Representational Analysis of Reber Grammar in a Simple Recurrent Network

Patricia Shih

1 Introduction

Numerous everyday tasks are organized into sequences. From the series of left and right turns in navigating from place-to-place, to the combination of words strung together to construct grammatical sentences, these forms of actions are similar in that they all contain inherent sequential structure. Language in particular possesses many implicit rules of this kind. Therefore, it has been suggested that the same basic mechanism for learning sequential dependencies could possibly also form the basis of language acquisition (Rumelhart and McClelland, 1986).

The two dichotomous theories of language development posit that either language ability can emerge from entirely basic neural mechanisms, or alternatively, it is an innate, specialized faculty of the human brain (Pinker, 2007). In the latter case, models of language processing have been built upon domain-specific modules, composed of functionally discrete linguistic components (e.g., syntax, morphemes, phonemes, and lexemes). An underlying assumption is that the computations performed by the language modules are uniquely specified by the genetic architecture, and therefore a less specific model would not be able to develop the same linguistic representations. However, evidence from computational modeling has begun to generate support for the former case arguing for basic mechanisms. At least one candidate architecture, namely the simple recurrent network (SRN), has demonstrated the ability to extract abstract linguistic structures.

The simple recurrent network is composed of at least three layers: an input layer, hidden layer and a context layer. The key “recurrent” function in this network is realized by, first, copying the current activation state of the hidden layer to the context layer, and then having the context layer provide the representation of the previous state back to the hidden layer in the next time step. The end result is a network with a short memory of preceding events, which can be used to resolve tasks with temporal dependencies. A simpler task of predicting the next item in a first-order sequence (i.e., containing no dependencies) can be solved with just a two-layer network consisting of one input and one hidden layer. However, the correct response for a given input from a sequence that has higher-order dependencies could depend on any number of previous items, of which a basic two-layer network would not be able to keep track. In contrast, the recurrent network has knowledge of preceding items in a sequence, which can be used to disambiguate the input activity by associating it with a distinct background of contextual activity.

The basic three-layer design of the SRN was first introduced by Elman (1990). In this seminal paper, Elman demonstrated the ability of his network to extract latent linguistic units without explicit instruction. In one simulation using a letter-prediction task, the author presented the network with one letter at a time from a single stream of concatenated sentences containing no spaces and trained it to predict the next letter in the sequence. Importantly, the letter-in-word stream did not delineate where one word ended and another word started. Nevertheless, the pattern of errors committed by a trained network indicated the presence of implicit morphological structure embedded in the order of the sequence. Specifically, when presented with the same input stream as the one used for training, the network systematically made large errors at the beginning of a word and gradually decreased its errors for letters toward the end of learned words. Further, assuming that the network can indeed develop functional symbolic representations of whole words, the SRN was then trained on a similar task, but this time using individual words as the single input pattern. Cluster analysis on the activations in the hidden layer revealed superordinate representations of lexical categories (i.e., nouns and verbs), as well as subordinate lexical semantic subcategories. Thus taken together, these simulation results from letter and word prediction tasks provide some strong evidence that the SRN architecture is sufficient for learning basic abstract concepts in language.

While Elman (1990) convincingly demonstrated that his SRN was capable of developing simple linguistic representations, the nature of these representations had not been specifically studied. Since the networks were trained using sentences generated from a natural language (English), which contains thousands and thousands of words and a complex grammatical system, a comprehensive study of the abstract internal representations associated with those sequence tasks would have been difficult. Therefore, Servan-Schreiber et al. (1991) probed the nature of these representations by training SRNs on another letter prediction task using sequences generated from a well-defined artificial grammar: the Reber finite-state automaton. The Reber transition network is comparatively more simplistic than natural grammars; having a small, fixed number of nodes and directed-edges allowed for more specific inferences to be made from the representational analysis. In general, the study found that the activity of the hidden units layer contained node representations, although some redundancy was observed with 15 units compared to three. Moreover, context activity in the network indicated a gradual progression from zeroth-order to higher-order sequential representations as it learned the task.

Notably, many of the SRN representational analyses performed by Elman (1990) and Servan-Schreiber et al. (1991) employed hierarchical cluster analysis on activity from just the hidden units layer. Thus, context representations were never directly examined in these papers, but rather, they were inferred by interpreting the effects of context on the responses of the hidden units. The rationale for focusing on the hidden layer was not explicitly stated. However, it might be possible that cluster plots for the context layer is comparatively more difficult to interpret, as sometimes the key factors shared among members that were grouped into the same clusters might not be easily definable – especially when increasingly abstract categories are involved.

Another approach to examining internal representations is to use multivariate pattern classification analysis (MVPA) to determine the features that contribute the most to a particular class. Classification of the patterns of activity can be performed for each layer of the SRN and on an individual trial-by-trial basis. One benefit of this approach is that it can answer questions related to the consistency of representations across predicted classes, as well as highlight prediction errors and possibly indicate other kinds of representation that might be more appropriate. Thus representational analysis with MVPA can capture relationships between predicted and actual class labels, while cluster-based methods is slightly more exploratory with no clear way to approach interpreting the cluster results.

The objective of the present study is therefore to use MVPA to examine the internal representations of Reber grammar in both the hidden and context layers of the SRN. Based on past research, the expectation is to be able to successfully classify by nodes in the hidden layer after training, which should reflect distinct conjunctions of letter and context. Without the function of the context layer, the network should not be able to disambiguate identical letters at different nodes. Furthermore, since the context layer contributes to this differentiation by providing the hidden layer with a memory of prior activation states, internal representations of context units is hypothesized to reflect the activity of preceding nodes in a sequence.

2 Methods

2.1 Task

The network model was trained to predict letters of sequences that follow the Reber grammar (Figure 1) (Reber, 1976). Using this finite-state automaton, four hundred sequences were generated, with an average length of 6.9 characters, resulting in approximately 2760 individual trials. The construction of grammatical sequences always starts at node 0 with letter B and must end at node 5, choosing letter E. At each node, two letter transitions are

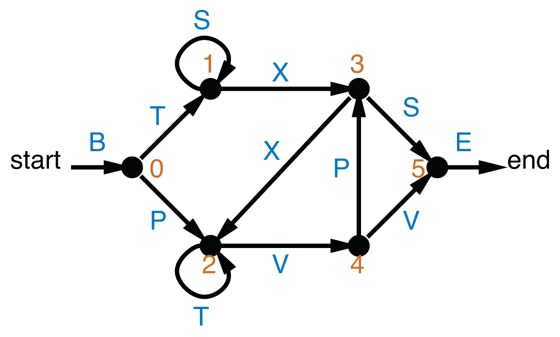

Figure 1: Reber finite-state automaton. Construction of grammatical Reber sequences start at node 0 with letter B and end at node 5, choosing the letter E. Each letter transition has a 0.5 transition probability and must follow the path specified by the structure.

possible, with a transition probability of 0.5. In addition, five of seven letters are repeated at a second node in the Reber network and some nodes have recurrent transition paths (e.g., T and S loops at node 1 and node 2, or poly-nodal recurrent paths from node 2 to 4 to 3 and back to 2). The fact that letters can repeat at different positions in a string and that the same letter can precede different transition states indicate that predicting the next letter in the sequence will require the ability to learn higher-order sequential dependencies.

2.2 Network model and training procedure

2.2.1 Simple recurrent network architecture

A simple recurrent network with an underlying Leabra framework (O'Reilly and Munakata, 2000) was constructed with one 1x6-units input layer, one identically-sized output layer, one 5x6-units hidden layer, and one 5x6-units context layer (Figure 2). Three parameters were changed from the standard recurrent network: 1) the degree of copying of the hidden layer state to the context layer (fm_hid = 0.9), 2) the amount of previous state information retained in the context layer from the previous trial (fm_prv = 0.1), and the learning rate (lrate = 0.001) during training. The learning rate was lowered from the original (0.1) because the average sum of squared errors (SSE) during training indicated that the network was over-

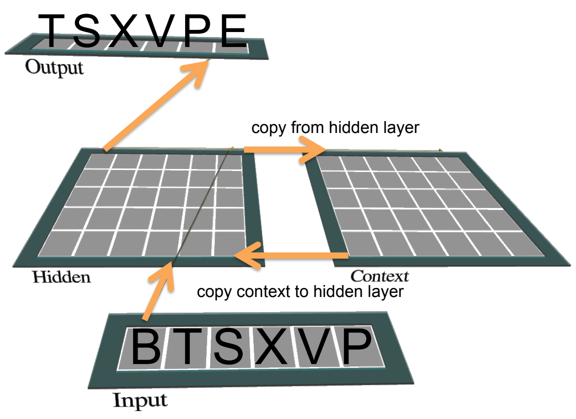

Figure 2: A simple recurrent network (SRN) architecture for learning sequences of letters. Letter representation from input units is sent to the hidden layer and this representation is copied to the context layer. At the next time step, the context layer copies the previous state back to the hidden layer to supply context of previous letters in the sequence to the hidden layer.

fitting to the sequences presented in trials early in the epoch.

2.2.2 Network training procedure

Networks underwent 200 training epochs to learn to predict the next letter in a Reber string. Two hundred epochs was more than sufficient for the minimization and stabilization of SSE in training. The final average SSE hovered around 0.34 (Figure 3). As expected, trained networks had an overall average SSE much lower than that of the untrained control networks with fixed random weights (Figure 4). Of note, the consistently high average SSE observed from testing the naïve networks on the sequences (with learning turned off) reaffirms that new sets of random seeds were successfully initialized for control networks. To further ensure that learning was not accidentally occurring during testing, the first control network underwent 5 epochs of testing to check for any early indications of learning-related decreases in SSE. A total of ten trained networks and ten untrained control networks were tested on the Reber sequences. Five additional untrained networks with the copy function disabled (fm_hid = 0) were used as another set of control comparisons.

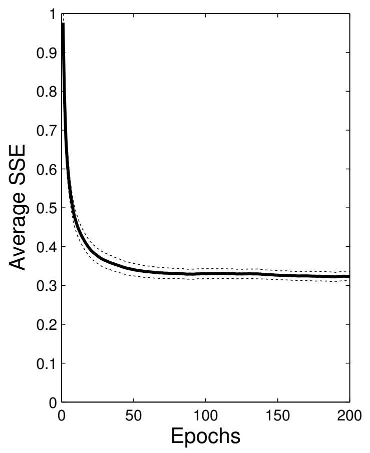

Figure 3: Training of networks stabilized consistently at around 0.34 sum of squared errors (SSE). Plot shows the mean SSE after each epoch during training, over 200 epochs, averaged across 10 networks and with 95% confidence intervals indicated by dotted lines.

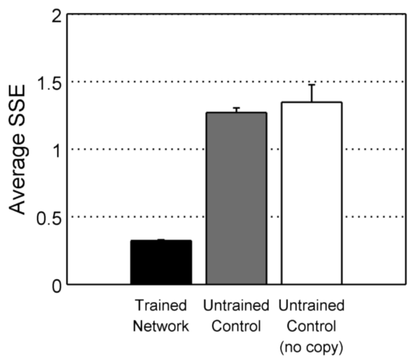

Figure 4: Trained networks with lower sum of squared errors compared to control networks. Control networks were initialized with new random seeds and tested on Reber sequences without prior training. Sample size was 10 each for the first two groups of train and untrained networks. Sample size was 5 for the last untrained control group with the copy function of the hidden layer disabled ("no copy").

2.3 Representational analysis of SRN activity

2.3.1 Pattern classification approach

Activity while the network performed the letter prediction task was recorded from hidden and context units. For all trained networks, activity was obtained from the last (200th) epoch during training. These activation patterns were used to investigate internal representations in the SRN. To remove overall mean effects from the analysis, unit-wise demeaning was performed by subtracting the mean activation across time from each time point.

Training of classifiers was performed for each node by simplifying the classification problem into two-class discrimination, i.e., a one-versus-rest approach to multi-class data. Logistic regression analysis was implemented as the main technique for defining decision boundaries, since it is appropriate for binary classifications and it provides a convenient likelihood statistic describing the probability of classification success. Training was performed on half of the recorded data from each layer, selected equally at random. The remaining half was saved for performance testing of the trained classifiers. As there were 30 units in each layer, each classifier consisted of 30 features.

To avoid unnecessary bias from variable class size, the number of trials for each class (i.e., node) was fixed to the lowest number of trials available across all nodes. For certain nodes with many more trials than this fixed number, trials were selected uniformly at random. Note, however, that the frequency of specific letter transitions were not all equal. In addition, the size of the null class, containing all other trials that were not in the training class, was fixed to two times the size of the training class. Twelve independent null class sets were generated via random sampling with replacement from the full set of trials (minus the training class). The final classifier for the trained class was obtained by averaging the weights obtained from training against the 12 sets of null classes.

2.3.2 Classifier training algorithm

Two training algorithms using logistic regression were evaluated: 1) optimization of weights using maximum *a posteriori* estimation, from Princeton's Matlab MVPA Toolbox (Detre et al., 2006) and 2) optimization with maximum likelihood estimation and descending learning rate. Although the general pattern of results were similar, numerically the second method outperformed the first for nearly all classes examined — accessed by comparing standard measures of performance: classification accuracy, specificity, sensitivity. Therefore, only results from the second algorithm will be presented.

2.3.3 Assessment of representations in the SRN

Trial-by-trial activity within the hidden and context layers were separated into distinct classes based on its locus in the finite-state automaton. The performance of the classification analysis was assessed by constructing a confusion matrix and numerically examining accuracy (for correct identifications and rejections), sensitivity (correct identifications over the total number of true positives and false negatives), and specificity (correct rejections over the total number of true negatives and false positives) of each classifier. Further, test trials were also subcategorized by the individual letter transitions of each node, so that the classification of each trial type can be examined and inferences about specific representation can be made.

Additionally, for a quick comparison between representational analysis with MVPA versus hierarchical cluster analysis (i.e., the more common approach), clustering was performed on the average activation pattern of each letter transition in the set of trained SRNs. Linkage trees were constructed by grouping the nearest euclidian neighbors. However, note that this is only a cursory comparison between the two methods, and thus the choice of clustering letter transition activations average over all sequences differs from the original approach taken by Servan-Schreiber et al. (1991), which clustered the activation patterns of transitions in individual sequences.

3 Results

3.1 Representations in the hidden layer

Performance of the node classifiers on the hidden layer test set was relatively high for both trained and untrained networks (Figure 5A and 6A-B). Only minor differences were observed between the two. For both, the classifiers showed high sensitivity and specificity across tested items – i.e., highly sensitive in detecting transition trials associated with a particular node, as well as specific to correctly discriminate against trials associated with similar letters, but at other nodes (Figure 5B, top row). Interestingly, the confusion matrix results for trained and untrained networks suggest that latent node representations are already present in the sequence structure and do not necessarily require task training to emerge.

3.2 Representations in the context layer

Representations of nodes in the context layer, in comparison, appeared less consistent, and the algorithm failed to train a classifier with sufficient sensitivity for node 2 (Figure 5A, bottom row, and Figure 6A-B). To examine classification errors at a greater resolution, prediction accuracy was compared at the level of the individual letter transition trials at each node (Figure 5B). In addition, for ease of visualization, transitions were clustered and resorted in the confusion matrix for the context layer so that similar patterns of classification performance existed closer together along the y-axis. Although some node representations were present in the activity of the context layer, the results suggest that another higher-order representation might be present. Generally, classification errors occurred often in transition trials of preceding nodes, which would suggest a representation of the previous nodes in a sequence. However, not all transitions associated with a particular preceding node were classified in the same way and some unexpected classification of transition trials from second order nodes were also present (discussed later below).

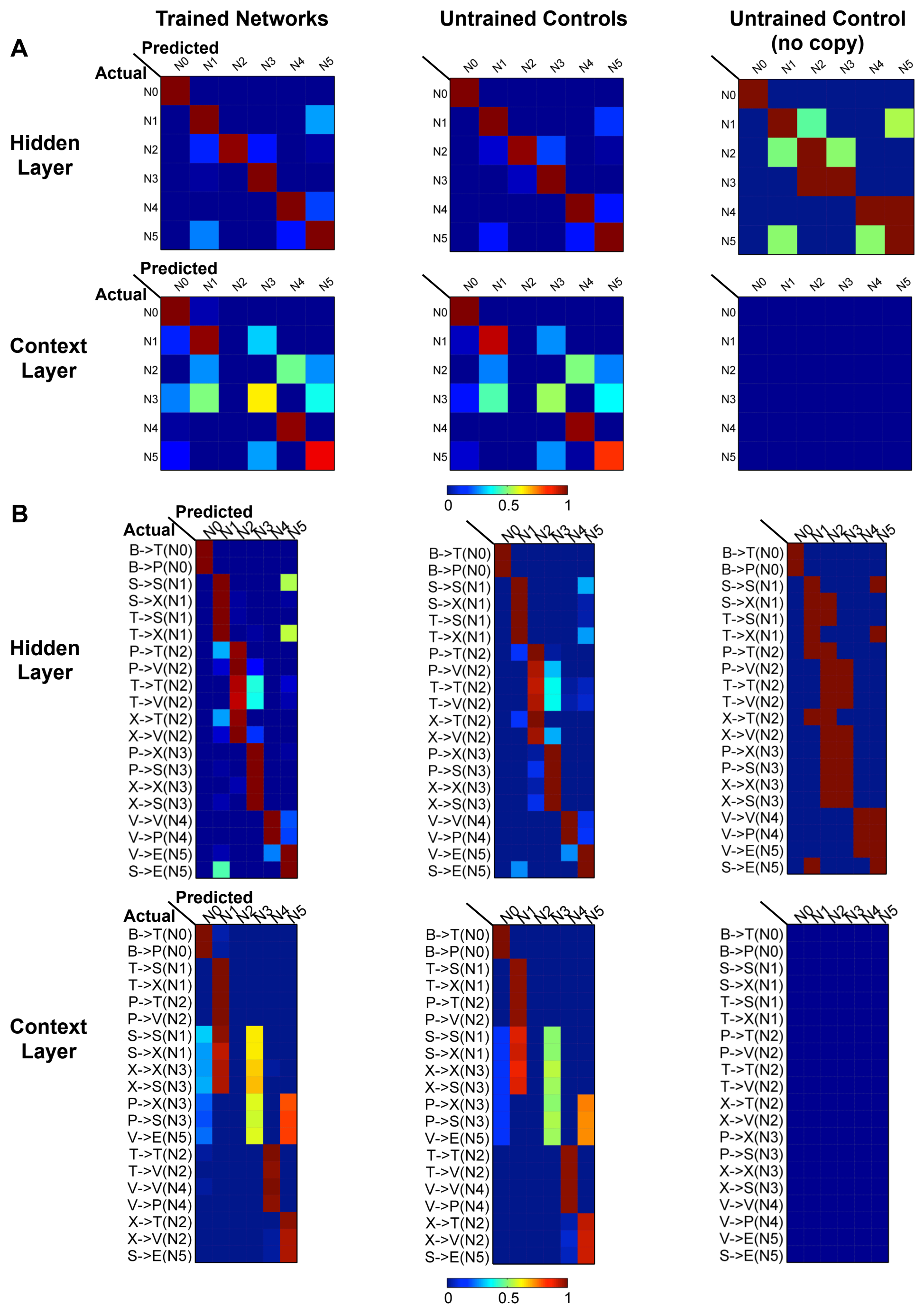

Figure 5: Reber grammar-generated sequences contain latent nodal information. (A) Confusion matrices for node classifiers show relatively accurate classification in the hidden layer. However, representation of nodes in the context layer were less consistent, particularly for node 2. (B) Subdividing trials into individual letter transitions shows that the node classifiers of the hidden layer were often able to correctly discriminate against any similar letters of other nodes; however, classifiers trained from networks with context layer function disabled produced more type I errors in comparison. Incorrect context layer classifications are presented in a resorted confusion matrix, with similar patterns of classification success grouped together in adjacent rows. The heat map can take on values from 0 (none of the actual data fit the predicted classifier) to 1 (all of the test data fit the predicted).

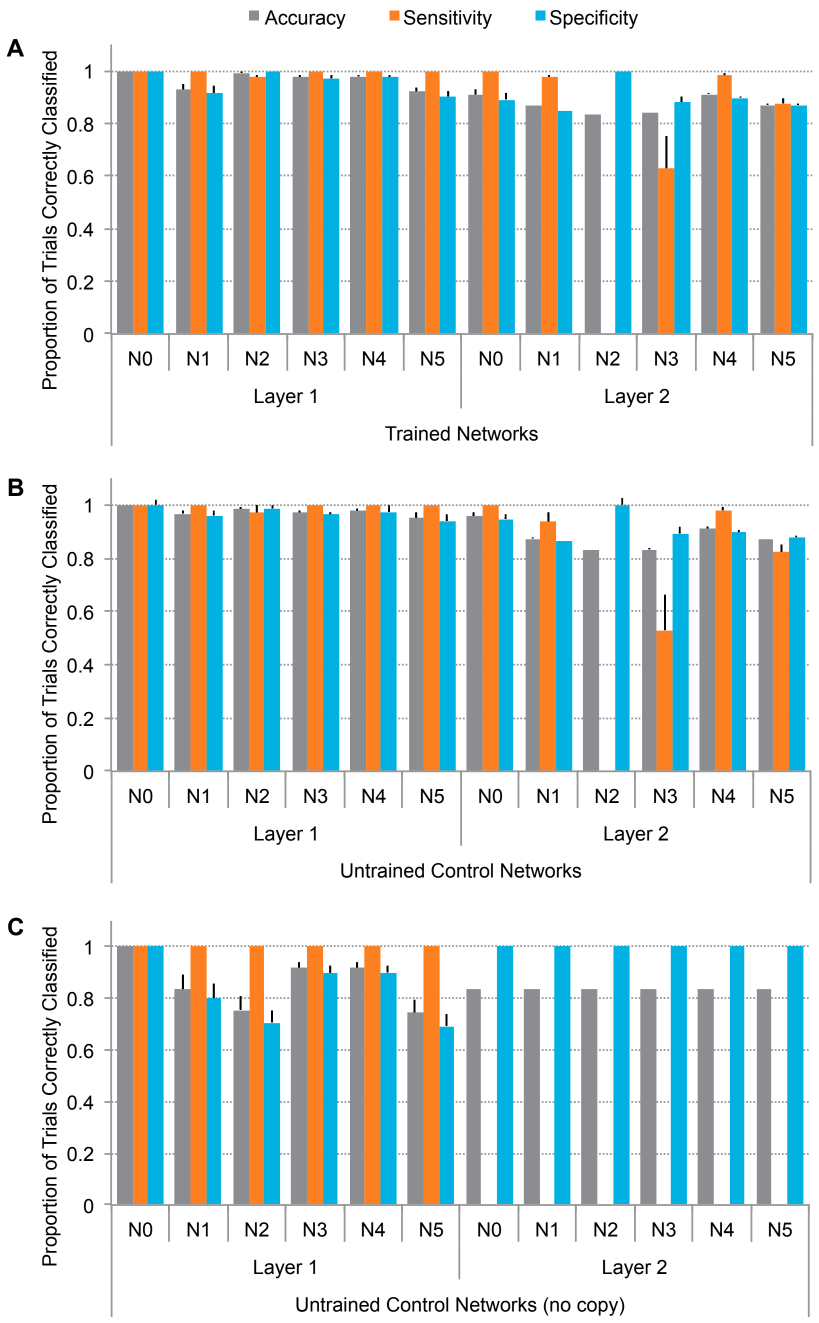

Figure 6: Classifier performance. Accuracy, sensitivity, and specificity of each of the node classifiers in (A) Reber grammar trained networks, (B) untrained control networks (with context layer intact), and (C) untrained control networks with copy from hidden layer function removed. Layer 1 = hidden layer; Layer 2 = context layer. Error bars indicate SEM*2.

Table 1: First-order letter transitions

	B	T	S	X	V	P	E
B		*				*	
T		*	*	*	*		
S			*	*			*
X		*	*	*	*		
V			*	*			*
P		*	*	*	*		
E							

3.3 Representations in untrained control networks

Since there were only minor differences in the pattern classification analysis between trained and untrained networks (with trained networks showing only subtle improvements in classification), manipulation of an additional model parameter was implemented to prevent the network from being able to detect any higher-order sequential dependency rules. This was achieved by disabling the copying of hidden layer activation states into the context layer. Classifiers trained from the hidden layer activity in networks providing no contextual background produced more type I errors in comparison (Figure 5A-B, 3rd column). Increases in false positive decisions appear to be a result of overlapping representations caused by constraining activation in the hidden layer to no higher than first-order structure (i.e., input and output; see overlapping letter transitions in Table 1). Nodes associated with a unique set of letters still showed high classification performance (node 0), while classifiers for nodes that shared the same set of letters often confused transition trials from other nodes (e.g., compare shared letters X & P of nodes 2 and 3 in Figure 1 with classification results in Figure 5A). Classification of the context layer is irrelevant, since its operation has been effectively disabled without a copy of the hidden layer activation state.

3.4 Comparison of MVPA and hierarchical cluster analysis

Hierarchical clustering found six main clusters of letter-transition activation patterns (Figure 7), which coincidentally is the same number as the total number of nodes in the Reber

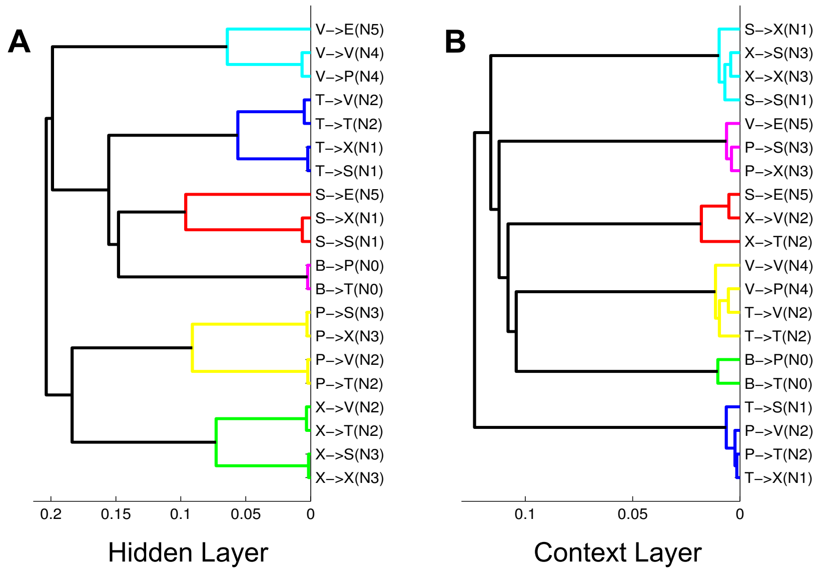

Figure 7: Hierarchical cluster analysis of (A) hidden layer and (B) context layer activations delineated six main clusters of task representations.

grammar. However, neither the activity of the hidden nor the context units indicate node-based organization in this analysis. While the hidden layer was clearly clustered based on the preceding letter in a transition, subgroups within these main clusters did at least show separation of letter transitions from differing nodes. In comparison, clustering of context unit activity were not based on the first letter in a transition; to some extent, they resembled context layer results from MVPA (c.f. Figure 5). In addition, the context representations from the hierarchical cluster analysis indicated a nested pattern of organization, which is a reasonable finding, since some letter transitions will precede a greater number of other letter transitions due to their position in the Reber grammar.

4 Discussion

The present study examined internal representations in SRNs using MVPA on activity recorded from the hidden and context layer. Classifiers were trained to distinguish among activation patterns at different nodes in the Reber finite-state transition grammar. As predicted, representations of distinct nodes were observed in the hidden layer and consisted of unique letter transitions from the respective nodes. However, the results from the context layer did not confirm the original hypothesis taken from Servan-Schreiber et al. (1991). Instead, context at each node was represented as a pattern of activity that reflected all letter transitions that could be possible under the same context.

4.1 Node representations in the hidden layer

Distinct representations of nodes in the hidden layer from the conjunction of input and context events is consistent with previous reports by Servan-Schreiber et al. (1991). However, puzzlingly, node representation in this SRN did not require prior exposure to the task. Both sets of classifiers, either trained from networks that had learned the Reber grammar or another that had been naïve to the task, exhibited comparable classification results. Specifically, the hidden layer of the untrained control network revealed latent node structure that appeared to be intrinsic to the input.

The detection of higher-order structure, in general, is made possible through recurrent connections via the context layer. The function of context units is to project prior states back to the hidden layer, thus naturally dissociating the input activity at different time points using different backgrounds of contextual activity. Therefore, the context layer is the key component of the SRN that gives it the capability to detect and learn higher-order sequential dependencies when lower-order rules are insufficient for determining the correct output response. When the copy operation of the hidden layer to context layer was disabled, thus forcing the network to only be able to perform simple input-output contingencies, node representation was no longer detectable. In these constrained networks, the same letter input had multiple associations, and therefore, the classifiers trained from these networks were not able to differentiate the representations of identical letters from different nodes.

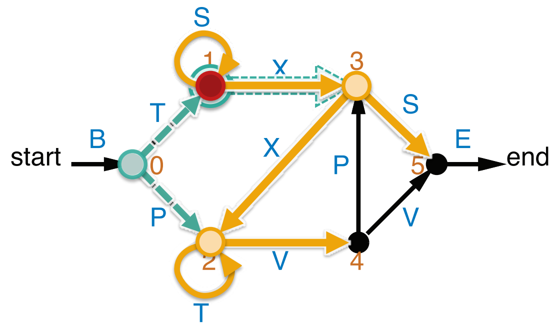

Figure 8: Context representation for node 1. Red circle indicates the location of node 1 in the Reber grammar. Teal color indicates preceding nodes and transitions, while the orange color indicates the nodes and transitions associated with context representation at node 1. Compare with classification results from Figure 5B.

4.2 Higher-order abstract representations in the context layer

What is represented in the context layer that allows each node to have relatively unique representations in the hidden layer? The classification results for the context layer suggest a more abstract relationship than what was originally hypothesized. Representations in the context layer are composed of a copy of the hidden layer activation state at the previous time step and diminishing traces of prior context states (adjustable with the fm_prv parameter). The original hypothesis took a more literal interpretation of "previous activation state" and generated predictions of node misclassification for letter transitions trials immediately following the preceding nodes, as those letter transitions would presumably share similar patterns of activation history. For example, at node 1, the context layer might have been predicted to hold a representation of the activation pattern of the hidden layer at node 0 (combined with node 1 due to the recurrent S_1-loop). Under these predictions, classification errors should have then occurred for node 0 transitions (see teal paths in Figure 8). However, inconsistencies between the expectations set by the original hypothesis and the actual findings dictate that the prior notion held about state representations of preceding nodes needs to be redefined.

Rather than directly representing the preceding letters and nodes per se, representations in the context layer could implicitly contain information about the past transitions in a sequence by reflecting predictions of valid transitions. Under this refined hypothesis, classifiers of the context layer is expected to confuse potential letter transition trials that are under similar contexts — specifically, second-order transitions that come after the same preceding node. The internal representation of context should therefore be a representation of potential transitions – in contrast to past transitions – that follow the same node in the grammar network.

Returning to the representation at node 1 as an example, one would expect the node 1 classifier to classify any letter transitions from nodes immediately succeeding node 0 and node 1 (i.e., letters associated with the node transitions 0→1, 0→2 and 1→3). In particular, classification "error" would be seen for $P_2 \rightarrow T_2$, $P_2 \rightarrow V_4$, $X_3 \rightarrow S_5$, and $X_3 \rightarrow X_2$ and correct classification would include the transitions $S_1 \rightarrow S_1$, $S_1 \rightarrow X_3$, $T_1 \rightarrow S_1$, and $T_1 \rightarrow X_3$. Representation at node 1 would not include transitions such as $T_2 \rightarrow T_2$ and $T_2 \rightarrow V_4$ at node 2, since the letter T_2 is not a valid transition from preceding node 0. Indeed, these are the exact patterns of classification successes and failures observed in the context layer (Figure 5B). Thus the notion of "context" appears to be represented by the affordance of actions/ transitions rather than the memory of previous transitions per se.

4.3 Limitations

A few limitations should be stated that prevented any confident interpretations from being made. First, although the frequency of trials at each node was made equal across all nodes, the frequencies of letter and letter-transitions were variable and thus might have provided some diagnostic node information to the network. Since the Leabra framework implements hebbian learning (Hebb, 2002) in its architecture, differences in frequency could fathomably affect statistical learning and show bias toward the more commonly occurring letter transitions in a sequence.

Additionally, representations in the context layer and subsequently the hidden layer can be manipulated as a function of the parameters that specify the amount to copy from the hidden layer to the context layer and the amount of information to keep in the context layer for the next trial. It is possible for the context layer to form even higher, higher-order representations. Notably, the classifier for node 2 showed the poorest performance compared to the rest. Training of node 2 was particularly difficult, since its position in the Reber transition network has hub-like properties. At node 2, lower-order context representation could be shared across nearly all nodes, which would make differentiation into distinct context events difficult. However, if the network were to acquire additional higher-order abstract representations, such as one that could keep track of sequence length, context at node 2 would be easier to separate.

Finally (and critically), there was a lack of a large difference between trained and untrained networks with everything else being equal. This might be a particularly surprising result, given that network training should affect sending and receiving weights and in turn directly affect unit activity in each layer. Furthermore, one of the main findings from Servan-Schreiber et al. (1991) are related to the effects of learning on internal representations in the SRN. One possible explanation for this discrepancy could be related to the sizes of the hidden and context layers. Servan and colleagues examined representations using three and 15 hidden units, whereas in the present study, the hidden layer contained 30 units, which provides the network with more resources and could allow for each of the 20 letter transitions in the grammar to be represented by a single unit. Therefore, it may be possible that with more units than lower-order representations, the network is able to immediately form representations of the higher-order structure in a bottom-up fashion rather than having to progressively acquire them through shifting weights around during learning.

4.4 Summary and conclusions

Using MVPA, unexpected internal representations that might not have been easily inferred from results obtainable from typical cluster-based methods were detected in the SRN. Representational analysis on the hidden layer revealed individual node representations, but surprisingly, the results also presently suggest that node representation is already detectable from the patterns of activity in untrained networks with larger hidden layers. The representations in the hidden layer were composed of unique letter transitions that have been dissociated from their overlapping lower-order structures. This conjunctive representation of input and context is made possible through the important functional contributions of the context units, which provides an elegant solution for learning sequential dependencies. But contrary to the original hypothesis which expected context units to encode the preceding letter(s) in a sequence, context representations appear to be even more abstract in nature. The internal representations at the nodes of the Reber grammar reflected all possible letter transitions from any node directly associated with the preceding node – that is, all letter transitions considered as under the same context.

Lastly, the results in the context layer could possibly point to a rudimentary mechanism for representing other abstract concepts such as action goals and plans. However, in regards to biological feasibility, even though evidence for SRNs has been provided for functions such as early language acquisition (Elman, 1990) and implicit sequence learning (Botvinick and Plaut, 2004), a neural correlate for the copy operation has yet to be found. Even so, it may nevertheless be possible for a similar basic architecture operating under the same general recurrent-connectivity principles to be implemented by the brain.

References

Botvinick, M. and Plaut, D. C. (2004). Doing without schema hierarchies: a recurrent connectionist approach to normal and impaired routine sequential action. *Psychol Rev*, 111(2):395–429.

Detre, G., Polyn, S. M., Moore, C., Natu, V., Singer, B., Cohen, J., Haxby, J. V., and Norman, K. A. (2006). The multi-voxel pattern analysis (mvpa) toolbox. In *Poster presented at the Annual Meeting of the Organization for Human Brain Mapping (Florence, Italy). Available at: http://www. csbmb. princeton. edu/mvpa*.

Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, 14(2):179–211.

Hebb, D. O. (2002). *The organization of behavior: A neuropsychological theory*. Lawrence Erlbaum.

O'Reilly, R. C. and Munakata, Y. (2000). *Computational explorations in cognitive neuroscience: Understanding the mind by simulating the brain*. MIT press.

Pinker, S. (2007). *The language instinct: How the mind creates language*. Harper Perennial Modern Classics.

Reber, A. S. (1976). Implicit learning of synthetic languages: The role of instructional set. *Journal of Experimental Psychology: Human Learning and Memory*, 2:88–94.

Rumelhart, D. E. and McClelland, J. L. (1986). Parallel distributed processing: explorations in the microstructure of cognition. volume 1. foundations.

Servan-Schreiber, D., Cleeremans, A., and McClelland, J. L. (1991). Graded state machines: The representation of temporal contingencies in simple recurrent networks. *Machine Learning*, 7(2):161–193.