KI - Künstliche Intelligenz manuscript No.
(will be inserted by the editor)

Reservoir Computing Trends

Mantas Lukoševičius · Herbert Jaeger · Benjamin Schrauwen

Received: March 1, 2012 / Accepted: date

Abstract Reservoir Computing (RC) is a paradigm of understanding and training Recurrent Neural Networks (RNNs) based on treating the recurrent part (the *reservoir*) differently than the readouts from it. It started ten years ago and is currently a prolific research area, giving important insights into RNNs, practical machine learning tools, as well as enabling computation with non-conventional hardware. Here we give a brief introduction into basic concepts, methods, insights, current developments, and highlight some applications of RC.

Keywords Reservoir computing · Recurrent neural network · Echo state network

1 Introduction

About ten years ago a new trend of understanding, training, and using Recurrent Neural Networks (RNNs) has been started with Echo State Networks (ESNs) [19, 21] and Liquid State Machines (LSMs) [34]. While the former came from the field of Machine Learning (ML) and the latter from computational neuroscience, both approaches share the same basic idea. It stems from the observation that, as long as an RNN possessed certain generic properties, supervised adaptation of all interconnection weights is not necessary, and only training a memoryless supervised readout from it is enough to obtain excellent performance in many tasks. The RNN is called a *reservoir* in this context.

This was a welcome discovery, since training RNNs has always been much more difficult than training feedforward neural networks. Cyclic dependencies in RNNs lead to bifurcations during training: infinitesimally small changes to RNN parameters can lead to drastic discontinuous changes in its behavior. This phenomenon may render classical gradient descent RNN training methods (like [52,53]) non converging [11]. Even if they do converge, this process is typically slow, computationally expensive, requires careful selection of learning parameters, and ends in a local minimum. Learning long-term dependencies in the data is hard [2] (but see [15] for an RNN architecture specialized on learning such dependencies, and [35] for recent progress in generic RNNs). Because of the complexity and computational costs, the number of neurons used in so-trained RNNs has typically been in the order of tens, which in turn limits their expressive capacity.

The approach started by ESNs and LSMs reinvigorated interest in RNN research and applications, a stream which became collectively known as Reservoir Computing (RC) [49]. It now has many more related methods and extensions of the original idea (see [30] for an extensive overview; `reservoir-computing.org` is a web portal collectively maintained by leading RC groups). We will mention a few selected variants here, but let us start with the original basic ESN RC approach.

2 The basic ESN approach

Here are the update equations of a typical RNN used in ML with leaky-integrated discrete-time continuous-

Mantas Lukoševičius · Herbert Jaeger
Jacobs University Bremen, Campus Ring 1,
28759 Bremen, Germany
E-mail: {m.lukosevicius, h.jaeger}@jacobs-university.de

Benjamin Schrauwen
Ghent University, Sint Pietersnieuwstraat 41,
9000 Ghent, Belgium
E-mail: benjamin.schrauwen@ugent.be

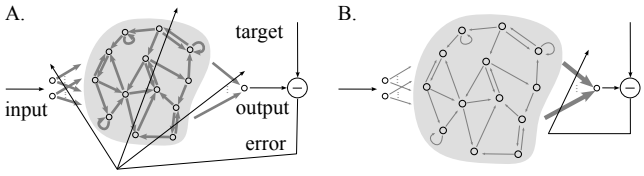

Fig. 1 The difference between a full gradient descent training of RNN (A.) and the ESN training (B.).

value units:

$$\tilde{\mathbf{x}}(n) = \tanh\left(\mathbf{W}^{\text{in}}[1;\mathbf{u}(n)] + \mathbf{W}\mathbf{x}(n-1)\right), \tag{1}$$

$$\mathbf{x}(n) = (1-\alpha)\mathbf{x}(n-1) + \alpha\tilde{\mathbf{x}}(n), \tag{2}$$

where n is discrete time, $\mathbf{u}(n) \in \mathbb{R}^{N_{\text{u}}}$ is the input signal, $\mathbf{x}(n) \in \mathbb{R}^{N_{\text{x}}}$ is a vector of reservoir neuron activations and $\tilde{\mathbf{x}}(n) \in \mathbb{R}^{N_{\text{x}}}$ is its update, all at time step n, $\tanh(\cdot)$ is applied element-wise, $[\cdot;\cdot]$ stands for a vertical vector concatenation, $\mathbf{W}^{\text{in}}$ and $\mathbf{W}$ are the input and recurrent weight matrices respectively, and $\alpha \in (0,1]$ is the leaking rate. The model is also frequently used without the leaky integration, which is a special case obtained by setting $\alpha = 1$ and thus $\tilde{\mathbf{x}}(n) \equiv \mathbf{x}(n)$. The linear readout layer is defined as

$$\mathbf{y}(n) = \mathbf{W}^{\text{out}}[1;\mathbf{u}(n);\mathbf{x}(n)], \tag{3}$$

where $\mathbf{y}(n) \in \mathbb{R}^{N_{\text{y}}}$ is network output, and $\mathbf{W}^{\text{out}}$ the output weight matrix. An additional nonlinearity can be applied to $\mathbf{y}(n)$ in (3), as well as feedback connections $\mathbf{W}^{\text{fb}}$ from $\mathbf{y}(n-1)$ to $\tilde{\mathbf{x}}(n)$ in (1).

The original method of RC introduced with ESNs [19] is to:

- generate a large random reservoir $(\mathbf{W}^{\text{in}}, \mathbf{W}, \alpha)$;
- run it using the training input $\mathbf{u}(n)$ and collect the corresponding reservoir activation states $\mathbf{x}(n)$;
- compute the linear readout weights $\mathbf{W}^{\text{out}}$ from the reservoir using linear regression, minimizing the mean square error of the network output w.r.t. the training target signal $\mathbf{y}_{\text{target}}(n)$;
- use the trained network on new input data $\mathbf{u}(n)$ by computing $\mathbf{y}(n)$ employing the trained output weights $\mathbf{W}^{\text{out}}$.

Let us look at these steps in more detail.

For the approach to work, the reservoir must possess the *echo state property*, which can roughly be described as fading memory of the input: trajectories of the reservoir state should converge given the same input, irrespective of the previous history. This is typically ensured by appropriately scaling recurrent connection weights $\mathbf{W}$ [19]. A few other parameters, most importantly the input weight $\mathbf{W}^{\text{in}}$ scaling and leaking rate α, should also be adjusted for an optimal validation performance in a given task.

While running the generated model with training data, the vectors $[1;\mathbf{u}(n);\mathbf{x}(n)]$ as in (3) are collected into a matrix $\mathbf{X}$ and the desired teacher targets $\mathbf{y}_{\text{target}}(n)$ into a matrix $\mathbf{Y}_{\text{target}}$, both having a column for every training time step n. The training is typically done by computing the output weights via ridge regression

$$\mathbf{W}^{\text{out}} = \mathbf{Y}_{\text{target}}\mathbf{X}^{\text{T}}(\mathbf{X}\mathbf{X}^{\text{T}} + \gamma^2\mathbf{I})^{-1}, \tag{4}$$

where $\mathbf{I}$ is the identity matrix and γ is a regularization parameter. For optimal results γ should also be selected through validation; note that the network needs no rerunning with a different γ to recompute $\mathbf{W}^{\text{out}}$. By avoiding training of RNN connections $\mathbf{W}$, the learning is done in a single pass through training data and the optimal output weights $\mathbf{W}^{\text{out}}$ are computed with a high precision using a closed-form solution (4). This also enables a practical use of reservoirs with size of thousands or even tens of thousands of units on contemporary computers [46]. Also note, that $\mathbf{Y}_{\text{target}}\mathbf{X}^{\text{T}}$ and $\mathbf{X}\mathbf{X}^{\text{T}}$ in (4) can be computed incrementally and stored in the memory, instead of $\mathbf{Y}_{\text{target}}$ and $\mathbf{X}$, for arbitrary long training data sequences. Alternatively, $\mathbf{W}^{\text{out}}$ can be continuously adapted by an online learning algorithm [19].

Such simple and efficient RNN training was demonstrated to outperform fully-trained RNNs in many benchmark tasks, e.g., [22,23,50,17,46]. Some examples of applications are presented in Section 7.

3 Perspectives on RC

The principles of RC can be perceived from several different perspectives.

There are certain parallels between RC and *kernel* methods in ML. The reservoir can be seen as a nonlinear high-dimensional expansion $\mathbf{x}(n)$ of the input signal $\mathbf{u}(n)$. For classification tasks, input data $\mathbf{u}(n)$ which are not linearly separable in the original space $\mathbb{R}^{N_{\text{u}}}$, often become linearly separable in the expanded space $\mathbb{R}^{N_{\text{x}}}$ of $\mathbf{x}(n)$ where they are separated by $\mathbf{W}^{\text{out}}$. At the same time, the reservoir serves as a memory, providing the temporal context. The "kernel trick" is typically not used in RC, however it is possible to do so by defining recursive temporal context-sensitive kernels that integrate over a continuum of $\mathbf{W}^{\text{in}}$ and $\mathbf{W}$, which can be used as in regular Support Vector Machines (SVMs), but for sequence data [13]. SVM-style readouts can also be trained from the reservoirs [41].

The separation between the fixed reservoir and the adaptive readout can also be arrived at when analyzing the dynamics of a full gradient descent RNN training, and optimizing it. In an efficient version of gradient descent RNN training introduced by Atiya and Parlos [1] the output weights $\mathbf{W}^{\text{out}}$ are adapted much more than $\mathbf{W}$ and $\mathbf{W}^{\text{in}}$ [38], which led to a further optimization where they remain constant, an RC method called BackPropagation-DeCorrelation (BPDC) [43]. BPDC is an online RNN learning algorithm which runs with $O(N_{\text{x}})$ time complexity.

From a biological perspective, RC gives a simple and yet powerful interpretation of how generic cortical circuits with no well-understood supervised adaptation can be utilized for purposeful computation [34]. Reservoirs also correspond well to how temporal information is spatially encoded in the brain and provides a context for interpreting current inputs [6]. Fixed RNNs for modeling parts of sensory-motor sequence [8] and speech [10] learning architectures have been employed even before the original ESN and LSM publications.

Another advantage or RC is that the same reservoir can be used as a generic computational substrate for multiple tasks concerning the same input. For each task a new readout can be learned independently and without interfering with what has been learned before. This might have a potential in aiming for a general purpose artificial intelligence mechanisms and corresponds well to the natural intelligence.

4 Other RC approaches

Despite the success of the original ESN approach depicted in Section 2, there are many extensions, modifications and improvements possible.

For example, intuitively, there should be something better than a random reservoir. The error landscape of the RNN parameter space for a given task is usually notoriously complicated (this is why gradient descent is difficult). The probability of exactly hitting the global, or even local, minimum in this landscape by picking a random point is virtually equal to zero. Often slight, but always present variations of performance among randomly sampled reservoirs confirm this.

The linear readout is also quite limited in its expressive power.

Guided by such intuitions the modern field of RC substantially widened and differentiated. It has moved away from the initial paradigm of having a fixed RNN and training only the output. However, what still sets the RC approach apart from other RNN training methods is that *the recurrent part (the reservoir) is generated or trained differently than the readout.* This has become the modern paradigm of RC.

The RC paradigm of separating the reservoir and readout training allows for these two research directions to be pursued virtually independently, and the best results from both to be combined. There are numerous different methods proposed in the literature for both of the directions.

Output training is in essence a standard ML problem, where virtually any method capable of learning an input-to-output mapping can be employed with their respective strengths and weaknesses.

For the reservoir part, there has been also a large number of proposals in the literature. They can roughly be classified into three categories:

- Generic methods for generating RNNs with different neuron models, connectivity patterns and dynamics;
- Unsupervised adaptation of the reservoir, based on the input data $\mathbf{u}(n)$, but not $\mathbf{y}_{\text{target}}(n)$;
- Supervised, or semi-supervised, like reinforcement learning, adaptation of the reservoir, using task-specific information from both $\mathbf{u}(n)$ and $\mathbf{y}_{\text{target}}(n)$, but exploiting it differently than for the readout training.

Since the readout training can be very efficient, the quality of a reservoir for a particular task can be tested quickly by measuring the error of the readout. This makes RC a convenient and popular testing ground for many types of RNN models, topologies, unsupervised, reinforcement, and biologically inspired adaptation algorithms.

Most of these different approaches are reviewed in [30], updated in [29]. For the sake of brevity only a few of those have been mentioned here.

The numerous proposed RC modification introduced multiple improvements, often case-specific, extending the power of RC to new domains, and offering new insights into the workings of RNNs. The original ESN approach of Section 2, however, still holds its ground for its combination of simplicity and power.

5 Beyond neural networks

The RC principle can also be seen as a strategy to implement useful computations on generic dynamical systems, treating them as reservoirs, either in simulations or even in physical instantiations. Thus RC has spread well beyond the world of artificial neural networks. In particular, it enables useful computation on hardware platforms where, e.g., it is hard or just impractical to implement equivalents of basic electronic logic gates and memory cells. Potential and functioning examples

include: analog electronics [40,39], randomly crystallized nonlinear electronic networks [44], opto-electronic [26,36] and optical [47] systems, or just a bucket filled with water [12]. Readouts from such reservoirs are typically implemented in more conventional ways.

Many of these directions are very active research areas. In a long run such non-neural, physical reservoirs might significantly complement or even, for some domains, replace the omni-present digital electronic computers.

6 Training of the dynamics

Even if the reservoir is kept fixed, for some tasks the trained readouts are fed back to the reservoir and thus the training process changes its dynamics. In other words, a recurrence exists between the reservoir and the readout. Pattern generation is a typical example of such task. This is either realized by feedback connections $\mathbf{W}^{\text{fb}}$ from the trained output $\mathbf{y}(n-1)$ to the reservoir $\tilde{\mathbf{x}}(n)$, or by looping the output $\mathbf{y}(n-1)$ as an input $\mathbf{u}(n)$ for the next update step n in a predictive generator mode in (1). Note, that these two options are equivalent and just a matter of notation: $\mathbf{u}(n)$ and $\mathbf{W}^{\text{in}}$ instead of $\mathbf{y}(n-1)$ and $\mathbf{W}^{\text{fb}}$. In some cases, however, both external input and output feedback can be present.

This extends the power of RC, because it no longer relies on fixed random input-driven dynamics to construct the output, but the dynamics are adapted to the task. This power has its price, because stability issues arise here. In order to avoid falling prey to the same difficulties as with full RNN training algorithms, two strategies are used in RC.

The first strategy is to disengage the recurrent relationship between the reservoir and the readout using *teacher forcing* and treat output learning as a feedforward task. This is done by feeding the desired output $\mathbf{y}_{\text{target}}(n-1)$ through the feedback connections $\mathbf{W}^{\text{fb}}$ (or $\mathbf{W}^{\text{in}}$) instead of the real output $\mathbf{y}(n-1)$ while learning. The target signal $\mathbf{y}_{\text{target}}(n)$ "bootstraps" the learning process and if the output is learned with high precision (i.e., $\mathbf{y}(n) \approx \mathbf{y}_{\text{target}}(n)$), the recurrent system runs much the same way with the real $\mathbf{y}(n)$ in feedbacks after training as it did with $\mathbf{y}_{\text{target}}(n)$ during training.

There are some caveats here. The approach works very well if the output can be learned precisely [19]. However, if this is not the case, the distorted feedback leads to an even more distorted output and feedback at the next time step, and so on, with the actual generated output $\mathbf{y}(n)$ quickly diverging from the desired $\mathbf{y}_{\text{target}}(n)$. Even with well-learned outputs the dynamical stability of the autonomous running system is often an issue. In both cases the problem is alleviated by some kind of regularization of the weights or "immunization" of the state and/or the feedbacks with noise.

The second strategy is using specialized RC learning algorithm to train the outputs $\mathbf{W}^{\text{out}}$ while the real feedbacks are present. The before-mentioned BPDC algorithm is an efficient online option with an optimal time complexity [43]. A recent approach named *FORCE* learning uses a powerful 2nd-order online learning algorithm to vigorously adapt $\mathbf{W}^{\text{out}}$ in the presence of the real feedbacks [45]. By the initial fast and strong adaptation of $\mathbf{W}^{\text{out}}$ the feedbacks $\mathbf{y}(n)$ are kept close to the desired $\mathbf{y}_{\text{target}}(n)$ already from the beginning of the learning process, similar to teacher forcing. It appears that FORCE learning is well suited to yield very stable and accurate neural pattern generators.

7 Applications

RC methods have been widely employed in more or less academic applications. The nature of these applications spans all kinds that are amenable to supervised modeling of temporal systems, e.g., temporal pattern classification, temporal pattern generation, time series prediction, timing, routing, memorizing, or controlling nonlinear systems. We refrain from giving an ad hoc selection here; googling *"echo state" network application* will retrieve a few hundreds of relevant instances.

Useful hints for setting up RC learning systems for practical tasks are given in [20,48]. It should be clearly appreciated that, like always in machine learning, achieving very good results requires experience, experimentation, and insight into the nature of the respective task. Furthermore, an understanding of basic principles of machine learning is a necessary precondition. Specifically, an insightful use of cross-validation and regularization is key for good performance. RC is not a miracle method that can be used out-of-the-box and then be expected to excel.

Instead of attempting a comprehensive overview, we will highlight a number of applications in which the authors have been (or still are) personally involved.

Speech recognition. One of the textbook examples of temporal sequence recognition is speech recognition. ESNs and LSMs have already early on been applied to this domain. The first approaches focused specifically on isolated recognition of Japanese vowels [20] and digits [51,49]. The first attempt to continuous speech recognition was based on a rather atypical setup: a large committee of predictive classifiers using ESNs [42]. It showed good results on a benchmark dataset, but due to the use of a custom acous-

tic front-end, it is not trivial to compare to state-of-the-art work. More recently, in the European FP7 project ORGANIC (reservoir-computing.org/organic) which set out to establish neurodynamical architectures as viable alternative to statistical methods for speech and handwriting recognition, different approaches to speech recognition have been applied. In [46] it was demonstrated that competitive phoneme recognition rates can be achieved using straightforward application of the ESN setup on a hard benchmark dataset. Based on this front-end, ESN-HMM hybrids are currently being investigated to realize word recognition with excellent results. Research on noise-robust recognition using ESNs [24] also demonstrate that they perform better than classic HMM approaches.

Handwriting recognition. Handwriting recognition is in many respects very similar to speech recognition, and traditionally similar computational approaches have been employed [5]. Therefore it is no coincidence that the Organic project also hosts an industrial partner who develops text recognition solutions, e.g., for car number plate reading (easy) or address recognition in automated postal parcel sorting plants (difficult). This partner, Planet intelligent systems GmbH, has been developing ESN-based recognition modules in a long-standing cooperation with the Machine Learning group at Jacobs University. Important customers of Planet's parcel sorting technology are FedEx and the US Postal Services. ESN-based offline text recognition functions by scanning the text with a virtual linear camera from left to right, obtaining a time series of pixel vectors, which is passed to a hierarchical reservoir recognizer architecture. On subsequent layers, increasingly aggregate "chunks" are recognized (e.g., letters → words). Importantly, no explicit segmentation routine is necessary ("segmentation-free" processing). The different layers are trained individually in a supervised way, which requires training data that are teacher-annotated on each representational level. Planet seeks collaboration with academic partners, and – quite remarkably – allows scientific results which emerge from such collaborations to be published (e.g., [31,23]). Planet furthermore has made its very large annotated training dataset available to the scientific community as a benchmark (reservoir-computing.org/organic/benchmarks/294).

Robot motor control. ESNs can be conveniently trained as deadbeat controllers for nonlinear plants. The setup for such controllers is detailed in the original ESN patent document [18] and had first been employed in practice for the tracking control of omniwheel Robocup robots at Fraunhofer AIS (now Fraunhofer IAIS) [37]. The training principle is to feed the controller ESN with the current plant output observation and an n-timestep-delayed version of the same, which enables the ESN to acquire an nth order model of the plant. In exploitation, the direct plant output feedback channel is replaced by the reference signal while the input which in training received the delayed output observation now receives the direct observations. In a very different way, ESNs are currently being explored as neural pattern generators for the humanoid iCub robot (www.icub.org) within the European FP7 project AMARSi (www.amarsi-project.eu). Here, the objective is to obtain neural pattern generators which can be modulated by higher-level control input, e.g., in order to adapt frequency, amplitude, offset, phase, or waveform of the generated pattern. Modulatable neural pattern generation is an extensive field of research [16]. The innovation offered by ESNs is to obtain a *generic* learning mechanism by which an existing neural pattern generator can acquire essentially arbitrary novel modes of modulatability by learning [28].

Financial forecasting. Here is an episode worth telling. In a graduate seminar held at Jacobs University in 2007, a group of five students with no previous exposure to machine learning engaged in an international financial time series prediction contest (www.neural-forecasting-competition.com/NN3). The competition data consisted in a set of 111 time series of very diverse nature (it was part of the challenge to develop versatile predictors). Within 3 months, the students acquired the basic knowledge of standard data preprocessing methods used in the field, applied them to the raw data, developed ESN predictors, implemented them, submitted their predictions - and won the contest, against competitors with years of professional experience in financial forecasting [17]. An informal account of this story is given at minds.jacobs-university.de/teaching/highlights. The predictions were obtained by combining the outputs of ensembles of 500 independently created reservoirs whose sizes ranged around 100 units. This episode underlines the simplicity of RC modeling and its motivational capacity in education as much as it illustrates its modeling performance.

Medical. Ghent University has been actively pursuing the use of ESNs in bio-medical applications with great success. It has been applied to real-time detection of epileptic seizures, and this with very low

latency and high accuracy, outperforming the state-of-the-art [7]. This technology would enable treatments for epilepsy that are based on closing-the-loop: rapidly detecting the seizure and actively counteracting it using, e.g., medication or brain stimulation. Based on the good results on seizure detection, we also started investigating various forms of Brain Computer Interfaces (BCIs). The most overt result here was that ESNs are very good at detecting the so called "rest state": the interval between specific thoughts. Combining ESNs with Common Spatial Patterns lead to state-of-the-art results in motor imagery BCI [25].

Here we mentioned only applications in engineering and machine learning. This is one of two main directions of utilizing RC, the other being to model biological phenomena in the cognitive and neurosciences. This is often done with more biologically plausible reservoirs made up from spiking neurons, and is mostly associated with the "liquid state machine" flavor of RC. Pioneers in this area are Peter F. Dominey and Wolfgang Maass. Dominey actually was the first to explicitly spell out the RC principle as early as 1995 [8], and ever since he has continued to extend and refine his models of the corticostriatal processing loop for temporal sequence learning (e.g., [10,9,14]). Maass et al. widely explored the RC principle to understand generic computational properties of cortical microcircuits (e.g., [34,33,32]). Recently he and his group have added reinforcement learning [27] and Bayesian inferencing [4] to the picture of microcircuit adaptation. RC principles have been taken up by other leading researchers in computational neuroscience (e.g., [45,3]).

8 Resources

Leading European RC groups jointly maintain an RC web portal at `reservoir-computing.org`. Here potential users can find introductory tutorials, an extensive bibliography, an option to subscribe to an RC mailing list, and links to a choice of RC tools. Among the latter we want to point out the OGER engine, a very comprehensive Python-based toolbox with interfaces to a number of standard (spiking) neural simulators (supporting the computational neuroscience branch of RC) and numerous pre-installed validation, regularization, and optimization methods supporting the machine learning side of RC. This engine has been developed within the Organic FP7 project.

Acknowledgements The authors acknowledge support by the European FP7 project ORGANIC (`reservoir-computing.org/organic`). **Patent note.** The basic ESN architecture and algorithm are protected for commercial use by international patents held by the Fraunhofer Society [18].

References

1. Atiya, A.F., Parlos, A.G.: New results on recurrent network training: unifying the algorithms and accelerating convergence. IEEE Transactions on Neural Networks **11**(3), 697–709 (2000)
2. Bengio, Y., Simard, P., Frasconi, P.: Learning long-term dependencies with gradient descent is difficult. IEEE Transactions on Neural Networks **5**(2), 157–166 (1994)
3. Bernacchia, A., Seo, H., Lee, D., Wang, X.J.: A reservoir of time constants for memory traces in cortical neurons. Nature Neuroscience **14**(3), 366–372 (2011)
4. Buesing, L., Bill, J., Nessler, B., Maass, W.: Neural dynamics as sampling: A model for stochastic computation in recurrent networks of spiking neurons. PLoS Comp. Biol. **7**(11), e1002,211 (2011)
5. Bunke, H., Varga, T.: Off-line roman cursive handwriting recognition. In: B.B. Chaudhuri (ed.) Digital Document Processing, Advances in Pattern Recognition, pp. 165–183. Springer Verlag London (2007)
6. Buonomano, D.V., Maass, W.: State-dependent computations: spatiotemporal processing in cortical networks. Nature Reviews Neuroscience **10**(2), 113–25 (2009). URL `http://www.ncbi.nlm.nih.gov/pubmed/19145235`
7. Buteneers, P., Verstraeten, D., van Mierlo, P., Wyckhuys, T., Stroobandt, D., Raedt, R., Hallez, H., Schrauwen, B.: Automatic detection of epileptic seizures on the intracranial electroencephalogram of rats using reservoir computing. Artificial Intelligence in Medicine **53**(3), 215–223 (2011)
8. Dominey, P.F.: Complex sensory-motor sequence learning based on recurrent state representation and reinforcement learning. Biological Cybernetics **73**, 265–274 (1995)
9. Dominey, P.F.: From sensorimotor sequence to grammatical construction: Evidence from simulation and neurophysiology. Adaptive Behaviour **13**(4), 347–361 (2005)
10. Dominey, P.F., Ramus, F.: Neural network processing of natural language: I. sensitivity to serial, temporal and abstract structure of language in the infant. Language and Cognitive Processes **15**(1), 87–127 (2000)
11. Doya, K.: Bifurcations in the learning of recurrent neural networks. In: Proceedings of IEEE International Symposium on Circuits and Systems 1992, vol. 6, pp. 2777–2780 (1992)
12. Fernando, C., Sojakka, S.: Pattern recognition in a bucket. In: Proceedings of the 7th European Conference on Advances in Artificial Life (ECAL 2003), *LNCS*, vol. 2801, pp. 588–597. Springer (2003)
13. Hermans, M., Schrauwen, B.: Recurrent kernel machines: Computing with infinite echo state networks. Neural Computation **24**(1), 104–133 (2012). DOI http://dx.doi.org/10.1162/NECO_a_00200
14. Hinaut, X., Dominey, P.F.: A three-layered model of primate prefrontal cortex encodes identity and abstract categorical structure of behavioral sequences. J. Physiology - Paris **105**(1-3), 16–24 (2011)
15. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Computation **9**(8), 1735–1780 (1997)
16. Ijspeert, A.J.: Central pattern generators for locomotion control in animals and robots: A review. Neural Networks **21**, 642–653 (2008)

17. Ilies, I., Jaeger, H., Kosuchinas, O., Rincon, M., Šakėnas, V., Vaškevičius, N.: Stepping forward through echoes of the past: forecasting with echo state networks (2007). URL http://www.neural-forecasting-competition.com/downloads/NN3/methods/27-NN3_Herbert_Jaeger_report.pdf. Short report on the winning entry to the NN3 financial forecasting competition
18. Jaeger, H.: A method for supervised teaching of a recurrent artificial neural network (2000). URL http://www.wipo.int/patentscope/search/en/WO2002031764. International Patent
19. Jaeger, H.: The "echo state" approach to analysing and training recurrent neural networks. Tech. Rep. GMD Report 148, German National Research Center for Information Technology (2001). URL http://www.faculty.jacobs-university.de/hjaeger/pubs/EchoStatesTechRep.pdf
20. Jaeger, H.: Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the echo state network approach. GMD Report 159, Fraunhofer Institute AIS (2002). URL http://minds.jacobs-university.de/pubs
21. Jaeger, H.: Echo state network. Scholarpedia **2**(9), 2330 (2007). URL http://www.scholarpedia.org/article/Echo_state_network
22. Jaeger, H., Haas, H.: Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless communication. Science pp. 78–80 (2004)
23. Jaeger, H., Lukoševičius, M., Popovici, D., Siewert, U.: Optimization and applications of echo state networks with leaky-integrator neurons. Neural Networks **20**(3), 335–352 (2007)
24. Jalalvand, A., Triefenbach, F., Verstraeten, D., Martens, J.P.: Connected digit recognition by means of reservoir computing. In: Proceedings of Interspeech 2011, pp. 1725–1728 (2011)
25. Kindermans, P.J., Buteneers, P., Verstraeten, D., Schrauwen, B.: An uncued brain-computer interface using reservoir computing. In: Workshop : machine learning for assistive technologies, Proceedings (2010)
26. Larger, L., Soriano, M.C., Brunner, D., Appeltant, L., Gutierrez, J.M., Pesquera, L., Mirasso, C.R., Fischer, I.: Photonic information processing beyond Turing: an optoelectronic implementation of reservoir computing. Optics Express **20**, 3241–3249 (2012). DOI http://dx.doi.org/10.1364/OE.20.003241
27. Legenstein, R., Chase, S.M., Schwartz, A.B., Maass, W.: A reward-modulated hebbian learning rule can explain experimentally observed network reorganization in a brain control task. J. of Neuroscience **30**(25), 8400–8410 (2010)
28. Li, J., Jaeger, H.: Minimal energy control of an ESN pattern generator. technical report 26, Jacobs University Bremen, School of Engineering and Science (2011)
29. Lukoševičius, M.: Reservoir computing and self-organized neural hierarchies. Ph.D. thesis, Jacobs University Bremen, Bremen, Germany (2011)
30. Lukoševičius, M., Jaeger, H.: Reservoir computing approaches to recurrent neural network training. Computer Science Review **3**(3), 127–149 (2009). DOI 10.1016/j.cosrev.2009.03.005
31. Lukoševičius, M., Popovici, D., Jaeger, H., Siewert, U.: Time warping invariant echo state networks. IUB Technical Report 2, International University Bremen (2006). URL http://minds.jacobs-university.de/pubs
32. Maass, W.: Motivation, theory, and applications of liquid state machines. In: B. Cooper, A. Sorbi (eds.) Computability in Context: Computation and Logic in the Real World, pp. 275–296. Imperial College Press (2011)
33. Maass, W., Joshi, P., Sontag, E.: Computational aspects of feedback in neural circuits. PLOS Computational Biology **3**(1), 1–20 (2007)
34. Maass, W., Natschläger, T., Markram, H.: Real-time computing without stable states: a new framework for neural computation based on perturbations. Neural Computation **14**(11), 2531–2560 (2002). DOI http://dx.doi.org/10.1162/089976602760407955
35. Martens, J., Sutskever, I.: Learning recurrent neural networks with Hessian-free optimization. In: Proc. 28th Int. Conf. on Machine Learning (2011). URL http://www.icml-2011.org/papers/532_icmlpaper.pdf
36. Paquot, Y., Duport, F., Smerieri, A., Dambre, J., Schrauwen, B., Haelterman, M., Massar, S.: Optoelectronic reservoir computing. CoRR **abs/1111.7219v1**, 1–39 (2011). URL http://arxiv.org/abs/1111.7219v1
37. Salmen, M., Plöger, P.: Echo state networks used for motor control. In: Proc. IEEE Int. Conf. on Robotics and Automation (ICRA), pp. 1953 – 1958 (2005)
38. Schiller, U.D., Steil, J.J.: Analyzing the weight dynamics of recurrent learning algorithms. Neurocomputing **63C**, 5–23 (2005)
39. Schrauwen, B., D'Haene, M., Verstraeten, D., Stroobandt, D.: Compact hardware liquid state machines on FPGA for real-time speech recognition. Neural Networks **21**(2-3), 511–523 (2008)
40. Schürmann, F., Meier, K., Schemmel, J.: Edge of chaos computation in mixed-mode VLSI - a hard liquid. In: Advances in Neural Information Processing Systems 17 (NIPS 2004), pp. 1201–1208. MIT Press, Cambridge, MA (2005)
41. Shi, Z., Han, M.: Support vector echo-state machine for chaotic time-series prediction. IEEE Transactions on Neural Networks **18**(2), 359–72 (2007)
42. Skowronski, M.D., Harris, J.G.: Automatic speech recognition using a predictive echo state network classifier. Neural Networks **20**(3), 414–423 (2007). DOI doi:10.1016/j.neunet.2007.04.006
43. Steil, J.J.: Backpropagation-decorrelation: recurrent learning with O(N) complexity. In: Proceedings of the IEEE International Joint Conference on Neural Networks, 2004 (IJCNN 2004), vol. 2, pp. 843–848 (2004)
44. Stieg, A.Z., Avizienis, A.V., Sillin, H.O., Martin-Olmos, C., Aono, M., Gimzewski, J.K.: Emergent criticality in complex turing b-type atomic switch networks. Advanced Materials **24**(2), 286293 (2012). DOI DOI:10.1002/adma.201103053
45. Sussillo, D., Abbott, L.F.: Generating coherent patterns of activity from chaotic neural networks. Neuron **63**(4), 544–557 (2009). DOI DOI:10.1016/j.neuron.2009.07.018
46. Triefenbach, F., Jalalvand, A., Schrauwen, B., Martens, J.P.: Phoneme recognition with large hierarchical reservoirs. In: Advances in Neural Information Processing Systems 23 (NIPS 2010), pp. 2307–2315. MIT Press, Cambridge, MA (2011)
47. Vandoorne, K., Dierckx, W., Schrauwen, B., Verstraeten, D., Baets, R., Bienstman, P., Campenhout, J.V.: Toward optical signal processing using photonic reservoir computing. Optics Express **16**(15), 11,182–11,192 (2008)
48. Verstraeten, D.: Reservoir computing: Computation with dynamical systems. Phd thesis, Electronics and Information Systems, University of Ghent (2009). URL http://organic.elis.ugent.be/biblio

49. Verstraeten, D., Schrauwen, B., D'Haene, M., Stroobandt, D.: An experimental unification of reservoir computing methods. Neural Networks **20**(3), 391–403 (2007)
50. Verstraeten, D., Schrauwen, B., Stroobandt, D.: Reservoir-based techniques for speech recognition. In: Proceedings of the IEEE International Joint Conference on Neural Networks, 2006 (IJCNN 2006), pp. 1050 – 1053 (2006)
51. Verstraeten, D., Schrauwen, B., Stroobandt, D., Van Campenhout, J.: Isolated word recognition with the liquid state machine: a case study. Information Processing Letters **95**(6), 521–528 (2005)
52. Werbos, P.J.: Backpropagation through time: what it does and how to do it. Proceedings of the IEEE **78**(10), 1550–1560 (1990)
53. Williams, R.J., Zipser, D.: A learning algorithm for continually running fully recurrent neural networks. Neural Computation **1**, 270–280 (1989)

Mantas Lukoševičius received his BSc of Computer Science in Kaunas University of Technology, Lithuania; MSc and PhD in Jacobs University Bremen, Germany, where he continues as a postdoc in Herbert Jaeger's research group. He has also spent an inspiring summer in Yoshua Bengio's Machine Learning lab in University of Montreal, Canada. His current research interests involve Machine Learning and Recurrent Neural Networks.

Herbert Jaeger studied mathematics in Freiburg, did his PhD in Computer Science in the AI group of Ipke Wachsmuth at Bielefeld University, spent an illuminating summer at Luc Steels' VUB AI Lab in Brussels, was transformed to a roboticist/machine learner during a 5-year postdoc period in Thomas Christaller's AI/robotics institute at the GMD (subsequently Fraunhofer IAIS) in Birlinghoven, then headed the Intelligent Dynamical Systems unit in the same institute, and since 2002 is Associate Professor for Computational Science at Jacobs University Bremen. Since his freshman days he tries to develop mathematical languages to describe learning dynamical systems.

Benjamin Schrauwen received a M.Sc. degree in computer science at the University of Antwerp in 2000; and a M.Sc. and Ph.D. in computer engineering from the Ghent University in 2002 and 2008, respectively. Since 2010, he is Professor at Ghent University, affiliated with the Department of Electronics and Information Systems (ELIS) in the Faculty of Engineering. His research interests include machine learning and biologically inspired computing systems. He currently leads a research group of about 15 people with interests in reservoir computing, spiking neural networks, and deep learning architectures, with applications in the domains of speech recognition, robotics and biomedical signal processing.