Available online at www.sciencedirect.com

ScienceDirect

Neural Networks 20 (2007) 424–432

Neural Networks

www.elsevier.com/locate/neunet

2007 Special Issue

Learning grammatical structure with Echo State Networks

Matthew H. Tong[a,*], Adam D. Bickett[a], Eric M. Christiansen[b], Garrison W. Cottrell[a]

[a] *Department of Computer Science and Engineering, University of California at San Diego, 9500 Gilman Drive, Dept 0404, San Diego, CA 92093-0404, USA*
[b] *Swarthmore College, 500 College Avenue, Swarthmore, PA 19081, USA*

Abstract

Echo State Networks (ESNs) have been shown to be effective for a number of tasks, including motor control, dynamic time series prediction, and memorizing musical sequences. However, their performance on natural language tasks has been largely unexplored until now. Simple Recurrent Networks (SRNs) have a long history in language modeling and show a striking similarity in architecture to ESNs. A comparison of SRNs and ESNs on a natural language task is therefore a natural choice for experimentation. Elman applies SRNs to a standard task in statistical NLP: predicting the next word in a corpus, given the previous words. Using a simple context-free grammar and an SRN with backpropagation through time (BPTT), Elman showed that the network was able to learn internal representations that were sensitive to linguistic processes that were useful for the prediction task. Here, using ESNs, we show that training such internal representations is unnecessary to achieve levels of performance comparable to SRNs. We also compare the processing capabilities of ESNs to bigrams and trigrams. Due to some unexpected regularities of Elman's grammar, these statistical techniques are capable of maintaining dependencies over greater distances than might be initially expected. However, we show that the memory of ESNs in this word-prediction task, although noisy, extends significantly beyond that of bigrams and trigrams, enabling ESNs to make good predictions of verb agreement at distances over which these methods operate at chance. Overall, our results indicate a surprising ability of ESNs to learn a grammar, suggesting that they form useful internal representations without learning them.
© 2007 Published by Elsevier Ltd

Keywords: Echo state networks; Simple recurrent networks; Grammar learning

1. Introduction

An Echo State Network (or ESN) is a type of three-layered recurrent network with sparse, random, and crucially, untrained connections within the recurrent hidden layer. The scale of the internal connections is set so that the networks possess the Echo State Property: if given a long enough sequence, the network will always end up in the same state, regardless of the starting state. In other words, its internal state "echoes" the input sequence. The activation of the hidden units (also known as the echo layer or reservoir) is a product of the dynamics created by the internal random weights and the input, which are connected to the echo layer through weights that are also sparse, random, and untrained.[1] Given this setup, only one layer of weights must be trained, namely those leading to the output. These can be trained using any applicable fitting procedure, removing the need for lengthy, iterative training. Echo State Networks have been shown to be effective in a variety of domains (Eck, 2006; Jaeger, 2001, 2002; Salmen & Plöger, 2005). However, we could find no prior work in applying ESNs to natural language tasks, although work concurrent with ours (Frank, 2006a) begins to address some of these issues.

Elman's Simple Recurrent Networks (SRNs) are commonly used to model language oriented tasks. SRNs have an identical architecture to the ESNs just described, with two major differences. First, the input and internal connections are dense (usually complete connectivity is used) and all of the weights are trained, using truncated backpropagation through time (Williams, 1989). Thus, the main difference between SRNs and ESNs is the use of learning to develop internal representations in the service of a particular task. The basic architecture of the two networks is depicted in Fig. 1.

The task described in (Elman, 1991) is to predict the next word in a corpus of sentences given the previous words.

* Corresponding author. Tel.: +1 858 534 5948.
E-mail addresses: mhtong@ucsd.edu (M.H. Tong), abickett@ucsd.edu (A.D. Bickett), echrist1@swarthmore.edu (E.M. Christiansen), gary@ucsd.edu (G.W. Cottrell).

[1] Some versions also allow feedback from the output layer. These are not used in our work.

0893-6080/$ - see front matter © 2007 Published by Elsevier Ltd
doi:10.1016/j.neunet.2007.04.013

Table 1
Grammar used for training and test sets

S→NP VP "."
NP → PropN | N | N RC
VP → V | V NP
RC → who NP V | who VP
N → boy | girl | cat | dog | boys | girls | cats | dogs
PropN → john | mary
V → chase | feed | see | hear | walk | live | chases | feeds | sees | hears | walks | lives
With these additional restrictions:

- *Number agreement between N & V within a relative clause and between the head N and subordinate V.*
- *Verb arguments*:
  - *Hit, feed*: *require direct object* (*VP→V NP*)
  - *See, hear*: *optionally allow direct object* (*VP → V | V NP*)
  - *Walk, live*: *preclude direct object* (*VP→V*)
- *In the rule RC→who NP V, the Verb in the V must allow a direct object*

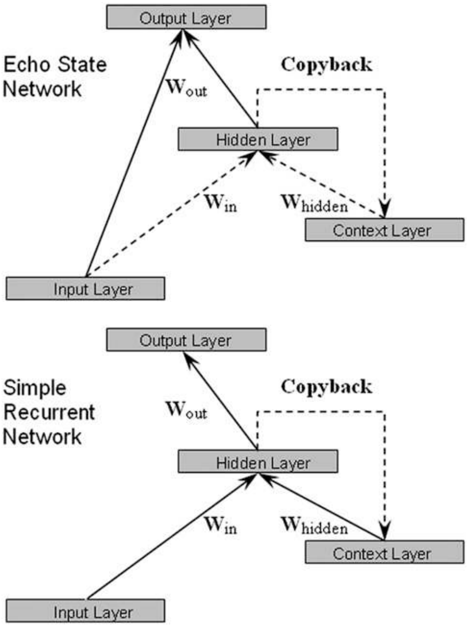

Fig. 1. Network architectures used in the experiments. Connections between layers are displayed as arrows, with fixed connections displayed as dotted lines and trained connections as solid lines. The recurrent connections are often presented as a connection between the hidden layer and itself, but showing the context layer (a copy of the state of the hidden layer at the previous time step) makes the learned connections more explicit and the connection between the two networks more clear. Labels on connections correspond to the weight matrices are described in Section 3.1.

Because the next word is not completely determined, the best a network can do is to output a probability distribution over the possible next words. Outputting the correct probability distribution in each context inherently requires sensitivity to the underlying grammar. For his work, Elman defined a recursive grammar for a subset of English that included subject–verb agreement and relative clauses (see Table 1). Simple Recurrent Networks achieve what he describes as a "high level of performance in prediction" on this task (Elman, 1991). The network's predictions reflect sensitivity to subject–verb agreement across intervening relative clauses, the gap structure of relative clauses, and the transitivity of the verbs. Interestingly, the network was unable to achieve this performance without a staged training procedure, which began with simple sentences, and gradually added more complex sentences to the training set, which Elman related to memory limitations during development (Elman, 1991, 1993).

The primary goal in this work is to assess whether an Echo State Network is able to perform comparably to an SRN on a natural language task, despite the drastic differences in training procedures. In particular, it is of interest whether the ESN can perform a language task that requires a significant amount of memory and generalization. Elman's task is a simplified natural language problem that nevertheless addresses several key aspects of English grammar, and hence a network that cannot alter its internal representations might be expected to fail. In particular, Elman's analysis of the internal representations of his network shows dimensions of the state space that are sensitive to the number of the subject, the transitivity of the verb, and the relative clause structure of the sentence. His analysis also shows that the network only stores what is required to perform the prediction task (such as the number of the subject), and only for as long as it is required by the task (up until the agreeing verb is reached). Elman sees the ability to learn such linguistically-sensitive internal representations as crucial for performing the task. Here, we will show that learning such internal representations is not as crucial as one might think.

2. Related work

In work concurrent with that presented here, Frank (2006a) also uses SRNs and ESNs to process a subset of the English language. However, this work focused on demonstrating that recurrent networks can handle weak systematicity, defined as the ability to process sentences containing novel combinations of words. The grammar used was a small regular grammar, with three basic sentence forms with at most one recursive relative clause. While the paper demonstrates the ability of SRNs and ESNs to correctly process novel token strings, it does not address the recursive nature of language or the necessity of tracking long term dependencies. In contrast, our work places little emphasis on weak systematicity, but uses a context-free grammar with dependencies that span a significant amount of intervening material (such as verb agreement in the context of subject relative clauses) to test whether the memory capacity of ESNs is sufficient to have potential in language modeling. Our context-free grammar is bounded by a maximum sentence length of 11 words, and could technically be expressed as a regular grammar; however, the multiple layers of recursion (up to three), variations in verb transitivity, and agreement dependencies would make the number of rules prohibitive. We also chose to emphasize a direct comparison with Simple Recurrent Networks, using the same grammar and task as (Elman, 1991), which established the usefulness of Simple Recurrent Networks in natural language. A later work (Frank, 2006b) looked at ESNs trained on a similar grammar to (Elman, 1991), including dependencies and recursion. However, the assessment examines strong systematicity using probe sentences with one layer of recursion. Our work is therefore unique in measuring performance over multiple layers of recursion with dependencies spanning large distances. Both works by Frank use a variation on the basic ESN architecture, with linear units in the reservoir and a hidden layer between the echo state layer and the output layer. Frank suggests that the hidden layer is necessary in order to achieve the types of generalization involving the type of systematicity that he tests for. We find that this additional layer is not necessary for our task. The main difference from Frank's work is that our training and testing grammars are identical, even though the testing set contains entirely novel sentences.

3. Methods

3.1. Network architectures

Echo State Networks (Jaeger, 2001) are recurrent neural networks, typically with three layers: an input layer, a recurrent hidden layer, and an output layer. Echo State Networks distinguish four sets of weights: $\mathbf{W}_{\text{in}}$, weights from the input layer to the hidden layer, $\mathbf{W}_{\text{hidden}}$, the recurrent connections from the previous state of the hidden layer to the current hidden layer, $\mathbf{W}_{\text{back}}$, recurrent connections from the previous output to the hidden layer, and $\mathbf{W}_{\text{out}}$, the connections from the input and hidden layers to the output layer. Depending on the task, the back connections in $\mathbf{W}_{\text{back}}$ are often omitted; we have done so here, as at any point in time the previous time step's output units are only a noisy version of the current step's inputs. Typically, either tanh or linear activations are used, with training significantly simplified if output units are linear, or if the outputs can be assumed to be in the linear range of the tanh function. Here we used a tanh activation function for the hidden units and linear output units. We describe the activation of the units at time step t by the vector $\mathbf{A}(t)$. Computation of the network can then be defined as:

$$\mathbf{A}_{\text{hidden}}(t) = \tanh(\mathbf{W}_{\text{in}}(\mathbf{A}_{\text{in}}(t)) + \mathbf{W}_{\text{hidden}}(\mathbf{A}_{\text{hidden}}(t-1)))$$
$$\mathbf{A}_{\text{out}}(t) = \mathbf{W}_{\text{out}}(\mathbf{A}_{\text{in}}(t), \mathbf{A}_{\text{hidden}}(t)).$$

All weights except $\mathbf{W}_{\text{out}}$ are fixed prior to training with random values chosen to ensure that there will be echo states. The Echo State Property (the defining property of Echo State Networks) states that given a long (approaching infinite) string of input, the hidden state of the network will be uniquely determined by the input history. In practice, the existence of echo states is achieved by giving random, sparse connections to the hidden layer ($\mathbf{W}_{\text{in}}$ and $\mathbf{W}_{\text{hidden}}$) and scaling the recurrent connections ($\mathbf{W}_{\text{hidden}}$) to have a spectral radius less than one. The magnitude of the spectral radius determines the persistence of memory; the closer the spectral radius is to one, the longer the history that determines the current hidden state. To scale the initial weights $\mathbf{W}'_{\text{hidden}}$ to a desired spectral radius α ($0 < \alpha < 1$), we calculate $\mathbf{W}_{\text{hidden}}$ as:

$$\mathbf{W}_{\text{hidden}} = \frac{\alpha \mathbf{W}'_{\text{hidden}}}{|\lambda_{\max}|}$$

where $\lambda_{\max}$ is the maximum eigenvalue of $\mathbf{W}'_{\text{hidden}}$.

Once these weights have been assigned, the network may be trained. Since the connections to the hidden layer ($\mathbf{W}_{\text{in}}$ and $\mathbf{W}_{\text{hidden}}$) are fixed and do not depend on the outputs of the network, the state of the hidden layer is independent of the task facing the network, and is entirely driven by the input and the network's internal dynamics. During training, the hidden layer is first presented with a series of inputs to allow its internal dynamics to settle. To accomplish this, we used a full pass over the training set, during which the weights are not trained. Once the network is thus primed, the hidden layer is observed as it is presented with the training set. Training consists only of setting $\mathbf{W}_{\text{out}}$ to map these states to the desired outputs; if the output unit activations are linear or approximately linear, a simple linear regression can be used to train these connections. This frees the Echo State Network from the lengthy training process associated with iterative methods such as backpropagation.

The hidden layer of the Echo State Networks we use in this work consisted of sigmoidal (tanh) units, and ranged in size from 50 to 1000 units. Internal weights ($\mathbf{W}_{\text{hidden}}$) were non-zero with a probability of .27, and were scaled to have a spectral radius of .98. Weights from the input to the hidden layer ($\mathbf{W}_{\text{in}}$) were non-zero with a probability of .2 and were uniformly sampled from $[-.5, .5]$. These values were obtained by a parameter search on a separate training set, using values reported in (Jaeger, 2001) as a guide. Overall, we found that adjustment of the weight parameters had surprisingly little effect. Weights from the input and hidden layers to the output layer ($\mathbf{W}_{\text{out}}$) were set using a simple linear regression over the weights using the teaching signal. The network code was based on (Cernansky, 2005). It is worth noting again that without back connections, the internal representations are *solely* a product of the dynamics due to the random recurrent weights and the input, with no influence from the actual prediction task.

In contrast, the Simple Recurrent Network used in (Elman, 1991) learns internal representations specific to the grammar. While the basic structure and operation of an SRN is identical to the ESNs described above, all three weight matrices are trained using backpropagation of error. Since backpropagation in recurrent networks is difficult due to the fact that error can be propagated back to all previous time steps in the series, SRNs take the approach of stopping the propagation at the previous time step. Thus, the gradient used in backpropagation is an approximation to the true gradient, which would require significantly more memory (Williams, 1989). Elman's view is that this makes the training procedure more cognitively plausible, being local in time and space. As shown in Fig. 1, this can be implemented by using a context layer whose contents are the state of the hidden layer at the prior time step. Elman's networks operate by learning an internal representation that facilitates the combination of previous states and input such that a mapping to the desired output is achievable.

In order to compare ESNs with SRNs on the performance measures we used, we trained an SRN on the task. Following (Elman, 1991), the SRN we implemented had 70 hidden units. In some versions of SRNs, additional non-recurrent layers are added between the input and hidden layers and between the hidden and output layers, in order to form distributed representations of the words. We found these additional layers unnecessary to achieve performance comparable to Elman's (1991) results, so they were omitted to maintain architectural similarity with Echo State Networks. The hidden units used tanh activation functions, while the output units used logistic activation functions. Training was performed using Netlab's (Nabley & Bishop, 2003) scaled conjugate gradient descent using the cross entropy error criterion. Elman reported that training in stages was necessary for his SRN to successfully learn its hidden representations and perform the prediction task. We also used staged training here, iteratively training the network on sets comprised of 0%, 25%, 50% and 75% complex sentences. Complex sentences are defined here as sentences containing at least one relative clause. The final training sets were identical to the ones used for the ESN experiments. The resulting network was then tested on the same test sets of 75% complex sentences used for the ESN networks. The datasets are described in detail in Section 3.2 below. Simple Recurrent Networks were given 1000 passes of the data to train their connections, compared to the dual pass (once to settle, once to train) allowed ESNs.

As a performance baseline for the networks, we compare the networks' performance against bigram and trigram models. The bigram, which is simply the empirical conditional probability distribution over the next word given the current word, provides a reference as to whether the networks are able to learn any knowledge of the grammar. Comparison against the trigram, the empirical conditional probability distribution of the next word given the previous two words, tests the networks' incorporation of the memory of the previous time step. The empirical distributions for bigrams and trigrams were created by keeping a count of the occurrences of each pair or triplet, respectively, in the training set, and dividing these counts by the total amount of pairs or triplets. While the size of the training set was sufficient to cover most valid word combinations, the estimates were smoothed by adding one to the count of each word pair and word triplet. The bigram and trigram word probability estimates can be given as

$$P_{\text{bigram}}(w(t+1)=a|w(t)=b)$$
$$=\frac{1+\sum_{s\in\text{train}}1(w(s+1)=a)1(w(s)=b)}{\sum_{i\in\text{words}}\left(1+\sum_{s\in\text{train}}1(w(s+1)=i)1(w(s)=b)\right)}$$

$$P_{\text{trigram}}(w(t+1)=a|w(t)=b,w(t-1)=c)$$
$$=\frac{1+\sum_{s\in\text{train}}1(w(s+1)=a)1(w(s)=b)1(w(s-1)=c)}{\sum_{i\in\text{words}}\left(1+\sum_{s\in\text{train}}1(w(s+1)=i)1(w(s)=b)1(w(s-1)=c)\right)}$$

where $w(t)$ refers to the presented word at time t, $1(\cdot)$ refers to the indicator function, train refers to the set of time steps from the training set, and a, b, and c are words in the language.

3.2. Stimuli

The network stimuli were based on those in (Elman, 1991). Inputs consist of a sequence of sentences, presented one word at a time. Each word is represented in a localist fashion, as a vector of length 24 which is all zeros except for a single one to signify the current word. For example, the word 'boy' is represented as a one followed by 23 zeros. Two words with the same root but which differ in number have different encodings; thus 'boy' and 'boys' are treated as entirely different tokens. The inputs are therefore completely orthogonal and do not encode any grammatical information. The network's target output is the vector corresponding to the following word in the sequence. To scale the activations appropriately for the tanh hidden units in the ESNs and the SRNs (LeCun, Bottou, Orr, & Müller, 1998), the inputs (and target outputs) were shifted by −.5 to be between −0.5 and 0.5. The lexicon of consists of 24 tokens including 10 nouns, 12 verbs, the relative pronoun 'who', and a period. The input sentences were based on the context-free grammar used in (Elman, 1991), which is given in Table 1. The grammar allows for simple relative clauses, requires verbs to agree in number with their respective nouns, and features transitive, intransitive, and optionally transitive verb types.

We developed five sets of data of 10,000 randomly generated sentences. These sets were divided into disjoint training and test sets with 90% of the sentences being used for training, and 10% for testing. Ground truth probabilities were calculated along with the creation of the datasets, reflecting the actual distribution in the grammar over words at each time step. Sentences containing at least one relative clause ("complex sentences") comprised 75% of the corpus. The sentences generated were up to 11 words in length, and had an average of 6.37 words per sentence (just over the 5–6 words used in a sentence with one layer of recursion). Elman's dataset included a few sentences of length 16, but averaged 6.03 words per sentence. As described in 3.1, we allowed for the staged training approach Elman required for SRNs by also generating training sets of 0%, 25%, and 50% complex sentences. These additional training sets were only used to train the SRNs. We report average results on the test set over these five datasets.

4. Performance analysis

The network's output was evaluated by comparing it to the context-dependent likelihood vector for the desired output. Because sentences were generated from sentence schemas that could be enumerated, we calculated the exact probability of

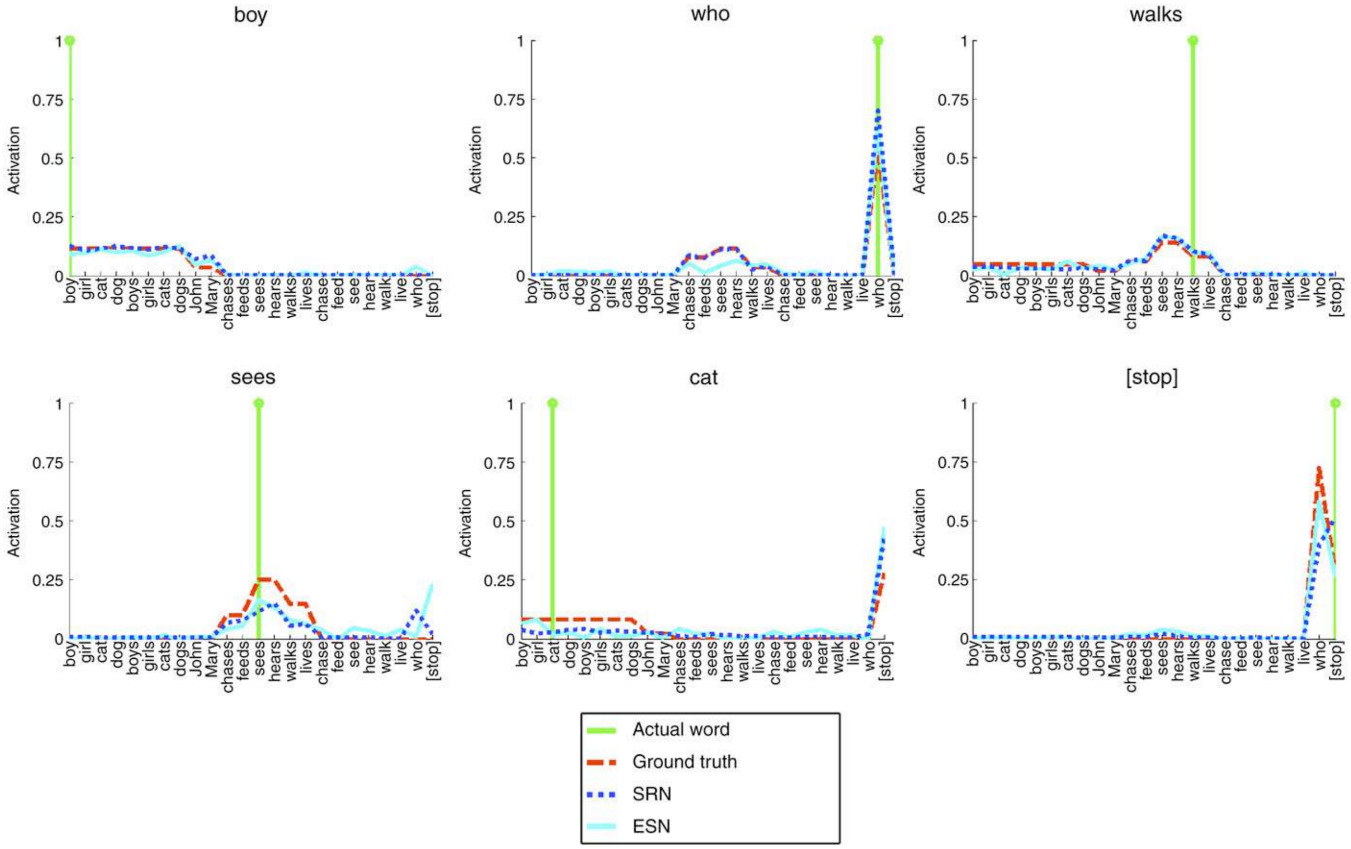

Fig. 2. Network output activations and actual probabilities for the sentence “Boy who walks sees cats” for ESNs and SRNs.

potential next words given the preceding words. This strategy becomes increasingly infeasible as sentence lengths grow, realistically limiting the maximum sentence length to 11 or 12 words (11 was used here). This differs slightly from (Elman, 1991), who used the empirical probabilities of the training corpus.

Fig. 2 shows the actual outputs of an ESN and SRN on the sentence “Boy who walks sees cat.” compared with the actual probabilities. Since we had shifted the targets for the ESNs to the $[-0.5, +0.5]$ range, we first added .5 back to the outputs to convert them back to the [0.0–1.0] range for ease of visualization. Some unit’s outputs were below -0.5, which results in values below 0 after shifting them back. We simply set these to 0 before comparing them to the targets. No units were above 1 after the shift. While these outputs are similar to the actual probabilities, it should be noted that they do not truly act as a probability distribution over the words as they are not guaranteed to sum to one.

4.1. *Metrics*

The primary comparable metric used in (Elman, 1991) was the cosine between the output and the empirical probability vectors, so we also computed this score for our network, only using the actual likelihoods from the grammar. The cosine score between two vectors is defined as

$$\cos(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\| \|\mathbf{y}\|}.$$

Here x and y are our network’s output and the actual probabilities in the grammar, respectively. The reported score is the average cosine over the test set. In addition to providing a metric of comparison between methods, the cosine score also allowed us to check our SRN results against Elman’s results (our SRNs actually outperform Elman’s reported cosine score of 0.852).

Looking at the maximally predicted word for each time step provides a second metric for measuring the success of the network. If this word turns out to be impossible in the grammar given the preceding words in the sentence, the network has made a mistake in understanding the context in which it is operating. While choosing the word that the network is most sure of could be considered a generous error metric, on average approximately two-thirds of the 24 words are impossible at any point in time. Because some tokens have a much higher probability of occurring than others, treating the outputs without normalization skews the results. Periods and ‘who’, for instance, are both unique function words in the grammar that occur very frequently and in some contexts with near or complete certainty. While nouns as a class are quite common, there are also ten possible nouns to choose from in contexts in which a noun can appear, making the actual probability of a particular noun relatively small. With a different network architecture and training procedure, where outputs are forced to be probabilities, the normalization might be unnecessary. However, the architecture may be unable to avoid a skew towards words like period and ‘who’ that are more frequent and have a greater dynamic range. It therefore makes sense to z-score (normalize them to 0 mean and unit standard deviation) the outputs of each output unit over time to provide a better measure of how certain the network is about its prediction, measuring in terms of standard deviations from mean instead of net activation. Z-scoring also benefited SRNs, so it was used there. Since bigrams and trigrams deal in actual context-dependent likelihoods, they do not need normalization. The maximum prediction score is given as

$$\mathrm{score}_{\mathrm{maxpred}} = \frac{\sum_{t\in\mathrm{test}}\sum_{w\in\mathrm{words}} 1(w = \arg\max_{i} A_i(t))1(p_t(w) > 0)}{|\mathrm{test}|}$$

where A_i refers to the activation of the ith output unit and $p_t(w)$ is the ground truth probability of word w at time t.

Treating word prediction as a binary classification problem of possible versus impossible provides a means to assess the accuracy of all predictions at each time step, as opposed to only the most confident prediction. For each time step, the network's output can be thresholded to determine which kinds of words are "possible" and which are "impossible". Z-scored outputs were collapsed into their respective classes (common nouns, proper nouns, 6 classes of verbs, who, and period) by taking the mean activation across units for analysis, as this is the granularity of interest. Then, receiver operator curves (ROC) were formed by sweeping through possible thresholds for this classification, θ. True positives (hits) occur when the mean activation of a class is above a given threshold, while false positives (false alarms) occur when a class is impossible despite the class being above the threshold. Dividing the count of true positives by the number of actually possible classes in the test set yields the hit rate, while the false alarm rate is formed by dividing the false positives by the count of actually impossible classes

$$\text{Hit rate} = \frac{\sum_{t\in\mathrm{test}}\sum_{c\in\mathrm{class}} 1(A_c(t) > \theta)1(p_t(c) > 0)}{\sum_{t\in\mathrm{test}}\sum_{c\in\mathrm{class}} 1(p_t(c) > 0)}$$

$$\text{False alarm rate} = \frac{\sum_{t\in\mathrm{test}}\sum_{c\in\mathrm{class}} 1(A_c(t) > \theta)1(p_t(c) = 0)}{\sum_{t\in\mathrm{test}}\sum_{c\in\mathrm{class}} 1(p_t(c) = 0)}.$$

Plotting these two rates yields an ROC curve, with the area under the curve being an indication of performance; an area of 1 would signify perfect separability of possible and impossible cases, while an area of 0.5 would show that the networks show no ability to discriminate. This measure also allowed us to examine the performance on each class of words individually.

One of the dependencies the network must maintain is the agreement in number of a verb with its subject. For example, given a subject of 'John', 'sees' is a possible verb while 'see' is not. The distance between a subject and verb can be arbitrarily large, as in the case of "Cats who dogs who like John chase see". When a verb appears, the mean of the outputs corresponding to the verbs of the correct number should be higher than the outputs for verbs of the incorrect number. If this is not the case, a verb agreement error has occurred. The verb agreement rate is the fraction of the total number of verbs for which the networks selected the correct number. The verb agreement score is then

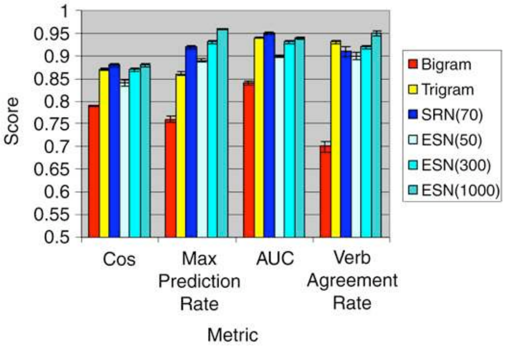

Fig. 3. Summary of results. For ESNs and SRNs, the parenthesized number signifies the number of units in the hidden layer. The cosine score was the cosine of the angle between a network's predictions and the ground truth probabilities. Max prediction rate is the rate at which the most strongly predicted word is possible in the grammar. AUC is the area under the ROC generated by determining whether a class of words is possible or impossible. Verb agreement measured whether predicted verbs agreed in number with their corresponding noun. See Section 4.1 for further explanation of the metrics used.

$$\mathrm{score}_{\mathrm{agreement}} = \frac{\sum_{t\in\mathrm{test}} 1(w(t) \in \mathrm{verb})1(\overline{A}_{\mathrm{correctverb}}(t)) > (\overline{A}_{\mathrm{incorrectverb}}(t))}{\sum_{t\in\mathrm{test}} 1(w(t) \in \mathrm{verb})}$$

where $\overline{A}_{\mathrm{correctverb}}$ and $\overline{A}_{\mathrm{incorrectverb}}$ refer to the average activation of units corresponding the correct and incorrect verb classes respectively.

4.2. Results

A Simple Recurrent Network with 70 hidden units has approximately the same number of trained connections as an Echo State Network with 300 hidden units. At this level, Simple Recurrent Networks score higher than Echo State Networks by the cosine and AUC metrics, while ESNs outperform SRNs in max prediction rate and verb agreement. These results are displayed in Figs. 3 and 4. However, the differences in performance for cosine and AUC were highly significant ($p < .01$), while the gains by ESNs in max prediction rates were only marginally significant ($p < .05$) and insignificant in verb agreement.

However, adding additional units to an Echo State Network comes cheaply, because trained connections scale linearly with the number of hidden units. In contrast, trained connections in a Simple Recurrent Network scale quadratically, and large hidden layers slow training down substantially. As the number of units in the Echo State Network increased, ESNs showed significant improvement by all four metrics. While SRN scores on the cosine and AUC metrics remained higher, the difference between the two diminished, with the difference on the cosine ceasing to be significant when ESNs had 400 or more hidden units. The advantages of ESNs on verb agreement and max prediction continued to increase as well, becoming highly significant.

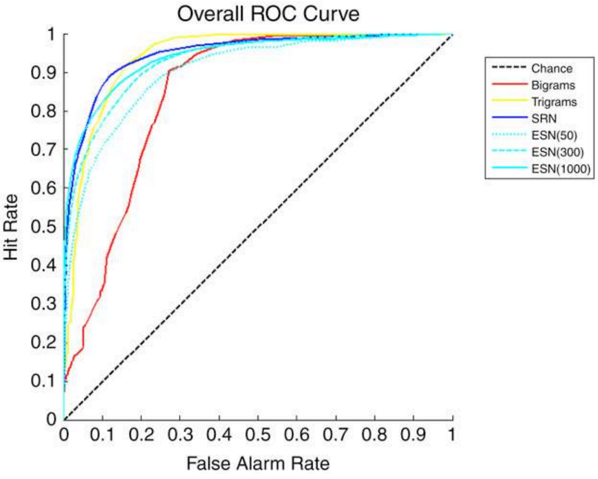

Fig. 4. Overall ROC. The parenthesized number for the two types of networks denotes the number of units contained in the network's hidden layer. The overall ROC views the task of word prediction as a classification task of a class of words being impossible in a given context. A category of words is classified as impossible if the mean activation of its normalized output units is below threshold across words in a class. These classifications are compared with the ground truth probabilities to determine whether the classification was correct.

Some insight into why SRNs perform better on two metrics can be gained by comparing their output units and training regimes. SRNs were trained using a cross-entropy error metric and used sigmoidal output units, while ESNs minimized squared error and used linear outputs. These differences meant that SRNs' outputs more closely resembled actual probabilities, likely improving their cosine score. In addition, the sigmoidal outputs allow and encourage an SRN to modify its connections into output units to drive net inputs strongly negative and bring the output to zero. This increases the effective distance between impossible outputs and outputs that were possible with low probability. The linear nature of our ESN's output layer constrains the net input into the output units of impossible and slightly possible words to be close together, and penalizes equally for straying above and below zero for impossible words.

The AUC was also calculated for 6 categories of interest (averaging AUC scores for proper and common nouns and for the singular and plural forms of each class of verbs), with the results displayed in Table 2. Both ESNs and SRNs are at near perfect performance for periods, which is of particular interest since knowing when a sentence can end is among the most memory intensive of tasks. Both networks did worst at predicting intransitives, likely because they are more likely than other verbs to have dependencies spanning large amounts of intervening material. The fact that ESN's performance on intransitives was particularly impacted is surprising and warrants further study.

As expected, both networks did substantially better than bigrams by all metrics. The networks' performance comparison with trigrams was a different story, however. Fundamentally, a comparison with trigrams enables us to examine how well a network is able to store information from the previous time step's input in its recurrent hidden layer. While results both here and in the following section show that both networks are capable of this, they tend to be more prone to noise than trigrams, because trigrams explicitly store the empirical probabilities conditioned on the previous input. Therefore, for cases where a memory of the previous two words is sufficient to provide the probabilities of the next word, trigrams tend to perform better than either network. As will be further discussed in the next section, these cases cover a large amount of Elman's grammar. As shown in Fig. 3, its results on three metrics are comparable with both networks (although ESNs are able to significantly outperform it on all but AUC with a hidden layer with at least 750 units). Its performance on the max prediction metric, however, shows that both ESNs and SRNs have memories that exceed the last two words. For instance, when the previous two words are of the form [$who\ V_{\text{intransitive}}$] the next word could either be a period (*Dogs chase cats who live.*) or a verb (*Dogs who live chase cats.*). Another common example of this is the form [$N V_{\text{tran}}$], where the next word can either be a noun (*Dogs chase cats.*), a period (*John feeds cats who dogs chase.*), or a verb (*Cats who dogs chase see John.*). In these cases, trigrams are forced to guess. Inspection of the predicted words confirms that trigrams make many errors in the cases where a two word memory is insufficient, many more than either kind of network. This demonstrates that the networks must be able to access longer memories with some reliability.

Table 2
Summary of AUC scores for 6 word classes of interest

	Nouns	Transitive verbs	Optionally transitive verbs	Intransitive verbs	Who	Stop
Bigram	0.93	0.84	0.81	0.78	1.00	0.84
Trigram	0.93	0.96	0.96	0.93	1.00	0.94
SRN(70)	0.96	0.96	0.97	0.94	1.00	0.98
ESN(50)	0.95	0.91	0.90	0.80	1.00	0.94
ESN(300)	0.96	0.94	0.94	0.86	1.00	0.97
SRN(1000)	0.95	0.96	0.97	0.89	1.00	0.99

The parenthesized number for the networks refers to the size of the hidden layer of the network. Scores for singular and plural forms of verbs were averaged for ease of reading, but were calculated separately. The ROC was calculated by making a classification based on the mean activation of units in a category. Classifications are then compared against the ground truth probabilities to assess performance.

4.3. Verb agreement over distances

In order to further investigate the memory of the networks, we focused on the ability of the networks to track verb agreement over intervening material. Verb agreement errors provide one of the best measures of whether a system has learned the grammar. The verb agreement error described above can be parceled out by distance, showing how performance varies as demands on memory increase. Unfortunately, the grammar actually offers additional information, resulting in bigrams and trigrams performing better than expected and reducing the difficulty of this task. Table 3 shows how memory demands vary as distance increases, and provides example sentences. Table 4 shows the error rates of the various systems on this task as a function of the distance from the agreeing noun. There is no entry for "5" or "8" because of a quirk of this grammar — no verb can be 5 or 8 words away from its corresponding noun. Based on the structures in Table 3,

Table 3
Sample sentences of varying distances between a verb and its corresponding noun

Distance	Memory required	Example sentence
1	1	***Dogs*** **live**.
2	2	***Dogs*** who **walk** live.
3	1	Dogs who ***walk*** **live**.
4	2	Dogs who ***chase*** cats **live**.
4	4	***Dogs*** who cats chase **live**.
6	4	Dogs who ***chase*** cats who walk **live**.
6	6	***Dogs*** who boys who walk feed **live**.
7	5	Dogs who ***chase*** boys who feed cats **live**.
7	7	***Dogs*** who boys who feed cats chase **live**.
9	7	Dogs who ***chase*** cats who boys who walk feed **live**.
9	9	***Dogs*** who cats who boys who run feed see **live**.

Distances of 5 and 8 do not appear in the grammar and are therefore not shown. The specifics of the grammar allow information on number to be accessible in some instances at a shorter distance than the full distance between noun and verb, reducing the memory required. In the provided example sentence, the target verb is given in **bold** and closest word determining its number is given in ***italics***.

bigrams and trigrams are able to perform perfectly at a distance of 3, and trigrams perform at half of chance at distance 4. Approximately 86% of verbs in the corpus have their number uniquely determined by the previous two words.

Simple Recurrent Networks and Echo State Networks with 50 units perform approximately as well as trigrams here, although ESNs in particular appear to have a noisy memory leading to more errors. Additional hidden units significantly increased the memory span of the ESN, such that with a comparable number of connections to SRNs (300 hidden units), ESNs demonstrate performance indicative of a longer term memory. When the hidden layer is expanded to 1000 units, the ESN performs significantly better than chance for all distances. However, the performance at a distance of nine words requires some qualification; sentences with dependencies of this length are rare, and this small sampling creates a statistical irregularity in two of our data sets whereby the previous two words bias the probabilities towards a performance above chance. This is reflected in the better than chance performance of trigrams at this distance. Since the differences in performance between trigrams and 1000 unit ESNs at this distance is not significant, it would be presumptuous to claim a memory of length seven for ESNs (although it is worth noting that SRNs do not show this boost in performance at this distance). However, ESNs do show performance indicating a memory of at least five, as they perform above chance at a distance of seven words. This ability to maintain information over a long range is a strong indication that the ESN is capturing essential aspects of the grammar.

4.4. Discussion

The results demonstrate that ESNs have the ability to learn to be sensitive to grammatical structure. The overall performance is comparable with the results of (Elman, 1991, 1993), yet, unlike Simple Recurrent Networks, ESNs perform this task without specialized learned internal representations. Given the prominence that such learned internal representations enjoy in the literature, this is a surprising result. ESNs compensate for this inability to learn useful representations with a large reservoir of input-driven random dynamics that capture some of the statistical regularities of the input.

Elman (1991, 1993) argues strongly that starting with small, simple data sets aids in learning more complex ones. His SRNs made use of this mechanism to enable learning a data set which initially they were unable to learn by slowly ramping up the complexity of the stimuli. The one-phase learning of ESNs is incapable of making use of such a staged learning mechanism. Nevertheless, we show here that such a mechanism is, at least in this case, not essential.

Our comparison with the statistical baselines of bigrams and trigrams allowed a partial quantification of the memory of these networks. The results of the max prediction error and the verb agreement over distances show that both kinds of networks have memory exceeding that of trigrams. Adding units to the hidden layer increases the capacity of networks' memory (a performance improvement with additional units is also reported in (Frank, 2006a)). However, increasing the size of the hidden layer scales substantially better for ESNs than SRNs; trained parameters are linear in the hidden layer size, not quadratic.

One contribution of this work is to partially account for how large numbers of randomly wired neurons can participate in complex structured behavior. One of the mysteries of human cognition is how the brain succeeds in "wiring" itself with limited innate structure. Showing that even completely random connections are nevertheless useful in producing a

Table 4
Verb agreement error by distance between a verb and its corresponding noun

Distance	% of verbs	Bigram	Trigram	SRN(70)	ESN(50)	ESN(300)	ESN(1000)
1	28.03	0.00	0.00	0.00	0.02	0.04	0.03
2	38.15	0.53	0.00	0.00	0.00	0.00	0.00
3	14.47	0.00	0.00	0.00	0.02	0.00	0.00
4	9.93	0.48	0.23	0.39	0.47	0.29	0.08
6	5.66	0.51	0.47	0.48	0.53	0.40	0.26
7	2.34	0.51	0.50	0.45	0.48	0.46	0.43
9	1.42	0.54	0.42	0.57	0.47	0.61	0.36

The size of the hidden layer in the networks is given by the parenthesized number following their name. Entries in the table that signify verb agreement errors that are not statistically significantly better than chance are shaded. No sentences with distances of 5 and 8 words are allowed by the grammar. The percentage of verbs occurring at a given distance from their corresponding noun in the corpus is also given.

facility as uniquely human as natural language serves to demonstrate the power of random connections and diminish the need to appeal to innate structure. To further the biological connection, it may prove worthwhile to explore Liquid State Machines (Maass, Natschläger, & Markram, 2002), which employ similar untrained recurrent hidden layers but use more realistic spiking dynamics and connection structures based on cortical parameters.

In future work, we intend to pursue a more detailed analysis of the data collected so far. From a cognitive modeling perspective, we are interested in the sorts of errors the two systems tend to make. The issue is whether SRNs make more "cognitively plausible" errors than ESNs. The comparison with SRNs indicate that the current ESNs may be suffering unnecessarily due to their use of linear output units by making improbable and impossible words difficult to differentiate. Using a different output activation function and training method would not fundamentally change the advantages of ESNs, but may significantly improve performance. Finally, the language generated by the grammar is both too complex, resulting in center-embedded sentences no human can easily parse, as well as too simple, spanning a tiny vocabulary and a small portion of the grammar. Elman's grammar is also not ideal for investigating the ability of a network to learn grammars with long range dependencies because even systems with minimal memory (bigrams and trigrams) can capture a large amount of the grammatical structure. The results presented here and in (Frank, 2006a, 2006b) are insufficient to indicate whether ESNs are capable of scaling to a large natural language corpus. However, because their parameters scale linearly with network size, ESNs do show some promise in this area.

5. Conclusions

Echo State Networks succeeded in producing the probabilities of the next word in complex sentences. It is impossible to determine at this time whether these results can scale up to full natural human language. However, the results suggest that they are capable of performing the task using a corpus that Elman designed to answer questions he considers fundamental to modeling language: "What is the nature of linguistic representations?", "How can complex structural relationships such as constituency be represented?", and "How can the apparently open-ended nature of language be accommodated by a fixed-resource system?" (Elman, 1991). ESNs and SRNs, both fixed-resource systems, were capable of accommodating the subset of natural language used here. By applying an Echo State Network to this task, we were able to show that the first two questions, at least for this corpus, may be answered by the words "random" and "randomly". Furthermore, ESNs achieve performance comparable to that of SRNs without the staged learning procedure necessary for SRNs.

What makes ESNs particularly interesting is that the hidden layer is driven completely by the input and the dynamics generated by randomly weighted connections. While SRNs function by having the hidden layer learn to map functionally identical states into the same region of the representational space, ESNs are incapable of this kind of learning at the hidden layer. Surprisingly, they don't seem to require it.

Acknowledgments

The authors wish to thank Cynthia Taylor for her contribution to earlier versions of this work, Gary's Unbelievable Research Unit, and the UCSD AI research group for helpful feedback. This work was supported on NSF grant DGE-0333451 to GWC.

References

Cernansky, M. (2005). Echo State Network simulator. http://www2.fiit.stuba.sk/~cernans/main/download.html.

Eck, D. (2006). Generating music sequences with an Echo State Network. In *NIPS 2006 workshop on Echo State Networks and liquid state machines*.

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

Elman, J. L. (1993). Learning and development in neural networks: The importance of starting small. *Cognition*, *48*, 71–99.

Frank, S. L. (2006a). Learn more by training less: Systematicity in sentence processing by recurrent networks. *Connection Science*, *18*, 287–302.

Frank, S. L. (2006b). Strong systematicity in sentence processing by an Echo State Network. In *Proceedings of ICANN 2006* (pp. 505–514).

Jaeger, H. (2001). The echo state approach to analysing and training recurrent neural networks. *Technical report GMD report 148*. German National Research Center for Information Technology.

Jaeger, H. (2002). A tutorial on training recurrent neural networks. Covering BPTT, RTRL, EKF, and the Echo State Network Approach. *GMD report 159*. German National Research Center for Information Technology.

LeCun, Y., Bottou, L., Orr, G. B., & Müller, K. -R. (1998). Efficient backprop. In G. E. Orr, & K. -R. Müller (Eds.), *Neural networks: Tricks of the trade*. Berlin: Springer.

Maass, W., Natschläger, T., & Markram, H. (2002). Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Computation*, *14*(11), 2531–2560.

Nabley, I., & Bishop, C. (2003). Netlab. http://www.ncrg.aston.ac.uk/netlab/index.php.

Salmen, M., & Plöger, P. G. (2005). Echo State Networks used for motor control. In *Proceedings of the 2005 IEEE international conference on robotics and automation* (pp. 1953–1958).

Williams, R. J. (1989). Complexity of exact gradient computation algorithms for recurrent neural networks. *Technical report NU-CCS-89-27*. Boston: Northeastern University, College of Computer Science.