The Influence of the Activation Function on Reservoir Computers
Der Einfluss der Aktivierungsfunktion auf Reservoir Computer

Masterarbeit der Fakultät für Physik
der
Ludwig-Maximilians-Universität München

vorgelegt von

Joschka Herteux

betreut von
Dr. Christoph Räth

München, den 15. März 2021

Contents

Chapter 1

Introduction

Chaotic and nonlinear systems are all around us. Examples come from physics (the double pendulum, the three-body problem, turbulence, acoustic chaos, weather,...), medicine and biology (heart beat, neural dynamics, population dynamics,...) and even social sciences (economics, financial time series,...). Unpredictability is often considered the defining property of chaos. Even so, there are a number of methods known to achieve forecasting at least on short time scales. In addition to traditional approaches to global modeling and local linear fits [1] there has been a rise of artificial neural networks (ANN) as a tool to predict and analyze nonlinear systems [2, 3, 4, 5]. Among the latter there has been especially much attention towards the framework of Reservoir Computing (RC). RC is a machine learning method that has been independently discovered as Liquid State Machines (LSM) by Maass et al. [6],as echo state networks (ESN) by Jaeger [7] and as fractal prediction machine (FPM) by Tino and Dorffner [8]. It can be considered a specific kind of recurrent neural network (RNN), where only a memoryless, usually linear readout is trained while the weights of the recurrent connections are kept fixed. In comparison to RNNs with long-short-term-memory (LSTM), which use a costly backpropagation through time algorithm for training and are thus affected by the vanishing-gradient-problem, RC is extremely CPU-efficient and straightforward to train as one can use a simple, one-step linear regression procedure. At the same time it is able to deliver comparable and in some cases even better results [9, 10]. Besides its excellent capacity for short-time forecasting it garnered the most attention in the complex systems community with its ability to reconstruct strange attractors from data [11] and its applicability to high-dimensional input and output [12].
Arguably the most attractive feature of RC is however its generalizability beyond RNNs. The reservoir can consist of one of a wide variety of nonlinear dynamical systems itself, where ANNs are just a convenient special case. On the one hand this makes it an important object of theoretical study, since it delivers a framework to understand information processing in dynamical systems in general [13, 14, 15]. On the other hand this enables the application as physical RC: energy-efficient hardware implementations using photonic systems[16], spintronic systems [17] and many more are conceivable and being developed [18].
In light of this the natural question arises which properties are necessary, advantageous or optimal for a dynamical system to serve as a reservoir. While in the general context much work has been focused on the echo-state-property [7, 19, 15, 20], recent studies in the RNN framework focused mainly on the influence of the size and topology of the network on the prediction capabilities [21, 22, 23, 24, 25].
Another quite important aspect, which has been the subject of comparatively little research is the activation function. It is the only source of nonlinearity in the system and plays thus a central role. In this thesis we try to understand this role by systematically studying the effect of changes in the activation function on short- and long-term prediction capabilities.

We evaluate them by means of the forecast horizon as well as the largest Lyapunov exponent and the correlation dimension. To compensate for the high variance in results that is typical in RC [23] we create large statistics in our numerical experiments.
We mostly deal with two aspects of the activation function. Firstly, we examine the effect of the scaling of the hyperbolic tangent activation function, which controls its nonlinearity. We propose that there is a regime of nonlinearity in its domain, where inputs ideally should fall for best performance. By tuning the scaling we can control the location and size of this region (or equivalently the width of the input distribution). We find that for a wide variety of dynamical systems, the functional relation between the average forecast horizon and the nonlinear scaling parameter is qualitatively similar. The location of the main maximum of this curve roughly follows a power law with respect to the standard deviation of the input. At the same time there are other, typically less pronounced peaks with higher variety for shallow scalings corresponding to input distribution that lie largely in the linear regime of the activation function.
Secondly, we analyze the effect of the symmetry properties of the activation function. We find that the antisymmetry of the hyperbolic tangent leads to antisymmetry in the whole system. We show experimentally that this causes an inability to learn a large class of dynamics correctly. The result is the emergence of something we call mirror-attractor. This way we are able to explain some previously purely empirical results about the suitability of certain reservoir designs. To solve this problem we propose different modifications to our basic ESN: A bias term in the readout, an input shift in the activation function, a mix of even and odd activation functions, and two versions of a quadratic extension of the readout. We numerically test their proneness to the mirror-attractor, their ability to fulfill a standard task, where the symmetry comes into play, and finally their performance on a task specifically designed to be extremely sensitive to symmetry. We find that input shift and either quadratic readout are the most suitable, while mixed activation functions lead to slightly worse performance and the output bias is not able to break the symmetry in practice.
This work is organized as follows: In chapter 2 we introduce the methods used. In section 2.1 we introduce the dynamical systems with strange attractors we use as sources of synthetic data. Section 2.2 introduces the framework of RC and the basic implementation we mainly work with, which we call simple ESN. In section 2.3 we explain the measures we used to evaluate the performance of an ESN. In particular section 2.3.3 contains a short empirical comparison between different methods to numerically calculate the largest Lyapunov exponent to justify our decision for the Rosenstein-Kantz algorithm. In section 2.4 we introduce a method to teach a single ESN to reconstruct and predict the strange attractors of multiple dynamical systems we developed.
In chapter 3 we deal with the scaling of the activation function. Section 3.1 describes the motivation and intuition behind the experiments in section 3.2. Chapter 4 is dedicated to our research regarding the symmetry of the activation function. In section 4.1 we describe the mirror-attractor as a specific type of failed prediction we observe. In 4.2 we analytically prove that it is caused by a symmetry of the reservoir equations and demonstrate the consequences. We describe the different symmetry-breaking ESN design in section 4.3 and test them experimentally in section 4.4.
Finally we conclude by discussing the results of our work in chapter 5. Additional information can be found in the appendix.

Chapter 2

Methods

2.1 Strange Attractors

To study the ability of the reservoir to learn chaotic dynamics we need suitable training data. To be able to properly evaluate the performance and analyze possible problems it is useful to use synthetic data, where we understand the origins and properties of the attractor to some degree and can create new data as much as we need.
For this purpose we simulate a variety of different systems, which exhibit chaotic behavior and thus have strange attractors. A strange attractor is a fractal volume in the phase space of a dynamical system, which attracts any trajectory in some neighborhood. Thus, a trajectory cannot leave the attractor. Further, any semi-infinite trajectory starting in the attractor has to be dense in every part of it. A more rigorous definition can be found in any textbook on chaos or nonlinear dynamics.
The most famous strange attractor, which has been widely used in this context, is the *Lorenz attractor*. This system is often said to have historically started the study of chaotic systems [26]. It has been developed as a simplified model for atmospheric convection and exhibits chaos for certain parameter ranges. It is defined by the equations

$$\begin{aligned} \dot{x} &= \sigma(y - x) \\ \dot{y} &= x(\rho - z) - y \\ \dot{z} &= xy - \beta z \ , \end{aligned} \tag{2.1}$$

where we use the standard parameters $\sigma = 10$, $\beta = 8/3$ and $\rho = 28$. We simulate the dynamics by integrating these equations using the fourth order Runge-Kutta method with timesteps of $\Delta t = 0.02$. We use varying starting points on the attractor.
The equations are symmetric under the transformation $(x, y, z) \to (-x, -y, z)$. Furthermore, the mean of the attractor in z-direction is far away from the origin at $\bar{z} \approx 23.5$. This makes it an especially useful example for breaking the symmetry in the reservoir, which will be important in chapter 4.
As we will explore in chapter 4, it was previously believed that the mentioned symmetry of the Lorenz equations make it unsuitable for reservoir computing. For this reason sometimes a modified version of the Lorenz equations is used. Generally there is some utility in a test case without special symmetries. We simply call this *Modified Lorenz System.* The formal breaking of the symmetry is achieved by adding the term x to the z-component such that the equations read:

$$\begin{aligned} \dot{x} &= \sigma(y - x) \\ \dot{y} &= x(\rho - z) - y \\ \dot{z} &= xy - \beta z + x \ . \end{aligned} \tag{2.2}$$

We make much use of this system in chapter 3 with the same parameters as the regular Lorenz equations. Besides the broken symmetry, the attractor mostly keeps its properties. The mean and standard deviation of a typical trajectory as well as Lyapunov spectrum and correlation dimension are all approximately the same. A visual inspection also does not reveal any significant differences.

In chapter 4 we use the less known Halvorsen equations as a secondary test case [27]

$$\begin{aligned}\dot{x} &= -\sigma x - 4y - 4z - y^2 \\ \dot{y} &= -\sigma y - 4z - 4x - z^2 \\ \dot{z} &= -\sigma z - 4x - 4y - x^2 \ .\end{aligned} \tag{2.3}$$

with $\sigma = 1.3$. We simulate the dynamics in the same way as for the Lorenz equations. This system also exhibits chaotic behavior but does not have any symmetries under inversion of its coordinates. Instead it has a cyclic symmetry under their permutation. However, this should not be relevant in our context.
In chapter 3 we are interested in a large scale statistical comparison between different dynamical systems. Thus, we introduce a number of other nonlinear dynamical systems , namely the Rössler system [28], Rabinovich-Fabrikant equations [29], Rucklidge system [30], the Chua circuit [31], the Chen system [32], Thomas' cyclically symmetric attractor [33] and the Complex Butterfly attractor [34]. All these systems are $D = 3$ dimensional but differ in properties like Lyapunov exponents, correlation dimension, size of the attractor and the nature of their nonlinearity. However, to keep things simple, they all belong to the same category of autonomous dissipative flows, which also encompasses Lorenz and Halvorsen system. This means they are all continuous-time, phase space contracting, time invariant dynamical systems. The parameters for all of them except Lorenz and Rössler are taken from the textbook *Chaos and Time-Series Analysis* by Sprott [27].
In section 3.2.3 we use two additional test systems of this kind, which exhibit only local nonlinearities. They are the Simplest Piecewise Linear Chaotic Flow [35] and the Double Scroll attractor [27].
The explicit equations of all these systems can be found in the appendix.

2.2 Reservoir Computing and Simple ESN

There is a multitude of ways to design an ESN. In this work we use several different variants, which will be introduced in the following sections, when they are needed. For now we will set the general framework and focus on the most basic version we use.
In general the input is fed into the reservoir. The reservoir is itself some dynamical system with nonlinear behavior, which is influenced by the input. A usually linear readout is trained to translate the state of the reservoir into the desired output. The reservoir state is then a random, high-dimensional, nonlinear transformation of all previous input data. This naturally gives it a kind of memory.
Furthermore, the dynamical system in question should have the *echo state property*. The state of the reservoir continuously loses its dependence on past states over time. Any initial condition of the reservoir will converge to the same state when driven by the same input sequence for a long enough time [7, 36].

In an ESN the dynamics of the reservoir are generally governed by an update equation for the reservoir state $\mathbf{r}_t \in \mathbb{R}^N$ of the form

$$\mathbf{r}_{t+1} = f(\mathbf{A}\mathbf{r}_t, \mathbf{W}_{in}\mathbf{x}_t) \tag{2.4}$$

Here f is called activation function, the adjacency matrix $\mathbf{A} \in \mathbb{R}^{N\times N}$ represents the network, $\mathbf{x}_t \in \mathbb{R}^{d_x}$ is the input fed into the reservoir and $\mathbf{W}_{in} \in \mathbb{R}^{d_x\times N}$ is the input matrix. There are many possible choices for f and ways to construct $\mathbf{A}$ and $\mathbf{W}_{in}$. Here we always create $\mathbf{A}$ as an Erdös-Renyi random network. Sparse networks have been found to be advantageous[21]. Some other standard topologies have been tried, but were not found to be generally superior over Erdös-Renyi [23, 21]. Reservoir topologies have also been optimized for specific tasks by the use of genetic algorithms [37, 38] The weights of the network are drawn uniformly from $[-1, 1]$ and afterwards rescaled to fix the spectral radius ρ to some fixed value. ρ is a free hyperparameter, which is often treated as essential in theoretical considerations.
We chose $\mathbf{W}_{in}$ to be also sparse, in the sense that every row has only one nonzero element. This means every reservoir node is only connected to one degree of freedom of the input as it was done in [11]. For the calculations in chapter 4 we fixed the number of nodes per dimension to be the same plus or minus one. The nonzero elements are drawn uniformly from the interval $[-1, 1]$ and then rescaled with a factor s_{input}, which is another free hyperparameter.
From this we can then compute the output $\mathbf{y}_t \in \mathbb{R}^{d_y}$. Here we are interested in the prediction case, where we train the ESN to approximate $\mathbf{y}_t \approx \mathbf{x}_{t+1}$. The prediction is then created by feeding the output of the reservoir back into the input in a kind of closed loop. Thus, the dimension of input and output are the same, so we use $d_x = d_y := d$. The readout is characterized by

$$\mathbf{y}_t = \mathbf{W}_{out}\tilde{\mathbf{r}}_t \tag{2.5}$$

where typically $\tilde{\mathbf{r}}_t = \mathbf{r}_t$, but $\tilde{\mathbf{r}}_t \in \mathbb{R}^{\tilde{N}}$ can also be some nonlinear transformation or extension of $\mathbf{r}_t$. The readout matrix $\mathbf{W}_{out} \in \mathbb{R}^{d\times\tilde{N}}$ is the only part of the ESN that is trained. This is typically done via simple Ridge Regression [39].
To train the reservoir the training data $\mathbf{x}^{train} = \{\mathbf{x}_0, ..., \mathbf{x}_{T_{train}}\}$ is fed into the reservoir to compute the sequence $\mathbf{r}^{train} = \{\mathbf{r}_0, ..., \mathbf{r}_{T_{train}+1}\}$. The first T_{sync} timesteps of $\mathbf{r}$ are then discarded. This transient period is only used to synchronize the reservoir with the training data. This frees the ESN of any influence of the reservoir's initial condition thanks to the echo state property. This period where the reservoir is driven by an external signal is often referred to as the listening stage.

Now the readout matrix can be calculated by minimizing

$$\sum_{T_{sync}\leq t\leq T_{train}} \| \mathbf{W}_{out}\tilde{\mathbf{r}}_t - \mathbf{v}_t \|^2 - \beta \| \mathbf{W}_{out} \|^2 \ , \tag{2.6}$$

where we get another hyperparameter β from the regularization. The target output $\mathbf{v}_t$ is in the case of prediction just $\mathbf{x}_{t+1}$. We thus get [39]

$$\mathbf{W}_{out} = (\tilde{\mathbf{r}}^T\tilde{\mathbf{r}} + \beta 1)^{-1}\tilde{\mathbf{r}}^T\mathbf{v} \ , \tag{2.7}$$

where $\tilde{\mathbf{r}} \in \mathbb{R}^{\tilde{N}\times T}$ is $\tilde{\mathbf{r}}^{train}$ in matrix form after discarding the synchronization steps and $\mathbf{v}$ is analogous.

Our basic setup is close to what Jaeger [7] originally proposed and it is one of the most widely used variants. We call it simple ESN because all other designs we use are extensions of it. It is defined by the following equations:

$$\mathbf{r}_{t+1} = \tanh(\mathbf{A}\mathbf{r}_t + \mathbf{W}_{in}\mathbf{x}_t) \tag{2.8}$$

$$\mathbf{y}_t = \mathbf{W}_{out}\mathbf{r}_t \tag{2.9}$$

The activation function is a sigmoidal function, specifically a hyperbolic tangent, which is the typical choice. The reservoir states are not transformed before the readout. With this setup successful predictions of different datasets have been made in many cases. However, we can sometimes see very specific ways in which they fail. These problems are the focus of chapter 4.

Generally it is not fully understood how or under which circumstances RC works. There are analytic results in some cases, e.g. the ESN introduced in section 4.3.3 has been shown to be universal [36]. Specifically this means it can in principle approximate any causal and time-invariant filter with the fading-memory property. However, this does not necessarily show why it actually works so well in practice. Also, it does not explain the success of the plethora of alternative designs.

A theoretical framework based on invertible generalized synchronization was layed out by Lu et al. in [11] and developed further in [14]. This means that during the listening stage (in the limit of infinite time) the reservoir state $\mathbf{r}_t$ converges to a continuous function ϕ of $\mathbf{x}_t$.[1] It is notable that we demand $\mathbf{r}_t$ and not $\mathbf{r}_{t+1}$ to synchronize to $\mathbf{x}_t$. This means there is already a connection to the future state we want to predict, not just the past state that was fed in. However, in the case where $\mathbf{x}_{t+1}$ is a bijective function of $\mathbf{x}_t$, this is technically the same.

Based on this there are four necessary conditions for a reservoir computer to be able to replicate a strange attractor as claimed in [11](paraphrased):

1. The listening reservoir achieves generalized synchronization so $\mathbf{r}_t \approx \phi(\mathbf{x}_t)$ for the time interval covered in training.
2. The synchronization function ϕ is one-to-one, or at least carries enough information about its input to recover $\mathbf{x}_t$ from $\phi(\mathbf{x}_t)$.
3. Training is successful in finding a function ψ such that $\psi(\phi(\mathbf{x}_t)) = \mathbf{x}_t$.
4. The attractor approached by the reservoir in the listening stage is also stable during prediction.

Conditions 1-3 can at least be made plausible. Given the echo state property, generalized synchronization can be guaranteed. The training of ψ can be understood as a linear combination, where the degrees of freedom of the reservoir are the basis functions. It is clear that a high number and variety of their dynamics is beneficial.

Condition 4 deals with the long term behavior of the prediction. It is a priori not clear why even a reasonably well trained ψ, which is decent for short-time prediction, would not introduce enough deviations into the dynamics of the reservoir to make it go astray in the closed loop case. This requires the stability of the attractor $\phi(A)$ in the reservoir state space corresponding to the attractor A we want to replicate in the state space of the input. This can be measured by checking if the Lyapunov exponents (see section 2.3.3) of the reservoir transverse to $\phi(A)$ are all negative or just by checking the general quality of attractor reconstruction. Even though there is no theoretical explanation of this yet, this seems to hold true in practice in most cases where decent short-time prediction is possible.

The proper choice of hyperparameters is also an open question. The spectral radius in particular has been the subject of many theoretical studies. It is often wrongly assumed that $\rho < 1$ is a necessary and/or sufficient condition for an ESN to have the echo state property. Obviously, this property is necessary for the listening stage. However, it has been found that $\rho < 1$ is in fact neither necessary nor sufficient, even though it often works in practice [19]. Similarly, it is often assumed that a value of ρ close to 1 puts the ESN at the so called edge of chaos or edge of stability. This is supposed to be the regime of best information processing capacity. However, recently it has been shown that this not true in general [40].

[1] In general this can be the full state of the system that produced the input $\mathbf{x}_t$ called $\mathbf{s}_t$. This is important in cases where $\mathbf{x}_t$ is the product of some possibly nonlinear or dimensionality-reducing measurement $\mathbf{x}_t = h(\mathbf{s}_t)$. We will assume $\mathbf{s}_t = \mathbf{x}_t$ to keep things simple, since this is always the case in this work.

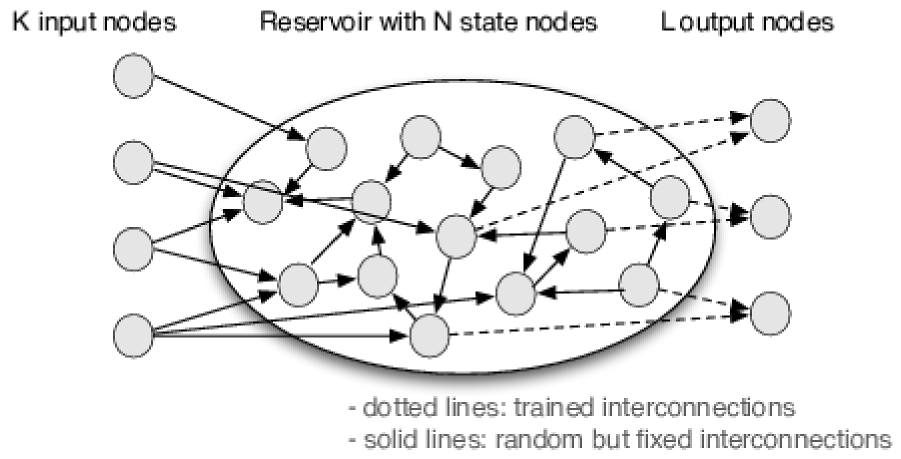

K input nodes
Reservoir with N state nodes
L output nodes
- dotted lines: trained interconnections
- solid lines: random but fixed interconnections

Figure 2.1: Schematic illustration of reservoir computing from [41].

Because of these complications it is practically impossible to find the best hyperparameters for a given problem from theoretical analysis. Instead they are usually fixed by some kind of hyperparameter optimization algorithm.
In the following we always use a network with $N = 200$, $T_{train} = 10500$ and $T_{sync} = 500$ and average degree $k = 4$ unless otherwise stated. The spectral radius ρ, the regularization parameter β and the input scaling s_{input} are optimized for specific problems with a hyperparameter optimization procedure described in the appendix.

2.3 Measures

2.3.1 Forecast Horizon

One aspect of the ESN that is of great interest is its capacity for short-time prediction. This area is often the most important one in applications, since statistical long-term behavior can also be inferred with traditional methods.
An obvious measure of this ability would be the RMSE with respect to the difference between prediction and testdata. This concept is already being used in the optimization of the parameters in the readout in equation 2.6. However, in the context of evaluating the ESN there are some disadvantages to this approach. Since we are dealing with chaotic systems, the prediction will invariably demerge from the real trajectory at some point and soon after move completely independently in the attractor. For a long prediction, the initial phase of accurate forecasting will be dominated in the calculation of the RMSE by this later stage. Thus, the results would be essentially random.
To combat this, we can introduce a threshold on the order of magnitude of reliable short-time prediction. Then we only use the timesteps before this threshold in the calculation of the RMSE. This makes it more useful, but the results are very dependent of the choice of the threshold even if a proper normalization is used. Setting it too high restores the previous problems, while setting it too low can easily omit the interesting point of departure. Finding a threshold fitting for a large number of different predictions to make them comparable is extremely difficult at best. And even then we face an additional problem: On average we expect exponential growth of small errors with a rate close to the largest Lyapunov exponent (see Sec. 2.3.3) for a reasonable predictor. However, this can vary strongly for a single realization. When working with the Lorenz system, we often see the two trajectories separate

System	x	y	z
Lorenz	5.8	8.0	6.9
Mod. Lorenz	5.8	8.0	6.
Roessler	3.1	2.8	3.4
Halvorsen	3.4	4.3	3.8
Thomas	1.3	1.3	1.3
Chua	0.67	0.12	0.96
Chen	8.3	9.7	7.8
Comp. Butt.	1.2	2.1	1.3
Rabinovich	0.23	0.40	0.19
Rucklidge	3.0	1.7	2.3
Normalized Lorenz	0.7	0.9	0.8
Lorenz (Halvorsen-Lorenz)	0.3	0.3	0.3
Halvorsen (Halvorsen-Lorenz)	0.3	0.3	0.3
Rucklidge+2Roessler	5.6	4.3	5.2
Roessler+2Rucklidge	6.8	4.5	5.8
0.4Mod. Lorenz+Halvorsen	4.8	5.2	4.7
0.5Mod. Lorenz+3Rabinovich	3.2	4.5	3.7

Table 2.1: Values of $\boldsymbol{\delta}$ for all systems used in this work.

very suddenly. On the other hand, we also see cases where they stay relatively close for a long time. This kind of behavior varies from realization to realization as well as between training datasets. The former case will often result in a RMSE orders of magnitude higher than the latter, making even averages unreliable.
Further, the interpretation of the RMSE is not straightforward. Even with a sensible normalization it depends strongly on the threshold and on the size of the attractor. It is not very helpful in getting a quick understanding of how long a prediction is reliable.
For these reasons it makes sense to instead use a measure that tries to quantify the time until this condition is not fulfilled anymore. As in [23, 24] we use the *forecast horizon.* It is defined as the time between the start of a prediction and the point where it deviates from the test data more than a fixed threshold. The exact condition reads

$$|\mathbf{v}(t) - \mathbf{v}_R(t)| > \boldsymbol{\delta} \ , \tag{2.10}$$

where the norm is taken elementwise. This means the forecast horizon is the first timestep, where the prediction is outside of a cuboid around the respective point in the test data. The dimensions of the cuboid are given by the elements of $\boldsymbol{\delta}$ doubled. In the typical case of fast exponential divergence this measure is not very sensitive to $\boldsymbol{\delta}$. The occasional slow divergence is somewhat more problematic. Still, there is no mechanism generating extreme outliers that make averages meaningless as with the RMSE.
The dependency on direction is useful especially if the relevant lengthscales of the studied dynamics differ for each dimension. For example this is the case for the chua circuit, which is one of the systems we examine in chapter 3. In general we set the elements of $\boldsymbol{\delta}$ to be about 15% of the total expansion of the attractor in the respective direction. We estimate the expansion as the difference between minimum and maximum value of a simulated trajectory of 100,000 timesteps. The resulting values are compiled in table 2.1.

With these parameters we get reasonable results and are able to make meaningful comparisons between realizations based on the same system. This does not allow for reliable comparisons between systems. Something like that would require at least more effort in the setting of analogous thresholds if it is even achievable with the forecast horizon. However, since this is not our goal in this work, we do not need to worry about it any further.

2.3.2 Correlation Dimension

A full evaluation of the prediction capabilities must also include the statistical long-term properties. These are often called the *climate*. An ESN that has successfully learned an accurate model of a dynamical system has to properly reproduce its attractor.
One important characteristic of a strange attractor is its fractal dimension. An overview on fractal dimension and ways to estimate it numerically can be found in [42]. We use the *correlation dimension*, which can be calculated from data using the Grassberger Procaccia algorithm [43].
The correlation dimension is based on the following idea: Assume we have a set $\{\boldsymbol{x}_i\}$ of points on the attractor with $i \in 1, .., n$. We now calculate the fraction $n_i(r)$ of points within a distance r from a specific point $\boldsymbol{x}_i$. This is an approximation of the pointwise mass function at $\boldsymbol{x}_i$ and can be written as

$$n_i(r) = \frac{1}{N-1} \sum_{j=1, j\neq i}^{N} \theta(r - |\mathbf{x}_i - \mathbf{x}_j|) \; . \tag{2.11}$$

Locally we assume this mass to scale with the distance as

$$n_i(r) \propto \quad r^{D_i} \tag{2.12}$$

from which the pointwise dimension is formally defined as

$$D_i = \lim_{r \to 0} \frac{\log n_i(r)}{\log r} \; . \tag{2.13}$$

This local measure of dimension is generally not the same in ever part of the attractor. For a global measure we need to average it over the complete set. We do this before taking the limit and get the correlation integral

$$C(r) = \lim_{N \to \infty} \frac{1}{N(N-1)} \sum_{i,j=1, i\neq j}^{N} \theta(r - |\mathbf{x}_i - \mathbf{x}_j|) \; , \tag{2.14}$$

The correlation dimension is then defined by the relation

$$C(r) \propto r^{\nu} \; , \tag{2.15}$$

as

$$\nu = \lim_{r \to 0} \frac{\log C(r)}{\log r} \tag{2.16}$$

analogously to the pointwise dimension.
To estimate this numerically one calculates the values of $C(r)$ for a wide range of r. In principle the dimension should then be the slope of a plot of $\log C(r)$ against $\log r$. However, in practice the scaling of equation 2.15 only holds in some intermediate region between r_{min} and r_{max}. Obviously, r_{min} is constrained from below by the minimum distance of two points in the set and r_{max} has an upper bound from the size of the attractor, but usually they are much closer together.
In this region we can use least-squares Linear Regression to make fit of the curve and obtain

ν as the slope. Since we want to create large statistics it makes sense to speed up the process even further and simply approximate the slope as

$$\frac{\log C(r_{max}) - \log C(r_{min})}{\log r_{max} - \log r_{min}} \ . \tag{2.17}$$

Of course this comes with a loss of accuracy, but we easily achieved average values which differed only a few percent from the literature value with a standard deviation of less than 1% for trajectories with 10000 timesteps as evidenced in tables 4.1 and 4.2.
Finding the appropriate scaling region is a task of its own, but again we have no need for high precision here. In principle any set of limits that gives reasonable results for the test data of a specific system is sufficient to make a comparison to the predicted trajectory of that same system. Thus, we mostly fixed these parameters by hand. However, for the experiments in chapter 4 we used a simple algorithm to automate the task. It performs essentially a grid search for r_{min} and r_{max} in their respective constraints for a given dataset. The fitness of each parameterset is then evaluated by considering the normalized root mean square error (NRMSE) of a linear fit to the curve, the closeness of the resulting ν to the literature value (if available) and the fraction ζ of the maximum possible region spanned by r_{min} and r_{max}. The last point turned out to be necessary as a primitive form of regularization to combat overfitting. In the most extreme case, two consecutive points on the discrete approximation of $\log C(r)$ can always be fitted perfectly with a line, which can easily approximate the desired slope by chance. Thus, we introduce a penalty for overly small distances between r_{min} and r_{max}.
This results in the loss function

$$L(r_{min}, r_{max}) = a_{fit}E_{fit} + a_{lit}(\nu_{lit} - \nu_{measured})^2 + a_{length}\zeta^{-3}, \tag{2.18}$$

where a_{fit}, a_{lit} and a_{length} are adjustable weights. E_{fit} is the NRMSE from the Linear Regression. The second term is the error in reproducing the literature value and the third term is the cost associated with a small length of the scaling region. We observed this cubic scaling in combination with a small weight to work quite well.
The procedure of estimating r_{min}, r_{max} and from them ν is illustrated in figure 2.2. The limits have been chosen as optimal for this particular datatset. Using a different starting point for the trajectory has an influence. Thus, it makes sense to consider them with an accuracy of about $O(0.1)$.

2.3.3 Lyapunov Exponents

In order to properly characterize our time series, we also need a measure of their temporal behavior. In the context of chaotic systems the Lyapunov exponents are an obvious choice. They indicate the stability of a dynamical system regarding small perturbations. This makes them especially interesting when dealing with chaotic systems. Additionally, there are well known methods to calculate them either from data or from the equations governing the flow of the system. Thus, they are accessible in practice, even if the origin of a time series is unknown, and simultaneously we have even more options in the case of synthetic data. Here we describe the data-driven Rosenstein-Kantz algorithm [44] as well as a method to calculate it from equations for continuous cases (e.g. Lorenz equations 2.1) as well as for discrete cases (e.g. our ESNs).
Lyapunov Exponents have a rich mathematical theory behind them, which has been described in detail for example in [45]. For our purposes the following short introduction will suffice.

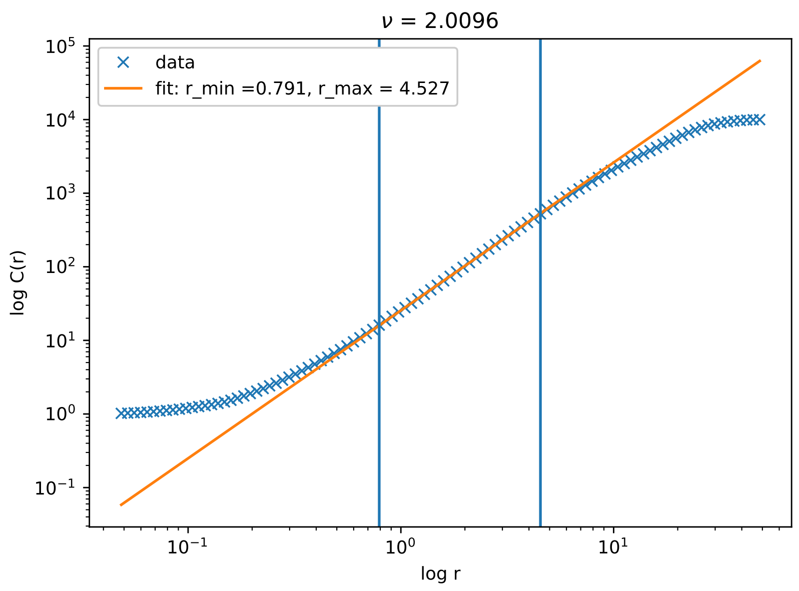


Figure 2.2: Estimating the correlation dimension of a Lorenz trajectory by using a linear fit

It is based on the respective sections in [46, 47, 1, 48].
A continuous dynamical system is defined by the differential equation

$$\dot{x} = f(x) \tag{2.19}$$

where x is the d-dimensional state vector and f is called vector field. The vector field generates a flow ϕ with

$$\dot{\phi}^t(x_0) = f(\phi^t(x_0)) \tag{2.20}$$

and

$$x(t) = \phi^t(x_0) \tag{2.21}$$

To characterize chaos we are interested in the way this system reacts to small perturbations. Let's consider the initial condition x_0 and the perturbation d_0. If a system exhibits chaos we expect u to grow exponentially. Then by taking a Taylor-expansion of the flow we can approximate

$$\phi^t(x_0 + d_0) = \phi^t(x_0) + D_{x_0}\phi^t(x_0) \cdot d_0 \tag{2.22}$$
$$d_t = D_{x_0}\phi^t(x_0) \cdot d_0 \tag{2.23}$$

In a chaotic system we also expect exponential growth of the perturbation described by

$$\|d_t\| = \|d_0\| \exp(\lambda t) \tag{2.24}$$

Thus, we define the local maximum Lyapunov exponent

$$\lambda(x_0) = \lim_{t\to\infty} \lim_{d_0\to 0} \frac{1}{t} \ln \frac{\|d_t\|}{\|d_0\|} = \lim_{t\to\infty} \lim_{d_0\to 0} \frac{1}{t} \ln \|D_{x_0}\phi^t(x_0)\| \tag{2.25}$$

Taking the limit in d_0 is necessary, because a finite perturbation is bounded by the size of the attractor and will stop its exponential growth, once it has reached this scale. The limit in time is necessary to make λ a well-defined quantity. It also makes it invariant under smooth transformations of the trajectory, which is especially useful in real applications, where we are often confronted with preprocessed data, different ways to measure, embeddings, etc.
The maximum Lyapunov exponent is an important characteristic of any chaotic system and can easily be estimated from data by using the Rosenstein-Kantz algorithm, which we briefly review in section 2.3.3. It corresponds to the direction of fastest divergence. However, for a complete picture it would be necessary to understand the behavior in all degrees of freedom. Thus, we are interested in the way a volume in phase space transforms, when allowed to evolve according to the given flow. For this we need a whole spectrum of Lyapunov exponents. Starting with the one in the direction of fastest separation (the maximum Lyapunov exponent), then the one in the direction of fastest separation chosen from all directions perpendicular to the first and so on. This already sounds a lot like the Gram-Schmidt method, which is no coincidence. It will be heavily used in the numerical calculation in section 2.3.3. For a chaotic system at least one Lyapunov exponent is positive, while the others can be negative, indicating exponential decay rather than growth. In fact, the sum of all Lyapunov exponents, which represents the overall rate of phase space expansion or contraction, can not be positive in a physically meaningful system [1].

Rosenstein-Kantz Algorithm

The maximum Lyapunov exponent can be calculated from a prerecorded, e.g. measured, trajectory. The standard method to do this is the Rosenstein-Kantz algorithm [44, 1]. The basic idea is to designate a random datapoint in the time series as as x_0. Then identify all points in a neighborhood of some spatial distance ϵ or simply the nearest neighbors, excluding those with a temporal distance to x_0 below some threshold τ. This makes use of the fact, that a long trajectory will almost always be dense in the attractor. Thus, different sections will be close enough to be considered perturbations of each other.
Now one simply has to follow the further temporal development of x_0 and all the points in the neighborhood and their distances over some time t. To get rid of local fluctuations and noise and simply get a good statistic one can then average in some way over different x_0, t and all the points in the respective neighborhoods.
In practice this means the following: We first calculate the distances of all points in the time series. Then we identify the nearest neighbor pairs. We exclude those, which do not pass the temporal threshold. Then for a range of timesteps between t_{min} and t_{max} we calculate the mean of the logarithmic distances of the former nearest neighbor pairs, which we call S. The functional dependency of this quantity on the timestep t should then be linear. Similarly to the procedure of estimating the correlation dimension, we can in principle find the maximum Lyapunov exponent as the slope of this line by applying Linear Regression and again we can speed things up by only using $S(t_{min})$ and $S(t_{max})$ instead.
Ultimately we get the maximum Lyapunov exponent as

$$\lambda = \frac{S(t_{max}) - S(t_{min})}{t_{max} - t_{min}} \tag{2.26}$$

Numerical Calculation of the Lyapunov Spectrum from Equations

The numerical calculation of the Lyapunov spectrum differs slightly depending on if the system in question is continuous or discrete. However, any method is based on estimating

the time development of $Y_t(x_0) := D_{x_0}\phi^t(x_0)$ for different x_0
In the continuous case, $Y_t(x_0)$ follows the so-called *variational equation*

$$\dot{Y}_t(x_0) = D_x f(\phi^t(x_0)) \cdot Y_t(x_0) \tag{2.27}$$

which can easily be verified by the chain rule. So to simulate the development of the state vector x together with Y, we have to numerically solve

$$\begin{pmatrix} \dot{x} \\ \dot{Y} \end{pmatrix} = \begin{pmatrix} f(x) \\ D_x f(x) \cdot Y \end{pmatrix} \tag{2.28}$$

with initial condition

$$\begin{pmatrix} x(0) \\ Y(0) \end{pmatrix} = \begin{pmatrix} x_0 \\ I \end{pmatrix}$$

(2.29)

To do this we can simply use the fourth order Runge-Kutta method.
Now following the algorithm by Benettin et al. [49] we define a volume in phase space spanned by a basis of linear independent vectors $v_1^0, ...v_d^0$ around an origin x_0. The development of this volume for a short time T is then

$$(v_1^T, ...v_d^T) = Y_T(x_0) \cdot (v_1^0, ...v_d^0) \tag{2.30}$$

which means that, if we choose the identity matrix I as initial condition $v_1^0, ...v_d^0$, we simply have to integrate the variational equation 2.27 as described above. To actually measure the volume we can orthonormalize the spanning vectors by using the Gram-Schmidt procedure and get $w_1^T, ..., w_d^T$ so the Volume is $Vol\{v_1^T, ..., v_d^T\} = \prod_i \|w_i^T\|$.
The local estimate for the Lyapunov exponents is then

$$\lambda_i(x_0) = \frac{1}{T} \ln \|w_i^T\| \tag{2.31}$$

ordered by decreasing size. For a global estimate we can continue the simulation of the trajectory and repeat the process using $w_1^T, ..., w_d^T$ as new initial condition. By continually applying this on a long enough trajectory we essentially sample from the whole attractor. The final result for the Lyapunov exponents is then aquired by averaging over the local estimates.
In the case of a discrete system, like the ESN, the main ideas are the same with the key-difference being the time development of Y. We write the discrete map as $x_{t+1} = f(x_t)$. Instead of having to approximately integrate the variational equation, the chain rule makes the calculation a matter of simple matrix multiplication:

$$D_{x_0} f^t(x_0) = D_x f(x)|_{x=f^{t-1}(x_0)} \cdots D_x f(x)|_{x=f(x_0)} \cdot D_x f(x)|_{x=x_0} := J_{t-1} \cdots J_1 \cdot J_0 \tag{2.32}$$

The algorithm by Eckmann and Ruelle [50] now uses the well-known QR composition to fulfill a role similar to the Gram-Schmidt orthonormalization method before. Firstly, we apply it to the Jacobian at the initial condition $J_0 := Q_1 R_1$By defining

$$J_k = J_{k+1}^* Q_k^T \tag{2.33}$$

with the QR-decomposition $J_k^* = Q_k R_k$ for $k >= 2$ and putting it into equation 2.32 we get

$$D_{x_0} f^t(x_0) = Q_t R_t \cdots R_1 := Q_t V_t \tag{2.34}$$

It can be shown that the diagonal elements $v_{ii}(t)$ of the product of R-matrices V_t can be used to approximate the Lyapunov exponents as

$$\lambda_i = \frac{1}{t} \ln v_{ii}(t) \tag{2.35}$$

which becomes exact in the limit of large t.
In practice this can be implemented by repeatedly applying the map to simulate a typical long trajectory of the system, while also computing the local Jacobian at every step. Then the J_k^* and their QR-decompositions can be calculated iteratively.
This can also be used to find the Lyapunov spectrum of the prediction produced by an ESN [51]. Because of the recurrent connections it is not possible to just calculate the Jacobian of the output. Instead, one needs to apply the algorithm to the dynamics of the reservoir state itself in the closed loop case, which e.g. for a simple ESN is given by

$$\mathbf{r}_{t+1} = \tanh((\mathbf{A} + \mathbf{W}_{in}\mathbf{W}_{out})\mathbf{r}_t) \tag{2.36}$$

where we can easily find the Jacobian. If the prediction has d degrees of freedom and the Number of nodes in the network is $N > d$, then the largest d Lyapunov exponents of the reservoir are also the Lyapunov exponents of the prediction. This is rooted in the invariance of the Lyapunov exponents under smooth transformations.

Comparison

Since the data we use in this work to train our ESNs is synthetic and we have access to the underlying equations, both methods of calculating at least the maximum Lyapunov exponent are viable. The same is true in principle for the predictions of the ESN, which is governed by the reservoir equations. To not overcomplicate things we want to only use one approach in this thesis and thus a proper evaluation of their advantages and disadvantages from a theoretical as well as empirical standpoint is appropriate.
The equations-based methods have some obvious advantages beginning with the fact that they deliver the whole Lyapunov spectrum and not only the largest exponent. In applications that rely on this the Rosenstein-Kantz algorithm is obviously unsuitable. While this is not explicitly the case in the present work, the additional measures of attractor climate would certainly strenghten the meaningfulness of the result. Further, we intuitively expect the knowledge of the equations to be beneficial for the accuracy of the estimation. To test this we simulated 500 trajectories of the modified Lorenz system with different initial conditions and calculated their largest Lyapunov exponent with the Rosenstein-Kantz algorithm. To investigate the influence of the number of iteration steps we compare the results for using differently sized fractions of the trajectories. Then we used the equations-based algorithm for continuous dynamical systems to do the same. We then compared their dependence on the number of iteration steps. The results can be seen in figures 2.3a, 2.3c, 2.3b and 2.3d.

The plots seem to confirm our hypothesis. Both methods converge to the best result with longer time series but this happens at a much faster rate with the equations-based approach. However, there is a caveat to this because iteration steps in the two cases do not mean the

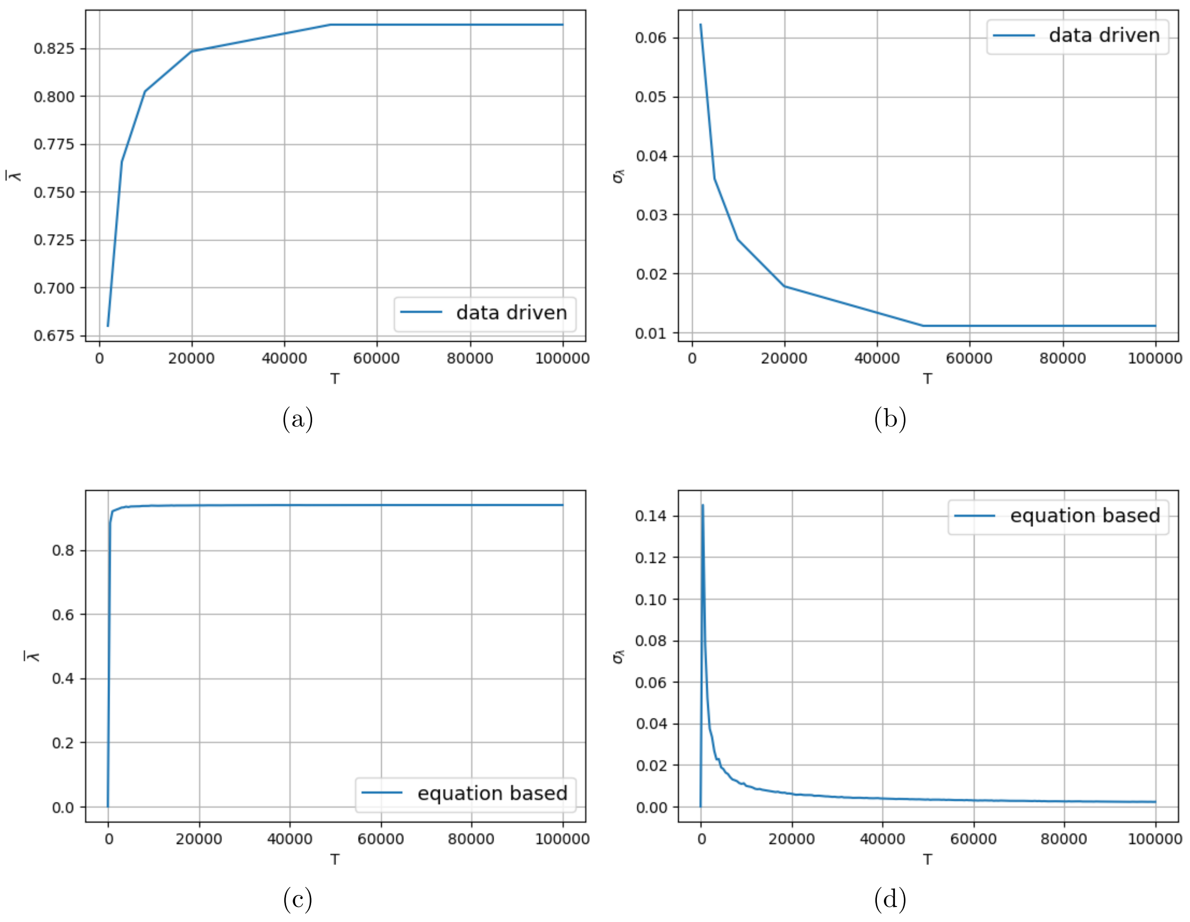


Figure 2.3: Comparison of convergence in data driven approach (Rosenstein-Kantz algorithm) and equation based approach. For both cases the average maximum Lyapunov exponent $\bar{\lambda}$ and its standard deviation σ_λ are plotted against the number of iteration steps. Figure a): average maximum Lyapunov exponent as calculated with the Rosenstein-Kantz algorithm. Figure b): Standard deviation of the maximum Lyapunov exponent as calculated with the Rosenstein-Kantz algorithm. Figure c): average maximum Lyapunov exponent as calculated from the equations. Figure b): Standard deviation of the maximum Lyapunov exponent as calculated from the equations.

same thing. The data-driven method involves mostly integrating the equations of motion to create the synthetic data (which would be necessary in any case) and calculating the nearest neighbor pairs. The first task has a time complexity of $O(N)$, where N is the number of steps, and the second task can be done with $O(NlogN)$ by using the k-d-tree algorithm.
When using the second method, we additionally integrate the variational equation 2.27 to approximate the Jacobian. This makes the time cost of the simulation much higher, but it only scales with $O(N)$. Thus, we expect the average time cost per time step of Rosenstein-Kantz to be smaller than that of the equations-based method for short overall length of the trajectory, but to be overtaken for long trajectories.
We are interested in making many calculations with different networks and initial conditions for the training data. Thus, it would be impractical to simulate very long trajectories beyond $O(10000)$ timesteps. We measured the average time cost on our system of both methods depending on the length of the simulated trajectory with 100 realizations in every point. The results are compiled in figure 2.4.

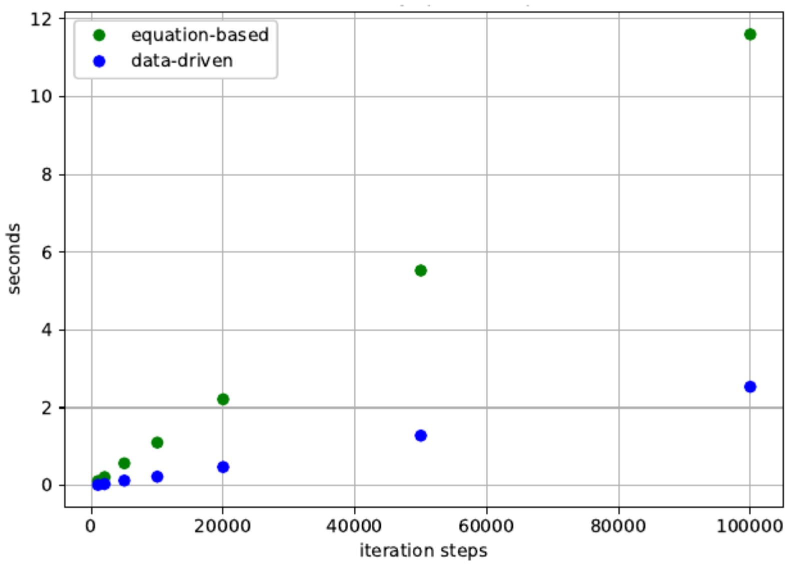


Figure 2.4: Comparison between time cost of both approaches depending on number of iteration steps.

We observe that one iteration step of the Rosenstein-Kantz algorithm is much faster in the regime we are interested in. We usually use trajectories of length 10000 timesteps as training data. The time needed to calculate the largest Lyapunov exponent of this data is approximately the same as needed to let the equation-based approach run for 2000 steps. According to figure 2.3b and 2.3d the respective results would both have a standard deviation of about 0.01. Therefore, the advantage in accuracy is not actually very pronounced.
The situation is generally similar when we apply the discrete algorithm to the reservoir equations. Because of the comparably high dimensionality of the reservoir state, the computation of the Jacobian is expensive. However, we also observed another problem in this case. For small values of the regularization parameter β the results of this method are extremely unstable under variation of the network. Figure 2.5 shows the standard deviation of the largest Lyapunov exponent from 100 different simple ESNs trained on the same data depending on β.

For β below 10^{-10} it diverges completely. As we would expect, this is not the case for any single network predicting trajectories with different initial conditions, since the regularization only has an influence on the training process. Still, this observation is highly surprising, given that the same behavior is not seen for the data-driven algorithm. While the climate of the prediction generally seems to profit from higher values of beta, which is also visible in the correlation dimension, the magnitude of the variation here is extremely unusual. This seems to imply a oversensitivity of the numerical algorithm and not of the actual Lyapunov exponent. However, we have no explanation for the nature of this oversensitivity. This seems to be a disadvantage of using the equations-based approach in our case.
There are some further arguments in favor of the data-driven method, which mainly center around the capacity to compare different results. Firstly, since the equations we use to produce training data are continuous and the ESN is a discrete system, we are forced to use two different algorithms in each case. We can never be quite sure if a difference in the variance of Lyapunov exponent of the output and the input is due to the performance of the ESN or due to the different methods of measurement. Thus, there is an issue with the comparability of the prediction and the test data.

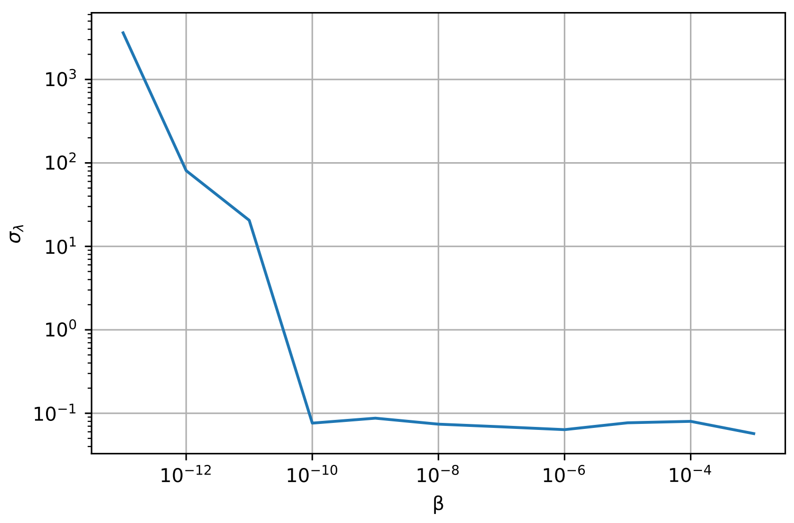


Figure 2.5: Standard deviation in largest Lyapunov exponent of a reservoir calculated directly from its equations depending on the regularization parameter β. Both axes have logarithmic scale.

Secondly, in chapter 4 we will be using a variety of different designs of the ESNs with different reservoir equations and readouts. The specifics of the algorithm are not the same, since this means different Jacobians and in the case of a nonlinear readout a more complicated relationship between Lyapunov spectrum of the reservoir itself and its output. Thus, there is an issue with the comparability between ESN designs.
Finally, in real applications the underlying equations of the training data are usually not known. In this case one would be forced to use a data-driven algorithm by default. It is useful to have a standard procedure that is also applicape in this case and does not only work for synthetic data. Thus, there is an issue with comparability to real applications.
In light of all of these arguments, the measurements in this work are always done with the Rosenstein-Kantz algorithm, even though there is still some degree of arbitrariness to this decision. We also want to note that there are a plethora of other (although in many cases related) algorithms for this task, which we did not test. An overview can be found in [47].

2.4 Training from Different Sources and Multifunctionality

In chapter 4 we will face the need to teach a single ESN with a single readout to predict trajectories on two completely independent strange attractors living in the same space. To be able to do this we developed a method that relies on the memorylessnes of the readout to combine training data from multiple different sources.
Since the readout is simply a direct mapping from the reservoir state to the output, it is not influenced by the broader temporal development of the system. All the available memory about the previous states of the input is encoded in a single instantiation of the reservoir state. This is made possible by the reservoir's recurrent connections and its high dimensionality.
For this reason the temporal and causal relationship between the reservoir states themselves does not matter for computing the readout matrix $\mathbf{W}_{out}$. This means that for example we can completely change the order of pairs of reservoir state and target in equation 2.5 without affecting the result. This is also obvious from a close look at the equation: In both relevant terms $\tilde{\mathbf{r}}^T\tilde{\mathbf{r}}$ and $\tilde{\mathbf{r}}^T\mathbf{v}$ the time steps are simply summed over in a matrix multiplication.

Knowing this we can train our ESN on data from multiple distinct trajectories and even different dynamical systems. Let's consider the case of two different trajectories A and B, either from the same or two different dynamical systems: We start by synchronizing with the data from trajectory A and record $\mathbf{r}_A^{train}$ after the initial transient period. This is just the regular listening stage. We do however not calculate $\mathbf{W}_{out}$ yet. Instead we repeat the listening stage with the data from trajectory B. Now the transient period has the additional use of letting the reservoir forget about the first batch of data. This way we get $\mathbf{r}_A^{train}$ and $\mathbf{r}_B^{train}$. We simply concatenate them to get a single dataset $\mathbf{r}^{train}$, from which we finally compute the readout matrix. As desired output we use an analogous concatenation of the target data of both trajectories. The concatenation in both cases is applied along the time axis as if the two time series followed each other chronologically. Because of the mentioned memorylessnes and because the transient period at the second training stage was discarded, the transition between the two trajectories in itself does not influence training. In principle this can easily be generalized to any number of trajectories.
If the different trajectories stem from different strange attractors, then this has the potential to achieve our stated goal. It is a priori not clear if an ESN is actually able to exhibit multistability of two or more distinct attractors.
To test this we first used two dynamical systems, the Lorenz and the Halvorsen system. We calculate trajectories from both and adjusted their means and variance to make sure they do not overlap with each other. We then carried out the above procedure. Further details and results of this experiment can be found in section 4.4.3 and figure 4.10, where they are presented in the context for which we originally developed this method. The main result in this context is that it was indeed possible to reproduce the dynamics of both attractors with the same ESN, although not with the simple ESN. One modification that makes it possible is the inclusion of a random bias vector or input shift in the activation function as introduced in section 4.3.3 for reasons that will be thoroughly explored in chapter 4.
Similar results have been found by two recent works [52, 14]. They use slightly different methods to achieve this goal and link this ability of the reservoir to the neuroscience term multifunctionality. Neural networks in the human brain are able to switch between different tasks corresponding to different stable attractors of their dynamics. At the point of original publication of the research in chapter 4 the authors were unaware of this.
Because of this we are now interested in the limits of multifunctionality in our ESN. To create a harder testcase we simulate multiple different strange attractors from section 2.1. Since differently sized timesteps were found to have a negative influence [52], we use the same size of timesteps $\Delta t = 0.02$ in all cases, even though they are not necessarily ideal for all systems. We rescale every trajectory to have a maximum expansion of 1 in every dimension and shift their means to the different corners of a $2 \times 2 \times 2$ cube around the origin. This way they do not overlap, but still fill out a volume in phase space relatively densely. We train an ESN on all of them with the described method. We use an ESN with input shift with $N = 2000$, $\beta = 10^{-6}$ and otherwise the same hyperparameters as for the analogous setup in section 4.4.3. By manual trial-and-error we find that we can reconstruct up to seven different attractors with reasonable reliability. The result can be seen in figure 2.6.

We observe that some strange attractors seem to be more prone to disrupt the multistability of the entire system. This seems to loosely correlate to dynamical systems we generally found to be harder to predict (Complex Butterfly attractor, Double Scroll attractor). Their inclusion tends to lead to predictions of themselves and other systems to leave their attractor and instead converge to another trained attractor or even an untrained fixed point. This could of course also be caused by the hyperparameters, which are only optimized for the Lorenz and Halvorsen system.
We further observe that it is in most cases possible to reconstruct 8 mean-shifted copies of the same attractor and possibly more. It is unclear if or how the ESN might be able to take

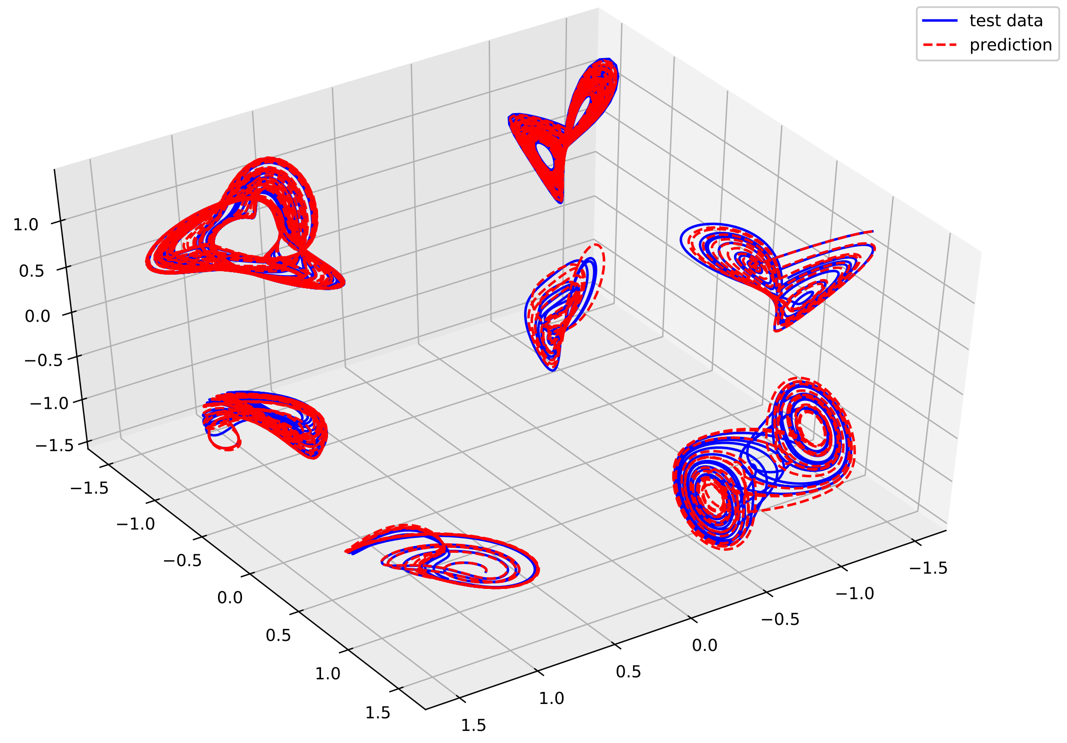


Figure 2.6: Seven different strange attractors reconstructed with a single ESN with a single readout matrix. Upper corner: Lorenz system, upper right corner: Rucklidge system, lower right corner: Chua circuit, lower corner: Roessler system, lower left corner: Rabinovich-Fabrikant equations, upper left corner: Halvorsen system, middle: Chen system

advantage of their similarity even though there is no underlying set of equations known to create these dynamics with this particular translational symmetry.
This opens up many possible further research topics. Firstly, a more systematic and rigorous analysis of the preliminary results above is necessary. It is unclear how large a reservoir really has to be for multifunctionality, what makes different systems more or less compatible and what the real limits are etc.
Furthermore, the ability to train from different sources allows for a plethora of possible new ways to do RC. By including a large number of short trajectories leading into an attractor from all sides, one might be able to increase its stability. An especially complicated part of the attractor could be given more weight by supplementing the regular training data with additional trajectories covering this region.
In general this makes it easy to freely model a new complex dynamical system not on an equation level, but on a level of desired behavior by combining trajectories from different sources as needed and training an ESN on them. How well all of this would actually work in practice and where the limits lie could be another new area of research.
However, all of this is outside of the scope of this work. Here we are satisfied with knowing that the described mechanism allows for the experiment in section 4.4.3.

Chapter 3

Scaling of the Activation Function

The results presented in this chapter have largely been published in [24].

3.1 Motivation and Theory

The focus of this thesis is the activation function of the ESN. Thus, a reasonable first step is to tune it in a controlled way and observe the effect on the prediction quality. An interesting way to adjust this activation function is to include an additional factor in the argument which affects its scaling. This leads to the following new equation for the simple ESN:

$$\mathbf{r}_{t+1} = \tanh(a[\mathbf{A}\mathbf{r}_t + \mathbf{W}_{in}\mathbf{W}_{out}\mathbf{r}_t]) \ . \tag{3.1}$$

We call the new factor a nonlinear scaling parameter and its influence can be understood in three different ways:
Firstly, we see a as not actually being something new. Since we already have the scaling of W_{in} and the spectral radius ρ as hyperparameters we could simply tune them simultaneously to get the same effect. Thus, the nonlinear scaling parameter does not change our model and really just serves as a tool to move along a specific line through hyperparameter space. Optimizing hyperparameters is often left to black-box algorithms in ML since the underlying rules are often extremely complicated. For ESNs specifically it has been found recently that the performance landscape of an ESN with a fixed network and fixed training data shows infinitely-fine scaled alternations [53], which is what causes the high variation in results. Still, there are heuristics to find regions in parameter space which are on average more suitable than others. The most trivial example of this would be the obvious fact that extremely small or large values will make the algorithm useless. We expect to find such a heuristic in this case, because of the other two viewpoints on a.
Secondly, as the name suggests, the nonlinear scaling parameter tunes the nonlinearity of the activation function. This is a crucial property for reservoir computing. Since the reservoir itself, as well as the output function, is usually linear, the activation function is the only source of nonlinearity in the system. This has well-known effects on the memory of the reservoir known as Memory-Nonlinearity tradeoff [54, 55, 56, 57]. However, in the present study we focus on systems where the role of memory is small. One might expect that a high degree of nonlinearity is advantageous if the dynamics which created the training data are highly nonlinear.
Finally, we can understand the tuning of a as a scaling of the distribution of input data the activation function gets. This is especially relevant for the simple ESN, where we observe

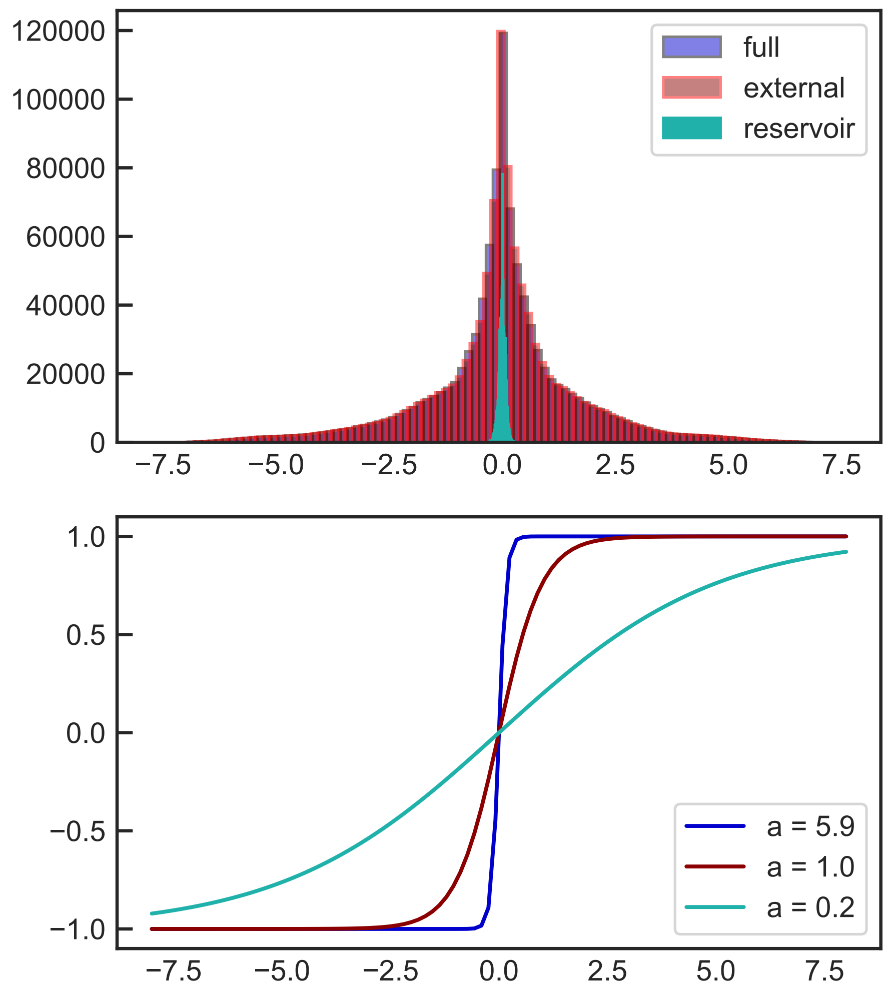


Figure 3.1: Top plot: Distribution of arguments of the activation function during training period split into the contribution from the reservoir (green), the input term (red) and total (blue) Bottom plot: Hyperbolic tangent for different nonlinear scaling factors.

that shifting and rescaling of the training data, which is usually considered in ML, leads to a failure of the prediction in many cases. We explore the reasons behind and solutions for this in chapter 4, but for now we will take it as a given. Therefore, the width of the distribution of the input data, which is basically the width of the underlying strange attractor, is a characteristic feature for us. In principle it is in a complex relationship with the distribution of reservoir states during the run of the ESN and with the overall distribution of the argument of the activation function. We can simultaneously tune all of them by changing a.
Let us examine this last viewpoint a little. In figure 3.1 we can see the distribution of arguments (in all nodes of the network) in the activation function during a typical training period on data from the modified Lorenz system in the upper plot.

The argument is split up into the reservoir term $\mathbf{Ar}$ and the external term $\mathbf{W}_{in}\mathbf{u}$. We used the hyperparameters we found to be optimal for this problem in a random search procedure described in the appendix. Specifically, we have $\rho = s_{input} = 0.17$. As we can see this leads to a strong domination of the external term over the reservoir term. Since the state of any node is the output of a hyperbolic tangent function, it lies in $[-1, 1]$, while the z-coordinate of the modified Lorenz system can go up to about 40. This gives us an upper bound for

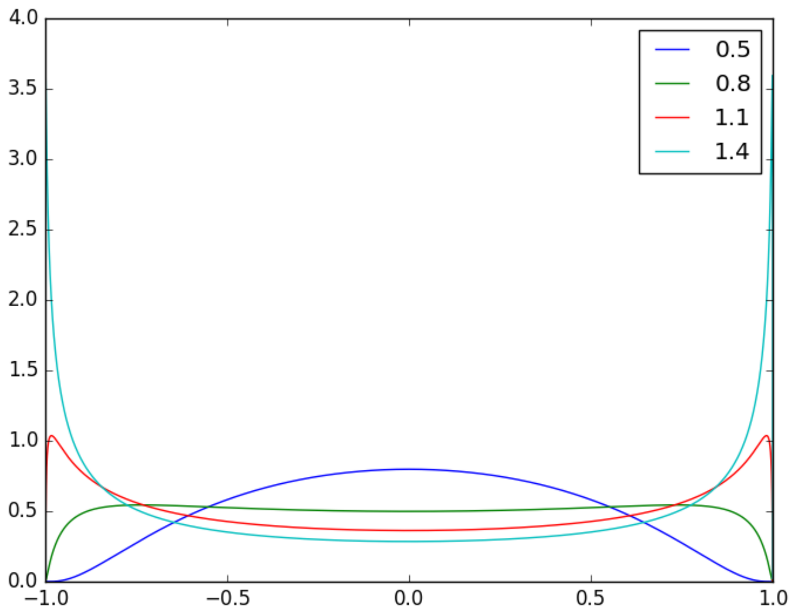


Figure 3.2: Gaussian distribution with different widths after applying the hyperbolic tangent.

the reservoir term itself. Since $\mathbf{A}$ is the adjacency matrix of an undirected network and thus symmetric we know

$$\|\mathbf{A}\mathbf{r}\| \leq \rho\|\mathbf{r}\| \leq \rho\sqrt{N} \approx 2.4 \ . \tag{3.2}$$

In practice this upper bound is still quite an overestimate as we can see in the plot that the contribution of the reservoir term is never more than $O(0.1)$. This allows us to see this distribution as approximately independent of a, when we count a as part of the activation function, even though it technically influences the distribution of reservoir states.
Further, in comparison of the upper and lower plot of figure 3.1 it can be seen how the non-linear scaling parameter controls which regime of the hyperbolic tangent is activated by the input. Generally, for small arguments it shows approximately linear behavior as evidenced by its Taylor expansion $\tanh(x) = x - \frac{1}{3}x^3 + O(x^5)$. Large arguments on the other hand lie in the saturation region, where they basically get mapped to 1 or -1, which makes it hard for the readout to distinguish them. Only in a region of intermediate absolute value do we see clearly nonlinear behavior. Thus, we intuitively assume that the ESN shows the best performance, the more of the distribution of input data lies in this "dynamical" region. The blue line in the lower plot shows the course of $\tanh(ax)$ with $a = 5.9$, which given our actual choices of ρ and s_{input} is equivalent to $\rho = s_{input} = a = 1$. This standard choice of hyperparameters would clearly lead to a high saturation.
The effect can be further studied by looking at the pure distribution of reservoir states before application of $\mathbf{A}$. If we model the input distribution as gaussian we can analytically calculate the resulting distribution of reservoir states. In figure 3.2 this is plotted for different widths of the original distribution.

As expected, a high width (corresponding to a high a) leads to increasing accumulation of r at -1 and 1. Lower values can transform the distribution to something loosely resembling

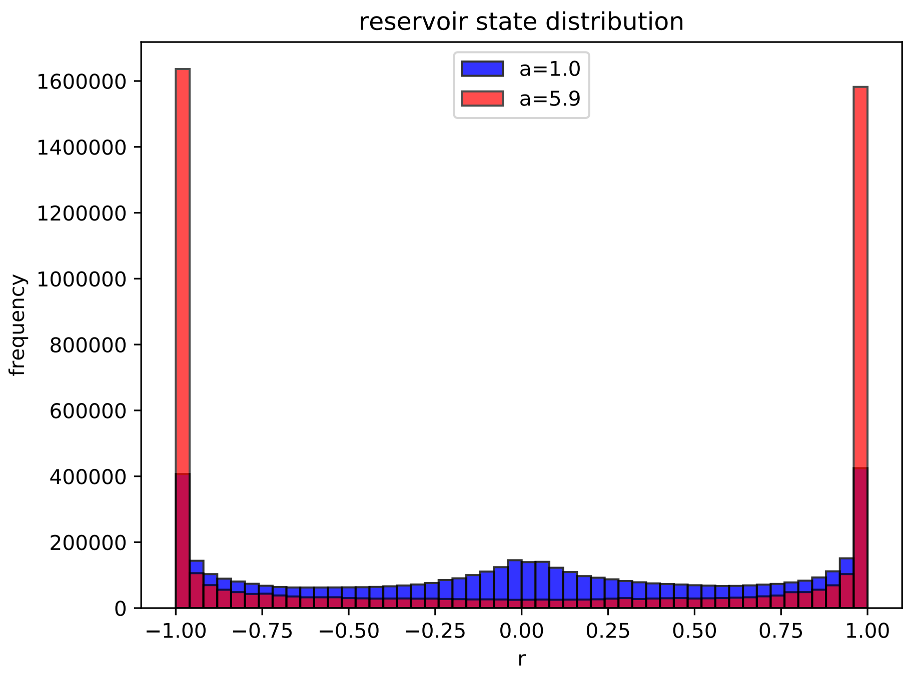


Figure 3.3: Distribution of reservoir states over all nodes during training on modified Lorenz data for different values of a. In both cases the training data was the same trajectory of 10000 timesteps length and the same network and input matrix.

a uniform one as shown by the green curve, where the width is 0.8. Approximating this distribution seems ideal to use the whole range of possible values of r and make a distinction of different reservoir states by the readout as easy as possible.
To get a better understanding of this we looked at the real distribution of r-values during training for the standard choice of parameters and the one given by hyperparameter optimization. The corresponding histograms are plotted in figure 3.3. Indeed, the optimized parameters lead to much less saturation, but notably there are still significant peaks at the extremes. The necessity of this might stem from the non-gaussian properties of the input. The fat tails, which are partly due to the high values of z-coordinates in this specific dataset, tend to lie in the saturation region. At the same time, the input distribution has a much sharper peak than a normal distribution. We can attribute this to the influence of the input matrix, whose nonzero elements are uniformly distributed between -1 and 1. This peak starts to reappear in the distribution of reservoir states when a is tuned down and to negatively influence performance. Thus, the optimal choice of a has to balance a tradeoff between these two harmful influences. Furthermore, this points to the possibility of more sophisticated methods to actually remove these obstacles without destroying relevant features of the original data or the complexity of the RC model.

With all these viewpoints in mind, the rest of this chapter will be an empirical study of the effect of this nonlinear scaling parameter on prediction quality quantified by the measures introduced in section 2.3. To really understand it, we try to find its optimal value for for a variety of different systems used as training data and with large statistics in different trajectories of these systems and random networks.

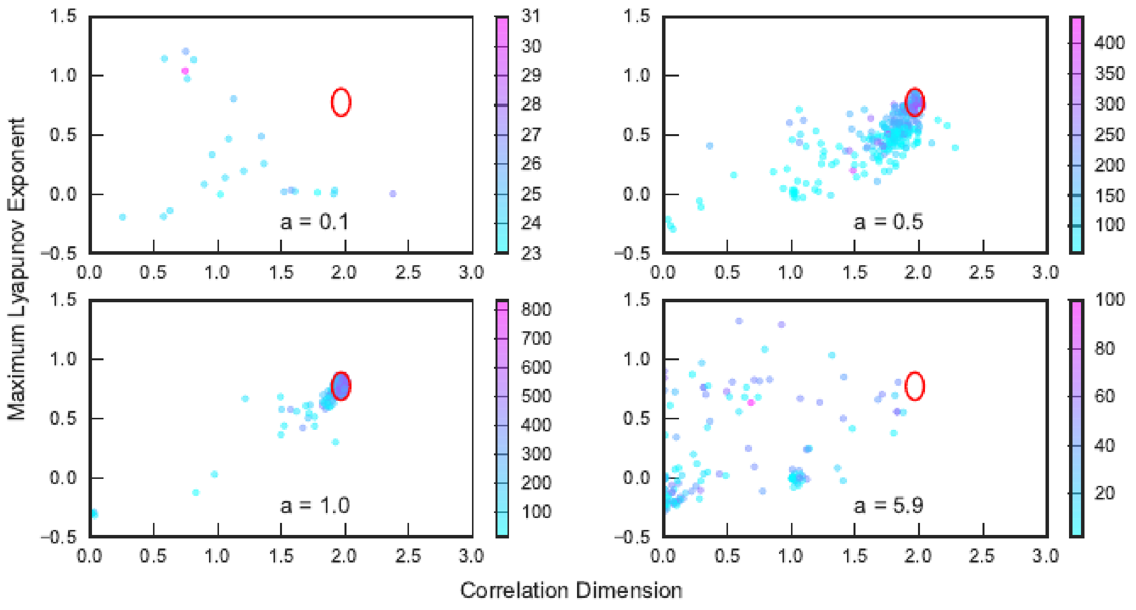


Figure 3.4: Largest Lyapunov exponent scattered against correlation dimension for different values of the nonlinear scaling parameter a based on $N = 300$ realization each. The colours denote the forecast horizon of the predictions and the red ellipses show the three σ errors of the correlation dimension ($\sigma = 0.024$) and the largest Lyapunov exponent ($\sigma = 0.039$) calculated from simulations of the actual system.

3.2 Results

3.2.1 Analysis of the Modified Lorenz

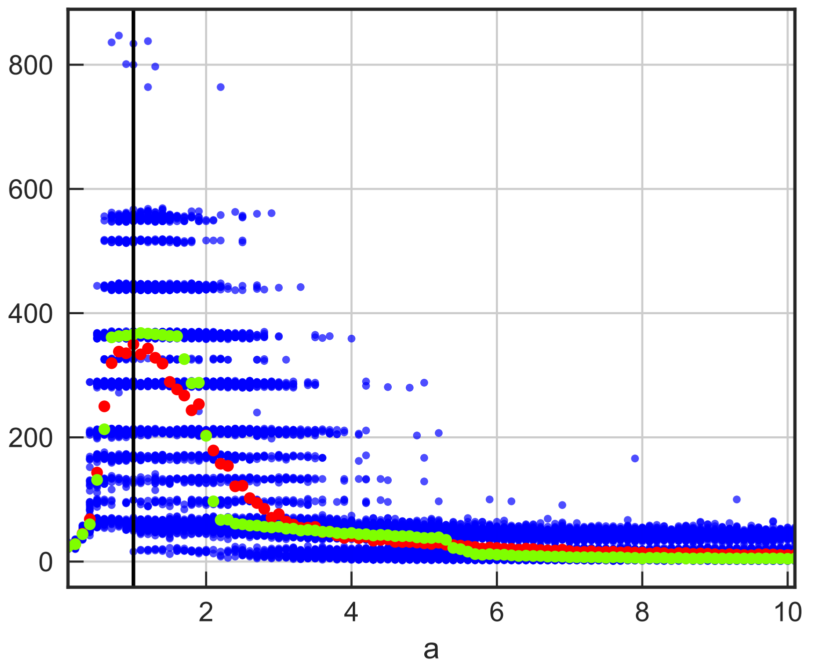
We simulated $N = 300$ realizations for different values of a. We then evaluated the forecast horizon as well as the long-term climate for each realization. The bottom right plot in figure 3.4 shows the largest Lyapunov exponent scattered against the correlation dimension for the modified Lorenz system. The results are based on the above described default setup with the nonlinear scaling factor set to $a = 5.9$. The red ellipse shows the three σ errors of the correlation dimension and the largest Lyapunov exponent. Those are calculated from simulations of the actual equations of the Lorenz system for $N = 500$ different initial conditions. We can clearly see that for $a = 5.9$ all points are widely spread outside this error ellipse, and are therefore to be classified as bad predictions. This is because they do not well resemble the long-term climate. While some realizations lead to meaningful values for the largest Lyapunov exponent, the correlation dimension is badly reconstructed in particular.

To find the optimal value for a, we systematically analyzed multiple realizations for a number of different values of a between 0 and 10. In this case we varied the network, but not the training or test data. This is shown in Fig 3.5a, where the blue points correspond to the forecast horizon of the single realizations. In addition, the red and green dots represent the average and median value across all realizations for a given value of a. There is a large variability in the forecast horizon, especially for a-values close to the optimum. This is in line with the results of [23] and [53]. Further, the blue points tend to accumulate at specific forecast horizons with varying but mostly similar distances. They likely correspond to points in the predicted trajectory, which are either locally especially unstable, like the center of the

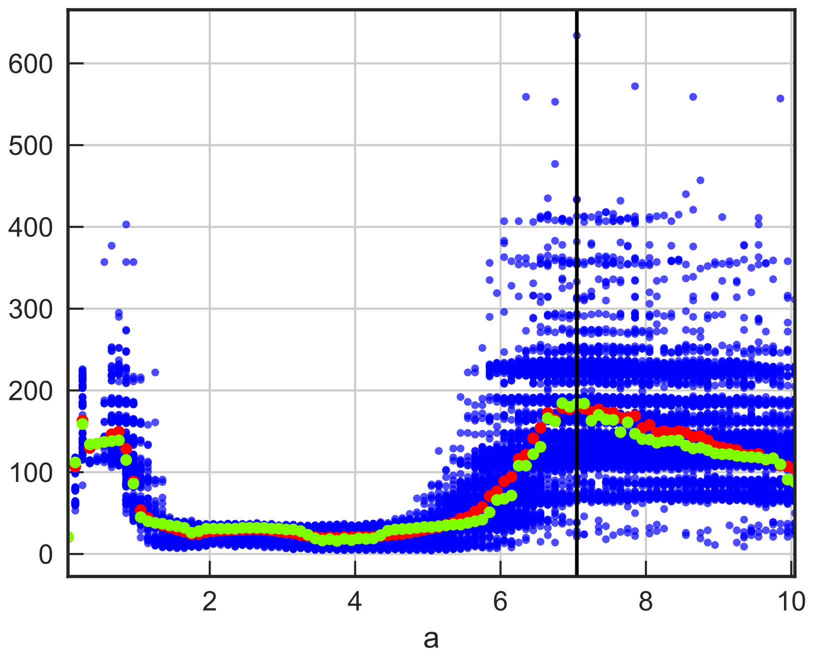
Lorenz attractor or to which the forecast horizon is especially sensitive because of its cuboid shape. The latter becomes relevant at the maxima of the trajectory in any of the coordinates. We determine the optimal value for a such that the average is maximized. This leads to an optimal value of around $a = 1.0$, which is in line with our expectation given that we carried out a hyperparameter optimization in the beginning. For validating the above arguments, we turn back to figure 3.4. The bottom left plot shows the results for the optimal choice of a, where many outliers, and thus bad predictions disappeared. Moreover, there is now a compact cloud of points around the error ellipse, and therefore the overall prediction quality is significantly better as compared to the case $a = 5.9$ in the bottom right plot. In contrast, setting $a = 0.1$ and $a = 0.5$ as shown in the top plots leads to a complete breakdown of the prediction ability of the system. The reason that one can see only a few points in the top left plot is the following. The prediction quality for $a = 0.1$ completely collapses in most cases such that we obtain NaN results for our calculations of the largest Lyapunov exponent. This happens in cases where the prediction jumps between multiple points in a cyclical fashion. Instead of the desired strange attractor it follows some stable periodic orbit. Consequently, this leads to division by zero and generally only occurs for unsuitable parameter choices – in this case for too small values of a. As both examples in the top plots correspond to arguments of the activation function being in the linear regime of the hyperbolic tangent, this demonstrates that nonlinearity in the activation function is essential for predicting complex nonlinear systems. Besides the results for the reproduction of the long-term climate, we also show the forecast horizon encoded in the colors of the points. Equivalently, the longest forecast horizon can be achieved by choosing the optimal value for a, whereas smaller or larger values both lead to worse results. Another interesting result is that realizations, which well resemble the long-term climate have a higher forecast horizon than those failing to properly reconstruct the climate.

3.2.2 General Relationship between $a_{optimal}$ and σ

In addition to the Lorenz system, we carried out the same analysis for other nonlinear complex systems such as the Chua circuit, the Rössler system and other autonomous dissipative flows as summarized in section 2.1. Figures analogous to figure 3.5a can be found in the appendix. They mostly show a very similar functional form to the modified Lorenz system. The most significant exception will be discussed in section 3.2.3. Figure 3.6 shows the results for their optimal values of a scattered against the standard deviation of the input. In addition, we also constructed combinations of the systems used, in order to fill the gap in between the standard deviations of the Halvorsen model (0.53) and the modified Lorenz system (1.56). We can clearly see that there is a relationship between the optimal a and the input standard deviations. This intuitively makes sense, since the dynamical regime of the hyperbolic tangent needs to be at a different range for different distributions. Surprisingly, this seems to dominate over effects of other system-specific properties. Therefore, as a rule of thumb, the optimal value for the nonlinear scaling parameter is given by $a_{opt} = c/\sigma(\mathbf{W}_{in}\mathbf{u})^b$ with $b = 0.80$ and $c = 1.22$ determined by the fitted red curve. This provides a good starting point for the hyperparameter optimization. However, it is always recommended to run a system specific analysis as shown in Fig 3.5a.
We also looked at a larger parameter range for a and found that the average forecast horizon is monotonically declining for values of $a > 10$, which are not shown here. Equivalent results for a_{opt} are gained by carrying out the same analysis for the above mentioned systems based on the reproduction of the correlation dimension. In particular the results for the Chua circuit indicate that there is a significant potential for system specific optimizations.



(a)



(b)

Figure 3.5: Forecast horizon of a) the modified Lorenz system and b) the Chua circuit plotted for different values of the nonlinear scaling parameter a with $N = 300$ realizations for each a (blue). The average (red) as well as the median (green) value are shown for each value of a. The black horizontal line marks the optimal choice for a.

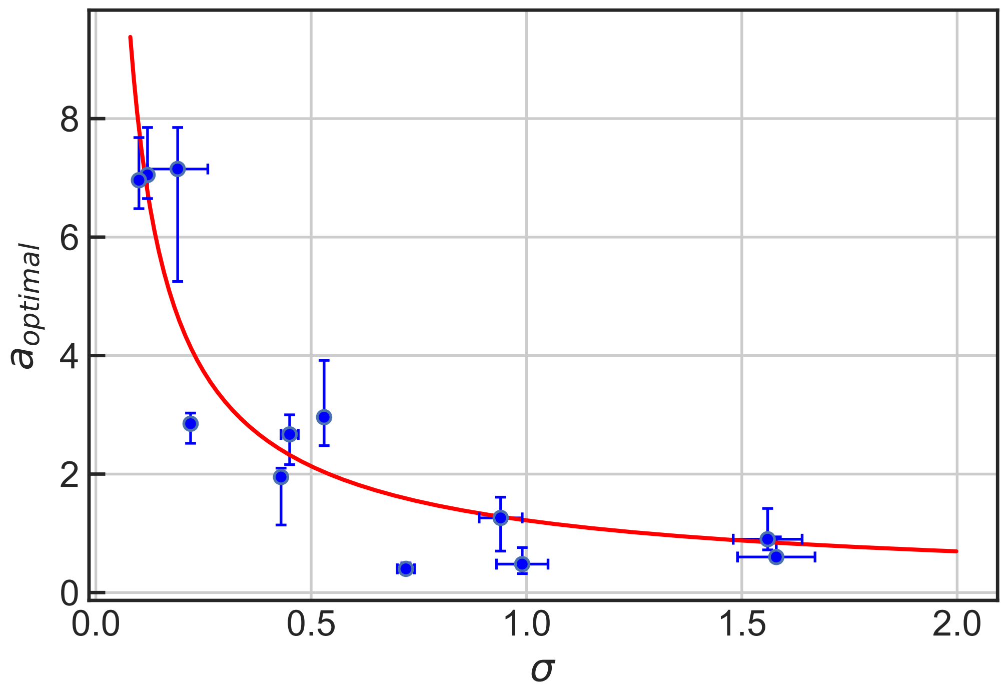


Figure 3.6: Blue dots: Optimal value for the nonlinear scaling parameter a – based on the maximum of the average forecast horizon over all realizations for a given a – plotted against the standard deviation σ of the input data. The vertical bars denote the range of values for a, where the average forecast horizon is up to 10% lower than for the optimal a, while the horizontal bars represent the standard error. The red line represents a fit through the points. Dynamical systems from left to right: Rabinovich-Fabrikant equations, Chua circuit, Complex Butterfly attractor, Thomas' cyclically symmetric attractor, Rössler system, Rucklidge system, Halvorsen model, *blended system:* 0.4*Modified Lorenz system + Halvorsen Model, *blended system:* 0.5*Modified Lorenz system + 3*Rabinovich-Fabrikant equations, *blended system:* Rössler + 2*Rucklidge system, Modified Lorenz system and Chen system

3.2.3 Local Nonlinearities

One example for behavior not following this heuristic is the Chua circuit. The Chua circuit yields optimal predictions not only for $a = 7.05$ following the above introduced rule of thumb, but also shows another peak for small values of around $a = 0.75$ as shown in Fig 3.5b. This might be related to the fact that the equations of the Chua attractor only have a local nonlinearity at $x = \pm 1$, making the linear regime very successful anywhere else. Further evidence for this hypothesis is a smaller, but similar peak found for the Complex Butterfly attractor, whose nonlinearity is only local as well. At the same time the Thomas attractor also exhibits a weak peak at this point and an even weaker version can be found for example in the Rabinovich-Fabrikant equations. Both of these have global nonlinearities in their equations. To further explore this idea we repeated the analysis on two additional systems of equations with local nonlinearities: The Double Scroll attractor and Simplest Piecewise Linear Flow. We find the characteristic peak in both of their forecast horizon curves higher than for any of the fully nonlinear systems.

This leads us to believe that this behavior is strongly related to though not fully explained by local nonlinearities (or piecewise linearities). Another clue that there is more to the story is the complex fine structure the peaks reveal under closer examination. They do not nearly extend all the way to zero as one would expect, if approximation of linearity was the only relevant factor. Also, they show a separation into multiple peaks and a generally complicated functional form that is much more idiosyncratic to the specific data set than the part that follows the heuristic of section 3.2.2.

Chapter 4

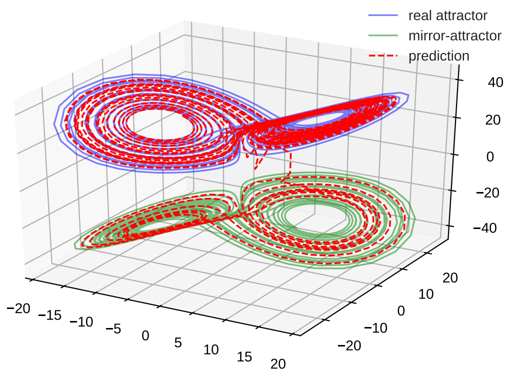
Symmetries in the Reservoir Equations

The contents of this chapter have mostly been published in [58].

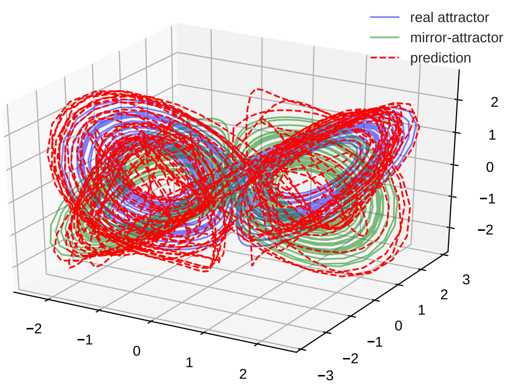
4.1 The Mirror-Attractor

As already mentioned in Sec 2.2 the simple ESN design we introduced back then sometimes shows very severe failures as illustrated in figure 4.1a and figure 4.1b. When predicting the Lorenz attractor, the prediction sometimes jumps to an inverted version of the training dataset, which we call *mirror-attractor*. We observe this phenomenon quite regularly and for long predictions there are cases where the prediction later switches back to the original attractor or even jumps back and forth. This unexpected behavior is obviously not just a random fluke or statistical outlier. This raises the question of how frequently this happens exactly and under which circumstances.
To investigate the severity of the problem, we created 1000 trajectories on the Lorenz attractor and used them to train 1000 ESNs with different networks and input matrices. The hyperparameters we used were the same as those we used in chapter 3 for the modified Lorenz. Each one then made a prediction of 500000 timesteps. Our goal was to to measure the frequency of jumps to the mirror-attractor. Because the Lorenz attractor lies completely above the $z = 0$ plane, we start by recording any crossing of this plane (*zero crossing*). To distinguish between actual jumps and cases, where the prediction simply runs a little bit outside of the actual attractor, we look at the mean value of the z-coordinate for the part of the trajectory on this side of the plane. For a decent attractor reconstruction this should be about ± 23 respectively. Indeed we immediately observe that most zero crossings fulfill this condition. However, this would exclude predictions where the jump back happens very fast. Observation and the results of chapter 3 tell us that for this setup it is extremely rare for the attractor reconstruction to fail severely in any other way than to create a mirror-attractor. Thus, we simply count any zero crossing with a mean z-coordinate more than 15 away from the origin as jump. This measure is of course somewhat arbitrary, but since a definite distinction between a “normal” zero crossing and a very short jump cannot be made from data, some vagueness is unavoidable. Empirically we find this to agree with visual inspection.

Using this we found that 98.5% of zero crossings also lead to an attractor jump. This indicates that crossing this boundary could be what triggers the event in most cases. The average number of jumps in a single trajectory was 30.429, showing that this problem is quite prevalent. At the same time 18% of the predictions did not exhibit a jump at any point. Among



(a)



(b)

Figure 4.1: Failed predictions of the Lorenz system with a simple ESN. Figure a): The data is not preprocessed and the trajectory jumps down to the mirror-attractor. Figure b): Zero-mean, normalized Lorenz data. The trajectory jumps frequently between the original and the mirror-attractor.

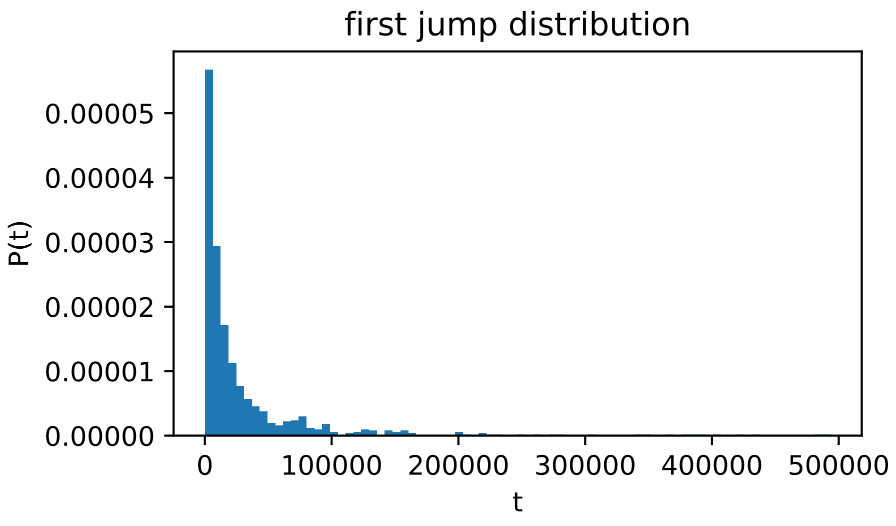


Figure 4.2: Normalized histogram of times of the first jump between real attractor and mirror-attractor predictions of the Lorenz system made with simple ESNs.

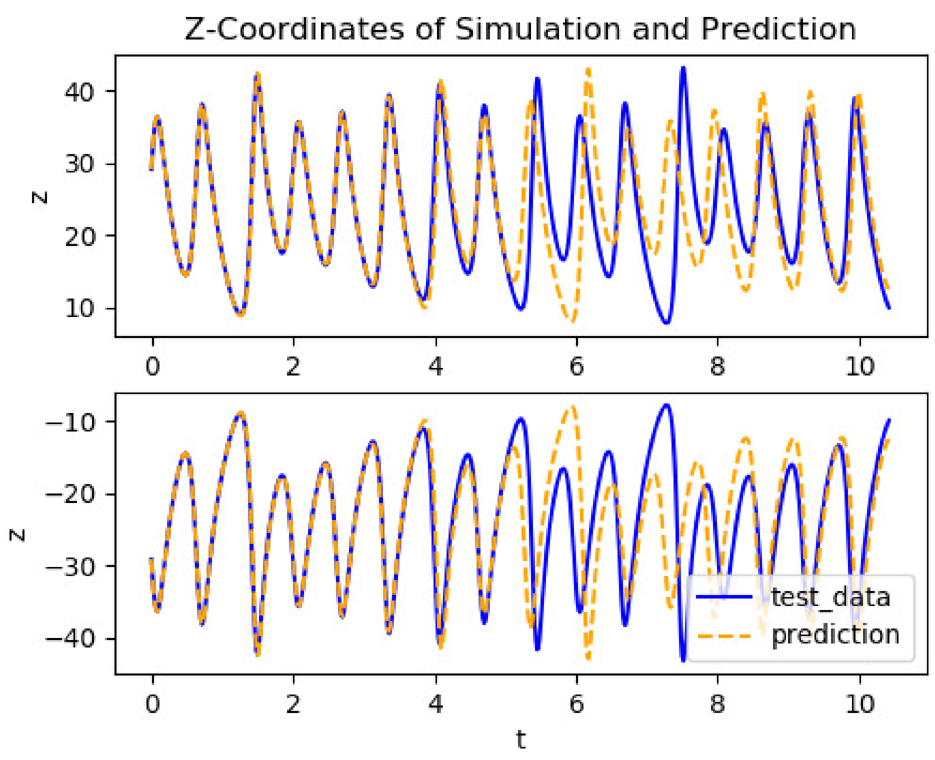


Figure 4.3: Prediction of the z-coordinate after synchronization with original data (upper) and inverted data (lower) for the simple ESN. The t-axis is given in units of Lyapunov times.

those that did, the average time of the first jump was about 31000 timesteps (539 Lyapunov times). The distribution of first jumps is shown in figure 4.2.

While this is concerning, it still allows for decent short-time predictions. We were able to reach average forecast horizons of about 400 timesteps (7 Lyapunov times) after hyperparameter optimization. However, when the data is brought to zero-mean the ability to make accurate predictions largely breaks down. After hyperparameter optimization we get a forecast horizon of about 90 timesteps (1.6 Lyapunov times). Since the two attractors overlap, the prediction jumps between them very frequently as we can see in figure 4.1b. Sometimes it even travels outside of both for a short time. Since this kind of preprocessing is considered good practice in machine learning and usually leads to better results, this shows a severe failure of the method.
Difficulties in predicting the Lorenz system, which is a widespread test case for ESNs, when using this kind of setup have been noted by previous studies. They were linked to the symmetry of the Lorenz equation under the transformation $(x, y, z) \to (-x, -y, z)$. This seems to go back to Lu et al. [59], where this was correctly identified as a problem for the specific case of using the reservoir as an observer. However, we observe these problems in prediction tasks and with other datasets as well, showing that this was only an instantiation of a general property of the simple ESN. In the next section we will give a theoretical explanation of this phenomenon by means of mathematical analysis. We will show that it is not a symmetry of the Lorenz equations, but a symmetry of the ESN that causes it.

4.2 Theory

Assume $\mathbf{r}_0 = \mathbf{0}$ w.l.o.g. because of the echo state property. Let us now analyze what happens, when instead of the original training sequence $\mathbf{x}^{train} = \{\mathbf{x}_0, ..., \mathbf{x}_{T_{train}}\}$ we use its inverted version $-\mathbf{x}^{train} = \{-\mathbf{x}_0, ..., -\mathbf{x}_{T_{train}}\}$ to train the readout matrix. In the following we use $\mathbf{x}$ instead of $\mathbf{x}^{train}$ for simplicity.

$$\mathbf{r}_0(-\mathbf{x}) = \mathbf{0} = -\mathbf{r}_0(\mathbf{x}) \tag{4.1}$$

$$\begin{aligned} \mathbf{r}_1(-\mathbf{x}) &= \tanh(-\mathbf{W}_{in}\mathbf{x}_n) & (4.2)\\ &= -\tanh(\mathbf{W}_{in}\mathbf{x}_n) & (4.3)\\ &= -\mathbf{r}_1(\mathbf{x}) & (4.4) \end{aligned}$$

This serves as the base case for our mathematical induction. We follow up with the induction step. Assume

$$\mathbf{r}_t(-\mathbf{x}) = -\mathbf{r}_t(\mathbf{x}) \tag{4.5}$$

Then

$$\begin{aligned} \mathbf{r}_{t+1}(-\mathbf{x}) &= \tanh(\mathbf{A}\mathbf{r}_t(-\mathbf{x}) - \mathbf{W}_{in}\mathbf{x}_t) & (4.6)\\ &= -\tanh(\mathbf{A}\mathbf{r}_t(\mathbf{x}) + \mathbf{W}_{in}\mathbf{x}_t) & (4.7)\\ &= -\mathbf{r}_{t+1}(\mathbf{x}) & (4.8) \end{aligned}$$

Overall we get $\mathbf{r}^{train}(-\mathbf{x}) = -\mathbf{r}^{train}(\mathbf{x})$. So the dynamics of the reservoir only changed sign and are otherwise unaffected. This is a consequence of the antisymmetry of the hyperbolic tangent.
Obviously, because of the linearity of the readout, we also get

$$\mathbf{y}_t(-\mathbf{x}) = -\mathbf{W}_{out}\mathbf{r}_t(\mathbf{x}) = -\mathbf{y}_t(\mathbf{x}) \tag{4.9}$$

And finally

$$\begin{aligned} \mathbf{W}_{out}(-\mathbf{x}) &= (\mathbf{r}^T(-\mathbf{x})\mathbf{r}(-\mathbf{x}) + \beta 1)^{-1}\mathbf{r}^T(-\mathbf{x})(-\mathbf{x}) & (4.10)\\ &= (\mathbf{r}^T(\mathbf{x})\mathbf{r}(\mathbf{x}) + \beta 1)^{-1}\mathbf{r}^T(\mathbf{x})\mathbf{x} & (4.11)\\ &= \mathbf{W}_{out}(\mathbf{x}) & (4.12) \end{aligned}$$

From this we can conclude three things: Firstly, training the simple ESN on $\mathbf{x}$ and training on $-\mathbf{x}$ is completely equivalent. The resulting parameters in $\mathbf{W}_{out}$ are the same.
Secondly, it follows that for tasks other than prediction, where the output is not fed back into the reservoir, changing the sign of the input sequence always leads to a change in the sign of the output sequence. It is not possible to map both of them to the same non-zero output or differentiate them in any way but the sign. It is therefore abundantly clear that the simple ESN is not universal.
Finally, in the case of a prediction task, where the loop is closed, the simple ESN will always exhibit a mirror-attractor. It is identical to the real attractor it learned, only with inverted coordinates. Of course this is not a problem in the special case, where the original attractor is pointwise symmetric at the origin itself. Otherwise this can clearly influence the prediction of a trajectory on the original attractor. In cases where they overlap they are incompatible. When they do not overlap, but are close enough to each other jumps may become possible.

In figure 4.3 this is demonstrated by comparing the predictions of an already trained simple ESN after being synchronized with additional Lorenz data either unchanged or inverted. We can see that the prediction of inverted data is simply the inversion of the prediction of the original data, just as expected from theory.
For a jump to happen, the reservoir has to arrive at a state that matches better with the mirror-attractor than with the real one. Since the reservoir has memory it is not obvious how fast input data from the phase-space region of the mirror-attractor can actually create this effect. As stated above we found that crossing the zero in the z-direction leads to a jump in 98.5% of cases. We further observed that inverting the input in a single timestep was reliably enough to push the prediction on the mirror-attractor. This implies a strong sensitivity to the input data with regards to inducing jumps. Independently of symmetries a different reservoir design, e.g. using leaky integrator neurons, might in principle inhibit this effect by increasing the memory.
It is clear that this kind of symmetry creates a significant limitation for the simple ESN. We can therefore easily explain the previous problems with this kind of approach as well as the so far mostly empirical success of some methods combating them. In many recent publications the readout was extended with a some kind of nonlinear transformation. The empirical advantage of this has been explored without theoretical explanation by Chattopadhyay et al. [10]. Typically quadratic terms are included in the readout (see Sec. 4.3.2). This was to our knowledge originally introduced by Lu et al. [59] in order to specifically solve the problem of the symmetry of the Lorenz equations, when using the ESN as an observer. It has since been used successfully in many more general cases without theoretical explanation. From our analysis it is now clear that this readout breaks the antisymmetry of the ESN as a whole, which is completely independent of any symmetries of the input data.
A similar analysis can be fruitful on many different designs of ESN. For example in a recent Paper by Carroll and Pecora [22] the following update equation was used:

$$r_i(t+1) \quad = \quad \alpha r_i(t) + (1-\alpha)\tanh(\sum_{j=1}^{M} A_{ij} r_j(t) + w_i x(t) + 1) \tag{4.13}$$

Where w_i are the elements of what they call *input vector*. Empirically they found that the performance suffered for $w_i = 1\ \forall i$ compared to $w_i \in \{+1, -1\}$. We can now explain this. A closer look reveals that there is a symmetry under the transformation $x(t) \to -x(t) - \frac{2}{w}$ for any constant $w_i = w\ \forall i$. We expect this to lead to a mirror-attractor similar to what we found for the simple ESN. The only difference is that the symmetry is not around the origin in this case. As soon as w_i takes on different values for different i this symmetry is broken. This method of getting rid of the mirror-attractor by breaking the symmetry will be explored in the next section.
However, while we will focus on the symmetry, since we can show its existence with a high degree of mathematical rigorosity, the use of only odd activation functions is problematic in a more general sense. We can get an intuition for this by regarding the extreme case of an ESN without recurrent connections. Then every node is simply a function of the current input signal and training the readout means finding the optimal linear combination of these to approximate a given function. The nodes differ by the random parameters introduced through the elements of $\mathbf{W}_{in}$ and $\mathbf{A}$. In general we want this set of functions to approximate a basis in the corresponding function space to be as powerful as possible. We immediately see some issues. Firstly, the ESN will always map an input of zero to an output of zero. Secondly, it becomes obvious that, since the readout simply consists of odd functions, any function of the input we can approximate will also be odd. Ultimately this model just consists of a sum of hyperbolic tangent functions with different scaling, which is quite obviously far from universal. Indeed we observed that the simple ESN mostly loses its capacity to learn when ρ

is set to zero.
Obviously, the story is more complicated outside of this special case, as evidenced by the relative success of the simple ESN. It is still true that a sequence of zero inputs will be mapped to a sequence of zero outputs. This is technically a special case of the symmetry described above. The second issue we found in the $\rho = 0$ case is more complicated now and a proper understanding would require a rigorous mathematical analysis of the nodes as functions of the whole input sequence instead of only a single input and linear combinations of them in an appropriate Hilbert space. We will not attempt this here, although some work has been done in this direction by Dambre et al. [13].
However, if we understand the nodes as functions of the concatenation of input datum and reservoir state $\tilde{\mathbf{x}} = \{r_1, ..., r_N, x_1, .., x_d\}$ things become somewhat simpler. We basically get the same picture as for the $\rho = 0$ case except for higher dimensional argument in the activation functions. This makes it easy to see that the antisymmetry of the simple ESN corresponds to the nodes being an incomplete basis of the Hilbert space. However, it obscures the complex temporal dynamics and the memory of the reservoir.

4.3 Ways to Break the Symmetry

To better understand what is the best way to break the harmful antisymmetry in the simple ESN we test four different designs. There are two main ways of approaching the problem. We can either break the symmetry in the reservoir, e.g by changing the activation function, or in the readout. When choosing the latter option, equation 4.5 still holds. The dynamics of the reservoir still do not change meaningfully with the sign of the training data. Only during prediction does the influence of the readout actually come into play.
We use two designs following each approach.

4.3.1 Output Bias

This is one of the simplest ways to break the symmetry. The readout is changed by using $\tilde{\mathbf{r}} = \{r_1, r_2, ..., r_N, 1\}$. Effectively this leads to

$$\mathbf{y}_t = \mathbf{W}_{out}\tilde{\mathbf{r}}_t = \tilde{\mathbf{W}}_{out}\mathbf{r}_t + \mathbf{b} \tag{4.14}$$

where $\mathbf{b} \in \mathbb{R}^d$ is called *bias-term* and is fixed in the Linear Regression. This very basic extension of Linear Regression is often already seen as good practice. Formally this breaks the symmetry.

$$\mathbf{y}_t(-\mathbf{r}_t) = -\tilde{\mathbf{W}}_{out}\mathbf{r}_t + \mathbf{b} = -\mathbf{y}_t(\mathbf{r}_t) + 2\mathbf{b} \tag{4.15}$$

we note however that this way there are only d parameters to represent the difference under sign-change. Furthermore, we observe that for networks with more than a few nodes the magnitude of the bias terms is of the same order as the magnitude of the other parameters in the readout. Thus, the influence of the bias is relatively small. We will see in the following sections if this is enough to effectively break the symmetry.

4.3.2 Lu Readout

As previously mentioned, a quadratic extension of the readout has recently become popular after its introduction by Lu et al. [59]. We use two different variants of this approach. In its

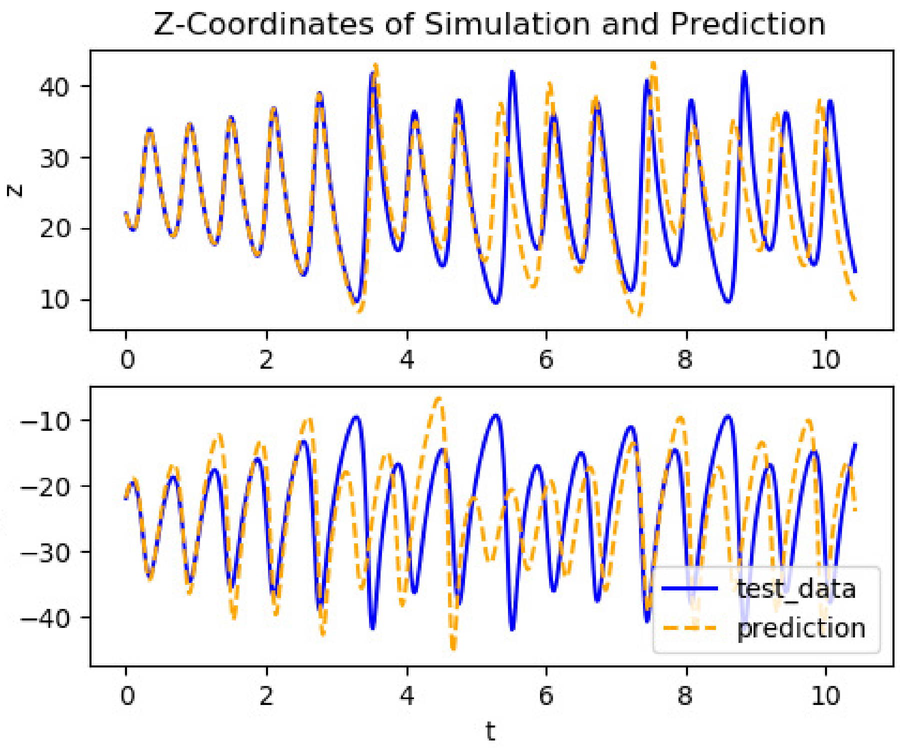


Figure 4.4: Prediction of the z-coordinate after synchronization with original data (upper) and inverted data (lower) for the ESN with output bias. The t-axis is given in units of Lyapunov times. The prediction of inverted data is perturbed but still in the mirror-attractor.

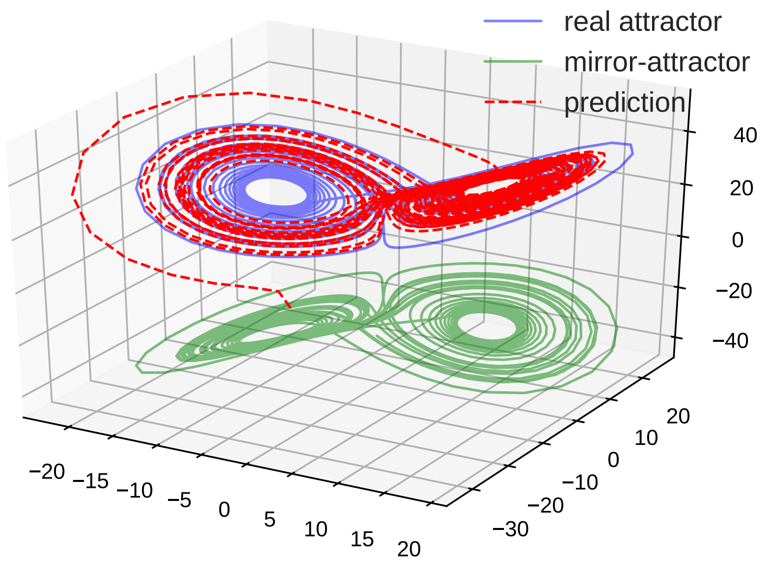


Figure 4.5: Prediction after synchronizing with inverted data for the ESN with extended Lu readout.

most powerful version it consists of using $\tilde{\mathbf{r}} = \{r_1, r_2, ..., r_N, r_1^2, r_2^2, ..., r_N^2\}$ in the readout.

Effectively we get

$$\mathbf{y}_t = \mathbf{W}_{out}\tilde{\mathbf{r}}_t = \mathbf{W}^1_{out}\mathbf{r}_t + \mathbf{W}^2_{out}\mathbf{r}^2_t \ . \tag{4.16}$$

Where $\mathbf{W}_{out} \in \mathbb{R}^{d\times 2N}$ can be divided in $\mathbf{W}^1_{out}$ and $\mathbf{W}^2_{out} \in \mathbb{R}^{d\times N}$ And

$$\begin{aligned} \mathbf{y}_t(-\mathbf{r}_t) &= -\mathbf{W}^1_{out}\mathbf{r}_t + \mathbf{W}^2_{out}\mathbf{r}^2_t & (4.17)\\ &= -\mathbf{y}_t(\mathbf{r}_t) + 2\mathbf{W}^2_{out}\mathbf{r}^2_t \ . & (4.18) \end{aligned}$$

The number of parameters that represent the difference under sign-change is $d \times N$. In the following we will call this *extended Lu readout.* We use this in order to test the full potential of this approach. However, the higher number of parameters in the output matrix makes a quantitative comparison to other approaches unfair. For this purpose we also test a second version.
In the original work by Lu et al. only half of the nodes are squared in the readout and each is only used either in its linear or in its quadratic form. This can be achieved with a transformation of the reservoir state like $\tilde{\mathbf{r}} = \{r_1, r_2^2, r_3, r_4^2, ..., r_{N-1}, r_N^2\}$, where we assumed N to be even. We call this *Lu readout* or *regular Lu readout.* This way the number of parameters is unaffected, which makes a fair comparison with the other methods possible.
On the first glance this means giving up the idea of linear combination we established at the end of section 4.2. However, we can stay in that framework by understanding this readout as linear combination after squaring of the nodes. This way we see that we have actually introduced even functions into the basis used in the readout.
We note that this does not solve the issue of always mapping zero to zero, which is however only relevant in a very exotic case.

4.3.3 Input Shift

This design for an ESN has been proven to be universal by Grigoryeva and Ortega [36], which naturally excludes any problems with symmetry. This is also recommended in “A practical guide to Applying Echo State Networks” by Lukoševičius [60].
For this design the activation function of the simple ESN is extended by including a random bias term in every node of the reservoir. It can be written as a random vector $\gamma \in \mathbb{R}^N$ and gives the following new update equation:

$$\mathbf{r}_{t+1} = \tanh(\mathbf{A}\mathbf{r}_t + \mathbf{W}_{in}\mathbf{x}_t + \gamma) \ . \tag{4.19}$$

The readout is unchanged from the simple ESN. Unlike the first two methods this breaks the symmetry in the reservoir itself.
We draw the elements of γ uniformly from $[-s_\gamma; , s_\gamma]$ where s_γ is a new hyperparameter to be optimized. This was a somewhat arbitrary choice for simplicity. In principle we could instead use a normal distribution, the distribution of the training data, etc.
Another clear advantage of this approach is visible if we ignore the recurrent connections again. Even with this severe simplification the ESN with input shift would constitute a working variation of a feedforward NN. This kind of feedforward NN with a single random layer is known as Extreme Learning Machine [61, 62] and has been shown to be highly successful.

4.3.4 Mixed Activation Functions

Another way of breaking the symmetry directly in the reservoir is to replace some of the odd tanh activation functions with even functions. Mixing in even functions can give us access

ESN design	F.H. in Δt (τ_λ)	$\lambda \pm \sigma$	$\nu \pm \sigma$
Simple ESN	90.9(1.6)	0.2 ± 0.2	1.6 ± 0.5
Output Bias	149.7(2.6)	0.3 ± 0.2	2.0 ± 0.5
Mixed Activations	538.2(9.4)	0.87 ± 0.03	1.97 ± 0.13
Input Shift	629.3(11.0)	0.87 ± 0.02	1.978 ± 0.008
Lu Readout	558.4(9.7)	0.87 ± 0.04	1.96 ± 0.17
Ext. Lu Readout	631.3(11.0)	0.87 ± 0.02	1.978 ± 0.008
Test Data	∞	0.87 ± 0.02	1.978 ± 0.008

Table 4.1: Performance of the different ESN designs on zero-mean Lorenz data. Comparison to the original Lorenz data in last row. Forecast horizon (F.H.) given in units of timesteps and Lyapunov times in brackets.

to the whole function space as any function can be divided in an even and an odd part. As even function we simply used $\tanh^2$. We assigned half of the nodes connected to each input dimension to be even nodes where this activation functions is used.
This is closely related to the Lu readout. The somewhat subtle difference is that the squared nodes in this case are also directly involved in the dynamics of the reservoir, even in the open loop case, e.g. during training. This means the symmetry in equation 4.5 is actually broken. This promises a richer behavior than in the former method. However, this also does not allow for different responses to zero as input. Thus, while we can approximate a bigger subspace of the space of functions mapping $\{r_1, ..., r_N, x_1, .., x_d\}$ to the output, we still have at least this one hole. Since a finite set of nodes can not be a complete basis and we are dealing with approximations anyway, the practical implications of this are not a priori clear.

4.4 Results

4.4.1 Predicting The Mirror-Attractor

To test the ability of the four methods to break the symmetry of the simple ESN, we tried to force them to predict the mirror-attractor of the Lorenz equations after being trained with regular data. If the symmetry is truly broken, we expect this prediction to fail completely, indicating that the ESN did not learn anything about the mirror-attractor.
To accomplish this we trained our ESNs with regular Lorenz data and then synchronized it with the inverted next 500 timesteps of the simulation. We then measured the forecast horizon of the prediction in regards to the (also inverted) test data. For comparison, we also looked at the prediction after synchronization with the same data without inversion.
In figure 4.4 we see the behavior of the ESN with output bias. It differs from before in that the prediction of the mirror-attractor is not simply the inversion of the regular prediction as for the simple ESN in figure 4.3. However, even though it is generally a worse prediction, it clearly follows the inverted trajectory. The ESN has still learned a slightly perturbed version of the mirror-attractor.
When using input shift, Lu readout or mixed activations, we never observed a prediction staying in the vicinity of the mirror-attractor. Most of them instead leave it immediately and quickly converge to the real Lorenz attractor as in figure 4.5. In some cases the trajectory finds some other fixed point instead, but it never stays in the mirror-attractor. Qualitatively we get the same behavior when using mixed activations or input shift instead.
Furthermore, we made 1000 predictions of the mirror-attractor with all four designs while varying the network and the starting point of the training data. The distribution of forecast

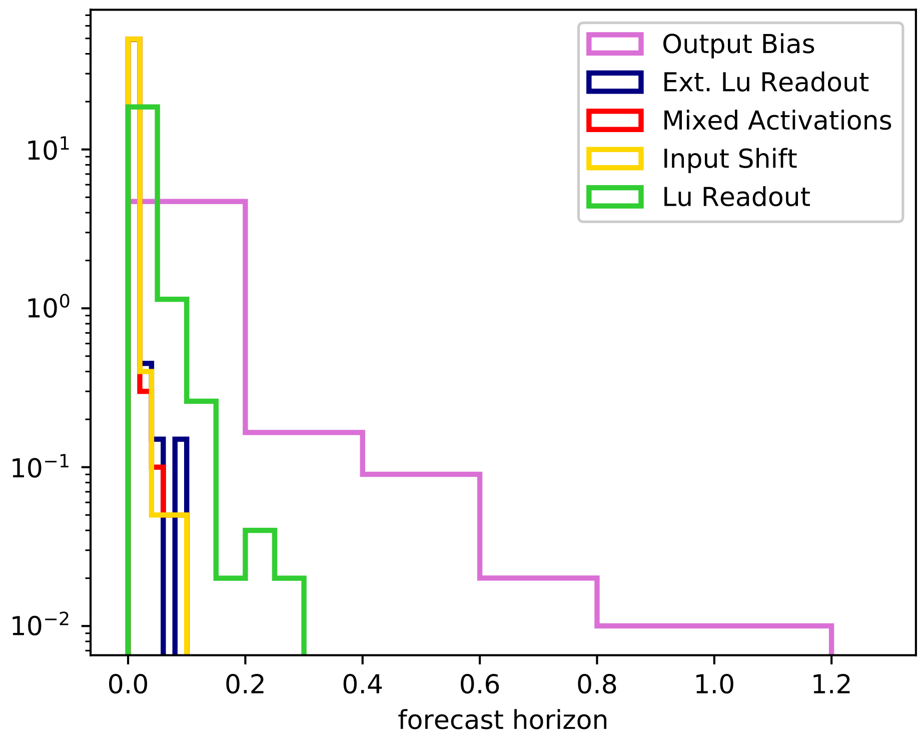


Figure 4.6: Normalized histogram of the forecast horizon (in units of Lyapunov times) with respect to the inverted test data after synchronizing with inverted training data for the four different symmetry-breaking designs.

horizons is shown in figure 4.6. The output bias clearly sticks out as the only method showing the ability to predict the mirror-attractor. Some realizations reach forecast horizons up to one Lyapunov time while the other methods never go beyond 0.1 Lyapunov times corresponding to only O(1) timesteps. An exception is the regular Lu readout, where the prediction tends to stay on the mirror-attractor slightly longer than for input shift, extended Lu readout and mixed activation functions. However, all of these trajectories converge to the original attractor afterwards.
We also tested the rate of jumps between the attractors for the ESN with output bias analogously to the simple ESN in Sec. 4.1 by making 1000 predictions with 500000 timesteps. Network and training data were varied for each realization. This time 30.5% of them did not show any jump. For the others the average time of the first jump was after about 33000 timesteps (574 Lyapunov times). 95.2% of times the z-coordinate crossed zero it lead to a jump. This might indicate a small improvement, but the fundamental problem has not been solved by the output bias.
We did not observe any jumps when we used the other methods.

4.4.2 Zero-mean Lorenz

To further compare the performance of the different ESNs, we test the ability to learn and predict Lorenz data where the mean has been shifted to the origin. As already discussed, this leads to overlap between real and mirror-attractor. So even though this preprocessing is usually preferred, it makes the simple ESN's problems with antisymmetry especially severe. We also rescaled all data to have a standard deviation of 1.
To get reliable quantitative results, we first carried out a hyperparameter optimization on

this task for every design used. We used a grid search with 100 realizations for each point in parameter space. The best parameters are chosen on the basis of the highest average forecast horizon. Further details and results can be found in the appendix.
With the optimized hyperparameters we created 1000 realizations for each design and measured forecast horizon, largest Lyapunov exponent and correlation dimension. To accurately represent the climate we used predictions with a length of 20000 timesteps. The results are compiled in table 4.1 and figure 4.7 and 4.8.
The simple ESN's forecast horizon consistently lies in a region below 3.5 Lyapunov times with a mean of 1.6 and similarly to the first task the output bias offers a noticeable but small improvement with a mean of 2.6 Lyapunov times. The other four methods to break the symmetry all seem to work in principle. Their forecast horizons mostly lie between 7 and 14 Lyapunov times. Extended Lu readout and input shift show no significant difference with an average forecast horizon of about 11 Lyapunov times, while mixed activation functions and regular Lu readout show a lower average forecast horizon of 9.4 and 9.6 respectively.
In agreement with the results for the forecast horizon, the simple ESN's climate produces values of largest Lyapunov exponent and correlation dimension far away from the desired region. Again the output bias is only a small improvement. All results for extended Lu readout and input shift lie in the direct vicinity of the target and the mean values and standard deviations match those of the test data within uncertainty. Regular Lu readout and mixed activation functions also reproduce the correct mean values but with much higher variance. This can be attributed to the clear outliers visible in the plots. We note that the results for the climate are less reliable than those for the forecast horizon since we did not optimize the hyperparameters for this task.
In some cases the predicted trajectory diverged completely or got stuck in a fixed point. This

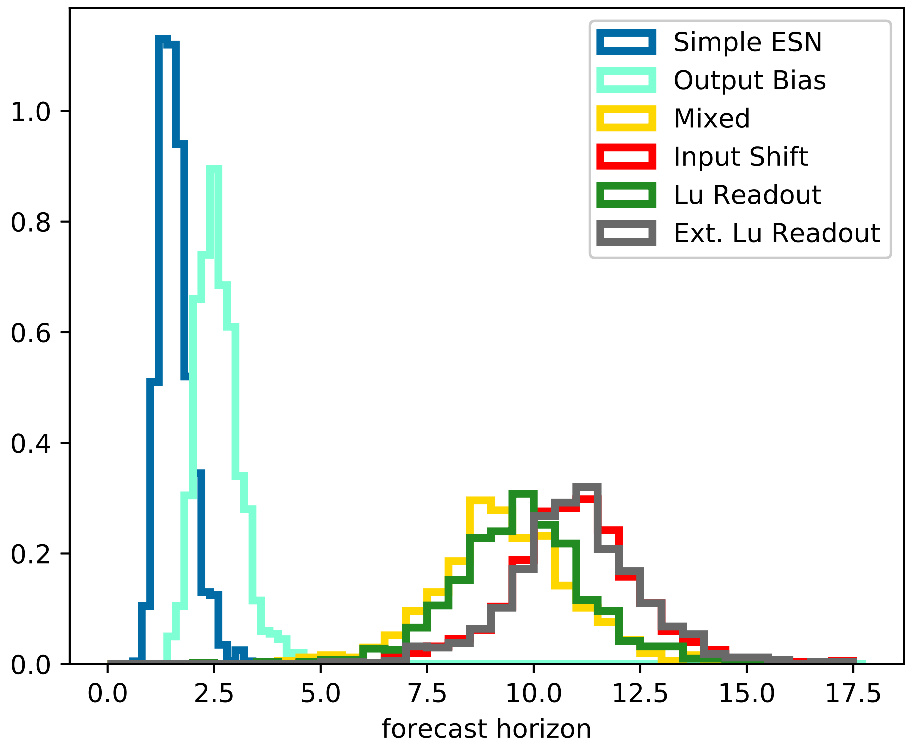


Figure 4.7: Normalized histogram of forecast horizons (in units of Lyapunov times) for normalized, zero-mean Lorenz data with all five variants of ESN. Based on 1000 realizations (varying training data, network and starting point of prediction) for each.

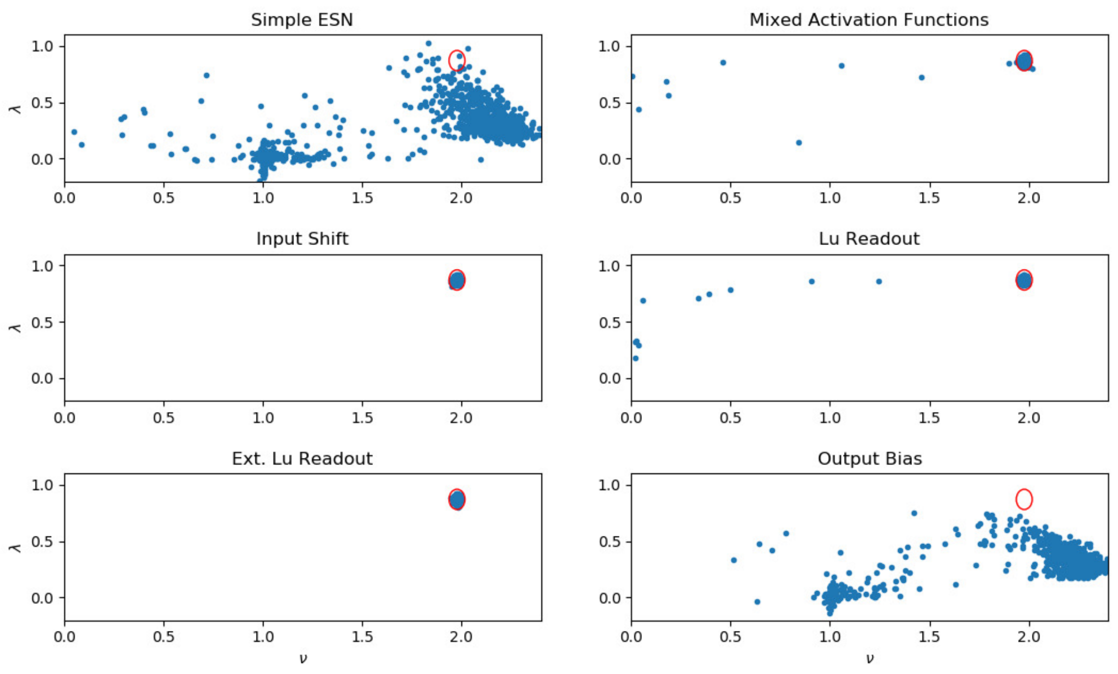


Figure 4.8: Scatterplot of the Largest Lyapunov Exponent against the correlation dimension when predicting zero-mean Lorenz data. Red ellipse corresponds to five times the standard deviation of the test data. Based on 1000 realizations for each setup as in figure 4.7.

made the proper calculation of the largest Lyapunov exponent impossible and thus these values are excluded from that statistic. This happened 6 times with the simple ESN setup, 30 times with mixed activation functions and 5 times, when using a regular Lu readout. Input shift, extended Lu readout and even output bias did not show this behavior in any prediction in this experiment.
Overall using a regular Lu readout or mixed activation functions did not perform quite as well on this standard task as input shift and extended Lu readout.

4.4.3 Overwriting the Mirror-Attractor

Finally, we compare the five different ESNs on a task that we specifically designed to test their symmetry breaking abilities. For this goal we create a dataset by simulating both the Lorenz and the Halvorsen system. The mean of the Lorenz data is shifted to $(1, 1, 1)$ and the mean of the Halvorsen data is shifted to $(-1, -1, -1)$. Both are rescaled so that no datapoint has a distance higher than 1 from the mean in any dimension. This ensures that the two attractors do not overlap while lying completely in the region of each others mirror-attractor. To train the ESN to simultaneously be able to predict both systems we use the method introduced in section 2.4.
This way the ESN has to learn dynamics that are governed by a completely different set of equations instead of the mirror-attractor. In the end it should be able to predict both attractors depending on the starting point of the reservoir states. This way our ESN exhibits multifunctionality. Since the original publication of this work, another publication has come out, where this property is achieved and thoroughly examined [52].
Since this is a more difficult task and to make sure that possible failures are not just due to a lack of nodes we use $N = 500$ in this experiment. However, we made similar qualitative observations for smaller and larger networks.

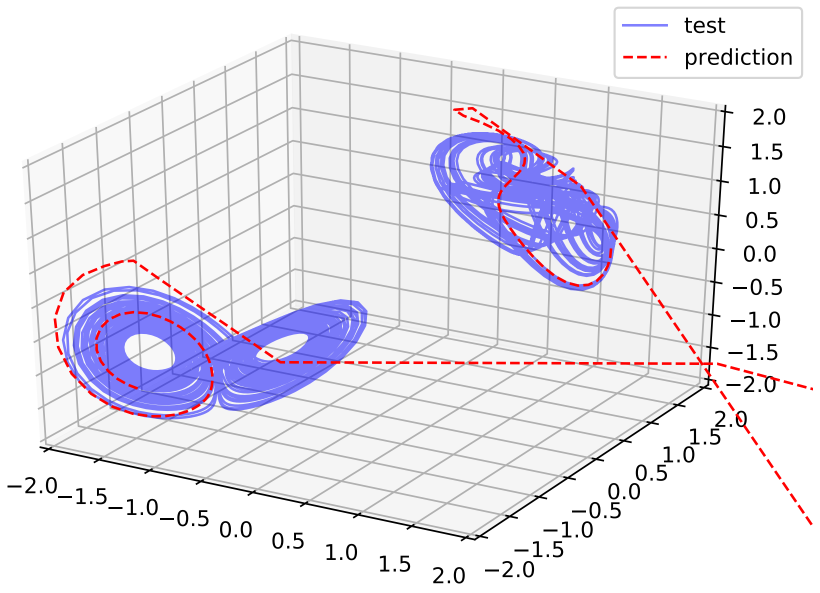


Figure 4.9: Example of predictions after training the same network on Lorenz and Halvorsen data simultaneously with the simple ESN setup.

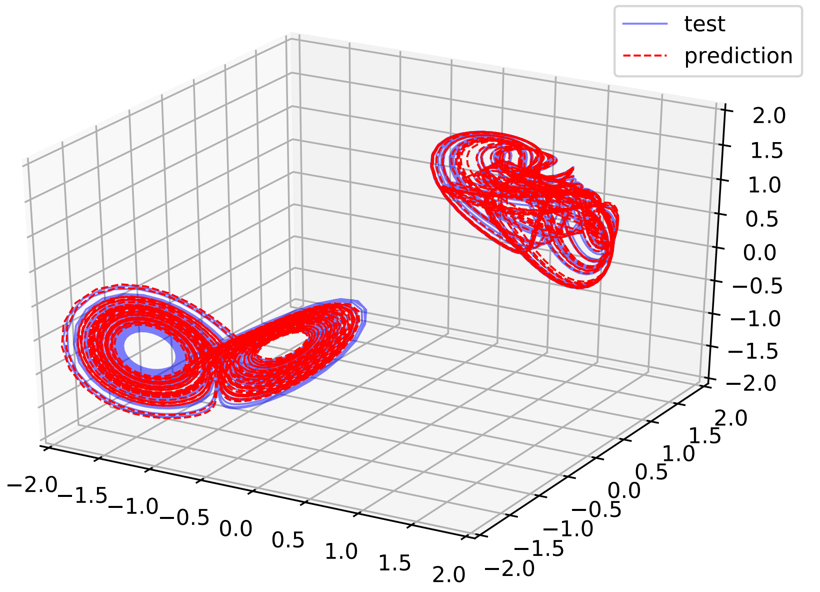


Figure 4.10: Example prediction of Lorenz and Halvorsen attractor with a single ESN with input shift.

To be able to do a quantitative analysis, we first performed a hyperparameter optimization as described in the appendix. We used the product of forecast horizons on both systems as a measure of performance. Afterwards we carried out the same experiment as in Sec. 4.4.2 with this combined dataset. We always made a prediction on both attractors with the same network. The results are compiled in table 4.2.

Unsurprisingly we observe that the simple ESN is not able to master this task (see figure 4.9). Most predictions completely diverge from both attractors. In the handful of cases where one of the predictions actually reproduced the climate of one attractor, the other one was always a complete failure. Qualitatively we see the same results when including an output bias.

Since all predictions with the simple ESN and almost all with the ESN with output bias either diverged or converged to some fixed point, we could not provide meaningful results for the climate. We note however that the short-term prediction of the Lorenz system was actually significantly better for both than in Sec. 4.4.2. We attribute this to the higher number of nodes. Still the inability to reproduce long-term behavior indicates that this kind of problem can only be solved with a properly broken symmetry as we assumed.

Again the other methods to break the symmetry are all successful (see as an example figure 4.10) and predict both attractors with the same training quite well. The results in the forecast horizon for input shift, mixed activation functions and the regular Lu readout are very similar with an average of about 9.8 Lyapunov times (Lorenz) and 10.7 Lyapunov times (Halvorsen). However, only the input shift was able to reproduce the climate with the same accuracy as the test data.

It is notable that the extended Lu readout performed significantly better than the others in terms of the forecast horizon on both systems. The averages were 10.6 Lyapunov times (Lorenz) and 11.6 Lyapunov times (Halvorsen). In contrast, there was also a small number of predictions that completely diverged or got stuck in a fixed point with this design. This occurred 26 times with the extended Lu readout and 8 times with the regular Lu readout. As in Sec. 4.4.2 these predictions are not included in the Lyapunov exponent statistic. The same did not happen when using input shift or mixed activation functions. This might be due to the fact that the Lu readout does not break the symmetry in the reservoir itself. For this more complicated task the additional parameters in the readout might not always be sufficient to encode the difference in the dynamics for a sign change. It could be related to the fact that those dynamics are completely independent of each other.

ESN design	F.H. in Δt (τ_λ)	$\lambda \pm \sigma$	$\nu \pm \sigma$
Simple ESN	183.2(3.2)	-	-
	54.2(0.8)	-	-
Output Bias	226.3(3.9)	-	-
	57.7(0.9)	-	-
Mixed Activations	563.4(9.8)	0.87 ± 0.03	1.99 ± 0.02
	709.4(10.5)	0.74 ± 0.03	1.88 ± 0.04
Input Shift	570.5(9.9)	0.87 ± 0.02	1.992 ± 0.007
	721.4(10.7)	0.74 ± 0.03	1.88 ± 0.04
Lu Readout	561.8(9.7)	0.87 ± 0.02	1.97 ± 0.18
	719.4(10.7)	0.74 ± 0.03	1.87 ± 0.04
Ext. Lu Readout	610.9(10.6)	0.87 ± 0.05	1.93 ± 0.35
	784.1(11.6)	0.74 ± 0.03	1.87 ± 0.03
Test Data	∞	0.87 ± 0.02	1.993 ± 0.007
	∞	0.74 ± 0.03	1.87 ± 0.04

Table 4.2: Performance of the different ESN designs on combined Lorenz and Halvorsen data. Upper value is always Lorenz and lower Halvorsen. Comparison to the original data in last row. Forecast horizon (F.H.) given in units of timesteps and Lyapunov times in brackets.

Chapter 5

Discussion

In this thesis the mechanism behind Reservoir Computing in the form of Echo State Networks was examined with particular attention to the role of the activation function. The task in question was the reconstruction of strange attractors as well as short term predictions of chaotic systems. For this purpose we compared the performance of ESNs with different parameters or designs on different problems by creating large statistics and measuring average short and long term prediction capabilities. This was motivated by and supplemented with qualitative and partly even analytic theoretical considerations.
In chapter 3 we tested the effect of rescaling the activation function on the simple ESN. We found that forecasting usually works best when the dynamics of the input data are mostly confined to the nonlinear regime of the hyperbolic tangent. This ensures a wide variety and thus high separability of reservoir states.
Based on this we empirically found a relationship between the optimal value for the nonlinear scaling parameter and the standard deviation of the input data. This roughly corresponds to a power law. The relationship was found by comparing the data from a number of different dynamical systems. Therefore, it is to some degree independent of the exact nature of the input data. There are however still considerable deviations from the relation in some cases. Overall it constitutes a useful heuristic for the choice of hyperparameters. Even though the fine structure of the loss landscape of a single ESN is extremely complex [53], this can provide a good starting point for an in depth optimization procedure.
In addition to this main region of high prediction capability, we also found a second peak, which is much more pronounced in some systems than in others. This peak is located at rather small values of a, where most of the input lies in the linear regime of the hyperbolic tangent. We found that those of the studied systems with only local nonlinearities were particularly prone to this.
Overall this shows that even in this "averaged" picture there are still quite a lot of complexities in the hyperparameter landscape and room for system specific optimization.
In chapter 4 we showed a mathematical proof for the antisymmetry of the simple ESN with regards to changing the sign of the input. This is a consequence of the antisymmetry of the activation function. It makes it impossible to fully learn the dynamics of any attractor that is not point symmetric around the origin. In practice we observed that the prediction jumps to an inverted version of the real attractor we call mirror-attractor. This is especially disastrous if the two overlap. From this we conclude that this setup is not suitable for general tasks and should not be used in the future.
Furthermore, we note that the sensitivity to this kind of symmetries with regards to the input is a universal property of ESNs and Reservoir Computers in general. This is in no way limited to the specifics of the simple ESN. It must be kept in mind in every reservoir design and can explain the empirical success or failure of some of them.

In our experiments with the output bias we found that formally breaking the symmetry alone is not enough to solve the problems associated with it. It was only able to improve the performance marginally and we still observed the appearance of an only slightly perturbed mirror-attractor. This might be due to the fact that the number of parameters representing the symmetry break in this approach is too low to accurately model the difference.
We were however able to successfully break the symmetry and solve the problem with three other approaches: Introducing an input shift into the activation function, using a mixture of even and odd activation functions and including squared nodes in the readout. All of them were able to eliminate the mirror-attractor and make qualitatively good predictions even for zero-mean Lorenz data, where the overlap with the mirror-attractor is a severe problem for the simple ESN. They were further all able to master the task of predicting a dataset made of Lorenz and Halvorsen data, where it was necessary to learn completely different dynamics in the regime of the mirror-attractor.
The input shift proved in our test as a very useful and reliable tool to break the symmetry. The performance in all tasks was consistently good in short-time prediction. It successfully reproduced the climate statistics of the test data for the zero-mean Lorenz dataset and even for the combined Lorenz and Halvorsen data. This method was the only one to never produce outliers of completely failed predictions. It is also worth stressing that to our knowledge universality is only proven for ESNs with input shift. One disadvantage of this approach is the additional hyperparameter, which has to be optimized.
Even though the Lu readout in its regular form also seems to have broken the symmetry successfully, the results were generally not quite on par with the input shift. However, this gap was bridged by the extended Lu readout, which poses a convenient way to add more parameters to the model without increasing the size of the reservoir. For the zero-mean Lorenz data this lead to a performance essentially identical with that of the input shift. In the case of the combined dataset the short-time prediction surpassed the input shift, while the climate was not reproduced with the same accuracy. This improvement is likely due to the higher number of parameters and thus higher complexity in the readout. While this is of course accompanied by an increase in computational time, it demonstrates the general possibility to enhance the prediction abilities of a given reservoir by extending the readout. Even though the regular Lu readout seems to be enough to break the symmetry, using even higher order nonlinear transformations of nodes and an even bigger output matrix could further increase the performance. An application of this could be found in physical RC, where the dynamics of the reservoir might be inaccessible. One might however consider making the dynamics of the reservoir more complex while keeping the simple linear readout to be more in line with the philosophy of RC.
The ESN with mixed activation functions performed similarly well to the ESN with regular Lu readout. This is interesting, because the former can be understood as using squared nodes not only in the readout, but also in the dynamics of the reservoir. The results imply that this does not lead to a meaningful improvement. At least for our implementation we also found it to have a higher time cost. Thus, we do not recommend its use in the given form. However, the usage of different functions, different ratios, etc. could lead to better performance. Further research in this direction is needed.
In light of these results we recommend both the input shift and the Lu readout as methods to break the symmetry.
Additionally we found a method to let an ESN learn the dynamics of multiple different strange attractors from different data sources. This ability is similar to the property of multifunctionality in neuroscience. The same method could in principle be used in many different ways, for example to train an ESN on more than one trajectory of the same attractor to increase its stability or sample some part of the phase space more densely. This opens up many possible directions for further research.

Overall this work shows, that RC is a powerful tool for prediction of nonlinear systems, which can however still be improved upon in many small and big ways. We found that much care should be placed on the choice and handling of the activation function, since it is extremely influential for the dynamics of the reservoir. Symmetries in particular were shown to be an important property to keep in mind.

Reservoir Computing, like the rest of Machine Learning, is often considered a black box. Practitioners tend not to bother trying to really understand its complex inner workings as long as it gives good results. We hope that the research presented here will help to make the walls of that box a little more transparent.

Appendix A

Equations of additional systems

All of the chaotic systems used were found in the textbook by J. C. Sprott [27]. This is also true of the values of the respective parameters except for the Roessler system.

Roessler System

Equations:

$$\begin{aligned}\dot{x} &= -y - z\\ \dot{y} &= x + ay\\ \dot{z} &= b + z(x - c)\ ,\end{aligned} \tag{A.1}$$

Parameters:
$a = 0.5$, $b = 2$, $c = 4$.

Chua Circuit

Equations:

$$\begin{aligned}\dot{x} &= \alpha(y - x + bx + \frac{1}{2}(a - b)(|x + 1| - |x - 1|))\\ \dot{y} &= x - y + z\\ \dot{z} &= -\beta y\ ,\end{aligned} \tag{A.2}$$

These equations are only nonlinear locally at $x = \pm 1$.

Parameters:
$\alpha = 9$, $\beta = \frac{100}{7}$, $a = \frac{8}{7}$, $b = \frac{5}{7}$.

Complex Butterfly Attractor

Equations:

$$\begin{aligned}\dot{x} &= a(y - x)\\ \dot{y} &= -z sign(x)\\ \dot{z} &= |x - 1|\ ,\end{aligned} \tag{A.3}$$

These equations are only nonlinear locally at $x = 1$ and $x = 0$.

Parameters:
$a = 0.55$.

Chen System

Equations:

$$\begin{aligned} \dot{x} &= a(y - x) \\ \dot{y} &= (c - a)x - yz + cy \\ \dot{z} &= xy - bz \ , \end{aligned} \tag{A.4}$$

Parameters:
$a = 35$, $b = 3$, $c = 28$.

Rucklidge System

Equations:

$$\begin{aligned} \dot{x} &= -\kappa x + \lambda y - yz \\ \dot{y} &= x \\ \dot{z} &= -z + y^2 \ , \end{aligned} \tag{A.5}$$

Parameters:
$\kappa = 2$, $\lambda = 6.7$.

Rabinovich-Fabrikant Equations

Equations:

$$\begin{aligned} \dot{x} &= y(z - 1 + x^2) + \gamma x \\ \dot{y} &= x(3z + 1 - x^2) + \gamma y \\ \dot{z} &= -2z(\alpha + yx) \ , \end{aligned} \tag{A.6}$$

Parameters:
$\alpha = 1.1$, $\gamma = 0.87$.

Thomas' Cyclically Symmetric Attractor

Equations:

$$\begin{aligned} \dot{x} &= -bx + sin(y) \\ \dot{y} &= -by + sin(z) \\ \dot{z} &= -bz + sin(x) \ , \end{aligned} \tag{A.7}$$

As the name implies this sytem is symmetric under cyclical permutations of the variables.

Parameters:
$b = 0.18$.

Double Scroll Attractor

Equations:

$$\begin{aligned}\dot{x} &= y \\ \dot{y} &= z \\ \dot{z} &= -a(z + y + x - sign(x)) \ ,\end{aligned} \tag{A.8}$$

These equations are only nonlinear locally at $x = 0$.

Parameters:
$a = 0.8$.

Simples Piecewise Nonlinear Flow

Equations:

$$\begin{aligned}\dot{x} &= y \\ \dot{y} &= z \\ \dot{z} &= -az - y + |x - 1| \ ,\end{aligned} \tag{A.9}$$

These equations are only nonlinear locally at $x = 1$.

Parameters:
$a = 0.6$.

Appendix B

Forecast Horizon Plots for Additional Systems

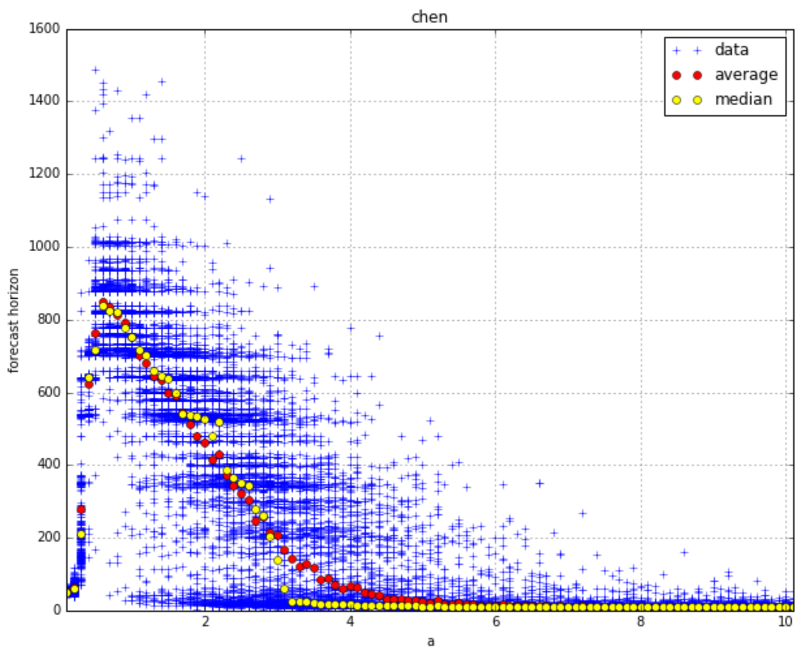


Figure B.1: Forecast Horizon versus a in the Chen system.

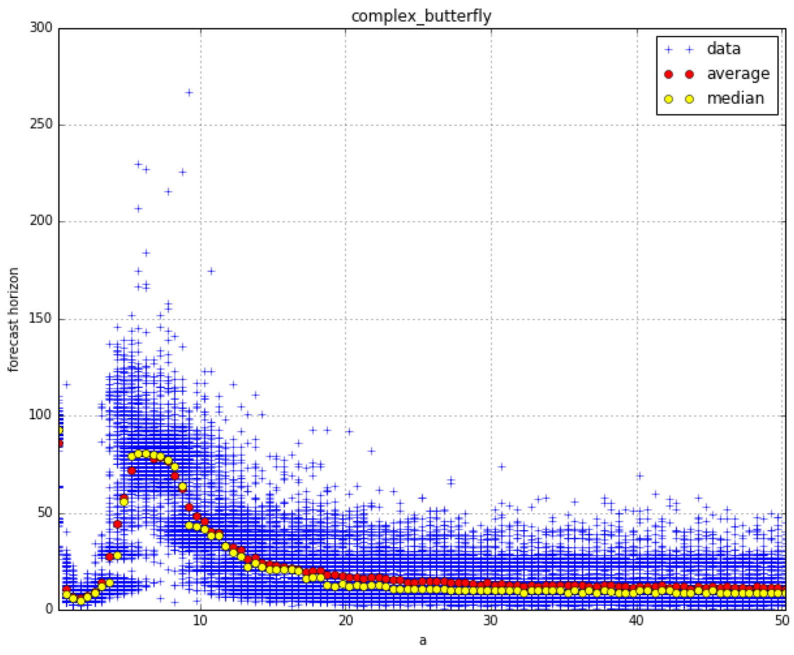


Figure B.2: Forecast Horizon versus a in the Complex Butterfly (large range).

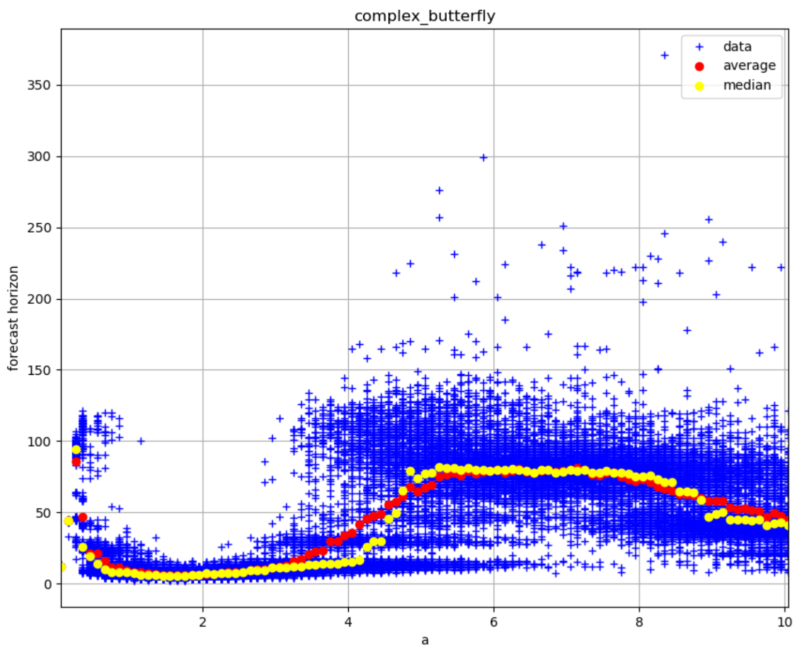


Figure B.3: Forecast Horizon versus a in the Complex Butterfly (short range).

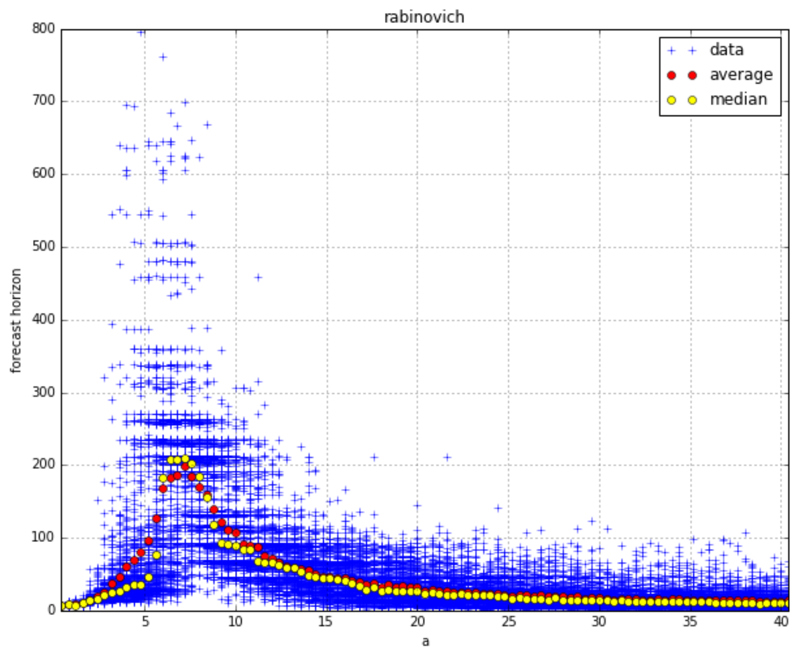


Figure B.4: Forecast Horizon versus a in the Rabinovich-Fabrikant equations (large range).

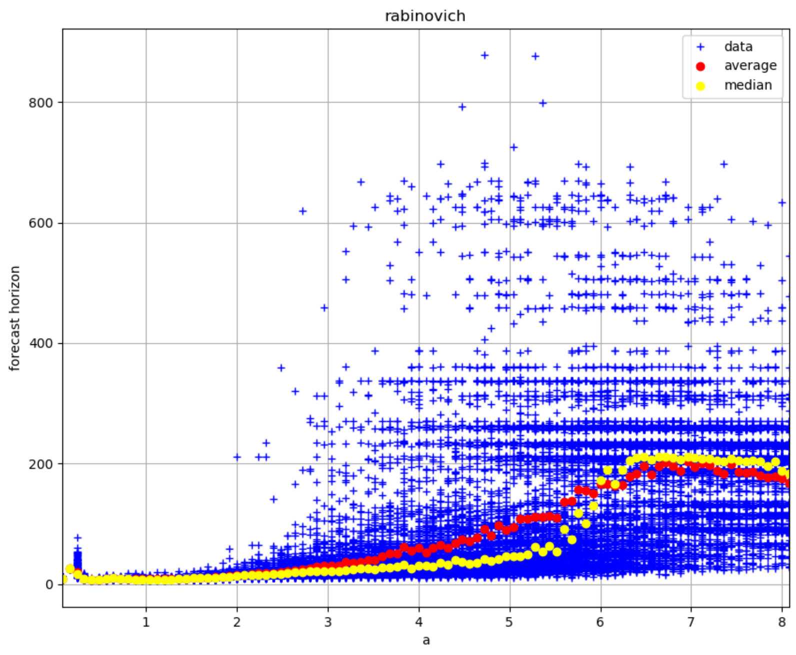


Figure B.5: Forecast Horizon versus a in the Rabinovich-Fabrikant equations (short range).

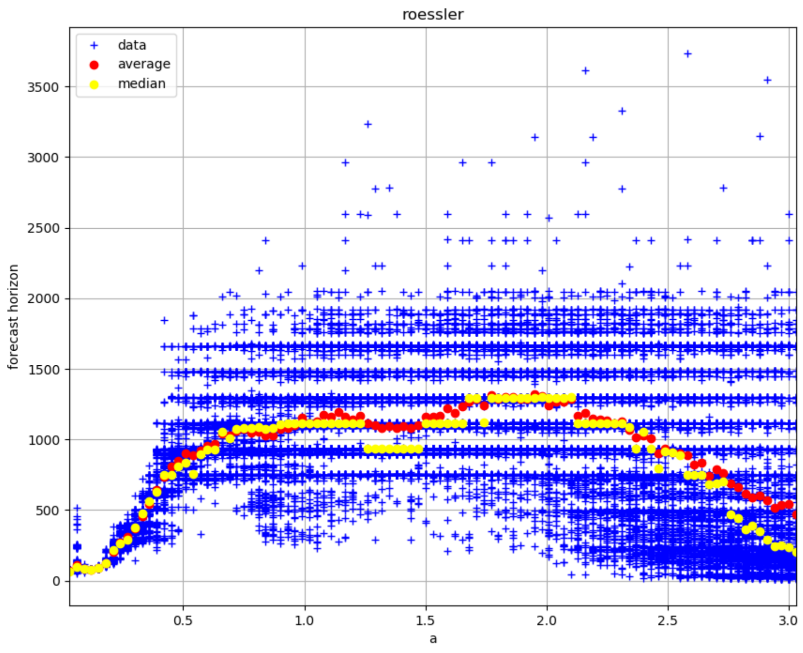


Figure B.6: Forecast Horizon versus a in the Roessler system.

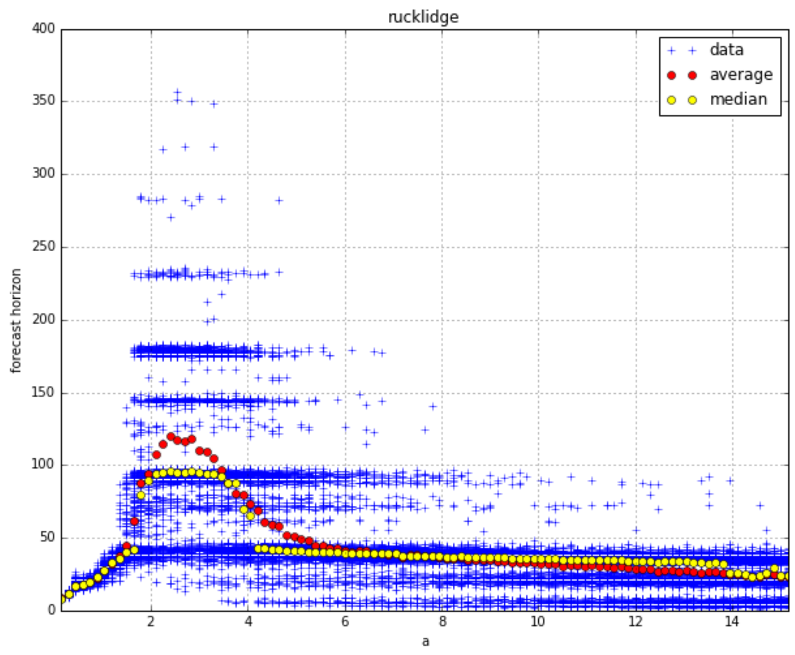


Figure B.7: Forecast Horizon versus a in the Rucklidge system.

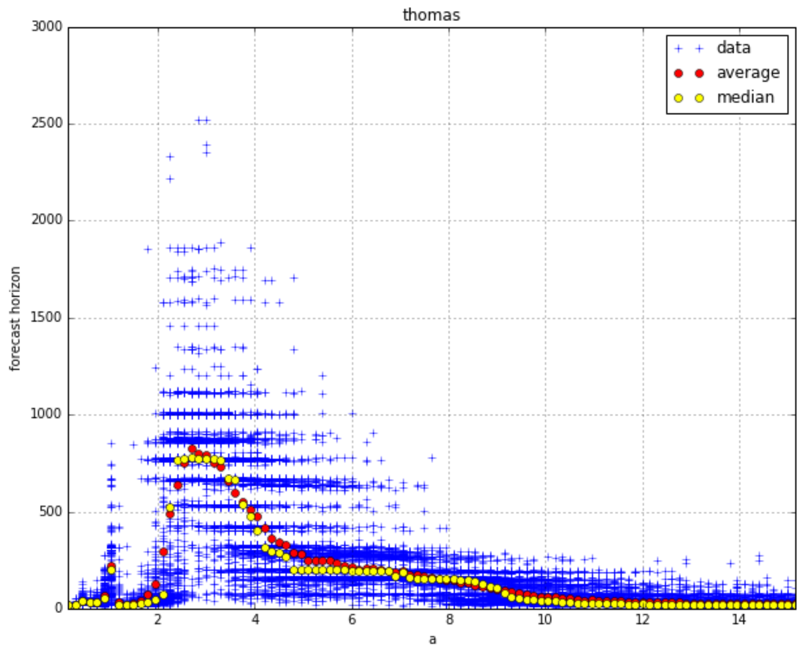


Figure B.8: Forecast Horizon versus a in Thomas' cyclically symmetrical attractor..

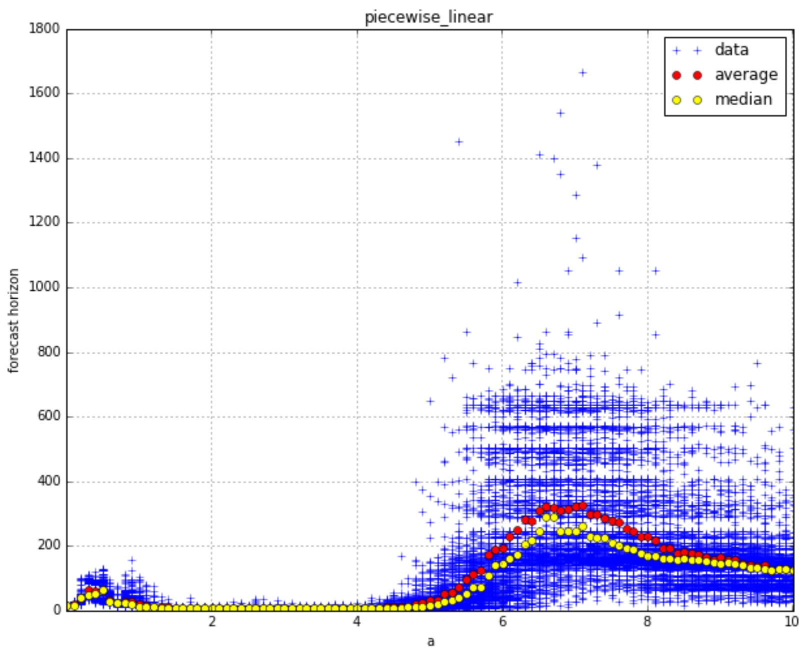


Figure B.9: Forecast Horizon versus a in the simplest piecewise linear chaotic flow.

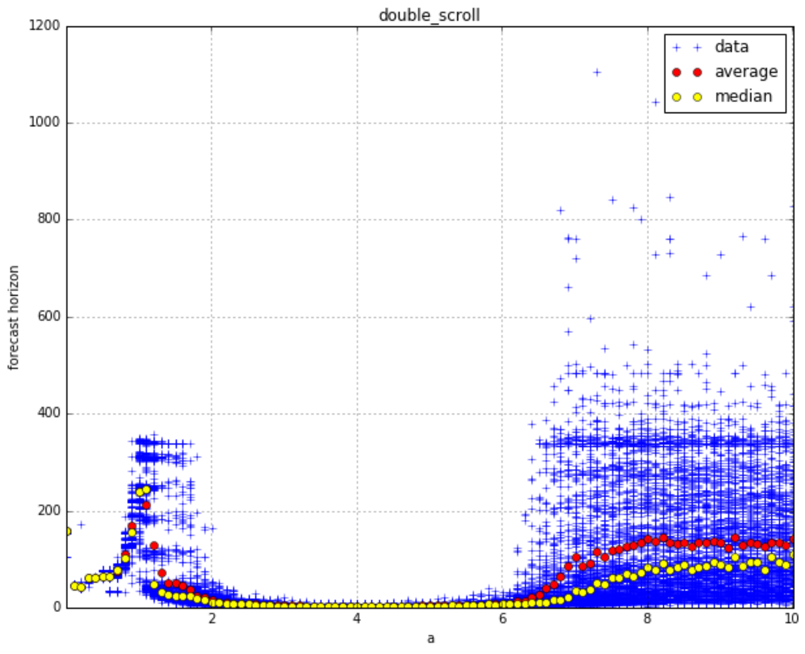


Figure B.10: Forecast Horizon versus a in the double scroll attractor.

Appendix C

Hyperparameter Optimization

As discussed shortly in section 2.2 and examined partly on chapter 3, the proper choice of hyperparameters for an ESN is extremely problem-specific and can not generally be derived from theoretical principles. Thus, some kind of hyperparameter optimization procedure is necessary.
A number of advanced methods have been successfully applied for this goal such as gradient descent [63], bayesian methods [64] or genetic algorithms [37]. However, for the sake of simplicity and because of the useful data it produces we use random search and grid search procedures. This is naturally only possible because of the low training time of RC. Even so, the hyperparameter optimization was the part of this work that needed the highest amount of computation time.
We used a random search for chapter 3 before moving on to the grid search in chapter 4, since it is easier to interpret. Both will be shortly explained in the following sections.

C.1 Lorenz without Preprocessing

This hyperparameter optimization procedure was devised and carried out by Jonas Aumeier for [24].

To find the most suitable parameter values for the spectral radius of the reservoir $\rho(\mathbf{A})$ in chapter 3, the scale for $\mathbf{W}_{in}$ and the regularization constant β, a simple random search procedure with uniform sampling from the parameter space was used. The search ranges for the hyper parameter optimization are [0; 2.5] for $\rho(\mathbf{A})$, [0; 2.0] for $\mathbf{W}_{in}$ and $[\log_e 10^{-11};\ \log_e 0.1]$ for β, which has been scaled onto a logarithmic scale for better coverage of small values. The objective function is the forecast horizon, as defined in Section 2.3.1, averaged over $N = 30$ realizations. The term *realizations* means running reservoir computing with the exact same parameters but different random realizations of the reservoir $\mathbf{A}$ and the input function $\mathbf{W}_{in}$. Each of the realizations is starting from randomly chosen coordinates obtained from simulating a very long trajectory of the Lorenz system. In order to assure independent trajectories, small scale uniform noise is added. The optimal values are shown in table C.1

ρ	0.17
s_{input}	0.17
β	1.9×10^{-11}

Table C.1: Results of the hyperparameter optimization

C.2 Grid Search

The Hyperparameter Optimization for chapter 4 was carried out as a simple grid search with the aim to maximize the forecast horizon. For the simple ESN we searched over a and ϵ, with

$$s_{input} = a(1-\epsilon) \tag{C.1}$$
$$\rho = a\epsilon \tag{C.2}$$

This different but equivalent parameterization seems more natural, knowing the results of chapter 3. The same was done for the ESN with output bias and the ESN with Lu readout. For the ESN with input shift we additionally varied the scale s_γ and for the mixed activations we replaced a with a_1 and a_2, which were optimized for the tanh-nodes and the $\tanh^2$-nodes separately.

C.3 Predicting the mirror-attractor

Since we were less interested in quantitative results in this case we did not perform a real hyperparameter optimization procedure. Instead we reused the parameters from section C.1 for the simple ESN and manually adjusted the parameters for the others to reproduce the Lorenz attractor reasonably well. This left us with the parameters in table C.2.

	Simple	Out. Bias	Lu readout	Mixed	Inp. Shift
a	0.32	0.32	0.32	-	0.32
ϵ	0.5	0.5	0.5	0.5	0.5
β	1.9×10^{-11}	1.9×10^{-11}	1.9×10^{-11}	1.9×10^{-11}	1.9×10^{-11}
s_γ	-	-	-	-	13.
a_1	-	-	-	0.32	-
a_2	-	-	-	0.32	-

Table C.2: Hyperparameter choices for the Prediction of the mirror-attractor.

C.4 Zero-mean Lorenz

In the case of the zero-mean Lorenz data we simulated 100 trajectories of training data and 100 trajectories of test data. At every point in hyperparameter space we generate a new network and a new $\mathbf{W}_{in}$ for each trajectory. We then choose the hyperparameters with the highest average forecast horizon.
Since it did not seem to depend strongly on the other hyperparameters, the problem or the specific design, we simply set $\beta = 1.9 \times 10^{-11}$ as in the first task to save time with the already very costly grid search.
The results are compiled in tables C.3, C.4, C.5, C.6, C.7 and C.8.

Hyperparameter	Min	Max	Step Size	Optimal
a	0.1	3.0	0.1	1.0
ϵ	0.0	1.0	0.05	0.7

Table C.3: Hyperparameter range and results for the simple ESN on zero-mean Lorenz data.

Hyperparameter	Min	Max	Step Size	Optimal
a	0.1	3.0	0.1	1.0
ϵ	0.0	1.0	0.05	0.7

Table C.4: Hyperparameter range and results for the ESN with output bias on zero-mean Lorenz data.

Hyperparameter	Min	Max	Step Size	Optimal
a	0.1	3.0	0.1	0.9
ϵ	0.0	1.0	0.05	0.55

Table C.5: Hyperparameter range and results for the ESN with regular Lu readout on zero-mean Lorenz data.

Hyperparameter	Min	Max	Step Size	Optimal
a	0.1	3.0	0.1	1.3
ϵ	0.0	1.0	0.05	0.4

Table C.6: Hyperparameter range and results for the ESN with extended Lu readout on zero-mean Lorenz data.

Hyperparameter	Min	Max	Step Size	Optimal
a	0.2	3.0	0.2	1.2
ϵ	0.0	1.0	0.1	0.6
s_γ	0.3	3.0	0.3	1.5

Table C.7: Hyperparameter range and results for the ESN with input shift on zero-mean Lorenz data.

Hyperparameter	Min	Max	Step Size	Optimal
a_1	0.2	3.0	0.2	0.6
a_2	0.2	3.0	0.2	0.8
ϵ	0.1	1.0	0.1	0.6

Table C.8: Hyperparameter range and results for the ESN with mixed activation functions on zero-mean Lorenz data.

C.5 Halvorsen and Lorenz

For the combined dataset of Halvorsen and Lorenz attractor we simulate 100 training and test trajectories of each system and use them as described in Sec. 4.4.3. As before we train a completely new reservoir on each trajectory for every point in parameter space. For every realization the product of the two forecast horizons is calculated. The optimal hyperparameters are chosen to maximize this this product averaged over the trajectories.
For this task we did include the regularization parameter β in the search with a logarithmic scale from 10^{-13} to 0.001 in 11 steps. We consistently found $\beta = 10^{-10}$ to be the best choice for all designs.
The results are compiled in tables C.9, C.10, C.11, C.12, C.13 and C.14.

Hyperparameter	Min	Max	Step Size	Optimal
a	1.1	4.0	0.1	2.0
ϵ	0.0	1.0	0.05	0.4

Table C.9: Hyperparameter range and results for the simple ESN on combined Lorenz and Halvorsen data.

Hyperparameter	Min	Max	Step Size	Optimal
a	1.1	4.0	0.1	1.9
ϵ	0.0	1.0	0.05	0.4

Table C.10: Hyperparameter range and results for the ESN with output bias on combined Lorenz and Halvorsen data.

Hyperparameter	Min	Max	Step Size	Optimal
a	1.1	4.0	0.1	1.9
ϵ	0.0	1.0	0.05	0.3

Table C.11: Hyperparameter range and results for the ESN with regular Lu readout on combined Lorenz and Halvorsen data.

Hyperparameter	Min	Max	Step Size	Optimal
a	1.1	4.0	0.1	2.3
ϵ	0.0	1.0	0.05	0.45

Table C.12: Hyperparameter range and results for the ESN with extended Lu readout on combined Lorenz and Halvorsen data.

Hyperparameter	Min	Max	Step Size	Optimal
a	1.2	4.0	0.2	3.0
ϵ	0.0	1.0	0.1	0.1
s_γ	0.3	3.0	0.3	1.5

Table C.13: Hyperparameter range and results for the ESN with input shift on combined Lorenz and Halvorsen data.

Hyperparameter	Min	Max	Step Size	Optimal
a_1	1.2	4.0	0.2	2.8
a_2	1.2	4.0	0.2	2.4
ϵ	0.1	1.0	0.1	0.2

Table C.14: Hyperparameter range and results for the ESN with mixed activation functions on combined Lorenz and Halvorsen data.

Bibliography

[1] H. Kantz and T. Schreiber, *Nonlinear time series analysis*, vol. 7. Cambridge university press, 2004.

[2] G. Zhang, B. Patuwo, and M. Y. Hu, “A simulation study of artificial neural networks for nonlinear time-series forecasting,” *Computers Operations Research*, vol. 28, no. 4, pp. 381–396, 2001.

[3] F. Liang, “Bayesian neural networks for nonlinear time series forecasting,” *Statistics and computing*, vol. 15, no. 1, pp. 13–29, 2005.

[4] O. Kocadağlı and B. Aşıkgil, “Nonlinear time series forecasting with bayesian neural networks,” *Expert Systems with Applications*, vol. 41, no. 15, pp. 6596–6610, 2014.

[5] Y. Tang, J. Kurths, W. Lin, E. Ott, and L. Kocarev, “Introduction to focus issue: When machine learning meets complex systems: Networks, chaos, and nonlinear dynamics,” *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 30, no. 6, p. 063151, 2020.

[6] W. Maass, T. Natschlaeger, and H. Markram, “Real-time computing without stable states: A new framework for neural computation based on perturbations,” *Neural Computation*, vol. 14, no. 11, pp. 2531–2560, 2002.

[7] H. Jaeger, “The “echo state” approach to analysing and training recurrent neural networks-with an erratum note,” *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, vol. 148, no. 34, p. 13, 2001.

[8] P. Tino and G. Dorffner, “Predicting the future of discrete sequences from fractal representations of the past,” *Machine Learning*, vol. 45, no. 2, pp. 187–217, 2001.

[9] P. R. Vlachas, J. Pathak, B. R. Hunt, T. P. Sapsis, M. Girvan, E. Ott, and P. Koumoutsakos, “Backpropagation algorithms and reservoir computing in recurrent neural networks for the forecasting of complex spatiotemporal dynamics,” 2019.

[10] A. Chattopadhyay, P. Hassanzadeh, D. Subramanian, and K. Palem, “Data-driven prediction of a multi-scale lorenz 96 chaotic system using a hierarchy of deep learning methods: Reservoir computing, ann, and rnn-lstm,” 2019.

[11] Z. Lu, B. R. Hunt, and E. Ott, “Attractor reconstruction by machine learning,” *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 28, no. 6, p. 061104, 2018.

[12] J. Pathak, B. Hunt, M. Girvan, Z. Lu, and E. Ott, “Model-Free Prediction of Large Spatiotemporally Chaotic Systems from Data: A Reservoir Computing Approach,” *Physical Review Letters*, vol. 120, p. 024102, Jan 2018.

[13] J. Dambre, D. Verstraeten, B. Schrauwen, and S. Massar, “Information processing capacity of dynamical systems,” *Scientific reports*, vol. 2, no. 1, pp. 1–7, 2012.

[14] Z. Lu and D. S. Bassett, “Invertible generalized synchronization: A putative mechanism for implicit learning in neural systems,” *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 30, no. 6, p. 063133, 2020.

[15] P. Verzelli, C. Alippi, and L. Livi, “Learn to synchronize, synchronize to learn,” *arXiv preprint arXiv:2010.02860*, 2020.

[16] G. Van der Sande, D. Brunner, and M. C. Soriano, “Advances in photonic reservoir computing,” *Nanophotonics*, vol. 6, no. 3, pp. 561–576, 2017.

[17] D. Prychynenko, M. Sitte, K. Litzius, B. Krüger, G. Bourianoff, M. Kläui, J. Sinova, and K. Everschor-Sitte, “Magnetic skyrmion as a nonlinear resistive element: A potential building block for reservoir computing,” *Physical Review Applied*, vol. 9, no. 1, p. 014034, 2018.

[18] G. Tanaka, T. Yamane, J. B. Héroux, R. Nakane, N. Kanazawa, S. Takeda, H. Numata, D. Nakano, and A. Hirose, “Recent advances in physical reservoir computing: A review,” *Neural Networks*, vol. 115, pp. 100–123, 2019.

[19] I. B. Yildiz, H. Jaeger, and S. J. Kiebel, “Re-visiting the echo state property,” *Neural networks*, vol. 35, pp. 1–9, 2012.

[20] C. Gallicchio, “Chasing the echo state property,” *arXiv preprint arXiv:1811.10892*, 2018.

[21] A. Griffith, A. Pomerance, and D. J. Gauthier, “Forecasting chaotic systems with very low connectivity reservoir computers,” *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 29, no. 12, p. 123108, 2019.

[22] T. L. Carroll and L. M. Pecora, “Network structure effects in reservoir computers,” *arXiv preprint arXiv:1903.12487*, 2019.

[23] A. Haluszczynski and C. Räth, “Good and bad predictions: Assessing and improving the replication of chaotic attractors by means of reservoir computing,” *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 29, no. 10, p. 103143, 2019.

[24] A. Haluszczynski, J. Aumeier, J. Herteux, and C. Räth, “Reducing network size and improving prediction stability of reservoir computing,” *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 30, no. 6, p. 063136, 2020.

[25] T. Carroll, “Path length statistics in reservoir computers,” *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 30, no. 8, p. 083130, 2020.

[26] E. N. Lorenz, “Deterministic nonperiodic flow,” *Journal of the atmospheric sciences*, vol. 20, no. 2, pp. 130–141, 1963.

[27] J. C. Sprott and J. C. Sprott, *Chaos and time-series analysis*, vol. 69. Citeseer, 2003.

[28] O. E. Rössler, “An equation for continuous chaos,” *Physics Letters A*, vol. 57, no. 5, pp. 397–398, 1976.

[29] M. Rabinovich and A. Fabrikant, “Stochastic self-modulation of waves in nonequilibrium media,” *J. Exp. Theor. Phys*, vol. 77, pp. 617–629, 1979.

[30] A. M. Rucklidge, “Chaos in models of double convection,” *Journal of Fluid Mechanics*, vol. 237, pp. 209–229, 1992.

[31] T. Matsumoto, L. Chua, and M. Komuro, “The double scroll,” *IEEE Transactions on Circuits and Systems*, vol. 32, no. 8, pp. 797–818, 1985.

[32] G. Chen and T. Ueta, “Yet another chaotic attractor,” *International Journal of Bifurcation and chaos*, vol. 9, no. 07, pp. 1465–1466, 1999.

[33] R. Thomas, “Deterministic chaos seen in terms of feedback circuits: Analysis, synthesis,” labyrinth chaos”,” *International Journal of Bifurcation and Chaos*, vol. 9, no. 10, pp. 1889–1905, 1999.

[34] A. S. Elwakil, S. Ozoguz, and M. P. Kennedy, “Creation of a complex butterfly attractor using a novel lorenz-type system,” *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 49, no. 4, pp. 527–530, 2002.

[35] S. J. Linz and J. Sprott, “Elementary chaotic flow,” *Physics Letters A*, vol. 259, no. 3, pp. 240–245, 1999.

[36] L. Grigoryeva and J.-P. Ortega, “Echo state networks are universal,” *Neural Networks*, vol. 108, pp. 495–508, 2018.

[37] A. A. Ferreira and T. B. Ludermir, “Comparing evolutionary methods for reservoir computing pre-training,” in *The 2011 International Joint Conference on Neural Networks*, pp. 283–290, IEEE, 2011.

[38] Y. Yamaguti and I. Tsuda, “Functional differentiations in evolutionary reservoir computing networks,” *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 31, no. 1, p. 013137, 2021.

[39] A. E. Hoerl and R. W. Kennard, “Ridge regression: Biased estimation for nonorthogonal problems,” *Technometrics*, vol. 12, no. 1, pp. 55–67, 1970.

[40] T. L. Carroll, “Do reservoir computers work best at the edge of chaos?,” *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 30, no. 12, p. 121109, 2020.

[41] F. Wyffels, B. Schrauwen, D. Verstraeten, and D. Stroobandt, “Band-pass reservoir computing,” in *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, pp. 3204–3209, IEEE, 2008.

[42] J. Theiler, “Estimating fractal dimension,” *JOSA A*, vol. 7, no. 6, pp. 1055–1073, 1990.

[43] P. Grassberger and I. Procaccia, “Measuring the strangeness of strange attractors,” *Physica D: Nonlinear Phenomena*, vol. 9, no. 1-2, pp. 189–208, 1983.

[44] M. T. Rosenstein, J. J. Collins, and C. J. De Luca, “A practical method for calculating largest lyapunov exponents from small data sets,” *Physica D: Nonlinear Phenomena*, vol. 65, no. 1-2, pp. 117–134, 1993.

[45] L. Barreira, *Lyapunov exponents*. Springer, 2017.

[46] M. Sandri, “Numerical calculation of lyapunov exponents,” *Mathematica Journal*, vol. 6, no. 3, pp. 78–84, 1996.

[47] K. Geist, U. Parlitz, and W. Lauterborn, “Comparison of different methods for computing lyapunov exponents,” *Progress of theoretical physics*, vol. 83, no. 5, pp. 875–893, 1990.

[48] K. T. Alligood, T. D. Sauer, and J. A. Yorke, *Chaos*. Springer, 1996.

[49] G. Benettin, L. Galgani, A. Giorgilli, and J.-M. Strelcyn, “Lyapunov characteristic exponents for smooth dynamical systems and for hamiltonian systems; a method for computing all of them. part 1: Theory,” *Meccanica*, vol. 15, no. 1, pp. 9–20, 1980.

[50] J.-P. Eckmann and D. Ruelle, “Ergodic theory of chaos and strange attractors,” in *The theory of chaotic attractors*, pp. 273–312, Springer, 1985.

[51] J. Pathak, Z. Lu, B. R. Hunt, M. Girvan, and E. Ott, “Using machine learning to replicate chaotic attractors and calculate lyapunov exponents from data,” *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 27, no. 12, p. 121102, 2017.

[52] A. Flynn, V. A. Tsachouridis, and A. Amann, “Multifunctionality in a reservoir computer,” *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 31, no. 1, p. 013125, 2021.

[53] Y. Mabrouk and C. Räth, “Calibrated reservoir computers,” *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 30, no. 11, p. 113134, 2020.

[54] M. Inubushi and K. Yoshimura, “Reservoir computing beyond memory-nonlinearity trade-off,” *Scientific reports*, vol. 7, no. 1, pp. 1–10, 2017.

[55] A. Goudarzi, A. Shabani, and D. Stefanovic, “Exploring transfer function nonlinearity in echo state networks,” in *2015 IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA)*, pp. 1–8, IEEE, 2015.

[56] D. Verstraeten, B. Schrauwen, M. d’Haene, and D. Stroobandt, “An experimental unification of reservoir computing methods,” *Neural networks*, vol. 20, no. 3, pp. 391–403, 2007.

[57] P. Verzelli, C. Alippi, and L. Livi, “Echo state networks with self-normalizing activations on the hyper-sphere,” *Scientific reports*, vol. 9, no. 1, pp. 1–14, 2019.

[58] J. Herteux and C. Räth, “Breaking symmetries of the reservoir equations in echo state networks,” *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 30, no. 12, p. 123142, 2020.

[59] Z. Lu, J. Pathak, B. Hunt, M. Girvan, R. Brockett, and E. Ott, “Reservoir observers: Model-free inference of unmeasured variables in chaotic systems,” *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 27, no. 4, p. 041102, 2017.

[60] M. Lukoševičius, *A Practical Guide to Applying Echo State Networks*, pp. 659–686. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012.

[61] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, “Extreme learning machine: a new learning scheme of feedforward neural networks,” in *2004 IEEE international joint conference on neural networks (IEEE Cat. No. 04CH37541)*, vol. 2, pp. 985–990, Ieee, 2004.

[62] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, “Extreme learning machine: theory and applications,” *Neurocomputing*, vol. 70, no. 1-3, pp. 489–501, 2006.

[63] L. A. Thiede and U. Parlitz, “Gradient based hyperparameter optimization in echo state networks,” *Neural Networks*, vol. 115, pp. 23–29, 2019.

[64] J. Yperman and T. Becker, “Bayesian optimization of hyper-parameters in reservoir computing,” *arXiv preprint arXiv:1611.05193*, 2016.

Acknowledgements

First of all, I want to thank my supervisor Christoph for giving me this opportunity, being supportive of all my ideas and motivating me every step of the way.
I also want to thank Jonas, Sebastian and Youssef for many great discussions about reservoir computing and everything else. This did not only help me to understand the subject better but also made the experience of creating this thesis much more fun.
Similarly, I want to thank everybody from the Complex Plasma group for a really good work environment. Especially Peter and David, who were always quick to help, whenever I was set out to prove that being able to write the code for a neural network doesn't mean you know anything about IT. Sadly, my time to get to know the team better was shortened by the COVID-19 pandemic.

And last but not least I want to thank my family for always being there for me. And especially my father for sparking my scientific curiosity.

Erklärung:

Hiermit erkläre ich, die vorliegende Arbeit selbständig verfasst zu haben und keine anderen als die in der Arbeit angegebenen Quellen und Hilfsmittel benutzt zu haben.

München, den 15. März 2021

Unterschrift