Echo State Network Reservoir Shaping and Information Dynamics at the Edge of Chaos

(エコーステートネットワークの状態設計とカオスの縁における情報ダイナミクス)

A dissertation submitted to the Department of Adaptive Machine Systems
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in Engineering
at the
OSAKA UNIVERSITY

Joschka Boedecker

February 2011

Thesis Supervisor:	Minoru Asada
Title:	Department of Adaptive Machine Systems,
	Graduate School of Engineering, Osaka University
Thesis Committee:	Minoru Asada, Chair
	Koh Hosoda
	Hiroshi Ishiguro

Abstract

This thesis investigates ways to optimize the hidden layer (reservoir) of a specific type of recurrent neural network, called Echo State Network (ESN), in order to improve performance reliably on a variety of tasks. First, an initialization strategy based on permutation matrices for the hidden layer connectivity is tested. This approach is shown to vastly improve network performance on tasks which require long memory and is also able to perform highly nonlinear mappings. Second, an unsupervised, local learning rule is derived that aims at a high entropy of reservoir codes while keeping neuron activity sparse. This learning rule extends previous approaches which change the intrinsic plasticity (IP) of reservoir neurons using a gain and a bias factor in the neurons' transfer function. A moderate improvement over random networks is shown, and a limitation of the IP approach with standard sigmoidal activation functions is identified. Furthermore, a specific dynamics regime located between stable and chaotic dynamics is studied. Networks whose dynamics operate in this region have been shown to exhibit greatly increased computational capabilities. The reasons for this phenomenon are, however, not fully understood. In this thesis, an information-theoretic framework is adopted to measure the components of universal computation of ESN reservoirs as the networks undergo the phase transition from stable to chaotic dynamics. By measuring these components directly and on a local level, we gain novel insights over existing work. We show that both information transfer and information storage are maximized at the phase transition point. Moreover, we discuss implications of these results with respect to different task requirements, and possibilities for reservoir optimization.

Acknowledgments

This thesis would not have been possible without the support of many mentors, colleagues, friends, and my family. I wish to express my gratitude to all of them.

I want to thank my supervisor professor Minoru Asada for his constant support throughout these many years in his lab, for his guidance, his patience, and for letting me work on a topic which I was deeply interested in. Although his schedule is always filled with appointments, I appreciate that he never failed to find time to meet with me and discuss my work.

I am thankful to professors Hiroshi Ishiguro and Koh Hosoda for serving as members of my thesis committee, and for their valuable feedback in many seminars, talks, and discussions. Valuable feedback was also given by professors Yuichiro Yoshikawa, Yukie Nagai, Tomomichi Sugihara, Yasutake Takahashi, Takashi Takuma, and Masaki Ogino. Professor Ogino also was absolutely selfless in his help with my efforts to get financial support in form of a JSPS scholarship before starting the PhD course. Without his help, spending a whole night on more than one occasion to translate (and undoubtedly improve!) my application documents, I would certainly not have succeeded. I am very thankful for all his help in this and many other matters. Dr. Hidenobu Sumioka helped tirelessly with all the paperwork connected to the scholarship, translating many documents and making sure I would not miss important deadlines. I also thank Ms Shizu Okada and Mrs Yayoi Imahashi for help in many scholarship related questions, as well as Mrs Mizuki Kawashima, Mrs Yoshiko Ogura, and Ms Hiroko Takaoka for scheduling and re-scheduling my meetings with professor Asada.

Professor Norbert Michael Mayer introduced me to the fascinating world of Reservoir Computing and Echo State Networks, and I learned a lot from working with him. Very special thanks go to Dr. Oliver Obst for long years of friendship, great collaboration, guidance, moral support, and much more. I am sure I could not have finished this PhD without his help on so many levels. I'd also like to thank many friends and colleagues in the lab: my long-time partner in many projects Dr. Rodrigo da Silva Guerra, Dale Thomas, Richard Keely, Mai Nguyen, Jacob Pyke, Beata Joanna Grzyb, Dr. Hidenobu Sumioka, Dr. Shuhei Ikemoto, Toshihiko Shimizu, Fabio Dalla Libera, Dr. Tomoyuki Noda, Dr. Sawa Fuke, Dr. Shinya Takamuku, Kenichi Narioka, Hishashi Ishihara, Tsukasa Nakano, all the exchange students from Koblenz University, especially Dennis Fassbender, Andreas Held, Benedikt Schmidt, and Henrik Simon, and Tobias Hoenisch. They all have made this long journey enjoyable with excellent friendship, great teamwork, much needed distractions, discussions beyond research, and much more. In fact, all members of the Asada lab and the ERATO Asada project have surely supported me in one way or another, and I'd like to thank them. Professor Uli Furbach in Koblenz helped me get around the obstacles in the final phase of my Master's thesis. He also provided a great environment for learning, exploration, and research with the AGKI. Many thanks go to Dr. Mikhail Prokopenko, head of the Adaptive Systems group at CSIRO's ICT Centre in Sydney, Australia, for hosting me twice for several weeks. I had an enormously productive time on both occasions, and a lot of fun in the process. I thank all members of the group for their warm welcome and a great research environment. Special thanks also to Dr. Joseph Lizier for excellent collaboration and great discussions. During my stays in Sydney, Dr. Oliver Obst and Annemarie Dosen provided a home away from home, letting me stay with them for weeks at a time. Heaps of thanks go out to them, I had an amazing time!

Last but certainly not least, I would like to thank my dear wife Mie for all her support, her help, and her patience throughout these many years. I am very glad we shared this journey, and I am looking forward to lots more to come. I am very grateful also for the support by all members of my family, both in Japan, and in Germany. Finally, I express my gratitude for financial support from the Japan Society for the Promotion of Science in form of a fellowship for young researchers.

February, 2011 Joschka Boedecker

List of Figures

1 (a) Feedforward neural network: information flows only forward (b) Example of a recurrent neural network architecture: due to the loops in the network connectivity, the information can "cycle" around the network and is available for computation over some period of time. This provides the network with a temporal context. (c) Reservoir network: a specific type of recurrent neural network which has recurrent connections in the hidden layer, implementing a *fading memory* of past inputs. Only output connections are adapted through training in this model (see main text for more information). 2

2 (left) A user engaging in imitative interaction with a humanoid robot [1] (right) The recurrent neural network with parametric bias (RNNPB) which was used in the same study as a mirror system to imitate demonstrated movements. 2

3 Echo State Network used to learn whole-body kinematics of the humanoid robot ASIMO (from [2]). 3

4 The left image shows the setup of an Echo State Network which was used to learn forward and inverse kinematics of the Mitsubishi PA10 robot arm (right image) simultaneously. The network transients were used to implement a task space controller which achieved excellent generalization to untrained locations (from [3]) 4

5 The research landscape related to topics in the thesis. 7

6 Memory capacity of recurrent neural networks of threshold units versus mean input level $\bar{u}$ and variance of recurrent weight strength σ^2. The network was trained on a 3-bit parity task with each node having an in-degree (number of incoming connections) of $K = 4$. The best performance (indicated by the dark areas in the plot) can be found around the critical line (from [4]). 19

7 Average performance of quantized ESNs for different quantization levels m depending on the in-degree K and standard deviation σ of the reservoir weights in three different task settings. The phase transition is indicated by the black dashed line. Again, highest performance (dark red colors) are observed at around the phase transition line (from [5]). 21

8 Relation of Chapters 3, 4, and 5 to work presented in the literature survey of this chapter. 22

9 Setup of the ESN used in the MC task experiments. The difference in the other experiments was that only one output node was used. . . . 24

10 The time-series for the three benchmarks used to evaluate different aspects of networks performance: (a) Uniform random input, (b) Output of the NARMA 30th order system, (c) Mackey-Glass attractor with $\tau = 17$. 26

11 Homeostatic regulation based on intrinsic plasticity: the incoming signal drives the network while a gain and a bias parameter in the transfer function (a and b in the logistic function above) are adapted to achieve the desired output distribution of neuron activity values. 28

12 (left) Visualization of a 50×50 permutation matrix with black squares indicating a zero, and white squares indicating a value of one; (right) rings of (randomly) different sizes are the result of the connectivity defined by a permutation matrix. The rings here correspond to the the matrix on the right. 31

13 Average memory capacity (graph a) and normalized root mean squared error (NRMSE) for the NARMA modeling and the Mackey-Glass prediction tasks (graphs b and c, respectively) in the four different conditions, averaged over 50 simulation runs with standard dev. in parenthesis. Note that the errors for Mackey-Glass are scaled by a factor of 10^{-4}. For the memory capacity (a), higher bars are better, while lower bars in the errors are better for the prediction tasks (b and c). 32

14 The $\mathrm{MC_k}$ curves for the uniform random input data. The plot indicates how well the input signal can be reconstructed ($\mathrm{MC_k}$) for increasing delay times k 33

15 Plot of the reservoir matrix eigenvalues in the complex plane for a 100 node network (a) in the RND condition, (b) for a PMT matrix, and (c) in the after training with IPLAP. All matrices have been scaled to have a spectral radius of 0.95 35

16 Result of IP learning with a single fermi neuron with and without self-recurrence and a learning rule for the exponential output distribution. IP successfully produces the desired output distribution. 36

17 (a) Effect of IP learning on a single tanh neuron without self-recurrence, trained with a learning rule for a Gaussian output distribution. IP learning fails to achieve the desired result: the best it can do is to drive the neuron to a uniform output distribution, which has the smallest distance (for the given transfer function) to the desired distributions; (b) same as (a), but with self-recurrence added to the neuron. The achieved output distribution is significantly more Gaussian-shaped than without the self-recurrence. The effect is amplified in a network where the neurons receive additional inputs with similar distributions. All units were trained using 100000 training steps and uniformly distributed input data on the interval $[-1, 1]$. 37

18 Change in the hidden neuron activations as the network dynamics go from the ordered (left panel) to the chaotic regime (right panel). The dynamics in the phase-transition region are shown in the middle (from [5]). 40

19 Information storage of a node is the amount of information in its past x_n^k that is relevant to predicting its future x_{n+1}. 41

20 Information transfer between a source node Y and a destination node X is defined as the information y_n provided by the source about the destination's next state x_{n+1} that was not contained in the past x_n^k of the destination. 42

21 Numerical estimation of the largest Lyapunov exponent λ. Trajectories are kept close by resetting the distance to γ_0 after each update step in order to avoid numerical overflows (illustration after [6]). See text for more details. 44

22 Testing different initial separations in the LE calculations. 46

23 Testing different lengths for the period to wash out transient effects. 47

24 Testing different combinations of history length and kernel radii. (top) $k = 1$ and $r = 0.252$ (middle) $k = 2$ and $r = 0.423$ (bottom) $k = 3$ and $r = 0.577$ 49

25 (top) Memory capacity vs estimated Lyapunov exponent (bottom) Normalized root mean squared error (NRMSE) vs estimated Lyapunov exponent 51

26 (top) Average transfer entropy in the reservoir for the memory capacity task vs estimated Lyapunov exponent (bottom) Average transfer entropy in the reservoir for the NARMA task vs estimated Lyapunov exponent 52

27 (top) Average active information storage in the reservoir for the memory capacity task vs estimated Lyapunov exponent (bottom) Average active information storage in the reservoir for the NARMA task vs estimated Lyapunov exponent 53

28 (left) mean squared errors of the prediction over the test data for different coupling strengths and fixed $\omega = 0$. (right) mean squared error for different ω using a fixed coupling of $e = 0.75$. Reported results are averages over 50 runs. 59

29 Uniform input on the interval $[-1; 1]$ and a $\tanh(\cdot)$ transfer function lead to the output distribution in the histogram. IP selects a slice of this distribution, as illustrated by the vertical lines. Adapting gain and bias changes width and position of the slice. 64

30 Average information dynamics in RBNs in relation to average connectivity $\bar{K}$ (from [7].) 66

31 Architecture of an echo state network. In echo state networks, usually only the connections represented by the dashed lines are trained, all other connections are setup randomly and remain fixed. The recurrent layer is also called a *reservoir*, analogously to a liquid, which has fading memory properties. As an example, consider throwing a rock into a pond; the ripples caused by the rock will persist for a certain amount of time and thus information about the event can be extracted from the liquid as long as it has not returned to its single attractor state — the flat surface. 77

List of Tables

1 Settings of IP parameters σ (for IPGAUSS) and c (for IPLAP) for the different benchmark tasks. These settings were determined empirically through systematic search of the parameter space for the optimal values. 25

2 Results of the experiments (averaged over 50 simulation runs) for the three benchmarks in four conditions in tabular form (cf. Fig. 13). Errors for Mackey-Glass are scaled by a factor of 10^{-4}. 31

Contents

Abstract **iii**

Acknowledgments **v**

1 Introduction **1**

1.1 Motivation 5

1.2 Problem Statement and Thesis Objectives 6

1.3 Structure Of The Thesis 6

2 Related work **11**

2.1 Reservoir Assessment 11

2.2 Reservoir Optimization 12

2.2.1 Generic Methods 13

2.2.2 Unsupervised Methods 15

2.3 Dynamical Systems at the Edge of Chaos 17

2.4 Reservoir Computing at the Edge of Chaos 18

3 Initialization and self-organized optimization of recurrent neural network connectivity **23**

3.1 Network Setup and Benchmark Description 24

3.1.1 Short Term Memory Capacity 26

3.1.2 NARMA 30th Order System 27

3.1.3 Mackey-Glass Time-Series 27

3.2 IP Learning and a Rule for a Laplace Output Distribution 27

3.3 Reservoirs Based on Permutation Matrices 30

3.4 Results 30
3.5 IP Revisited 34

4 Information processing in Echo State networks at the edge of chaos 39
4.1 Information-Theoretical Measures 40
4.2 Estimating the Criticality of an Input-Driven ESN 42
4.3 Results 44
4.3.1 Effect of the Initial Separation Size on Numerical Estimation of the Largest Lyapunov Exponent 45
4.3.2 Different Transient Times for the Lyapunov Exponent Calculation 46
4.3.3 Tradeoff of Kernel Size versus History Length in the Transfer Entropy Calculations 47
4.3.4 Transfer and Memory are Maximized at the Edge of Chaos . . 48

5 Improving Recurrent Neural Network Performance Using Transfer Entropy 55
5.1 Network Dynamics and Output-Weight Training Procedure 56
5.2 Adaptation of Information Transfer 57
5.3 Experimental Results 58
5.3.1 Prediction of Autoregressive Coupled Processes 58
5.3.2 Prediction of Mackey-Glass Time Series 59

6 Discussion 61
6.1 Permutation Matrices as a Computationally Inexpensive Initialization Method 62
6.2 Limitations of IP Learning for Standard Sigmoid Neurons 62
6.3 Reservoirs and Intrinsic Computational Capabilities 64
6.4 Relation to Information Dynamics in Random Boolean Networks . . . 65
6.5 Edge of Chaos Is Not Universally Beneficial for Computation 66
6.6 No one-to-one Relation Between Intrinsic Computation and Task Performance 67
6.7 Information Transfer Between Input and Desired Output Can Be Used To Successfully Guide Reservoir Adaptation 68

7 Conclusion and future work **69**
7.1 Summary and Contributions 69
7.2 Directions for future work 71
7.2.1 Reservoir Optimization 71
7.2.2 Reservoir Assessment 72
7.2.3 Reservoir Computing at the Edge of Chaos 73

A Reservoir Computing **75**

B Review of Relevant Information-theoretic Quantities **79**

Bibliography **81**

Chapter 1

Introduction

Recurrent neural networks (RNNs, see Fig. 1 (b,c)) are powerful tools for nonlinear systems modeling and temporal sequence learning. They are computationally more powerful than simple feedforward architectures (see Fig. 1 (a)) since they can exploit long-term dependencies in the input when calculating the output. This means they are able to implement *dynamical systems* which have a state, as opposed to the functional mappings which are implemented by feedforward neural networks (FFNN). In addition, they are more biologically plausible, making use of feedback connections which are abundant in neural circuits (e.g. in the human neocortex [8, 9]).

RNNs have been used for many important applications in engineering, such as increasing the fuel efficiency of the widely popular Toyota Prius [10] or optimizing control of industrial and technical systems [11, 12]. They have also been used successfully for learning and controlling robot behaviors in various studies. In [1], for instance, the authors used a recurrent neural network with parametric bias (RNNPB) [13] to enable the robot to learn movement patterns from a human demonstrator. The network acted as a mirror system which, after training, was able to regenerate movement patterns for the robot in synchrony with the human demonstrator, as well as interesting responses to novel patterns (see Fig. 2). Rolf et al. [2] presented a recurrent neural network of the reservoir computing type (see below) which is able to learn whole-body kinematics of a humanoid robot (see Fig. 3). The network was trained by the backpropagation-decorrelation (BPDC) algorithm [14], while the hidden layer

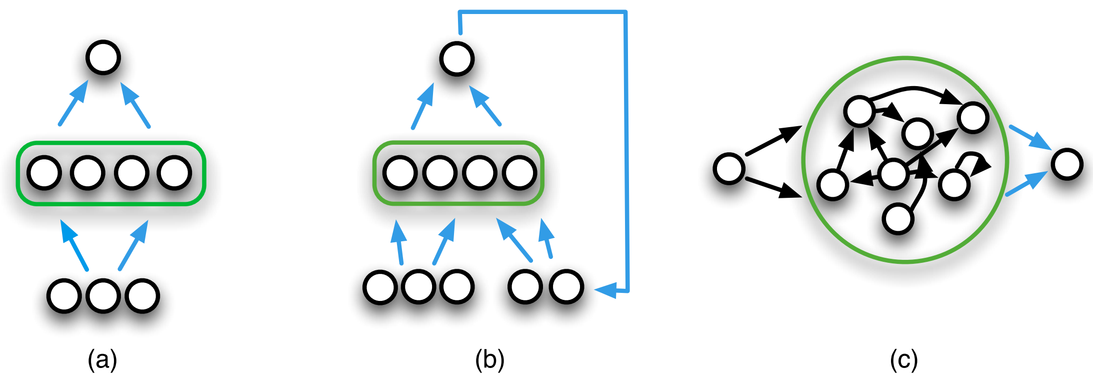


Figure 1: (a) Feedforward neural network: information flows only forward (b) Example of a recurrent neural network architecture: due to the loops in the network connectivity, the information can "cycle" around the network and is available for computation over some period of time. This provides the network with a temporal context. (c) Reservoir network: a specific type of recurrent neural network which has recurrent connections in the hidden layer, implementing a *fading memory* of past inputs. Only output connections are adapted through training in this model (see main text for more information).

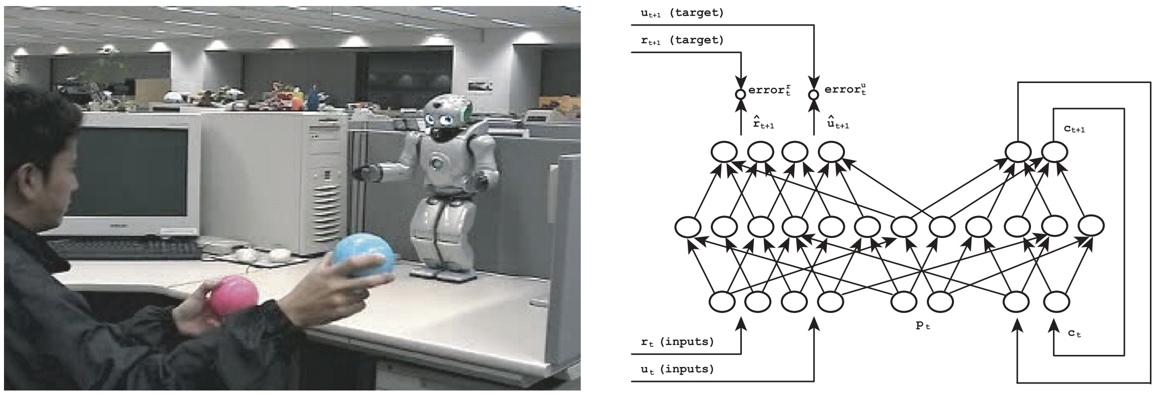


Figure 2: (left) A user engaging in imitative interaction with a humanoid robot [1] (right) The recurrent neural network with parametric bias (RNNPB) which was used in the same study as a mirror system to imitate demonstrated movements.

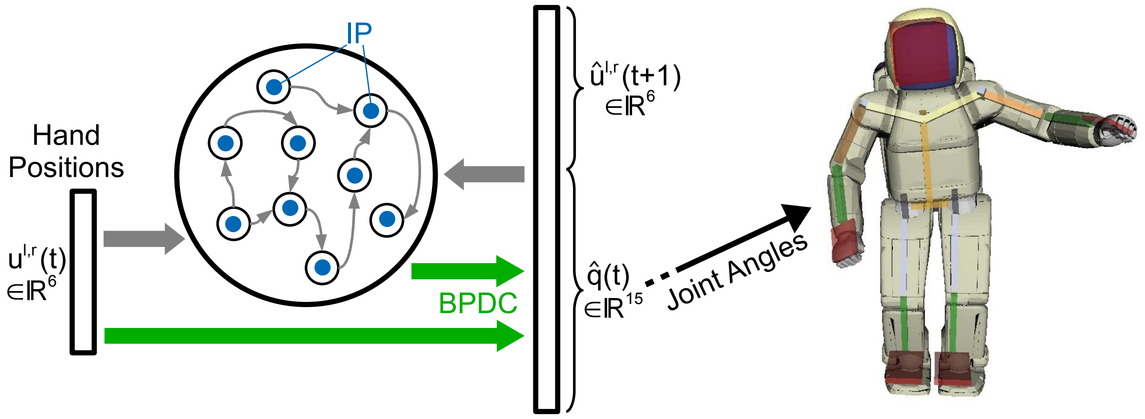


Figure 3: Echo State Network used to learn whole-body kinematics of the humanoid robot ASIMO (from [2]).

was adapted by the intrinsic plasticity (IP) learning rule [15, 16] which will be presented in detail in later chapters of this thesis. The training procedure proved very efficient and generalization of the network to new movement patterns was shown to be very good. Reinhart and Steil [3] introduced a control framework using a recurrent neural network for goal-directed movement generation. The network is able to learn forward and inverse kinematics of a robot arm simultaneously (see Fig. 4), and the network dynamics are exploited to implement a nonlinear task space controller. The approach allows for efficient online learning and execution, as well excellent generalization of the learned kinematic model. Several other studies in robotics exist (including [17, 18, 19, 20, 21, 22, 23, 24], proving the usefulness of recurrent neural networks for learning (and generating) the nonlinear mappings and temporally extended ordered sequences typical for this domain.

Analysis and training of RNNs is rather involved when compared with methods for the simpler FFNNs. Early training algorithms like Backpropagation Through Time [25] (BPTT) or Real-Time Recurrent Learning [26] (RTRL) suffered from high computational complexity and slow convergence [27, 28, 29]. In the last few years, however, RNNs have been increasingly popular—mainly due to new, efficient architectures and training methods published in recent years (surveyed in [30]). These

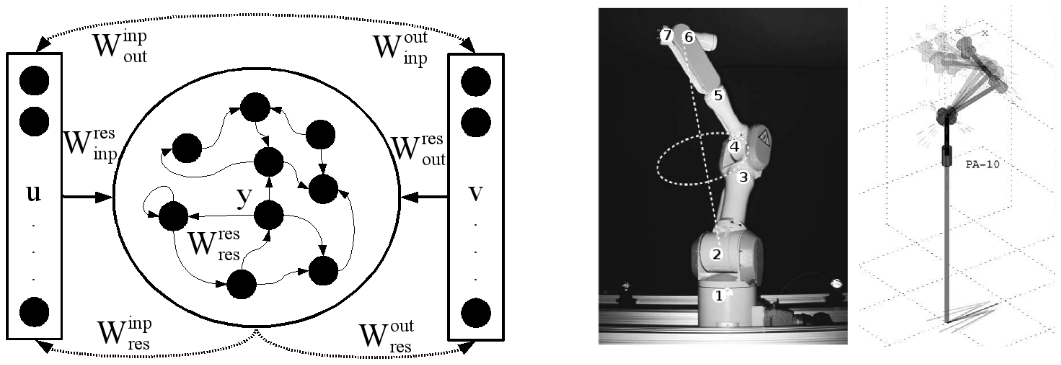


Figure 4: The left image shows the setup of an Echo State Network which was used to learn forward and inverse kinematics of the Mitsubishi PA10 robot arm (right image) simultaneously. The network transients were used to implement a task space controller which achieved excellent generalization to untrained locations (from [3])

include, for instance, self-organizing maps for time series [31], long short-term memory (LSTM) networks [32], Evolino [33, 34], and also the Reservoir Computing (RC) approach [35, 36, 37].

Reservoir Computing is a new paradigm in recurrent neural network training based on two main insights:

1. Instead of training every connection of a recurrent neural network, it suffices to initialize the network with random weights and only train the linear output connections. The fixed, randomly initialized parts of the network will then act as a nonlinear filter of the inputs to the network.

2. The hidden layer (also called *reservoir* or *liquid*) should have fading memory properties in order to guarantee stability of the network.

These insights were developed independently by the groups of Herbert Jaeger who used it for his Echo State Networks (ESNs) [35], and Wolfgang Maass who presented the Liquid State Machine (LSM) [37]. More details on the RC approach are given in Appendix A.

While Reservoir Computing networks successfully overcome the problems that complicated training and application of more traditional RNNs, several new open research questions became apparent. Two particular questions of these are addressed in this thesis. Below, we describe the motivation behind these questions, and give a precise statement of which problems are investigated.

1.1 Motivation

A problem of the RC approach is that there is considerable variation in performance when different random reservoir initializations are used with all other network parameters remaining fixed [38]. The approach essentially trades off reduced training complexity (only training the output connections) with optimal coding within the reservoir for a given problem. It is obvious that random initialization will lead to sub-optimal performance when compared with matrices which have been optimized for the problem at hand.

One possibility is to find reservoir initialization strategies that will, on average, reliably lead to better results than random initialization for a large class of problems. Another promising approach that avoids the shortcomings of earlier training algorithms is to train the reservoir and the output connections *separately*, i.e., with different learning algorithms. Both forms of optimization change or *shape* the dynamics of the reservoir with the goal of improving overall performance at one, or possibly several tasks.

Apart from the question which initialization or training procedure is most suitable in order to achieve good performance on a given task, a further important open research question is to explain the phenomenon that performance of reservoir computing networks seems to be maximized at the border between stable and unstable dynamics regimes. This kind of phenomenon has been observed in other dynamical systems [39, 40, 41] and there have been previous studies in the context of RNNs [4, 42, 5], but no universally accepted explanation has been found so far. A detailed account of which elements of computation enable this performance maximization is important not only for Reservoir Computing itself, but also in the wider context of understanding general distributed computation in complex systems.

1.2 Problem Statement and Thesis Objectives

From this motivation, we summarize the main problem addressed in this thesis in the following question:

How can Echo State Network reservoirs be reliably improved over standard initialization, and how can we assess their quality?

More specifically, we break this down into two sub-questions which will be addressed in the main part of the thesis:

- How can reservoirs be shaped (using alternative initialization and unsupervised pre-training) to improve task performance reliably over standard reservoirs?
- How to quantify the elements of intrinsic computation in ESN reservoirs, and how do these change at the phase transition between ordered and chaotic dynamics (the so-called *edge of chaos*)?

The first of these questions is addressed in Chapter 3 and 5, and aims to make a contribution to the blue region in Fig. 5. The second question is dealt with in Chapter 4. It also aims to contribute to the wider problem of designing high-performing reservoirs, but is meant to additionally contribute to the green colored region in Fig. 5 which includes the question how computation in reservoirs changes at the order-chaos phase transition.

1.3 Structure Of The Thesis

The paragraphs below briefly describe the contents of each chapter in this thesis.

Chapter 2

In Chapter 2, an overview of related work is given. This includes relevant work in reservoir assessment, reservoir optimization, dynamical systems at the edge of chaos, and investigation of computational capabilities of reservoirs at the edge of chaos.

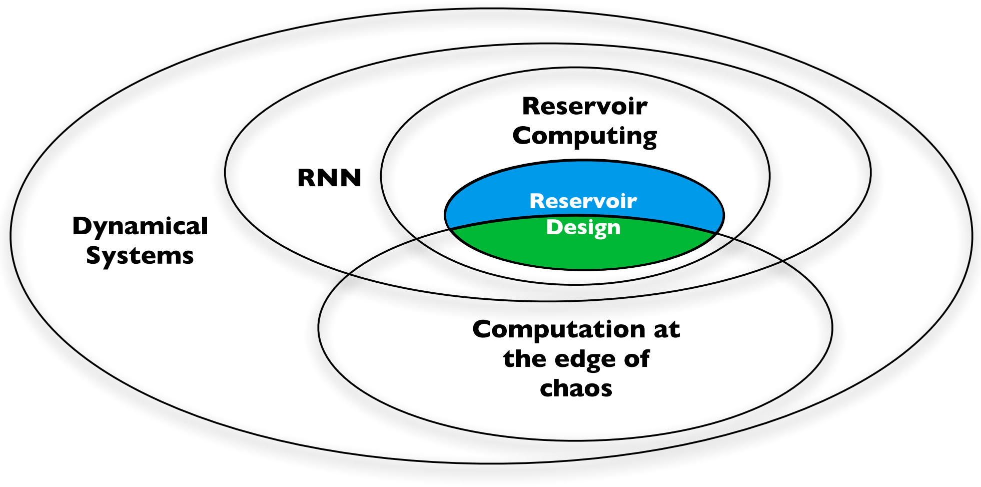


Figure 5: The research landscape related to topics in the thesis.

Chapter 3

Chapter 3 presents our work on initialization and self-organized optimization of recurrent neural network connectivity. We derive a new local learning rule which changes internal parameters of each reservoir neuron in order to achieve an output distribution which strikes a balance between high entropy and sparse activity in the reservoir. This approach of changing a neuron's internal parameters is called *intrinsic plasticity* (IP). We compare this new learning rule with a similar IP rule found in the literature, with standard reservoirs using uniform random connectivity (as a baseline), and with reservoirs based on permutation matrices. Permutation matrices had been found to be very useful in a specific benchmark test for chaotic time series prediction, but had not been tested widely otherwise. They were included in our comparison in order to investigate their performance in benchmarks with different requirements. Our results showed that permutation matrices can vastly outperform the other methods if long memory is required for the task. Additionally, they performed very well in a non-linear system modeling task which also required a window of past inputs in order to compute the output. The other methods did better in a chaotic time series prediction task, however. Analysis of the output distributions of both IP methods revealed a limitation of this approach in the reservoir context with standard sigmoidal units.

Chapter 4

Information processing of Echo State Networks at the edge of chaos is the topic of Chapter 4. We use the framework of information dynamics [7] to quantify the elements of intrinsic computation, namely memory, communication, and processing[1]. We use the short-term memory capacity and the nonlinear system modeling benchmark from Chapter 3 and test networks whose weights are gradually increased. To assess whether the networks are in the ordered or the chaotic dynamical regime, or just in between the two at the phase-transition region (edge of chaos, also called *critical point*), we measure the Lyapunov characteristic exponent. We show, in line with a conjecture from previous work in dynamical systems, that the intrinsic computational capabilities of the tested networks are indeed maximized at the phase-transition between ordered and chaotic dynamics. To our knowledge, this is the first study to present direct quantitative evidence for this conjecture at the level of individual neurons.

Chapter 5

An approach to improve recurrent neural network connectivity based on transfer entropy (as discussed in Chapter 4) is presented in Chapter 5. We show how the information transfer between input and desired output of a system can be interpreted as the learning goal of the task, which is then used to adapt local parameters governing memory length in each reservoir node. The approach is tested on synthetic data in two different benchmarks and is able to improve the network performance reliably for online and offline output training methods.

Chapter 6

In Chapter 6, we discuss the results presented in Chapters 3, 4, and 5. In particular, we explore the limitation of IP in the reservoir context. We also discuss the relationship between task performance and reservoir dynamics at the edge of chaos, and

[1]We concentrate on memory and communication of reservoir nodes in this study; processing will be addressed in future work.

the correspondence between observed maximization of computational capabilities (in terms of memory and information transfer) and task performance.

Chapter 7

In Chapter 7 we summarize our contributions to the field of Reservoir Computing and related areas (see Fig. 5). Directions for future work are pointed out, and concluding remarks are given.

Appendix A

This Appendix gives a very short introduction to Reservoir Computing, emphasizing the Echo State Network model in particular. It presents a formal description of Echo State Networks and the Echo State Property.

Appendix B

The final Appendix presents a brief review of basic quantities from information theory. We only give an overview of concepts that are relevant to the description of active information storage and transfer entropy as used in Chapters 4 and 5.

Before the main results of this thesis are presented, we will survey related work that forms the basis for our own studies in the next chapter.

Chapter 2

Related work

In this chapter, we will review work that is closely related to the contents of the following chapters. We start out with an overview of different measures for assessing reservoir quality. After that, reservoir optimization techniques are surveyed, concentrating on generic and unsupervised methods. These are most related to the results presented in Chapters 3 and 5. Our review of these topics will closely follow the presentation in [43].

We continue with a look at existing literature on the theme of dynamical systems at the edge of chaos. We will first present some of the classical research on cellular automata (CA) and random boolean networks (RBN) which has led to the conjecture that support for intrinsic computation might be maximized in dynamical systems at the border between stable and chaotic dynamics—the edge of chaos. We then take a more focused look at reservoir networks poised at this region, and the consequences for task performance, which are also the topics studied in Chapter 4.

2.1 Reservoir Assessment

Many different measures for the quality of a reservoir, i.e., its expected usefulness for computation in different tasks, have been proposed in the literature. An often used criterion for reservoir quality assessment is its short term *memory capacity* (MC). It was introduced in [44] and measures the degree to which a reservoir can reconstruct an input from time step n at step $n + k$ for different delays k (see Section 3.1 for a formal definition). We use it as an evaluation criterion in the studies in Chapters 3 and 4.

A desirable property of reservoirs is for the cross-correlation matrix of their unit activations to have a small *eigenvalue spread* (EVS). A small value for the EVS is necessary in order to use efficient online learning algorithms like stochastic gradient descent. In standard ESNs, it was reported that the EVS can reach values as high as 10^{12} which prohibits use of these learning techniques [45]. Attempts to achieve low correlation of reservoir activations themselves have been reported in [45].

Several researchers have measured, and tried to increase, the entropy in the reservoir, either for the reservoir overall [45, 38] or for outputs of individual units in the reservoir [16, 46, 47]. In [48], the entropy of individual units was increased in connection with other plasticity mechanism, and the entropy of the reservoir as a whole was measured in the analysis (see also below).

The Lyapunov (characteristic) exponent (LE) is a measure which allows to assess the criticality of a dynamical system, i.e., whether its dynamics are in the stable regime, in the chaotic regime, or in between, in the phase transition (or critical) region. It has been used in the RC context for instance in [42] and in [49]. The measure is interesting because of evidence that dynamical systems show increased computational performance right at the order-chaos phase transition (more on this below in Section 2.3). We used the LE in our investigation of the information dynamics in ESN at the critical point in Chapter 4, and it will be defined formally there. Several other measures for the quality of reservoirs using binary units are described below in Section 2.4.

2.2 Reservoir Optimization

For our overview on reservoir optimization techniques, we adopt the classification used in [43]:

- *Generic* guidelines in order to arrive at good reservoirs which don't take the task into account (they make no use of either the input statistics or an output error)

- *Unsupervised* reservoir pre-training based on the input statistics. The output is not considered.

- *Supervised* reservoir pre-training using both input and output data.

As mentioned above, however, we will only consider the generic and unsupervised methods here since they are the most relevant for the work in subsequent chapters. For a more complete overview, please refer to [43]. In that overview, the authors also stress a point we would like to reiterate here: the fact that no reservoir will be optimal for all conceivable problems. This is due to the well-known "no free lunch" theorem of supervised learning due to Wolpert [50], stating that no bias can exist which will improve accuracy of a model with respect to *all* types of problems. Nevertheless, reservoirs can be improved for one or possibly several similar tasks at hand. In the following, we consider generic methods for reservoir improvement.

2.2.1 Generic Methods

Generic methods for producing reservoirs do not take into account the input or output data. They are general recommendations on how to set up reservoirs that can be expected to perform well in practice. Also included are tuning certain global reservoir parameters by hand for a given task, and initialization of the reservoir topology with weights that are not drawn from a uniform random probability distribution.

Early recommendations on how to create good reservoirs were given in Jaeger's initial publications [35, 51] on the echo state approach. He recommended to use a large number of neurons for the reservoir, and to use a sparse random connectivity between reservoir neurons. This should produce reservoirs with a rich dynamics where neurons are only loosely coupled, and produce a large and varied set of responses to stimulation which could then be used by the readout neurons. Any reservoir would naturally have to be designed to have the *Echo State Property* which, in simple terms, means that it implements a fading memory (refer to Appendix A for a formal definition).

Different kinds of topologies have been tested for the reservoir connectivity. In [52] small-world networks [53], scale-free networks [54], and topologies created by growing networks according to biologically inspired principles were tested. The networks were evaluated according to the overall normalized root mean squared error (NRMSE) and the eigenvalue spread of the reservoir unit activation cross-correlation matrix. No significant difference, however, was found for any of the tested reservoirs compared to the standard random networks.

Hajnal and Lőrincz [55] suggested permutation matrices for the connectivity of the reservoir. In order to create a permutation matrix, one takes an identity matrix and randomly shuffles its columns. This results in a very sparse connectivity with one neuron connecting to only one other neuron. In addition, the reservoir matrix will have full rank which, the authors noticed, correlates with higher performance. They tested this topology on a Mackey-Glass time series prediction task and reported improved performance over random reservoirs. We took up this idea and tested it on two additional tasks with differing requirements in the experiments in Chapter 3.

In [56], an ESN with spatial organization in the reservoir was proposed under the name SODESN (spatially organized distributed echo state network). Connectivity in this topology was restricted to connections with immediate neighbors. An interesting point in this work was that the ESN nodes themselves were distributed over a grid of sensor network nodes, and the SODESN was used to learn a model of each node's response to input data and data from its neighbors. This model could then be used for fault detection and predicted data could substituted for a node with faulty measurements. The particular topology was well suited for a sensor network with limited communication ability between nodes. It would be interesting to compare it to other existing ESN topologies.

Another generic method is to divide the reservoir into different modules. Such an approach was implemented in [57] which used lateral inhibition between the reservoir modules to achieve a decoupling of their signals. This enabled learning of a task which is known to be notoriously difficult for standard ESNs, namely the prediction of a signal constructed by superimposing different oscillators (see also [58]).

The Evolino approach [33, 59, 34] uses a reservoir which consists of modules specially designed to hold signals in memory for a very long time (cf. LSTM networks [60]). The output weights are trained by evolutionary algorithms.

Next, we turn to unsupervised methods for reservoir adaptation.

2.2.2 Unsupervised Methods

The unsupervised methods presented in this section try to optimize a reservoir by using the statistics of the input, but do not consider any desired output.

There has been early work on unsupervised optimization of recurrent neural networks using LSTM units by Klapper-Rybicka et al. [60]. There, networks were trained using two different kinds of information-theoretic optimization methods. One method was the binary information gain optimization (BINGO). It de-correlates the outputs of a network whose logistic units are interpreted as stochastic binary variables, finding independent dichotomies in the data. The other method was a technique called nonparametric entropy optimization (NEO). It uses kernel density estimation to approximate signal probabilities, and optimizes their entropy. Results showed that the networks trained with these algorithms could successfully perform unsupervised discrimination and classification of temporal sequences.

First attempts by Jaeger to use unsupervised Hebbian and Anti-Hebbian learning for reservoir adaptation for a smaller EVS were reported to be unsuccessful [45]. Several works using *intrinsic plasticity* (IP) in the reservoir optimization context have been published in the literature. This mechanism models changes in a neurons' intrinsic excitability in response to stimulation. Specifically, it models the phenomenon that biological neurons lower their firing threshold over time when they receive a lot of stimulation, and raise it when stimulation is lacking (see e.g. Zhang and Linden [61] and Daoudal and Debanne [62] for more details). A learning rule based on this phenomenon was introduced by Triesch [15], and further studied in [63]. It aims to maximize the information in the output of a neuron, an approach related to the one in [64]. Steil [16] showed a marked increase in performance using IP for Fermi neurons in an ESN trained by backpropagation-decorrelation (BPDC) [14]. For tanh units, an IP approach was investigated in [46], shaping the reservoir outputs to follow a Gaussian distribution. Our work on IP in Chapter 3 is closely related to this, extending this work using Laplace distributions.

A combination of IP with STDP (spike timing dependent plasticity) was studied in [47] for reservoirs with simple spiking units. The results indicated that this combination leads to reservoirs more robust to perturbations, and an increased performance in terms of short-term memory capacity and time series prediction. The combination was also reported to drive networks close to the edge of chaos region we study in Chapter 4. Synergistic effects of IP with STDP for single neurons had previously been reported by Triesch [63]. In later work [48], the authors of the study above introduced the Self-Organizing Recurrent Neural Network (called SORN). In addition to the approach in [47], they combed IP and STDP with a third plasticity rule called synaptic scaling. IP, in this study, controls the firing probability of a neuron and keeps it in a specified range, STDP detects correlations (or anti-correlations) in the data, and synaptic scaling normalizes the sum of all incoming weights to a neuron, keeping their relative proportional strength. The interaction of these plasticity mechanism enabled adaptation of the reservoir to the input data while keeping the dynamics in a "healthy" range. SORN was shown to outperform standard reservoir networks on different benchmarks by a large margin.

To conclude this section, we look at two unsupervised learning techniques which adapt the reservoir globally as opposed to the more local approaches above. Mayer and Browne [65] proposed a biologically inspired approach using a reservoir that learns to predict itself. The original reservoir activations are mixed with self-predictions in a ration $(1 - \alpha) : \alpha$. The coefficient α can be changed to get different reservoir properties. For $\alpha = 0$, the network behaves like standard ESN. For $\alpha = 1$, the reservoir becomes "autistic", relying solely on its own predictions and ignoring new input. Values of α in between these extremes were shown to lead to highly nonlinear reservoir signals that cannot always be observed in standard reservoirs.

In [38], an algebraic way to generate reservoirs with a uniform eigenvalue distribution (around the unit circle in the complex plane) in an unsupervised manner is presented. The generated reservoirs were reported to have a high entropy of unit activations. They were tested on the memory capacity task (cf. Section 3.1.1) and other synthetic problems and showed consistent performance improvement over fixed random reservoirs.

After our overview of work on reservoir optimization, we now turn to the literature closely related to our study on information processing in ESN at the edge of chaos.

2.3 Dynamical Systems at the Edge of Chaos

In a recent overview book chapter, Legenstein and Maass [66] ask the question: "[w]hat makes a dynamical system computationally powerful"? We follow their review in parts of this section, tracing some of the early work that sparked the interest in dynamical systems at the edge of chaos.

As early as 1969, Kauffman [67, 68] studied model systems for regulatory gene networks. His model networks are known as random Boolean networks (RBNs) and consist of a number of units supported by Boolean functions, and directed random connections between them. At each time step in a simulation, the dynamics of the network is iterated using the Boolean functions of each unit and inputs from their connections. A parameter K determines the average number of connections for each unit. Kauffman studied the behavior of RBNs as a function of this parameter and empirically found that his networks exhibited a phase transition from ordered to chaotic dynamics at $K = 2$. This phase transition was later analytically verified by Derrida and Pomeau [69] in their Annealed Approximation.

The conjecture that computational capabilities of dynamical systems might be maximized at this phase transition between ordered and chaotic dynamics was put forth by Wolfram [39]. He studied cellular automata (CAs) which are similar to RBNs with the difference that only local connections exist between neighboring units. Further, each unit can have one out of several possible states, as opposed to only two in RBNs. Computation in the context of CAs has the meaning of transforming an input to an output pattern where the input is the system's initial state, the program is implemented by the rules and interactions of the units, and the final pattern after evolution of the state is the output [41]. Wolfram identified four different classes for the CA dynamics and found that one class CAs showed neither stable nor chaotic behavior, but had long transients and evolved "to complex localized structures" [39].

Langton [40] later studied Wolfram's CA classes in a systematic way and confirmed his conjecture using information theoretic analysis. He asked under which conditions a dynamical system will "support the basic operations of information transmission, storage, and modification constituting the capacity to support computation?" [40]. His analysis pointed to the edge of chaos as the region generating the most powerful systems in terms of computation.

Mitchell et al. [41] however, repeated some of the experiments in [40] and found differing results. Consquently, Mitchell and colleagues challenged the idea that the dynamics at the edge of chaos would universally beneficial for computations in dynamical systems. We note here that this is also in line with results we present in Chapter 4. In [41], another meaning for computations in CAs is mentioned. The authors emphasize that "computation is not interpreted as the performance of a 'useful' transformation of the input to produce the output. Rather, it is measured in terms of generic, structural computational elements such as memory, information production, information transfer, logical operations, and so on". Furthermore, they pointed out that these "intrinsic computational elements [do] not rely on a semantics of utility" [41]. Legenstein and Maass [66] note in their review that these intrinsic computations can be used by a readout, mapping system states to outputs. This is a central idea in reservoir computing. It is also one of the motivations for our study in Chapter 4 which aims to quantify these computations, and identify the system dynamics that maximize them.

Lizier et al. [7] used the information dynamics framework [70] to quantify the fundamental nature of computation in RBNs at the order-chaos phase transition. They found that the ordered regime is dominated by information storage, while information transfer between network units is dominant in the chaotic phase. It was shown that both of these computational operations seem to be in balance around the critical point. In Chapter 4, we use the same framework to analyze the information dynamics of Echo State Networks and we will see that these results do not necessarily generalize to systems with analog units (see also the notes below on [5] which finds marked differences information processing in binary and analog systems).

In the following, we will examine studies which have looked at the behavior of Reservoir Computing networks at the edge of chaos.

2.4 Reservoir Computing at the Edge of Chaos

The phenomenon of increased computational performance in recurrent neural networks at the edge of chaos has been addressed in the literature before.

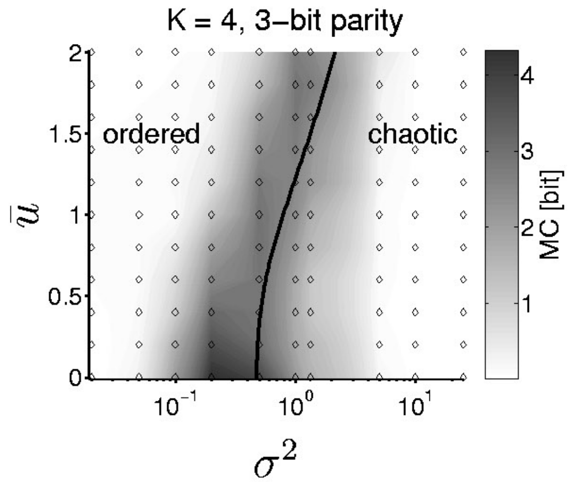


Figure 6: Memory capacity of recurrent neural networks of threshold units versus mean input level $\bar{u}$ and variance of recurrent weight strength σ^2. The network was trained on a 3-bit parity task with each node having an in-degree (number of incoming connections) of $K = 4$. The best performance (indicated by the dark areas in the plot) can be found around the critical line (from [4]).

Bertschinger and Natschläger [4] examined networks of threshold units operating on input streams and found computational performance maximized at the phase transition (see Fig. 6). They proposed the “network mediated separation” (NM-separation) criterion as a measure to quantify computational capability. This NM-separation is formed by the state distance of successive inputs, minus the state difference due to different initial states. In the stable regime, initial differences fade away quickly, and separation between inputs is low, hence the NM-separation has a low value. In the chaotic regime, small initial differences are amplified so there is a high separation, but no fading memory, resulting in a small value for the NM-separation. At the phase transition point it was found to be maximized.

In [71], the authors proposed two new measures in the context of Liquid State Machines (LSM) [37], another reservoir computing approach using spiking neuron models closer to the detailed biology. They suggested to consider the kernel quality and the generalization ability of a reservoir. Informally, the kernel quality measures how well a reservoir can map different inputs to different reservoir states. The generalization ability, on the other hand, will be high if the reservoir will only map a part of all possible inputs to different reservoir states. The computational capabilities of a reservoir, the authors argued, will then be characterized as a trade-off between the two. They showed that it is most efficient at the edge of chaos. These results were later refined in [42].

The phase transition from ordered to chaotic dynamics was also examined in [72] and in more detail in [5]. The analysis was done in order to shed light on the empirical observation that binary (including spiking) reservoirs are much more sensitive to changes in topology than analog reservoirs. The authors studied the behavior of a family of reservoir networks which interpolate between binary and analog networks by using different state quantizations. A novel calculation for the Lyapunov exponent based on branching process theory was used to assess the criticality of the networks as the in-degree (number of incoming connections to each neuron) and the reservoir weight connection strengths were varied (see Fig. 7). Their analysis revealed a qualitative difference in the computational performance around the phase transition region. In binary reservoirs, the phase transition was much more abrupt for high in-degree reservoir nodes, and the region of best performance was much narrower than in the analog case. A new mean-field predictor (which is a generalization and simplification of the one in [4]) was used to show a fundamental difference in how binary and analog circuits integrate information on shorter and longer time scales.

After this survey of related work (refer to Fig. 8 for an illustration of how previous work relates to the main chapters of the thesis), we will now describe our own experiments and results in the following chapters, which build on – and extend – work presented above.

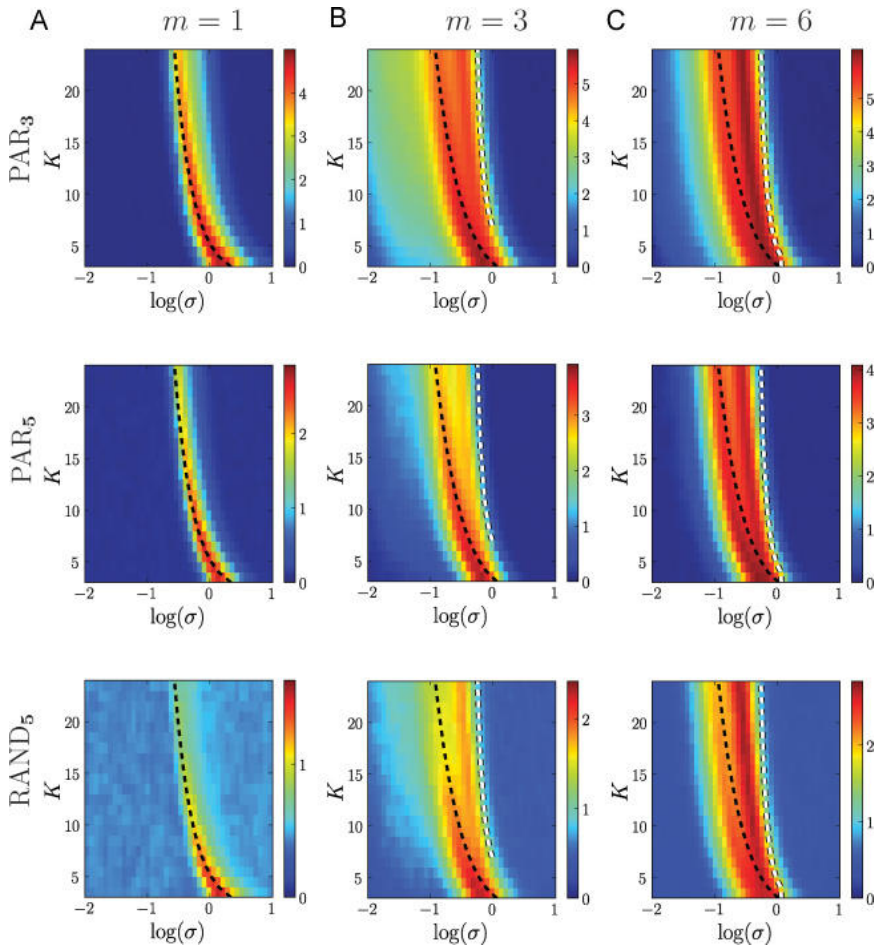


Figure 7: Average performance of quantized ESNs for different quantization levels m depending on the in-degree K and standard deviation σ of the reservoir weights in three different task settings. The phase transition is indicated by the black dashed line. Again, highest performance (dark red colors) are observed at around the phase transition line (from [5]).

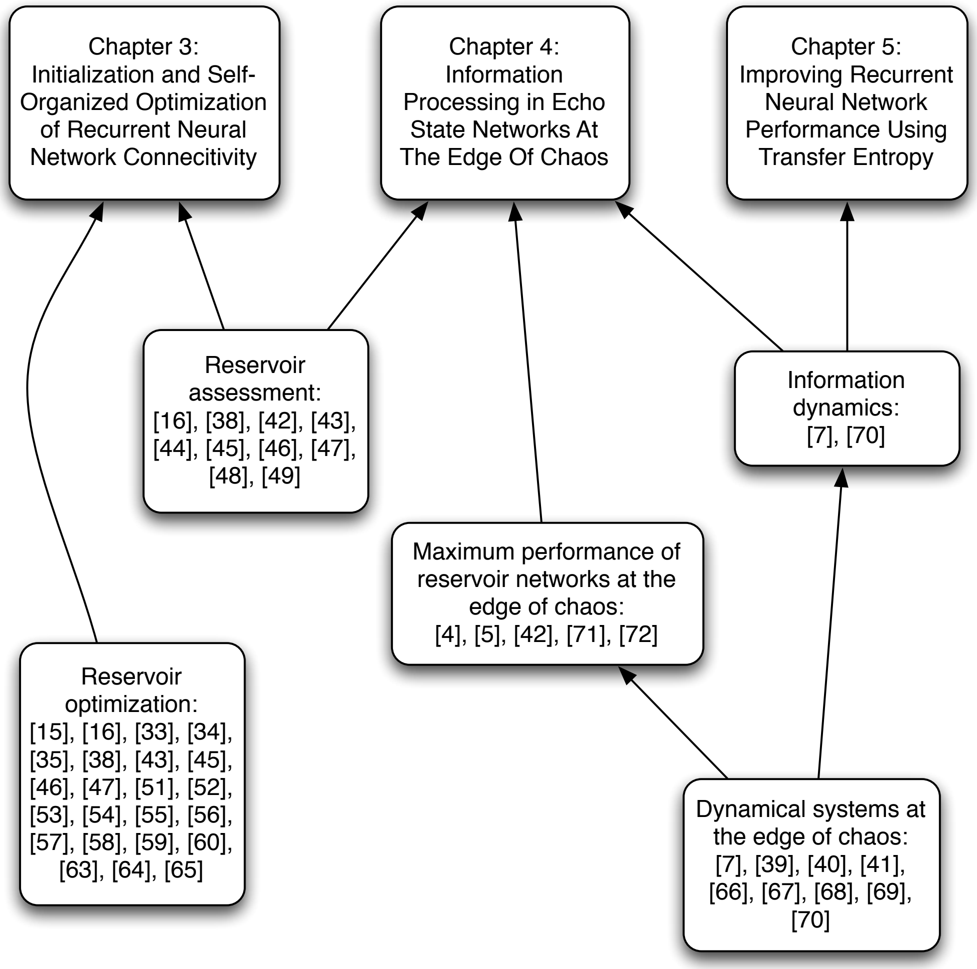

Chapter 3: Initialization and Self-Organized Optimization of Recurrent Neural Network Connecitivity
Chapter 4: Information Processing in Echo State Networks At The Edge Of Chaos
Chapter 5: Improving Recurrent Neural Network Performance Using Transfer Entropy
Reservoir assessment: [16], [38], [42], [43], [44], [45], [46], [47], [48], [49]
Information dynamics: [7], [70]
Maximum performance of reservoir networks at the edge of chaos: [4], [5], [42], [71], [72]
Reservoir optimization: [15], [16], [33], [34], [35], [38], [43], [45], [46], [47], [51], [52], [53], [54], [55], [56], [57], [58], [59], [60], [63], [64], [65]
Dynamical systems at the edge of chaos: [7], [39], [40], [41], [66], [67], [68], [69], [70]

Figure 8: Relation of Chapters 3, 4, and 5 to work presented in the literature survey of this chapter.

Chapter 3

Initialization and self-organized optimization of recurrent neural network connectivity

Here, we study the effect of approaches aiming to reduce the dependency of ESN performance on a given random initialization. In order to do this, we investigate two different approaches: the first one is based on a very specific initialization (cf. [73, 55]) of ESN reservoirs; the second one implements a pre-training phase for the reservoirs using the local IP adaptation approach as mentioned earlier in Section 2.2.2.

The reservoir initialization method is based on the idea of optimally exploiting the high dimensionality of the reservoir. The methods based on IP, while also using high-dimensional reservoirs, aim to adapt the reservoir for a high entropy of codes. Moreover, we investigate an IP based learning rule for high sparsity of codes as these have been shown to improve information processing [74].

We evaluate the different reservoir shaping and initialization methods using three different standard benchmarks. We find that reservoirs initialized with orthogonal column vectors in their connectivity matrix exhibit superior short-term memory capacity, and are also able to perform well in tasks requiring highly non-linear mappings.

Furthermore, we identify a problem with an existing IP based rule and point out limitations of the approach if traditional neuron models are used.

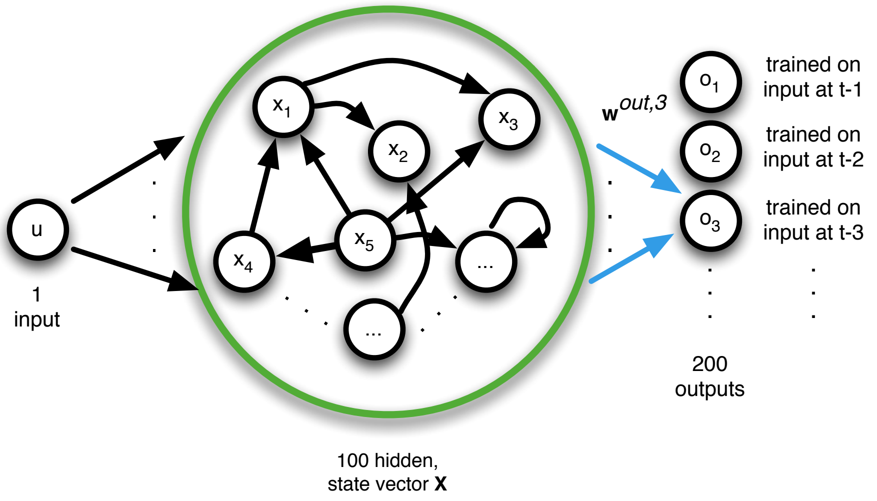


Figure 9: Setup of the ESN used in the MC task experiments. The difference in the other experiments was that only one output node was used.

3.1 Network Setup and Benchmark Description

The first set of experiments evaluated the short-term memory capacity (MC, mentioned in Section 2.1) of the different networks. In addition, we evaluated the networks on the task of modeling a 30th order NARMA (nonlinear autoregressive moving average) system, and with respect to their one-step prediction performance on the Mackey-Glass time-series. These tasks cover a reasonably wide spectrum of tests for different useful properties of reservoirs and are widely used in the literature, e.g. in [44, 46, 75, 35, 16].

For all of the experiments, we used ESNs with 1 input and 100 reservoir nodes (see Fig. 9). The number of output nodes was 1 for the NARMA and Mackey-Glass tasks, and 200 for the MC evaluation. In the latter, the 200 output nodes were trained on the input signal delayed by k steps ($k = 1 \ldots 200$). The input weights were always initialized with values from a uniform random distribution in the range $[-0.1, 0.1]$.

To compute the output weights, the reservoir node activations (state vector $\mathbf{X}$)

Table 1: Settings of IP parameters σ (for IPGAUSS) and c (for IPLAP) for the different benchmark tasks. These settings were determined empirically through systematic search of the parameter space for the optimal values.

	σ	c
MC	0.09	0.08
NARMA	0.05	0.06
Mackey-Glass	0.07	0.05

over the last 1000 of a total of 2000 steps were collected in a 100×1000 matrix $\mathbf{S}$:

$$\mathbf{S} = \begin{bmatrix} \mathbf{X}(1001) & \mathbf{X}(1002) & \mathbf{X}(1003) & \dots & \mathbf{X}(2000) \end{bmatrix}$$
$$= \begin{bmatrix} x_1(1001) & x_1(1002) & x_1(1003) & \dots & x_1(2000) \\ x_2(1001) & x_2(1002) & x_2(1003) & \dots & x_2(2000) \\ x_3(1001) & x_3(1002) & x_3(1003) & \dots & x_3(2000) \\ \vdots & \vdots & \vdots & \dots & \vdots \\ x_{100}(1001) & x_{100}(1002) & x_{100}(1003) & \dots & x_{100}(2000) \end{bmatrix}.$$

For all tasks, the output weights were then computed by offline pseudoinverse regression using matrix $\mathbf{S}$ and a matrix of desired values for the output node(s). In the case of the MC task, the delayed input was used for training as follows: let the 1000×1 matrix of desired outputs for delay k be:

$$\mathbf{D}_k = [\mathbf{u}(1001-k)\dots\mathbf{u}(2000-k)]^T,$$

e.g. for the example of $\mathbf{w}^{out,3}$ ($k = 3$) illustrated in Fig. 9:

$$\mathbf{D}_3 = [u(998)\, u(999)\, u(1000)\, \dots\, u(1997)]^T$$

The output weights $\mathbf{w}^{out}$ for output node $\mathbf{o}_k$ can then be computed as

$$\mathbf{w}^{out,k} = \mathbf{S}^\dagger \mathbf{D}_k,$$

with $(\cdot)^\dagger$ denoting the pseudoinverse, and $k = 1 \dots 200$.

In all three benchmark tasks, the parameter μ of the IP learning rule (both for **IPGAUSS** and **IPLAP**) was set to 0. The other IP related parameters were set according to table 1. For both IP methods the reservoir was pre-trained for 100000 steps

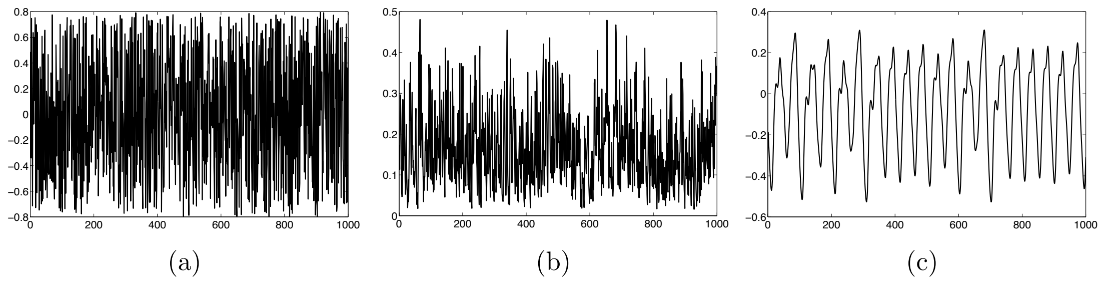


Figure 10: The time-series for the three benchmarks used to evaluate different aspects of networks performance: (a) Uniform random input, (b) Output of the NARMA 30th order system, (c) Mackey-Glass attractor with $\tau = 17$.

in order to ensure convergence to the desired probability distribution, with a learning rate of 0.0005. In all conditions, the spectral radius of the reservoir connectivity matrix was scaled to 0.95 (prior to pre-training in case of IP).

Different input time-series were used for training the output weights and for testing in all cases. The input length for testing was always 2000 steps. The first 1000 steps of the reservoir node activations were discarded to get rid of transient states due to initialization with zeros before calculating the output weights and the test error.

3.1.1 Short Term Memory Capacity

To evaluate the short-term memory capacity of the different networks, we computed the k-delay memory capacity (MC_k) defined in [44] as

$$MC_k = \frac{cov^2(\mathbf{u}_{t-k}, \mathbf{o}_t)}{\sigma^2(\mathbf{u}_{t-k})\sigma^2(\mathbf{o}_t)}$$

This is essentially a squared correlation coefficient between the desired signal delayed by k steps and the reconstruction by the kth output node of the network. The actual short-term memory capacity of the network is defined as $MC = \sum_{k=1}^{\infty} MC_k$, but since we can only use a finite number of output nodes, we limited their number to 200, which is sufficient to see a significant drop-off in performance for the networks in all of the tested conditions. The input for the MC task was random values sampled from a uniform random distribution in the range $[-0.8, 0.8]$.

3.1.2 NARMA 30th Order System

For the NARMA 30th-order modeling task, the input time series $x(t)$ was sampled from a uniform random distribution between $[0, 0.5]$. The desired output at time $t+1$ was calculated as:

$$y(t+1) = 0.2y(t) + 0.004y(t)\sum_{i=0}^{29} y(t-i) + 1.5x(t-29)x(t) + 0.001$$

3.1.3 Mackey-Glass Time-Series

Finally, the Mackey-Glass time-series is computed by integrating the system

$$\dot{y} = \frac{0.2y(t-\tau)}{1+y(t-\tau)^{10}} - 0.1y(t),$$

from time step t to $t+1$. The τ parameter was set to 17 in order to yield a mildly chaotic behavior.

The evaluation for the NARMA modeling and the Mackey-Glass prediction tasks was done using the normalized root mean squared error measure, defined as:

$$NRMSE = \sqrt{\frac{\langle(\tilde{y}(t) - y(t))^2\rangle_t}{\langle(y(t) - \langle y(t)\rangle_t)^2\rangle_t}},$$

where $\tilde{y}(t)$ is the sampled output and $y(t)$ is the desired output.

3.2 IP Learning and a Rule for a Laplace Output Distribution

IP learning was introduced in [15] as a way to improve information transmission in neurons while adhering to homeostatic constraints like limited energy usage. For a fixed energy expenditure (represented by a fixed mean of the neurons output distribution), the distribution that maximizes the entropy (and therefore the information transmission) is the exponential distribution. This was used for single neurons in [15] and for neurons in a reservoir in [16] where it lead to a performance improvement over standard random reservoirs. In [46], IP learning for a Gaussian output distribution

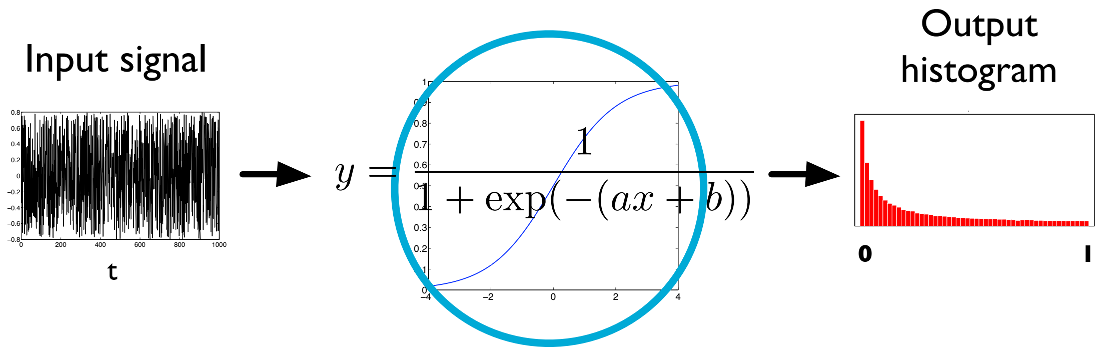


Figure 11: Homeostatic regulation based on intrinsic plasticity: the incoming signal drives the network while a gain and a bias parameter in the transfer function (a and b in the logistic function above) are adapted to achieve the desired output distribution of neuron activity values.

of reservoir neurons was investigated, which is the maximum entropy distribution if, in addition to the mean, the variance of the output distribution is fixed. Again, an overall increase in performance was noted for several benchmark problems.

A Laplace distribution would lead to sparser codes than the Gaussian, and our hypothesis was that enough entropy would be preserved for a good input signal approximation. Researching Laplace output distributions was also suggested in [46] for similar reasons. Here, analogous to the calculations in [15, 46], we derive an IP learning rule for this distribution to test our hypothesis.

In order to model the changes in intrinsic excitability of the neurons, the transfer function of our neurons is generalized with a gain parameter a and a bias parameter b:

$$y = f(x) = \tanh(ax + b).$$

The Laplace distribution, which we desire as the reservoir neurons' output distribution, is defined as

$$f(x \mid \mu, c) = \frac{1}{2c}\exp\left(-\frac{|x-\mu|}{c}\right), \quad c \neq 0.$$

Let $\tilde{p}_y(y)$ denote the sampled output distribution of a reservoir neuron and let the desired output distribution be $p(y)$, thus $p(y) = f(y \mid \mu, c)$. In the learning process, we try to minimize the difference between $\tilde{p}_y(y)$ and $p(y)$, which can be measured with the Kullback-Leibler divergence D_{KL}. Thus, we try to minimize:

$$\begin{aligned}
D_{KL} &= \int \tilde{p}_y(y) \log \left(\frac{\tilde{p}_y(y)}{\frac{1}{2c} \exp(-\frac{|y-\mu|}{c})} \right) dy \\
&= \int \tilde{p}_y(y) \log \tilde{p}_y(y) dy - \int \tilde{p}_y(y) \log(\frac{1}{2c}) dy \\
&\quad - \int \tilde{p}_y(y) \log \left(\exp(-\frac{|y-\mu|}{c}) \right) dy \\
&= \int \tilde{p}_y(y) \log \tilde{p}_y(y) dy + \int \tilde{p}_y(y) \frac{|y-\mu|}{c} dy + C \\
&= \int \tilde{p}_y(y) \log \left(\frac{\tilde{p}_x(x)}{\frac{dy}{dx}} \right) dy + \int \tilde{p}_y(y) \frac{|y-\mu|}{c} dy + C \\
&= \int \tilde{p}_y(y) \log \tilde{p}_x(x) dy - \int \tilde{p}_y(y) \log(\frac{dy}{dx}) dy \\
&\quad + \int \tilde{p}_y(y) \frac{|y-\mu|}{c} dy + C \\
&= \log \tilde{p}_x(x) + E \left(- \log(\frac{dy}{dx}) + \frac{|y-\mu|}{c} \right) + C
\end{aligned}$$

where – as in relevant previous work – we have made use of the relation $\tilde{p}_y(y)dy = \tilde{p}_x(x)dx$ where $\tilde{p}_x(x)$ is the sampled distribution of the input. Writing this as $\tilde{p}_y(y) = \frac{\tilde{p}_x(x)}{\frac{dy}{dx}}$ and substituting it for $\tilde{p}_y(y)$ in the first term of the above equation. In order to minimize the function D_{KL}, we first derive it with respect to the bias parameter b:

$$\begin{aligned}
\frac{\partial D_{KL}}{\partial b} &= \frac{\partial}{\partial b} E \left(- \log(\frac{dy}{dx}) + \frac{|y-\mu|}{c} \right) \\
&= E \left(- \frac{\frac{\partial^2 y}{\partial b \partial x}}{\frac{dy}{dx}} + \frac{(y-\mu)(1-y^2)}{c|y-\mu|} \right)
\end{aligned}$$

The first term in the above equation is

$$\frac{\partial^2 y}{\partial b \partial x} \left(\frac{dy}{dx} \right)^{-1} = \frac{-2ay(1-y^2)}{a(1-y^2)} = -2y$$

so we have:

$$\frac{\partial D_{KL}}{\partial b} = E \left(2y + \frac{y(1-y^2+\mu y) - \mu}{c|y-\mu|} \right) \quad y \neq \mu.$$

The derivation with respect to the gain parameter a is analogous and yields:

$$\frac{\partial D_{KL}}{\partial a} = E\left(2xy + \frac{yx(1-y^2+\mu y)-\mu x}{c|y-\mu|} - \frac{1}{a}\right) \quad y \neq \mu.$$

From these derivatives, we identify the following learning rules for stochastic gradient descent with learning rate η:

$$\Delta b = -\eta\left(2y + \frac{y(1-y^2+\mu y)-\mu}{c|y-\mu|}\right).$$

$$\begin{aligned}\Delta a &= -\eta(-\frac{1}{a}) - \eta\left(2xy + \frac{yx(1-y^2+\mu y)-\mu x}{c|y-\mu|}\right)\\ &= \frac{\eta}{a} + \Delta b x.\end{aligned}$$

3.3 Reservoirs Based on Permutation Matrices

Orthogonal networks [73] have an orthogonal reservoir matrix $\mathbf{W}$ (i.e. $\mathbf{W}\mathbf{W}^{\mathbf{T}} = 1$) and linear activation functions. These networks are inspired by a distributed version of a delay line, where input values are embedded in distinct orthogonal directions, leading to high memory capacity [73]. Permutation matrices, as used by [55], consist of randomly permuted diagonal matrices and are a special case of orthogonal networks. Here, and in [55], the hyperbolic tangent (tanh) activation function was used, in order to facilitate non-linear tasks beyond memorization.

3.4 Results

We tested ESN with four different conditions for the connectivity matrix of the reservoir. In condition **RND**, the reservoir matrix was initialized with uniform random values between $[-1, 1]$. Condition **PMT** tested a permutation matrix for the reservoir connectivity. Finally, we used IP optimization with a Gaussian distribution (cf. [46]) in **IPGAUSS** and a Laplace distribution (as derived above) in **IPLAP**. In all conditions, the reservoirs were scaled to have a spectral radius of 0.95. In the case of the IP methods, this scaling was done once before the IP training was started.

The results of the experiments are given in Table 2, averaged over 50 simulation runs for each of the four conditions. The networks in the **PMT** condition essentially

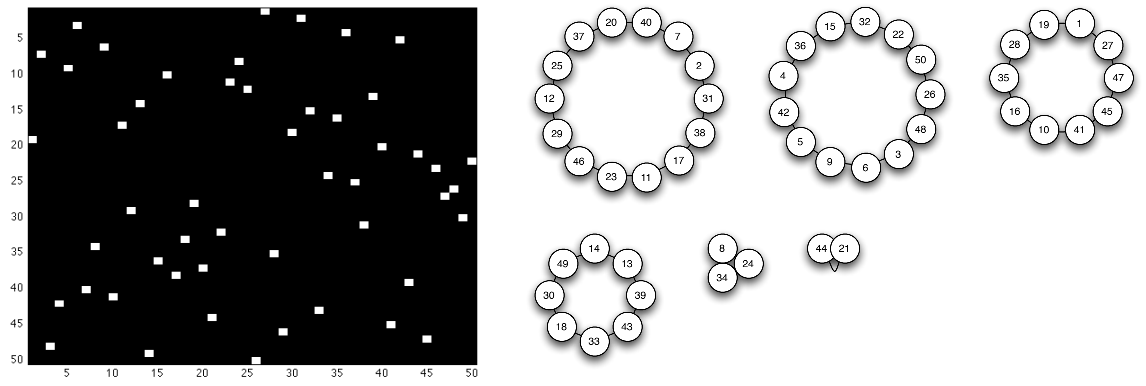


Figure 12: (left) Visualization of a 50 × 50 permutation matrix with black squares indicating a zero, and white squares indicating a value of one; (right) rings of (randomly) different sizes are the result of the connectivity defined by a permutation matrix. The rings here correspond to the the matrix on the right.

Table 2: Results of the experiments (averaged over 50 simulation runs) for the three benchmarks in four conditions in tabular form (cf. Fig. 13). Errors for Mackey-Glass are scaled by a factor of 10^{-4}.

	PMT	**RND**	**IPGAUSS**	**IPLAP**
Memory Capacity	**62.501** (5.086)	31.884 (2.147)	33.019 (2.464)	32.175 (3.127)
$\text{NRMSE}_{\text{NARMA}}$	**0.385** (0.022)	0.473 (0.035)	0.465 (0.053)	0.482 (0.041)
$\text{NRMSE}_{\text{Mackey-Glass}}$	3.373 (0.292)	2.411 (0.242)	2.802 (0.416)	**2.375** (0.416)

show double the memory capacity of networks in the other conditions, while networks pre-trained with **IPGAUSS** and **IPLAP** have very similar values and show a slight increase compared to condition **RND**. Fig. 14 shows plots of the individual MC_k curves for all conditions in the MC task, with the curve for **PMT** showing much longer correlations than all the others. The results for the NARMA modeling task are less pronounced, but look similar in that the **PMT** networks perform better than the other tested conditions. The normalized root mean squared error (NRMSE) for **IPLAP** and **IPGAUSS** is very similar again, however, only **IPGAUSS** has a slight advantage over **RND**. For the Mackey-Glass one-step prediction, the performance of the **IPLAP** networks is better than the other ones, slightly ahead of **RND** (difference well within

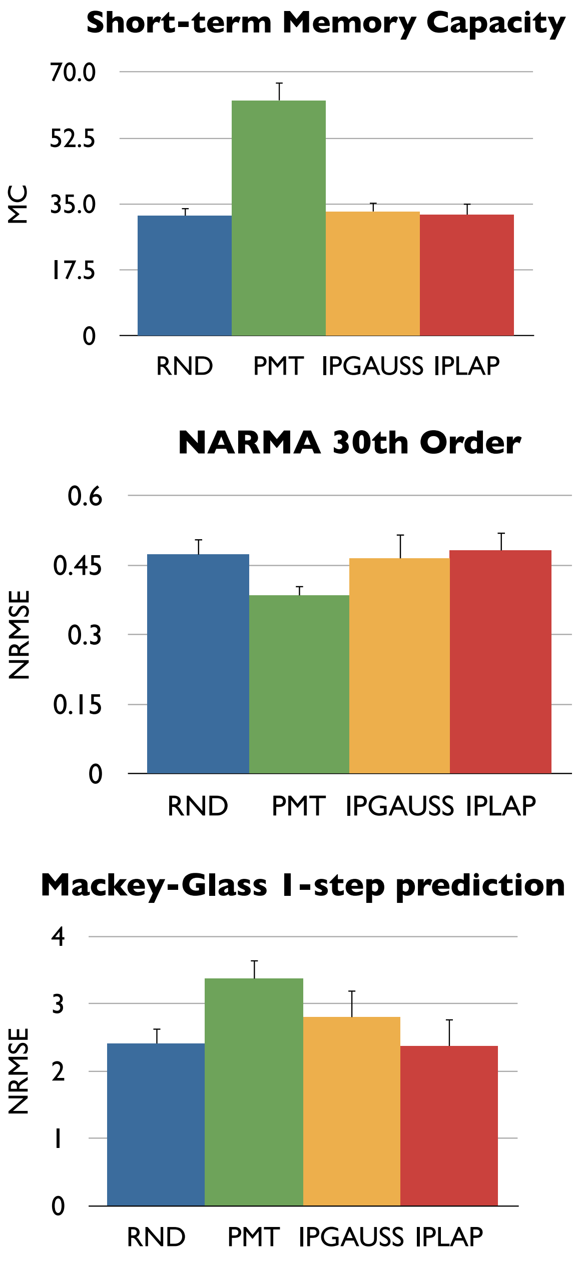


Figure 13: Average memory capacity (graph a) and normalized root mean squared error (NRMSE) for the NARMA modeling and the Mackey-Glass prediction tasks (graphs b and c, respectively) in the four different conditions, averaged over 50 simulation runs with standard dev. in parenthesis. Note that the errors for Mackey-Glass are scaled by a factor of 10^{-4}. For the memory capacity (a), higher bars are better, while lower bars in the errors are better for the prediction tasks (b and c).

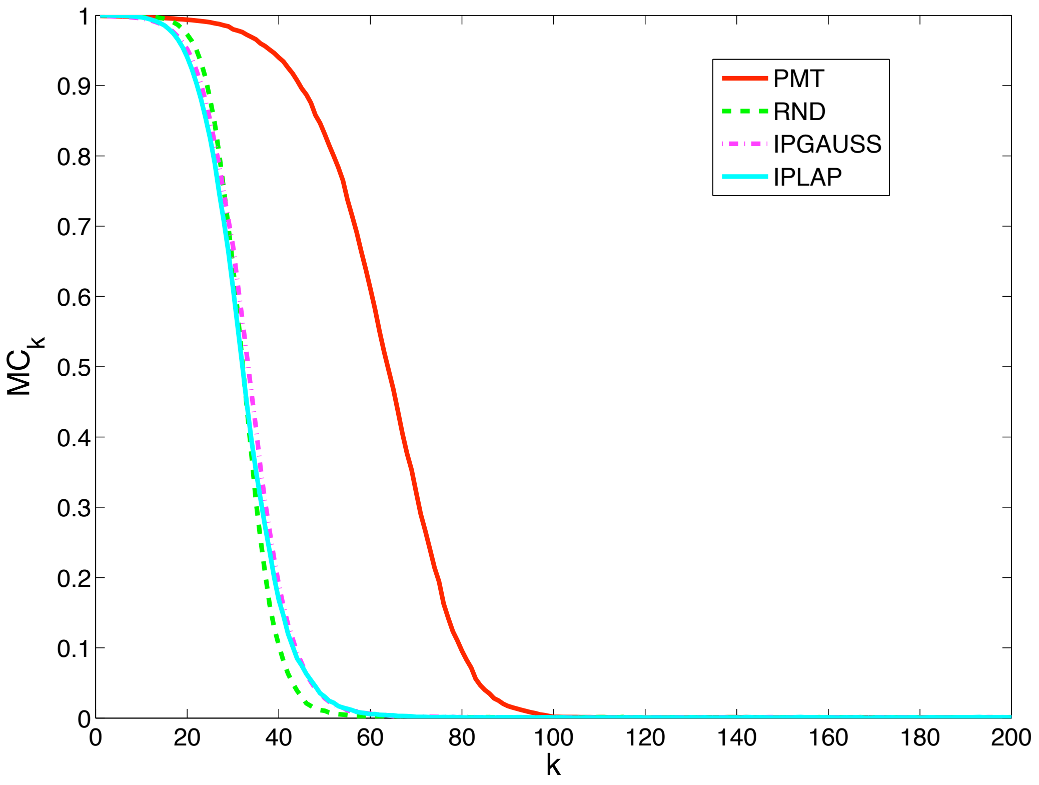


Figure 14: The $\mathrm{MC_k}$ curves for the uniform random input data. The plot indicates how well the input signal can be reconstructed ($\mathrm{MC_k}$) for increasing delay times k

.

the standard deviation of the error, however). The **PMT** networks perform worst on this task.

The superior performance of networks in **PMT** on the short-term memory task could be expected: networks with a connectivity based on permutation matrices form are a particular instance of orthogonal networks, which, in turn, can be seen as a multidimensional version of a delay line. From a system theory perspective, the eigenvalues (poles) of the linearized system implemented by the network correspond to bandpass filters with center frequencies according to their angle in the complex plane [38]. Larger eigenvalues will lead to longer time constants for the filters, preserving information for longer time in the network. Figure 15 (b) shows that the eigenvalues of the connectivity matrix of a 100 node ESN in the **PMT** condition are all of the same magnitude, and are spaced relatively uniformly just below the unit circle (the reservoir matrix was scaled to have a maxiumum absolute eigenvalue of 0.95 in this case, i.e., the matrix elements were either 0 or 0.95). The filters implemented by this network will thus have long time constants and provide support for many different frequencies in order to reconstruct the input signal. Compare this to the distribution of the eigenvalues of a connectivity matrix of an equally sized **RND** network in Figures 15 (a) and (c): they are much less uniformly distributed and have very different magnitudes, resulting in a mixture of both longer and shorter time constants for the network.

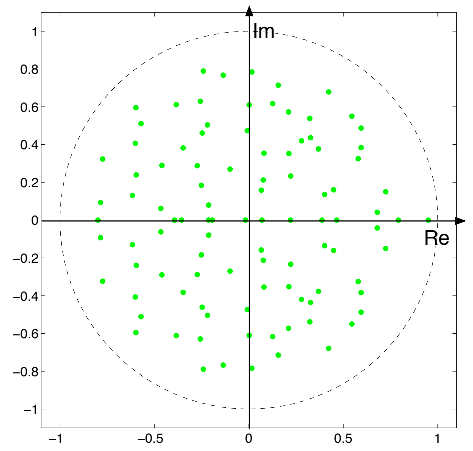
The **PMT** networks also outperform the other methods on the highly non-linear NARMA task, which is less obvious. The NARMA task needs long memory, which the orthogonal reservoirs in **PMT** are able to provide; but one might suspect (also based on the poor performance of ring-shaped reservoirs in [46]) that the specific, rather sparse connectivity would not be able to perform the kind of non-linear mappings that the task requires (since there is less interaction between neurons in the reservoir than in networks which are more densely connected). The results show that this is not the case.

The Mackey-Glass prediction task requires shorter time constants and less memory than the other two tasks. In this case, the **IPLAP** networks perform best, slightly ahead of the **RND** condition. The **PMT** networks have the same spectral radius as the ones in **RND**, however, all eigenvalues in **PMT** have the same (large) magnitude. Therefore, the network is missing elements implementing shorter time constants, which would let it react to fast changes in the input. The best results we got on this task were actually achieved using ESN with fermi neurons and IP learning with an exponential output distribution (results not shown). In this case, the results were significantly better than the ones in **RND** (cf. also [16]).

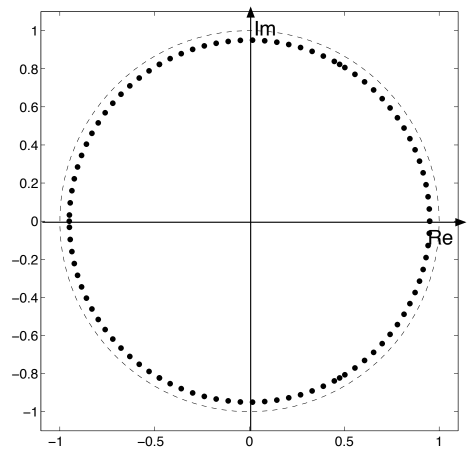
3.5 IP Revisited

A closer investigation of the almost identical performance of both IP methods revealed that **IPLAP** also generated normally distributed output, very similar to **IPGAUSS**. To better understand the effect of the different IP rules, we used IP to approximate

(a)



(b)



(c)

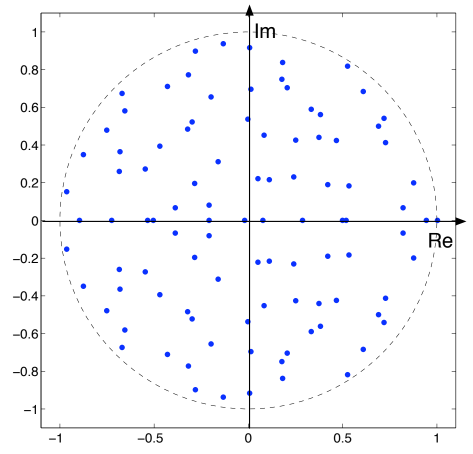


Figure 15: Plot of the reservoir matrix eigenvalues in the complex plane for a 100 node network (a) in the RND condition, (b) for a PMT matrix, and (c) in the after training with IPLAP. All matrices have been scaled to have a spectral radius of 0.95

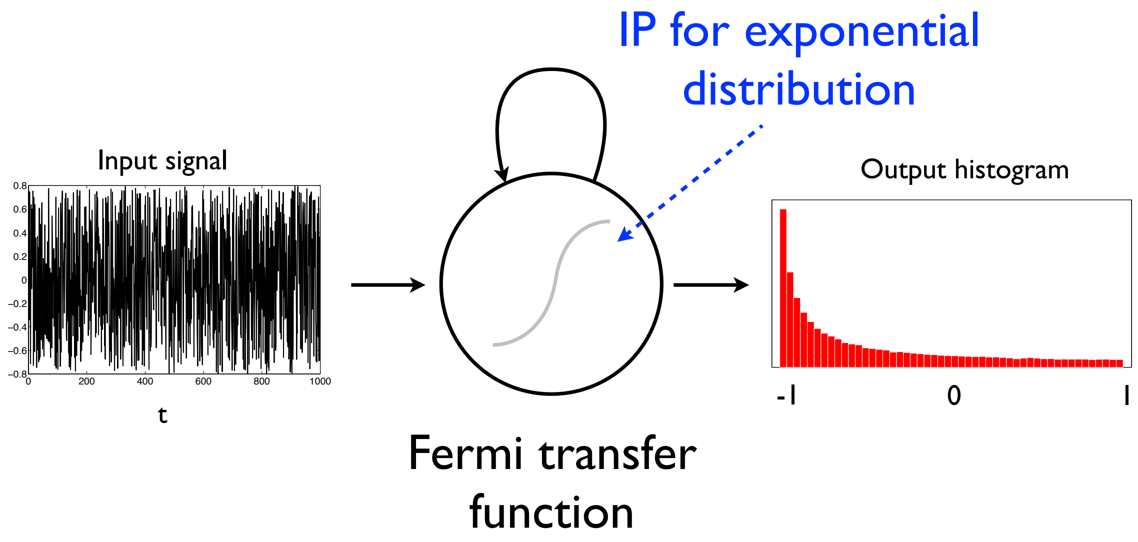
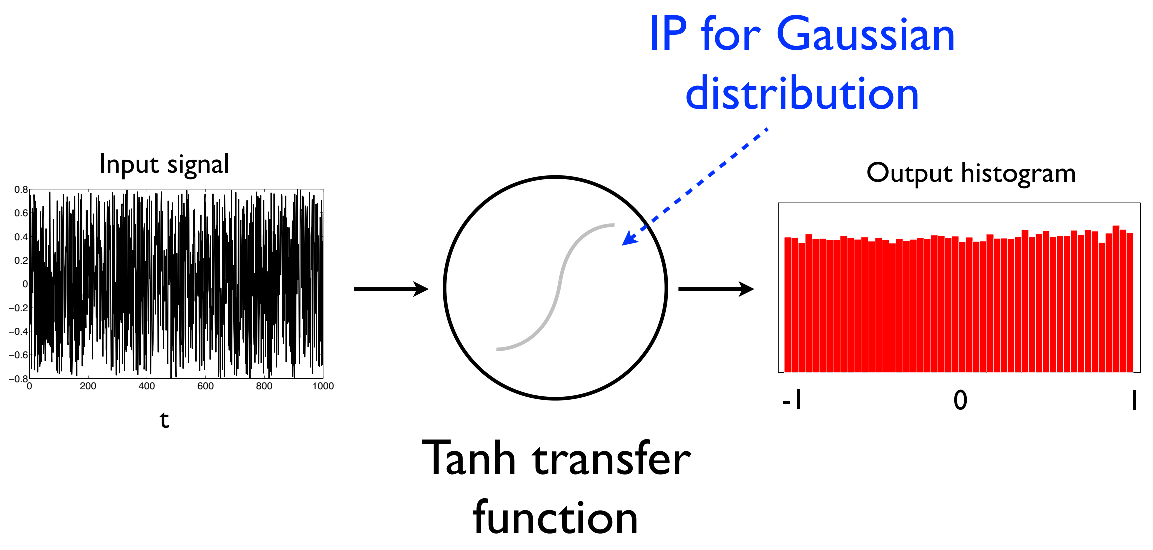


Figure 16: Result of IP learning with a single fermi neuron with and without self-recurrence and a learning rule for the exponential output distribution. IP successfully produces the desired output distribution.

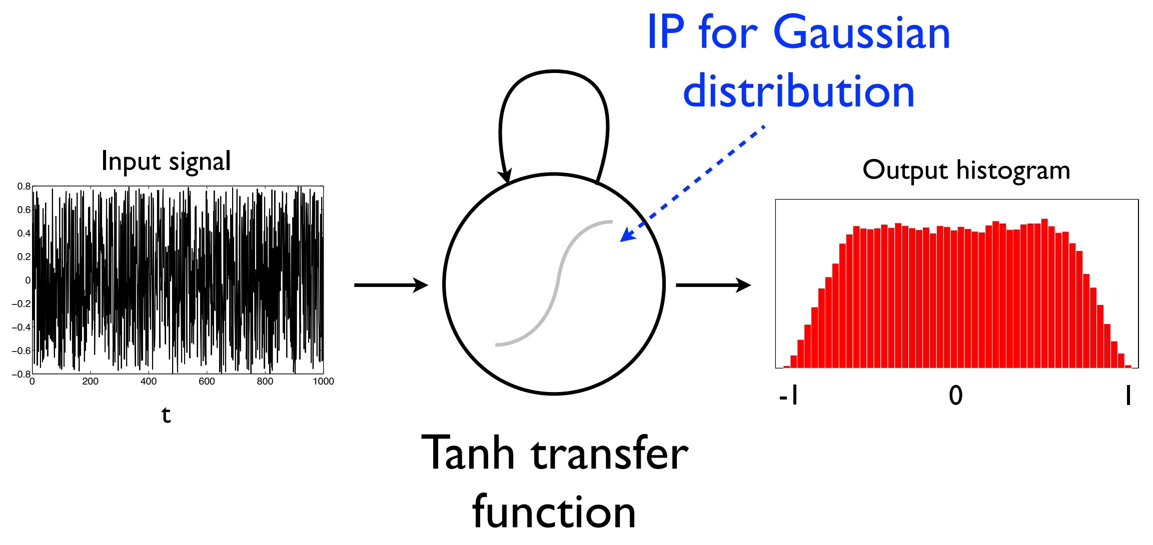
the Laplace, the Gaussian (both with a tanh activation function), and the exponential distribution (logistic activation function), respectively, with a single feedforward unit and uniformly distributed input on the interval $[-1, 1]$. As expected, the IP learning rule can successfully generate exponentially distributed output values (Fig. 16), with and without self-recurrence.

IP fails, however, to generate output distributions that resemble the Gaussian or the Laplace (Fig. 17, a and b) if no self-recurrence is present. This seems surprising in particular for the Gaussian, as IP has successfully been used to shape the output distribution of a reservoir [46] towards that distribution. A possible explanation for this phenomenon is discussed later in Section 6.

In the next chapter, we leave the topic of reservoir optimization and turn instead to reservoir assessment, and to the question how elements of intrinsic computation in reservoirs change as ESNs go through the phase transition from ordered to chaotic dynamics. We will return to reservoir optimization with an approach based on adaptation of information transfer in the reservoir in Chapter 5.



(a)



(b)

Figure 17: (a) Effect of IP learning on a single tanh neuron without self-recurrence, trained with a learning rule for a Gaussian output distribution. IP learning fails to achieve the desired result: the best it can do is to drive the neuron to a uniform output distribution, which has the smallest distance (for the given transfer function) to the desired distributions; (b) same as (a), but with self-recurrence added to the neuron. The achieved output distribution is significantly more Gaussian-shaped than without the self-recurrence. The effect is amplified in a network where the neurons receive additional inputs with similar distributions. All units were trained using 100000 training steps and uniformly distributed input data on the interval $[-1, 1]$.

Chapter 4

Information processing in Echo State networks at the edge of chaos

A fundamental question in Reservoir Computing is how the recurrent hidden layer or reservoir should be prepared, designed or *guided*, to best facilitate the training of connections to output units and consequently maximize task performance. It has been previously shown that the ability of reservoir computing networks to achieve the desired computational outcome is maximized when the network is prepared in a state near the *edge of chaos* [42, 72, 5]. This refers to a critical state between ordered dynamics (where disturbances quickly die out) and chaotic dynamics (where disturbances are amplified). This property is particularly interesting because of evidence in the literature that cortical circuits are tuned to criticality [see e.g. 76, 77, 78]. The reasons why network performance is increased near the edge of chaos are, however, not yet fully understood.

We mentioned several related works back in Section 2.4 and these quantitative studies have surely helped to gain insight into the increased computational performance at the critical point. However, we argue that they measured the elements of ongoing computation only *indirectly* and on a *global scale* (network perspective).

In this chapter, we seek to *directly* measure the computational capabilities of the reservoir as it undergoes the phase transition to chaotic dynamics. In particular, we will measure the information storage at each neuron, and information transfer between each neuron in the reservoir. This contrasts with examining the entropy

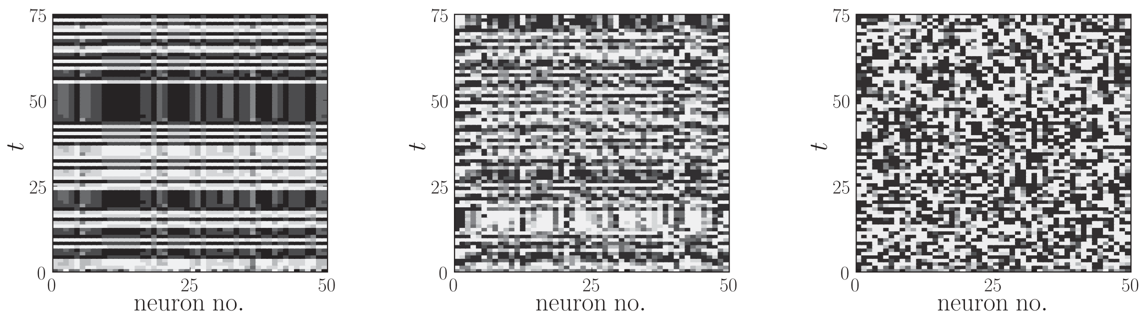


Figure 18: Change in the hidden neuron activations as the network dynamics go from the ordered (left panel) to the chaotic regime (right panel). The dynamics in the phase-transition region are shown in the middle (from [5]).

of each unit alone, since these measures relate directly to the computational tasks being performed. Furthermore, it means that we can directly quantify whether the computational properties provided by the reservoir are maximized at the edge of chaos, and we can do so on a more *local scale* (node perspective). Finally, the general applicability of these measures allow us to compare the computations in different kinds of dynamical systems.

4.1 Information-Theoretical Measures

A natural framework in order to describe distributed computation in dynamical systems is found in *information theory* [79, 80]. It has proven useful in the analysis and design of a variety of complex systems [81, 82, 83, 84, 85, 86, 7], as well as in theoretical neuroscience [87, 88, 89, 90]. A short review of basic information-theoretic quantities that are the basis for the measures we use for information storage and transfer in multivariate systems is given in Appendix B.

We are interested in the process by which each variable or node X in a system updates or *computes* its next state. Such computations utilize information storage from the node itself, and information transfer from other nodes.

The *information storage* of a node is the amount of information in its past that

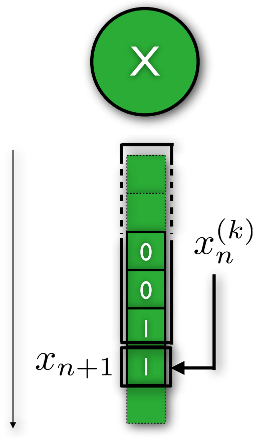


Figure 19: Information storage of a node is the amount of information in its past x_n^k that is relevant to predicting its future x_{n+1}.

is relevant to predicting its future. We quantify this concept using the *active information storage* (AIS) to measure is the stored information that is *currently in use* in computing the next state of the node [91, 70]. The active information storage for a node X is defined as the average mutual information between its semi-infinite past $x_n^{(k)}$ and its next state x_{n+1}:

$$A_X = \lim_{k\to\infty} \sum_{x_{n+1},x^{(k)}} p(x_{n+1}, x^{(k)}) \log_2 \frac{p(x_n^{(k)}, x_{n+1})}{p(x_n^{(k)})p(x_{n+1})}. \tag{1}$$

$A_X(k)$ represents an approximation with finite history length k.

From our computational perspective, a node can store information regardless of whether it is causally connected with itself; i.e. for ESNs, this means whether or not the node has a self-link. This is because information storage can be facilitated in a distributed fashion via one's neighbors, which amounts to the use of stigmergy (e.g. see [92]) to communicate with oneself [70].

The *information transfer* (see Fig. 20) between a source and a destination node is defined as the information provided by the source about the destination's next state that was not contained in the past of the destination. The information transfer is formulated in the *transfer entropy* (TE), introduced by Schreiber [93] to address concerns that the mutual information (as a de facto measure of information transfer) was a symmetric measure of statically shared information. The transfer entropy

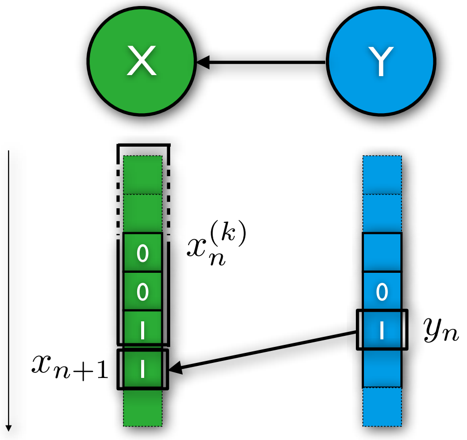


Figure 20: Information transfer between a source node Y and a destination node X is defined as the information y_n provided by the source about the destination's next state x_{n+1} that was not contained in the past x_n^k of the destination.

from a source node Y to a destination node X is the mutual information between the previous state of the source[1] y_n and the next state of the destination x_{n+1}, *conditioned* on the semi-infinite past of the destination $x_n^{(k)}$ (as $k \to \infty$ [94]):

$$T_{Y \to X} = \lim_{k \to \infty} \sum_{\mathbf{u}_n} p(\mathbf{u}_n) \log_2 \frac{p(x_{n+1}|x_n^{(k)}, y_n)}{p(x_{n+1}|x_n^{(k)})}, \tag{2}$$

where $\mathbf{u}_n$ is the state transition tuple $(x_{n+1}, x^{(k)}, y_n)$. Again, $T_{Y \to X}(k)$ represents finite-k approximation.

4.2 Estimating the Criticality of an Input-Driven ESN

In order to determine whether a dynamical system has ordered or chaotic dynamics, it is common to look at the average sensitivity to perturbations of its initial conditions [69, 4, 5]. The rationale behind this is that small differences in the initial conditions of two otherwise equal systems should eventually die out if the system is

[1]The transfer entropy can be formulated using the l previous states of the source. However, where only the previous state is a causal information contributor (as for ESNs), it is sensible to set $l = 1$ to measure direct transfer only at step n.

in the ordered phase, or persist (and amplify) if it is in the chaotic phase. A measure for the exponential divergence of two trajectories of a dynamical system in state space with very small initial separation is the Lyapunov (characteristic) exponent (LE). Although a whole spectrum of Lyapunov exponents is defined, the rate of divergence is dominated by the largest exponent. It is defined as:

$$\lambda = \lim_{k\to\infty} \frac{1}{k} \ln\left(\frac{\gamma_k}{\gamma_0}\right)$$

with γ_0 being the initial distance between the perturbed and the unperturbed trajectory, and γ_k being the distance at time k. For sub-critical systems, $\lambda < 0$ and for chaotic systems $\lambda > 0$. A phase transition thus occurs at $\lambda \approx 0$ (called the *critical point*, or *edge of chaos*).

Since this is an asymptotic quantity, it has to be estimated for most dynamical systems. We adopt here the method described in [95, Chap. 5.6]. Two identical networks are simulated for a period of 1000 steps (longer durations were tried but found not to make a significant difference). After this initial period serving to run out transient random initialization effects, proceed as follows.

1. Introduce a small perturbation into a unit n of one network, but not the other. This separates the state of the perturbed network $\mathbf{x}^2$ from the state of the unperturbed network $\mathbf{x}^1$ by an amount γ_0[2].

2. Advance the simulation one step and record the resulting state difference for this k-th step $\gamma_k = \|\mathbf{x}^1(k) - \mathbf{x}^2(k)\|$. The norm $\|\cdot\|$ denotes the Euclidean norm in our case, but can be chosen differently.

3. Reset the state of the perturbed network $\mathbf{x}^2$ to $\mathbf{x}^1(k) + (\gamma_0/\gamma_k)(\mathbf{x}^2(k) - \mathbf{x}^1(k))$. This renormalization step keeps the two trajectories close in order to avoid numerical overflows (see Fig. 21 for an illustration of these steps).

In [95], γ_k is added to a running average and steps 2 and 3 are performed repeatedly until the average converges. Here, we repeat these simulation and renormalization

[2]This initial separation has to be chosen carefully. It should be as small as possible, but still large enough so that its influence will be measurable with limited numerical precision on a computer. We found 10^{-12} to be a robust value in our simulations, which is also recommended by Sprott [96] for the precision used in this study.

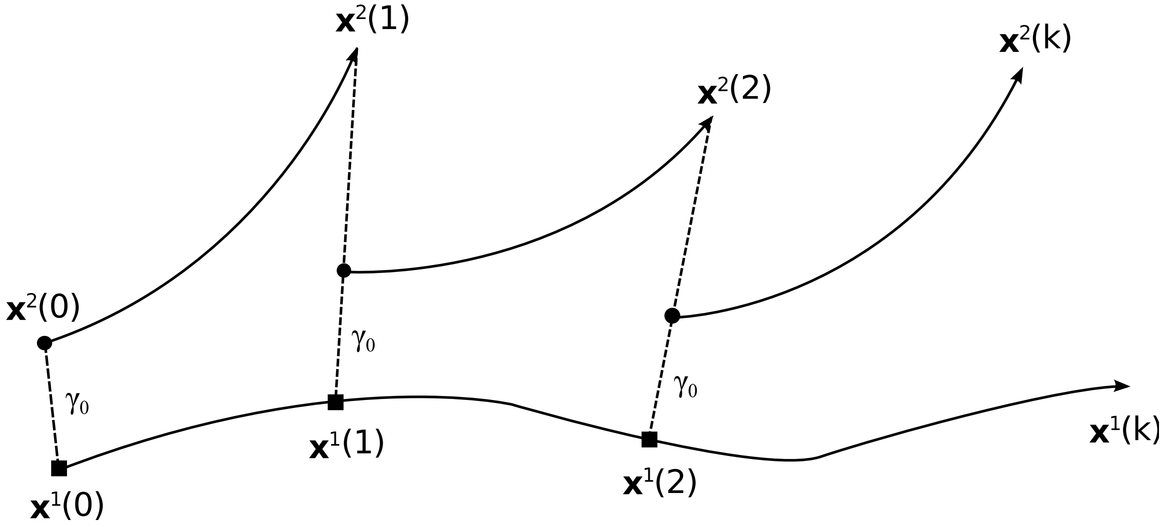


Figure 21: Numerical estimation of the largest Lyapunov exponent λ. Trajectories are kept close by resetting the distance to γ_0 after each update step in order to avoid numerical overflows (illustration after [6]). See text for more details.

steps for a total of 1000 times (again, longer durations were tested, but found not to change results significantly), and then average the logarithm of the distances along the trajectory as $\lambda_n = \langle \ln(\gamma_k/\gamma_0) \rangle_k$.

For each reservoir with N units that is tested, we calculate N different λ_n values, choosing a different reservoir unit n to be perturbed each time. These values are then averaged to yield a final estimate of the Lyapunov exponent $\lambda = \langle \lambda_n \rangle_n$.

4.3 Results

In order to investigate the relation between information transfer, active information storage, and criticality in ESNs, we used networks whose reservoir weights were drawn from a normal distribution with mean zero and variance σ^2. We changed this parameter between simulations so that $\log \sigma$ varied between$[-1.5, -0.5]$, increasing in steps of 0.1. A more fine grained resolution was used close to the edge of chaos, between $[-1.2, -0.9]$. Here, we increased log σ in steps of 0.02. We recorded the estimated Lyapunov exponent λ as described in Section 4.2, the information measures described in the previous section, and a parameter for task performance described below.

The active information storage was measured for each reservoir unit, and the transfer entropy between each reservoir unit pair. A history size of $k = 2$ was used in the TE and AIS calculations, and kernel estimation with a fixed radius of 0.2 was used to estimate the required probabilities. We recorded 15000 data points for each time series after discarding 1000 steps to get rid of transients. The output weights were trained with 1000 simulation samples using a one-shot pseudoinverse regression. Input weights were drawn uniformly between $[-0.1, 0.1]$.

We used two common benchmark tasks to evaluate network performance. The first task was used to assess the memory capacity of the networks as defined in Jaeger [44]. For this task, ESNs with a single input, 150 reservoir nodes, and 300 output nodes were used. The input to the network was a uniformly random time series drawn from the interval $[-1; 1]$. Each of the outputs was trained on a delayed version of the input signal, i.e. output k was trained on input(t - k), $k = 1 \ldots 300$. To evaluate the short-term memory capacity, we computed the k-delay memory capacity (MC_k) as defined in Section 3.1.1.

The second benchmark task we used was again to model a 30th-order NARMA system as described in Section 3.1.2. We trained networks with a single input, 150 reservoir neurons, and one output neuron. As before, the performance for this task was evaluated using the normalized root mean squared error measure described in Section 3.1.3.

4.3.1 Effect of the Initial Separation Size on Numerical Estimation of the Largest Lyapunov Exponent

Figure 22 shows curves for different initial separation sizes using the networks in the MC task. The log σ parameter is given along the x-axis, LE is on the y-axis. For initial separation values $> 10^{-13}$, the LE was overestimated due to the nonlinearity of the tanh function and the numerical limit of Matlab for the values between $[-1, 1]$. This limit can be tested using the command "eps" in Matlab; its value is about $2.2 * 10^{-16}$ for values around 1.0.

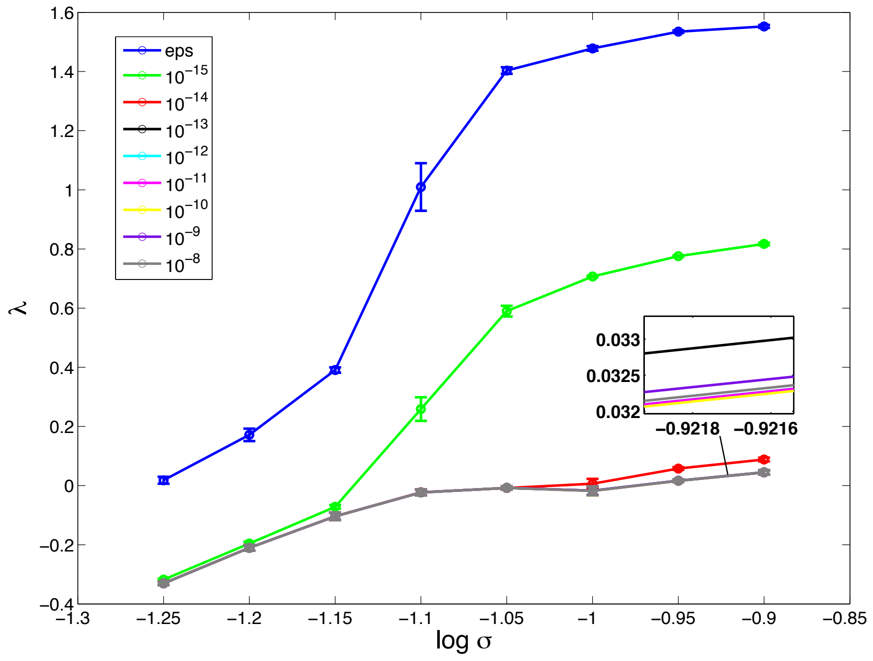


Figure 22: Testing different initial separations in the LE calculations.

We finally used 10^{-12} as initial separation in the experiments, a value also recommended by Sprott (2004) [96] for the precision we used.

4.3.2 Different Transient Times for the Lyapunov Exponent Calculation

We conducted tests discarding 100, 1000, and 10000 steps to get rid of transient effects due to random initializations. The results, shown in Fig. 23, indicated that there was no significant difference between a period of 1000 and 10000 steps. In order to cut down computational load, we then chose to wait for 1000 steps.

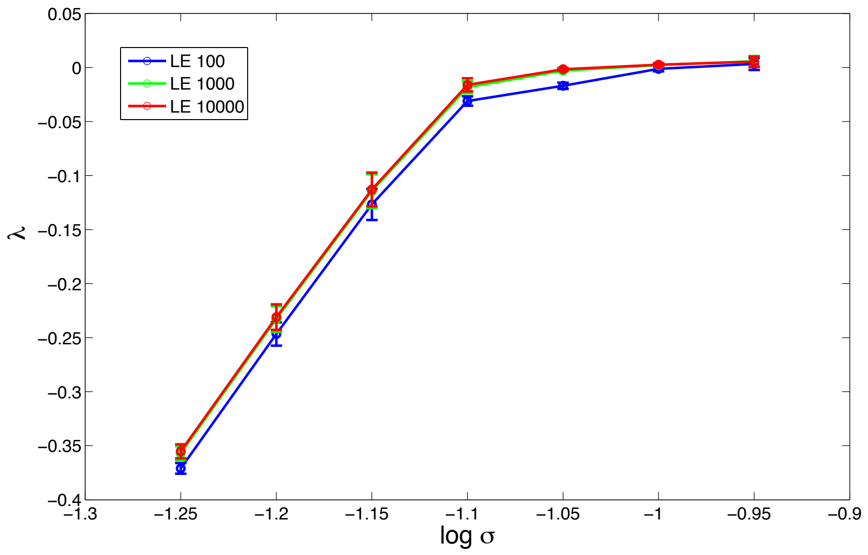


Figure 23: Testing different lengths for the period to wash out transient effects.

4.3.3 Tradeoff of Kernel Size versus History Length in the Transfer Entropy Calculations

Let N denote the length of the time series, r the kernel radius, k the history length, and n the average number of data samples in each bin. We take l as the difference between the smallest and the largest value of the range of possible values our network units can assume. Since this range is the interval $[-1, 1]$, $l = 2$. We require $n = 3$ which was recommended in [82] in order to get statistically meaningful results. The number of bins in each considered dimension of the data d is given by $d = l/r$. For the TE calculations, we have to consider the current step of the source variable, the previous step of destination variable, and the k last steps of the source. This means we have to search a space of d^{k+2} dimensions. With $n = N/d^{k+2}$, we establish the relation of r and k as

$$\left(\frac{l}{r}\right)^{k+2} = \frac{N}{2n} \tag{3}$$

The factor 2 in the denominator of the right hand side is due to the fact that r denotes a kernel radius, not a diameter. Fixing a history size, Eq. 3 can be used to compute the corresponding kernel radius. We tested different combinations of these two parameters for a time series length of $N = 1500$ with $n = 3$ as described above, and compared the results for the NARMA 30th order task ($\log \sigma$ was varied from -1.2 to -0.9 in steps of 0.1, 15 repetitions). The combinations we tested were $k = 1$ and $r = 0.252$, $k = 2$ and $r = 0.423$, $k = 3$ and $r = 0.577$. The results of this test are shown in Fig. 24. All of the plots show the same major qualitative trends in the data.

We finally opted for a history size $k = 2$, and a kernel size of $r = 0.2$ in our experiments with the time series of length 15000. Our reasoning was that this would make it possible to distinguish a maximum of 10 different levels of quantization in the $[-1, 1]$ range. This is a somewhat arbitrary criterion, but the plots in Fig. 24 tell us that the results should be robust concerning relative values nonetheless.

4.3.4 Transfer and Memory are Maximized at the Edge of Chaos

The results of the experiments described above are shown in Fig. 25 (top) for the MC task, and in Fig. 25 (bottom) for the NARMA modeling task. For each value of $\log \sigma$, the simulations were repeated 50 times (the clusters that can be observed in the figures are the result of slightly different LE values for each of these repetitions). The MC performance in Fig. 25 (top) shows a lot of variance, but a general increase can be seen as the LE approaches the critical value zero. After peak performance is reached very close to this point, the performance drops rapidly. The performance in the NARMA task does not show as much variation. The NRMSE stays around 0.8 for LE values from -0.9 to -0.4. As the LE approaches zero, the NRMSE decreases from around 0.5 to its lowest value of 0.4125 at LE -0.081. Shortly after that, however, as the LE approaches zero even more closely, the NRMSE increases sharply and reaches values as high as 142 (LE -0.011). After this peak, the NRMSE values stay at an increased level of about 2.

In order to arrive at a single value for the TE and AIS per reservoir, we took averages over all the nodes in the reservoir. The TE plots in Fig. 26 and AIS plots in Fig. 27 show very similar behavior for both tasks. Both TE and AIS can hardly

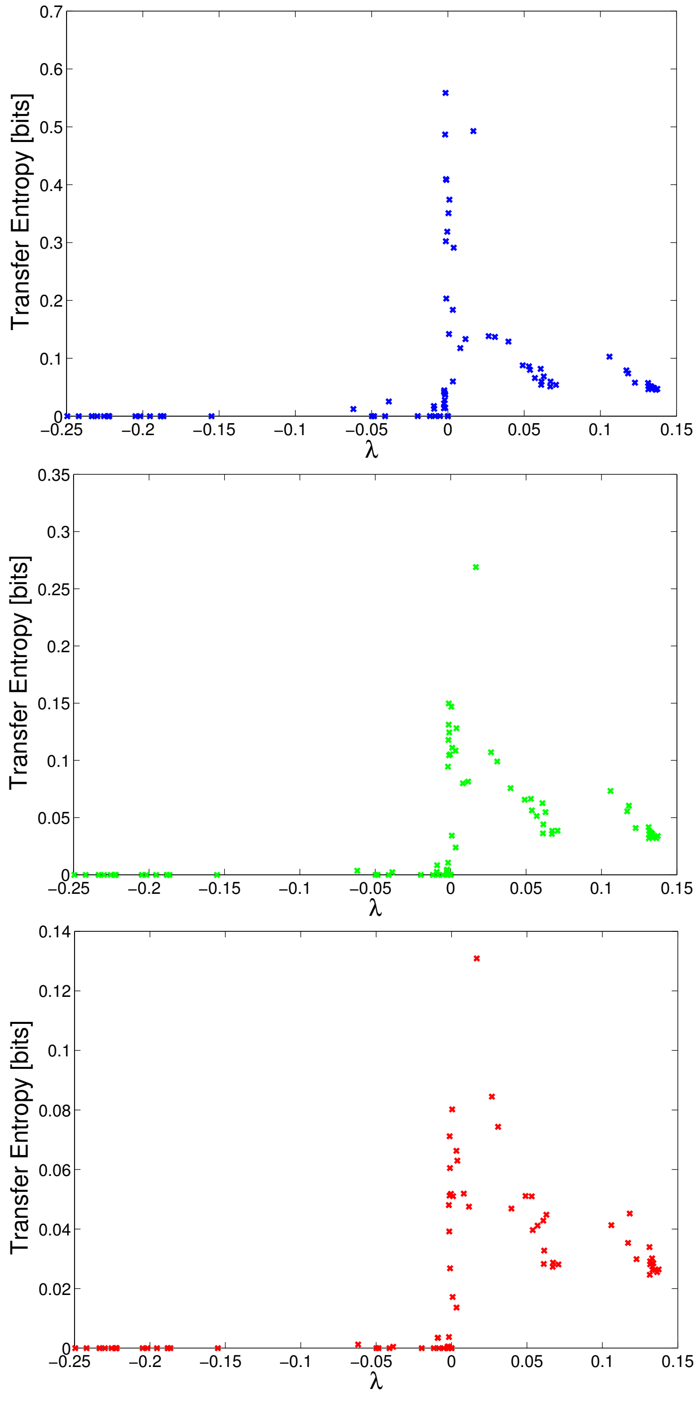
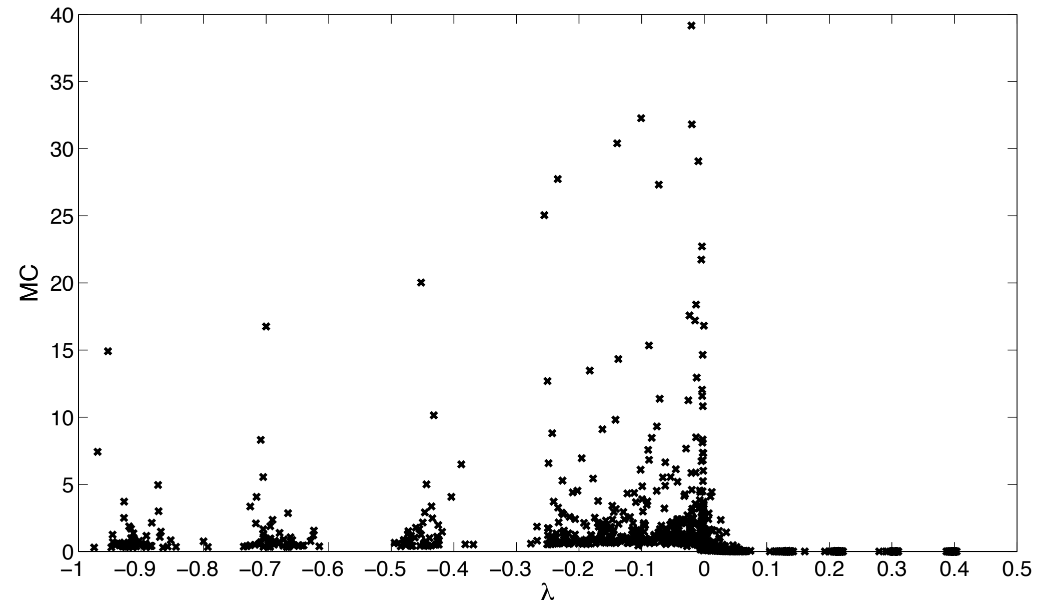


Figure 24: Testing different combinations of history length and kernel radii. (top) $k = 1$ and $r = 0.252$ (middle) $k = 2$ and $r = 0.423$ (bottom) $k = 3$ and $r = 0.577$

be measured for LE values below -0.2. Around the critical point, however, there is a sharp increase in TE/AIS, followed by a sharp decline between LE values 0 and about 0.05. Both quantities stay at a slightly elevated level compared to the values in the stable regime after that, decreasing only slowly.

In the chapter that follows, we will present another reservoir optimization approach. It uses the information-theoretic measure of transfer entropy, presented above, as a criterion for adaptation of a local memory parameter of each reservoir neuron.



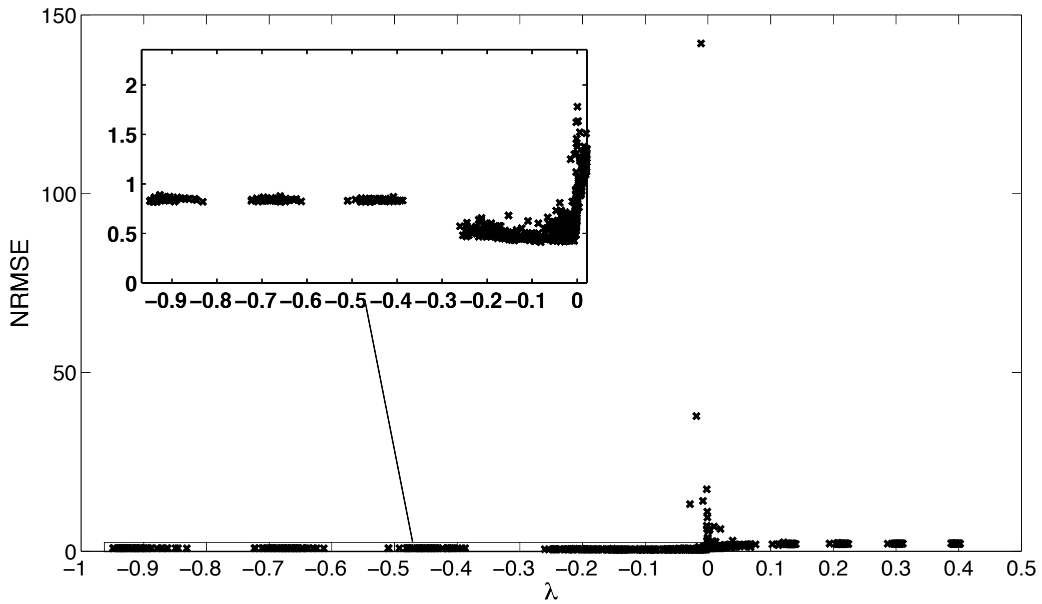
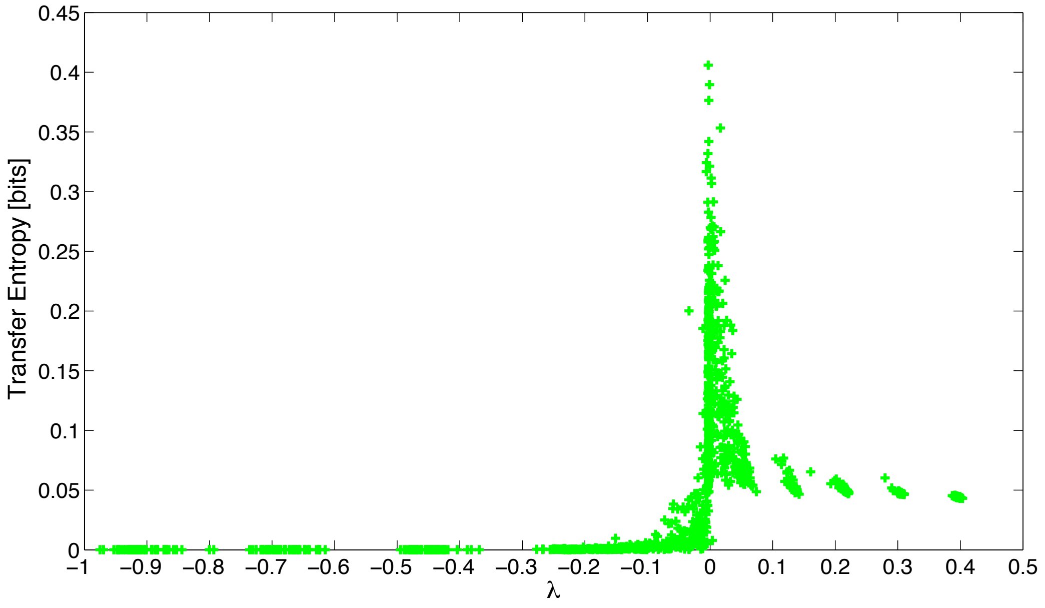


Figure 25: (top) Memory capacity vs estimated Lyapunov exponent (bottom) Normalized root mean squared error (NRMSE) vs estimated Lyapunov exponent



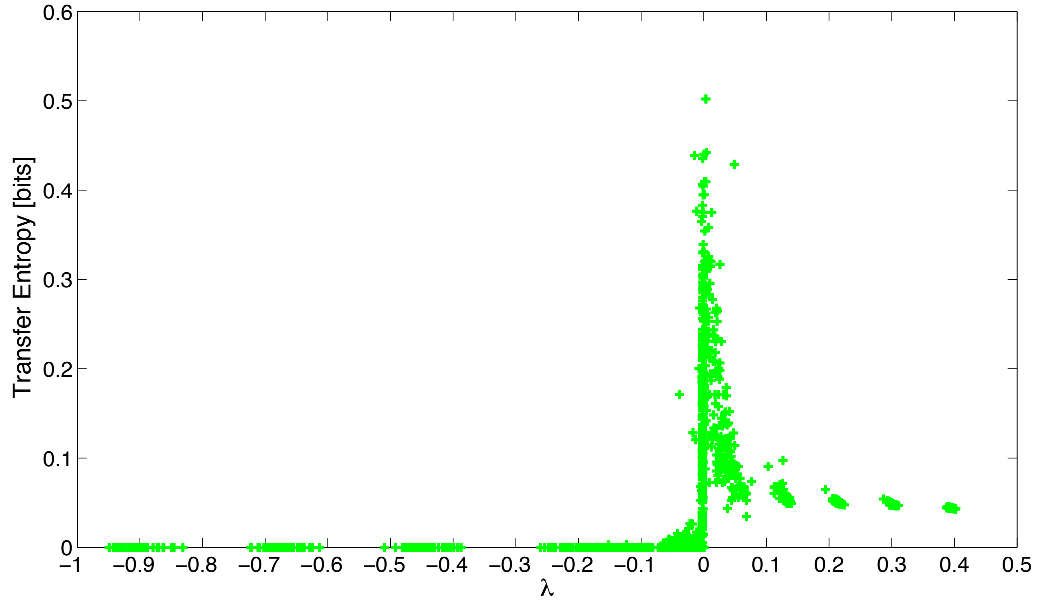
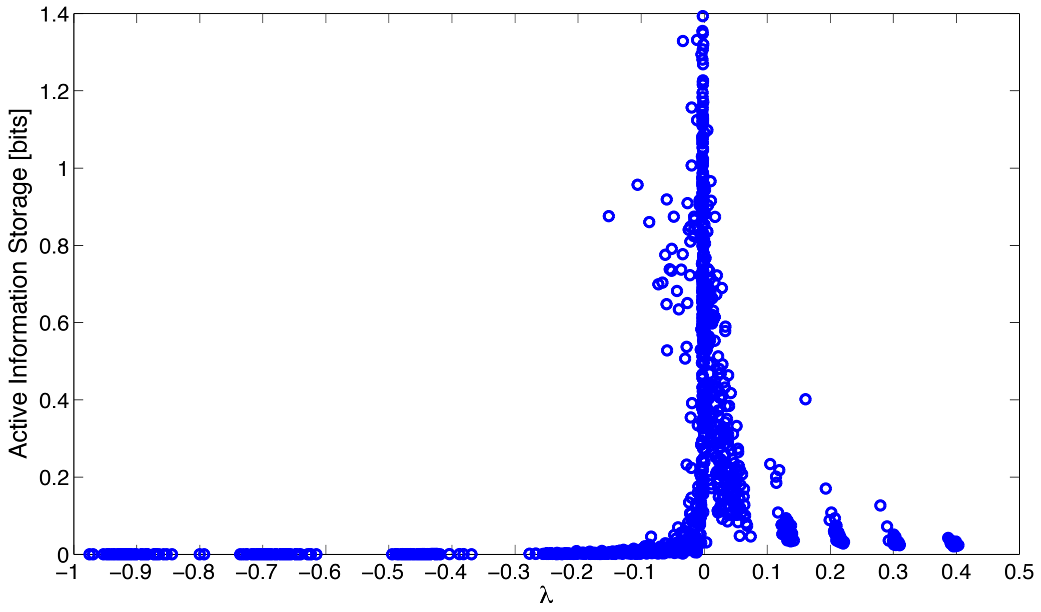


Figure 26: (top) Average transfer entropy in the reservoir for the memory capacity task vs estimated Lyapunov exponent (bottom) Average transfer entropy in the reservoir for the NARMA task vs estimated Lyapunov exponent



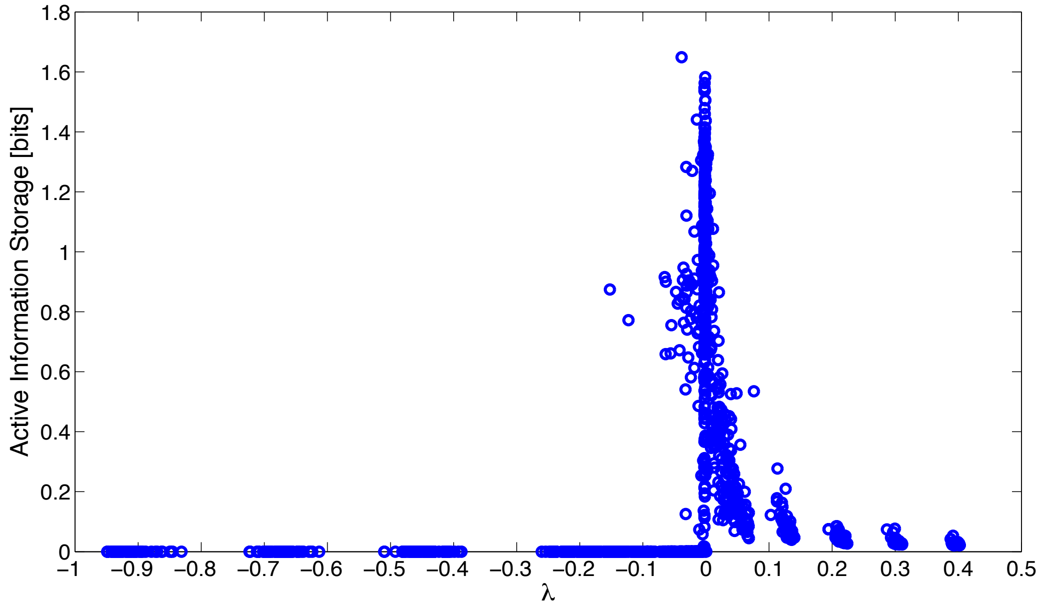


Figure 27: (top) Average active information storage in the reservoir for the memory capacity task vs estimated Lyapunov exponent (bottom) Average active information storage in the reservoir for the NARMA task vs estimated Lyapunov exponent

Chapter 5

Improving Recurrent Neural Network Performance Using Transfer Entropy

IP learning [15, 16], as discussed in Chapter 3, has been used as an approach to optimize reservoir encoding specific to the input of the network. It is, however, only dependent on the input data, and does not take the desired output of the system into account, i.e., it is not guaranteed to lead to optimized performance with respect to the learning task of the network. Ideally, we would like to retain the principle of a self-organized approach to optimize reservoirs, but to guide self-organization based on the overall learning goal.

The approach presented in this chapter, for the first time, leads to a method that optimizes the information transfer at each individual unit, dependent on properties of the information transfer between input and output of the system. The optimization is achieved by tuning self-recurrent connections, i.e., the means to achieve this optimization can be viewed as a compromise between Hebbian [97] and IP learning. Using synthetic data, we show that this reservoir adaptation improves the performance of offline echo state learning, and is also suitable for online learning approaches like backpropagation-decorrelation learning [14] or recursive least squares (RLS, see e.g. [36]).

We first define the network dynamics and output weight training in the following section, then explain the adaptation procedure based on the information transfer, and finally present experimental results in two different benchmarks.

5.1 Network Dynamics and Output-Weight Training Procedure

We consider the case where we have a one-dimensional input vector $\mathbf{u}$. The learning goal for our system is a one step-ahead prediction of a one-dimensional output vector $\mathbf{v}$. Departing from the usual reservoir dynamics described in Appendix A, we use

$$\mathbf{x}(k+1) = diag(\mathbf{a})\mathbf{W}\mathbf{y}(k) + (\mathbf{I} - diag(\mathbf{a}))\mathbf{y}(k) + \mathbf{w}^{in}\mathbf{u}(k) \tag{4}$$
$$\mathbf{y}(k+1) = \mathbf{f}(\mathbf{x}(k+1)), \tag{5}$$

where $x_i, i = 1, \ldots, N$ are the neural activations, $\mathbf{W}$ is the $N \times N$ reservoir weight matrix, w^{in} the input weight, $\mathbf{a} = [a_1, \ldots, a_N]^T$ a vector of local decay factors, $\mathbf{I}$ is the identity matrix, and k the discrete time step. In this work, we use $f(x) = \tanh(x)$. The a_i represent a decay factor, or coupling of a unit's previous state with the current state; they are computed as:

$$a_i = \frac{2}{1+m_i},$$

where m_i represents the memory length of unit i ($m_i \in \{1, 2, 3, \ldots\}$). All memory lengths are initialized to $m_i = 1$, so that $a_i = 1$, i.e. the reservoir has the usual update rule. Increasing individual m_i during an adaptation will increase the influence of a units past states on its current state.

The output weights of the network are trained using the Recursive Least Squares (RLS) algorithm. The RLS update rule can be described with the following set of equations:

$$\alpha_t = \mathbf{d}_t - \mathbf{w}^{out}_{t-1} \cdot \mathbf{x}_t, \tag{6}$$
$$\mathbf{g}_t = \mathbf{p}_{t-1} \cdot \mathbf{x}_t / (\lambda + \mathbf{x}_t^T \cdot \mathbf{p}_{t-1} \cdot \mathbf{x}_t), \tag{7}$$
$$\mathbf{p}_t = (\mathbf{p}_{t-1} - \mathbf{g}_t \cdot \mathbf{x}_t^T \cdot \mathbf{p}_{t-1})/\lambda, \tag{8}$$
$$\mathbf{w}^{out}_t = \mathbf{w}^{out}_{t-1} + (\alpha_t \cdot \mathbf{g}^T), \tag{9}$$

where α_t represents the *a priori error* vector between desired output $\mathbf{d}_t$ and current input, $\mathbf{p}_t$ the inverse of the autocorrelation, and λ is close to 1 and is an exponential forgetting factor. RLS has been applied to ESN learning in [36].

5.2 Adaptation of Information Transfer

Adaption of the reservoir to the learning goal introduces two extra steps to the learning procedure. In a first step, we determine the required history size l to maximize the information transfer from input u to output v, i.e. a first idea may be to look for a value

$$l_{max} = \arg\max_{l} T_{\mathbf{u}\to\mathbf{v}}(1, l).$$

Using increasingly larger history sizes may, however, always increase the transfer entropy (by possibly smaller and smaller values). To optimize the information transfer, we will instead be looking for the smallest value $\hat{l}$ that does not increase the transfer entropy $T_{u\to v}(1, \hat{l}-1)$ by more than a threshold ϵ, i.e.

$$T_{\mathbf{u}\to\mathbf{v}}(1, \hat{l}+1) \leq T_{\mathbf{u}\to\mathbf{v}}(1, \hat{l}) + \epsilon \quad \text{and} \tag{10}$$

$$T_{\mathbf{u}\to\mathbf{v}}(1, l) > T_{\mathbf{u}\to\mathbf{v}}(1, l-1) + \epsilon \quad \text{for all } l < \hat{l}. \tag{11}$$

From this first step, we learn the contribution of the size of the input history to the desired output (the learning goal of the system): some input-output pairs may require a larger memory of the input history to be informative about the next output state, other outputs may be more dynamic, and be dependent on the current input state only.

We take this information into the second step, which consists of a pre-training of the reservoir. Here, the local couplings of the reservoir units are adapted so that the transfer entropy from the input of each unit to its respective output is optimized *for the particular input history length* $\hat{l}$. The idea behind this step is to locally adjust the memory at each unit to approximate the required memory for the global task of the system. Pre-training is done in epochs of length ℓ over the training data. Over each epoch θ, we compute, for each unit i, the transfer entropy from activations $x_i^{(\ell)}$ to output $y_i^{(\ell)}$:

$$te_i^{\theta} = T_{x_i^{(\ell)}\to y_i^{(\ell)}}(1, \hat{l}).$$

If the information transfer during the current epoch θ exceeds the information transfer during the past epoch by a threshold (i.e., $te_i^{\theta} > te_i^{\theta-1} + \epsilon$), the local memory length m_i is increased by one. Likewise, if $te_i^{\theta} < te_i^{\theta-1} - \epsilon$, the local memory length is decreased by one, down to a minimum of 1. After each epoch, all m_i and a_i are adapted according to this rule, and used to compute activations over the next epoch. Once the training data is exhausted, pre-training of the reservoir is finished and the a_i are fixed. For the subsequent training we compute the output weights by linear regression with data as used in the pre-training. In additional experiments, we use RLS online learning, where adaptation and training of output weights were run in the same loop.

5.3 Experimental Results

We tested our method using a one-step ahead prediction of unidirectionally coupled maps, and a one-step ahead prediction of the Mackey-Glass time series as described in Section 3.1.3.

5.3.1 Prediction of Autoregressive Coupled Processes

As first experiments we studied our approach using a one-step ahead prediction of two unidirectionally coupled autoregressive processes:

$$u_{t+1} = 0.7u_t + 0.7\cos(0.3t) + n_t^x(0,\sigma^2) \text{ and} \tag{12}$$

$$v_{t+1} = 0.7v_t + eu_{t-\omega+1} + n_t^y(0,\sigma^2)\,, \tag{13}$$

where the parameter $e \in [0,1]$ regulates the coupling strength, $\omega \in \{0,1,2,\ldots\}$ an order parameter, and $n_t^x(0,\sigma^2)$ and $n_t^y(0,\sigma^2)$ are independent Gaussian random processes with zero mean and standard deviation $\sigma = 0.4$. For each trial, we generated time series $\mathbf{u}$ and $\mathbf{v}$ (random initial conditions; time series divided into 10000 values for training and 1200 values for testing; the first 200 values of both training and testing were used to prime the reservoir), where the task of our system was a one-step ahead prediction of $\mathbf{v}$ using $\mathbf{u}$. The reservoir was initialized using a random, sparse recurrent weight matrix ($|\lambda| = 0.95$), with 40 internal units. Figure 28 (left)

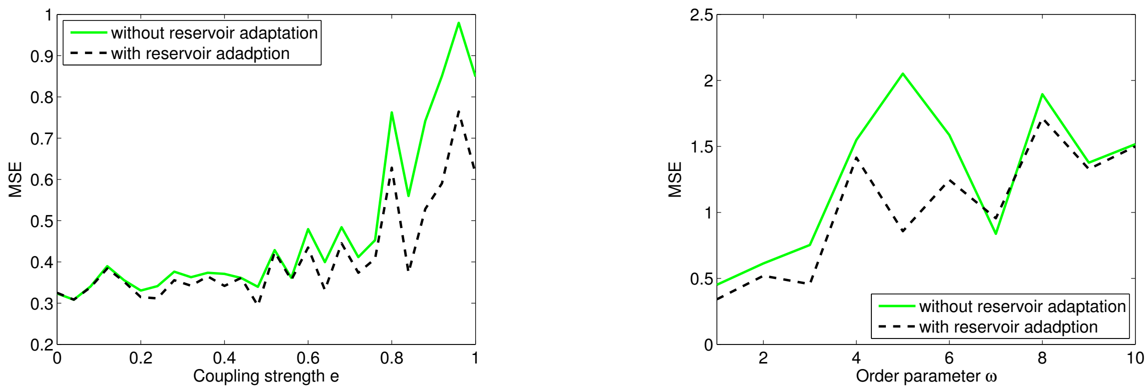


Figure 28: (left) mean squared errors of the prediction over the test data for different coupling strengths and fixed $\omega = 0$. (right) mean squared error for different ω using a fixed coupling of $e = 0.75$. Reported results are averages over 50 runs.

displays the mean square errors of the prediction over the test data for different coupling strengths and fixed $\omega = 0$ for both echo state learning with and without adaptation of information transfer in the reservoir. All values are averaged over 50 trials; for each individual trial the same reservoir and time series have been used once with and without adaptation. The prediction using the reservoir adaptation is better over almost the entire range of e, with the improvement becoming more significant as the influence of the input time series becomes larger. Figure 28 (right) is a plot of the mean square error for different ω using a fixed coupling of $e = 0.75$. In all but one cases the reservoir adaptation improves results.

5.3.2 Prediction of Mackey-Glass Time Series

A further experiment was prediction of the widely used Mackey-Glass time series (see e.g. [35, 55, 16]) with parameter τ set to 17. The first task using this time series was again a one-step ahead prediction using a reservoir size of 40 units. For this task, the transfer entropy between input and output time series is maximized already for smaller values of l compared to our first experiment (l was typically around 2 for Mackey-Glass one-step ahead prediction), i.e., the information used from the previous state to predict the next state is already quite high. The reservoir adaptation lead to an average improvement of the MSE (averaged over 50 runs) from $0.4530 \cdot 10^{-6}$ to $0.0751 \cdot 10^{-6}$. Individually, in 48 of the 50 runs, the same reservoir performed better with adaptation than without adaptation.

Instead of offline learning, we also used RLS in the same loop with our reservoir adaptation. To consider data from earlier stages of the adaptation less, we used a forgetting factor $\lambda = 0.995$. Again, the adaptation improved performance, from $9.1 \cdot 10^{-6}$ to $7.2 \cdot 10^{-6}$; a fine-tuning of λ may further improve the results.

After the presentation of new results concerning reservoir optimization and assessment in this and the two preceding chapters, we will discuss these results, their implications, and their relation to other work in the literature in the following chapter.

Chapter 6

Discussion

Compared to “traditional” recurrent neural network learning methods such as BPTT or RTRL, the reservoir computing paradigm represents an important simplification, and therefore, a significant step forward for recurrent neural network technologies. On the other hand, it is clear that the approach gives up many of the degrees of freedom the networks would normally have by fixing the recurrent layer connectivity. Advanced methods such as LSTM networks [32] share the advantage of fast learning with ESN, but without restricting the networks to a fixed connectivity, using a more involved architecture and training method. For ESN, it has been shown that fixing the connectivity has the effect that different random initializations of a reservoir will lead to rather large variations in performance if all other parameters of the network setup remain the same [38]. There have been proposals on how to manually design ESN in order to give performance that will consistently be better than random initializations [38], but there are no universally accepted standard training algorithms to adapt the connectivity in a problem specific and automatic way, before the output connections are trained.

Below, we discuss the results of our investigations of permutation matrices for reservoir connectivity, IP adaptation aiming at Laplace output distributions for the reservoir neurons, the limitation of IP with standard sigmoid units that we found, the results on the intrinsic computational capabilities of ESNs at the edge of chaos, and our reservoir optimization based on transfer entropy.

6.1 Permutation Matrices as a Computationally Inexpensive Initialization Method

The benefit of permutation matrices for the reservoir connectivity lies in the simple and inexpensive way in which they can be constructed. We showed that they implement a very effective connectivity for problems involving a long input history as well as non-linear mappings (if their spectral radius is set high enough). For problems requiring a mixture of slow and fast responses, their usefulness is limited. Furthermore, the method is general and not problem-specific.

6.2 Limitations of IP Learning for Standard Sigmoid Neurons

The IP based approaches we present and reference throughout Chapter 3 represent a problem-specific training method. These approaches make use of the input signal in order to shape the output of the reservoir neurons according to a desired probability distribution. We extended existing work, and derived a new learning rule to shape the reservoir node outputs according to a Laplace distribution. Moreover, we pointed out limitations of this method when standard sigmoidal neuron types are used in the network.

Concerning this last point, the illustration in Fig. 29 sheds light on the reason why an approximation of some distributions with IP is more difficult than others: given a uniform input distribution and a sigmoid transfer function, IP learning selects a slice from an output distribution that peaks towards either end of the input range, but never in the center. The output of an IP trained self-recurrent unit gives an insight why it is possible to achieve a Gaussian output distribution in a reservoir (Fig. 17, b). The *central limit theorem* from the field of statistics states that the sum of several i.i.d. random variables approximates a Gaussian. Even though in case of a recurrent reservoir not all inputs to a unit will be i.i.d., IP has to make input distributions only similar to each other to approximate a normal distribution in the output. This also explains why the output of an IP Laplace trained reservoir is normally distributed (several inputs with equal or at least very similar distributions are summed up). For uniform input and a single unit without recurrence, the best IP can do is to choose the linear part of the activation function, so that the output is also uniformly distributed (a slice more in the middle of Fig. 29). With self-recurrent connections, this leads to initially uniform distributions, which sum up. The resulting output, and eventually the whole reservoir output distribution become more and more Gaussian. A consequence of this effect is that IP with sigmoid transfer functions cannot be generalized to arbitrary distributions.

These results are in agreement with two important points that have been suggested for versatile networks, i.e., networks which should perform well even when faced with several different input signals or which might be used for tasks with different requirements. Ozturk et al. [38] proposed that the eigenvalue distribution of reservoir matrices should be as uniform as possible, and that it would be needed to scale the effective spectral radius of the network up or down. For this scaling, they suggested an adaptable bias to the inputs of each reservoir node. With regard to this proposed requirement, we observed another limitation of purely IP-based reservoir pre-training: in our experiments (also reported in [16]), the IP learning rule always increased the spectral radius of the reservoir matrix (dependent on the setting of the IP parameters, cf. [46]), and never decreased it (this is only true for reservoirs which are initialized with spectral radius < 1). This leads to longer memory, making it harder for the network to react quickly to new input, and causing interference of slowly fading older inputs with more recent ones. To alleviate this problem, a combination of different plasticity mechanism as studied in [48] seems promising. Interestingly, the authors of that study note that the combination of the different plasticity mechanisms can drive network dynamics away from the order-chaos phase transition region (which was hypothesized to be the region of best computational capabilities), yet showing improved performance for tasks with structure in the input data.

This brings us to the discussion of the part of the thesis which examined precisely this region (Chapter 4). The conjecture that computational performance of dynamical systems is maximized at the edge of chaos can be traced back at least to [40], and a significant number of works have addressed this issue (cf. our presentation in Section 2.3). A number of quantitative studies, including those mentioned in Chapter 2,

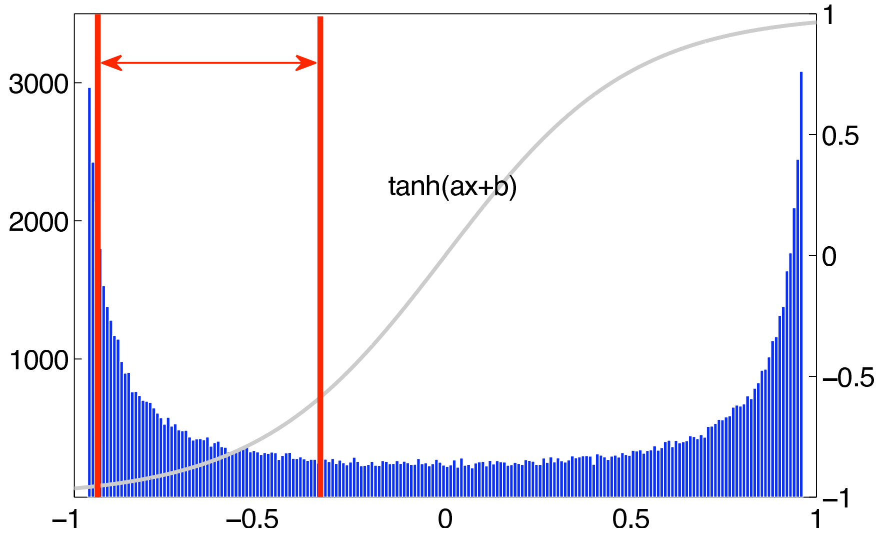


Figure 29: Uniform input on the interval $[-1; 1]$ and a $\tanh(\cdot)$ transfer function lead to the output distribution in the histogram. IP selects a slice of this distribution, as illustrated by the vertical lines. Adapting gain and bias changes width and position of the slice.

have been presented and have helped to elucidate the mechanisms underlying this maximization of computational performance. We adopted a more general framework in Chapter 4 and at the same time are able to measure the elements contributing to ongoing computation more directly and in a more localized fashion.

6.3 Reservoirs and Intrinsic Computational Capabilities

By investigating the information dynamics of ESN reservoirs, this thesis provides new insights into the problem of relating computation in recurrent neural networks to elements of Turing universal computation – information transfer and information storage. Our motivation for this study was to explore why tuning the ESN reservoir to the edge of chaos here produces optimal network performance for many tasks. Certainly, we confirmed previous results [42, 5] which have shown that performance peaks at the edge of chaos (for the MC task in our case). We then quantitatively showed, using an information-theoretic approach, that this is due to maximized computational properties (information storage and transfer) near this state. This also indicates that information transfer and information storage are potential candidates to guide self-organized optimization for the studied (and maybe other) systems (see, however, the points below).

6.4 Relation to Information Dynamics in Random Boolean Networks

Our results for these information dynamics through the phase transition in ESNs are similar to previous observations of these dynamics through the order-chaos phase transition in Random Boolean Networks (RBNs) [7]. A distinction however is that in the RBNs study, the information storage was observed to be maximized slightly on the ordered side of the critical point and the information transfer was maximized slightly on the chaotic side of the critical point (see Fig. 30). This is in contrast to our results here, where both maximizations appear to coincide with criticality.

Both results, however, imply maximization of computational properties *near* the critical state of the given networks. The similarity of the results seems natural on one hand (given similar descriptions of the phase transitions in both systems), but on the other hand these two types of networks are quite different. Here, we used analog activations and connections, whereas RBNs have discrete connections and binary states (supported by Boolean logic). Also, our networks are input driven, and RBNs (in Lizier et al. [7]) are not. Since we know that the transition from binary to analog networks can change system dynamics to a very large degree [5], the similarity in results across these network types is intriguing. The implications are quite interesting also, since relevant natural systems in each case are suggested to operate close to the edge of chaos (gene regulatory networks for RBNs, and cortical networks here).

We must place a number of caveats on these results however. Certainly, the computational capability of the network will be dependent on the input, and we will not find universal behavior through the order-chaos phase transition.

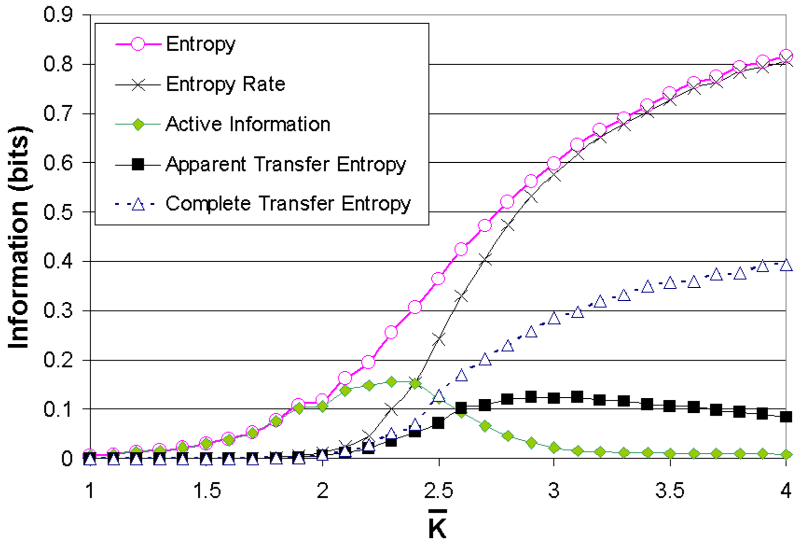


Figure 30: Average information dynamics in RBNs in relation to average connectivity $\bar{K}$ (from [7].)

6.5 Edge of Chaos Is Not Universally Beneficial for Computation

We also note that the network is always performing some computation, and does not need to be at the critical state to do so. While the critical state may maximize computational capabilities, the given task may require very little in terms of computation. For these reasons, it is known that systems do not necessarily evolve the edge of chaos to solve computational tasks [41]. Moreover, neural networks are applied to a large variety of different tasks, and certainly not all of them will benefit from networks close to criticality. For instance, training a network for fast input-induced switching between different attractors ("multiflop" task) is known to work best with reservoirs whose spectral radius is small, i.e. those on the very stable side of the phase transition [cf. 35, Sect. 4.2]. Instead of a long memory, this tasks requires the networks to react quickly to new input. We also see that the networks in the NARMA task show best performance slightly before the phase transition, while performance is actually worst right at the measured edge of chaos. A possible explanation for this might be that the memory in the network actually gets *too* long. The networks in this task need access to the last 30 inputs in order to compute a correct output, but if information stays in the reservoir from inputs older than 30 steps, it might interfere with the ongoing computation. Fig. 25(left) for the memory capacity task supports this to some extent, showing that memory capacity reaches values in excess of 30 around the critical point. As mentioned above, Lazar et al. [48] present evidence that RNNs with reservoirs shaped through a combination of different plasticity mechanism (IP, synaptic scaling, and a simple version of spike timing dependent plasticity) actually drive the network *further away* from the critical region while outperforming networks with fixed random reservoirs close to that region, at least for the task they tested (predicting the next character in a sequence).

6.6 No one-to-one Relation Between Intrinsic Computation and Task Performance

Performance on the two tasks we studied shown in Fig. 25 is still quite good while the network remains in the ordered regime, even though storage and transfer are not measured to be very high here. This suggests that much of the storage and transfer we measure in the reservoir is not related to the task – an interesting point for further investigation. The effect of different reservoir sizes on the computational capabilities may be interesting to investigate: while the memory capacity increases with the number of reservoir units, the prediction of some time series will only require a finite amount of memory. Adjusting the reservoir size to the point so that the reservoir is exactly large enough for the given task and data may produce networks where the computational capabilities are only dedicated to the task at hand.

We emphasize that our main finding is that information storage and transfer are maximized near the critical state, regardless of the resulting performance. Indeed, there is certainly not a one-to-one correspondence between either of the information dynamics measures and task performance. We also note the results of Lizier et al. [98], showing that maximizing these functions in other systems does not necessarily lead to complex behavior. Therefore, we see our results as a promising starting point for an understanding of the individual computational properties of ESN nodes.

6.7 Information Transfer Between Input and Desired Output Can Be Used To Successfully Guide Reservoir Adaptation

The information-theoretic approach to reservoir optimization presented in Chapter 5 uses a local adaptation of a units internal state, based on properties of the information transfer between input and desired output of the system. The approach could improve performance in conjunction with offline echo state regression, as well as with RLS online learning. In our experiments we have used only a small number of internal units – our goal was to show the capability of our approach compared to standard echo state learning. In first additional experiments (not reported here), we found that for a larger number of units our adaptation leads to an even larger improvement compared to echo state learning without adaptation. In conclusion, we think that this approach to quantify the computational properties of the *task* (as opposed to just the computational capabilities of the *reservoir*) may be a promising approach to shed light on the reasons for the gap between task performance and intrinsic computational capabilities mentioned above.

In the following last chapter of this thesis, we will now summarize achieved results and contributions, and we will point to some directions for improvements and extensions in further research.

Chapter 7

Conclusion and future work

7.1 Summary and Contributions

In this thesis, we presented new results which contribute to the fundamental research question of how Echo State Network reservoirs can be reliably improved over standard methods, and how to assess reservoir quality in terms of intrinsic computation. We presented and compared different approaches for reservoir improvement on a variety of benchmarks. In the process, we found a fundamental limitation of one of the approaches, which is widely used in the research community. We also investigated the phenomenon of increased task performance at the order-chaos phase transition in Echo State Networks. Using the information dynamics framework, we quantified the elements of intrinsic computation in the reservoir, and measured how they change at the critical point.

In summary, the following contributions have been made in this thesis:

1. **More systematic investigation of permutation matrices as reservoir matrices using several benchmarks:** Permutation matrices had been proposed for the connectivity of ESN reservoirs before. They were found to be very suitable for predicting chaotic time-series, but a thorough study of their performance in several standard benchmarks had been missing. We included permutation matrices in our experimental comparison of reservoir shaping methods and found them to be very useful for tasks requiring long memory and highly

nonlinear mappings. On tasks with periodicity and dominant time scales in the input signal, permutation matrices were found to be of limited use.

2. **Derivation and testing of a new local learning rule for self-organized reservoir optimization:** Inspired by the intrinsic plasticity of biological neurons, this learning rule changes only internal parameters of each reservoir neuron (as opposed to connection weights between neurons). We built on previous work with the aim to achieve a balance between sparse activity (which is known to increase short-term memory capacity) in the reservoir, and maximization of the information for each reservoir neuron output (which helps to separate different inputs coming into the reservoir).

3. **Identification of a fundamental limitation of this widely used family of IP-based learning rules in the context of Echo State Networks:** While comparing the different reservoir optimization schemes, we noticed that the IP rules for Gaussian and Laplace output distributions both produced *only Gaussian* output distributions for the reservoir neuron activity. We could attribute this finding to the central limit theorem of statistics. This made clear that the previously published IP rule for Gaussian distributions only works in a network of neurons connected together, and because the central limit theorem works in its favor. It also prevents the IP rule for Laplace distributions from converging on the right distribution, revealing that more degrees of freedom in the adaptation of the transfer function, or different transfer functions altogether, will have to be used to achieve any desired output distribution.

4. **Quantification of the components of intrinsic computation in Echo State Networks reservoirs:** By measuring the active information storage and transfer entropy on the level of individual units of the reservoir, we were able to quantify the components of intrinsic computation locally and directly. We found that the intrinsic computational capability of ESNs in terms of these components is maximized as the networks approach the phase transition from ordered to chaotic dynamics, and it falls off sharply after that.

5. **More evidence that edge of chaos is not universally beneficial for task

performance: Evidence has been presented in the literature both in favor and against the conjecture that dynamical systems show best computational performance at the edge of chaos. Here, we showed that this is true for *intrinsic* computation, i.e., the general support of universal computation through elements such as memory and information transfer in ESNs. However, our results also indicated that there is not a direct mapping between intrinsic elements of computation and task performance (at least to the extend that we measured these elements). Therefore, we can not conclude that the edge of chaos should be the region where an ESN will always show the best performance for any given task.

6. **An Information-theoretic method to adapt Echo State Network reservoirs dependent on the task goal:** Our approach optimizes the information transfer at each individual unit, dependent on properties of the information transfer between input and output of the system. This leads to reliable performance improvement in the benchmarks we tested, both in connection with online and offline training of output connections.

7.2 Directions for future work

Below, we point out a number of directions how the results presented in this thesis could be used and extended in future work.

7.2.1 Reservoir Optimization

First steps towards using the information transfer to improve performance of reservoirs have been presented in Chapter 5. We tuned information transfer of individual units locally by adapting self-recurrence, dependent on the learning goal of the system. This could be extended, e.g. by using several plasticity mechanisms together with the goal of achieving a synergistic effect for the improvement of information transfer appropriate for a given task.

In our study in Chapter 3, permutation matrices proved to be very useful as reservoir connectivity for different tasks. We also saw, however, that all the poles in these matrices have the same radius in the complex plane. This means that the network implements a filter with only one time constant, depending on this radius, as opposed to uniform random matrices whose poles have radii of different magnitude resulting in filters with different time constants. An idea would be to use adaptive biases at each reservoir neuron as proposed in [38]. These biases are adapted based on an output error signal and allow the time constants to be adjusted for each pole individually, while leaving the desirable uniform distribution of the poles around the unit circle unaltered.

An interesting topic to study might be the effect of different ring sizes in permutation matrices on task performance. The different rings implement distributed delay lines of different lengths, and a mechanism to determine the best mixture of rings for a given task seems useful.

As mentioned briefly before, in order for the IP methods to generate arbitrary desired output distributions, transfer functions with more adjustable parameters than just bias and gain would be useful to study. For certain distributions, transfer functions of different shapes than the standard sigmoid ones might be needed. Furthermore, the simultaneous use of different plasticity rules for reservoir shaping (as demonstrated e.g. in [48]) is a very promising approach, and should be investigated further. It is important to understand their inter-play and mutual benefits.

In the field of robotics, there have been interesting applications of ESNs, e.g. for inverse kinematics and whole body motion learning of humanoid robots as mentioned in the introduction of this thesis. In studies such as [2] and [3], the IP rule has been used for reservoir optimization, and it was emphasized that it was necessary in order to achieve motions with long trajectories. It would be interesting to see how reservoirs at the edge of chaos (either hand-tuned or automatically guided to the critical point) would compare in this application.

7.2.2 Reservoir Assessment

A worthwhile extension of our work on quantifying the elements of intrinsic computation in ESN reservoirs would be to also measure information modification in addition to active information storage and transfer entropy. This measure was proposed in [70] and it might complete the picture of universal computation capabilities in ESNs. Furthermore, it might help to clarify the reasons for the gap between maximization of intrinsic computation performance, and observed task performance pointed out earlier. In addition, the information dynamics framework might be useful to gain insight into how the different plasticity mechanisms drive networks away from the edge of chaos in [48], but still achieve superior performance.

7.2.3 Reservoir Computing at the Edge of Chaos

Related to the point raised above on reasons for measuring information modification, we certainly need to investigate how to quantify and predict how much a network faced with a certain task will actually benefit from dynamics close to the edge of chaos. Again, the work presented in Chapter 5 could be a starting point for that, but certainly much work remains to be done.

Appendix A

Reservoir Computing

Recurrent loops are abundant in the neural circuits of the mammalian cortex. Massive reciprocal connections exist on different scales, linking different brain areas as well as connecting individual neurons in cortical columns. In these columns as many as 80% of the synapses of neocortical interneurons form a dense local network [9] using very specific connectivity patterns for different neuron types [8]. These recurrent microcircuits are very stereotypical and repeated over the entire neocortex [99].

Two challenges for computational models of the neocortex are (a) explaining how these stereotypical microcircuits enable an animal to process a continuous stream of rapidly changing information from its environment [37], and (b) how these circuits contribute to the prediction of future events, one of the critical requirements for higher cognitive function [9].

To address these challenges, a mathematical model for generic neural microcircuits, namely the *liquid state machine* (LSM), was proposed by Maass et al. [37]. The framework for this model is based on real-time computation without stable attractors. The neural microcircuits are considered as dynamical systems, and the time-varying input is seen as a perturbation to the state of the high-dimensional excitable medium implemented by the microcircuit. The neurons act as a series of non-linear filters, which transform the input stream into a high-dimensional space. These transient internal states are then transformed into stable target outputs by readout neurons, which are easy to train (e.g. in order to do prediction or classification of input signals) and avoid many of the problems of more traditional methods of recurrent neural network training like slow convergence and vanishing gradients (first described in Hochreiter [29], see also Bengio et al. [100] and Hochreiter et al. [27]). This approach to neural modeling has become known as *reservoir computing*, and the LSM is one particular kind of model following this paradigm.

Echo state networks (ESN) [35, 36] are another reservoir computing model similar to LSM. They implement the same concept of keeping a fixed high-dimensional *reservoir* of neurons, usually with random connection weights between reservoir neurons small enough to guarantee stability. Learning procedures train only the output weights of the network to generate target outputs, but while LSM use spiking neuron models, ESN are usually implemented with sigmoidal nodes, which are updated in discrete time steps.

In the following, we describe Echo State Networks more formally and in more detail. For information on other reservoir computing approaches, including LSMs, please refer to the reviews in [49] and [43]).

Echo State Networks

ESN provide a specific architecture and a training procedure that aims to solve the problem of slow convergence [35, 36] of earlier recurrent neural network training algorithms. ESN are normally used with a discrete-time model, i.e. the network dynamics are defined for discrete time-steps t, and they consist of inputs, a recurrently connected hidden layer (also called *reservoir*) and an output layer (see Fig. 31).

We denote the activations of units in the individual layers at time t by $\mathbf{u}_t$, $\mathbf{x}_t$, and $\mathbf{o}_t$ for the inputs, the hidden layer and the output layer, respectively. We use $\mathbf{w}^{in}$, $\mathbf{W}$, $\mathbf{w}^{out}$ as matrices of the respective synaptic connection weights. Using $f(x) = \tanh x$ as output nonlinearity for all hidden layer units, the network dynamics is defined as:

$$\mathbf{x}_t = \tanh(\mathbf{W}\mathbf{x}_{t-1} + \mathbf{w}^{in}\mathbf{u}_{t)}$$
$$\mathbf{o}_t = \mathbf{w}^{out}\mathbf{x}_t$$

The main differences of ESN to traditional recurrent network approaches are the setup of the connection weights and the training procedure. To construct an ESN, units in the input layer and the hidden layer are connected randomly. Connections

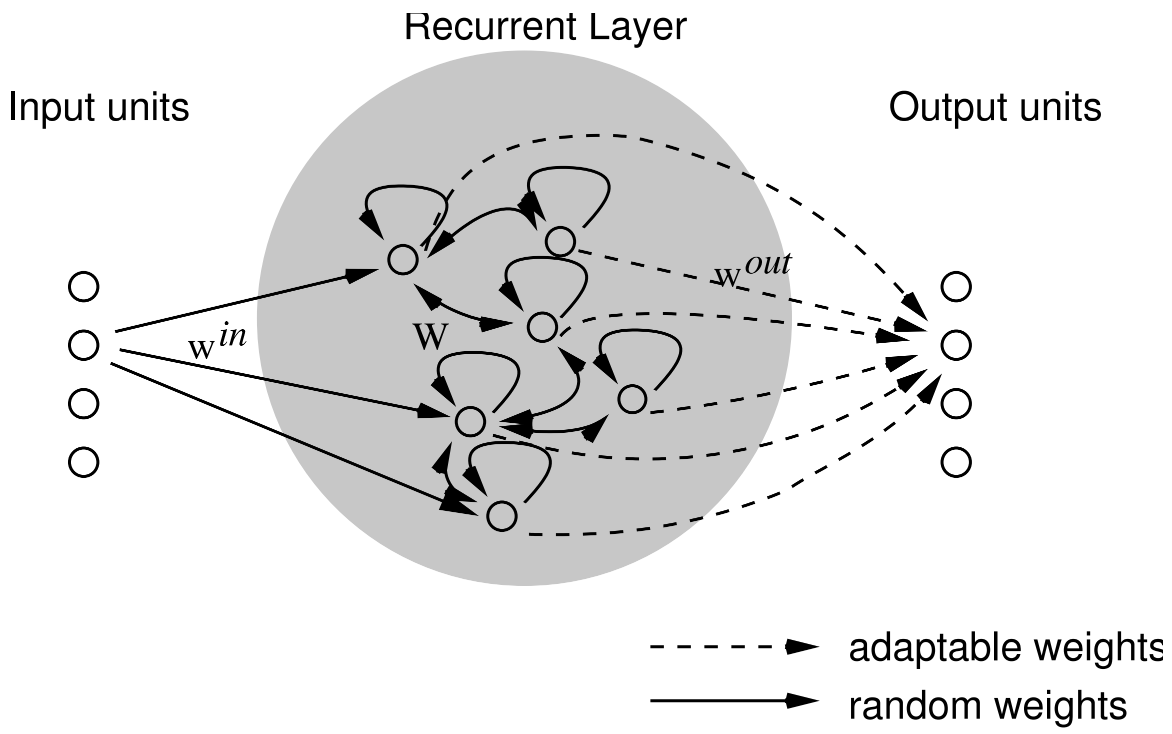


Figure 31: Architecture of an echo state network. In echo state networks, usually only the connections represented by the dashed lines are trained, all other connections are setup randomly and remain fixed. The recurrent layer is also called a *reservoir*, analogously to a liquid, which has fading memory properties. As an example, consider throwing a rock into a pond; the ripples caused by the rock will persist for a certain amount of time and thus information about the event can be extracted from the liquid as long as it has not returned to its single attractor state — the flat surface.

between the hidden layer and the output units are the only connections that are trained, usually with a supervised, offline learning approach using linear regression: Training data are used to drive the network, and at each time step t, activations of all hidden units $\mathbf{x}(t)$ are saved as a new column to a state matrix. At the same time, the desired activations of output units $\mathbf{o}_{\text{teach}}(t)$ are collected in a second matrix. Training in this approach then means to determine the weights $\mathbf{w}^{out}$ so that the error $\epsilon_{\text{train}}(t) = (\mathbf{o}_{\text{teach}}(t) - \mathbf{o}(t))^2$ is minimized. This can be achieved using a simple linear regression (see Jaeger [35] for details on the learning procedure).

For the approach to work successfully, however, connections in the reservoir cannot be completely random; ESN reservoirs are typically designed to have the *echo state property.* The definition of the echo state property has been outlined in Jaeger [35] and is summarized below.

The Echo State Property

Consider a time-discrete recursive function:

$$\mathbf{x}_{t+1} = F(\mathbf{x}_t, \mathbf{u}_{t+1}) \tag{14}$$

that is defined at least on a compact sub-area of the vector-space $\mathbf{x} \in R^n$, with n the number of internal units. The $\mathbf{x}_t$ are to be interpreted as internal states and $\mathbf{u}_t$ is some external input sequence, i.e. the stimulus.

Definition Assume an infinite stimulus sequence
$\bar{\mathbf{u}}^{\infty} = \mathbf{u}_0, \mathbf{u}_1, \ldots$, and two random initial internal states of the system $\mathbf{x}_0$ and $\mathbf{y}_0$. From both initial states $\mathbf{x}_0$ and $\mathbf{y}_0$ the sequences $\bar{\mathbf{x}}^{\infty} = \mathbf{x}_0, \mathbf{x}_1, \ldots$ and $\bar{\mathbf{y}}^{\infty} = \mathbf{y}_0, \mathbf{y}_1, \ldots$ can be derived from the update equation Eq. (14) for $\mathbf{x}_{t+1}$ and $\mathbf{y}_{t+1}$. If, for all right-infinite input sequences $\bar{\mathbf{u}}^{+\infty} = \mathbf{u}_t, \mathbf{u}_{t+1}, \cdots$ taken from some compact set U, for any $(\mathbf{x}_0, \mathbf{y}_0)$ and all real values $\epsilon > 0$, there exists a $\delta(\epsilon)$ for which $\|\mathbf{x}_t - \mathbf{y}_t\| \leq \epsilon$ for all $t \geq \delta(\epsilon)$ (where $\|\cdot\|$ is the Euclidean norm), the system $F(\cdot)$ will have the echo state property relative to the set U.

In simple terms, the system has echo state property if different initial states converge (for all inputs taken from U).

Appendix B

Review of Relevant Information-theoretic Quantities

The (Shannon) *entropy* is a fundamental measure that estimates the average uncertainty in a sample x of stochastic variable X. It is defined as

$$H_X = -\sum_x p(x)\log_2 p(x)$$

If a base two logarithm is used in this quantity as above, entropy is measured in units of bits.

The *joint entropy* of two random variables X and Y is a generalization to quantify the uncertainty of their joint distribution:

$$H_{X,Y} = -\sum_{x,y} p(x,y)\log_2 p(x,y).$$

The *conditional entropy* of X given Y is the average uncertainty that remains about x when y is known:

$$H_{X|Y} = -\sum_{x,y} p(x,y)\log_2 p(x|y).$$

The *mutual information* between X and Y measures the average reduction in uncertainty about x that results from learning the value of y, or vice versa:

$$I_{X;Y} = H_X - H_{X|Y}.$$

The *conditional mutual information* between X and Y given Z is the mutual information between X and Y when Z is known:

$$I_{X;Y|Z} = H_{X|Z} - H_{X|Y,Z}.$$

Bibliography

[1] Masato Ito and Jun Tani. On-line imitative interaction with a humanoid robot using a dynamic neural network model of a mirror system. *Adaptive Behavior*, 12(2):93–115, 2004.

[2] Matthias Rolf, Jochen J Steil, and Michael Gienger. Efficient exploration and learning of whole body kinematics. In *IEEE 8th International Conference on Development and Learning (ICDL 2009)*, Shanghai, CH, 06/2009 2009.

[3] Felix R. Reinhart and Jochen J Steil. Reaching movement generation with a recurrent neural network based on learning inverse kinematics. In *IEEE Conf. Humanoid Robotics*, pages 323–330, Paris, 2009. IEEE, IEEE.

[4] Nils Bertschinger and Thomas Natschläger. Real-time computation at the edge of chaos in recurrent neural networks. *Neural Computation*, 16(7):1413–1436, 2004. ISSN 0899-7667. doi: http://dx.doi.org/10.1162/089976604323057443.

[5] Lars Büsing, Benjamin Schrauwen, and Robert Legenstein. Connectivity, dynamics, and memory in reservoir computing with binary and analog neurons. *Neural Computation*, 22(5):1272–1311, 2010.

[6] Douglas Zhou, Yi Sun, Aaditya V. Rangan, and David Cai. Spectrum of lyapunov exponents of non-smooth dynamical systems of integrate-and-fire type. *Journal of Computational Neuroscience*, 28:229–245, 2010.

[7] Joseph T. Lizier, Mikhail Prokopenko, and Albert Y. Zomaya. The information dynamics of phase transitions in random boolean networks. In Seth Bullock, Jason Noble, Richard Watson, and Mark A. Bedau, editors, *Proceedings of the*

Eleventh International Conference on the Simulation and Synthesis of Living Systems (ALife XI), Winchester, UK, pages 374–381, Cambridge, MA, 2008. MIT Press.

[8] Rodney Douglas, Henry Markram, and Kevan Martin. Neocortex. In Gordon M. Shepard, editor, *The Synaptic Organization of the Brain*, pages 499–558. Oxford University Press, 5th edition, 2004.

[9] W. Maass and H. Markram. Theory of the computational function of microcircuit dynamics. In S. Grillner and A. M. Graybiel, editors, *Microcircuits. The Interface between Neurons and Global Brain Function*, pages 371–392. MIT Press, 2006.

[10] Danil V. Prokhorov. Toyota prius hev neurocontrol and diagnostics. *Neural Networks*, 21(2-3):458 – 465, 2008. ISSN 0893-6080. doi: DOI:10.1016/j.neunet.2007.12.043.

[11] Chia-Feng Juang, Shui-Tien Huang, and Fun-Bin Duh. Mold temperature control of a rubber injection-molding machine by tsk-type recurrent neural fuzzy network. *Neurocomputing*, 70(1-3):559 – 567, 2006. ISSN 0925-2312. doi: DOI:10.1016/j.neucom.2005.11.003.

[12] Ya-Fu Peng and Chih-Min Lin. Adaptive recurrent cerebellar model articulation controller for linear ultrasonic motor with optimal learning rates. *Neurocomputing*, 70(16-18):2626 – 2637, 2007. ISSN 0925-2312. doi: DOI:10.1016/j.neucom.2006.05.018.

[13] Jun Tani. Learning to generate articulated behavior through the bottom-up and the top-down interaction process. *Neural Networks*, 16:11–23, 2003.

[14] Jochen J. Steil. Backpropagation-decorrelation: Online recurrent learning with o(n) complexity. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, volume 1, pages 843–848, 2004.

[15] Jochen Triesch. A gradient rule for the plasticity of a neuron's intrinsic excitability. In *Artificial Neural Networks: Biological Inspirations – ICANN 2005*, pages 65–70. Springer, 2005.

[16] Jochen J. Steil. Online reservoir adaptation by intrinsic plasticity for backpropagation-decorrelation and echo state learning. *Neural Networks*, 20 (3):353–364, April 2007. ISSN 0893-6080. doi: http://dx.doi.org/10.1016/j.neunet.2007.04.011.

[17] Paul G. Plöger, Adriana Arghir, Tobias Günther, and Ramin Hosseiny. Echo state networks for mobile robot modeling and control. In Daniel Polani, Brett Browning, Andrea Bonarini, and Kazuo Yoshida, editors, *RoboCup 2003: Robot Soccer World Cup VII*, volume 3020 of *Lecture Notes in Computer Science*, pages 157–168. Springer Berlin / Heidelberg, 2004.

[18] Prashant Joshi and Wolfgang Maass. Movement generation with circuits of spiking neurons. *Neural Computation*, 17(8):1715–1738, 2005.

[19] Harald Burgsteiner. Training networks of biological realistic spiking neurons for real-time robot control. In *Proceedings of the 9th International Conference on Engineering Applications of Neural Networks*, pages 129–136, 2005.

[20] Masato Ito, Kuniaki Noda, Yukiko Hoshino, and Jun Tani. Dynamic and interactive generation of object handling behaviors by a small humanoid robot using a dynamic neural network model. *Neural Networks*, 19(3):323 – 337, 2006. ISSN 0893-6080. doi: DOI:10.1016/j.neunet.2006.02.007.

[21] Yuichi Yamashita and Jun Tani. Emergence of functional hierarchy in a multiple timescale neural network model: A humanoid robot experiment. *PLoS Comput Biol*, 4(11):e1000220, 11 2008. doi: 10.1371/journal.pcbi.1000220.

[22] Eric Antonelo, Benjamin Schrauwen, and Dirk Stroobandt. Modeling multiple autonomous robot behaviors and behavior switching with a single reservoir computing network. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, pages 1843–1848, 2008.

[23] Felix R. Reinhart and Jochen J Steil. Goal-directed movement generation with a transient-based recurrent neural network controller. In *Advanced Technologies for Enhanced Quality of Life*, pages 112–117. IEEE Computer Society, IEEE Computer Society, 2009.

[24] Matthias Rolf, Jochen J Steil, and Michael Gienger. Learning flexible full body kinematics for humanoid tool use. In *Int. Symp. Learning and Adaptive Behavior in Robotic Systems (Best Paper Award)*, Canterbury, UK, 09/2010 2010.

[25] Paul J. Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

[26] Ronald J. Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1(2):270–280, 1989.

[27] S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber. Gradient flow in recurrent nets: The difficulty of learning long-term dependencies. In S. C. Kremer and J. F. Kolen, editors, *A Field Guide to Dynamical Recurrent Neural Networks*. IEEE Press, 2001.

[28] S. Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(2):107–116, 1998.

[29] Sepp Hochreiter. Untersuchungen zu dynamischen neuronalen netzen. Diploma thesis, Technische Universität München, 1991.

[30] Barbara Hammer, Benjamin Schrauwen, and Jochen J. Steil. Recent advances in efficient learning of recurrent networks. In Michel Verleysen, editor, *Proceedings of the European Symposium on Artificial Neural Networks (ESANN)*, pages 213–226, 2009.

[31] B. Hammer, A. Micheli, N. Neubauer, A. Sperduti, and M. Strickert. Self organizing maps for time series. In *Proceedings of WSOM 2005*, pages 115–122, 2005.

[32] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, 1997. ISSN 0899-7667. doi: http://dx.doi.org/10.1162/neco.1997.9.8.1735.

[33] Jürgen Schmidhuber, Daan Wierstra, and Faustino J. Gomez. Evolino: Hybrid neuroevolution/optimal linear search for sequence learning. In Leslie Pack Kaelbling and Alessandro Saffiotti, editors, *IJCAI*, pages 853–858. Professional Book Center, 2005. ISBN 0938075934.

[34] Jürgen Schmidhuber, Daan Wierstra, Matteo Gagliolo, and Faustino Gomez. Training recurrent networks by evolino. *Neural Comput.*, 19(3):757–779, 2007. ISSN 0899-7667. doi: http://dx.doi.org/10.1162/neco.2007.19.3.757.

[35] Herbert Jaeger. The "echo state" approach to analysing and training recurrent neural networks. Technical Report 148, GMD – German National Research Institute for Computer Science, 2001.

[36] Herbert Jaeger and Harald Haas. Harnessing Nonlinearity: Predicting Chaotic Systems and Saving Energy in Wireless Communication. *Science*, 304(5667): 78–80, 2004. doi: 10.1126/science.1091277.

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

[38] Mustafa C. Ozturk, Dongming Xu, and José C. Príncipe. Analysis and design of echo state networks. *Neural Computation*, 19(1):111–138, 2007. ISSN 0899-7667. doi: http://dx.doi.org/10.1162/neco.2007.19.1.111.

[39] Stephen Wolfram. Universality and complexity in cellular automata. *Physica D*, 10:1–35, 1984.

[40] Chris G. Langton. Computation at the edge of chaos: phase transitions and emergent computation. *Physica D*, 42(1-3):12–37, 1990.

[41] Melanie Mitchell, Peter T. Hraber, and James P. Crutchfield. Revisiting the edge of chaos: Evolving cellular automata to perform computations. *Complex Systems*, 7:89–130, 1993.

[42] Robert Legenstein and Wolfgang Maass. Edge of chaos and prediction of computational performance for neural circuit models. *Neural Networks*, 20(3):323–334, 2007.

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

[44] Herbert Jaeger. Short term memory in echo state networks. Technical Report 152, GMD – German National Research Institute for Computer Science, 2001.

[45] Herbert Jaeger. Reservoir riddles: suggestions for echo state network research. In *Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN 2005)*, volume 3, pages 1460–1462, 2005.

[46] Benjamin Schrauwen, Marion Wardermann, David Verstraeten, Jochen J. Steil, and Dirk Stroobandt. Improving reservoirs using intrinsic plasticity. *Neurocomputing*, 71(7-9):1159–1171, 2008. doi: http://dx.doi.org/10.1016/j.neucom.2007.12.020.

[47] Andreea Lazar, Gordon Pipa, and Jochen Triesch. Fading memory and time series prediction in recurrent networks with different forms of plasticity. *Neural Networks*, 20(3):312–322, 2007.

[48] Andreea Lazar, Gordon Pipa, and Jochen Triesch. Sorn: a self-organizing recurrent neural network. *Frontiers in Computational Neuroscience*, 3(23), 2009. ISSN 1662-5188. doi: 10.3389/neuro.10.023.2009.

[49] David Verstraeten, Benjamin Schrauwen, Michiel D'Haene, and Dirk Stroobandt. An experimental unification of reservoir computing methods. *Neural Networks*, 20(4):391–403, 2007.

[50] David H. Wolpert. The supervised learning no-free-lunch theorems. In *In Proceedings of the 6th Online World Conference on Soft Computing in Industrial Applications*, pages 25–42, 2001.

[51] Herbert Jaeger. Tutorial on training recurrent neural networks, covering bptt, rtrl, ekf and the "echo state network" approach. GMD Report 159, German National Research Center for Information Technology, 2002.

[52] Benjamin Liebald. Exploration of effects of different network topologies on the esn signal crosscorrelation matrix spectrum. Bachelor's thesis, Jacobs University Bremen, 2004.

[53] Duncan J. Watts and Steven H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, 1998.

[54] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 10 1999.

[55] Márton Hajnal and András Lőrincz. Critical echo state networks. *Artificial Neural Networks – ICANN 2006*, pages 658–667, 2006.

[56] Oliver Obst. Distributed fault detection in sensor networks using a recurrent neural network. *CoRR*, abs/0906.4154, 2009.

[57] Yanbo Xue, Le Yang, and Simon Haykin. Decoupled echo state networks with lateral inhibition. *Neural Networks*, 20(3):365–376, 2007.

[58] Jochen J. Steil. Several ways to solve the mso problem. In *Proceedings of the 15th European Symposium on Artificial Neural Networks (ESANN 2007)*, pages 489–494, 2007.

[59] Daan Wierstra, Faustino J. Gomez, and Jürgen Schmidhuber. Modeling systems with internal state using evolino. In *GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation*, pages 1795–1802, New York, NY, USA, 2005. ACM. ISBN 1-59593-010-8. doi: http://doi.acm.org/10.1145/1068009.1068315.

[60] Magdalena Klapper-Rybicka, Nicol N. Schraudolph, and Jürgen Schmidhuber. Unsupervised learning in lstm recurrent neural networks. In *ICANN*, pages 684–691, 2001.

[61] W. Zhang and D. J. Linden. The other side of the engram: Experience-driven changes in the neuronal intrinsic excitability. *Nature Reviews Neuroscience*, 4: 885–900, 2003.

[62] G. Daoudal and D. Debanne. Long-term plasticity of intrinsic excitability: Learning rules and mechanisms. *Learning and Memory*, 10:456–465, 2003.

[63] Jochen Triesch. Synergies between intrinsic and synaptic plasticity mechanisms. *Neural Computation*, 19(4):885–909, 2007. ISSN 0899-7667 (Print). doi: 10. 1162/neco.2007.19.4.885.

[64] Anthony J. Bell and Terrence J. Sejnowski. An information-maximization approach to blind separation and blind deconvolution. *Neural Computation*, 7(6): 1129–1159, 1995. ISSN 0899-7667 (Print).

[65] Norbert M. Mayer and Matthew Browne. Echo state networks and self-prediction. In Auke Jan Ijspeert, Masayuki Murata, and Naoki Wakamiya, editors, *Biologically Inspired Approaches to Advanced Information Technology*, volume 3141 of *Lecture Notes in Computer Science*, pages 40–48. Springer Berlin / Heidelberg, 2004.

[66] Robert Legenstein and Wolfgang Maass. What makes a dynamical system computationally powerful? In S. Haykin, J. C. Principe, T. Sejnowski, and J. McWhirter, editors, *New Directions in Statistical Signal Processing: From Systems to Brains*, pages 127–154. MIT Press, 2007.

[67] Stuart Kauffman. Metabolic stability and epigenesis in randomly constructed genetic nets. *Journal of Theoretical Biology*, 22:437–467, 1969.

[68] Stuart Kauffman. *The Origins of Order: Self-Organization and Selection in Evolution.* Oxford University Press, 1993.

[69] B. Derrida and Y. Pomeau. Random networks of automata: A simple annealed approximation. *Europhysics Letters*, 1(2):45–49, 1986.

[70] Joseph T. Lizier, Mikhail Prokopenko, and Albert Y. Zomaya. A framework for the local information dynamics of distributed computation in complex systems, 2008. URL `http://arxiv.org/abs/0811.2690`. arXiv:0811.2690.

[71] Wolfgang Maass, Robert A. Legenstein, and Nils Bertschinger. Methods for estimating the computational power and generalization capability of neural microcircuits. In *Advances in Neural Information Processing Systems 17 (NIPS 2004)*, pages 865–872. MIT Press, 2005.

[72] Benjamin Schrauwen, Lars Büsing, and Robert A. Legenstein. On computational power and the order-chaos phase transition in reservoir computing. In Daphne Koller, Dale Schuurmans, Yoshua Bengio, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 21 (NIPS 2008)*. MIT Press, 2009.

[73] Olivia L. White, Daniel D. Lee, and Haim Sompolinsky. Short-term memory in orthogonal neural networks. *Physical Review Letters*, 92(14):148102.1–148102.4, 2004.

[74] David J. Field. What is the goal of sensory coding? *Neural Computation*, 6(4): 559–601, 1994.

[75] Ali Ajdari Rad, Mahdi Jalili, and Martin Hasler. Reservoir optimization in recurrent neural networks using Kronecker kernels. In *Proceedings of the IEEE Int. Symposium on Circuits and Systems*, pages 868–871, 2008.

[76] John M. Beggs and Dietmar Plenz. Neuronal Avalanches in Neocortical Circuits. *J. Neurosci.*, 23(35):11167–11177, 2003.

[77] Dante R. Chialvo. Critical brain networks. *Physica A: Statistical Mechanics and its Applications*, 340(4):756–765, 2004.

[78] John M Beggs. The criticality hypothesis: how local cortical networks might optimize information processing. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 366(1864):329–343, 2008.

[79] Claude E. Shannon and Warren Weaver. *The Mathematical Theory of Communication*. University of Illinois Press, Urbana, IL, 1949.

[80] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. John Wiley & Sons, New York, NY, 2nd edition, 2006.

[81] Alexander S. Klyubin, Daniel Polani, and Chrystopher L. Nehaniv. All else being equal be empowered. In Mathieu S. Capcarrère, Alex A. Freitas, Peter J. Bentley, Colin G. Johnson, and Jon Timmis, editors, *Proceedings of the 8th European Conference on Artificial Life*, volume 3630 of *Lecture Notes in Artificial Intelligence*, pages 744–753. Springer, 2005.

[82] Max Lungarella and Olaf Sporns. Mapping information flow in sensorimotor networks. *PLoS Computational Biology*, 2(10):e144, 2006.

[83] Olaf Sporns and Max Lungarella. Evolving coordinated behavior by maximizing information structure. In Luis Mateus Rocha, Larry S. Yaeger, Mark A. Bedau, Dario Floreano, Robert L. Goldstone, and Alessandro Vespignani, editors, *Proceedings of the Tenth International Conference on the Simulation and Synthesis of Living Systems*, pages 323–329. MIT Press, 2006.

[84] Mikhail Prokopenko, Vadim Gerasimov, and Ivan Tanev. Evolving spatiotemporal coordination in a modular robotic system. In Stefano Nolfi, Gianluca Baldassarre, Raffaele Calabretta, John C. T. Hallam, Davide Marocco, Jean-Arcady Meyer, Orazio Miglino, and Domenico Parisi, editors, *From Animals to Animats 9, 9th International Conference on Simulation of Adaptive Behavior, SAB 2006*, volume 4095 of *Lecture Notes in Computer Science*, pages 558–569. Springer, 2006.

[85] Lars A. Olsson, Chrystopher L. Nehaniv, and Daniel Polani. From unknown sensors and actuators to actions grounded in sensorimotor perceptions. *Connection Science*, 18(2):121–144, 2006.

[86] N. Ay, N. Bertschinger, R. Der, F. Güttler, and E. Olbrich. Predictive information and explorative behavior of autonomous robots. *European Physical Journal B*, 63:329–339, 2008.

[87] SP Strong, R Koberle, RRD van Steveninck, and W Bialek. Entropy and information in neural spike trains. *Physical Review Letters*, 80:197–200, 1998.

[88] Aonan Tang, Christopher Honey, Jon Hobbs, Alexander Sher, Alan Litke, Olaf Sporns, and John Beggs. Information flow in local cortical networks is not democratic. *BMC Neuroscience*, 9(Suppl 1):O3, 2008.

[89] A Tang and D Jackson. A maximum entropy model applied to spatial and temporal correlations from cortical networks in vitro. *J Neuroscience*, 28:505–518, 2008.

[90] Alexander Borst and Frédéric E. Theunissen. Information theory and neural coding. *Nature Neuroscience*, 2:947 – 957, 1999.

[91] Joseph T. Lizier, Mikhail Prokopenko, and Albert Y. Zomaya. Detecting non-trivial computation in complex dynamics. In Fernando Almeida e Costa, Luis Mateus Rocha, Ernesto Costa, Inman Harvey, and António Coutinho, editors, *Proceedings of the 9th European Conference on Artificial Life (ECAL 2007), Lisbon, Portugal*, volume 4648 of *Lecture Notes in Artificial Intelligence*, pages 895–904, Berlin / Heidelberg, 2007. Springer.

[92] Alexander S. Klyubin, Daniel Polani, and Chrystopher L. Nehaniv. Tracking information flow through the environment: Simple cases of stigmergy. In Jordan Pollack, Mark Bedau, Phil Husbands, Takashi Ikegami, and Richard A. Watson, editors, *Proceedings of the Ninth International Conference on the Simulation and Synthesis of Living Systems*, pages 563–568, Cambridge, MA, USA, 2004. MIT Press.

[93] Thomas Schreiber. Measuring information transfer. *Physical Review Letters*, 85(2):461–464, July 2000.

[94] Joseph T. Lizier, Mikhail Prokopenko, and Albert Y. Zomaya. Local information transfer as a spatiotemporal filter for complex systems. *Physical Review E*, 77(2):026110, 2008.

[95] Julien Clinton Sprott. *Chaos and Time-Series Analysis.* Oxford University Press, 2003. ISBN 0198508409.

[96] Julien Clinton Sprott. Numerical calculation of largest lyapunov exponent, August 2004. URL `http://sprott.physics.wisc.edu/chaos/lyapexp.htm`.

[97] Donald O. Hebb. *The Organization of Behavior: A Neuropsychological Theory.* Wiley, New York, 1949.

[98] Joseph T. Lizier, Mikhail Prokopenko, and Albert Y. Zomaya. Coherent information structure in complex computation. *Theory in Biosciences*, 2010. Accepted for the Theory in Biosciences, Guided Self-organization 2009 special issue.

[99] G. Silberberg, A. Gupta, and H. Markram. Stereotypy in neocortical microcircuits. *Trends in Neurosciences*, 25(5):227–230, 2002.

[100] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transaction on Neural Networks*, 5(2):157–166, 1994.

Published Papers by the Author

Papers related to the thesis

Journal papers with review

1. Joschka Boedecker, Oliver Obst, Joseph T. Lizier, Norbert Michael Mayer, Minoru Asada, "Information Processing in Echo State Networks at the Edge of Chaos" *Theory in Biosciences*, accepted.

2. Joschka Boedecker, Oliver Obst, Norbert Michael Mayer, Minoru Asada, "Initialization and Self-Organized Optimization of Recurrent Neural Network Connectivity" *HFSP Journal* Vol. 3, No. 5, pp.340-349, 2009.

International conference papers with review

1. Oliver Obst, Joschka Boedecker, Minoru Asada, "Improving Recurrent Neural Network Performance using Transfer Entropy " In *Proceedings of the 17th International Conference on Neural Information Processing (ICONIP 2010)*, 2010.

2. Joschka Boedecker, Oliver Obst, Norbert Michael Mayer, Minoru Asada, "Studies on Reservoir Initialization and Dynamics Shaping in Echo State Networks" In *Proceedings of the 17th European Symposium on Artificial Neural Networks (ESANN 2009)*, 2009.

Other papers

Journal papers with review

1. Norbert Michael Mayer, Joschka Boedecker, and Minoru Asada, "Robot motion description and real-time management with the harmonic motion-description protocol" *Robotics and Autonomous Systems*, Vol. 157, No. 8, pp. 870–876, 2009.

International workshop and conference papers with review

1. Joschka Boedecker and Minoru Asada, "SimSpark - Concepts and Application in the 3D Soccer Simulation League", In *Workshop "The Universe of RoboCup Simulators" at the 1st Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR 2008)*, 2008.

2. Rodrigo da Silva Guerra, Joschka Boedecker, Norbert Michael Mayer, Shinzo Yanagimachi, Yasuji Hirosawa, Kazuhiko Yoshikawa, Masaaki Namekawa, and Minoru Asada, "Introducing physical visualization sub-league", In *Proceedings of RoboCup 2007: Robot Soccer World Cup XI, Lecture Notes in Computer Science*, 2008.

3. N. Michael Mayer, Joschka Boedecker, Kazuhiro Masui, Masaki Ogino, and Minoru Asada, "HMDP: A new protocol for motion pattern generation towards behavior abstraction", In *Proceedings of RoboCup 2007: Robot Soccer World Cup XI, Lecture Notes in Computer Science*, 2008.

4. Rodrigo da Silva Guerra, Joschka Boedecker, Shinzo Yanagimachi, and Minoru Asada, "Introducing a new minirobotics platform for research and edutainment", In *Proceedings of the 4th International Symposium on Autonomous Minirobots for Research and Edutainment*, 2007.

5. Rodrigo da Silva Guerra, Joschka Boedecker, Norbert Michael Mayer, Hiroshi Ishiguro, and Minoru Asada, "A new minirobotics system for teaching and

researching agent-based programming", In *Proceedings of the World Conference on Computer and Advanced Technology in Education (CATE07)*, 2007.

6. N. Michael Mayer, Joschka Boedecker, Minoru Asada, "On Standardization of in the RoboCup Soccer Humanoid Leagues", In *2nd Workshop on Humanoid Soccer Robots at the IEEE-RAS International Conference on Humanoid Robots (HUMANOIDS 2007)*, 2007.

7. Norbert Michael Mayer, Joschka Boedecker, Rodrigo da Silva Guerra, and Minoru Asada, "3D2real: Simulation league finals in real robots", In *Proceedings of RoboCup 2006: Robot Soccer World Cup X, Lecture Notes in Computer Science*, pp. 25-34, 2006.

8. Oliver Obst, Joschka Boedecker, "Flexible Coordination of Multiagent Team Behavior using HTN Planning", In *Proceedings of RoboCup 2005: Robot Soccer World Cup IX, Lecture Notes in Computer Science*, 2006.

9. Minoru Asada, N. Michael Mayer, Joschka Boedecker, Sawa Fuke, Masaki Ogino. "The Robocup Soccer Humanoid League: Overview and Outlook", In *1st Workshop on Humanoid Soccer Robots at the IEEE-RAS International Conference on Humanoid Robots (HUMANOIDS 2006)*, 2006.

10. Joschka Boedecker, Norbert Michael Mayer, Masaki Ogino, Rodrigo da Silva Guerra, Masaki Kikuchi, and Minoru Asada, "Getting closer: How Simulation and Humanoid League can benefit from each other", In *Proceedings of the 3rd International Symposium on Autonomous Minirobots for Research and Edutainment*, pp.93-98, 2005.

Workshop and conference papers without review

1. Rodrigo da Silva Guerra, Joschka Boedecker, Hiroshi Ishiguro, Minoru Asada, "Successful Teaching of Agent-Based Programming to Novice Undergrads in a Robotic Soccer Crash Course.", *25th SIG-CHALLENGE Workshop*, pp. 21–26, 2007

2. Rodrigo da Silva Guerra, Joschka Boedecker, Minoru Asada, "Physical Visualization Sub-League: A New Platform for Research and Edutainment.", *25th SIG-CHALLENGE Workshop*, pp. 15–20, 2007

3. Rodrigo da Silva Guerra, Joschka Boedecker, K. Yamauchi, Taro Maekawa, Minoru Asada, Hirosawa, Masaaki Namekawa, K. Yoshikawa, Shinzo Yanagimachi, K. Nishimura, "CITIZEN Eco-Be! and the RoboCup Physical Visualization League.", *Micromechatronics Lectures – The Horological Institute of Japan*, 2006

4. Rodrigo da Silva Guerra, Joschka Boedecker, Norbert Mayer, Shinzo Yanagimachi, Tasuji Hirosawa, Kazuhiro Yoshikawa, Masaaki Namekawa, Minoru Asada, "CITIZEN Eco-Be! League: bringing new flexibility for research and education to RoboCup.", *23rd SIG-CHALLENGE Workshop*, pp.13-18, 2006