

An overview of reservoir computing: theory, applications and implementations

Benjamin Schrauwen David Verstraeten Jan Van Campenhout
Electronics and Information Systems Department, Ghent University, Belgium
`Benjamin.Schrauwen@UGent.be`

Abstract. Training recurrent neural networks is hard. Recently it has however been discovered that it is possible to just construct a random recurrent topology, and only train a single linear *readout* layer. State-of-the-art performance can easily be achieved with this setup, called Reservoir Computing. The idea can even be broadened by stating that any high dimensional, driven dynamic system, operated in the correct dynamic regime can be used as a temporal ‘kernel’ which makes it possible to solve complex tasks using just linear post-processing techniques.

This tutorial will give an overview of current research on theory, application and implementations of Reservoir Computing.

1 Introduction

In machine learning, feed-forward structures, such as artificial neural networks, graphical Bayesian models and kernel methods, have been extensively studied for the processing of non-temporal problems. These architectures are well understood due to their feed-forward and non-dynamic nature. Many applications are however temporal, such as prediction (weather, dynamic systems, financial data), system control or identification, adaptive filtering, noise reduction, robotics (gait generation, planning), vision and speech (recognition, processing, production). In short, many real-world tasks.

It is possible to solve temporal problems using feed-forward structures. In the area of dynamical systems modeling, Takens [55] proposed that the (hidden) state of the dynamical system can be reconstructed using an adequate delayed embedding. This explicit embedding converts the temporal problem into a spatial one. The disadvantages of this approach is the artificially introduced time horizon, many parameters are needed when a long delay is introduced and it is not a ‘natural’ way to represent time.

A possible solution to this problem is adding recurrent connections to the feed-forward architectures mentioned above. These recurrent connections transform the system into a potentially very complex dynamical system. In the ground-breaking work of Hopfield [16], the dynamics of a Recurrent Neural Network (RNN) were controlled by constructing very specific topologies with symmetrical weights. Even though this system is - in general - chaotic, the Hopfield architecture depends critically on the presence of point attractors. It is also used in an autonomous, offline way: an initial state is imposed, and then the dynamic system is left running until it ends up in one of its many attractors.

Another line of research, initiated by the back-propagation-through-time learning rule for recurrent networks by Werbos [61, 62] (which was later re-

discovered in [42]), is to train all the weights in a fully (or sparsely) connected recurrent neural network. The literature describes many possible applications, including the learning of context free and context sensitive languages [41, 11], control and modeling of complex dynamical systems [54] and speech recognition [40, 12]. RNNs have been shown to be Turing equivalent [25] for common activation functions, can approximate arbitrary finite state automata [33] and are universal approximators [10]. So, theoretically, RNNs are very powerful tools for solving complex temporal machine learning tasks. Nonetheless, the application of these rules to real-world problems is not always feasible (due to the high computational training costs and slow convergence) [14] and while it is possible to achieve state-of-the-art performance, this is only reserved for experts in the field [38]. Another significant problem is the so-called *fading gradient*, where the error gradient gets distorted by taking many time steps at once into account, so that only short examples are usable for training. One possible solution to this is a specially constructed Long Short Term Memory (LSTM) architecture [44], which nonetheless does not always outperform time delayed neural networks.

In early work of Buonomano [3], where a random network of spiking neurons with short-term plasticity (also known as dynamic synapses) was used, it was shown that this plasticity introduces much slower dynamics than can be supported by the recurrent topology. He used a supervised correlation-based learning rule to train a separate output layer to demonstrate this. The idea of using a random recurrent network, which is left untrained, and which is processed by a simple classification/regression technique was later independently reinvented by Jaeger [17] as the Echo State Network and Maass [29] as the Liquid State Machine.

The Echo State Network (ESN) consists of a random, recurrent network of analog neurons that is driven by a (one- or multi-dimensional) time signal, and the activations of the neurons are used to do linear classification/regression. The ESN was introduced as a better way to use the computational power of RNNs without the hassle of training the internal weights, but from this viewpoint the reservoir acts as a complex nonlinear dynamic filter that transforms the input signals using a high-dimensional temporal map, not unlike the operation of an explicit, temporal kernel function. It is even possible to solve several classification tasks on an input signal by adding multiple readouts to a single reservoir. Jaeger proposed that for ESNs the reservoir should be scaled such that they operate on the edge of stability, by setting the spectral radius¹ of the connection matrix close to, but less than one.

The Liquid State Machine (LSM) by Maass [29] was originally presented as a general framework to perform real-time computation on temporal signals, but most description use an abstract cortical microcolumn model. Here a 3D structured locally connected network of spiking neurons is randomly created using biologically inspired parameters and excited by external input spike trains. The responses from all neurons are projected to the next cortical layer where

¹The magnitude of the largest eigenvalue. This is also a rough measure of the global scaling of the weights in the case of an even eigenvalue spread.

the actual training is performed. This projection is usually modeled using a simple linear regression function, but the description of the Liquid State Machine supports more advanced readout layers such as a parallel perceptron which is then trained using the P-delta learning rule. Also, descriptions of Liquid State Machines using other node types are published, such as a network of threshold logic gates [26]. In the LSM literature it has been shown that reservoir perform optimally if the dynamic regime is at the edge of chaos [27].

Yet other research by Steil [53] showed that the current state-of-the-art learning rule for RNNs (Athyra-Parlos recurrent learning [1]) actually has the same weight dynamics as what was proposed by Jaeger and Maass: the outputs weights are trained, and the internal weights are only globally scaled up or down a bit (which can be ignored if the weights are initially well scaled). This lead to a learning rule for RNNs called Backpropagation Decorrelation (BPDC) where again only the output weights are trained, but in this case the output nodes are nonlinear and the training is done in an on-line manner.

The fact that this idea was independently discovered several times, and that it is also discovered based on the theoretical study of state-of-the-art RNN learning rules, underlines the importance of the concept. However, the literature concerning these ideas was spread across different domains, and ESN and LSM research did not really interact. In [58], the authors of this tutorial proposed that the ideas should be unified² into a common research stream, which they propose to call Reservoir Computing (RC). The RC field is very active, with a special session at IJCNN 2005, a workshop at NIPS 2006, a special issue of Neural Network which will appear in April 2007 and the ESANN 2007 special session.

The ideas behind Reservoir Computing can be approached grounded in the field of recurrent neural network theory, but a different and equally interesting view can also be presented when reservoirs are compared to kernel methods [46]. Indeed, the key idea behind kernel methods is to pre-process the input by applying a form of transformation from the input space to the *feature space*, where the latter is usually far higher-dimensional (possibly even infinite-dimensional) than the former. The classification or regression is then performed in this feature space. The power of kernel methods lies mainly in the fact that this transformation does not need to be computed explicitly, which would be either too costly or simply impossible. This transformation into a higher-dimensional space is similar to the functionality of the reservoir, but there exist two major differences: first, in the case of reservoirs the transformation is explicitly computed, and second kernels are not equipped to cope with temporal signals.

2 Theory

In this section we will present the general RC setup, learning rules, reservoir creation and tuning rules, and theoretical results concerning the computational

²In that paper they showed that this unification by experimental validation and by presenting a MATLAB toolbox that supports both ESNs and LSMs.

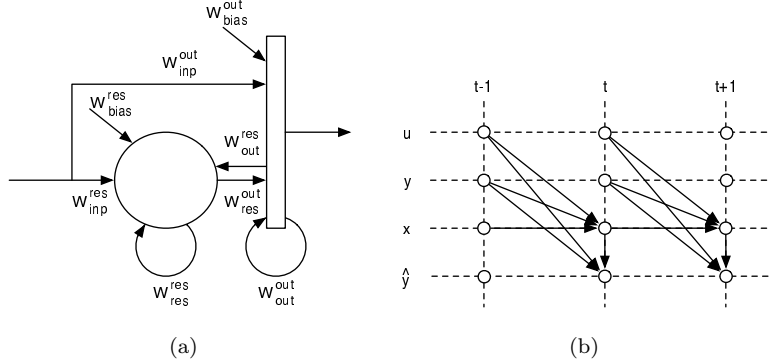


Fig. 1: The Figure a) shows the general model of an RC system, and Figure b) shows the exact timing setup in case of teacher-forced operation.

power and optimal dynamic regime.

General model Although there exist some variations of the global setup of an RC system, a generic architecture is depicted in Figure 1a. During training with teacher forced the state equations are:

$$\begin{aligned} \mathbf{x}(t+1) &= f(W_{\text{res}}^{\text{res}}\mathbf{x}(t) + W_{\text{inp}}^{\text{res}}\mathbf{u}(t) + W_{\text{out}}^{\text{res}}\mathbf{y}(t) + W_{\text{bias}}^{\text{res}}) \\ \hat{\mathbf{y}}(t+1) &= W_{\text{res}}^{\text{out}}\mathbf{x}(t+1) + W_{\text{inp}}^{\text{out}}\mathbf{u}(t) + W_{\text{out}}^{\text{out}}\mathbf{y}(t) + W_{\text{bias}}^{\text{out}}. \end{aligned}$$

Here, all weights matrices to the reservoir (W_{\cdot}^{res}) are initialized at random, while all connections to the output (W_{\cdot}^{out}) are trained. The exact timing of all the different components in these equations is important, and has been used differently in many publications. We propose, from a logical and functional point of view, the timing presented in Figure 1b. When using the system after training with the $W_{\text{out}}^{\text{res}}$ connections, the computed output $\hat{\mathbf{y}}$ is fed back into the reservoir.

When using spiking neurons (a LSM), the spikes need to be converted to analog values before they can be processed using for example linear regression. This conversion is usually done using an exponential filtering [26] and resampling of the spike train. If a readout would be implemented using a spiking learning rule, this conversion using filtering could be dropped, but as of now, no such readout function has been described.

Node types Many different neuron types have already been used in literature: linear nodes, threshold logic gates, hyperbolic tangent or Fermi neurons (with or without an extra leaky integrator) and spiking neurons (with or without synaptic models and dynamic synapses). There is yet no clear understanding on which node types are optimal for given applications. For instance, the network with

the best memory capacity³ consists of linear neurons, but on most other tasks, this node type is far from optimal. There has been some evidence [58] that spiking neurons might outperform analog neurons for certain tasks. And for spiking neurons, it has been shown that synapse models and dynamic synapses improve performance [13].

Adding node integration to analog neurons makes it possible to change the timescale at which the reservoir operates [22]. When making the transition from a continuous time-model to a discrete time-model using Euler integration, a time constant is introduced, that has the effect of “slowing” down the reservoir dynamics compared to the input.

Theoretical capabilities Besides the promising performance on a variety of practical applications, some work has also been done on the theoretical properties of reservoirs. Specifically, the first contributions on LSM and ESN both formulate and prove some necessary conditions that - in a general sense - reservoirs should possess in order to be good reservoirs.

The concept of the LSM - while strongly inspired by ideas from neuroscience - is actually rather broad when formulated theoretically [29]. First, a class of filters is defined that needs to obey the *point-wise separation property*, a quite nonrestrictive condition (all linear filters or delay lines obey it, but also spiking neurons with dynamic synapses). Next, a readout function is introduced that operates on the outputs of these filters and that needs to have the *universal approximation property*, a requirement that seems quite hard but which is for instance satisfied by a simple linear regression function. When both these conditions are satisfied, the resulting system can approximate every real-time (i.e. immediate) computation on temporal inputs. Thus, at least in theory, the LSM idea is computationally very powerful because in principle it can solve the majority of the difficult temporal classification and regression problems. Similarly, theoretical requirements for the ESN have been derived in [17]. In particular, necessary and sufficient conditions for a recurrent network to have *echo states* are formulated based on the spectral radius and largest singular value of the connection matrix. The *echo state property* means - informally stated - that the current state of the network is uniquely determined by the network input up to now, and, in the limit, not by the initial state. The network is thus state forgetting.

While these theoretical results are very important and necessary to add credibility to the idea of reservoir computing, unfortunately they do not offer any immediate indications on how to construct reservoirs that have certain ‘desirable’ properties given a certain task, nor is there consensus on what these desirable properties would be. Therefore, a lot of the current research on reservoirs focuses on reducing the randomness of the search for a good reservoir and offering heuristic or analytical measures of reservoir properties that predict the performance.

³Memory capacity (MC) is a measure of how much information contained in the input can be extracted from the reservoir after a certain time. See [18].

Reservoir creation and scaling Reservoirs are randomly created, and the exact weight distribution and sparsity has only a very limited influence on the reservoir's performance. But, in the ESN literature, the reservoirs are rescaled using measures based on stability bounds. Several of these measures have been presented in literature. Jaeger proposed the spectral radius to be slightly lower than one (because the reservoir is then guaranteed to have the echo state property). However, in [2] a tighter bound on the echo state property is described based on ideas from robust control theory, that is even exact for some special connection topologies. In LSM literature such measures do not exist, which makes it harder to construct reservoirs with a given desired dynamic 'regime' - an issue made even harder due to the dynamic properties of even simple spiking neurons (e.g. refractoriness).

In [36] it was shown that when no prior knowledge of the problem at hand is given, it is best to create the reservoir with a uniform pole placement, so that all possible frequencies are maximally covered, an idea which originated from the identification of linear systems using Kautz filters.

The random connectivity does not give a clear insight in what is going on in the reservoir. In [9] a reservoir was constructed consisting out of nodes that only have a random self-connection. This topology, called Simple ESN, consists effectively of a random filter bank and it was shown that its performance on several tasks is only slightly worse than a full ESN.

Training The original LSM concept stated that the dynamic reservoir states can be processed by any statistical classification or regression technique, ranging from simple linear models to kernel based techniques. The ESN literature however only uses linear regression as a readout function. Given the 'revival' of linear techniques (like kernel methods, Gaussian processes, ...) which are very well understood, and that adding other non-linear techniques after a reservoir makes it unclear what the exact influence of the non-linear reservoir is, we chose to only focus on linear processing.

The readout can both be trained using off-line (batch) or on-line learning rules. When performing off-line training, teacher forcing is used when the output is fed into the reservoir and readout. A weight matrix must be found that minimizes $(Aw - B)^2$ where the A matrix consists of a concatenation of all inputs to the readout (reservoir states, inputs, outputs and bias) and the B matrix consists of all desired outputs. Weights can be found using the Moore-Penrose pseudo inverse. In the original papers of Jaeger, added noise to the A matrix was used for regularization. It is however better to use e.g. ridge regression where the norm of the weights is also minimized using a regularization parameter. The regularization parameter can then be found using cross-validation.

On-line training of reservoirs using Least Mean Squares learning algorithm is quite inefficient due to a poor eigenvalue spread of the cross-correlation matrix. The Recursive Least Squares learning rule is better suited as has been shown in [19].

Reservoir Adaptation Although that the original RC concept uses fixed randomly created reservoirs, and this is considered to be its main advantage, there now is quite some research done on altering the reservoirs to improve performance on a given application.

One train of thought is to find an optimal reservoir given a certain task. This idea is based on the fact that the performance of randomly chosen reservoirs form a distribution. Given some search algorithm it is thus easy to perform better than average by choosing the right reservoir. This has been done with Genetic Algorithms [57], Evolino [45]. Another possibility is to start out with a large reservoir, and to prune away ‘bad’ nodes given a certain task. This idea is presented in Dutoit et al. in this session.

Maybe the purest idea of reservoir adaptation are techniques where the reservoir parameters are changed using an unsupervised learning rule, given an input, not the expected output! This way the dynamics can be tuned on the given input without explicitly specializing to an expected response. The first idea would be to use Hebbian based unsupervised learning rules, but in [20] Jaeger stated that these cannot significantly improve performance. This is probably because the simple correlation or anti-correlation from Hebbian rules is too limited to improve the reservoir dynamics.

Recently, a new way of adapting the reservoir parameters have been presented based on imposing certain output distributions on each of the reservoir nodes. This can be achieved by only changing global neuron parameters like gain and threshold, not all individual weights. These type of learning rules are called Intrinsic Plasticity, originally discovered in real neurons [64], derived for analog neurons by Triesch [56], and first used in RC by Steil [52]. IP is able to very robustly impose a given dynamic regime, whatever the network parameters. Certain tasks become tractable with reservoirs only when IP is used as shown in the contribution of Steil to this session. And tuning the IP distributions parameters has a smaller final performance variance than when setting the spectral radius as shown in the contribution by Verstraeten et al. in this session.

Structured reservoirs Initially, reservoirs were uniform structures with no specific internal structure. If we look at the brain, it is obvious that a specific structure is present, and is important. Recently, Haeussler et al. [13] showed that when creating an LSM with the global, statistical structure as what is discovered in the brain, the performance significantly increases as compared to an unstructured reservoir.

It has been theorized that a single reservoir is only able to support a limited number of ‘timescales’. This can be alleviated by using a structured reservoir as in [63]. Here several reservoirs are interconnected with sparse inhibitory connections with as aim to decorrelate the different sub-reservoirs. This way the different timescales can ‘live’ in their own sub-reservoir.

Measures of dynamics The performance of a reservoir system is highly dependent on the actual dynamics of the system. However, due to the high degree

of nonlinear feedback, a static analysis of the dynamic regime based on stability measures is bound to be inaccurate. In the original work on ESNs, Jaeger frequently uses the spectral radius of the reservoir connection matrix as a way of analyzing or controlling reservoir dynamics. Also, for linear systems a spectral radius is a hard limit for the stability of the dynamics, but this is only an approximation for nonlinear systems such as reservoirs and even then only for very small signal values (around a zero total input to the sigmoid, where it is somewhat linear). The actual reservoir dynamics, when applied to a specific task, are highly dependent on the actual inputs, since a reservoir is an input-driven system that never reaches a steady state. In [36] it was shown that the effective spectral radius of the linearized system that models the reservoir is very dependent on a constant bias input or the scaling of the input: large inputs or bias leads to a smaller effective spectral radius. It is even possible to construct a chaotic reservoir, which becomes transiently stable when driven by a large enough external signal [35].

These considerations, along with the quest for an accurate measure or predictor of reservoir performance, have led to the development of a measure of reservoir dynamics that also takes the actual task-dependent input into account. In [58], the application of a local Lyapunov-like measure to reservoir dynamics is introduced and its usability as a performance predictor is evaluated. It seems that for the tasks considered there, the pseudo-Lyapunov measure offers a good measure for the task-dependent reservoir dynamics and also links static parameters (such as the spectral radius) of reservoirs performing optimally on certain tasks to their actual dynamics.

3 Applications

Reservoir computing is generally very suited for solving temporal classification, regression or prediction tasks where very good performance can usually be attained without having to care too much about specifically setting any of the reservoir parameters. In real-world applications it is however crucial that the ‘natural’ time scale of the reservoir is tuned to be in the same order of magnitude as the important time scales of the temporal application.

Several successful applications of reservoir computing to both ‘abstract’ and real world engineering applications have been reported in literature. Abstract applications include dynamic pattern classification [18], autonomous sine generation [17] or the computation of highly nonlinear functions on the instantaneous rates of spike trains [31]. In robotics, LSMs have been used to control a simulated robot arm [24], to model an existing robot controller [5], to perform object tracking and motion prediction [4, 28], event detection [20, 15] or several applications in the Robocup competitions (mostly motor control) [34, 37, 43]. ESNs have been used in the context of reinforcement learning [6]. Also, applications in the field of Digital Signal Processing (DSP) have been quite successful, such as speech recognition [30, 60, 49] or noise modeling [21]. In [39] an application in Brain Machine Interfacing is presented. And finally, the use of reservoirs for chaotic

time series generation and prediction have been reported in [18, 19, 51, 50].

In many areas such as chaotic time series prediction and isolated digit recognition, RC techniques already outperform state-of-the-art approaches. A striking case is demonstrated in [21] where it is possible to predict the Mackey-Glass chaotic time series several orders of a magnitude farther in time than with classic techniques!

4 Implementations

The RC concept was first introduced using an RNN as a reservoir, but it actually is a much broader concept, where any high dimensional dynamic system with the right dynamic properties can be used as a temporal ‘kernel’, ‘basis’ or ‘code’ to pre-process the data such that it can be easily processed using linear techniques. Examples of this are the use of a real bucket of water as reservoir to do speech recognition [8], the genetic regulatory network in a cell [7, 23] and using the actual brain of a cat [32].

Software toolbox Recently, an open-source (GPL) toolbox was released⁴ that implements a broad range of RC techniques. This toolbox consists of functions that allow the user to easily set up datasets, reservoir topologies, experiment parameters and to easily process results of experiments. It supports analog (linear, sign, tanh, ..) and spiking (LIF) neurons in a transparent way. The spiking neurons are simulated in an optimized C++ event simulator. Most of what is discussed in this tutorial is also implemented in the toolbox.

Hardware At the time of writing, already several implementations of RC in hardware exist, but they all focus on neural reservoirs. A SNN reservoir is already implemented in digital hardware [47], while in EU FP6 FACETS project they are currently pursuing a large scale aVLSI implementation of a neuromorphic reservoir. An ANN reservoir has already been implemented on digital hardware using stochastic bit-stream arithmetic [59], but also in aVLSI [48].

5 Open research questions

RC is a relatively new research idea and there are still many open issues and future research directions. From a theoretical standpoint, a proper understanding of reservoir dynamics and measures for these dynamics is very important. A theory on regularization of the readout and the reservoir dynamics is currently missing. The influence of hierarchical and structured topologies is not yet well understood. Bayesian inference with RC needs to be investigated. Intrinsic plasticity based reservoir adaptation needs also be further investigated. From an implementation standpoint, the RC concept can be extended way beyond RNN, and any high-dimensional dynamical system, operating in the proper dynamical

⁴It is freely available at <http://www.elis.ugent.be/rct>.

regime (which is as of now theoretically still quite hard to define), can be used as a reservoir.

References

- [1] A. F. Atiya and A. G. Parlos. New results on recurrent network training: Unifying the algorithms and accelerating convergence. *IEEE Neural Networks*, 11:697, 2000.
- [2] M. Buehner and P. Young. A tighter bound for the echo state property. *IEEE Neural Networks*, 17(3):820–824, 2006.
- [3] D. V. Buonomano and M. M. Merzenich. Temporal information transformed into a spatial code by a neural network with realistic properties. *Science*, 267:1028–1030, 1995.
- [4] H. Burgsteiner. *On learning with recurrent spiking neural networks and their applications to robot control with real-world devices*. PhD thesis, Graz University of Technology, 2005.
- [5] H. Burgsteiner. Training networks of biological realistic spiking neurons for real-time robot control. In *Proc. of EANN*, pages 129–136, Lille, France, 2005.
- [6] K. Bush and C. Anderson. Modeling reward functions for incomplete state representations via echo state networks. In *Proc. of IJCNN*, 2005.
- [7] X. Dai. Genetic regulatory systems modeled by recurrent neural network. In *Proc. of ISNN*, pages 519–524, 2004.
- [8] C. Fernando and S. Sojakka. Pattern recognition in a bucket. In *Proc. of ECAL*, pages 588–597, 2003.
- [9] G. Fette and J. Eggert. Short term memory and pattern matching with simple echo state networks. In *Proc. of ICANN*, pages 13–18, 2005.
- [10] K.-I. Funahashi and N. Y. Approximation of dynamical systems by continuous time recurrent neural networks. *Neural Networks*, 6:801–806, 1993.
- [11] F. Gers and J. Schmidhuber. Lstm recurrent networks learn simple context free and context sensitive languages. *IEEE Neural Networks*, 12(6):1333–1340, 2001.
- [12] A. Graves, D. Eck, N. Beringer, and J. Schmidhuber. Biologically plausible speech recognition with LSTM neural nets. In *Proc. of Bio-ADIT*, pages 127–136, 2004.
- [13] S. Haeusler and W. Maass. A statistical analysis of information-processing properties of lamina-specific cortical microcircuit models. *Cerebral Cortex*, 17(1):149–162, 2006.
- [14] B. Hammer and J. J. Steil. Perspectives on learning with recurrent neural networks. In *Proc. of ESANN*, 2002.
- [15] J. Hertzberg, H. Jaeger, and F. Schönherr. Learning to ground fact symbols in behavior-based robots. In *Proc. of ECAI*, pages 708–712, 2002.
- [16] J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. In *Proc. of the National Academy of Sciences of the USA*, volume 79, pages 2554–2558, 1982.
- [17] H. Jaeger. The “echo state” approach to analysing and training recurrent neural networks. Technical report, German National Research Center for Information Technology, 2001.
- [18] H. Jaeger. Short term memory in echo state networks. Technical report, German National Research Center for Information Technology, 2001.
- [19] H. Jaeger. Adaptive nonlinear system identification with echo state networks. In *Proc. of NIPS*, pages 593–600, 2003.
- [20] H. Jaeger. Reservoir riddles: Suggestions for echo state network research (extended abstract). In *Proc. of IJCNN*, pages 1460–1462, August 2005.
- [21] H. Jaeger and H. Haas. Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless telecommunication. *Science*, 308:78–80, 2004.

- [22] H. Jaeger, M. Lukosevicius, and D. Popovici. Optimization and applications of echo state networks with leaky integrator neurons. *Neural Networks*, 2007. to appear.
- [23] B. Jones, D. Stekelo, J. Rowe, and C. Fernando. Is there a liquid state machine in the bacterium escherichia coli? *IEEE Artificial Life*, submitted, 2007.
- [24] P. Joshi and W. Maass. Movement generation and control with generic neural microcircuits. In *Proc. of BIO-AUDIT*, pages 16–31, 2004.
- [25] J. Kilian and H. Siegelmann. The dynamic universality of sigmoidal neural networks. *Information and Computation*, 128:48–56, 1996.
- [26] R. Legenstein and W. Maass. *New Directions in Statistical Signal Processing: From Systems to Brain*, chapter What makes a dynamical system computationally powerful? MIT Press, 2005.
- [27] R. A. Legenstein and W. Maass. Edge of chaos and prediction of computational performance for neural microcircuit models. *Neural Networks*, in press, 2007.
- [28] W. Maass, R. A. Legenstein, and H. Markram. A new approach towards vision suggested by biologically realistic neural microcircuit models. In *Proc. of the 2nd Workshop on Biologically Motivated Computer Vision*, 2002.
- [29] W. Maass, T. Natschläger, and H. Markram. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Computation*, 14(11):2531–2560, 2002.
- [30] W. Maass, T. Natschläger, and H. Markram. A model for real-time computation in generic neural microcircuits. In *Proc. of NIPS*, pages 229–236, 2003.
- [31] W. Maass, T. Natschläger, and M. H. Fading memory and kernel properties of generic cortical microcircuit models. *Journal of Physiology*, 98(4-6):315–330, 2004.
- [32] D. Nikolić, S. Häusler, W. Singer, and W. Maass. Temporal dynamics of information content carried by neurons in the primary visual cortex. In *Proc. of NIPS*, 2007. in press.
- [33] C. W. Omlin and C. L. Giles. Constructing deterministic finite-state automata in sparse recurrent neural networks. In *Proc. of ICNN*, pages 1732–1737, 1994.
- [34] M. Oubbati, P. Levi, M. Schanz, and T. Buchheim. Velocity control of an omnidirectional robocup player with recurrent neural networks. In *Proc. of the Robocup Symposium*, pages 691–701, 2005.
- [35] M. C. Ozturk and J. C. Principe. Computing with transiently stable states. In *Proc. of IJCNN*, 2005.
- [36] M. C. Ozturk, D. Xu, and J. C. Principe. Analysis and design of echo state networks. *Neural Computation*, 19:111–138, 2006.
- [37] P. G. Plöger, A. Arghir, T. Günther, and R. Hosseiny. Echo state networks for mobile robot modeling and control. In *RoboCup 2003*, pages 157–168, 2004.
- [38] G. V. Puskorius and F. L. A. Neurocontrol of nonlinear dynamical systems with kalman filter trained recurrent networks. *IEEE Neural Networks*, 5:279–297, 1994.
- [39] Y. Rao, S.-P. Kim, J. Sanchez, D. Erdogmus, J. Principe, J. Carmena, M. Lebedev, and M. Nicolelis. Learning mappings in brain machine interfaces with echo state networks. In *Proc. of ICASSP*, pages 233–236, 2005.
- [40] A. J. Robinson. An application of recurrent nets to phone probability estimation. *IEEE Neural Networks*, 5:298–305, 1994.
- [41] P. Rodriguez. Simple recurrent networks learn context-free and context-sensitive languages by counting. *Neural Computation*, 13(9):2093–2118, 2001.
- [42] D. Rumelhart, G. Hinton, and R. Williams. *Parallel Distributed Processing*, chapter Learning internal representations by error propagation. MIT Press, 1986.
- [43] M. Salmen and P. G. Plöger. Echo state networks used for motor control. In *Proc. of ICRA*, pages 1953–1958, 2005.

- [44] J. Schmidhuber and S. Hochreiter. Long short-term memory. *Neural Computation*, 9:1735–1780, 1997.
- [45] J. Schmidhuber, D. Wierstra, M. Gagliolo, and F. Gomez. Training recurrent networks by evoluno. *Neural Computation*, 19:757–779, 2007.
- [46] B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization and Beyond*. MIT Press, 2002.
- [47] B. Schrauwen, M. D’Haene, D. Verstraeten, and J. Van Campenhout. Compact hardware for real-time speech recognition using a liquid state machine. In *Proc. of IJCNN*, 2007. submitted.
- [48] F. Schürmann, K. Meier, and J. Schemmel. Edge of chaos computation in mixed-mode vlsi - ”a hard liquid”. In *Proc. of NIPS*, 2004.
- [49] M. D. Skowronski and J. G. Harris. Minimum mean squared error time series classification using an echo state network prediction model. In *Proc. of ISCAS*, 2006.
- [50] J. Steil. Memory in backpropagation-decorrelation $o(n)$ efficient online recurrent learning. In *Proc. of ICANN*, 2005.
- [51] J. Steil. Online stability of backpropagation-decorrelation recurrent learning. *Neurocomputing*, 69:642–650, 2006.
- [52] J. Steil. Online reservoir adaptation by intrinsic plasticity for backpropagation-decorrelation and echo state learning. *Neural Networks*, 2007. to appear.
- [53] J. J. Steil. Backpropagation-Decorrelation: Online recurrent learning with $O(N)$ complexity. In *Proc. of IJCNN*, volume 1, pages 843–848, 2004.
- [54] J. Suykens, J. Vandewalle, and B. De Moor. *Artificial Neural Networks for Modeling and Control of Non-Linear Systems*. Springer, 1996.
- [55] F. Takens, D. A. Rand, and Y. L.-S. Detecting strange attractors in turbulence. In *Dynamical Systems and Turbulence*, volume 898 of *Lecture Notes in Mathematics*, pages 366–381. Springer-Verlag, 1981.
- [56] J. Triesch. Synergies between intrinsic and synaptic plasticity in individual model neurons. In *Proc. of NIPS*, volume 17, 2004.
- [57] T. van der Zant, V. Becanovic, K. Ishii, H.-U. Kobialka, and P. G. Plöger. Finding good echo state networks to control an underwater robot using evolutionary computations. In *Proc. of IAV*, 2004.
- [58] D. Verstraeten, B. Schrauwen, M. D’Haene, and D. Stroobandt. A unifying comparison of reservoir computing methods. *Neural Networks*, 2007. to appear.
- [59] D. Verstraeten, B. Schrauwen, and D. Stroobandt. Reservoir computing with stochastic bitstream neurons. In *Proc. of ProRISC Workshop*, pages 454–459, 2005.
- [60] D. Verstraeten, B. Schrauwen, D. Stroobandt, and J. Van Campenhout. Isolated word recognition with the liquid state machine: a case study. *Information Processing Letters*, 95(6):521–528, 2005.
- [61] P. J. Werbos. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. PhD thesis, Harvard University, Boston, MA, 1974.
- [62] P. J. Werbos. Backpropagation through time: what it does and how to do it. *Proc. IEEE*, 78(10):1550–1560, 1990.
- [63] Y. Xue, L. Yang, and S. Haykin. Decoupled echo state networks with lateral inhibition. *IEEE Neural Networks*, 10(10), 2007.
- [64] W. Zhang and D. J. Linden. The other side of the engram: Experience-driven changes in neuronal intrinsic excitability. *Nature Reviews Neuroscience*, 4:885–900, 2003.