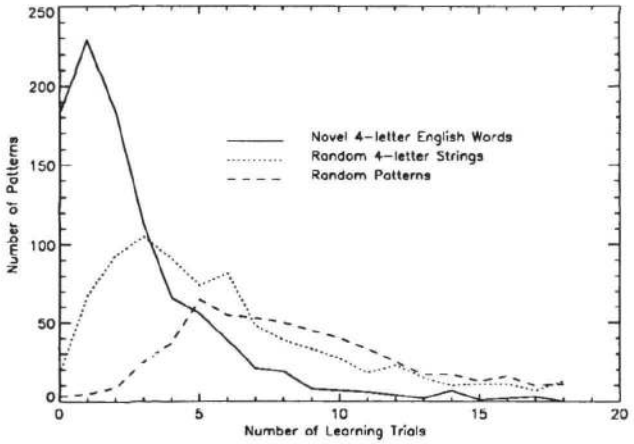

Figure 3: Number of weight updates to learn a new input, after training on 100 4-letter English words. The network learns novel English words the fastest, then 4-letter strings, then random bit patterns.

Not only do new inputs that share in the regularities of the training set produce fewer erroneous output bits, but they are also easier to learn, as shown in Figure 3.

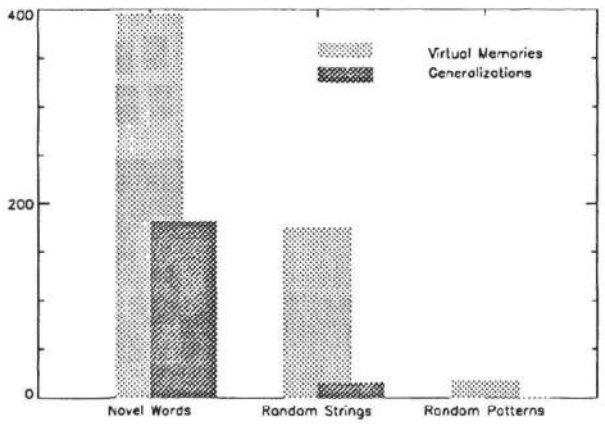

Figure 4: The number of generalizations and virtual memories for the network trained on 100 English 4-letter words.

In the following experiments we will summarize information on generalization and ease of learning of a new input by reporting just the number of generalizations (the number of novel patterns with zero incorrect bits) and the number of *virtual memories*. We define a virtual memory to be a novel input which can be trained to criterion while *leaving performance on the training set error-free*. Figure 4 shows both

the generalizations and virtual memories for 1000 untrained words, 1000 randomly selected 4-letter strings, and 1000 random bit patterns. (For computational time reasons, we restricted the number of learning trials when testing for a virtual memory to 5. All our results concerning their number are thus lower bounds only: The use of lower learning rates and/or larger number of trials could yield higher numbers. We will henceforth mean virtual memories that can be learned in less than 5 trials when we refer to virtual memories.) We observe that the number of generalizations and virtual memories is the biggest for the set of English words, then for the set of random 4-letter strings. For the random selection of 1000 random bit patterns, there were simply no generalizations.

4.2. Learning in the Cartesian product domain

In this section we describe the results of our main experiments, addressing learning in Cartesian product domains $X = X_1 \times X_2 \times \cdots \times X_n$ for various values of n and sets X_i.

Generalization: The number of generalizations in networks trained on 50 inputs in the case $A = \|X_i\| = 26$ and $n = 2,3, \ldots, 6$ is shown in Figure 5, in a semi-logarithmic plot. For the cases $n = 2$ and $n = 3$, it was possible to test the entire set of untrained inputs;

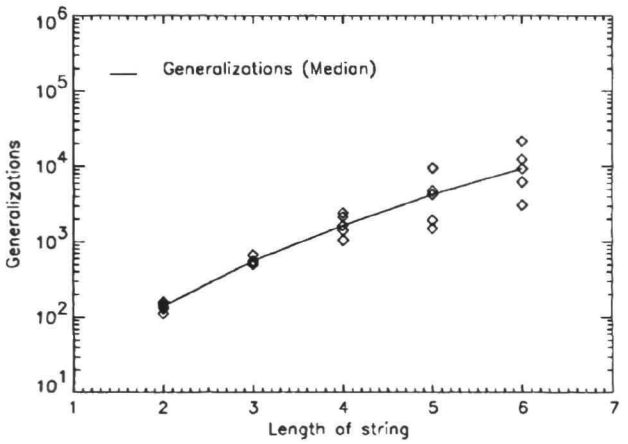

Figure 5: The number of generalizations for networks trained on sets of size 50, with $A = 26$, as n varies from 2 to 6.

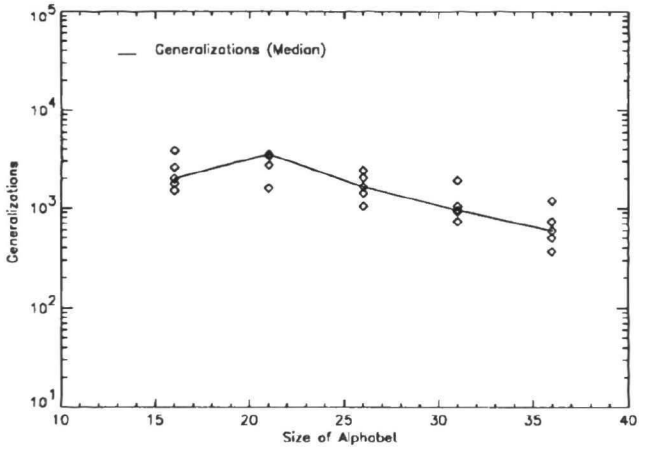

Figure 6: The number of generalizations for networks trained on sets of sizes 50, with $n = 4$, and $A = 16, 21, 26, 31, 36$.

The number of generalizations for these two cases is thus exact. For the cases $n = 4, 5, 6$, complete testing was not feasible for computational time reasons. We therefore tested a sample of T randomly-generated

inputs (with replacement). The number of generalizations plotted (and later, the number of virtual memories) is thus an estimate, assuming unbiased samples. For each value of the varying parameter (in this case n), we repeated our experiments 5 times (as in all subsequent experiments), each time starting with new random initial weights, a new randomly selected training set, and a new randomly selected testing set of T patterns in the cases $n = 4, 5, 6$. For $n = 4$ and 5, T was 10,000. For $n = 6$, T was 100,000 for generalizations and 10,000 for virtual memories. Figure 5 displays the number of generalizations obtained for each experiment, as well as the median number of generalizations obtained. Figure 6 shows the number of generalizations as the size of X_i's vary.

Virtual memories: Figure 7 shows the number of virtual memories in the case of a network trained on 50 input patterns, for $n = 2, 3, ..., 6$.

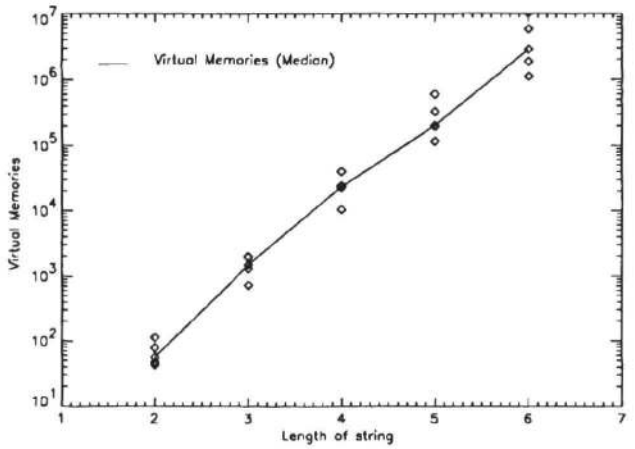

Figure 7: The number of virtual memories for networks trained on sets of size 50, with $A = 26$, as n varies from 2 to 6.

As the combinatorial complexity of the domain increases, we see that large numbers of virtual memories are obtained. For $n = 6$, for instance, we estimate that about 2.8 million virtual memories exist.

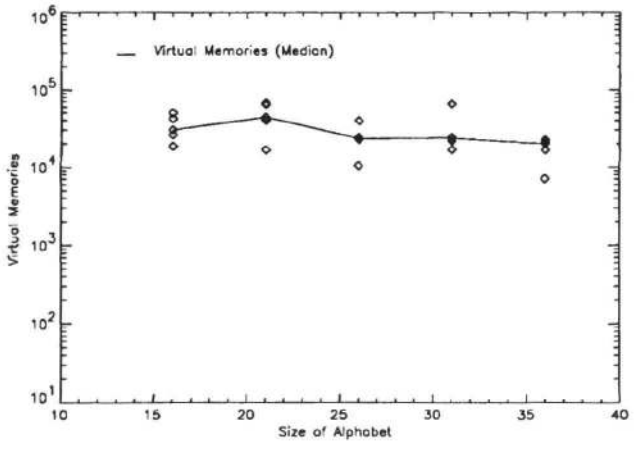

Figure 8: The number of virtual memories for networks trained on sets of sizes 50, with $n = 4$, and $A = 16, 21, 26, 31, 36$.

Figure 8 shows the number of virtual memories for networks trained on sets of size 50, with $n = 4$ and $A = 16, 21, 26, 31, 36$.

Discrimination tests: To see how well our networks were doing at discriminating elements from non-elements of X, we conducted a number of tests using random bit patterns. Although in all our experiments the number of hidden units was always smaller than the number of input and output units, thus preventing the network from computing the identity function, we needed to make sure that it was not computing even an approximation of it. This is confirmed in table 1, where the first row shows the ratio of the number of generalizations in a sample of 10,000 members of X (with $A = 26$ and $n = 4$) to the number of generalizations obtained by testing the same networks with 10,000 random bit patterns. The second row shows the ratios of the number of virtual memories in a sample of 10,000 members of X (with $A = 26$ and $n = 4$) and the number of virtual memories obtained by testing the same networks with 10,000 random bit patterns. We see that the networks generalize poorly for random bit patterns, about 1/35th as well as for elements of X. Similarly, there are about 1/10th as many random bit vectors which are virtual memories as elements of X. Statistical analyses, along with results of additional experiments on discrimination, can be found in Smolensky, Brousse & Mozer (forthcoming).

Experiments	1	2	3	4	5
Discrimination Ratio for Generalizations	46.99	22.99	51.86	35.22	31.46
Discrimination Ratio for Virtual Memories	6.81	10.40	10.41	12.19	7.26

Table 1: Ratios of generalizations for members of X and random-bit vectors, and ratios of vitual memories for members of X and random-bit vectors.

5. Conclusion

Further experiments and analyses to illucidate these results are in progress, but computational costs are a limiting factor. The data points alone displayed here represent the multiprocessor equivalent of roughly 5,400 hours of Sun3 time. The experiments reported above give optimistic results with respect to the generalization and interference problems for connectionist learning. A fuller discussion of these experiments may be found in Smolensky, Brousse & Mozer, forthcoming. While training a network from scratch may be a lengthy process, we have seen that once a network has acquired some knowledge of the combinatorial domain on which the training is performed, subsequent learning of members of the domain is much easier and less prone to interference than was previously thought. Although we do *not* provide evidence that connectionist induction algorithms are stronger than previously available inductive techniques, we do believe that we have provided evidence that connectionism is more compatible with an empiricist position on human learning than previous results would suggest—at least within combinatorial domains.

Acknowledgments

The authors would like to thank Michael Mozer and Jay McClelland, as well as Gary Bradshaw, Clayton Lewis, Michael Main, and Kelvin Wagner for insightful conversations about this research. Special thanks also to the users of the Encore Multimax machine at the University of Colorado at Boulder for tolerating intensive CPU usage. Computer simulations used a modification of the back propagation simulator of McClelland & Rumelhart (1988).

This work has been supported by NSF grants IRI-8609599 and ECE-8617947 to the second author, by a grant to the second author from the Sloan Foundation's computational neuroscience program, and by the Optical Connectionist Machine Program of the NSF Engineering Research Center for Optoelectronic Computing Systems at the University of Colorado at Boulder.

Appendix: Experimental parameters

In all our experiments, the momentum was 0.9 and the error criterion ε was 0.4 both for training and for testing of virtual memories. The learning rate was 0.01 for training and 0.2 for virtual memory learning, except for $n = 2$ where the training learning rate was 0.005. The vectors representing X_i were random binary vectors of length 8. H, the number of hidden units, was linearly increased as n increased according to $h = 5 \times n$, resulting in a constant compression factor of 8:5 from input to hidden units. Initial weights were generated pseudo-randomly, with equal probability in the interval [-0.5, 0.5]. Patterns in the training

set were presented to the network in a random order during an epoch, and weights were updated after each pattern. Because gradient descent suffers from the problem of local minima, some training sets could not be learned in a reasonable time. When that happened we simply started the experiment over. When the error for all inputs of the training sets reached 0, we trained again for 10 epochs to ensure stability. (Since weights are changed after each pattern presentation, a total error of 0 at the end of one epoch does not guarantee that the next will still contain error-free patterns.) All networks were standard three-layer feed-forward back-propagation networks, with bias on all hidden and output units.

The following table shows other relevant parameters. We only show minima and maxima for the number of epochs during training displayed in the rightmost columm. The column labeled "Training set" refers to the number of input patterns in the training sets used.

Figure	Domain X: A	Domain X: n	X: Constraint	Hidden Units	Training Set	Epochs
2	26	4	English	20	100	255
3	26	4	English	20	100	255
4	26	4	English	20	100	555
5	26	Varies	none	Varies ($5n$)	50	119-486
6	Varies	4	none	20	50	164-259
7	26	Varies	none	Varies ($5n$)	50	119-486
8	26	4	none	20	50	164-259

References

McClelland, J.L. & Rumelhart, D.E. (1988). *Explorations in Parallel Distributed Processing: A handbook of models, programs, and exercises.* Cambridge, MA: MIT Press/Bradford Books.

McCloskey, M., & Cohen N.J. (1988). Catastrophic interference in connectionist networks: The sequential learning problem. To appear in G. H. Bower (Ed.), *The Psychology of learning and motivation: Volume 23.*

Rumelhart, D.E., Hinton, G.E., & Williams, R.J. (1986). Learning internal representations by error propagation. In D. E. Rumelhart, J. L. McClelland, & the PDP Research Group, *Parallel distributed processing: Explorations in the microstructure of cognition. Volume 1: Foundations.* Cambridge, MA: MIT Press/Bradford Books.

Smolensky, P., Brousse, O., & Mozer, M. (forthcoming). Exponential growth of generalizations and virtual memories in connectionist combinatorial learning. To be submitted to *Cognitive Science.*

Smolensky, P. (1987). On variable binding and the representation of symbolic structures in connectionist systems. *Technical Report CU-CS-355-87.* Department of Computer Science, University of Colorado at Boulder.