Echo State Networks: Appeal and Challenges

Danil Prokhorov
Ford Research and Advanced Engineering
Dearborn, MI 48124

***Abstract*— The echo state network (ESN) has recently been proposed for modeling complex dynamic systems. The ESN is a sparsely connected recurrent neural network with most of its weights fixed a priori to randomly chosen values. The only trainable weights are those on links connected to the outputs. The ESN can demonstrate remarkable performance after seemingly effortless training. This brief paper discusses ESN in a broader context of applications of recurrent neural networks (RNN) and highlights challenges on the road to practical applications.**

I. Introduction

The ESN is a special case of fully connected one-hidden-layer RNN in which only a small fraction of recurrent connections in the hidden layer is active (this imitates biologically plausible sparsity), and only weights $\mathbf{W}$ of the connections from hidden nodes to the outputs are trainable:

$$\begin{aligned} \mathbf{z}(t+1) &= \mathbf{f}(\mathbf{W}_{st}\mathbf{z}(t) + \mathbf{W}_{in}\mathbf{x}(t)) \\ \mathbf{y}(t+1) &= \mathbf{W}\mathbf{z}(t+1) \end{aligned} \quad (1)$$

where $\mathbf{f}$ consists of bipolar sigmoids or $\tanh$, and $\mathbf{W}_{in}$ are weights from the network inputs $\mathbf{x}(t)$. The hidden layer $\mathbf{z}$ with its connections is termed the reservoir in [1]. It is proposed to choose all reservoir weights $\mathbf{W}_{st}$ at random, following a simple prescription in [1]. In addition, randomly chosen are the weights $\mathbf{W}_{in}$ which may serve not only the input connections but also the feedback connections from the outputs $\mathbf{y}$ and the bias connections. ESN augmented with squares of the reservoir nodes as well as direct (sometimes squared) connections from the inputs to the outputs are also proposed (see [2]) to enhance the ESN approximation basis by additional trainable connections.

The main ESN appeal is simplicity of its training. Indeed, its output weights can be trained by any suitable method (recursive least squares, etc.) in one-shot fashion, i.e., without repetitive passes through the training set which are usually required for training other RNN architectures. Backpropagation through time is not required either.

One could call RNN like ESN sparsely connected partially trained RNN, but we adopt the ESN as a generic name for all such networks. In retrospect, one can probably recognize these networks in published literature. For example, [4] discussed what amounts to a network of coupled oscillators with most of the weights fixed a priori.

II. Discussion

Among several ESN demonstrations in [1],[2],[3] long-term (iterated) predictions of several chaotic time series are especially interesting because they seem to be the best in terms of accuracy as compared with competitive techniques.

With our own implementation (following Jaeger's work closely, though using C language) we can basically confirm remarkable accuracy of long-term iterative predictions on the Mackey-Glass series of $\tau = 17$, as well as the one-step predictions on a set of non-trivial NARMA models as discussed in [2]. For the Mackey-Glass problem, we observe that **only a small fraction** of all solutions (which differ by randomly initialized weights $\mathbf{W}_{st}$ and $\mathbf{W}_{in}$) achieved the accuracy quoted (or better than) in [1] and [3]. Figure 1 illustrates our histogram for the maximum number of steps of the iterated predictions till the divergence between the target and the ESN prediction becomes visible (this occurs when the running squared error exceeds the threshold of 0.01). The majority of our solutions are capable of about 600 to 700 steps of iterated predictions till visible divergence. One of the solutions (0.1%) is excellent to about 2100 steps.

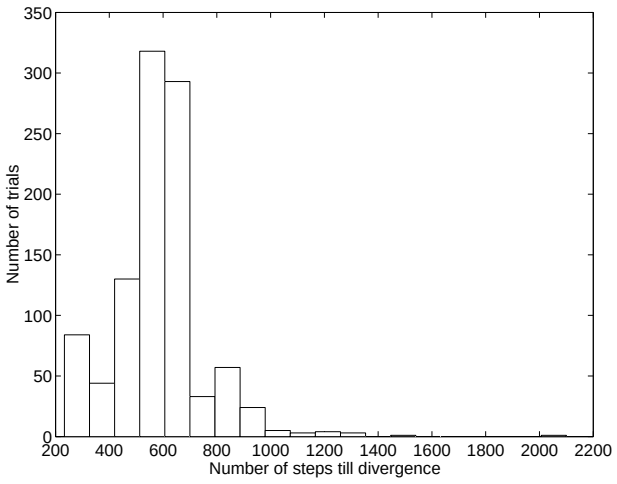

Fig. 1. The histogram of the Mackey-Glass series ($\tau = 17$) results for 1000 trials. It shows a distribution of the maximum number of the iterated prediction steps till the visible divergence between the target and the ESN prediction.

Figure 2 shows little correlation between the maximum number of the iterated prediction steps till the visible divergence and the training RMS error. Likewise, the length of iterated predictions till divergence and the absolute value of the maximum eigenvalue of the reservoir weight matrix $\mathbf{W}_{st}$ are not correlated (Figure 3). (Controlling the maximum eigenvalue is proposed in [1] as a suitable ESN design parameter.)

The Mackey-Glass problem of $\tau = 30$ (sampling of 1

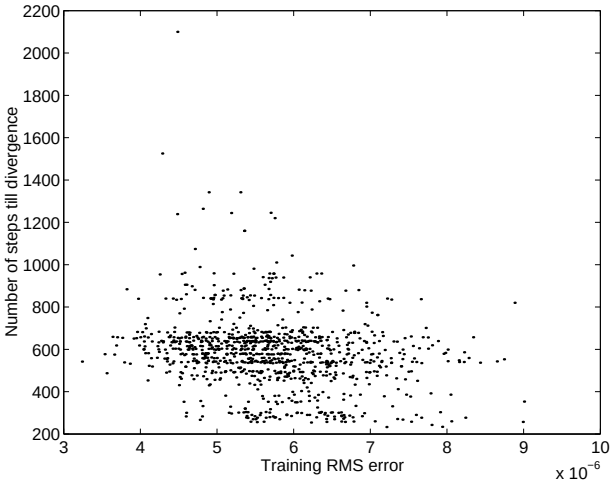

Fig. 2. The plot of the maximum number of the iterated prediction steps till the visible divergence vs. the training RMS error corresponding to the histogram of Figure 1. The statistically significant correlation coefficient is -0.23.

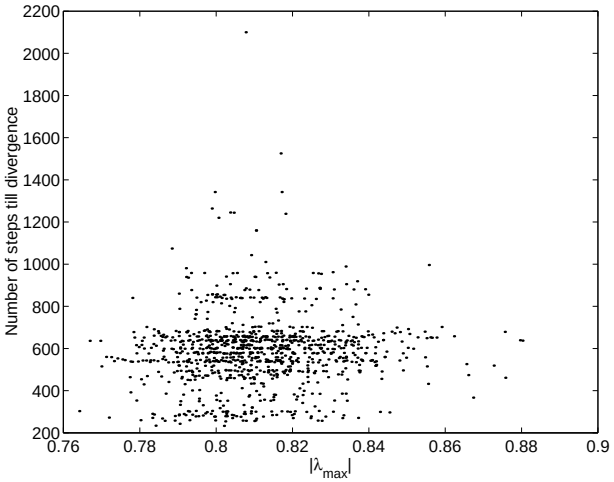

Fig. 3. The plot of the maximum number of the iterated prediction steps till the visible divergence vs. the absolute value of the maximum eigenvalue of the reservoir weight matrix for the same experiment as illustrated in the two previous figures. The statistically significant correlation coefficient is 0.07.

second) is known to be substantially harder than that of $\tau = 17$. The fine tracking to nearly 500 time steps quoted in [1, page 30] seems remarkable, but we have not been able to confirm this result using our implementation yet. Our best result is good to about 200 time steps.

When the Mackey-Glass series of $\tau = 30$ is subsampled every 6 seconds, the best results in [3] are about 120 steps of iterated predictions. Figure 4 shows our histogram for the maximum number of steps of the iterated predictions till the divergence between the target and the ESN prediction becomes visible. The majority of the solutions are capable of from 40 to 80 steps of iterated predictions till visible divergence. As in the case of the time series with $\tau = 17$, there is little correlation between the training RMS error and the test RMS error (or between other quantities of relevance).

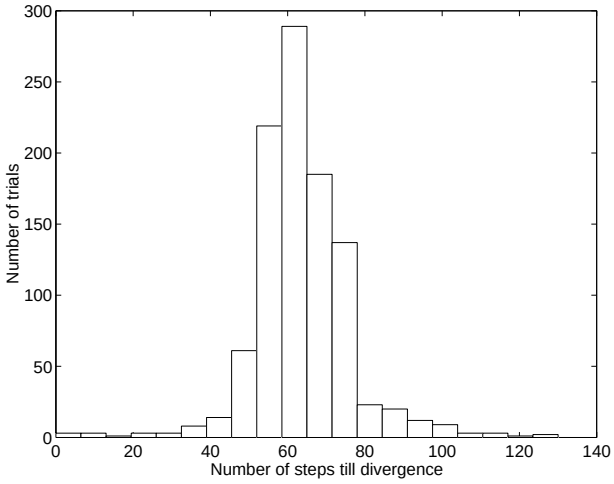

Fig. 4. The histogram of the Mackey-Glass series ($\tau = 30$) results for 1000 trials. It shows a distribution of the maximum number of the iterated prediction steps till the visible divergence between the target and the ESN prediction.

We believe that, at least, a part of the blame for our mixed success in the Mackey-Glass experiments may be ascribed to difficulties with creating a rich enough reservoir. It is our observation that an effective recipe for creating a reservoir adequate to a problem at hand may be not as simple as the one recommended in [1]. We also seem to be able to employ successfully functions other than $\tanh$ as $\mathbf{f}$ in (1) (e.g., $\sin$ and $\cos$), reservoirs with the absolute value of the maximum eigenvalue outside the unit circle (watch out for chaotic oscillations in the reservoir!), or delays chosen randomly from $\{1, 2, ..., d_{max}\}$ on connections between the nodes of the reservoir.

Of special interest for us are problems which can be handled well by more traditional RNN such as recurrent multilayer perceptrons (RMLP) etc., especially those for which good solutions have already been obtained by application of the backpropagation through time and extended Kalman filter (BPTT-EKF) methodology [5]. One problem of serious practical importance is the engine misfire detection problem introduced in [6] (its simplified version and data are available at http://www.geocities.com/challenge.html). It is representative of a class of problems featuring very heterogeneous and long time series (~ 100 K points or longer). All our attempts to solve the simplified version of the misfire detection problem with ESN failed. We think that the primary reason for our failure is the inability to obtain the adequate reservoir. It appears that, at least for the basic ESN design, training only output weights $\mathbf{W}$ just is not enough to compete with our previous (non-ESN) solutions. These solutions can be obtained readily, and they are much more compact than potential ESN solutions.

This brings us to the bigger question of computational complexity. Suppose that the ESN based solution features the same number of output weights as our current misfire detection RNN (about 500 weights). It is important to keep in mind that

this is an on-board (in-vehicle) application, and that the RNN is to be executed in real time (on the order of milliseconds). The RNN compactness does matter, and the current (and even projected) capabilities of in-vehicle computers are not adequate to run the ESN because this would have to employ a much larger number of total connections than the RNN.

A set of problems to which application of ESN apparently runs into serious obstacles is known as adaptive behavior with fixed weights [7]. RNN with weights fixed after training by methods based on the BPTT-EKF turn out to acquire properties that may only be attributed to conventional adaptive systems. Our attempts to train ESN for either the problem of learning all quadratic functions discussed in [7] or the problem of conditioned adaptive behavior [8] have achieved levels of performance noncompetitive with those obtained by our standard method. Yet, these problems admit solutions by relatively modest RNN. As for the quadratic function problem, our latest results indicate that 3-25R-1L RMLP (751 weights) is sufficient to obtain approximately the same level of performance as that of our previous solution. The previous best result was obtained by the significantly larger 3-30R-10R-1L RMLP (1440 trainable weights), as discussed in [7]. The 3-25R-1L stands for an RMLP with three inputs, one fully connected recurrent hidden layer of 25 nodes and one linear output. In fact, this network has the same architecture as (1) except that its "reservoir" is fully connected and trained. Presumably, an ESN with 751-weight $\mathbf{W}$ should be able to match the RMLP performance on this task, as implied by the results of comparative experiments in [2], but in our experiments even much larger 1000- or 2000-node ESN could not come close to the RMLP performance.

Interestingly, the RNN training by methods based on the BPTT-EKF is seldom very sensitive to initial (random) weights of the network. It is true that the BPTT-EKF implementation cost is higher than that of the ESN, and it can indeed take a few training sessions before the optimal values of the training parameters are found. However, we are confident that even ESN training can benefit from training the *entire* network, e.g., within the BPTT-EKF framework, although in such a case the preference could be given to training methods with $O(N)$ computational complexity (N is the number of trainable parameters).

Though the ESN (with a lucky reservoir!) for iterated predictions of chaotic time series exhibits impressive performance, we should emphasize that these demonstrations are made with *noiseless* time series (neither measurement nor process noise). Real time series are never noiseless. The presence of noise also limits fundamentally our ability to predict sufficiently far into the future. In the real world high-accuracy iterated predictions good to several hundred of time steps, as in the case of the Mackey-Glass series, are simply not feasible. Furthermore, time series data may be limited to a fixed and rather small number of points (e.g., see the challenging time series competition problem in [9]), precluding the use of humongous ESN and forcing to be creative in the network design. Interestingly, when the ESN iterated predictions are attempted for the well known 1000-point laser time series, the ESN performance advantage becomes more modest than the performance of other competitive techniques [3].

A. Applications of ESN to control

We are aware of only two papers discussing applications of ESN to control [10], [11]. In both of these papers, the ESN controller is trained on target control values u_{tgt} explicitly provided by a teacher, thereby reducing the control problem setting to that of the standard modeling. For example, when the plant function $x(t+1) = F(u(t), x(t))$ and the state x are available, $F(\cdot)$ can possibly be inverted to yield the target controls: $u_{tgt} = I(x_{tgt}, x(t))$, where $x(t+1) = x_{tgt}$ (the target state value). The usual ESN training can then be initiated, as detailed in [1, page 14]. Though in some applications inference of u_{tgt} can be done (perhaps, with the help of model predictive control methods), in many others such inference is either infeasible or unreliable.

The generality of model based (indirect) control with ESN is lower than that of the method for neurocontrol described in [12]. Indeed, the only requirement for application of the method [12] is the ability to compute sensitivities of plant model outputs to changes of its controls. This can efficiently be done by carrying out BPTT through *both* the plant model and the controller. Clearly, computational expenses would be very significant if we were to carry out BPTT through an ESN based controller of even modest size. This additional complexity may all but nullify the main appealing advantage of the ESN. To avoid this burden, we must either ignore the backpropagation pathways through the ESN or resort to the well known but less general neurocontrol methods of [13]. (In the latter case, the ESN could first model the plant represented in a simplified form called affine in controls, then the controller is synthesized as a simple function of the ESN outputs.) In either case, the neurocontrol quality might not be the best, especially if various disturbances and uncertainties are present.

We sketch an alternative scheme of indirect control with ESN. We assume the plant model affine in controls $\mathbf{u}$:

$$\begin{aligned} \mathbf{x}(t+1) &= \mathbf{F}(\mathbf{x}(t), \mathbf{u}(t)) \\ &= \mathbf{H}(\mathbf{x}(t)) + \mathbf{G}(\mathbf{x}(t))\mathbf{u}(t) \end{aligned} \tag{2}$$

where $\mathbf{H}$ is a vector function, $\mathbf{G}$ is a control matrix. If the instantaneous performance measure $U(t) = \mathbf{x}^T(t)\mathbf{Q}\mathbf{x}(t) + \mathbf{u}^T(t)\mathbf{R}\mathbf{u}(t)$, $\mathbf{Q} > 0$, $\mathbf{R} > 0$, then implementable parameterization of the optimal controller for our infinite horizon control problem may be expressed using the ESN with output vector $\boldsymbol{\lambda}$ as the following product

$$\mathbf{u}(t) = -\mathbf{R}^{-1}\mathbf{G}^T(\mathbf{x}(t))\boldsymbol{\lambda}(\mathbf{x}(t), \mathbf{W}) \tag{3}$$

Such ESN can thus be used within the derivative adaptive critic framework [14]. We first run the closed-loop system forward with the sequence of controls $\{\mathbf{u}\}_{t=t_0}^{t=t_k}$ provided by the ESN with constant weights $\mathbf{W}$ from (3). To train ESN, we then employ the dual equations

$$\boldsymbol{\lambda}_{tgt}(t) = \mathbf{Q}\mathbf{x}(t) + (\partial\mathbf{F}(t))/\partial\mathbf{x}(t))^T\boldsymbol{\lambda}_{tgt}(\mathbf{x}(t+1)) \tag{4}$$

by going backwards from $t = t_k$ to $t = t_0$ (set $\boldsymbol{\lambda}_{tgt}(\mathbf{x}(t+1)) = 0$ for $t = t_k$). The ESN training errors are

$$\boldsymbol{\lambda}_{tgt}(t) - \boldsymbol{\lambda}(\mathbf{x}(t), \mathbf{W}) \tag{5}$$

which are minimized in the usual way. This procedure (going forward, going backward, then updating $\mathbf{W}$ with errors (5)) is repeated till convergence of $\mathbf{W}$. Preliminary experiments indicate that the ESN based derivative adaptive critic is viable, especially for control of distributed parameter systems such as in [15] (the same ESN can be used to control as many nodes of the distributed system as needed by connecting all controller outputs to the same reservoir), but it remains to be seen how competitive it is in comparison with other approaches for control with RNN. It is still an interesting possibility that the same ESN with a sufficiently rich reservoir could be used to control (or model) many dynamic systems by employing its different outputs and an algorithm for switching between different sets of $\mathbf{W}$.

In principle, control without a plant model (direct control) is realizable with ESN, just as with any suitable and less complex controller parameterization [16]. For some applications one can train an inverse ESN controller using the well known approaches of [17] and [18]. Our experiments in direct adaptive ESN control suggest that simultaneous perturbation stochastic approximation (SPSA) [19] can successfully be used to adapt ESN controllers, but this (and other derivative-free approaches and Q-learning) can also be used to adapt parameters of *any* controller. The main issue here is the added significant complexity vs. realized gains in performance. Furthermore, direct adaptive approaches are often much slower than model based control methods. It is worth mentioning that industries often invest significant resources to obtain high-fidelity models of plants and create effective controllers using such models and other problem specific knowledge, which explains why there exists a significant bias in favor of model based control. Imposing uncertainties on parameters of plant models does not prevent us from training robust RNN controllers on instances of plant models, as detailed in [12].

III. Conclusion

With this brief paper, we hope to encourage a constructive dialogue about the merits of ESN from a broad prospective of applications. The ESN is especially remarkable when applied to iterated predictions of noiseless chaotic time series (with due reservations about the academic nature of such exercises). The ESN is also appealing to applications in which quick adaptation is crucial. Yet, on-board or embedded applications (which put the premium on computational complexity and require high utilization of all available resources) are likely to remain limited to networks other than ESN.

Increasingly powerful computers create opportunities for applications of large RNN such as ESN. In our opinion, even with modern computers the basic ESN design recipe is not sufficient to create a successful ESN for a variety of important applications, as discussed in this paper. Clearly, improved ESN design/training procedures are needed to increase the chances of getting a successful ESN in a reasonable number of trials.

Acknowledgment

The author would like to thank Drs. Lee Feldkamp and Herbert Jaeger for useful discussions.

References

[1] H. Jaeger, The "echo state" approach to analysing and training recurrent neural networks. GMD Report 148, German National Research Center for Information Technology, 2001.

[2] H. Jaeger, "Adaptive nonlinear system identification with echo state networks," *NIPS 2002*.

[3] H. Jaeger and H. Haas, "Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless telecommunications," *Science*, April 2, 2004, pp. 78–80.

[4] P. Baldi and K. Hornik, "Universal approximation and learning of trajectories using oscillators," *NIPS 1995*.

[5] L. A. Feldkamp, D. V. Prokhorov, C. F. Eagen, and F. Yuan, "Enhanced multi-stream Kalman filter training for recurrent networks," in [9], pp. 29–53.

[6] Marko, K., James, J., Feldkamp, T., Puskorius, G., Feldkamp, L., and D. Prokhorov. "Training recurrent neural networks for classification: realization of automotive engine diagnostics," *Proceedings of the World Congress on Neural Networks (WCNN)*, San Diego, CA, September 1996, pp. 845–850.

[7] D. Prokhorov, L. Feldkamp, and I. Tyukin, "Adaptive behavior with fixed weights in RNN: Overview." *Proceedings of International Joint Conference on Neural Networks '02*, Hawaii, 2002.

[8] L. Feldkamp, D. Prokhorov, and T. Feldkamp, "Simple and conditioned adaptive behavior from Kalman filter trained recurrent neural network," *Neural Networks*, vol. 16, pp. 683–689, 2003.

[9] J. Suykens and J. Vandewalle (eds), *Nonlinear Modeling: Advanced Black-Box Techniques*, Kluwer Academic Publishers, 1998.

[10] J. Hertzberg, H. Jaeger, and F. Schönherr, "Learning to ground fact symbols in behavior-based robots," *Proceedings of the 15th European Conference on Artificial Intelligence,* Lyon, France, July 2002, pp. 708–712.

[11] P. Joshi and W. Maas, "Movement generation with circuits of spiking neurons," *Neural Computation*, 2004.

[12] D. V. Prokhorov, G. V. Puskorius, and L. A. Feldkamp, "Dynamical neural networks for control," see in *A Field Guide to Dynamical Recurrent Networks*, J. Kolen and S. Kremer (Eds.), IEEE Press, 2001, pp. 257–289.

[13] K. S. Narendra and S. Mukhopadhyay, "Adaptive control of nonlinear multivariable systems using neural networks," *Neural Networks*, vol. 7, no. 5, 1994, pp. 737–752.

[14] D. Prokhorov, "Backpropagation through time and derivative adaptive critics: a common framework for comparison," see Chapter 15 in *Learning and Approximate Dynamic Programming*, J. Si et al. (eds), IEEE Press, 2004.

[15] D. V. Prokhorov, "Optimal Neurocontrollers for discretized distributed parameter systems," in *Proceedings of the American Control Conference*, Denver, CO, 2003, pp. 549–554.

[16] R. M. Sanner and J. E. Slotine, "Gaussian networks for direct adaptive control," *IEEE Trans. Neural Networks*, vol. 3, 1992, pp. 837–863.

[17] M. Kawato, Y. Uno, M. Isobe, and R. Suzuki, "Hierarchical neural network model for voluntary movement with application to robotics," *IEEE Control Systems Magazine*, vol. 8, no. 2, April 1988, pp. 8–15.

[18] B. Widrow and E. Walach. *Adaptive Inverse Control.* Prentice-Hall, 1994.

[19] J. C. Spall and J. A. Cristion, "Model-free control of nonlinear stochastic systems with discrete-time measurements," *IEEE Trans. Automatic Control*, vol. 43, no. 9, September 1998, pp. 1198–1210.