Using Echo State Networks to Count without a Counter[*]

June 16th, 2010

Peter Lubell-Doughtie[†]
University of Amsterdam
lubell@science.uva.nl

ABSTRACT

Echo state networks are a class of recurrent neural networks containing a set of nodes and recurrent connections not subject to adjustment during training and thereby greatly reducing the amount of time needed for training and the size of the solution space. Much work has been done applying this relatively new type of network to common problems in parallel distributed processing, such as past tense prediction and sentence processing. We continue this and apply echo state networks to the problem of counting without a counter. We find that echo state networks are successfully able to learn to count by predicting the length of input in a context free grammar. We find that networks with smaller reservoirs produce more general solutions, which perform better on test data.

1. INTRODUCTION

A general problem in the training of recurrent neural networks (RNNs) is that, because the weights between each node in the network and all the nodes it is connected to are variable, the solution space is enormous and the possibility of finding a local minimum error value, as opposed to a global minimum error value, is significant. The RNN method of reservoir computing seeks to alleviate this problem by holding the weights in a portion of the RNN fixed, this portion is called the reservoir, and only training weights on the output nodes, and perhaps the input and bias nodes. This approach is shown in Figure 1 which diagrammatically compares the network configurations and dynamics in a standard RNN, on the left, and a neural network using reservoir computing, on the right. The standard RNN updates all the weights on each iteration, while the reservoir computing based network updates only the output weights.

Benefits of the reservoir computing approach include that it is computationally universal (Maass et al. [2006]) and that it is easily extendible. Concerning extendability, if we were using a standard recurrent neural network, had trained it to perform some specific task, and now want to train it to perform a different but perhaps related task, we would have to retrain the entire network, potentially modifying each connection during training and making little use of the current network, beyond providing initialization values. Alternatively, if we had trained a reservoir computing based network to perform some specific task and are now retraining it to perform a different but perhaps related task, we could maintain the trained network and outputs with their parameters as is and then simply add an additional output layer with adjustable weights. In the first method, using simple recurrent networks, any representations already learned by the network will likely be destroyed as the network adjusts its weights to solve the new task. In contrast, using the reservoir computing method we maintain the representations already learned by the network and are able to learn an additional new representation solving the new task.

A further benefit of reservoir computing stems from the biological plausibility of the training method. The weights between neural connections within the brain do not adjust themselves according to the highly dynamic fashion exhibited by simple recurrent networks. Static or much more stable connection weights, as used in reservoir computing, are closer to the function of the brain. Additionally, the design – the neural connection topology – of the reservoir can be built to mimic the actual cortical networks of the brain. Designing reservoirs to approximate the actual construction of the brain has been shown to improve performance in Haeusler and Maass [2007].

1.1 Echo state networks

The two most often used approaches to reservoir computing are liquid state machines, as first reported in Maass et al. [2002], and echo state networks, as first reported in Jaeger [2001]. Liquid state machines are designed to offer a biologically realistic model of neural interaction whereas echo state networks are aimed more towards applications. Here we will concern ourselves only with echo state networks.

The defining characteristic of an echo state network is that it must satisfy the echo state property: the effect of a previous state $\mathbf{x}(n)$ and a previous input $\mathbf{u}(n)$ on a future state $\mathbf{x}(n+k)$ should gradually vanish over time, i.e. as $k \to \infty$. Whether or not the echo state property holds for a reservoir, $\mathbf{W}$, is related to the value of the largest absolute eigenvalue or spectral radius, $\rho(\mathbf{W})$, of the reservoir. In most cases if $\rho(\mathbf{W}) < 1$ then $\mathbf{W}$ has the echo state property, however this is not an implication of $\rho(\mathbf{W})$ being less than 1. In general a good reservoir can be created by setting $\rho(\mathbf{W})$ less than but close to 1.

The topology of the reservoir used by the echo state network has a significant impact on the function of the network and its ability to predict signals. Therefore, how to create

[*]Written for Neural Networks and Symbolic Reasoning, Professor Reinhard Blutner, 2nd Semester 2009-2010

[†]UvA Net ID: 6095445

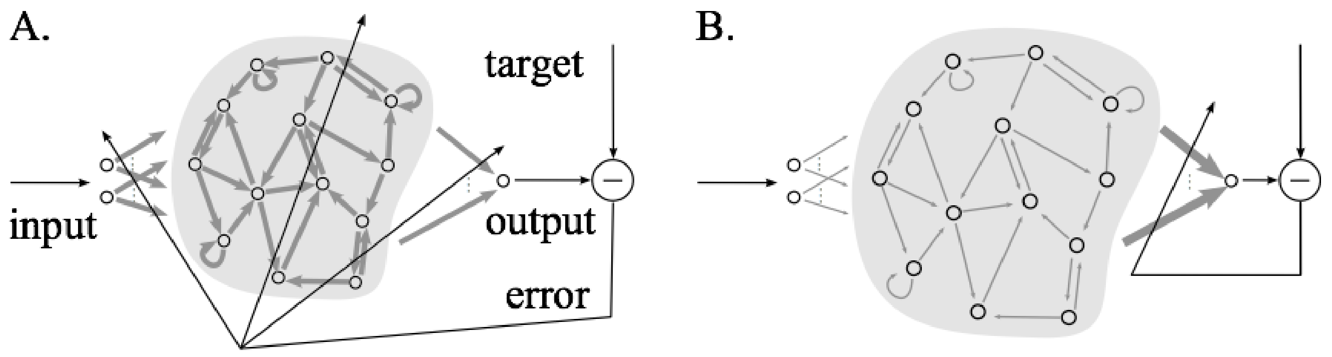

Figure 1: Training of a standard RNN displayed on the left hand side labelled A, compared to the training of a reservoir computing based network on the right hand side labelled B.

the topology of the reservoir used in an echo state network is an active area of research. We'd like our reservoir to be able to predict many features on multiple time scales within our data. To do this it is usually helpful to use a reservoir which is big, sparse, and randomly connected. However, in analyzing the results we will see that if the reservoir is "too" big over-fitting can occur.

1.2 Counting without a counter

It has been standardly assumed that for a system to simulate a context free language it must use something akin to a stack, such as in a pushdown automata, for it to keep track of state. This stack would predict relationships in temporal data by internally counting the appearance of signals. Parallel distributed processing systems have no predefined internal counting mechanism and would have to either create one during training or discover another method that they can use to keep track of state. In experiments it has been shown that neural networks opt for the first choice and will develop counters in order to predict state in a deterministic context free grammar (Rodriguez et al. [1999]).

We proceed in Section 2 with a description of previous work applying echo state networks to real world problems and specifically to next word prediction tasks. In Section 3 we present the implementation used for the echo state networks and the context free languages. In Section 4 we describe the experiments to be performed, then in Section 5 we present the results of these experiments. In Section 6 we discuss the results of the various experiments and finally we conclude in Section 7.

2. PREVIOUS WORK

Research into how to create and adjust the reservoir in an echo state network has investigated biologically inspired topologies, modular topologies using sub-reservoirs, and various other methods. There has also been research into global unsupervised methods which optimize reservoir structure based on the input signal but irrespective of the target signal. Other methods have researched supervised reservoir optimization, where reservoir structure is changed based on both the input signal and the target signal. The reservoirs used in our experiments are based on the classic echo state network approach in which the reservoir is randomly created and then scaled to have a specific spectral radius. This approach does not involve unsupervised or supervised optimization.

Echo state networks have been shown to outperform other methods, be they neural network methods or not, on a variety of prediction and classification tasks. Echo state networks have been shown to produce state of the art performance in predicting chaotic dynamics (Jaeger and Hass [2004]), the Japanese Vowel benchmark (Jaeger et al. [2007]), and spoken digit recognition (Verstraeten et al. [2006]). Relevant to the task at hand, recent work has show impressive performance of echo state networks on next word prediction. In Tong et al. [2007] echo state networks are able to not only predict the next word but are additionally able to make good predictions of verb agreement beyond the distance of bigrams and trigrams.

Frank and Čerňaský [2008] show that in sentence processing tasks echo state networks are able to outperform Markov models and, if inputs are represented in an alternate but still task independent manner, outperform standard simple recurrent networks, such as those used in the next word prediction task of Elman [1990]. The ability of an echo state network to outperform the simple recurrent networks used in Elman's original work is significant because it demonstrates that with more constrained dynamics superior results can be achieved. In additional work following up on the original Frank and Čerňaský [2008] paper, it was shown that the echo state network, with adjusted input weights, solves the next word prediction task by producing a simple finite state machine able to generalize well (Frank and Jacobsson [2010]).

The impressive performance of echo state networks on next word prediction tasks suggests that they might also perform well on other related tasks, such as the "counting without a counter" work done by Rodriguez et al. [1999]. In the original work Rodriguez et al. show that recurrent neural networks can be trained to count by predicting the next state in a deterministic context free language represented by sequences of strings of the form $a^n b^n$ for different n. In further work Rodriguez shows that simple recurrent networks can do even better and represent aspects of context sensitive languages by storing counting information (Rodriguez [2001]).

3. METHOD

3.1 Echo state network implementation

The echo state networks used are based on leaky integrator neurons as described in Lukoševičius and Jaeger [2009]. Given a neuron signal $\mathbf{x}$ and an input signal $\mathbf{u}$, the neuron signal at time step n is defined as:

$$\mathbf{x}(n) = (1-a)\mathbf{x}(n-1) + af(\mathbf{W}_{in}\mathbf{u}(n) + \mathbf{W}\mathbf{x}(n-1)) \quad (1)$$

where a is a leaking rate, which controls the speed of dynamics, $\mathbf{W}_{in}$ is the weight matrix for the input signals, and $\mathbf{W}$ is the weight matrix for the neuron signals – the reservoir. The output signal, $\mathbf{y}$, is generated by the network as:

$$\mathbf{y}(n) = \mathbf{W}_{out}\mathbf{x}(n) \quad (2)$$

and for a set of training examples can be viewed as a system of linear equations where:

$$\mathbf{W}_{out}\mathbf{X} = \mathbf{Y}_{target} \quad (3)$$

which is to be solved for $\mathbf{W}_{out}$. The implementation we use solves Equation 3 using linear regression by computing the pseudo inverse:

$$\mathbf{W}_{out} = \mathbf{Y}_{target}\mathbf{X}^{\mathsf{T}}(\mathbf{X}\mathbf{X}^{\mathsf{T}})^{+} \quad (4)$$

which we found to produce better results than ridge regression.

The reservoir was created with a spectral radius, $\rho(\mathbf{W})$, scaled to 0.9. The nonlinear function f used in Equation 1 is the hyperbolic tangent, the standard non-linear functions used with echo state networks. In each experiment the size of the reservoir was varied, as described below in the Experiments section.

3.2 Representation of context free languages

To evaluate the performance of echo state networks in the task of predicting context free languages we use the same inputs as those used in Rodriguez et al. [1999]. These inputs consist of strings of the form $a^n b^n$ with more strings for shorter values of the length n. The strings are distributed according to length n with 10 strings for $n = 1$, 6 strings for $n = 2$, 4 strings for $n = 3$, 3 strings for $n = 4$, and 1 string each for $n \geq 5$.

To test the network an additional set of strings was created consisting of one string of $a^n b^n$ each for $1 \leq n \leq 20$. Because echo state networks operate on real values, the strings were encoded with $a = 0$ and $b = 1$. The order of the strings was randomized before training and testing.

4. EXPERIMENTS

All experiments were performed using the Reservoir Computing Toolbox[1] for MATLAB. Four different experiments were performed with a different size reservoir used in each experiment. The reservoir size was evaluated containing 25, 50, 100, and 200 neurons.

If the network learns to count it should be able to predict the start of a sequence with a by, given the number of as that were presented previously, counting how many bs have been presented thus far and predicting an a after the number of bs seen is equal to the previous number of as seen. In terms of the coding used, this means that the network should be able to predict the transitions from 1 to 0, but should not be able to predict the transitions from 0 to 1, because given a string of as the next symbol could be either an a in the current string or a b signaling the end of the current string.

The network is presented with randomly ordered context free grammars $a^n b^n$ with n distributed as shown above in Section 3.2. These randomly ordered strings are concatenated to form a single long string with 204 0s and 1s. Again following the training procedure used in Rodriguez et al. [1999], the network is trained using the presentation of 13 examples of different 204 character strings. To evaluate the experiments performed we use the normalized mean square error (NMSE) of the training and testing sets.

[1]Reservoir Computing Toolbox:
`snn.elis.ugent.be/rctoolbox`

5. RESULTS

The results show a very clear trend indicating that networks with reservoirs that have a smaller number of neurons are able to generalize better and produce improved performance on the test set, while networks with reservoirs that have more neurons tend to have lower scores on the training set but do not generalize their results to the test set well. A reservoir of 25 neurons produces the best results on the test set, as indicated by the lowest test NMSE score, as shown in Table 1.

Experiment	Training NMSE	Test NMSE
25 Neurons	0.6784	**0.4877**
50 Neurons	0.5247	0.5562
100 Neurons	0.4008	0.7173
200 Neurons	**0.1427**	0.8968

Table 1: Results from simulating deterministic context free grammars using echo state networks with reservoirs of different size.

Given that half of the transitions should be unpredictable, the 25 neuron reservoir's test NMSE score of 0.4877 is impressive and shows that the network appears to have learned to count. The 50 neuron reservoir's test NMSE score of 0.5562 is similarly impressive and indicates that the network has also learned to count. The 100 neuron reservoir network has a higher test NMSE score and it is unclear from these results alone that it has learned to count. Below we will argue that, based on signals found by its output neurons, it has learned to find a significant amount of structure in the input signal and can count. The 200 neuron reservoir network has a significantly higher test NMSE score and does not appear to have learned to count.

5.1 The 25 neuron reservoir

Figure 2 shows the output of the 25 neuron reservoir network compared against what is expected for the test results. Recall that it is the transitions from 1 to 0 (that is b to a) which should be predictable by the network. If we only consider these transitions, that is look at only the beginning of the red lines at the y-axis value of 0 towards the bottom of the chart – in what resembles a saw tooth, it is clear that the network does a good job of predicting this part of the signal. Perhaps somewhat surprising, given the theoretical underpinnings, is that the simulated value, represented by the blue line, very often matches the signal, the red line, at the upper left of each saw tooth, which is the transition from 0 to 1 and should be unpredictable.

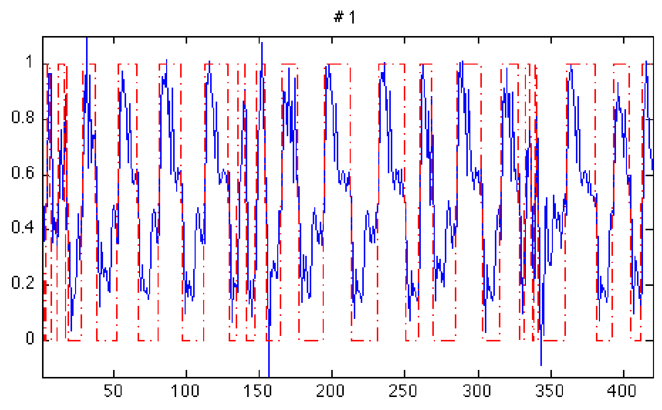

Figure 2: Expected output and simulated output for a reservoir of size 25. The expected output is indicated in red, and is either 0 or 1, while the simulated output is in blue.

It is not particularly clear how the network is solving the task. We see that often from the peak of the predicted value of the signal, at y-axis value 1, the signal slowly decreases and then sharply declines to match the target signal, as if it were counting down until the appearance of another 0. A "counting down" solution of this form would match the functioning of the simple recurrent networks developed in the original work of Rodriguez et al. [1999].

5.2 The 50 neuron reservoir

To determine whether this prediction pattern holds up across multiple networks we present the output of the 50 neuron reservoir network compared against what is expected for the test results in Figure 3. As can be seen, the network's solution to this problem shows precisely the same pattern as with the 25 neuron reservoir. However, with the 50 neuron reservoir solution it appears that the transitions from 0 to 1 may be more accurately predicted than the transitions from 1 to 0. Many of the transitions show the same counting down pattern, where the generated signal slowly decreases

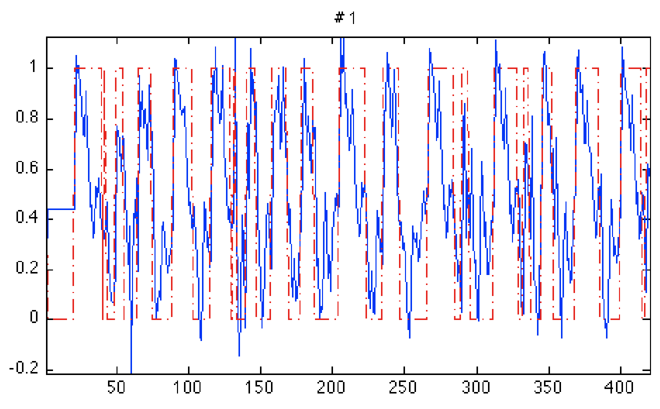

Figure 3: Expected output and simulated output for a reservoir of size 50. The expected output is indicated in red, and is either 0 or 1, while the simulated output is in blue.

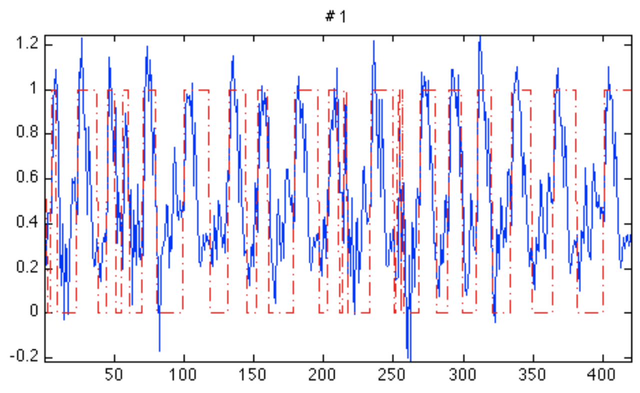

Figure 4: Expected output and simulated output for a reservoir of size 100. The expected output is indicated in red, and is either 0 or 1, while the simulated output is in blue.

from the 1 value and then sharply declines when the target signal reaches 0. In Section 6 below we'll further investigate this behavior.

5.3 The 100 and 200 neuron reservoirs

The test results for the 100 neuron reservoir network are presented in Figure 4. The network with a 100 neuron reservoir shows the same pattern as the other networks but there is significant degradation in its performance as it often overestimates the value 1. We see something similar to this in the solution found using a 50 neuron reservoir. In Figure 3, around the y-value 1 at position 200 on the x-axis, the estimated value of the network is greater than 1, this problem is exacerbated as the size of the reservoir increases. In the 200 neuron reservoir network overestimation is extremely problematic and largely responsible for the poor NMSE on the test data set.

6. DISCUSSION

After examining the results above the question remains: how do the echo state networks solve the task? To attempt to answer this question we'll examine graphs overlaying the predictions of the multiple output neurons used by the echo state network. In Figure 5, Figure 6, and Figure 7 the signal from each output neuron is presented in a different color.

6.1 Examining output neuron signals

Figure 5 presents the outputs of the neurons used by the 25 neuron reservoir network. In it we see, towards the top of the graph, a distinct output signal represented by the blue line, and towards the bottom there is another distinct output signal represented by the magenta line. Comparing Figure 5 with the solution graph of the 25 neuron network presented in Figure 2, note that the top signal, the blue line, is very close to the output solution, it tracks the top of the saw teeth. Referring back to Figure 5 we also see that there are many neurons below 0 tracking the value of the magenta line, more than there are above 0. The positive output signal, shown in blue, balances the large number of negative output signals to produce the minimum of 0 when all signals head toward zero and the maximum of 1 when signals diverge from 0.

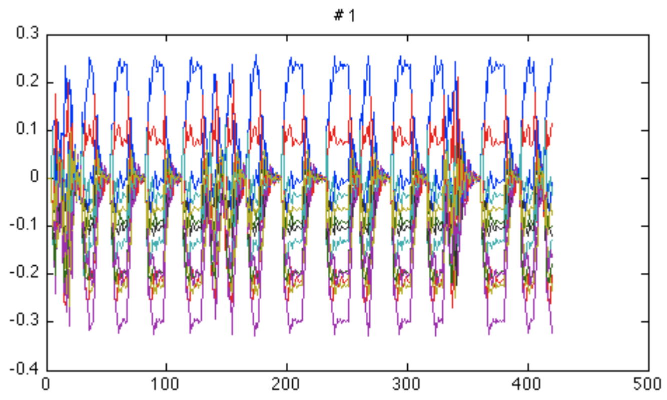

Figure 5: Overlay graph of the output neurons from the network with a reservoir size of 25 neurons.

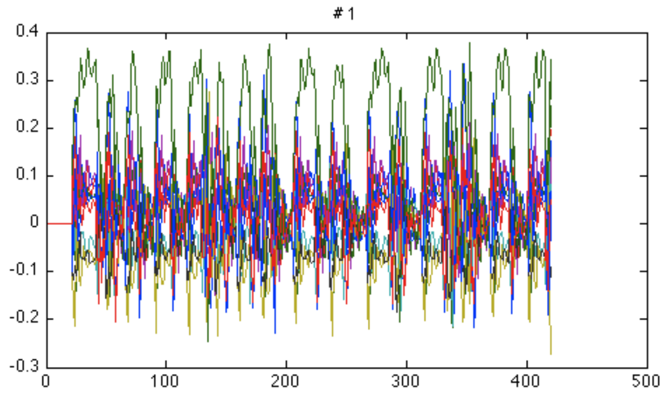

Figure 6: Overlay graph of the output neurons from the network with a reservoir size of 50 neurons.

In comparing the output neuron graph for the 50 neuron reservoir network in Figure 6 to the performance graph of the 50 neuron reservoir in Figure 3 we see a similar outlier signal, shown in green, that aligns with the saw tooth pattern seen in the network's solution. Even more than in the output graph for the 25 neuron reservoir network (Figure 5), the 50 neuron reservoir output relies very heavily on this single green signal to solve the problem. Additionally, it is interesting that the blue output signal in Figure 6 only appears at the shift from 0 to 1, seen at the start of each increase in the green signal. This blue signal appears to accurately predict the switch from 0 to 1.

Finally, in Figure 7, we see that the 100 neuron reservoir network shows a similar pattern as that seen in the 25 and 50 neuron reservoirs. Note that in the 100 neuron reservoir the output signals are significantly more well defined, that is they seem to be following specific features in the input signal, in contrast with the output signals for the 50 neuron reservoir in Figure 6, which are much more chaotic. Most importantly, the output neurons in Figure 7 have well defined shifts to zero, showing that all the neurons agree on when the value should be zero, although they often disagree as to what the value should be when it is not zero.

A further interesting pattern in Figure 7 is the resemblance of the output neurons to a falling-off sound wave as they decrease from their maximums to 0. It appears that

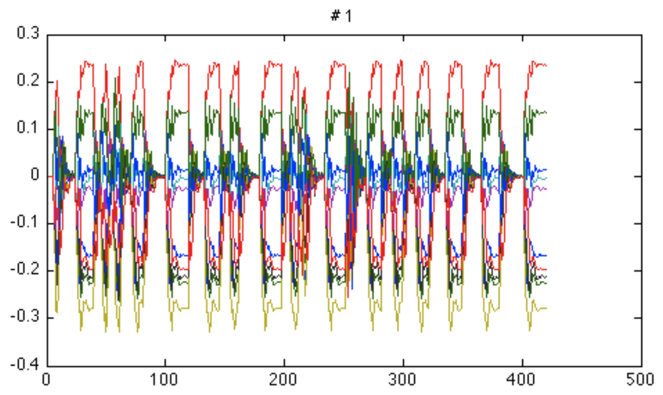

Figure 7: Overlay graph of the output neurons from the network with a reservoir size of 100 neurons.

this network, and the 25 neuron reservoir, which shows the same pattern, are predicting the 0 value by slowly decreasing all of their output signals towards 0. It also appears that the networks may be able to estimate the unpredictable transition from 0 to 1 by converging from 0 upon reaching it. This is clearly displayed just before the x-axis 400 value in Figure 7 where the signals all approach 0 and then immediately spike to different non-0 values after having reached it.

6.2 Predicting the transition from 1 to 0

In Figure 5, Figure 6, and Figure 7 we see that the output neurons agree significantly more on when the signal should be 0 than when the signal should be 1. In the 25 neuron reservoir graph and the 100 neuron reservoir graph, shown in Figures 5 and 7 respectively, there are very clearly defined points at which all signals converge and decrease to 0. In these figures we also see that for the initial 0 value in the target signal the various output neuron signals are highly divergent and form no consensus prediction, as we would expect since there is no data from which to predict this value. In contrast, for the second 0 in the target signal all the neuron predictions clearly converge to the 0 value, which is now predictable.

This is indicative of the actual "counting" which is occurring since it is the value of 0 – in the transition from 1 to 0 – which should be predictable by the network and indeed is predictable by the network as all output signals confirm this value. When the predicted value differs from 0 there is high disagreement between output neurons, suggesting the inability to find a clear signal in the data. The agreement on 0 values is the strongest evidence seen that these networks, including the 100 neuron reservoir network, have learned to count.

7. CONCLUSION

We have shown that echo state networks are able to produce results simulating a deterministic context free grammar and perform the same function of "counting without a counter" as produced with simple recurrent neural networks in Rodriguez et al. [1999]. Reservoirs with fewer neurons are able to produce results which generalize to new data better than reservoirs with more neurons, indicating that using a reservoir with too many neurons can lead to over-fitting.

However, networks with a larger reservoir are more capable of clearly identifying features within the input signal.

Based on superficial analysis the solutions produced by the echo state networks appear to predict the unpredictable transitions from 0 to 1 as well as the transitions from 1 to 0, the transitions which should be predictable. After deeper analysis, focusing on the signals produced by the various output neurons, it is clear that the network shows much more agreement with respect to the predictable transitions from 1 to 0 than the transitions from 0 to 1. This is what we would expect from a network that is storing state with regard to how many signals of one type have so far appeared.

Most significantly, the echo state network is able to solve the problem after training the reservoir for 13 examples of 204 character strings presented once each. In the original work the network was presented with about 249,000 strings and trained for approximately 2 million sweeps (Rodriguez et al. [1999]). The ability of the echo state network to produce competitive performance after significantly less training is quite impressive.

Further work includes evaluating different distributions over the strings in the test set and performing more in depth analysis of the networks used to solve the task. It would also be interesting to discover whether or not the network would be able to generalize with increased limitations in the training set. Work could also be done using variations in network connectivity or spectral radius.

References

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

S.L. Frank and M. Čerňaský. Generalization and systematicity in sentence processing by an echo state network. In K. Love, B.C. McRae and V.M. Sloutsky, editors, *Proceedings of the 30th Annual Conference of the Cognitive Science Society*, pages 733–738. Cognitive Science Society, 2008.

Stefan Frank and Henrik Jacobsson. Sentence-processing in echo state networks: a qualitative analysis by finite state machine extraction. *Connection Science*, 22(2):135–155, 2010.

Stefan Haeusler and Wolfgang Maass. A statistical analysis of information-processing properties of lamina-specific cortical microcircuit models. *Cerebral Cortex*, 17(1):149–162, 2007.

Herbert Jaeger. The “echo state network“ approach to analysing and training recurrent neural networks. Technical report, German National Research Center for Information Technology, 2001.

Herbert Jaeger and Harold Hass. Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless communication. *Science*, pages 78–80, 2004.

Herbert Jaeger, Mantas Lukoševičius, Dan Popovici, and Udo Siewert. Optimizing and applications of echo state networks with leaky-integrator neurons. *Neural Networks*, 20(3):335–352, 2007.

Mantas Lukoševičius and Herbert Jaeger. Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, 3(3):127–149, 2009.

Wolfgang Maass, Thomas Natschläger, and Henry Markram. Real-time computing without stable states: a new framework for neural computation based on perturbations. *Neural Computation*, 14(11):2531–2560, 2002.

Wolfgang Maass, Prashant Joshi, and Eduardo D. Sontag. Principles of real-time computing with feedback applied to cortical microcircuit models. *Advances in Neural Information Processing Systems 18 (NIPS 2005)*, pages 835–842, 2006.

Paul Rodriguez. A recurrent neural network that learns to count. *Neural Computation*, 13:2093–2118, 2001.

Paul Rodriguez, Janet Wiles, and Jeffrey Elman. A recurrent neural network that learns to count. *Connection Science*, 11(1):5–40, 1999.

Matthew H. Tong, Adam D. Bickett, Eric M. Christiansen, and Garrison W. Cottrell. Optimizing and applications of echo state networks with leaky-integrator neurons. *Neural Networks*, 20:424–432, 2007.

David Verstraeten, Benjamin Schrauwen, and Dirk Stroobandt. Reservoir-based techniques for speech recognition. In *Proceedings of the IEEE International Join Conference on Neural Networks (IJCNN 2006)*, pages 1050–1053, 2006.