COMPACT REPRESENTATIONS OF WORD LOCATION INDEPENDENCE IN CONNECTIONIST MODELS

Frédéric Dandurand
Jonathan Grainger

Laboratoire de psychologie cognitive, CNRS & Université de Provence
UMR 6146, Case D, Bâtiment 9, 3 Place Victor Hugo, 13331 Marseille Cedex 3
France
frederic.dandurand@univ-provence.fr
jonathan.grainger@univ-provence.fr

ABSTRACT
We studied representations built in Cascade-Correlation (Cascor) connectionist models, using a modified encoder task in which networks learn to reproduce four-letter strings of characters and words in a location-independent fashion. We found that Cascor successfully encodes input patterns onto a smaller set of hidden units. Cascor learned simultaneously regularities related to word structure ("word-ness", e.g., that certain letters never occur in certain positions) and position (location invariance). We also found the representations built are compatible with the concept of open bigrams: networks make the least error on contiguous, forward strings of two letters, marginally more on non-contiguous forward bigrams, and significantly more on backward or reversed bigrams.

KEY WORDS
Reading, language, open bigrams, modelling, simulation, neural networks.

1. Introduction

Several recent computational models of cognition posit a hierarchical system of processing in which simple and local features are gradually integrated into more abstract and complex features using receptive fields (e.g., [1]). The visual cortex (V1, V2) is known to detect contrasts and low level features such as lines and contours. Applied to the domain of reading and language processing, one can wonder what kinds of abstractions the visual system builds based on retinal information.

In this study, we investigate the representations built in a constructive neural network technique called cascade-correlation [2]. Using a modified encoder task, we ask if cascade-correlation (Cascor) networks perform data reduction and build compact and abstract representations, namely open bigrams [3].

2. Methods

Cascade-correlation (Cascor) is a connectionist (neural network) algorithm for supervised learning. In contrast to the popular backpropagation architecture, Cascor networks not only adjust connection weights as a result of learning, but also build an appropriate network architecture.

The Cascor algorithm begins with a simple network topology consisting of input and output units only, and recruits hidden units until the network has sufficient computational resources to solve some task. Learning in Cascor proceeds in an alternation of two phases. In input phase, input weights of candidates units in a pool are trained to maximize the covariance between their outputs and the residual error. The candidate unit with the highest covariance is then inserted into the network at the end of input phase. In output phase, all the weights connected to the output layer are trained to minimize residual network error. In input and output phases, learning is done using an algorithm for training feed-forward networks, such as QuickProp [4]. Changes of phase occur when covariance maximization or error reduction stagnates in the current phase.

For the experiments presented here, we used a variant of Cascor that has no direct input-output connections (i.e., all signal pathways go through a hidden layer), and which recruits all hidden units onto a single hidden layer, sometimes referred to as Flat Cascor. Cascor training parameters are as follows: learning rate = 0.175 in output phase and 1.0 in input phase, decay = 0.0002 in output phase and 0.0 in input phase, maximum growth factor = 2.0 in both phases, maximum epochs spent in current phase = 100 in both phase, change threshold = 0.01 in output phase and 0.03 in input phase, and patience = 8 in both phases. All simulations used candidate units with a standard sigmoidal activation function.

Learning in cascor occurs in batch: all training patterns are processed before a weight update is performed.

2.1 Task

Experiments presented use a modified encoder task, as used in previous work on word processing [5]. Encoder tasks involve reproducing input patterns on the output units by encoding inputs onto hidden units, then decoding hidden units representations at the outputs. Here, the encoder task is modified such that, on input

vectors longer than the outputs, actual data are presented at three different locations along the input vector. For example, we generate the following three training patterns the word WITH: (input → target output): WITH## → WITH, #WITH# → WITH, and ##WITH → WITH where # are blanks. All experiments use four-letter items (words or random character strings). We used local coding: each letter slot is encoded using 26 binary values indicating the presence and absence of a given letter, in alphabetical order. For instance, presence of letter A is encoded as [1 0 0 ... 0], B as [0 1 0 ... 0], Z as [0 0 … 1]. Blanks are coded using zeros in all positions [0 0 0... 0].

3. Results

3.1 Experiment 1: Generalization to unseen words

We first tested how Cascor networks generalized to unseen words and random strings. We trained networks on sub-sets of real words (previously used in [6]). The list contains 1179 words, and includes information about the frequency of occurrence of words in realistic corpuses of text. Words were selected in decreasing order of frequency (i.e., frequent words selected first) until training exceeded computational resources (which occurred with 120 words). A single network was trained in each condition until perfect accuracy on the train set. Networks were then tested on two sets: (1) 500 random four-character patterns (e.g., JPQM), and (2) words from the word list not used in training. Test items were presented in all three positions. To compute accuracy, the letter corresponding to the most active output in a slot was selected as network response for a letter. A word response was considered correct when all four constituent letter responses were correct. Accuracy was computed as the proportion of correct answers among test patterns.

Accuracy results (see Figure 1) suggest that networks have generalized to words unseen in training better than to random patterns. Thus, Cascor seems to have built a representation of "word-ness" as it learned position invariance, probably due to statistical regularities in words.

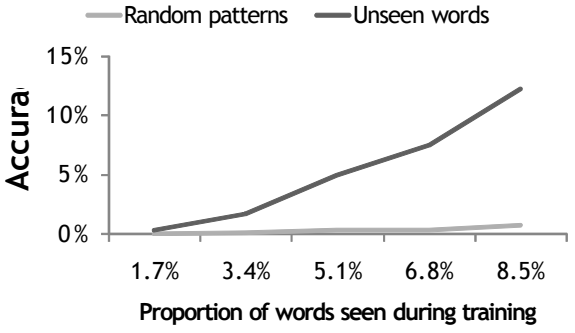

Figure 1 - Accuracy in the modified encoder task in which words are presented in 3 positions

Results about recruited hidden units (see Figure 2) suggest successful data reduction. To take 100 words as an example, Cascor recruited about 60 hidden units to encode 300 patterns (100 words x 3 positions). We also see that additional learning was necessary as new words are presented. Thus, position invariance did not seem to be learned in some abstract, symbolic way, but instead was sensitive to word regularities as well, corroborating the accuracy results. Figure 3 presents the number of epochs for training Cascor networks.

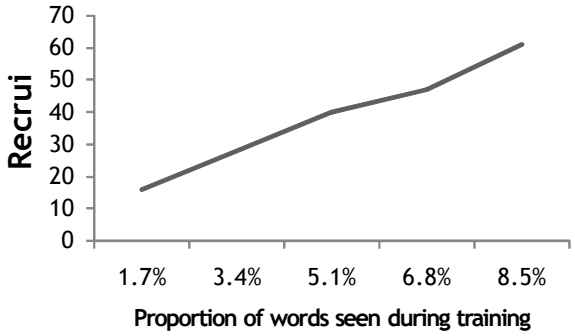

Figure 2 - Number of Cascor recruits necessary to learn the modified encoder task in which words are presented in 3 positions

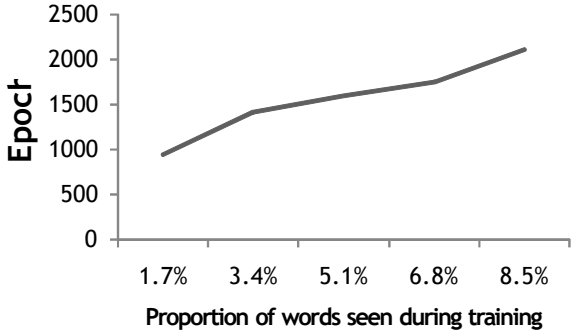

Figure 3 - Number of Cascor epochs necessary to learn the modified encoder task in which words are presented in 3 positions

3.2 Experiment 2: Generalization to unseen positions

Second, we investigated more directly how Cascor networks learned location independence by measuring generalization to known words but unseen or untrained positions while manipulating the number of words used for training. We randomly and uniquely assigned patterns (3 x number of words) to a train and a test set, with a probability of 10% of being a test pattern. For example, ##WITH and #WITH# may be assigned to the training set, and WITH## to the test set. Accuracy results are presented in Figure 4. With a logistic regression analysis, we found a significant intercept (-2.91, $z = -7.6$, $p < .001$) and a positive slope (0.013, $z = 2.7$, $p < .01$) suggesting that accuracy increased with the number of patterns used in training.

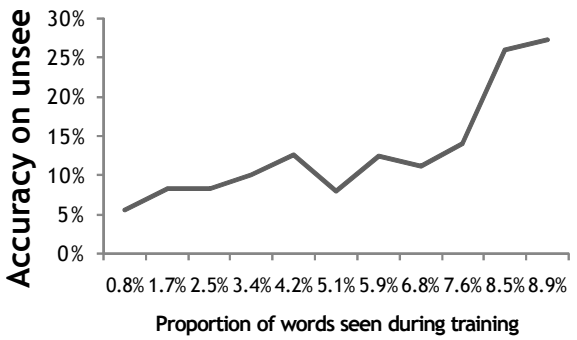

Figure 4 - Mean accuracy (6 networks per condition) on unseen positions as a function of the number of words included in the training set

3.3 Experiment 3: Open bigrams

Finally, we investigated the internal representations built while learning location independence. As argued in the introduction, we expect these representations to be more compact and abstract than the inputs they represent. Open bigrams code two-letters combinations, in such an abstract and largely position-independent fashion. For example, WITH is composed of following open bigrams: WI, WT, WH, IT, IH and TH. Activation of open bigrams can be modulated by distance (contiguous bigrams like WI are more active than non-contiguous bigrams such as IH). An important characteristic of open bigrams is that, while they allow for non-contiguous letter combinations, they preserve letter order. For example, IW is not an open bigram for the word WITH.

	1	2	3	4	5	6	7	8	9	10
Test patterns										
	#	#	#	**X**	**X**	**X**	**X**	#	#	#
Training patterns										
1	A	B	C	**D**	#	#	#	#	#	#
2	#	A	B	**C**	**D**	#	#	#	#	#
3	#	#	A	**B**	**C**	**D**	#	#	#	#
4	#	#	#	**A**	**B**	**C**	**D**	#	#	#
5	#	#	#	#	**A**	**B**	**C**	D	#	#
6	#	#	#	#	#	**A**	**B**	C	D	#
7	#	#	#	#	#	#	**A**	B	C	D

Table 1 – Illustration of training and test data. Xs indicate where the 4 letters of the test strings were presented (always in the center), and # indicate blanks. Each word in the train set is presented in 7 locations. We see that all central locations (slots 4 to 7) are trained on all the letters of the train data.

In experiment 3, training patterns consisted of five strings of four randomly selected letters. As shown in Table 1, training strings were presented in 7 positions and tested in central locations only (slots no. 4, 5, 6 and 7). With this design, letters were trained in all positions where they were tested. Thus, performance differences could not be attributed to certain letter-slot combinations trained more than others, and would more likely be about relationships between letters. Test strings were constructed as combinations of bigrams from words used as inputs. Two binary factors were crossed: (1) direction (forward or backward); (2) letter contiguousness (e.g., contiguous (e.g., AB) or not (e.g., AC)). Table 2 shows the set of test strings with hypothetical strings ABCD and EFGH.

We measured the discrepancy between target and actual output values using a standard SSE measure (the sum of square distances between outputs and targets over all test patterns). An ANOVA on SSE with two repeated factors (direction, letter contiguousness) revealed three significant effects: (1) a main effect of direction, F (1, 19) = 56, $p < .001$ indicating that networks produce less error for forward bigrams (M = 1096) than backwards bigrams (M = 1186); (2) a main effect of letter contiguousness, F (1,19) = 9.3, $p < .01$, showing that networks made less error on contiguous (M = 1128) than non-contiguous (M = 1153) bigrams; and (3) a direction by letter contiguousness interaction, F (1,19) = 7.3, $p < .05$, suggesting networks were less sensitive to letter contiguousness in reversed (inverted) bigrams than in forward bigrams (see Figure 5).

	Forward	Backward
Letter contiguous	ABEF; EFAB; ABFG; FGAB; ABGH; GHAB; BCEF; EFBC; BCFG; FGBC; BCGH; GHBC; CDEF; EFCD; CDFG; FGCD; CDGH; GHCD;	BAFE; FEBA; BAGF; GFBA; BAHG; HGBA; CBFE; FECB; CBGF; GFCB; CBHG; HGCB; DCFE; FEDC; DCGF; GFDC; DCHG; HGDC;
Letter non-Contiguous	ACEG; EGAC; ACEH; EHAC; ACFH; FHAC; ADEG; EGAD; ADEH; EHAD; ADFH; FHAD; BDEG; EGBD; BDEH; EHBD; BDFH; FHBD;	CAGE; GECA; CAHE; HECA; CAHF; HFCA; DAGE; GEDA; DAHE; HEDA; DAHF; HFDA; DBGE; GEDB; DBHE; HEDB; DBHF; HFDB;

Table 2 – Exhaustive set of test patterns for some hypothetical words ABCD and EFGH

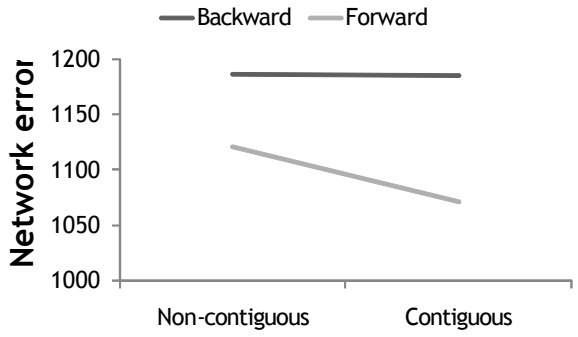

Figure 5 - Average error (SSE) of 20 networks by direction and letter contiguousness

4. Conclusion

Hierarchical cognitive models of printed word processing (e.g., [1]) build increasingly abstract and compact representations in successive layers. We have shown that a connectionist model based on Cascade-Correlation (Cascor) can perform such useful abstractions. Cascor successfully encoded input patterns onto a smaller set of hidden units. Results suggest Cascor learned simultaneously regularities related to word structure ("word-ness", e.g., that certain letters never occur in certain positions) and position (location invariance). We also find the representations built are compatible with the concept of open bigrams: networks make the least error on contiguous, forward strings of two letters, marginally more on non-contiguous forward bigrams, and significantly more on backward or reversed bigrams.

Future models will use more realistic input data. For instance, bitmap images of printed words could be used as more realistic approximations of the information available at the retina.

Acknowledgements

This project was supported by the Agence Nationale de la Recherche (grant no. ANR-06-BLAN-0337).

References

[1] S. Dehaene, L. Cohen, M. Sigman, & F. Vinckier, The neural code for written words: a proposal, *Trends in Cognitive Sciences, 9* (7), 2005, 335-341.

[2] S. E. Fahlman, & C. Lebiere, *The cascade-correlation learning architecture* (In Advances in neural information processing systems 2, D. S. Touretzky (ed.), Los Altos, CA: Morgan Kaufmann, 1990, 524-532).

[3] J. Grainger, & W. J. B. Van Heuven, *Modeling letter position coding in printed word perception* (In P. Bonin (Ed.), The Mental lexicon. New York : Nova Science Publishers, 2003, 1-24).

[4] S. E. Fahlman, *Faster-learning variations on back-propagation: An empirical study* (In T. J. Sejnowski, G. E. Hinton & D. S. Touretzky (Eds.), the Proceedings of the 1988 Connectionist Models Summer School. San Mateo, CA: Morgan Kaufmann, 1988).

[5] R. Shillcock, & P. Monaghan, The computational exploration of visual word recognition in a split model, *Neural Computation, 13*, 2001, 1171-1198.

[6] J. McClelland, & D. E. Rumelhart, *Explorations in parallel distributed processing* (Cambridge, MA: The MIT Press, 1988).