On the Importance of Sluggish State Memory for Learning Long Term Dependency

Jonathan A. Tepper[1], Mahmud S. Shertil, and Heather M. Powell

School of Science and Technology, Nottingham Trent University, Burton Street,Nottingham NG1 4BU, UK

Elsevier use only: Received date here; revised date here; accepted date here

Abstract

The vanishing gradients problem inherent in Simple Recurrent Networks (SRN) trained with back-propagation, has led to a significant shift towards the use of Long Short-term Memory (LSTM) and Echo State Networks (ESN), which overcome this problem through either second order error-carousel schemes or different learning algorithms respectively.

This paper re-opens the case for SRN-based approaches, by considering a variant, the Multi-recurrent Network (MRN). We show that memory units embedded within its architecture can ameliorate against the vanishing gradient problem, by providing variable sensitivity to recent and more historic information through layer- and self-recurrent links with varied weights, to form a so-called sluggish state-based memory.

We demonstrate that an MRN, optimised with noise injection, is able to learn the long term dependency within a complex grammar induction task, significantly outperforming the SRN, NARX and ESN. Analysis of the internal representations of the networks, reveals that sluggish state-based representations of the MRN are best able to latch on to critical temporal dependencies spanning variable time delays, to maintain distinct and stable representations of *all* underlying grammar states. Surprisingly, the ESN was unable to fully learn the dependency problem, suggesting the major shift towards this class of models may be premature.

Keywords: Simple Recurrent Networks; Vanishing Gradient Problem; Echo State Network; Grammar Prediction Task; Sluggish state space; Internal Representation.

[1] Corresponding author: J. Tepper, E-mail address: Jonathan.Tepper@ntu.ac.uk, Tel: +44 (0)115 8488363.

Acknowledgements: We are very grateful to the Libyan Bureau in London, UK for funding this research.

1. Introduction

Recent studies have demonstrated that the ability to learn nonlinear temporal dynamic behaviour is a significant factor in the solution of numerous complex problem-solving tasks, such as those found in many practical problem domains e.g. natural language processing [1, 2, 41, 42, 43], speech processing [3] and financial modelling [4, 5]. Recurrent neural networks (RNNs) are a class of connectionist network whose computational neurons produce activations based on the activation history of the network [6]. RNNs have nonlinear dynamics, allowing them to perform in a highly complex manner; network activations from previous time steps are fed back as input into the RNN at future time steps. In theory, the states of the hidden units can store data through time in the form of a distributed representation, and this can be used many time steps later to predict subsequent input vectors [7]. This particular characteristic distinguishes them from their feedforward counterpart (the Multi-layer Perceptron, MLP) and enables them to act as vector-sequence transducers [6].

Although there are numerous connectionist techniques for processing temporal information, historically, the most widely used RNN has been the Simple Recurrent Network (SRN) [1]. The SRN is a state-based model, similar in complexity to a Hidden Markov Model, and represents sequences of information as internal states of neural activation [8]. The SRN has proven remarkably useful for temporal problem domains such as natural language processing, and in particular, learning to model regular and simple context-free languages. Moreover, much research has been conducted to investigate the temporal processing ability of SRNs [8–15]. It has been shown that in some cases, the SRN and its variants are unable to learn time lagged information (dependencies) exceeding as few as 5 to 10 discrete time steps between relevant input events and target signals [18]. This is most likely due to their use of gradient descent learning where the gradient of the total output error, from previous input observations, vanishes quickly as the time lag between relevant inputs and errors increases [17]. SRNs have also been severely criticised for their lack of ability to model the combinatorial systematicity of human language [16, 36]. This, however, has cogently been refuted by Christiansen & MacDonald [47] who demonstrate that the SRN is able to make non-local generalisations based on the structural

regularities found in the training corpus [3] and appropriate constituent-based generalisations, providing further support for non-parametric usage-based models of language [48].

Researchers have investigated various architectural configurations to enhance the memory capacity of SRNs. For example, Ulbricht [19] introduced the Multi-Recurrent Network (MRN), which utilises variable-length memory banks with variable strength recurrent and self-recurrent links, to form a so-called sluggish state-based memory mechanism. The integration of variable state activations with the replication of state nodes, to form memory banks for representing temporal dependency, is a particularly distinctive feature, relative to other models in the SRN family. Dorffner [20] states that these sluggish state spaces can exploit the information from both recent time steps and the distant past to form a longer averaged history, and that this can help to solve long term dependency problems. The MRN has been successfully used to solve a number of complex prediction problems, yielding very competitive results over traditional SRNs, and has also faired competitively with Kernel methods [5, 19]. These published results justify the additional connections and resulting complexity of the MRN. Another approach which utilises additional memory units within the architecture to overcome the vanishing gradient issue, is the Nonlinear AutoRegressive model process with eXogenous input (NARX) network, introduced by Lin et al [34]. Unlike the MRN, the NARX does not utilise past state information; it was introduced to process temporal dependencies, primarily within the continuous domain. The NARX network is essentially an RNN. It contains feedback from the output layer and the input layer to create a context layer. This context layer represents temporal information explicitly, in the form of a shift-register of the previous activations. NARX networks therefore overcome the limitations of SRNs for encoding temporal dependency, by associating nodes with temporal information rather than state activations.

Despite the promise shown by the MRN and NARX networks over SRNs, there has been a strong shift away from the traditional SRN family of networks and towards more complex second order RNNs and those with different learning regimes. One such innovation is the Long Short-Term Memory (LSTM) introduced by Hochreiter & Schmidhuber [37] and developed further by Gers et al [21]. This is a second-order RNN that consists of multiple recurrently connected subnets, called memory blocks. Each block contains a set of internal units having activations controlled by three multiplicative units (input gate, forget gate and output gate). This block-based mechanism enables an error carousel to be formed which enables the LSTM to latch on to appropriate error information. The LSTM is considered to provide state-of-the-art performance in numerous temporal modelling tasks, however without appropriate external

resets, internal unit values can grow uncontrollably, creating instability. Ad hoc reset methods may therefore be required which adds to the complication of the design process.

Another type of RNN aimed at resolving the long temporal dependency problem is the Echo-state Network (ESN). The ESN has again exhibited state-of-the-art performance. It is similar in architecture to the SRN but it has an entirely different learning mechanism and so does not suffer from the vanishing or exploding gradient problem [22]. Training is reduced to a one-shot simple linear regression task applied to the output layer weights. ESNs have been applied with varying success to numerous problem domains such as behaviour classification, natural language processing [23, 42, 43, 44], and speech recognition [24]. ESNs are gaining particular popularity with those researchers seeking biologically plausible models of language and memory. For example, Dominey [42] used ESN-like models to better capture the principles of neurophysiology and address the issue identified by Friederici [45] to explore the role of subcortical structures in language processing. In particular, Dominey suggested '*corticostriatal plasticity plays a role in implementing the structural mapping processes required for assignment of open-class elements to their appropriate thematic role*' and both Dominey [42] and Hinaut & Dominey [43] therefore sought to apply ESNs to implement a mechanism for the real-time parallel processing of conceptual and grammatical structures. Indeed, Pascanu & Jaeger [44] recognise that Dominey's earlier work in cognitive neuroscience with the development of the Temporal Recurrent Network (TRN) [41], independently discovered the reservoir principle that underpins ESNs. Although state-of-the-art performance has been reported with ESNs for the iterated prediction of noiseless time series data, the usefulness of this for discrete problem domains such as grammar induction (and state representation) is questionable. Moreover, studies with ESNs for realistic problem domains have revealed the difficulty of creating the reservoir of interconnections (connections between hidden units) in a systematic way for a given problem. It can take the exploration of many reservoir configurations before a solution is found [22, 25]. Clearly, there is scope for advancing knowledge concerning the strengths and weaknesses of ESNs for different types of problem and a need for a principled approach to ESN application, appropriate to the problem domain in order to increase their utility. In particular, it will be interesting to determine whether the ESN learning algorithm is better able to discover the optimum solution for a complex grammar induction task than gradient descent-based learning methods used in traditional SRNs. This is important as RNNs, including SRNs and ESNs, are theoretically capable of representing universal Turing machines [46].

In this paper, we seek to ascertain whether the strong shift away from the SRN family of models trained with gradient descent for language acquisition tasks is premature. In particular, we provide further exploration of the MRN variant of the SRN, which appears to have gone largely unnoticed in the literature since 1996. In particular, we are interested in whether the unique MRN approach to associating temporal features with both nodes and state values, is sufficient to endow SRNs with superior power over ESNs enabling them to implicitly capture the sort of temporal dependency over variable time delays that may be associated with a complex grammatical structure. If this is shown to be the case, then this will have significant implications for current models of human memory and sentence comprehension that are dependent on ESN-like approaches, including those posed by [23, 42, 43, 44].

2. Network Architectures

2.1 Elman's Simple Recurrent Network

The SRN architecture employed in this study is depicted in Fig. 1. The activations of the hidden units of the network from time *t* are used as input to the network at time $t+1$. Recurrent connections provide the network with access to its prior state so the network has the ability to detect and learn temporal relationships within the data. The Input units I and hidden units (recurrent layer) R and the output units O are fully connected through the first order weight W^{RI} and W^{OR}, respectively, as in the feedforward Multilayer Perceptron (MLP). Time delay connections feed back the activities of recurrent (hidden) units $R^{(t)}$ to the context layer, i.e. $C^{(t)} = R^{(t-1)}$. Thus, each recurrent unit is fed by activities of all recurrent units from the previous time step through recurrent weights, W^{RC}. The context unit, composed of the activities of recurrent units from the previous time step, is treated as an extension of input units to the recurrent units. They represent the memory of the network.

Given input symbols in time $t, I^t = (I_1^t, \ldots, I_j^t, \ldots, I_{|I|}^t)$ and recurrent activities $R^t = (R_1^t, \ldots, R_j^t, \ldots, R_{|R|}^t)$, the recurrent unit's net input $\hat{R}_i^t$ and output activity R_i^t are computed as:

$$\hat{R}_i^t = \sum_j W_{ij}^{RI} I_j^t + \sum_j W_{ij}^{RC} R_j^{t-1} \quad (1)$$

where

$$R_i^t = f(\hat{R}_i^t) \quad (2)$$

The output unit k computes its net input $\hat{O}_i^t$ as:

$$\hat{O}_i^t = \sum_j W_{ij}^{OR} R_j^t \quad (3)$$

where

$$O_i^t = f(\hat{O}_i^t) \quad (4)$$

$|I|, |R| and\ |O|$ are the number of inputs, hidden and output units, respectively, and f is the activation function.

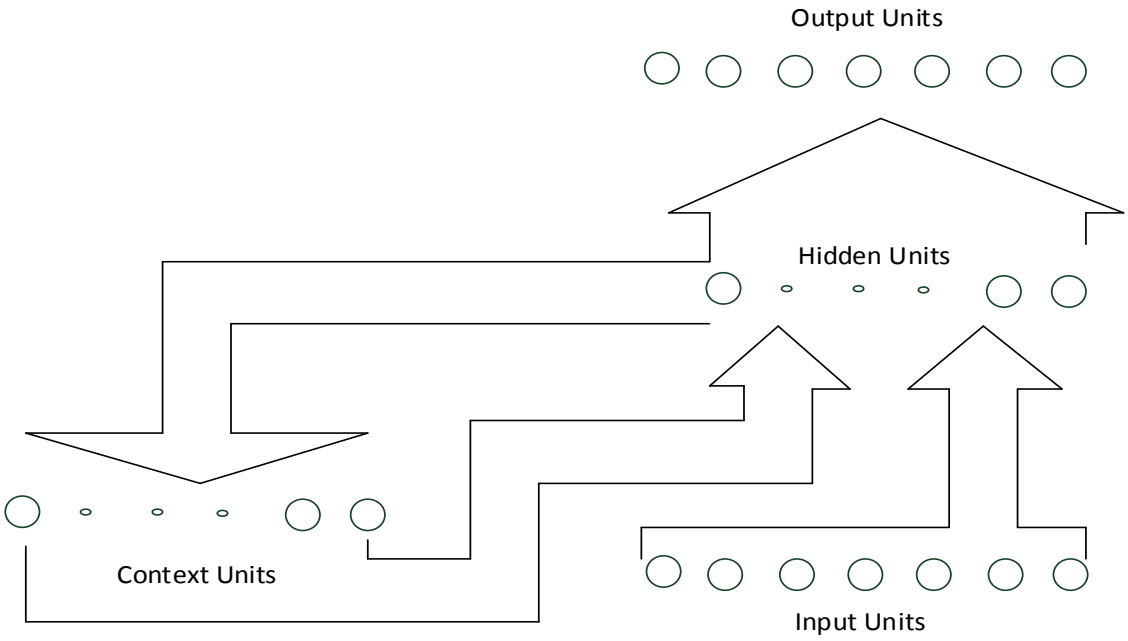


Fig 1. Simple Recurrent Network

2.2 Multi Recurrent Network

A further development of the SRN to enhance performance is to employ multiple feedback connections. Ulbricht [19] introduced the MRN architecture, illustrated in Fig 2. The construction provides four levels of feedback:

1. The output layer back to the input layer as established in Jordan networks [28].
2. The hidden layer back to the input layer, as found in SRNs [1].
3. The input layer back to itself.
4. The context units within the input layer back to themselves (self-recurrent links).

The above feedback connections form a memory bank that can be repeated with varying degrees of layer- and self-recurrency.

This enables an MRN to form a sluggish state-based memory that is both flexible and rigid: flexible in that some state units are very responsive to recent information; and rigid where others change more slowly, retaining past information for longer periods of time. The degree of rigidity and flexibility is dependent upon the ratio of the weight values for the self-recurrent weights and the layer-recurrent weights. The number of memory banks relates directly to the degree of granularity at which past and current information is integrated and stored.

In this work there is no input-layer recurrency (i.e. level 3. above). As with Binner et al. [5], four memory banks are used, see fig 3. Ulbricht [19] and Binner et al. [5] demonstrated that moving beyond four banks does not lead to enhanced performance; preliminary experiments have confirmed this result for this grammar induction task. Rather, it is the number of hidden units per memory bank that is pivotal to the performance of the network. The MRN function can be represented as follows:

$$y\,(t+1) = g\left(f\left(c(t), x(t), W_f(t)\right), W_g(t)\right) \qquad (5)$$

where: $y(t+1)$ indicates the predicted values of the symbol; *x(t)* is the external vector of input variables; *c(t)* is the concatenation of the previous hidden state vector with four delays of varying strength and summation of elements of previous output vector with four delays of varying strength; $W_f(t)$ is weight matrix connecting the input layer to the hidden layer; $W_g(t)$ is the weight matrix connecting the hidden layer to the output layer; vector function *f* returns activation vectors from the hidden layer; and function g returns activation vectors from the output layers.

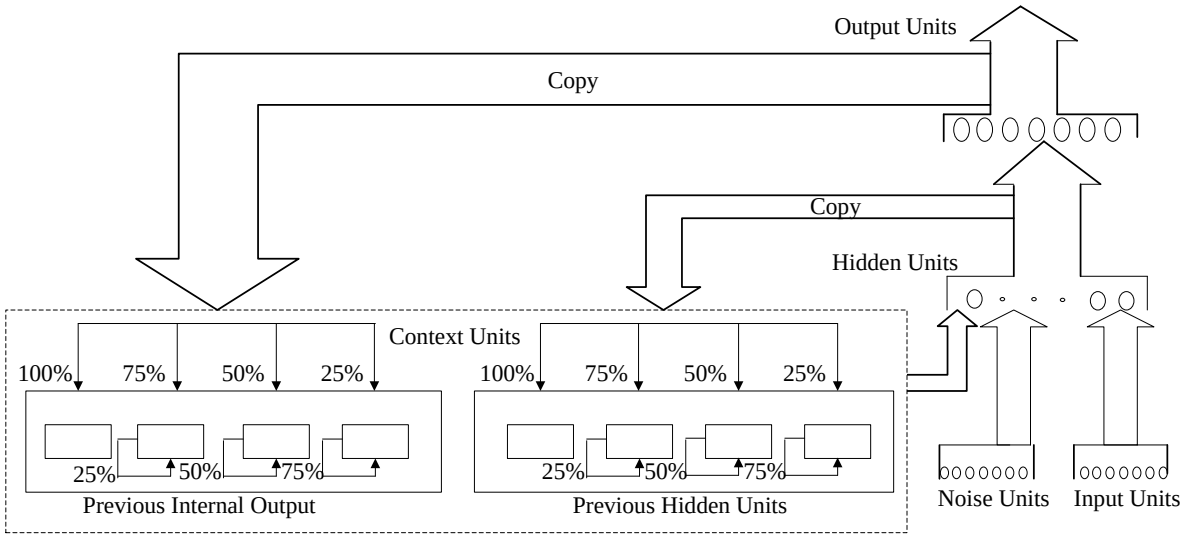


Fig 2. Architecture of Multi-Recurrent Network

We extended the MRN to include noise injection units (i.e. seven additional input units generating random values +/- 0.01) as prior experiments showed improved generalisation performance by up to 10% (noise injection did not give rise to any significant improvement for the other models tested here).

2.3 Echo State Network

ESNs were developed by Jaeger [29] to learn nonlinear systems for prediction tasks. Fig 3 shows the architecture of the ESN used in this research. The premise for this type of RNN is that the recurrent, dynamic part of the RNN does not need training. Instead, it functions as a non-specific memory and is known as a dynamic reservoir. It keeps information about the input sequence by allowing activations to rebound around the recurrent units [13, 23]. An ESN is a simple discrete-time RNN that has three layers: an input layer; a recurrent layer (also called reservoir, internal units or hidden units); and output layer. The reservoir is the core of the ESN structure: the readout layer extracts information from the reservoir. The network is fed each time step *t* with input vector u_t, which drives the dynamics of the recurrent layer, x_t, and output vector y_t. All connections from the input to the hidden units and between the units within the reservoir are fixed. The trainable weights are from the reservoir nodes and input layer to the output units; equation 6 describes the formula.

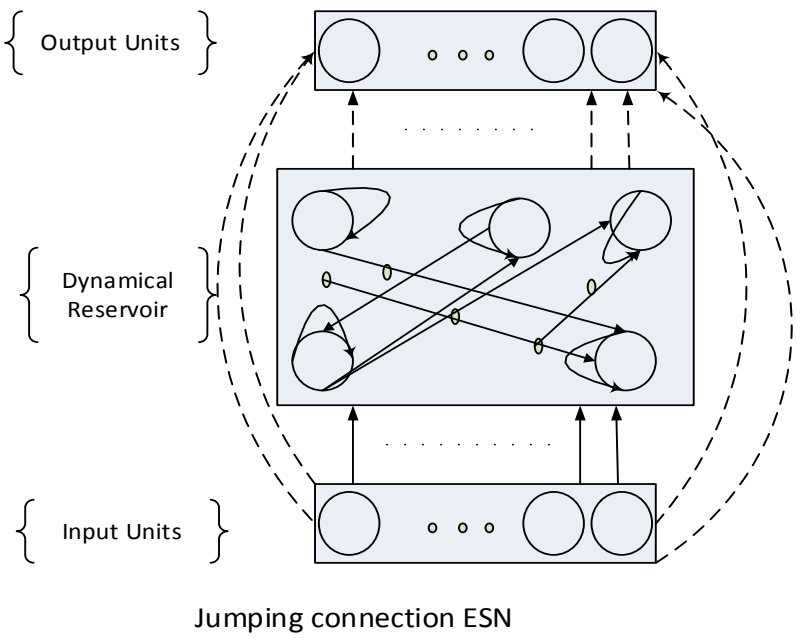


Fig 3. ESN Architecture: Solid arrows indicate fixed connections and dashed arrows indicate trainable connections.

$$V_{out(t)}=W_{out}(V_{in}(t),V_{hidden}(t)). \quad (6)$$

$$V_{hidden(t)}=\tanh(W_{in}(V_{in}(t)+ W_{hidden(}V_{hidden}(t-1))). \quad (7)$$

Where $V(t)$ is a vector that indicates the activation of the units at time t; W_{out} is the learned output weight matrix; W_{in} are the weights from the input layer to the hidden layer; and W_{hidden} are the recurrent connections from the previous state of the hidden layer to the current hidden layer.

The Echo State Property (ESP) states that given a long string of input, the reservoir of the network will be uniquely determined by the input history. Practically, the existence of echo states is obtained by giving random, sparse connections to the hidden layer and scaling the recurrent connections to have a Spectral Radius (SR) of less than one. The magnitude of the SR determines the stability of the memory; the closer the SR is to one, the longer the history that determines the current hidden state. The initial weights W_{hidden} are scaled to a desired SR α $(0 < \alpha < 1)$, and are calculated as:

$$W_{hidden} = \alpha \ddot{W}_{hidden} / |\delta_{max}|$$

Where δ_{max} is the maximum eigenvalue found in the matrix of eigenvalues, $\ddot{W}_{hidden}$.

When these weights are assigned, the network can be trained. Training involves setting W_{out} to map the training set to the desired output; if the output unit activations are linear or approximately linear, a simple linear regression can be used to train these connections. This frees the ESN from the lengthy training process associated with iterative algorithms such as backpropagation.

3. Description of Dataset and Methodology

The benchmark grammar used for this research is the Embedded Reber Grammar (ERG) [12, 27] (depicted in Fig. 4). It is an extension of the simple Reber grammar. We use a more complicated version than in [12, 27], as we allow self-recurrent transitions on states 3 and 4 as shown (up to four iterations). The ERG is a useful benchmark as it has been used by numerous other researchers [12, 18, 26, 27] for evaluating recurrent neural networks. Prediction of the next symbol in a sequence generated by the ERG represents a challenging task because the neural network must learn to remember the second symbol within the sequence (i.e. T or P), for potentially many time steps, in order to correctly identify the matching penultimate symbol of the sequence (i.e. T or P), before exiting the embedded grammar with symbol, E. Clearly, the longer the sequence of states traversed within one of the embedded sections, the more taxing this is on the memory mechanism. Such prediction tasks are thought to occur naturally within the brain as a fundamental feature of perceptual and cognitive processes [40].

The grammar symbols are represented utilising binary one-shot encoding i.e. one node per symbol: activations of 0 and 1 are used in all networks to indicate the presence or absence of an input symbol, except for the Echo-State Network where 0.2 and 0.8 are used. As with Cleeremans et al [12], we bias the dataset generated from the ERG to aid learning and enhance the performance of the network. Without biasing, the transitional probabilities from each state would be equal and set to be 0.5 (as there are a maximum of two transitions from each state). This gives rise to *symmetrical sequences* with respect to the T and P paths *within* each sub-grammar; they would be equally likely to be traversed, irrespective of whether they are in the top (T) sub-grammar or the bottom (P) sub-grammar. Servan-Schreiber et al [27] claim that this raises the difficulty of the prediction task (given the sub-grammars are identical) in a way that is unrealistic when considering how humans process natural language.

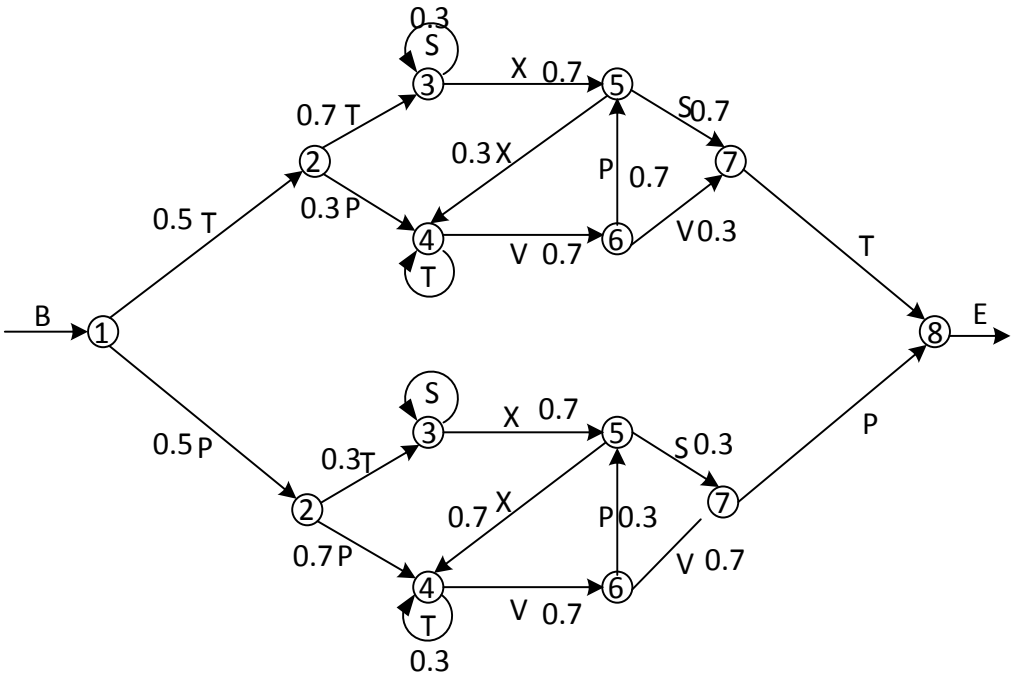


Fig 4. The ERG - a complex finite-state grammar containing embedded sub-grammars

They postulate that in natural language, when processing an embedded clause which separates say a subject from its auxiliary verb, the expectations of the concepts or words within the embedded clause are typically informed by the subject i.e. this expectation is maintained whilst anticipating and processing each word within the embedded clause. For example, consider the following two sentences (adapted from [27]):

(a) *The dog that chased the cat it saw is very playful*
(b) *The dogs that chased the cat they saw are very playful*

After the subjects of sentences (a) and (b) have been processed then it is likely a number of expectations will be generated as each subsequent word is processed e.g. a pack of dogs might chase something more substantial than a cat; a different morphological form of *chase* will be expected if assuming present tense. Also, there may be clues within the embedded clause itself as shown above, with (a) using ‘it’ to indicate singular and (b) ‘they’ to indicate plural. Such expectations are likely to be maintained, and cues encountered, when processing embedded clauses and these subsequently aid in determining the correct auxiliary verb, ‘is’ or ‘are’ (i.e. the appropriate exit from the associated embedded clause/grammatical structure).

As per [12] we therefore add a statistical bias to generate '*asymmetrical sequences*' by using transition probabilities of 0.7 or 0.3 (as shown in Fig 4). This generates embedded sequences with a higher frequency for the upper path relative to the lower path within the sub-grammar following T (upper embedded section) and a higher frequency for the lower path relative to the upper path within the sub-grammar following a P (lower embedded section).

Table 1 shows the distributions of the paths in the training dataset. As per [12], a dataset of 300000 sequences, generated randomly according to the probabilities in Fig 4, is used for training. 1000 randomly generated sequences are used for testing (sequence length 6 to 26, average 17.9). There is naturally occurring overlap within the training set, with only 212 sequences being truly unique to the test set. *Note that other researchers, such as [12, 18, 27, 47], also have this overlap but do not state the number of sequences unique to the test set and therefore do not provide true generalisation results.*

	Upper Embedded Section			Lower Embedded Section		
Initial symbol(s) in sequence	T	TT	TP	P	PT	PP
Number of sequences	150243	105204	45039	149757	44864	104893
% of Total	50.08	35.07	15.01	49.92	14.95	34.96
Balance of sequences within embedded sections (%)		70.02	29.98		29.96	70.04

Table 1: Statistical distribution of the training sequences between the upper and lower paths through the grammar (Total sequences = Number of T + Number of P = 300000)

In our experiments, the task of each network is simply to predict the next symbol in a sequence, given the current symbol and previous state (resulting from being presented with the preceding symbols in sequence). The prediction is considered correct if the highest output activation is for a valid successor symbol. If this criterion is not met, the network is considered to have failed in the prediction task. A pattern error-sensitive learning rate [15] is used as it was found to aid learning of the problem, relative to other adaptive learning rate regimes. Each training experiment is repeated 15 times (with the same architecture and training parameters) to account for the sensitivity these models have to the initial starting state (determined by the randomly generated initial weight values).

4. Performance of the Networks

A range of experiments spanning over a 100 different hyper-parameter configurations was undertaken to arrive at the optimum parameter values reported in Table 2.

We followed the methodology of [12] for model fitting and selection to maintain comparability, although we used *back-propagation through time* [38] for the SRN and MRN. This has the advantage of ensuring errors, and thus weight updates, are generated relative to a sequence of input symbol presentations rather than an individual input symbol. To successfully learn the task, the RNNs must learn to maintain internal states that differentiate between the two embedded grammars, even though the details within the embedded grammars are the same, i.e. when processing the upper T sub-grammar, the internal network state must maintain representation that it is in T (upper) embedded section and not P (lower); the same applies regarding retaining the equivalent historic information when it is processing the P (lower) section. However RNNs naturally gravitate towards a stable state when processing long sequences of identical input/output mappings, such that every new presentation of the same input symbol (e.g. in a long string within one of the embedded grammars) will tend to produce the same hidden unit response and thus the same output response, until eventually any hidden unit information representing the entry symbol (i.e. T or P) is lost.

	Memory Banks	Hidden Units	Learning Rate	Momentum	Weight Range	Spectral Radius	Connectivity
MRN	4	10	0.15	0.75	0.3	-	-
SRN	-	15	0.15	0.75	0.3	-	-
ESN	-	150	-	-	0.3	0.75	0.85

Table 2. Optimum hyper-parameters for each network. Note that the learning rate regime for the SRN and MRN is adaptive as per [15].

Servan-Schreiber et al [27] showed that increasing the number of hidden units, enables small but crucial information to be retained within the hidden state, for learning a prediction task for a limited range of embedding. However beyond this limited range, training becomes exponentially difficult, as previous input information, encoded within error gradients, is lost;

this is now known as the vanishing gradients problem [17]. This can be alleviated with some success by extended memory architectures designed to retain past input information, as shown by Lin et al [34]. It is therefore expected that this limitation will be clearly evident within the SRN performance; evident to a much lesser extent in the MRN performance (due its complex memory architecture for latching onto past information); and even less evident in the ESN performance (due to its entirely different learning mechanism). Table 3 shows the results of both the training and testing of the networks. As the test set is randomly generated from the ERG it is very likely that there is a natural overlap with those sequences generated for the training set. Table 3 therefore shows performance on the full test set (containing all sequences including those overlapping with the training set) and also the subset of the test sequences that do not overlap with the training set (thus providing for a more *pure* measure of generalisation performance). It can be observed clearly that the MRN has the superior performance over the ESN and surprisingly, that the SRN also out-performs the ESN, albeit much less significantly than the MRN.

	SRN	MRN	ESN
Training set (300000 seq)	92.91%	99.91%	91.7%
Full Test set (All 1000 randomly generated seq)	61.1%	97.7%	60.7%
Pure Test set (212 unique seq)	50.94%	95.28%	48.58%

Table 3. Comparative network performance: correctly predicted training and test Asymmetrical sequences.

In order to understand why the MRN is superior, we follow the methodology of Cartling [8] to analyse the resulting internal representations formed by each of the different models.

5. Analysis of Internal Representation

So far the underlying representations or 'hypotheses' formed by the recurrent networks have been treated as if they were in a 'black box'. The hidden units inside these networks express the grammatical knowledge encoded by the weights. By understanding how these units respond to each input symbol over time, it is possible to determine whether or not these networks have formed an adequate representation of the underlying ERG. Numerous researchers have studied the internal representation of RNNs [1, 30, 31, 32]. Due to its simplicity and clarity, we use the

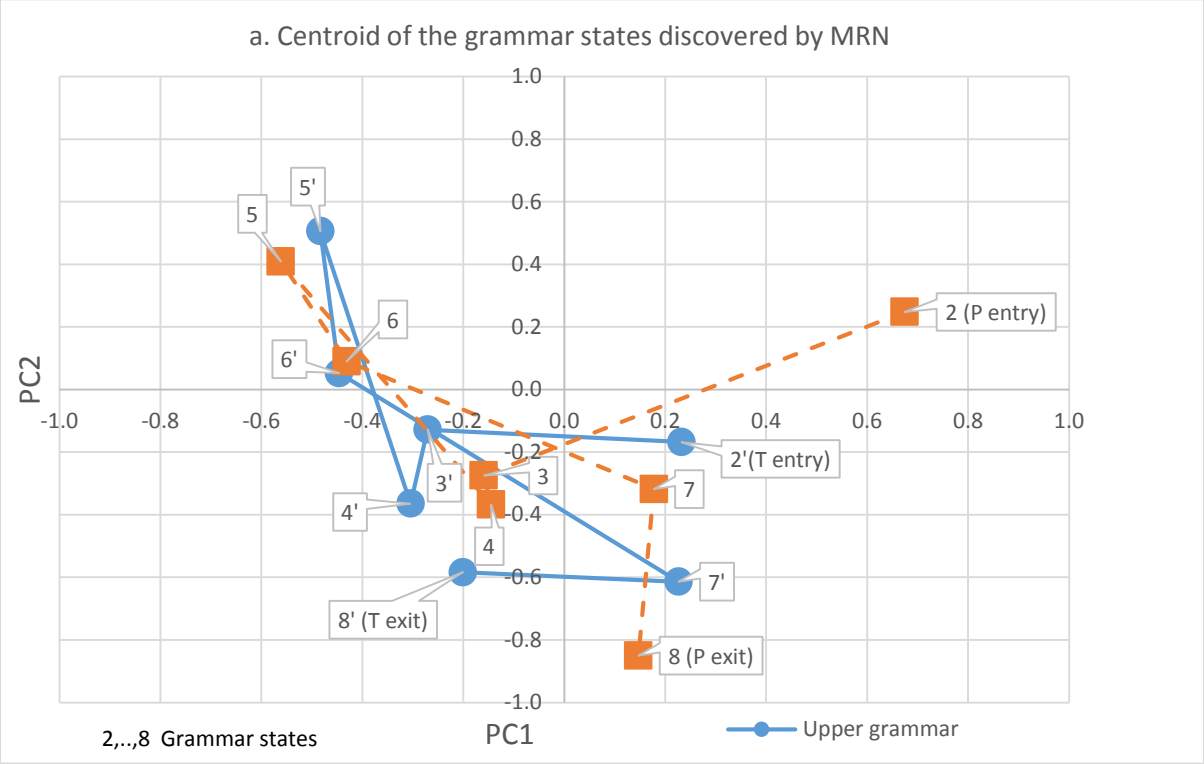
same approach as Cartling [8] who applied Principal Component Analysis (PCA) to the hidden state activations of an SRN, trained on a similar grammar induction task. The internal representation formed by the weights of an RNN after training can be revealed by analysing the resulting hidden unit activations and how they vary with respect to each new input within a sequence[33]. This can be determined by calculating the principal components of the covariance matrix of hidden unit activations and their respective mean values as follows.

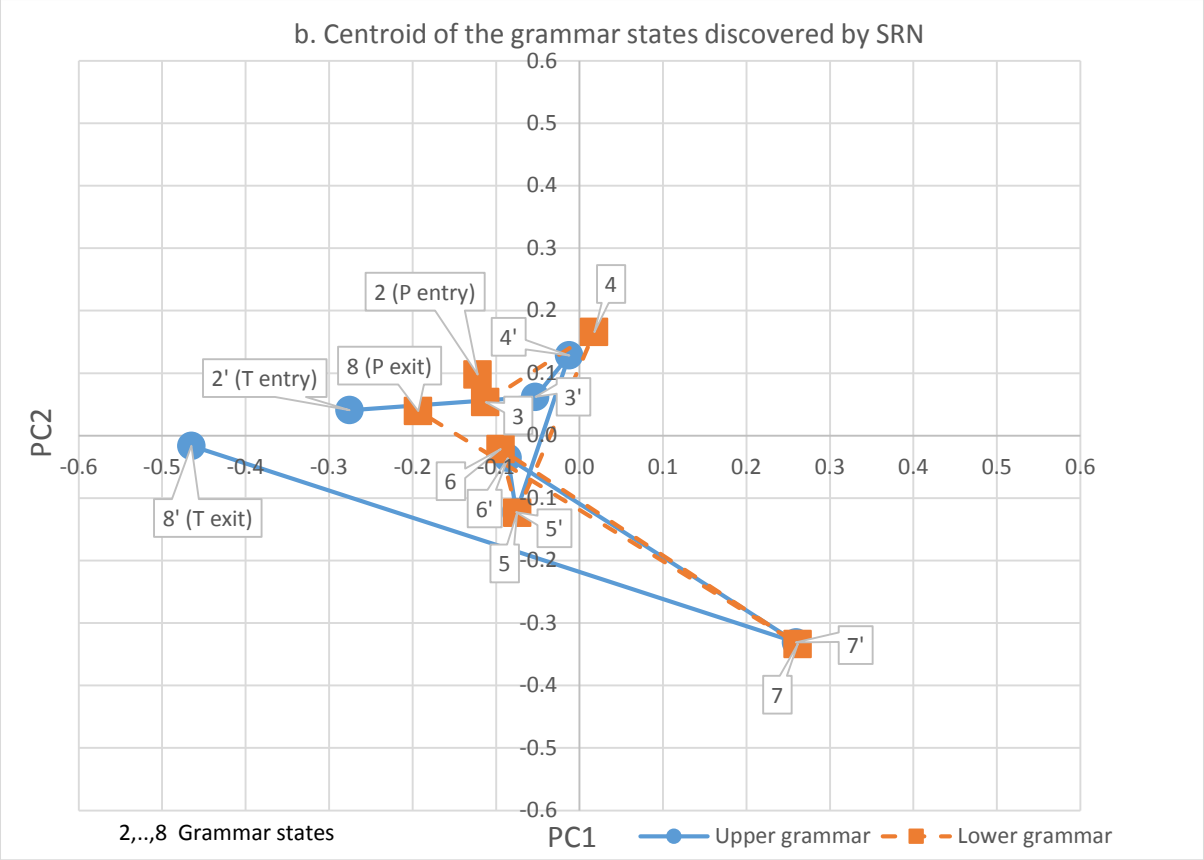
$$C_{ij} = \sum\left(\left(y_i^h - mean_{y_i^h}\right)\left(y_j^h - mean_{y_j^h}\right)\right),\ \ i = 1, \ldots n^h, j = 1, \ldots, n^h \qquad (6)$$

The average activation for each hidden unit is based on the unit's response to all symbols in the *training* set. The resulting principal components are ordered according to the magnitude of the corresponding eigenvalues, where the highest eigenvalue represents the most feature information captured within the internal (hidden) state and the lowest represents the least. Two Principal Components (PCs) can then be used to generate a two-dimensional view of the state-space of the networks. During the training process, it is expected that within its internal state, a trained RNN will form clusters of activations in state-space that represent the grammar states of the underlying ERG. It is further expected that where training has been successful, these grammatical state-based activations will themselves be arranged in patterns that distinguish between the upper and lower embedded grammar sections. The aim of the analysis based on PCA is to allow two-dimensional views of the networks' state-space to be generated, thereby facilitating visualisation of the above clusters and patterns and providing opportunity for further insight into the networks' inner workings and testing of our understanding of their operation. In this work, various two- dimensional subspaces formed by applying PCA to the hidden unit state activations after training were examined. Although different grammar states could be distinguished in a number of PC subspaces, the subspace relating to the eigenvectors corresponding to the first and second largest absolute eigenvalues of the covariance matrix (PC1 and PC2), proved to be the most useful.

In order to delineate the representations formed of the underlying embedded grammars in PCA space (using PC1 and PC2), we calculated the centroid for each grammar state (i.e. seven states for the upper sub-grammar and seven for the lower). This is easily calculated by passing all training or test sequences through the trained network and calculating the average PC1 and PC2 values for each grammar state. These average coordinates can then be plotted to reveal the RNN's representation of the underlying sub-grammars.

Figure 5 shows the resulting centroid plots in PC space computed for each of the RNNs under evaluation when all *test* sequences that were correctly predicted by the MRN were passed through. We selected the MRN as it correctly classified most of the unique test sequences i.e. 95% of the 212 sequences. The resulting plots are assessed to determine if they reveal differences in the underlying representations that are responsible for the poorer performance of the SRN and ESN.





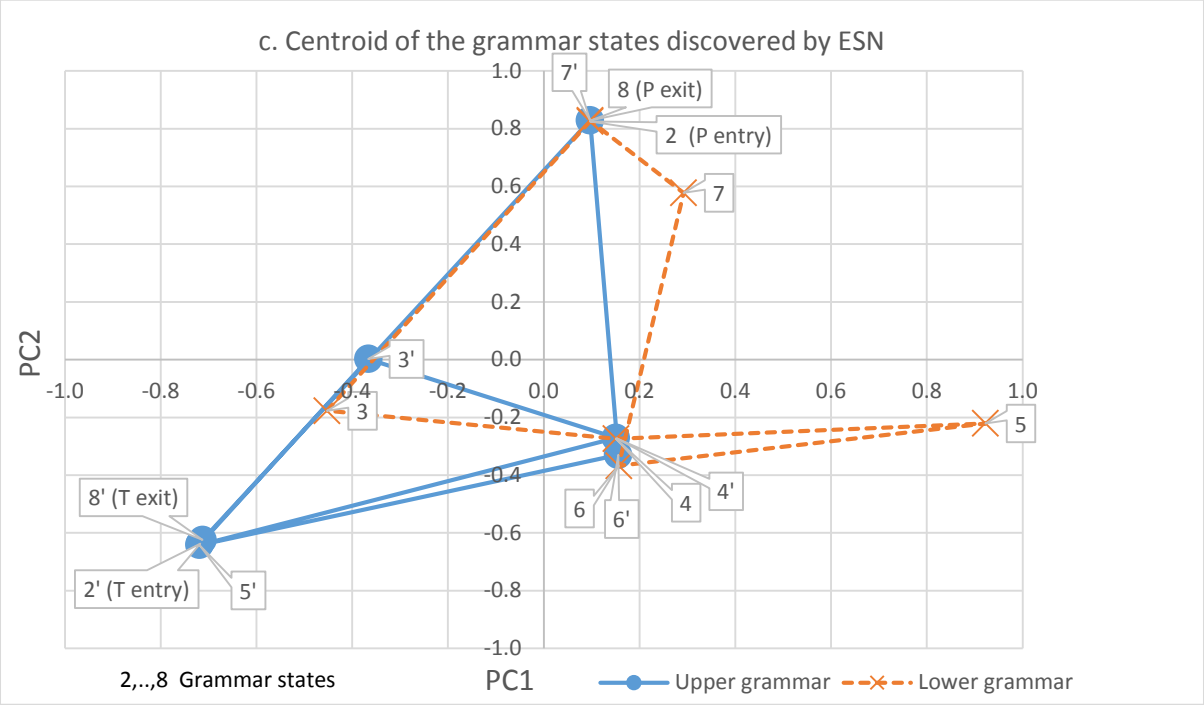


Fig 5. Trajectories of the centroid plots for each grammar state discovered by the: a. MRN; b. SRN; and c. ESN, in relation to both the upper (i.e. T entry/exit symbol) and lower (i.e. P entry/exit symbol) sub-grammars of the ERG.

We aim to understand, for the different network models, how well structured their internal representations of the ERG are. We consider how the centroid plots have been organised within and beyond each quadrant where Quadrant 1 (Q1) refers the top right quadrant; Q2 the top left; Q3 the bottom left; and Q4 bottom right. Fig 5a shows that the MRN has dedicated different quadrants to the entry and exit points of the embedded sub-grammars with Q4 used for *entry* to the upper (T) sub-grammar and *exit* of the lower (P) sub-grammar with different values for PC2 distinguishing between them. Q1 is dedicated to *entry* of the lower sub-grammar. Q3 captures trajectories from state 2 to states 3 and 4 of both sub-grammars, using the low negative range (between 0 and -0.39) of PC1 and PC2 to represent transitions between these states, including the self-recurrent transitions. Q2 represents the transitions from states 3 and 4 to states 5 and 6 for both sub-grammars, with medium (0.4 to 0.59) positive values of PC2 used to represent state 5 and low (0 to 0.399) positive values to represent state 6 of the upper and lower sub-grammars. Q4 captures transitions from states 5 and 6 to the exit state of the sub-grammars, state 7. Again, values of PC2 play a key role in distinguishing between states and in this case, the upper and lower sub-grammars. Negative medium (0.4 to 0.599) values of PC2 are used to represent state 7 of the upper grammar and low (0 to 0.399) negative values exit from the lower sub-grammar. High (0.6 to 1.0) positive PC2 values of Q4 are then used to represent the transition from state 7 of the lower grammar to state 8, the final state. Finally, Q3 captures the exit from the upper sub-grammar to the final state. Interestingly, whereas PC1 remains in the negative low range for Q3, PC2 again is pivotal to differentiating the role of Q3, with low negative values representing states 3 and 4 of both sub-grammars (as mentioned earlier) and reserving medium negative values to represent the exit from the upper sub-grammar. This is a strong illustration as to how the MRN's embedded memory architecture has enabled it to dedicate fine granular regions of state space to represent crucial aspects of the underlying grammar, such as entry into and exit from the upper and/or lower sub-grammars.

As can be seen from 5b, the SRN was less able to dedicate different regions of state space to the crucial states and transitions of the ERG. This is evident from the concentration of states within Q2 and in particular of the negative low values of PC1 and positive low values of PC2 to represent states 1 to 3 of the upper and lower sub-grammars and state 4 of the upper sub-grammar. This concentration of initial states around a narrow range of PC space is likely to make the SRN less able to distinguish transitions within the upper and lower sub-grammars (as reflected in its relatively poor generalisation performance). Q1 is only used (via a positive low value of PC1) to represent state 4 of the lower sub-grammar, with its nearest neighbour still

being state 4 of the upper sub-grammar. Q3 represents the transitions from states 3 and 4 to states 5 and 6 for both sub-grammars (similar to Q2 of the MRN), however, this again is within an extremely narrow range of PC space, with all representations of states 5 and 6 being contained within very low negative values of PC1 and PC2 illustrating a low separation within its representation of the underlying embedded sub-grammars. Likewise, its use of Q4 to represent transitions to state 7, the exit state for the embedded sub-grammars, does not sufficiently distinguish between the different sub-grammars, as the centroids are in almost identical positions. Finally, the SRN *is* able to adequately represent the transition from the sub-grammar exit state (7) to the finish state (8), with Q2 used for lower sub-grammar and Q3 for the upper. Again, this is concentrated into a very narrow range of PC2 space, albeit in a different direction for each of the sub-grammars, and this is likely to be the reason for its limited generalisation performance.

The ESN had learnt more of the training set than the SRN, however, its generalisation performance was slightly worse at 48.6% with it only able to generalise to novel sequences for the lower (P) sub-grammar – misclassifying *all* novel test sequences that traverse the upper (T) sub-grammar. When reviewing the differences between the use of the state space for the upper and lower grammar, it is clear that Q1 was used to capture the entry path to the lower sub-grammar and the transition to the top (T) path within it i.e. state 2 of the lower sub-grammar; whilst Q3 is used to capture entry path to the top sub-grammar and also the top (T) path of the grammar (states 3 and 5). However, it is notable that the transitions from state 2 to 3 of the lower sub-grammar, and then from state 3 to 5 are all represented by transitions across quadrants (Q1 to Q3 and then Q3 to Q4 respectively) thus showing the broad utilisation of distinct areas of state space. This cohesive representation explains why the ESN generalises very well to all strings traversing this top path of the lower sub-grammar. Q4 also captures the bottom paths of <u>both</u> sub-grammars (transitions from states 2, to 4 and then to 6). *We therefore suspect that the ESN's poor generalisation performance is due to the lack of sufficient separation of the transitions from state 2 to states 4 and then 6 (of both sub-grammars). The centroids for state 3 are also noticeably clustered close together.* Similar to the other models, the ESN learnt to associate the transition from states 5 and 6 to state 7, the exit point from the sub-grammars, with a dedicated region (Q1) of state space. Q1 is also used to traverse from state 7 of the lower sub-grammar to the finish state, 8. Clearly, activations clustered within Q1 and Q4 are essential for accurate recognition of sequences that traverse the lower sub-grammar. Likewise, sequences of activations within Q3 and Q1, with some influence from activations

within Q4, are essential for representing the upper sub-grammar. However, the key issue affecting generalisation performance is that there is insufficient separation between the bottom-paths of both sub-grammars.

In order to quantify the distances formed between respective states of the upper and lower sub-grammars, we calculated the Euclidean distances between the centroids of each pair of corresponding states between the upper and lower sub-grammars (e.g. between both states 2 in Fig. 4). The results are presented in Figure 6, where it can be seen that the MRN is more able to consistently maintain a distinct distance between the corresponding grammar states of the upper and lower sub-grammars.

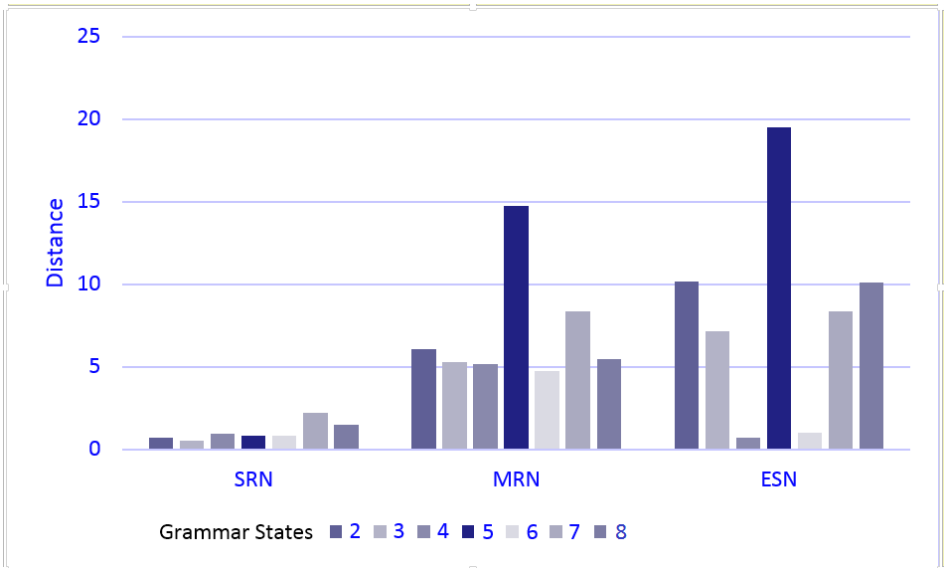


Fig *6*. Euclidean distances between centroid pairs representing corresponding states within the upper and lower embedded sub-grammars (for each model evaluated).

We contend that this is due to the ability of its sluggish state-based memory to latch onto and maintain information about the entry points (T and P) of the respective sub-grammars throughout the respective sub-grammar sequences.

In order to further assess the memory capacity of each of the models, we generated a number of test sequences that exceeded those found in the training data in terms of length and complexity. This was achieved by increasing the number of recursive transitions allowed at states 3 and 4 in within the embedded sub-grammars. Table 4 shows the results after testing the models on these long sequences; the minimum sequence length of this new test set was 28 symbols and the maximum 900 symbols. On examining the performance of each model, at 100 symbol intervals, it was clear that the MRN had learnt to more accurately recognise longer

sequences that traversed a broader range of states across both embedded sub-grammars, up to length 100, but following this, performance declines as the sequences become very long. It is relatively successful in representing the task as a whole including the long term dependency. The ESN on the other hand seems to have learnt part of the ERG (the lower sub-grammar) perfectly, regardless of length. It consistently misclassifies the exit state for the upper grammar whilst successfully predicting most of the lower section, and has therefore not learnt the long term dependency problem (largely due to the lack of distinctness between states 4 and 6).

SEQUENCE LENGTH	SRN	MRN	ESN
28 TO 100	40	70	40
101 TO 200	20	32	40
201 TO 300	16	20	48
301 TO 400	4	12	48
401 TO 500	0	8	56
501 TO 600	0	0	48
601 TO 700	0	0	44
701 TO 800	0	0	40
801 TO 900	0	0	40

Table 4. The performance of the networks when tested with randomly generated long sequences (% correct). Note that all lengths are beyond that of the training set.

Finally, Table 4 shows that the performance of the SRN degrades relatively rapidly as the sequence length increases beyond 100. In the state analysis it maintained very *small* distinct distances across most corresponding states of the embedded grammar. This enabled it to represent the breadth of states required better than the ESN, giving rise to a 2% improvement in performance for generalising to the 212 unseen test sequences. . We suggest that the poorer performance on the longer sequences is due to degradation of memory associated with the SRN feeding back and committing to 100% of current (as opposed to historic) state information at each time step.

The performance characteristics of the models appears in keeping with their respective internal representations of the underlying ERG: the MRN maintained sufficient distances between *all* of the corresponding states of the embedded grammars, the SRN maintained much smaller distances across all states whereas the ESN only maintained substantial distances across a *reduced set* of states for which it generalised well.

6. Discussion

The present study makes a number of noteworthy contributions to connectionism. Firstly, it provides further support for the findings reported by Lin et al 1998 [34], i.e. that using embedded memory architectures within SRNs may help to alleviate the inherent disadvantages of gradient descent learning. It may therefore not be strictly necessary to seek recourse to more complex second-order schemes or radically different learning algorithms in order to solve complex grammar induction problems. Assuming the number of hidden units and the ratio of weight values for the self-recurrent and layer-recurrent weights (and thus the number of memory banks) are optimised for the 'next symbol prediction' task, our results suggest that an MRN trained with back-propagation through time and simple noise injection, will be able to induce a stable and robust representation of the underlying temporal structure (or context-free grammar) that is generating the input sequences..

To further evaluate the significance of the MRN's architecture, we also applied the NARX network to the ERG task and although the NARX had *slightly* better generalisation performance than the SRN and ESN for the 212 unique test sequences with 51.9% accuracy, it was significantly below that of the MRN (95.3%). The sluggish memory mechanism of the MRN therefore appears superior to the NARX's input/output shift-register memory architecture in solving the prediction task. It seems better able to retain important input features, latching onto them and propagating them through time (within its context units) for use at future time-steps.

Secondly, we are the first to apply the ESN to the ERG induction task and reveal its serious limitations for long term dependency problems. This is in sharp contrast with the findings of Tong et al [23] who found the ESN was comparable to the SRN for the next symbol prediction task using Elman's context free grammar [1]; the advantage for the ESN here being that it required less adjustable parameters and training time and is therefore more efficient than the SRN. Little information, however, was reported regarding: the complexity of the input sequences; overlap between training and test data; the internal representations formed and whether the ESN preferred particular paths through the grammar. In our experiments, the ESN did not learn the dependency problem, unlike the MRN. However, the potential of the ESN is clearly exhibited through its ability to learn one of the ERG's sub-grammars perfectly, appearing to generalise to all strings traversing the lower sub-grammar irrespective of length. This is an interesting finding, particularly for those researchers, such as [23, 42, 43, 44], who prefer the ESN for modelling human memory and sentence comprehension.

Finally, Table 5 provides a high level comparison of the MRN performance presented in this paper, with published results from other connectionist approaches to the ERG prediction task.

	Architecture/Method				Training Set					Test Set		
	Network	**HU**	**SRT**	**RBS**	**NS**	**Max SL**	**ASL**	**SDSL**	**%L**	**NS**	**US**	**%G**
Servan-Schreiber et al. [27]	SRN With BP	15	0	80:20	? 900K 'exemplars'	8	5.8	1.3	?	20K (? symbols)	?	?
O'Connell [33]	SRN with PA And BPTT	15	0	90:10	2.5M strings (?)	?	?	?	99.5	1000 (? Symbols)	?	99.5 (once out of 20 nets)
Perez-Ortiz Et al [18]	LSTM with DEKF	8	?	50:50	? (1M Symbols)	N/A	N/A	N/A	?	? (1M Symbols)	?	99.9 SP (3 errors)
This work	MRN with BPTT	10	4	70:30	300K (2.3M symbols)	26	8.78	2.31	99.9	1000 (17.5k symbols)	212	97.7 (95.3 of US)

Table 5. A comparison of the MRN performance on the ERG, with other published approaches. *?: Data not provided, N/A: Not applicable, Average ASL: Average Sequence Length, DEKF: Decoupled Extended Kalman Filter, G: generalisation, HU: Hidden Units, L: learnt, NS: Number of sequences (or symbols where stated), PA: Periodically Attentive, RBS: Ratio of Biased Sequences, SDSL: Standard Deviation for sequence length, SL: sequences Length, SP: Sustainable Prediction, SRT: Self-Recurrent Transitions Allowed in ERG, US: Unique Sequences.*

As can be seen, due to the lack of consistency in the way in which researchers approach the grammar induction problem, report training and test sets, and interpret the resulting performance, a direct comparison is not possible. For example, all but Perez-Ortiz et al reset the context units at the beginning of each new training or test sequence. Perez-Ortiz et al have positioned the ERG as a continuous time problem where there is only a single input stream i.e. sequences are generated from a recursive ERG where there is a transition from the last symbol, E, of the grammar back to the first symbol, B. Learning continues even when the LSTM makes mistakes; and training and test are not divided into separate phases. They aim for the LSTM to generate error-free predictions (or sustainable predictions) for at least 1000 subsequent symbols where error-free refers to the LSTM only activating output nodes that correspond to valid next state transition symbols allowed by the ERG. Their rationale for this is that resetting the context units at the beginning of each sequence would require an 'external teacher' to segment the input into training sub-sequences; they are interested in not having any *a priori* knowledge of this kind, which gives rise to a continuous input stream. Given that in language, spoken or written,

there are usually clear boundary cues between contiguous sentences (which may or may not relate directly to one another), we consider the mainstream approach of resetting the context nodes at the beginning of each sequence as being more aligned to naturally occurring conditions. Also, Perez-Ortiz et al do not indicate the complexity of their input strings (e.g. the level of embedding allowed or the maximum length of any single pass through the ERG). Note that we have reported Perez-Oriz et al's best results achieved with the LTSM rather than the average, which was substantially poorer.

Servan-Schreiber et al do not explain whether their 900K training exemplars are strings or symbols (we assume strings) and whether or not the SRN had learnt them. Also, there was no indication regarding how distinct the test sequences were from the training sequences; to our knowledge, we are the only authors to explicitly report performance on *unseen* test cases for the ERG problem. Our results clearly show that the MRN is able to process *unseen* discrete sequences generated from the ERG of up to 500 symbols in length, whereas none of the other research has demonstrated such *pure generalisation* capacity in this way; in all cases their test strings overlapped significantly with their training strings i.e. a significant proportion of the test strings had been presented to the networks in training.

7. **Conclusion and Further Work**

In this study, we sought to ascertain whether the strong shift away from the SRN class of RNNs towards potentially more powerful alternatives such as the LSTM and ESN models may be premature for the task of grammar induction. More specifically, we re-opened the case for SRNs, by considering a variant of the SRN, referred to as the MRN. This uses an embedded memory architecture, consisting of layer recurrent and self-recurrent links, to ameliorate against the vanishing gradient problem associated with gradient-descent learning algorithms. We contrasted in detail, the performance of the MRN with that of the SRN and ESN, for the popular embedded Reber grammar induction task.

Our results showed that the MRN significantly outperforms both the SRN and ESN for the *complete* ERG induction problem. Although the ESN was unable to capture the long term dependency within the grammar, it was able to perfectly learn the lower sub-grammar of the ERG. Surprisingly, we find that even the standard SRN is slightly superior (by 1 and 2%) to the ESN for shorter string lengths (< 28 symbols) and falls behind the ESN performance for

strings with length that greatly exceeds the maximum found within the training set. Although the ESN has enjoyed much success in predicting continuous time-series, our results suggests caution when applying the ESN to discrete grammar induction tasks, and that further study is required. We suspect that the ESN's limitations are due to the fixed reservoir weights and dependence on an echo-state property, which restrict the nature and complexity of the features it is able to learn to represent (as revealed in our state space analysis for the ERG).

Unlike the other models, the MRN, due to its sluggish state-based memory, is able to dedicate well-defined regions of state space, in a co-ordinated fashion, to represent crucial temporal features of the *full* underlying grammar or generative model (e.g. long term dependency.) This work therefore provides support for further exploration of the MRN for modelling human memory and sentence comprehension. Our future work will investigate how better to optimise the MRN for such problems and also whether the generalisation performance of the MRN, when processing centre-embedded clauses, is akin to that of human working memory, building on the work of Cowan [39].

Reference

[1] Elman, J. (1991). "Distributed representations, simple recurrent networks, and grammatical structure". *Machine Learning*, vol 7, 195–224.

[2] D. Palmer-Brown, J. A. Tepper, and H. M. Powell (2002). "Connectionist natural language parsing," *Trends Cogn. Sci.*, vol. 6, no. 10, pp. 437–442, 2002.

[3] M. H. Christiansen and N. Chater (2001). "Connectionist psycholinguistics in perspective," in Christiansen, MH and Chater, N, (eds.) *Connectionist psycholinguistics*. pp. 19-75. Ablex: Westport, CT.

[4] T. Koskela, M. Varsta, J. Heikkonen, and K. Kaski (1998). "Temporal sequence processing using recurrent SOM," in *Knowledge-Based Intelligent Electronic Systems, 1998. Proceedings KES'98. 1998 Second International Conference on*, 1998, vol. 1, pp. 290–297.

[5] J. M. Binner, P. Tino, J. Tepper, R. Anderson, B. Jones, and G. Kendall (2010). "Does money matter in inflation forecasting?," *Phys. A Stat. Mech. its Appl.*, vol. 389, no. 21, pp. 4793–4808, 2010.

[6] J. F. Kolen and S. C. Kremer (2001). *A field guide to dynamical recurrent networks*. John Wiley & Sons, 2001.

[7] I. Sutskever and G. Hinton (2010). "Temporal-kernel recurrent neural networks," *Neural Networks*, vol. 23, no. 2, pp. 239–243, 2010.

[8] B. Cartling (2008). “On the implicit acquisition of a context-free grammar by a simple recurrent neural network,” *Neurocomputing*, vol. 71, no. 7–9, pp. 1527–1537, Mar. 2008.

[9] J. L. Elman (1995). “Language as a dynamical system,” *Mind as motion Explor. Dyn. Cogn.*, pp. 195–223, 1995.

[10] W. Deliang, L. Xiaomei, and S. C. Ahalt (1996). “On temporal generalization of simple recurrent networks,” *Neural Networks*, vol. 9, no. 7, pp. 1099–1118, 1996.

[11] L. Gupta, M. McAvoy, and J. Phegley (2000). “Classification of temporal sequences via prediction using the simple recurrent neural network,” *Pattern Recognit.*, vol. 33, no. 10, pp. 1759–1770, 2000.

[12] A. Cleeremans, D. Servan-Schreiber, and J. L. McClelland (1989). “Finite State Automata and Simple Recurrent Networks,” *Neural Comput.*, vol. 1, no. 3, pp. 372–381, Sep. 1989.

[13] S. L. Frank (2006). “Learn more by training less : systematicity in sentence processing by recurrent networks,” *Connection Sci*. vol. 18, no. 3, pp. 287–302, 2006.

[14] A. S. Noel Sharkey and S. Jackson (2000). “Are SRNs Sufficient for Modeling Language Acquisition?,” Oxford University Press, 2000, pp. 33–54.

[15] J. A. Tepper, H. M. Powell, and D. Palmer-Brown (2002) “A corpus-based connectionist architecture for large-scale natural language parsing,” *Conn. Sci.*, vol. 14, no. 2, pp. 93–114, 2002.

[16] I. Farkaš and M. W. Crocker (2008) “Syntactic systematicity in sentence processing with a recurrent self-organizing network,” *Neurocomputing*, vol. 71, no. 7, pp. 1172–1179, 2008.

[17] Y. Bengio, P. Simard, and P. Frasconi (1994). “Learning long-term dependencies with gradient descent is difficult,” *Neural Networks, IEEE Trans.*, vol. 5, no. 2, pp. 157–166, 1994.

[18] J. A. Pérez-Ortiz, F. A. Gers, D. Eck, and J. Schmidhuber (2003). “Kalman filters improve LSTM network performance in problems unsolvable by traditional recurrent nets,” *Neural Networks*, vol. 16, no. 2, pp. 241–250, 2003.

[19] C. Ulbricht (1994) “Multi-recurrent networks for traffic forecasting,” in *Proceedings Of The National Conference On Artificial Intelligence*, p. 883.

[20] G. Dorffner (1996). “Neural networks for time series processing,” in *Neural Network World*, 1996.

[21] F. Gers (2001). “Long Short-Term Memory in Recurrent Neural Networks,” *Lausanne, EPFL*, 2001.

[22] H. Jaeger (2002). *Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the“ echo state network” approach*. GMD-Forschungszentrum Informationstechnik, 2002.

[23] M. H. Tong, A. D. Bickett, E. M. Christiansen, and G. W. Cottrell (2007). “Learning grammatical structure with Echo State Networks.,” *Neural Netw.*, vol. 20, no. 3, pp. 424–32, Apr. 2007.

[24] M. D. Skowronski and J. G. Harris (2006). “Minimum mean squared error time series classification using an echo state network prediction model,” *2006 IEEE Int. Symp. Circuits Syst.*, no. m, pp. 3153–3156, 2006.

[25] A. Rodan and P. Tino (2011). “Minimum complexity echo state network.,” *IEEE Trans. Neural Netw.*, vol. 22, no. 1, pp. 131–44, Jan. 2011.

[26] S. E. Fahlman (1991) The recurrent cascade-correlation architecture. In: *Advances in neural information processing systems* 3, ed. R. P. Lippmann, J. E. Moody & D. S. Touretzky. Morgan Kaufmann

[27] D. Servan-Schreiber, A. Cleeremans, and J. L. McClelland (1991). “Graded state machines: The representation of temporal contingencies in simple recurrent networks,” *Mach. Learn.*, vol. 7, no. 2–3, pp. 161–193, 1991.

[28] M. I. Jordan (1986). “Attractor dynamics and parallelism in a sequential connectionist machine,” in *Proceedings of 9th Annual Conference of Cognitive Science Society*, 1986, pp. 531–546.

[29] H. Jaeger (2001). “The‘ echo state’ approach to analysing and training recurrent neural networks-with an erratum note’,” *Bonn, Ger. Ger. Natl. Res. Cent. Inf. Technol. GMD Tech. Rep.*, vol. 148, 2001.

[30] M. Craven and J. W. Shavlik (1994). “Using Sampling and Queries to Extract Rules from Trained Neural Networks.,” in *ICML*, 1994, pp. 37–45.

[31] R. Setiono and H. Liu (1995). “Understanding neural networks via rule extraction,” in *IJCAI*, 1995, vol. 1, pp. 480–485.

[32] J. Bullinaria (1997). “Analyzing the internal representations of trained neural networks,” *Neural Netw. Anal. Archit. Algorithms*, pp. 3–26, 1997.

[33] T. C. O'Connell (1995) *Using Periodically Attentive Units to Extend the Temporal Capacity of Simple Recurrent Networks*. Unpublished MSc Thesis, University at Albany, State University of New York, Department of Computer Science.

[34] T. Lin, B. G. Horne, and C. L. Giles (1998). “How embedded memory in recurrent neural network architectures helps learning long-term temporal dependencies,” *Neural Networks*, vol. 11, no. 5, pp. 861–868, 1998.

[35] M. Bodén and J. Wiles (2000). “Context-free and context-sensitive dynamics in recurrent neural networks,” *Conn. Sci.*, vol. 12, no. 3–4, pp. 197–210, 2000.

[36] G. F. Marcus (1998). “Can connectionism save constructivism?,” *Cognition*, vol. 66, no. 2, pp. 153–82, May 1998.

[37] S. Hochreiter and J. Schmidhuber (1997). “Long short-term memory,” *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[38] P. J. Werbos (1990). “Backpropagation through time: What it does and how to do it”. *Proceedings of the IEEE*, 78 (10), 1550-1560.

[39] N. Cowan (2001). “The magical number 4 in short-term memory: A reconsideration of mental storage capacity”. *Behavioral and Brain Sciences*, 24:87–185.

[40] A. Clark (2013). “Whatever next? Predictive brains, situated agents, and the future of cognitive science”. *Behavioral and Brain Sciences* (36):181–204.doi: 10.1017/S0140525X12000477

[41] P. F. Dominey, M. Arbib., and J. P. Joseph (1995). “A model of corticostriatal plasticity for learning oculomotor associations and sequences”. *Journal of Cognitive Neuroscience*, vol 7 no 25.

[42] P. F. Dominey (2013). “Recurrent temporal networks and language acquisition – from corticostriatal neurophysiology to reservoir computing”. *Frontiers in Psychology*, vol 4, part 500, doi: 10.3389/fpsyg.2013.00500.

[43] X. Hinaut., and P. F. Dominey (2013). “Real-time parallel processing of grammatical structure in the frontostriatal system: a recurrent network simulation study using reservoir computing”. *PLoS ONE* 8(2). e52946. doi: 10.1371/journal.pone.0052946

[44] R. Pascanu., and H. Jaeger (2011). “A neurodynamical model for working memory”. *Neural Networks*, vol. 24, 199–207. doi: 10.1016/j.neunet. 2010.10.003.

[45] A. D. Friederici (2012). “The cortical language circuit: from auditory perception to sentence comprehension”. *Trends Cogn. Sci*. 16, 262-268. doi:10.1016/j.tics.2012.04.001.

[46] H. T. Siegelmann., and E. D. Sontag (1991) “Turing computability with neural nets”. *Applied Mathematics Letters*, 4, 77-80.

[47] H. M. Christiansen., and M. C. MacDonald (2009) “A usage-based approach to Recursion in Sentence Processing”. *Language Learning*, 126-161, ISSN 0023-8333.

[48] N. Evans., and S. C. Levinson (2009) “The myth of language universals: Language diversity and its importance for cognitive science”. *Behavioral and Brain Sciences* (32):429–492.doi: 10.1017/S0140525X0999094X.