Master's Thesis

The Role of Network Variations for Reconstructing Chaotic Attractors with Reservoir Computing

Jonas Aumeier

Ludwig-Maximilians-Universität München
Department: Physics
Supervisor: Dr. Christoph Räth

April 14, 2020

Masterarbeit

Die Rolle von Netzwerkvariationen für die Vorhersage von chaotischen Attraktoren bei Reservoir Computing

Jonas Aumeier

Ludwig-Maximilians-Universität München
Fakultät: Physik
Betreuer: Dr. Christoph Räth

April 14, 2020

Contents

Chapter 1

Introduction to Reservoir Computing

We are living in a data driven era. Never have we been able to collect and store more data. Since the amount of information vastly exceeds human perception and cognition capabilities, we need to make use of tools to exploit the data surrounding us. One of these tools for turning data into knowledge is *machine learning* [40]. It can identify patterns contained within data or make predictions about the future [30]. Instead of just using it as a tool to compute outcomes using given rules, in machine learning we pass data and answers to let it find the rules for us. This is why machine learning is considered as a sub field of *artificial intelligence (AI)* using self-learning algorithms. It derives rules from past data, that are applied to current data to make predictions or classifications that form the basis for data-driven decisions [40]. Nowadays, examples include email spam filters, speech recognition, breast cancer identification [28] and design aid for antibiotics [29]. All these require machine learning in one or the other way.

The world we live in is nonlinear and therefore potentially chaotic. Examples range from the three-body problem in mechanics, to optics or fluid dynamics as well as the climate [45]. In this work we introduce a machine learning method called *reservoir computing*. We apply it to learn and replicate chaotic systems[1]. Once trained reservoir computing itself is a nonlinear dynamical system[2]. It is capable of distinguishing between various chaotic signals by comparing trained weights [6] and outperforms backpropagation through time in predicting long-term statistics of spatiotemporal chaos[3] [49].

In principle, chaotic systems are deterministic. Nevertheless, they exhibit a strong dependence on initial conditions, that makes long-term predictions impossible [35]. The first short-term prediction methods were introduced by Farmer and Sidorowich [12], using nonlinear prediction by delay embedding [43]. Reservoir computing allows for model free prediction of chaos in general, without embedding and delay time adaption[4] [6, 34]. We will see, that only short time series are necessary to grasp a systems dynamic and use it for prediction and modeling. From there it could be used for noise reduction, evaluating Lyapunov exponents and topological aspects[5].

[1] Chaotic attractors are the phase space representation of time series generated from chaotic systems. A more detailed characterization can be found in section 3.1 Chaotic Time Series.
[2] In Equation 2.4 the recursive equation is given.
[3] It is important to note, that this only holds true if all dynamical variables are available.
[4] Therefore, it is capable of handling incomplete data and system knowledge and can be used for inference of hidden variables [25].
[5] These are core interests when dealing with chaotic physical data [1]

We are using the modified Lorenz equations[6] to generate chaotic time series. On the one hand, we are assessing the short-term prediction accuracy through so called demerge time. On the other hand, we calculate Lyapunov exponents and correlation dimensions in order to compare the statistical long-term behavior of simulated and predicted time series. The objective of this thesis is to investigate the effect of variations on the network structure on prediction statistics, as continuation of Haluszczynski and Räth [17], Lu et al. [24] and Carroll and Pecora [7]. We propose node removal method improving climate replication accuracy, when applied to 10% of nodes. For even smaller networks our method clearly outperforms randomly initialized networks of the same size.

The following thesis is structured as follows. First, we classify reservoir computing with respect to network structure and learning method in section 2.1. In section 2.2 and section 2.3 the reservoir computing framework and implementation details are explained. An introduction to chaotic time series and their generation from the modified Lorenz equation is given in chapter 3. Followed by an overview of the measures used to characterize target and prediction in chapter 4.

After a parameter exploration focused on network topology in section 5.1 and section 5.2, we propose a method for enhancing reservoir computing efficiency with respect to network size in section 5.3. In a last step section 5.4, we suggest static and dynamic closed loop properties that correlate with prediction accuracy.

A final conclusion and outlook is given in chapter 6.

[6]See section 3.3 Modified Lorenz System for the exact differential equations and parameters.

Chapter 2

Reservoir Computing Setup

Reservoir Computing can be optimized either to make short-term predictions [citation] or to replicate the long-term behavior, also called *climate*. From a complex systems point of view the latter means to reproduce the attractor of dynamical system [16, 34, 17].

2.1 From Artificial Intelligence to Machine Learning to Reservoir Computing

There are two perspectives when it comes to classification of machine learning. First, one can classify machine learning in three different types according to the kind of data at hand [40, 19]:

1. Supervised Learning
   uses labeled data and direct feedback to predict or classify future data

2. Unsupervised Learning
   finds hidden structure within unlabeled data without using feedback

3. Reinforcement Learning
   learns to decide between possible actions within an environment through a reward system

Second, depending on the learning type, there is a bunch of models available. In this thesis we are assessing the prediction of chaotic time series using *reservoir computing*. Since we have the *true* data, it is considered as supervised learning.
Supervised learning can be subdivided into classification eg. spam mail or not and regression, where a relation between predictor variables (features) and continuous response variables (targets) is learned [40]. Because we aim to predict the future behavior of a time series the response variable simply is the time series one time step ahead. For this task we employ a *neural network* model, which again can be distinguished according to its internal structure. There are [20]:

1. Feedforward Networks Figure 2.1

2. Recurrent Networks Figure 2.2

While the first is (just) a function, mapping input to output in a static manner, the latter is a *dynamical system* due to its recurrent connections. Recurrent neural networks are well suited to work with sequential data. Biological neural nets are recurrent and applications are control of engines and generators, speech recognition as well as man machine interface, robotics, data analysis, filtering

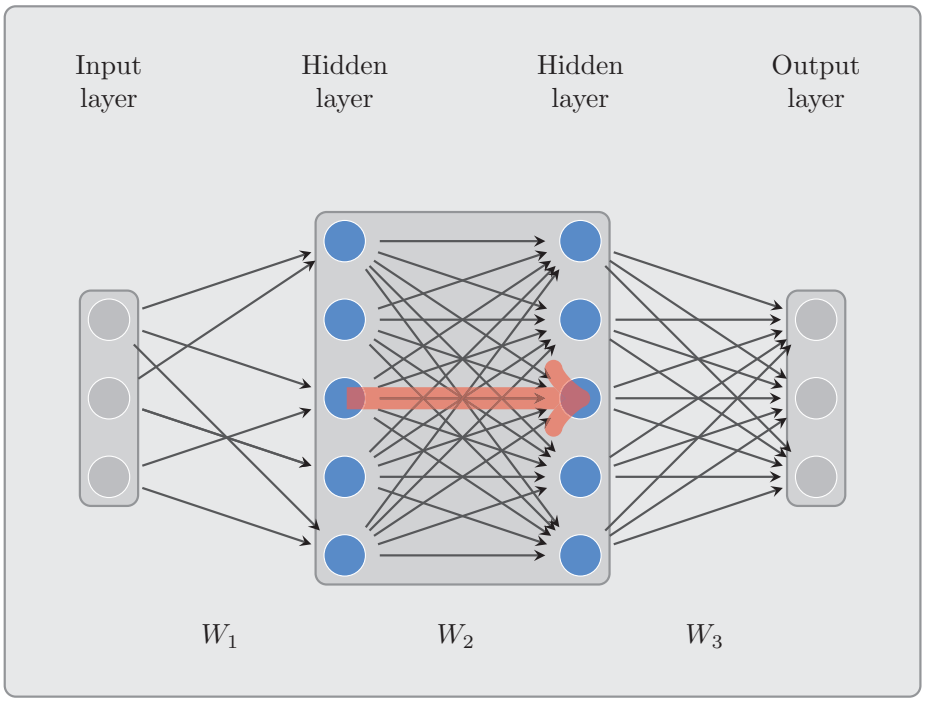


Figure 2.1: Feedforward neural network
W_1 connects the input to the actual neural network, labeled W_2. The red arrow highlights the feedforward structure of the network. W_3 transforms the internal states to the output.

and prediction, pattern classification [14]. One drawback is the difficulty of training them using backpropagation through time [20].

In the following we will discuss different aspects (setup, parameters, measures, limitations) of time series prediction using reservoir computing. How this computationally feasible approach of machine learning introduced by Jaeger [18] and Maass [27] works in detail is described in the next chapter.

Reservoir computing is classified as RNN due to the cyclic topology of the reservoir. Therefore it is a *dynamical system* rather than a function (like feed forward neural networks). The recurrency introduces *nonlinear memory*, which enables the system to process *time series* data. [26].

RNNs are Universal dynamical system approximators [13], therefore they are promising tools. Learning via gradient descent faces various challenges, such as the vanishing or exploding gradient problem discovered by Hochreiter and Bengio et al. [14] or parameter driven bifurcations [9]. Reservoir computing circumvents these difficulties by avoiding to update internal weights [26, 24, 52]. The learning consists of only one linear fit from high dimensional echo state space to the desired output.

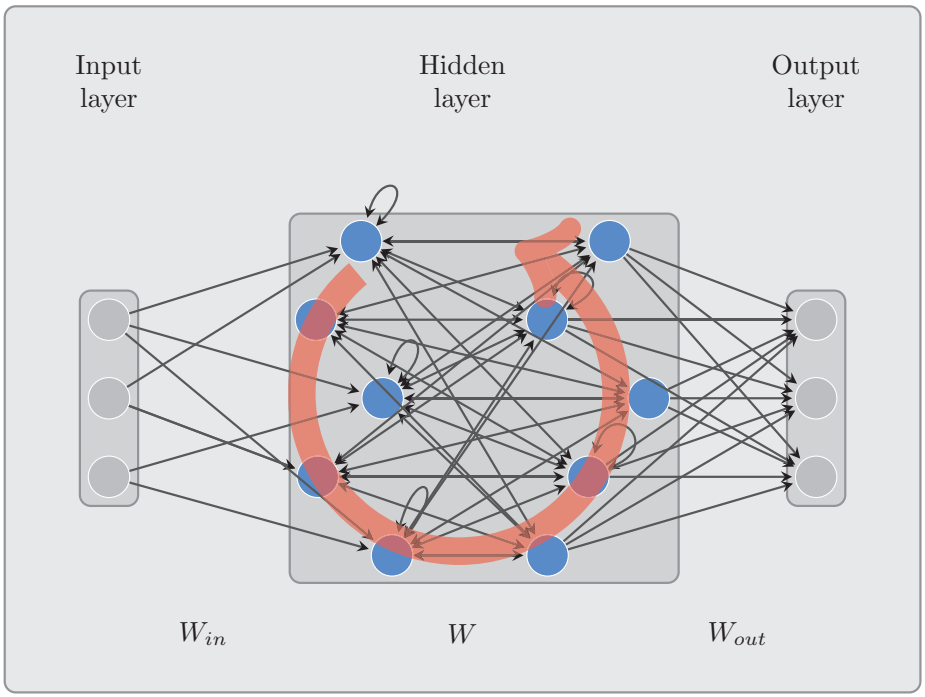


Figure 2.2: Recurrent neural network
W_{in} connects the input to the actual neural network, labeled W. The red arrow emphasizes the recurrent structure of the network. W_{out} transforms the internal states to the output.

2.2 General Formalism

At first it is important to mention, that Reservoir Computing differs from other RNN techniques. In the former, there is a conceptual difference between the (nonlinear) expansion of input data x to echo state space r and the readout of echo state space to the target. While the latter The former is subject to topological and statistical variations, while the latter usually consists of regularized linear regression. This separation resembles the kernel expansion idea [26]. The objective of this thesis is to optimize the expansion without increasing the size of the echo state space. The expansion involves a high degree of randomness through its major components W_{in} and the network W being random element wise and partly random topologically in most of the literature [18, 27, 25, 52, 26]. This gives rise to hope for finding more sophisticated and effective network structures.

Two types of cyclicity can be seen in Figure 2.3, where the inner circle (hidden to hidden connection) refers to the term *recurrent*. During training *teacher forcing* is used, meaning that the target is used as input. During prediction the system is used in *open loop* mode, meaning that the output is fed back as input [14].

During training one drives a dynamical system, the *reservoir*, with a certain (n-dimensional) time series as input and records the internal states of this *reservoir*. Instead of altering the weights of the reservoir, one trains a readout layer W_{out} to fit the recorded internal states to the desired output. In our case linear regression is used. During the prediction phase the reservoir is recurrently driven by the predicted output [26].

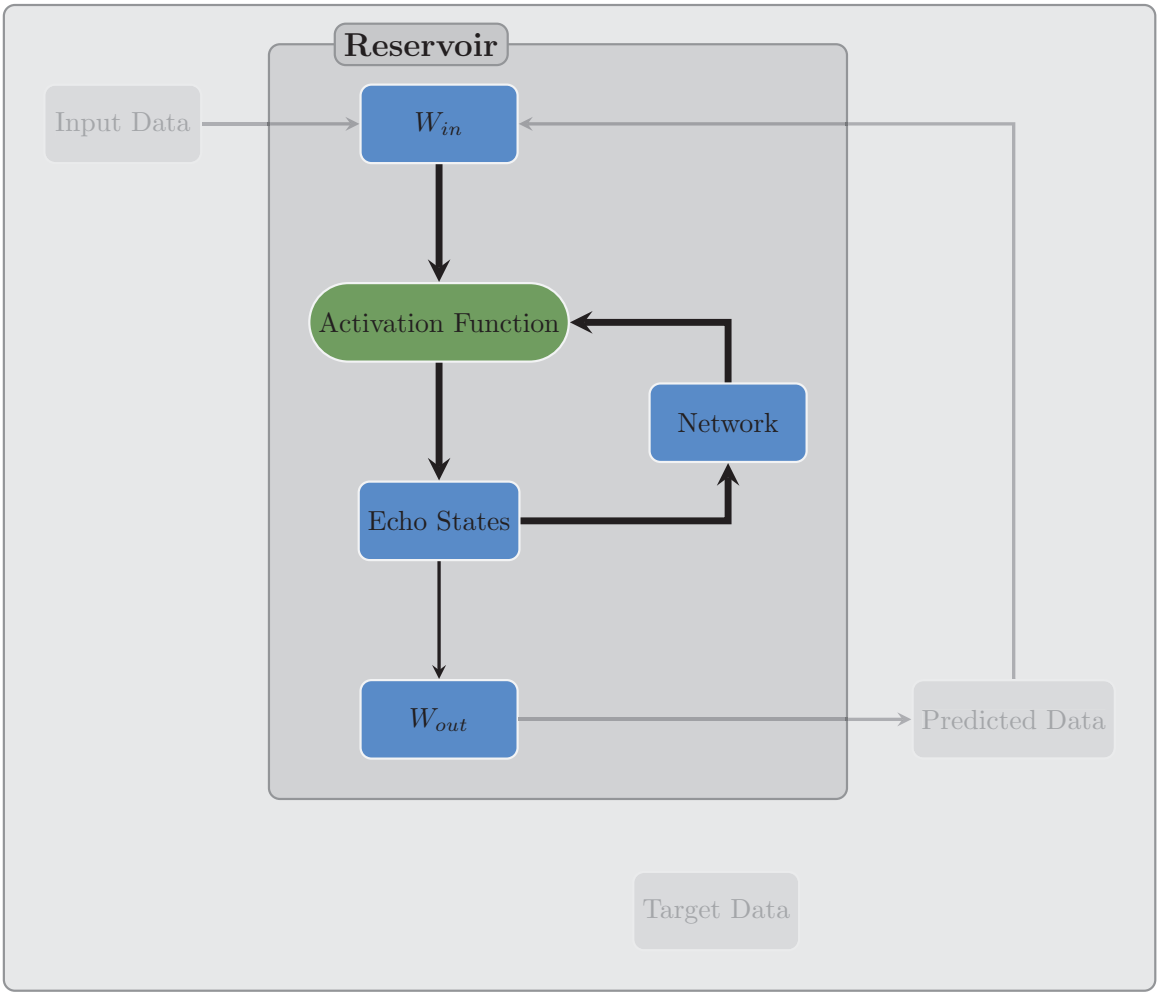


Figure 2.3: Reservoir Computing Setup - Overview
The graphic contains all relevant elements necessary for reservoir computing. A more detailed description of each of them is given in section 2.3.

2.3 Components and Procedure

W_{in} The input matrix maps the input data to the internal states of the reservoir. We use a structure with only one non-zero element per node, assigned with random uniform numbers. The range is symmetric to zero and the limit is considered as a parameter, W_{in} scale.

Network The reservoir computing network contains the information about internal connection between nodes. In contrast to other machine learning methods, the weights of this network are assigned initially, but not altered during training. In the predominant majority of reservoir computing literature the used **topology** is *random* Erdös-Rényi networks [33] [34, 25, 52, 18]. They are characterized by network size N and density ρ of nonzero entries. Other conceivable network types are *small world* [50] and *scale free* networks [5, 4]. The former exhibits a comparably small average path length between any pair of nodes, while preserving a high degree of clustering[1]. The latter is characterized by its power law degree distribution, resulting in only a few nodes with high degree and many sparse connected ones. A comprehensive characterization and an overview of many real world examples can be found in Amaral et al. [3].
The networks used in this thesis are created by the `networkx` python package. For small world networks the function
`networkx.watts_strogatz_graph(N, k=degree, p=0.1)` is used, yielding a network of size N, fixed average degree and - unless otherwise stated - with 10% of the nodes being randomly reconnected. One begins with a *regular* network ordering the nodes in a ring and connecting it to $d/2$ next neighbors. In a second step a fraction p_{sw} of connections randomly rewired. For scale free networks we use
`networkx.barabasi_albert_graph(N, degree/2)` resulting in a network with N nodes and a fixed average degree. Here one sets out with some small seed network, followed by successively adding nodes by connecting them with $d/2$ of links to the existing nodes via preferential attachment[2]. It is important to note, that the small world property becomes significant compared to a random network only when the degree is small compared to the networks size. In addition, it should be emphasized that the manifestation of the scale free property is limited to the available scales. In our examples it only goes up to a few hundred nodes per network, so the term scale free should be used with caution.

Random Number Assignment Up to this point we focused on the network topology. It was shown by Carroll and Pecora recently that there is a positive effect of allowing for other network entries apart from $[0, 1]$ to enrich the dynamics by adding -1 to the available range [7]. This motivates the common practice of assigning random values to the non-zero entries of networks generated by the aforementioned algorithms [52, 25, 24, 18, 17]. As for W_{in} random uniform numbers are allocated to the non-zero entries.

Spectral Radius In a next step the largest absolute eigenvalue, called spectral radius r[3], is calculated and the hole network is rescaled to match the desired spectral radius [26, 18].

Training During the first phase of training the input data is fed into the reservoir via W_{in} and the internal states are updated according to Equation 2.1. Figure 2.4 illustrates the necessary reservoir computing parts during this process. We use *tanh* as nonlinear activation function throughout this text.

[1] The following relation between average path length L and network size N holds:
$\log(N) \propto L$

[2] Preferential attachment means that the probability for an existing node to be connected to the new node is proportional to the old nodes degree

[3] Often λ is used for the spectral radius. To avoid confusion with the later introduced Lyapunov exponent we stick to this notation throughout the text.

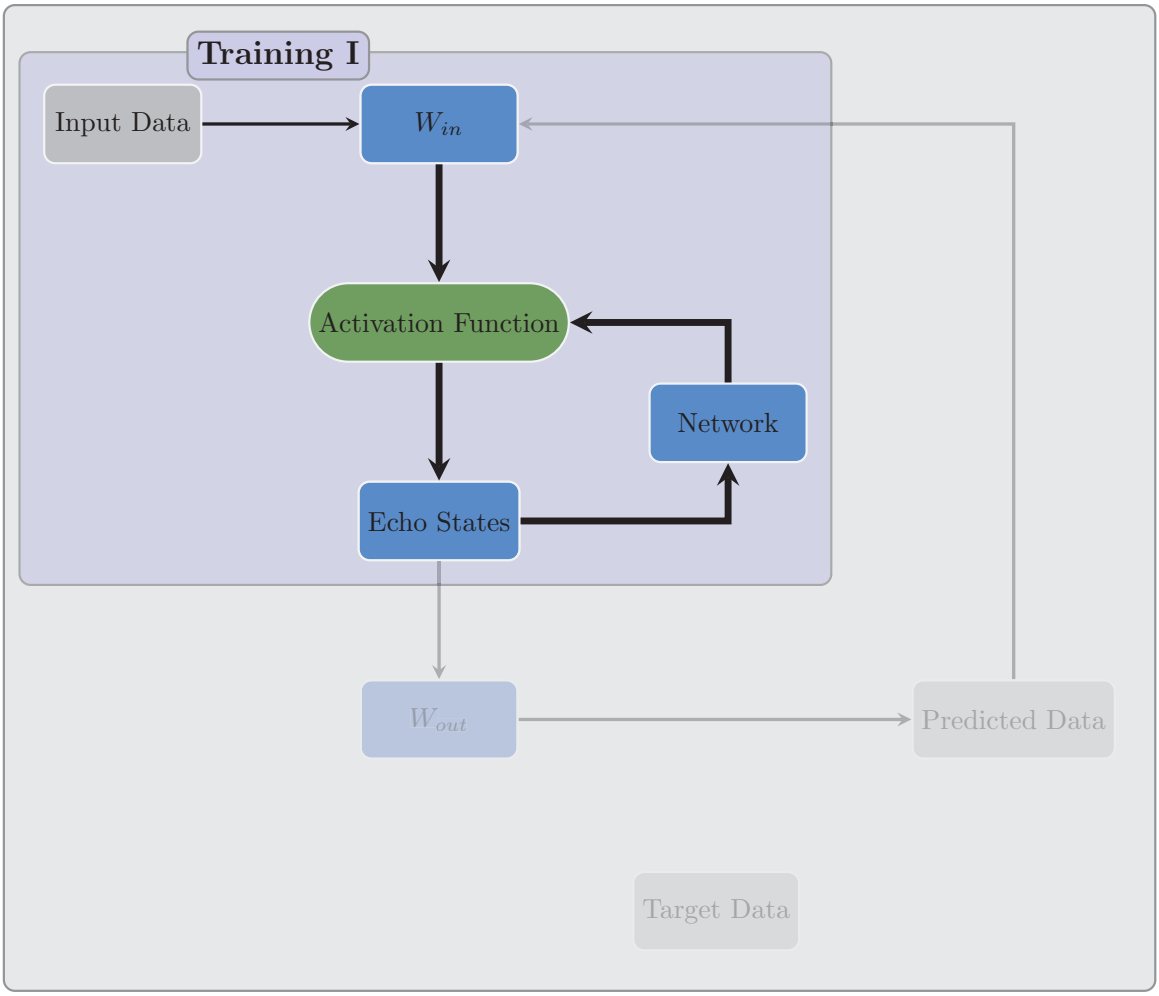


Figure 2.4: Reservoir Computing Setup - Training I
The active components during the echo state recording are highlighted by the purple box. The training data [Input Data] is fed into the echo states via W_{in} and the activation function. The recurrence in the network is responsible for the memory of past states when updating the echo states. The echo states are updated according to Equation 2.1 and recorded for the training phase.

$$\vec{r}(t+1) = tanh\left(W \cdot \vec{r}(t) + W_{in}\vec{x}(t)\right) \tag{2.1}$$

The echo states $\vec{r}(t)$ are recorded.

Regularized linear regression - Training II Next, Tikhonov regularized linear regression is used to fit the recorded echo state to the target time series $\vec{y}_{target}$ [24, 52]. The result of this fit is contained in the matrix W_{out} . The goal is to minimize

$$\sum_t ||W_{out} \cdot \vec{r}(t) - \vec{y}_{target}(t)||^2 - \beta||W_{out}||^2 \tag{2.2}$$

The regularization parameter β prevents from overfitting. The closed form of the regression reads in matrix notation [52]

$$W_{out} = \left(r^T r + \beta \mathbb{1}\right)^{-1} r^T y_{target} \tag{2.3}$$

The active components are shown in Figure 2.5.

The reason why reservoir computing is computationally much cheaper than other methods involving iterative gradient based methods it the described one-shot learning.

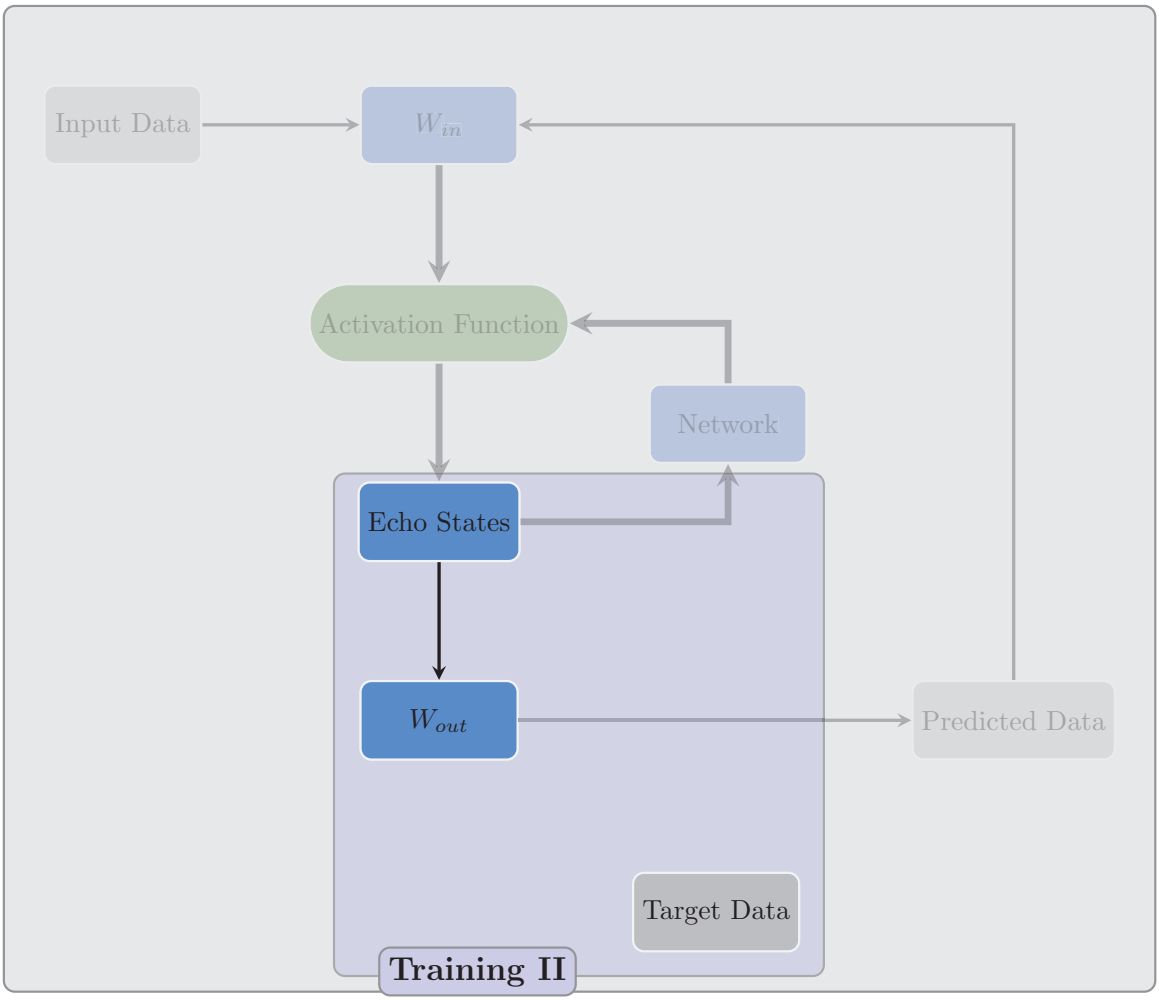


Figure 2.5: Reservoir Computing Setup - Training II
In this step the recorded echo states are used to fit them to the target data, using Equation 2.3.

Prediction After the training phase the system is rewired into a *closed loop* setup, thus the prediction output is fed back into the system as input. See Figure 2.6 for illustration. The relation $\vec{y} = W_{out} \cdot \vec{r}$ turns Equation 2.1 into a closed recurrent form,

$$\vec{r}(t+1) = tanh\left([W \cdot + W_{in} W_{out}]\, \vec{r}(t)\right) \tag{2.4}$$

The final step is to compare the predicted results with the true (or target) time series. In the next chapter chapter 3 explains how true data is generated. chapter 4 introduces measures on time series to compare true and predicted trajectories.

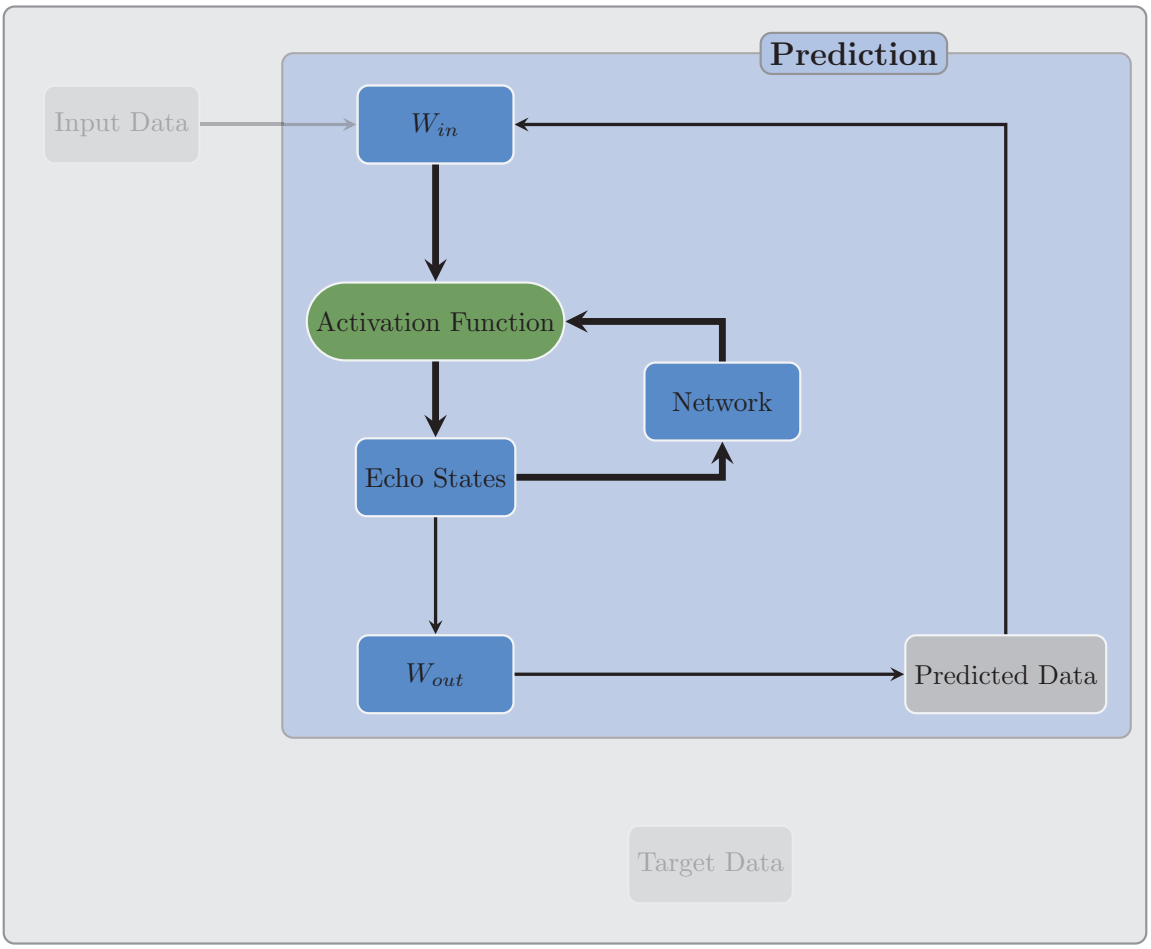


Figure 2.6: Reservoir Computing Setup - Prediction
The blue box emphasizes the relevant components during prediction. W_{out} is used to map echo states to predicted data. The two levels of recurrence become obvious, when looking at the arrows. First, there is a recurrence in the update of echo states [thick arrows]. Second, the output is fed back in as input one time step later [thin arrows], thus the prediction can be ran for an arbitrary number of steps.

Chapter 3

Synthetic Data

In the following chapter we describe which dynamical system is used for testing the performance of reservoir computing and how discrete datasets were obtained. The feature that we are looking for is chaotic behavior or in other words a *strange attractor* [1] [2]. The first deterministic chaotic system found was the three-body problem discovered by Henri Poincaré [36], while the chaotic attractor was characterized by Edward Lorenz [23].

3.1 Chaotic Time Series

A strange attractor A is the phase space representation of a chaotic time series and it is characterized [3] as follows [45, 44]:

- It is a *limit set* as times goes to infinity. It takes an infinite time to reach the attractor from an arbitrary initial condition.
- A is an *invariant set*, meaning that starting in A, the trajectory does not leave A as time goes to infinity.
- It is *bounded*, thus not stretched to infinity but can be enclosed within a region of finite hypervolume.
- A attracts an open set of initial conditions, meaning that A attracts all trajectories that start sufficiently close to it. In other words, there is an open set U containing A such that if $\vec{x}(0) \in U$, then the distance from $\vec{x}(t)$ to A tends to zero as $t \to \infty$. The largest such U is called the *basin of attraction* of A.
- A is a *fractal*, that means it is a *self-similar* structure in coordinate space.
- It is *transitive*. This means that if you start almost anywhere on the attractor, the dynamics will take you arbitrarily close to every other point on the attractor.
- It is usually, in our cases always, *chaotic*.
- A is *indecomposable*, in the sense that no proper subset of A satisfies the above conditions.

[1] In the following we will use attractor as a shorthand for strange attractor or chaotic attractor.
[2] The terminology strange refers to the fractal dimension of the attractor. See section 4.3 Correlation Dimension D_{cor} on how to quantify this feature.
[3] The following list is neither complete, nor necessarily a self-consistent definition.

- It is *aesthetically appealing*, as can been in Figure 3.1 Lorenz Attractor.

There are many examples of seemingly simple nonlinear equations that exhibit chaotic behavior in certain parameter ranges [45]. For an extensive list of chaotic systems see [44]. We need numerical integration to get from a differential equation to a discretized form of a trajectory. Only the latter can be used to train and test the reservoir computing framework.

3.2 Runge-Kutta Method

We are using the 4th order Runge-Kutta Method [37] for numerical integration of initial value problems. It is based on evaluating substeps in order to yield a higher accuracy than fullstep Euler Method. Thus, knowing the dynamical equations that define our system in question and choosing an initial value, one yields a discrete approximation of a trajectory. The only choices to make are the initial value and the time increment dt. The basin of attraction of the systems used in this thesis is large enough, that none of the initial conditions used did not converge to the attractor. In addition, we chose our starting points from a very long trajectory, adding small scale noise to decouple the resulting trajectories from each other.

3.3 Modified Lorenz System

One very canonical choice for a chaotic system are the Lorenz equations describing simplified convection. There are many versions of these equations in the literature. We will set out from the Lorenz 63́ system [23], as is done in [17, 34, 24, 25]. It exhibits a

$$(x, y) = (-x, -y) \tag{3.1}$$

symmetry [25], that is broken by introducing an extra term $+x$ in the z component [17]. This yields the modified Lorenz system

$$\dot{x} = -ax + ay \tag{3.2}$$
$$\dot{y} = bx - y - xz \tag{3.3}$$
$$\dot{z} = -cz + xy + x \tag{3.4}$$

with $a = 10$, $b = 28$, $c = \frac{8}{3}$.

To justify the name *butterfly attractor* Figure 3.1 gives an impression of how the trajectories of the modified lorenz system look like.

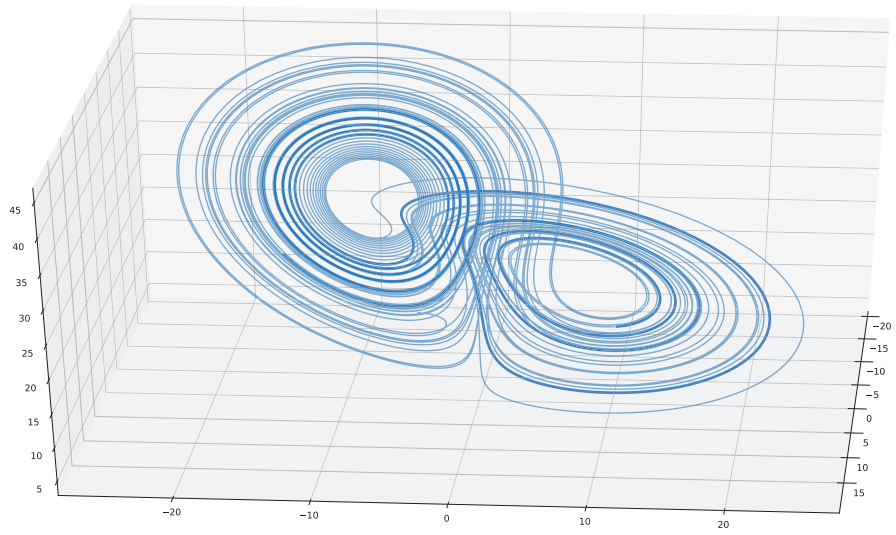


Figure 3.1: Lorenz Attractor
A three-dimensional plot of a simulated modified Lorenz trajectory. The two wings of the attractor are clearly visible.

Chapter 4

Measures

The reservoir computing training, described in section 2.2 General Formalism, is set up to minimize the error between the reservoir output and the simulated data. One natural measure, apart from that error, that emerges from the setup is to count the number of time steps the prediction stays within a predefined neighborhood of the test trajectory. We call this measure *demerge time* [17]. We know from experience that prediction will eventually depart from the simulated trajectory, so we want to characterize the long-term behavior of the prediction as well. Chaotic dynamical systems can be characterized by their sensitivity to initial conditions (largest Lyapunov exponent, see section 4.2) [32, 45] and by the fractal dimension of their attractor (correlation dimension, see section 4.3) [47, 11, 45]. In the following we describe how these two measures are obtained in a data driven manner[1], in order to apply them to the prediction. Finally, this enables us to compare the short-term accuracy as well as the statistical properties of the prediction compared to the simulation.

[1]The reservoir computing framework developed here is only applied to synthetic data throughout this thesis. Nevertheless, the goal was do built a set of methods that are applicable to time series without access to the dynamical equations behind. Therefore, the calculation of measures is based on time series data only.

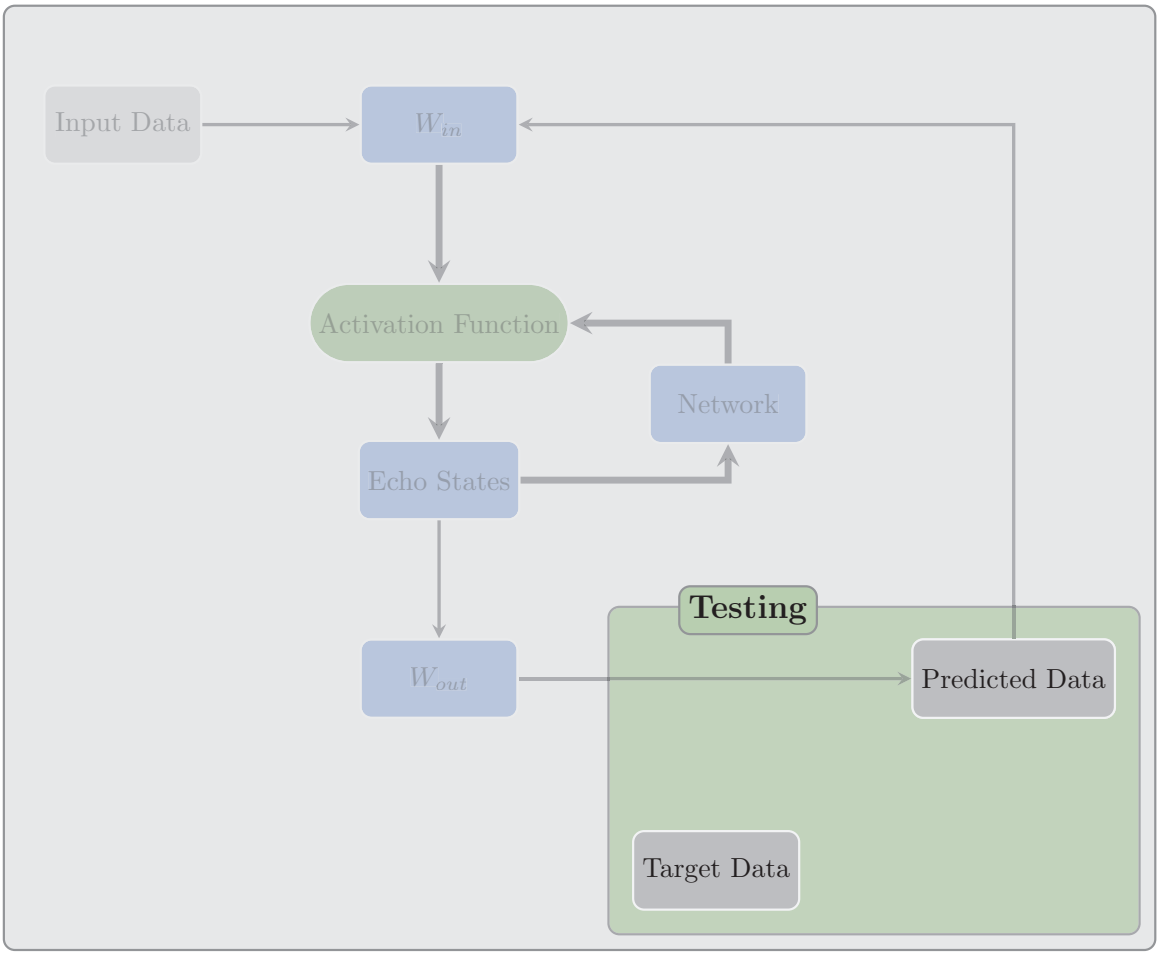


Figure 4.1: Reservoir Computing Setup - Testing
Testing the prediction means comparing it to the target data. The different measures for comparison are explained in detail in the following.

4.1 Demerge Time

The Demerge Time measures the interval length of very accurate short-term prediction. It is given by the number of timesteps after which the predicted trajectory differs more than ϵ from the true trajectory in one of the dimensions. Since the spread of the Lorenz attractor is non homogenous we set ϵ to 15 % of the spread of the training trajectory in each dimension respectively. In case of the modified Lorenz system this yields a triggering difference of $\epsilon = (5.8, 8.0, 6.9)$.

4.2 Lyapunov Exponent λ

The spectrum of Lyapunov Exponents is a measure to characterize chaotic behavior in bounded dynamical systems. Especially the largest Lyapunov Exponent quantifies the sensitivity to initial conditions, since it measures the separation rate of initially nearby points [44]. To calculate the largest Lyapunov Exponent from a given trajectory, one can use Rosenstein-Kantz method using next neighbors only [22]. See Appendix B for details of the algorithm.

4.3 Correlation Dimension D_{cor}

Strange attractors can be characterized through their fractal dimensions. We use the correlation dimension in this thesis. The correlation sum is given by

$$C_r = \lim_{N\to\infty} \frac{1}{N^2} \sum_{i,j}^{N} \theta\left(r - |\vec{x}_i - \vec{x}_j|\right), \tag{4.1}$$

counting all distances between points of the attractor within a certain radius. For the correlation dimension we are interested in the scaling exponent of the correlation sum with the radius:

$$C(r) \propto r^{\nu}. \tag{4.2}$$

Although, strictly speaking ν is only an upper bound on D_{cor} we will speak of ν as the correlation dimension in the following [15].
By visual inspection, the range of radii for the sphere was set to 0.5 to 5.0. The lower bound is affected by the density of points, namely the simulation time scale and the length of the trajectory. The upper bound depends on the total diameter of the attractor.

4.4 Assessing Lyapunov Exponent and Correlation Dimension

Depending on the length of the trajectory and the simulation time constant, the measures vary significantly. We used 5000 time steps and $dt = 0.02$ throughout this thesis. To check if the measures converge to the values in literature for long enough trajectories, we compare them for different trajectory lengths. As Figure 4.2 indicates, the shorter the time series the more the Lyapunov exponent is underestimated. The longer the time series, the less spread one obtains. The correlation dimension of the modified Lorenz system, using 100 trajectories with 50000 time steps each, calculates to 2.039 ± 0.005 , which is within the literature range for the proper Lorenz system of 2.05 ± 0.01.

Figure 4.3 Correlation dimensions for different trajectory lengths shows that the same overall behavior holds for the correlation dimension. The longer the trajectory the more the results approach the literature value. from below, but never reaches it. It is important to note that for our choice the measured correlation dimension is almost always less than two. The Lyapunov exponent calculates to

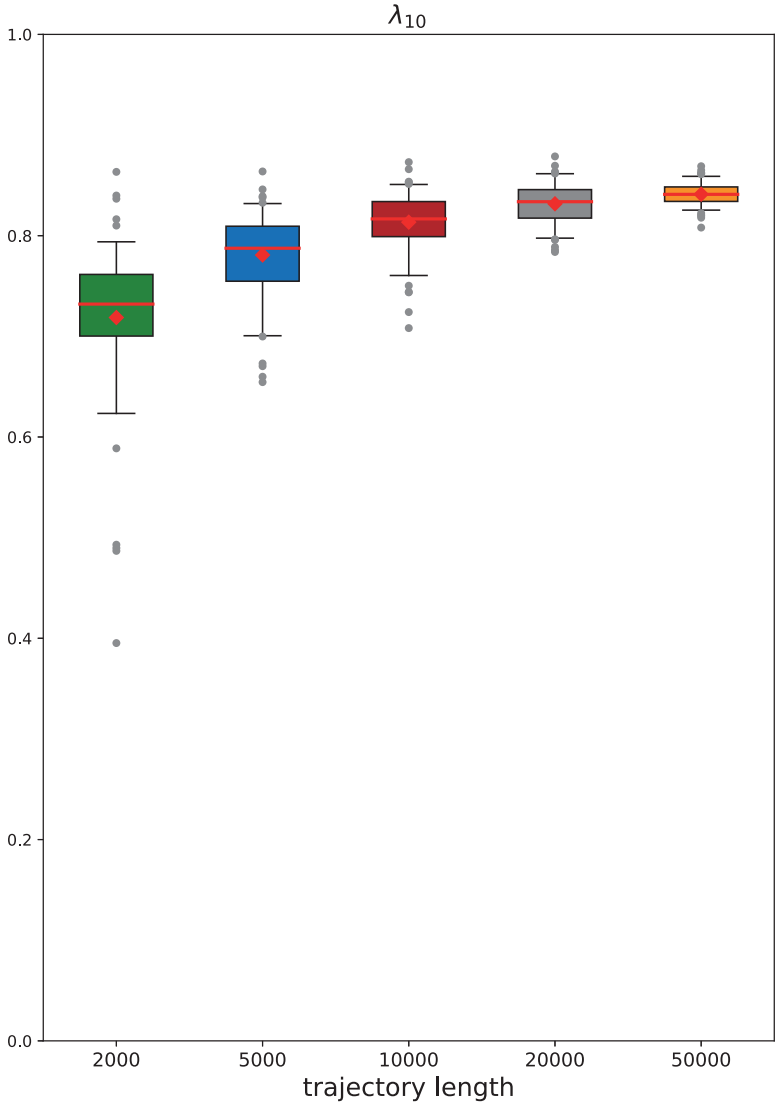


Figure 4.2: Lyapunov exponents for different trajectory lengths
Each box shows the distribution of Lyapunov exponents for 100 trajectories. The boxes correspond to different trajectory lengths. The boxes correspond to the inner two quartiles while the whiskers represent the 5th and 95th percentile. The mean is indicated as a red diamond, while the median is shown as a red line. The temporal threshold for identifying next neighbors is 10 time steps.

0.841 ± 0.010, while the value for the proper Lorenz system is is 0.89. This yields a Lyapunov time scale of $\tau_\lambda = \frac{1}{\lambda} = 1.189 \pm 0.014$. As we were using training and testing trajectories of 5000 steps[2] the Lyapunov exponent and the correlation dimension is systematically underestimated by 5.5 % and 0.5% , respectively. This is due to the lacking sampling density, which is also the case for predictions of the same length. We will use the measures only for relative comparison between true and predicted trajectories.

[2]This is the result of a compromise between capturing the properties of the attractor and saving computation time during simulation, training and prediction.

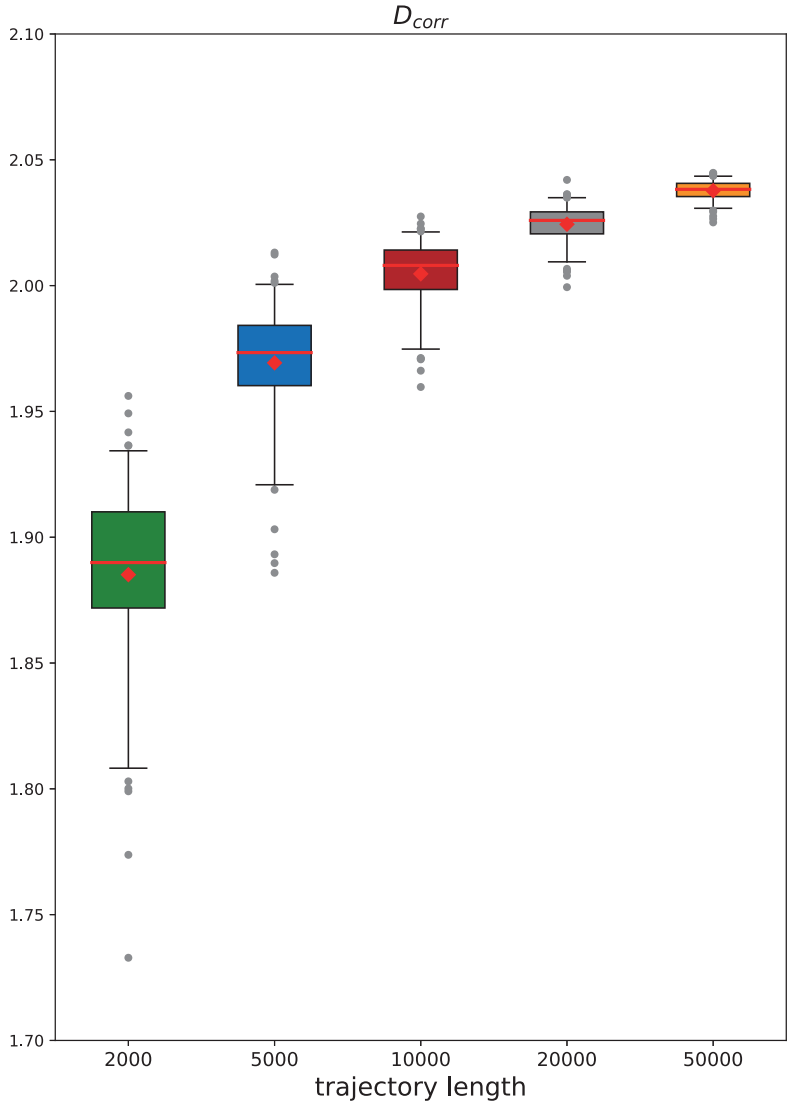


Figure 4.3: Correlation dimensions for different trajectory lengths
Each box shows the distribution of correlation dimensions for 100 trajectories. The boxes are plotted for different trajectory lengths. The boxes correspond to the inner two quartiles while the whiskers represent the 5th and 95th percentile. The mean is indicated as a red diamond, while the median is shown as a red line.

Chapter 5

Experiments

In the following a set of experiments is described. The goal was to get a deeper understanding of how reservoir computing works and how network structure affects replication of chaotic time series. After an explorative grid search in section 5.1, we focus on the network topology in section 5.2. Followed by section 5.3, where we effectively reduce the size of the network, in order to enhance prediction accuracy. In the last part section 5.4 we look for a relationship between predictive power and closed loop properties.

5.1 Parameter Exploration

First, we aim to get a better understanding for the different parameters of our setup. Therefore, we performed a parameter grid search. The most important relationships are presented and discussed in the following.

5.1.1 Parameterspace and Dataset

The grid search was carried out over a range of W_{in} scales, spectral radii r, network sizes N and three different network topologies and different degrees.
Small world and scale free networks are investigated in the publications by Haluszczynski et al. [17]. Cui et al. [8] also tried a mixture of the two and different network sizes showing a good time series prediction capability of mixed networks with little decrease in memory capability when using time series from autoregressive models. To complement this work, we tested random, scale free and small world networks of varying size and average degree using chaotic time series.
In order to enable a statistical analysis of the results later we perform 50 realizations per gridpoint. Each starting from a randomly chosen point from a list of 5000 starting points, resulting from randomly selected points from a sufficiently long simulated trajectory with small scale uniform noise to yield independent trajectories in the end. The parameters are shown in Table 5.1. The regularization parameter is fixed at $\beta = 0.0001$, unless otherwise indicated.

Table 5.1: Gridsearch Parameterspace

Parameter	Values
topology	random, scale free, small world
N	$100, 200, 300, 500$
degree	$2, 4, 6, 10$
r	$0.15, 0.30, 0.45, 0.60$
W_{in} *scale*	$0.15, 0.30, 0.45, 0.60$

5.1.2 Gridsearch Results and Discussion

In the following the relation between different parameters and the predictive power is analyzed.

Dependence on r and W_{in} scale In Figure 5.1 Gridsearch results for random, $N = 100$ networks scatter plots for different spectral radius and W_{in} scale combinations are shown. It is clear from visual inspection that the results exhibit less spread around the red target ellipse for smaller r and W_{in} scale values. If one of the parameters takes the smallest possible value of 0.15 the results are still acceptable, eventually showing large spread as both parameters take larger values. We only show the values up to 0.45 here as the trend is clearly visible.

Further plots for other topologies and network sizes are shown in Appendix C Further Gridsearch Results. Next, we compare different network sizes with fixed density of $\rho = 0.02$, as in [34, 24]. Based on Figure C.1 and Figure C.2 scatterplots, we identify 0.15 for both measures as the best overall performing value, taking all 3 topologies into account.

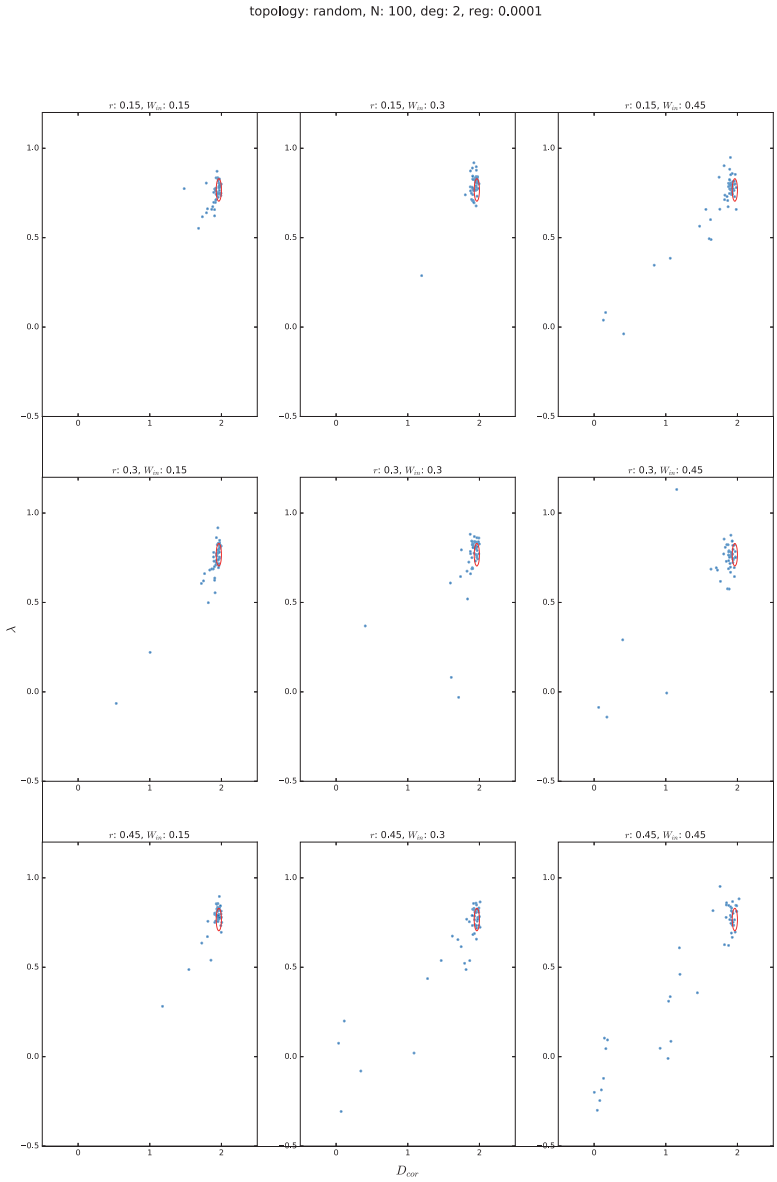


Figure 5.1: Gridsearch results for random, $N = 100$ networks
Each plot shows the correlation dimension versus the Lyapunov exponent for varying spectral radius r and W_{in} scale. r increases from top to bottom, while W_{in} increases from left to right. The network topology is random, the size $N = 100$ with $\rho = 0.02$. The red ellipse marks the 5th and 95th percentile of the respective measure for the simulated trajectories.

Dependence N In the next step we compare the correlation dimension distributions for individual network sizes. In order to represent the distributions for each N a range of percentiles are displayed in each plot. The values for predicted trajectories, as well as the simulated, are shown. Figure 5.2 depicts the results for random topologies, while Figure 5.4 and Figure 5.3 contain the small world and scale free network data, respectively. Especially the 5th percentile [lowest gray dashed line] reveals the differences between topologies. While random and scale free networks seem to have a larger spread at smaller networks, small world achieves the best results for $N = 100$. As seen in Figure 5.1 and section 5.1 the performance increases with size. It should be kept in mind, that the percentiles are taken from 50 values per N only and therefore are by far not noise free. For $N = 300$ all three topologies show little spread in correlation dimension, increasing slightly when going to $N = 500$.

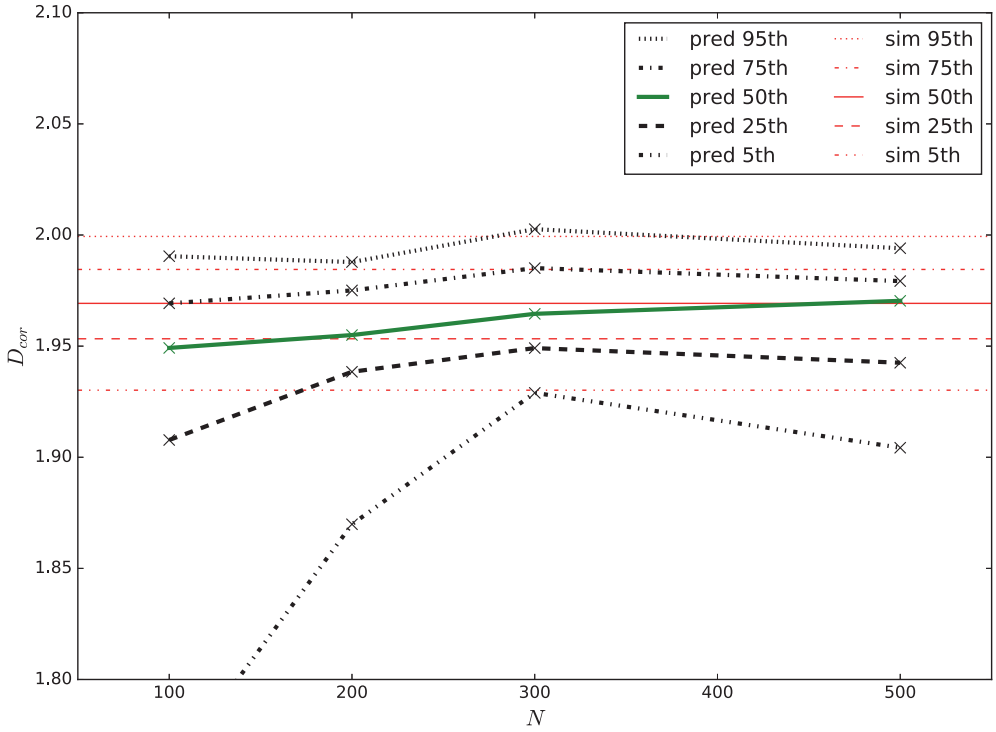


Figure 5.2: Correlation dimension for random networks of varying size
The plot shows data for $r = 0.15$ and $W_{in} = 0.15$. The topology is random and a fixed average density of 0.02 is used. The black and green lines show different percentiles of the correlation dimension distribution for predictions [pred]. The horizontal red lines show the same percentiles of the range of correlation dimensions calculated for the simulated trajectories [sim]. The lines represent the 95th, 75th, 50th, 25th and 5th percentile from top to bottom. The corresponding line style is given in the legend. The solid lines indicate the mean, in red for simulation and green for prediction. The prediction quality in terms of correlation dimension increases with N.

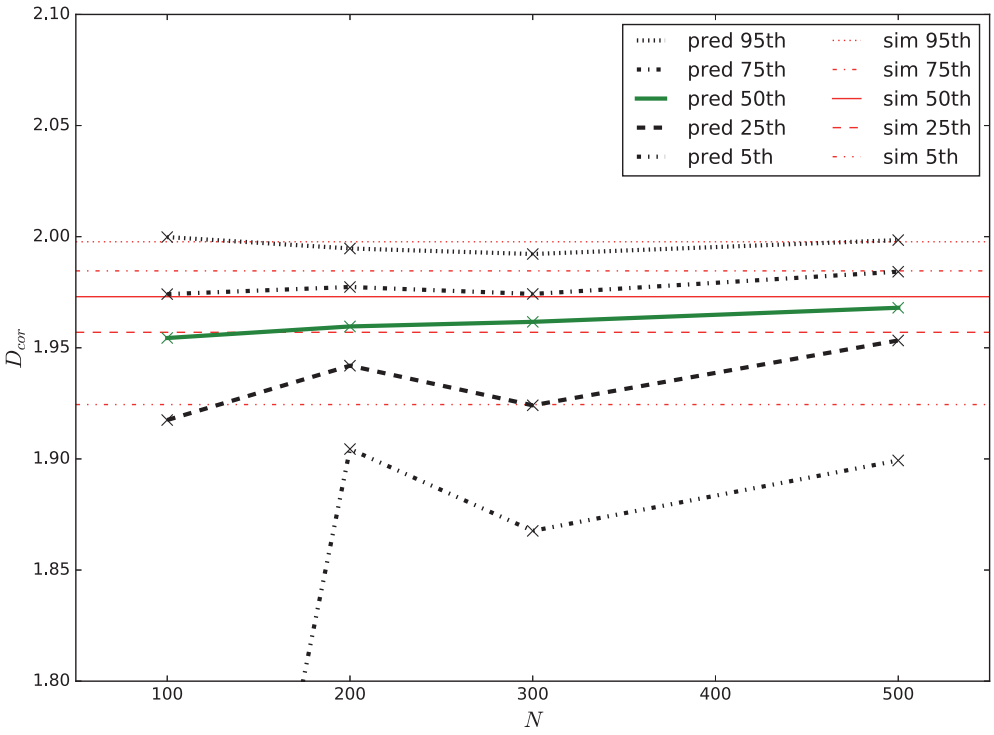


Figure 5.3: Correlation dimension for scale free networks of varying size
For a detailed description see Figure 5.2. The topology is scale free.

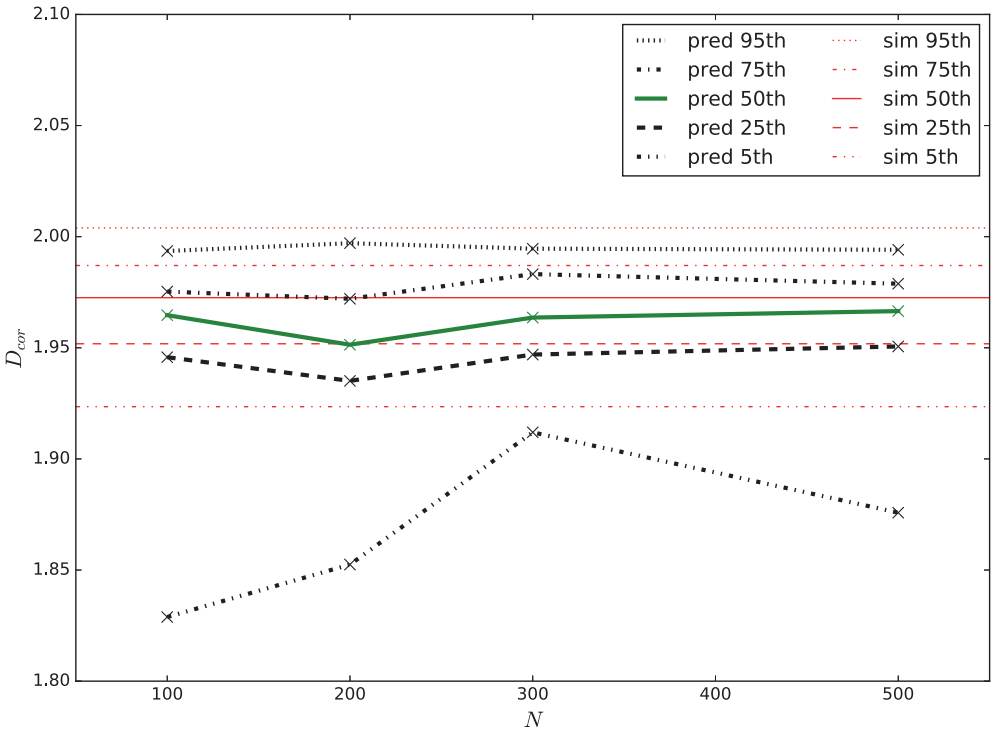


Figure 5.4: Correlation dimension for small world networks of varying size
For a detailed description see Figure 5.2. The networks have small world topology.

Dependence on ρ Even if in many examples in literature the networks used are sparse, the densities vary between $1.25\% - 20\%$ [34, 24, 17, 18, 52]. Therefore, the correlation dimension distribution over different densities ρ are compared for fixed network topologies and sizes. The plot type is the similar to Figure 5.2, Figure 5.3 and Figure 5.4 with varying ρ instead of N. As above the 5th percentile line exhibits the strongest dependence, as only a small fraction of predictions is not capable of reproducing the attractor structurally. Figure 5.5 shows a strong dependence on density for $N = 100$ random networks. The 5th percentile shows the most significant variations with a peak of marginally above 1.9 with an average degree $d = 6$. For comparison, the 5th percentile for simulated trajectories lies at 1.927, indicating good performance of this particular random network type.

Figure 5.6 and Figure 5.7 display only weak density dependence for small world and scale free. The 5th percentile for the prediction of scale free networks slightly exceeds 1.7 only for a density of 16%, while the small world networks are around 1.8 for almost all values except $\rho = 4\%$. The highest 5th percentile small world value is 1.829 for $d = 2$. Small world lies between scale free and random. As in Haluszczynski and Räth [17] the scale free topology yields the worst results, indicated by the broadest correlation dimension distribution. Therefore, we will focus our analysis on small world and random networks in the next part.

The 3 network topologies are characterized by their degree distribution[1]. In order to find an explanation for the decreased predictive power of scale free reservoirs, Figure 5.8 shows the largest magnitude W_{out} value per node plotted against its degree. It is obvious, that high degree nodes have lower weights associated with them. This might give an intuition why scale free networks are disadvantageous for predicting chaotic attractors in reservoir computing. The similar distributions for random and small world networks resemble the comparable results in Figure 5.5 and Figure 5.6. The correspondence between W_{out} and prediction quality is investigated further in section 5.3 Node Removal. But first the transition between regular, random and small world topology is investigated in the next section.

[1] As described in section 2.3 Network, path length is also a characteristic measure.

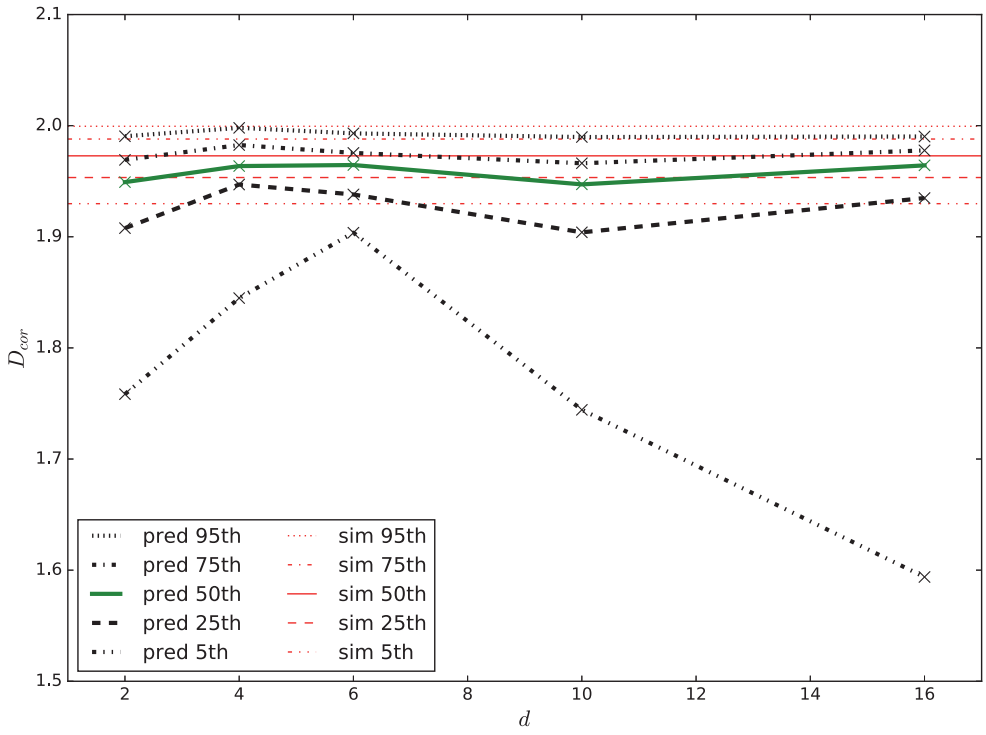


Figure 5.5: Correlation dimension for random, $N = 100$ networks of varying ρ
The figure is analog to Figure 5.2. Red lines show the percentiles of the correlation dimension distribution for each network density for simulations, gray and green lines the ones for predictions. The network topology is random and $N = 100$.

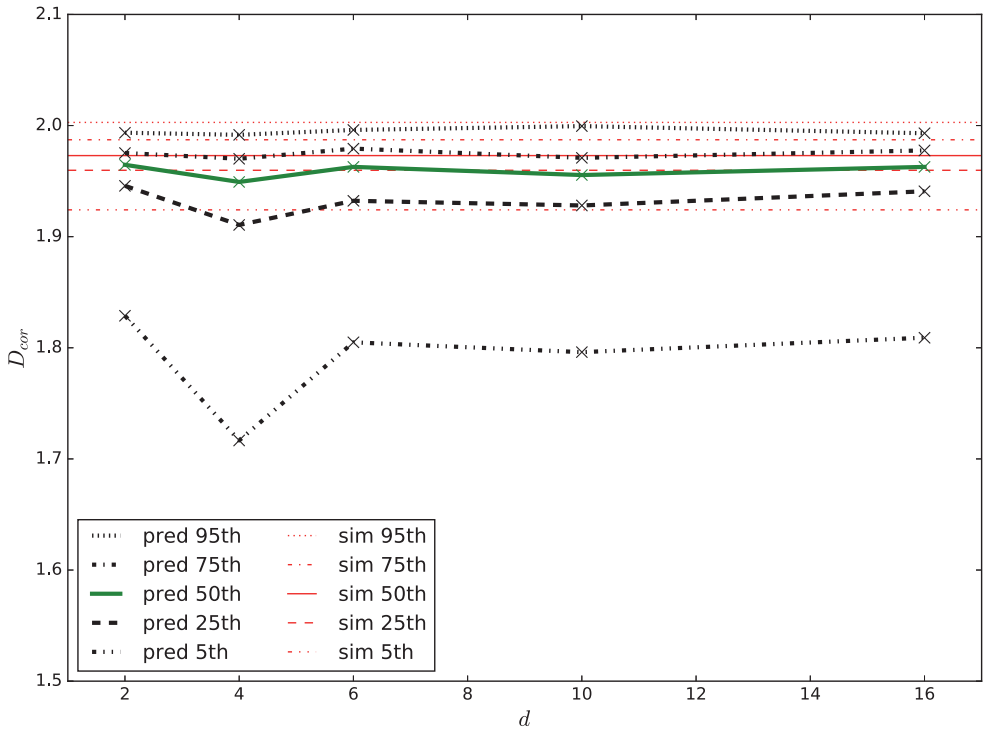


Figure 5.6: Correlation dimension for small world, $N = 100$ networks of varying ρ
The description is similar to Figure 5.5. The network topology is small world and $N = 100$.

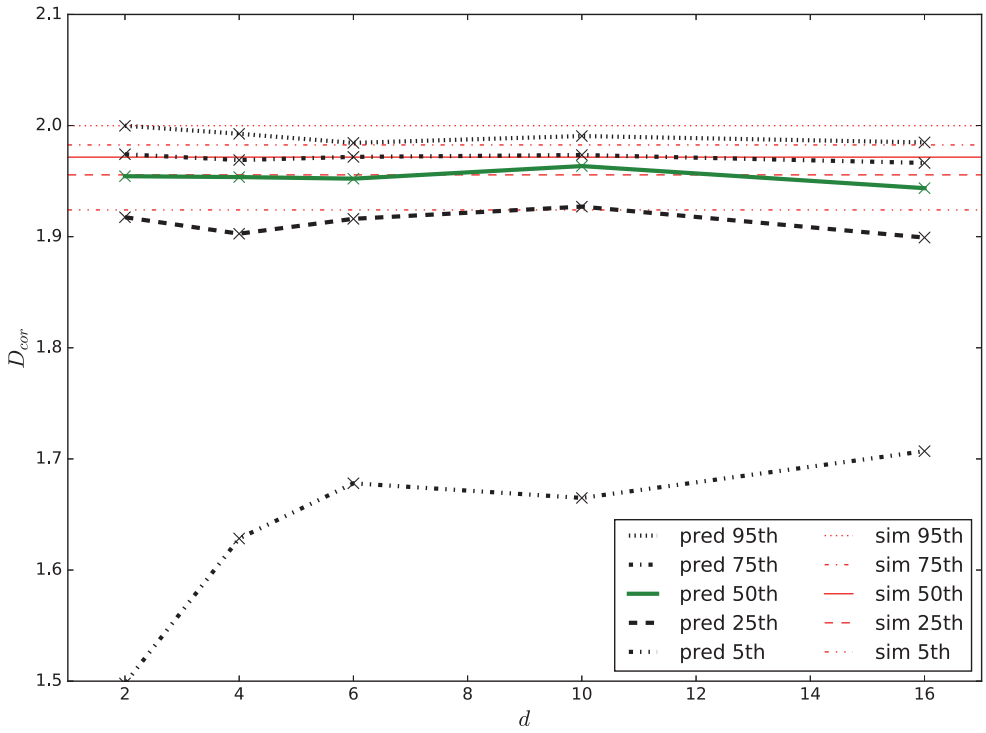


Figure 5.7: Correlation dimension for scale free, $N = 100$ networks of varying ρ
The description is similar to Figure 5.5. The network topology is scale free and $N = 100$.

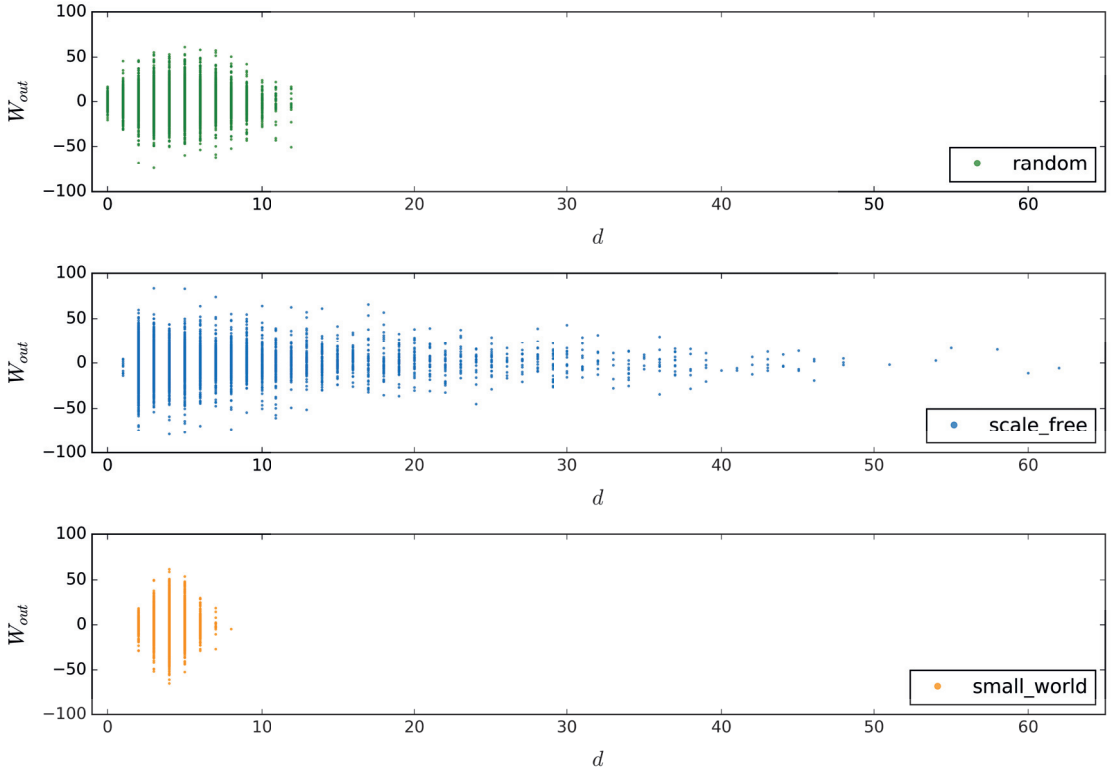


Figure 5.8: W_{out} and degree for different topologies
The network size is $N = 200$, the density $\rho = 0.02$. The W_{out} weight with the largest magnitude is compared with the degree of the respective node. The different topologies, starting at the top are: random [green], scale free [blue], small world [orange]. For each topology 100 realizations are shown.

5.2 From Regular to Small World to Random Networks

Now, we will exploit the fact that by varying the rewiring factor p_{sw} in generating small world networks, there is a continuous transition from regular to small world to random networks[2] [50].

5.2.1 Varying Rewiring Coefficient and Network Density

In this section we evaluate the effect of p_{sw} and d on long- and short-term prediction quality. In the following figures adjacency matrices of regular, small world and random networks are visualized. The network size is 200, the degree 4. The white grid entries correspond to zero entries in the matrix, the red ones to non-zero ones[3]. Figure 5.9, Figure 5.10 and Figure 5.11 show p_{sw} values of 0%, 30% and 100%, respectively. The chosen parameters for the upcoming experiment are given in Table 5.2. Each

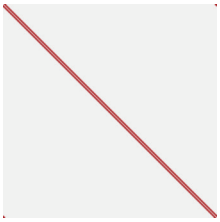


Figure 5.9: Regular network with $p_{sw} = 0.0$

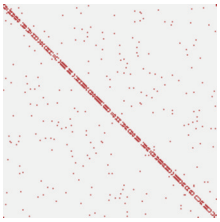


Figure 5.10: Small world network with $p_{sw} = 0.3$

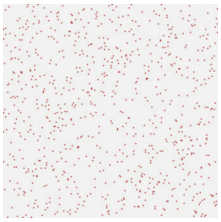


Figure 5.11: Random network with $p_{sw} = 1.0$

of the parameter tuples is sampled 250 times. The fixed parameters are chosen based on the previous results, discussed above in section 5.1.2.

Table 5.2: Parameters for varying p_{sw} and ρ

Parameter	Values
topology	regular → small world → random
p_{sw}	$0., 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0$
degree	$2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30$
N	200
r	0.15
W_{in} *scale*	0.15
β	0.0001

Figure 5.12 shows correlation dimension percentiles with respect to different degrees of rewiring p_{sw} and density ρ. Similar to the figures in section 5.1 Parameter Exploration the 5th percentile exhibits the most fluctuations. For higher percentiles the predictions are closer to simulated values. Densities in the range from 0.02 to 0.06 perform slightly better than higher and lower ones. The predicted correlation dimension is robust with respect to the rewiring coefficient.

[2] See section 2.3 Network for further details and the pioneering study of Watts and Strogatz [50] Fig.1 for a simple but striking graphical representation.

[3] In this visualization we focus on the topological aspect and neglect random number assignment in the visualization. The thickness of the red diagonal corresponds to the degree, as every subdiagonal is a next neighbor link. The diagonal itself is always zero, since we neglect self-loops in this work.

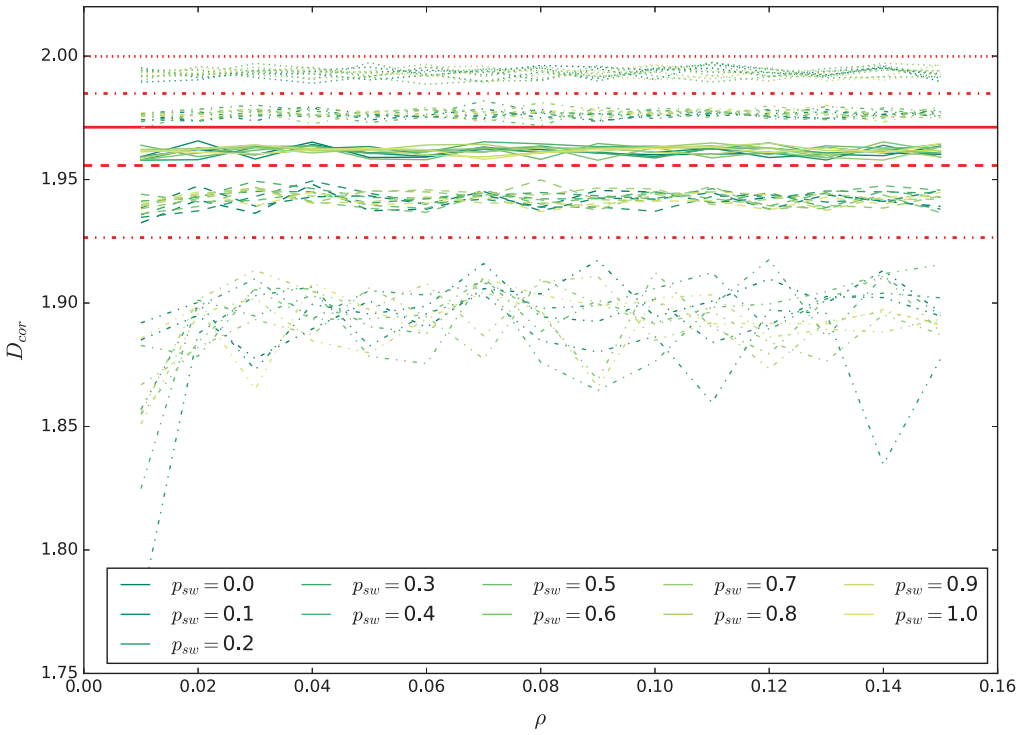


Figure 5.12: Correlation dimension for varying ρ and p_{sw}
The lines display different percentiles of the correlation dimension distribution for simulated and predicted trajectories. The line styles and the corresponding percentile from top to bottom for predictions [green] and simulations [red] are: dotted - 95th, dashdotted - 75th, solid - median, dashed - 25th, dashdotdotted - 5th. The different shades of green represent the rewiring coefficients p_{sw}. The lower the percentile, the higher the variation amplitude, but there is no strong dependency of correlation dimension distribution on density ρ or rewiring coefficient p_{sw}.

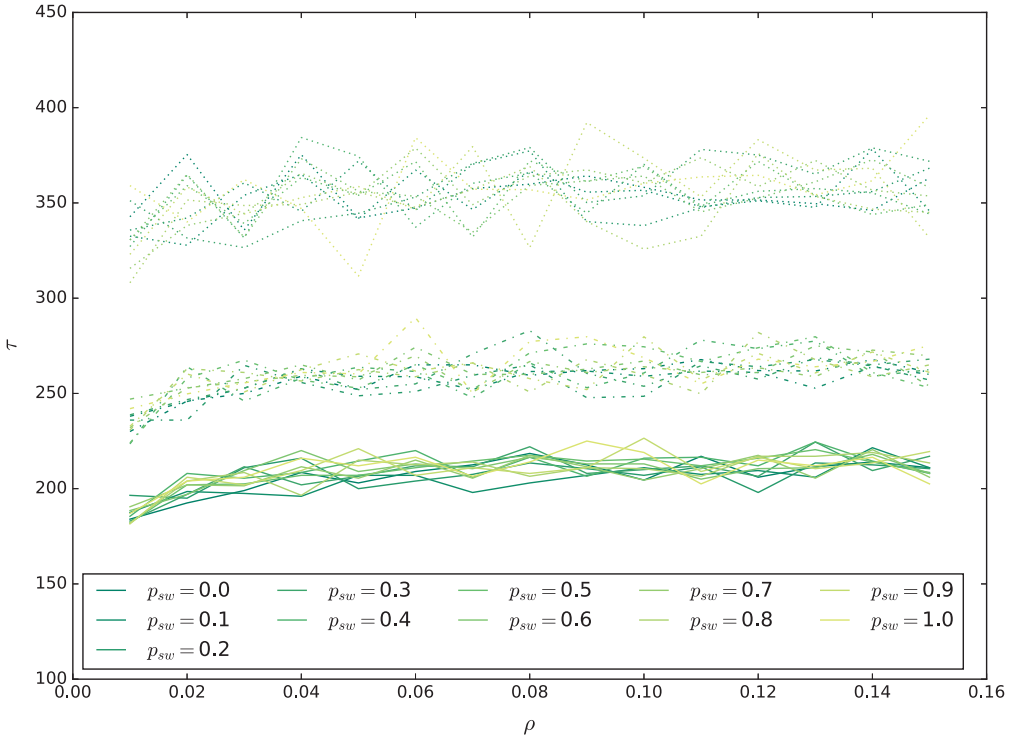


Figure 5.13: Demerge time for varying ρ and p_{sw}
The green lines depict the demerge time distribution of the predictions. The sets of curves from top to bottom represent the 95th, 75th and 50th percentiles. The line style coding is analog to Figure 5.12 As well as the different shades of green represent the rewiring coefficients p_{sw}. In as sets there is a slight trend to higher demerge times for higher network densities.

In order to obtain a more detailed picture of short-term prediction accuracy, we take a look at short-term prediction accuracy measure τ in Figure 5.13. Again, none of the rewiring coefficients [shades of green] shows a distinct behavior. Each percentile plotted varies in a comparably small corridor, set off from the simulated value. The before mentioned performance advantages for lower densities cannot be confirmed. There is a slight trend towards longer demerge times for higher degrees.

5.2.2 Rewiring Coefficient Conclusion

We conclude by noting, that the rewiring coefficient is neglectable with respect to short-term prediction accuracy and statistical attractor reconstruction. In terms of demerge time, denser networks perform slightly better. However, the statistical spatial characteristics, measured by correlation dimension, is reproduced moderately more precise by degrees between 0.02 and 0.06. We use random topology with a density of 0.02 in the following, as the dependence on it is weak and relatively sparse is the canonical choice[4].

[4]Jäger is using a density of 0.0125 [18], Pathak et. al [34], Lu et al. [24] and Haluszczynski and Räth [17] use 0.02.

5.3 Node Removal

We set out from random networks, now trying other means of varying network structure. In the following section we investigate the effect of removing nodes from the network with respect to prediction quality. Inspired by the *attack tolerance* idea [2] we remove nodes and their corresponding edges from the network and W_{in} [5]. Instead of diameter and cluster size we are interested in the change of the reservoir's climate replication capabilities.

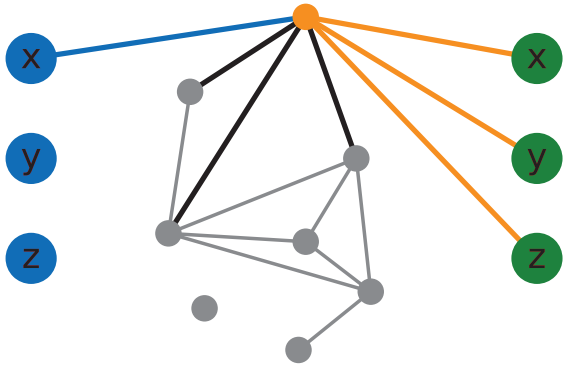
5.3.1 Node Removal Procedure

We remove a certain fraction of the N nodes, depending on their respective W_{out} values, since we assume a relation between importance and assigned W_{out} weight of a node. Firstly, we train the reservoir to yield the W_{out} matrix. Next we sort the nodes with respect to their maximal[6] absolute value of W_{out} weight. Finally, certain fractions of nodes are removed according to this order and the resulting networks are trained once again [7]. The node removal approach is visualized in Figure 5.14. Since the removal procedure, apart from the network size and the W_{in} weight distribution, changes the spectral radius, we scale each *random* reference network [section 2.3 Spectral Radius] accordingly, to prevent interference effects with the spectral radius.

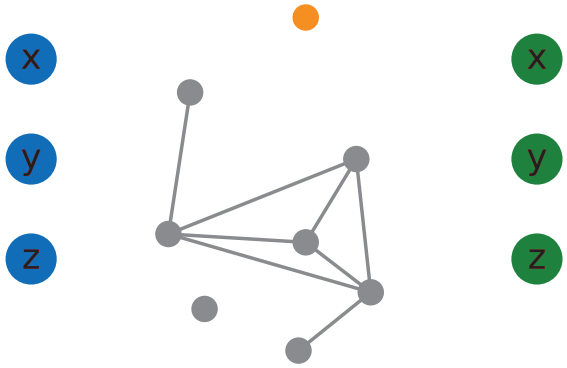
[5] This might resemble pruning of deep neural networks. In difference to pruning, the total network size is altered, as there is only one layer of trained weights in reservoir computing. The effect of modifying network structure of dynamical systems is nontrivial. Recent attempts on pruning RNNs are presented in [51, 31]. The approach presented here aimed to apply the idea of attack tolerance to networks in dynamical systems.

[6] Each node has as many W_{out} weights as the dimension of the dynamical system.

[7] There are obviously many other ways of defining the selection e.g. by taking the average of weights for each node, or even taking other criteria than W_{out} e.g. by looking at the echo states.



In this simplified representation of a trained reservoir we highlight the connections for one specific node [orange] only. The blue nodes correspond to input nodes and the blue edge to the W_{in} weights corresponding to the orange node. W_{out} [orange] connects the selected node with the output nodes [green]. Connections from the selected node to the rest of the network represented in black, while further nodes and edges are only schematically depicted [gray].



After removing all edges from the node [orange], we can also neglect the node itself. Therefore, the network is altered in size, spectral radius and degree distribution and also the distribution of W_{in} weights is changed.

Figure 5.14: Node Removal Schema

5.3.2 Parameters and Dataset

In the following we want to compare the performance of a reduced reservoir [red], one where we removed a certain fraction of nodes, with the original one [$N = 200$]and a random one with the same size and spectral radius as the reduced one. All of the plots in this section show a specific measure, explained in chapter 4 , and compare the prediction of these three networks and the simulation [sim] with each other. We are again using the modified Lorenz system , the size of the original random network was set to $N = 200$ with a average degree of 4, according to what we found before.

Since reservoir computing can be trained very quick, we used a simple *random search* with uniform sampling in order to find suitable hyperparamters. The hyperparamter space is spanned by spectral radius[8], W_{in} scale[9] and regularization parameter. The latter was rescaled to a logarthmic scale for better coverage of small values[10]. The objective function was the demerge time averaged over 30 realizations. Each realization uses one of the 5000 time series, as explained in detail in subsection 5.1.1 Parameterspace and Dataset, and random initializations of the complete reservoir computing system. The optimal values are shown in Table 5.3.

Table 5.3: Results of Hyperparamter Optimization for Node Removal Experiment

r	0.17
W_{in} scale	0.17
β	1.9×10^{-11}

[8] r range: 0 to 2.5
[9] W_{in} scale range: 0 to 2.0
[10] β range: $\log_e 10^{-11}$ to $\log_e 0.1$

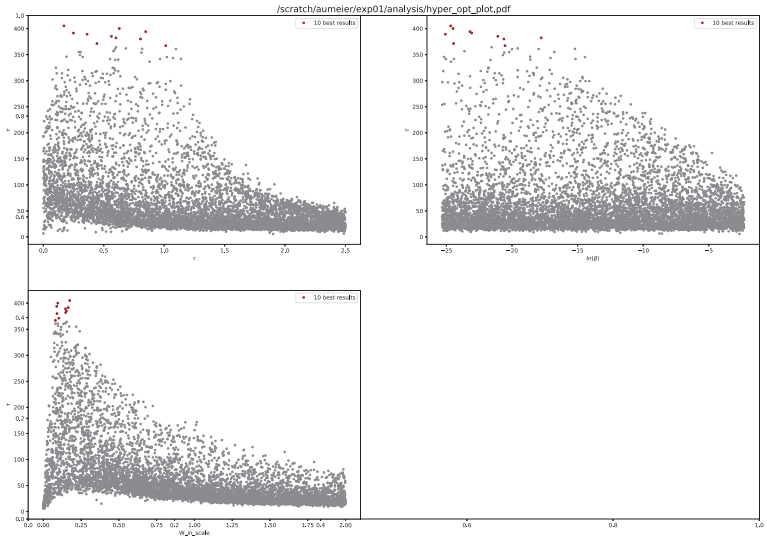


Figure 5.15: Hyperparameter optimization results
Projections of mean demerge time onto the three parameters, spectral radius [top left panel], logarithmic regression parameter [top right panel] and W_{in} scale [lower left panel]. The mean was taken over 30 realizations with the according parameters. The 10 highest averages are plotted in red.

5.3.3 Node Removal Effect on Prediction Quality

Removing 10 percent We begin with reducing the size of the network by 10 percent, removing the top, bottom as well as top and bottom (symmetrical) fraction of nodes ranked by their maximal absolute value of associated W_{out} weights. The results are shown in Figure 5.16 and Figure 5.17.

The distribution of correlation dimension of the simulated target trajectory is shown to give an intuition about the natural spread of the measurement when applied to relatively short trajectories of only 5000 timesteps using $dt = 0.02$. Its mean can be regarded as the true value. When one compares the reduced networks with their respective reference networks, one clearly sees that each of them performs better, resulting in a slightly narrower 1st to 3rd quartile box and a significantly higher 5th percentile whisker[11]. Comparing the results across the three selection paradigms top, bottom and symmetric, it is obvious that removing the top weighted nodes has the narrowest inner quartile box and a more than 0.5 higher 5th percentile whisker than the original $[N = 200]$ system. This observation supports the interpretation that removing the seemingly most important part of the network facilitates the network performance in prediction.

The same overall behavior can be seen when looking at the Lyapunov exponent. The according plot can be found in section D.1 Lyapunov Exponents 10 Percent.

The demerge time distributions in Figure 5.17 all show a median around 400 and a 95th percentile around 600 time steps. While top [0.1] has the narrowest inner quartile box, its 95th percentile whisker is a little lower than the reference. None of the medians exceeds the large network $N = 200$ value 405.

The described differences between all the mentioned networks are still moderate which is due to small fraction of removed nodes.

[11] Every reservoir in this section was tested on the same set of 500 trajectories. Some plots contain less than these 500 data points per label, since the measures do not always converge, and the node removal scheme can yield networks without a defined spectral radius. In this cases the respective trajectories are removed for all labels for better comparability and their number is given in the description.

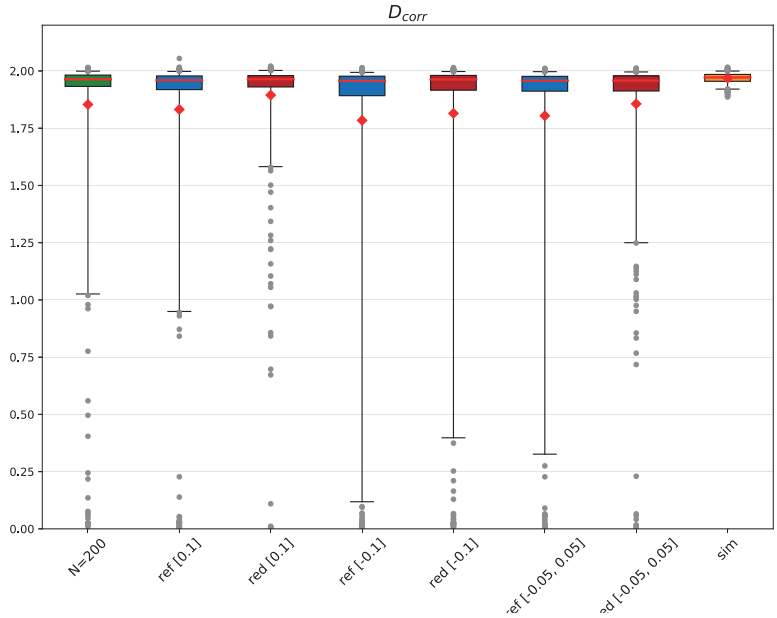


Figure 5.16: Correlation dimension 10%
Removed 2 trajectories. Only interpretation via 5th percentile whisker, everything else is good anyways due to large enough size of the reservoir.

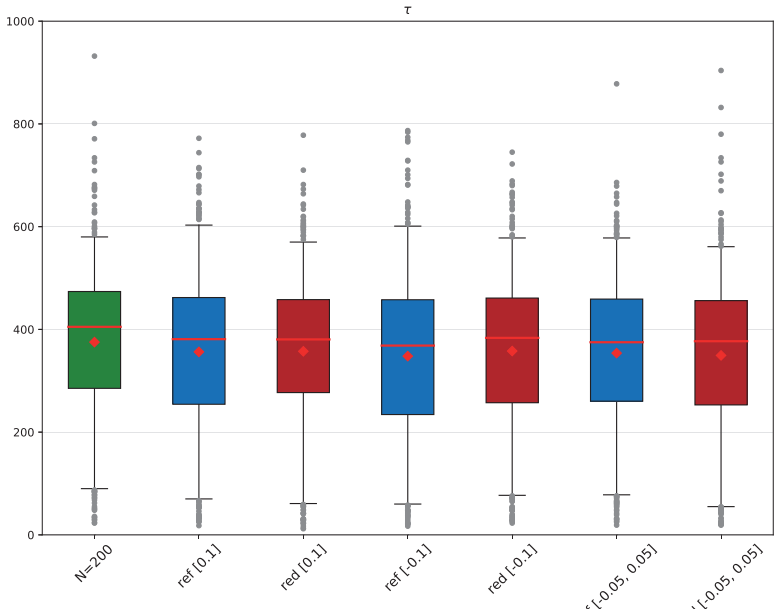


Figure 5.17: Demerge time 10%
Removed 2 trajectories. Boxplot and colors as in Figure 5.16 Correlation dimension 10%.

Removing 30 percent Next, we compare the correlation dimension results for networks reduced to $N = 140$ nodes. The network with top 30 % removed has the highest 5th percentile whisker of all networks and the least inner quartile spread of the small networks. The symmetrical reduced network performs almost as good. The highest spread is found for the red[-0.3] network. The same holds for the Lyapunov exponents, as shown in Figure D.2.

Looking at the demerge times a significant drop can be seen due to smaller network size. The best results can be seen for networks reduced from the bottom red[-0.3]. Its 75th percentile is the only small network exceeding 400 time steps. Top and symmetrical reduced networks are similar, and both outperform their reference networks.

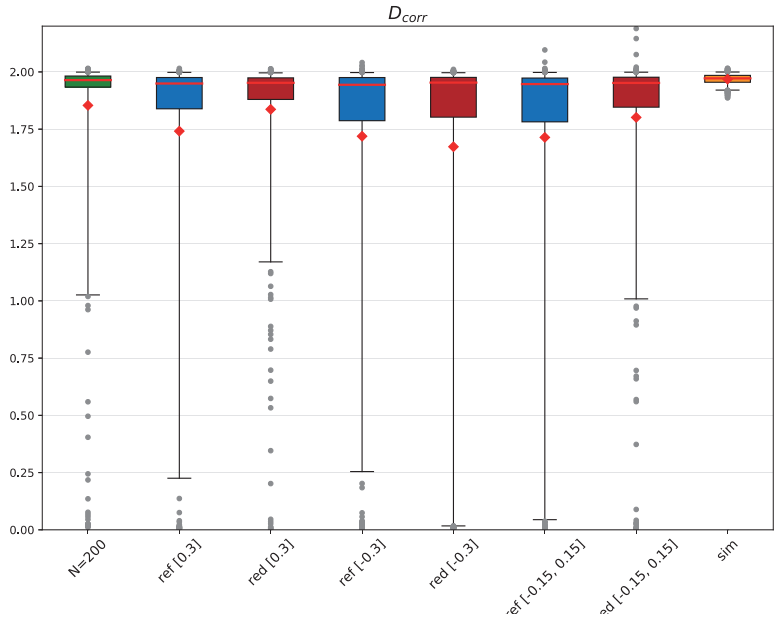


Figure 5.18: Correlation dimension 30%
Removed 3 trajectories.

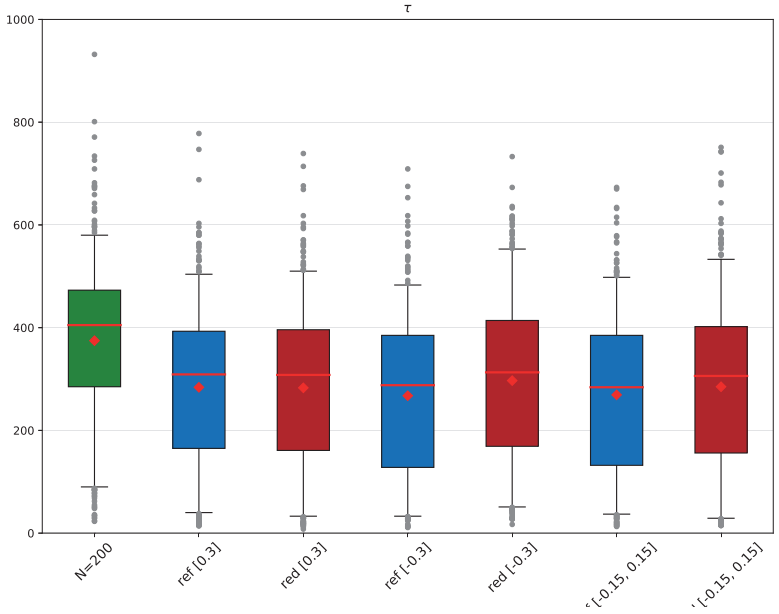


Figure 5.19: Demerge time 30%
Removed 3 trajectories.

Removing 40 percent We continue with reducing the original network more drastically by removing 40 percent of nodes. The following splits are investigated further: top, bottom, symmetric, bottom 30% and top 10% and vice versa 10% bottom with 30% top. We want to test the hypothesis that the associated W_{out} weights indicate the significance of the respective node. In comparison to $N = 140$ reservoirs the spread in correlation dimension (Figure 5.20) increased further, due to smaller network size. There are huge differences between the differnt splits when comparing correlation dimension distributions in Figure 5.20.

The red[-0.3, 0.1] split shows a good distribution from the 25th percentile on, but not from 0th to 25th. It is clearly visible from 5th and 25th percentile together with the mean, that top red[0.4] and red[-0.1, 0.3] show the best correlation dimension distribution, both relative to other splits and relative to their respective reference reservoirs. The best distribution is obtained from [-0.1, 0.3] as seen from a higher 5th and 25th percentile.

The overall tendency resembles the one of Figure 5.16 Correlation dimension 10%, that the nodes with highest W_{out} are not important to the networks dynamic. This could indicate that the top 30% are the most inhibiting ones. This will be tested in the following even more radical reduction scenario. For an analysis of the respective Lyapunov exponents see Figure D.3 Lyapunov Exponent 40%.

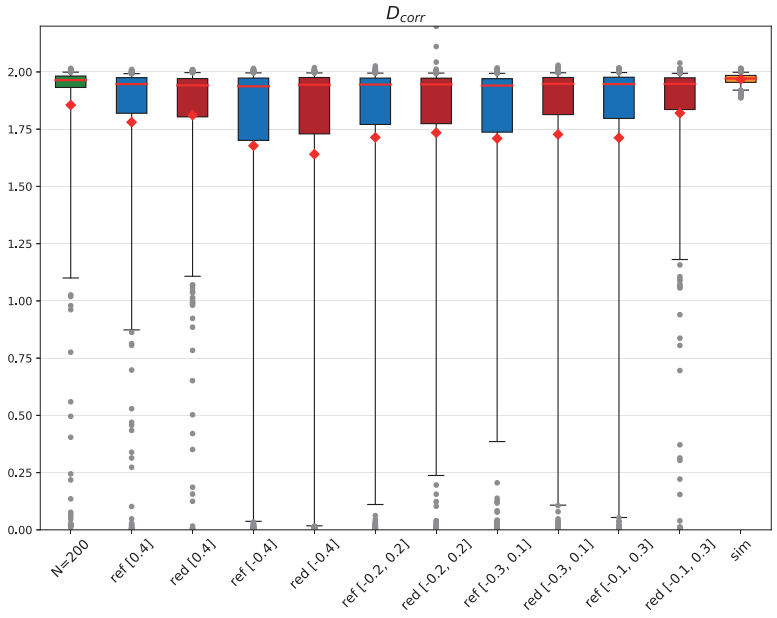


Figure 5.20: Correlation dimension 40%
The splits are labeled as follows: top [0.4], bottom [-0.4], symmetric [-0.2, 0.2], bottom 30% and top 10% [-0.3, 0.1] and vice versa 10% bottom with 30% top [-0.1, 0.3]. Removed 12 trajectories.

Demerge time differences also get more pronounced as the network size decreases. In 5.21 red[-0.3, 0.1] shows the best inner quartile, almost reaching 400 time steps. As well as the highest median 291, outperforming its reference network median by 39 time steps. red[0.4] is the first reduced network performing worse, than its reference network in terms of demerge time.

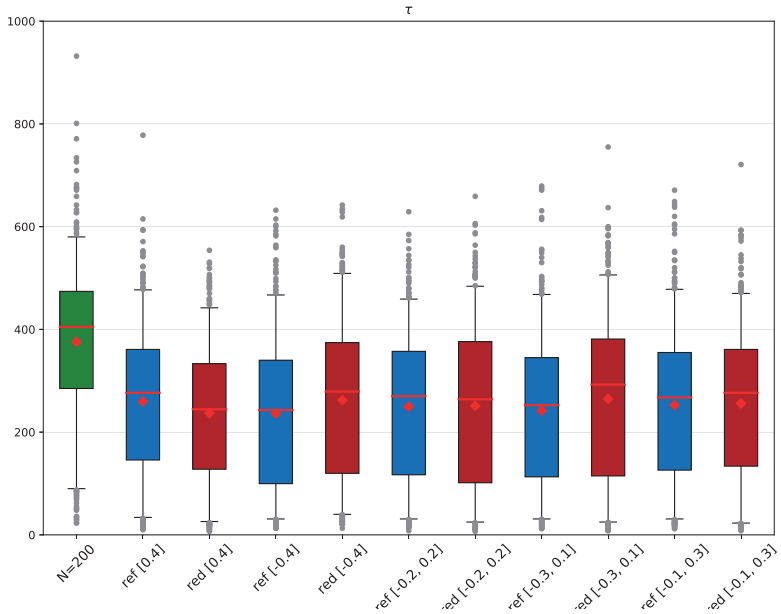


Figure 5.21: Demerge time 40%
The splits are labeled as follows: top [0.4], bottom [-0.4], symmetric [-0.2, 0.2], bottom 30% and top 10% [-0.3, 0.1] and vice versa 10% bottom with 30% top [-0.1, 0.3]. Removed 12 trajectories.

Removing 60 percent In the following we continue with removing 60 percent of nodes, yielding reduced networks of only $N = 80$. Since the network size is a crucial parameter (see Figure 5.2, Figure 5.4 and Figure 5.3), one could expect a significant decrease in prediction quality as seen in Figure 5.22 due to the small remaining network of only $N = 80$ nodes. In the following four different splits are compared with each other, namely top, bottom, symmetrically and in addition the reduced networks, where the bottom 50 percent and the top 10 percent were removed. We came to the conclusion, that around 30% of the higher weighted nodes seem to impede the networks performance. Therefore, we expect the [-0.3, 0.3] split to perform better than others splits.

For top [0.6] and bottom [-0.6] split the median as well as the inner quartiles [boxes] indicate worse performance then their respective reference networks. The relatively good performance of the symmetric and 50/10 split again contradicts the assumption, that absolute weight correlated with importance. It is evident that the symmetric split by far outperforms all others. Most remarkably its 25th percentile is 1.593 compared to 1.283 of its reference networks. Which is what we expected since we identified the top 30% as the most inhibitory. In addition, it should be noted that the median [red line] of the symmetrically reduced network lies within the range of the simulated data. That means, that 50 % of the reservoirs perform well in terms of correlation dimension, if one neglects the two overestimating realizations. The 5th percentile whiskers of all small networks lie in a neglectable regime already. This is why we base our discussion on inner quartiles and median value here, noting that the total distribution spread increases with shrinking network size.

Finally, we analyze demerge time distributions for $N = 80$ networks in Figure 5.23.

As for correlation dimension, the symmetrical split performs best. It shows the highest 75th percentile of 274 compared to 246 for the reference network. Only red[-0.5, 0.1] has a higher 95th percentile whisker of 400 compared to 392 for the symmetrical split. The Lyapunov exponents shown

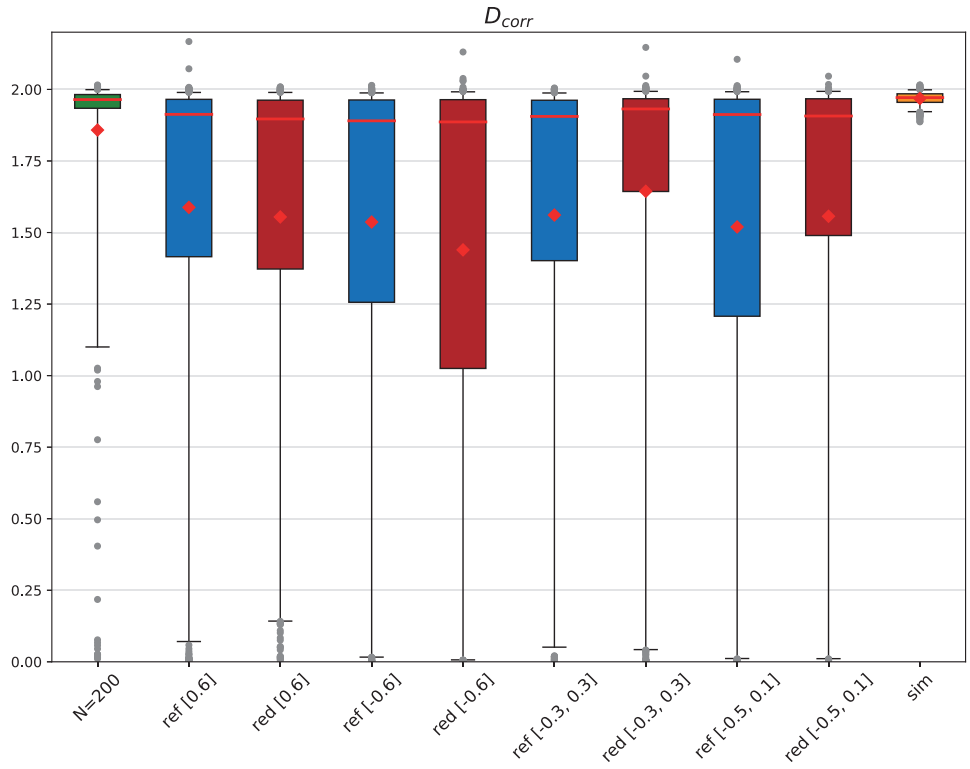


Figure 5.22: Correlation dimension 60 percent all splits
72 removed trajectories. See Figure 5.16 for detailed description.

in Figure D.4 also exhibit a good performance of the more symmetrical reduced networks.

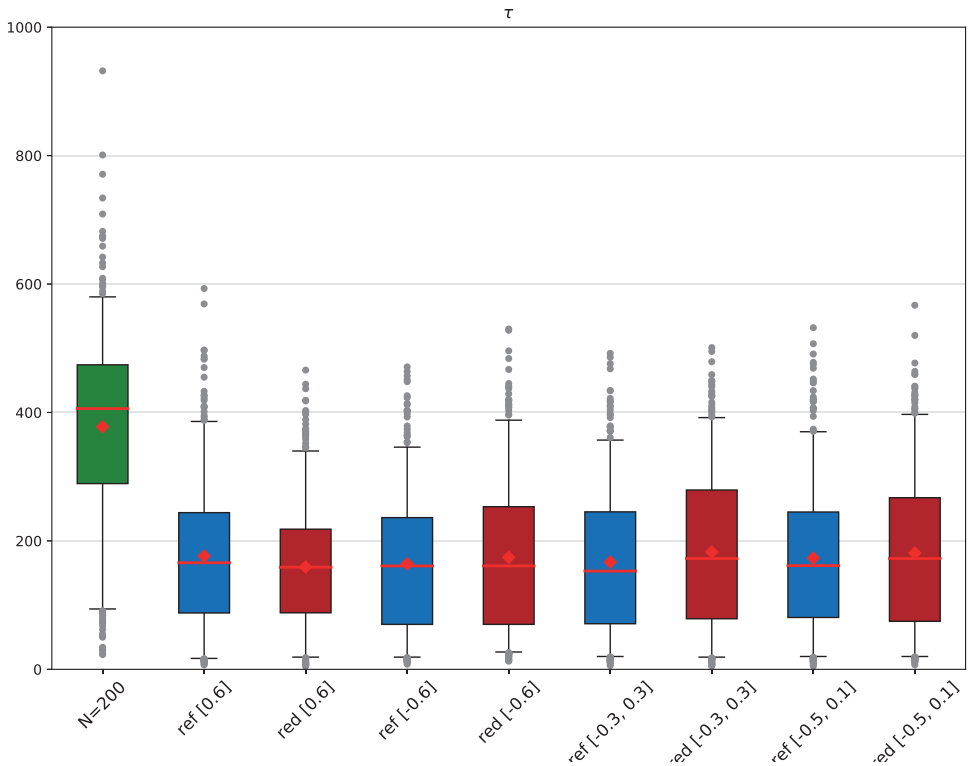


Figure 5.23: Demerge time 60 percent all splits
68 removed trajectories. See Figure 5.17 for detailed description.

5.3.4 Node Removal Conclusion

We have seen in Figure 5.16 Correlation dimension 10% and Figure 5.17 Demerge time 10% that removing the nodes with the largest W_{out} values gives the best results in terms of short and long-term prediction accuracy. Moving to smaller networks with $N = 140$ the best correlation dimensions are achieved by networks, where the top 30 % of nodes were removed. While short-term predictions, measured by demerge time, are almost identical Figure 5.19. The bottom reduced networks perform best. For even smaller networks the red[-0.1, 0.3] and red[0.4] split shows the highest correlation dimensions, and therefore the best structural attractor reconstruction. But red[-0.3, 0.1] has the highest demerge times, with a median of 291 time steps. red[-0.1, 0.3] and red[0.4] only have median demerge times of 274 and 242 time steps, respectively. When removing 60 % of nodes the picture becomes congruent again, when both measures (Figure 5.22 and Figure 5.23) indicate the symmetrical split red[-0.3, 0.3] as best predicting. Its demerge time 75th percentile of 274 outperforms its reference network by 28 time steps. The 25th percentile of the correlation dimension is 1.593, 0.310 better than the reference network. This indicates a convincing short-term prediction capability, as well as a superior structural attractor reconstruction, compared to same size networks.

5.4 Structural Analysis of Reservoir Computing Building Blocks

The node removal procedure has a considerable impact on the prediction quality, as was shown in subsection 5.3.3. Through the removal of the *correct* nodes we could achieve a significantly better network size to prediction quality ratio, than with any of the randomly initialized networks. This leaves the open question of what the reason behind these improvements is. In order to find a structural feature that affects prediction quality we compare reduced and reference reservoirs with each other.

5.4.1 Network and W_{in}

Removing 40 percent As the results for removing 40 % of the nodes still show reasonable results, while shrinking the reservoir considerably, we will focus our analysis on the best performing split, namely $[-0.1, 0.3]$ in Figure 5.21.

At first take the network into consideration. Figure 5.24 shows histograms of non-zero matrix entries of all 498 networks of the reduced [red] and the reference [blue] networks, respectively. Even if the spectral radii are the same in the two sets the reduced networks exhibit a narrower distribution of values. The average degree in each set was 2.533 for reference and 2.540 for reduced networks. A difference that hardly accounts for the long-term prediction accuracy differences, especially when looking at the degree independence shown in section 5.2.

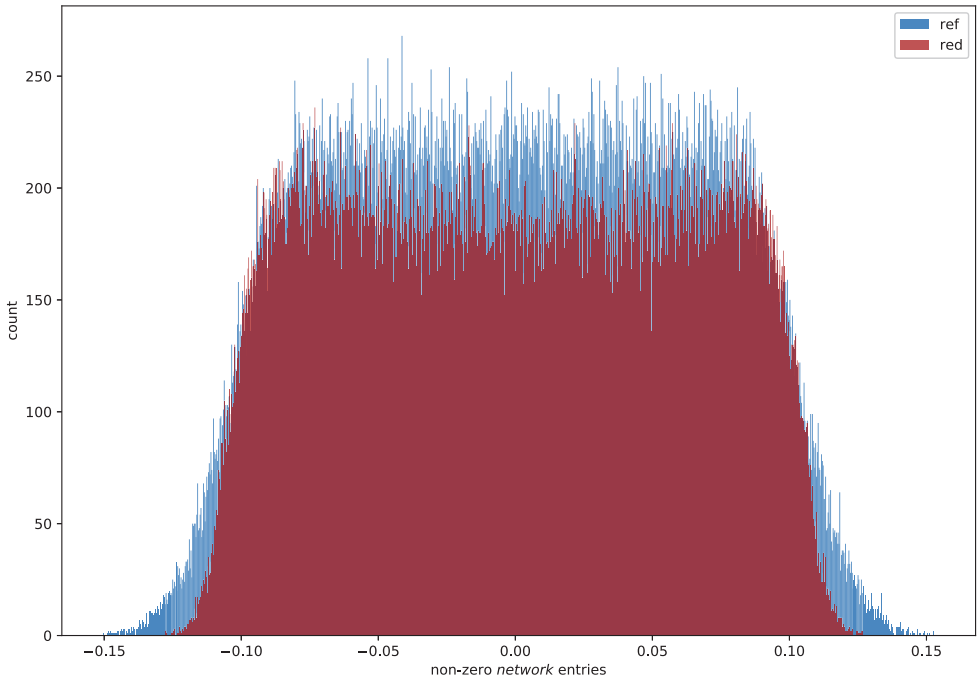


Figure 5.24: Histogram of non-zero network entries
Non-zero network entries of 498 realizations from the $[-0.1, 0.3]$ split. Reference networks are shown in blue and reduced networks in red.

A more drastic difference is visible when we look at the distribution of W_{in} . In Figure 5.25 Histogram of non-zero W_{in} entries one sees the distribution of non-zero values of W_{in} for all 498 realizations. Again, the reduced realizations are shown in red, while the reference reservoirs are in blue. While the reference W_{in} show a uniform distribution, as designed, the reduced W_{in} are peaked around ± 0.12, which performs even better than the uniform distribution in the range $[-0.17, 0.17]$.

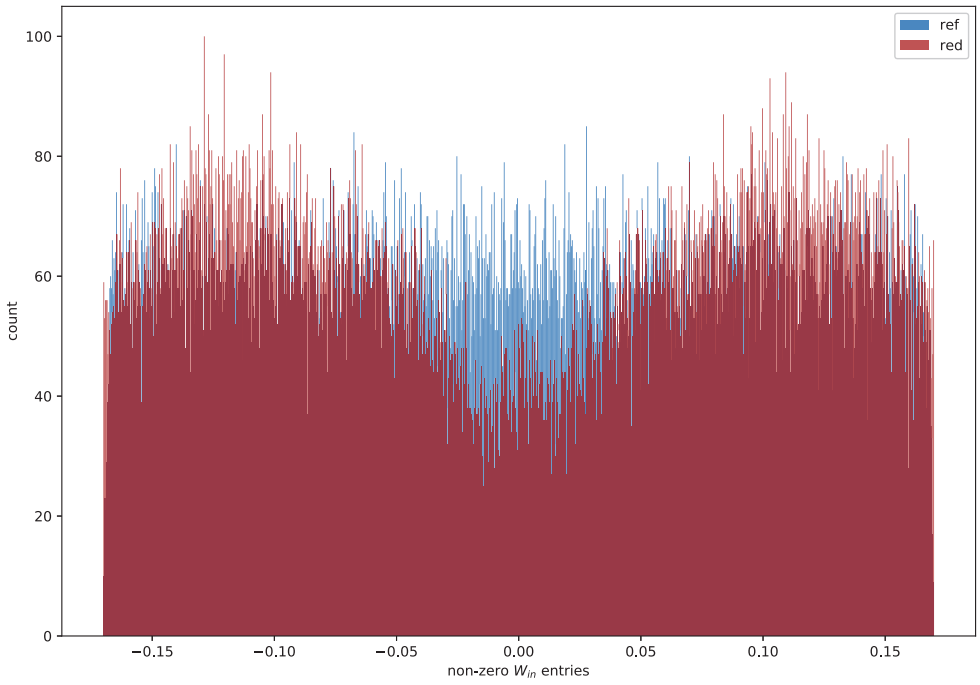


Figure 5.25: Histogram of non-zero W_{in} entries
Non-zero W_{in} entries of 498 realizations from the $[-0.1, 0.3]$ split. Reference networks are shown in blue and reduced networks in red.

5.4.2 Analysis of Closed Loop Setup

In the following we will not only take the network or W_{in} into consideration, but the reservoir as a whole [41]. After training we receive

$$r_{t+1} = tanh\left(\underbrace{[W_{in}W_{out} + W]}_{W_{all}} r_t\right) \tag{5.1}$$

as recursive equation for updating the echo states. While the *tanh* function contracts each component to $[-1, 1]$ in a non-linear way and thereby rotates the echo state vector, the argument consists of a simple linear map W_{all} . As a full time step contains only these two substeps, their interplay contains

the full learned dynamic. Obviously the activation function introduces a strong anisotropy[12], due to its contraction into the hyper cube. To prevent the nonlinear rotation one could scale the echo state back to the hyper sphere in each time step as nonlinearity as done by Verzelli et al. [48], instead of the often chosen *tanh* activation function.

In the following we present two ideas to tell good from bad trained reservoirs by taking the perspective of a closed loop autonomous system.

Eigenvalue analysis of W_{all} Here we are using the dataset from the node removal experiment above. The reservoirs reduced by 40 % (split $[-0.1, 0.3]$) show enough spread in terms of Lyapunov exponent and correlation dimension to divide the reservoirs into two categories. We call them good and bad for simplicity. Good contains only predictions that are within the spread of results of the simulated trajectory[13] , while bad contains predictions that are more than one spread interval away in at least one of the measures.

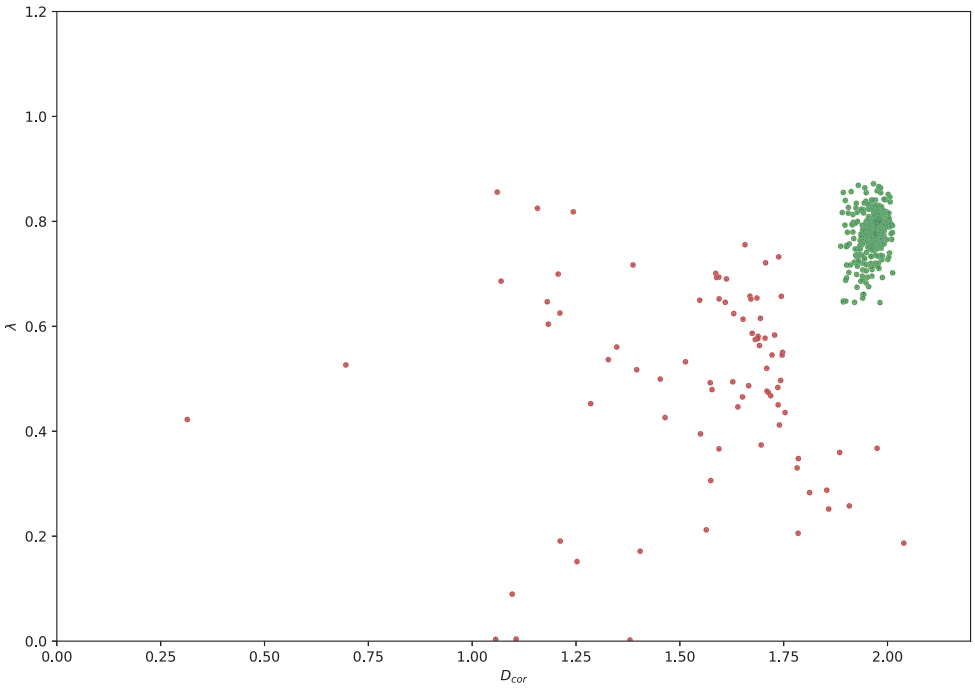


Figure 5.26: Scatter plot of $[-0.1, 0.3]$ split
Scatter plot of correlation dimension and Lyapunov exponent from the $[-0.1, 0.3]$ split experiment. Good realizations are within the spread of the measures of the simulation, bad ones one interval length further away.

Because the nonlinear activation function contracts the argument vector into a hypercube of edge length 2, a expansive component is necessary for a ongoing dynamic. Therefore we calculate the

[12]It should be kept in mind, that the anisotropy results in a strong basis dependence.
[13]See Figure 5.21 Demerge time 40% and Figure D.3 Lyapunov Exponent 40%

eigenvalues with the largest real parts of the linear substep of a time step, W_{all} , and compare them between the good and bad realizations. We begin with the largest eigenvalue of each realization.

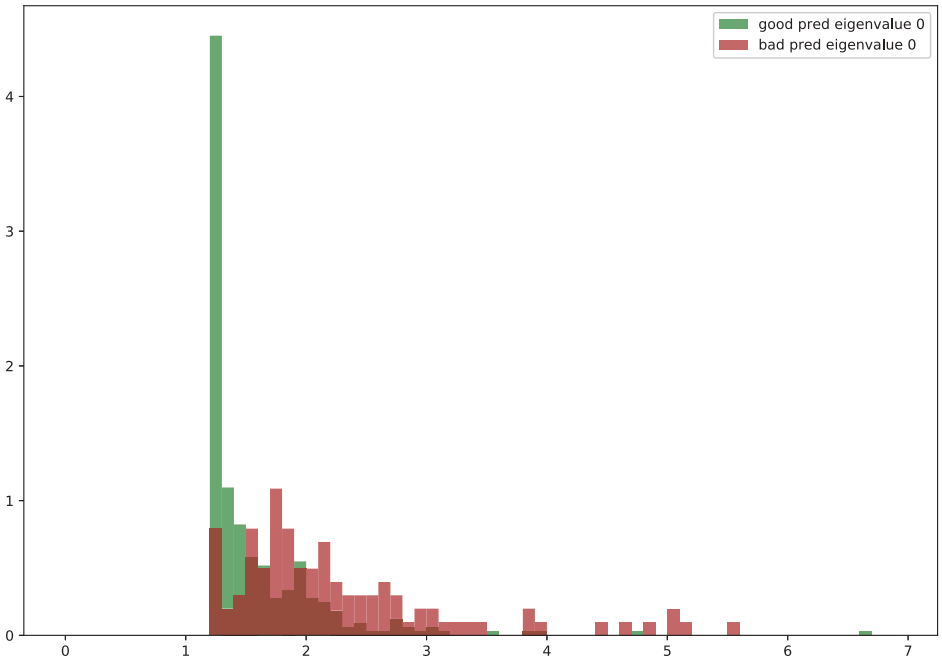


Figure 5.27: Largest eigenvalues of W_{all} in $[-0.1, 0.3]$ split
Normalized histograms of the eigenvalues (largest real part) of W_{all} , compared between good [green] and bad [red] predictions.

First, one has to note that as expected all largest eigenvalues are greater than 1 (expansive). Even if the distributions are not clearly separated from each other one can identify the smaller eigenvalues below 1.5 as the ones performing better compared to larger ones. The largest eigenvalue of the dataset 6.643 seems to be an exception, as it is associated with a good prediction. In the next histogram in Figure 5.28 we take a look at the distributions of second largest eigenvalues. Some of the good performing reservoirs exhibit second largest eigenvalues less than 1. The range of eigenvalues of good and bad predictions overlap, such that it appears unsuited as criterion. For the third largest eigenvalue the distributions separate partly again, see Figure 5.29. Below 0.909 (smallest bad eigenvalue) one finds only good realizations. Smaller eigenvalues tend to be less than 1 and the distributions do not separate anymore.
According to the observations made, it seems possible to predict the reservoirs performance from parts of the eigenvalue spectrum. Only a few of the 120 eigenvalues are greater than one, so we might look at the projections of r with the corresponding eigenvectors.

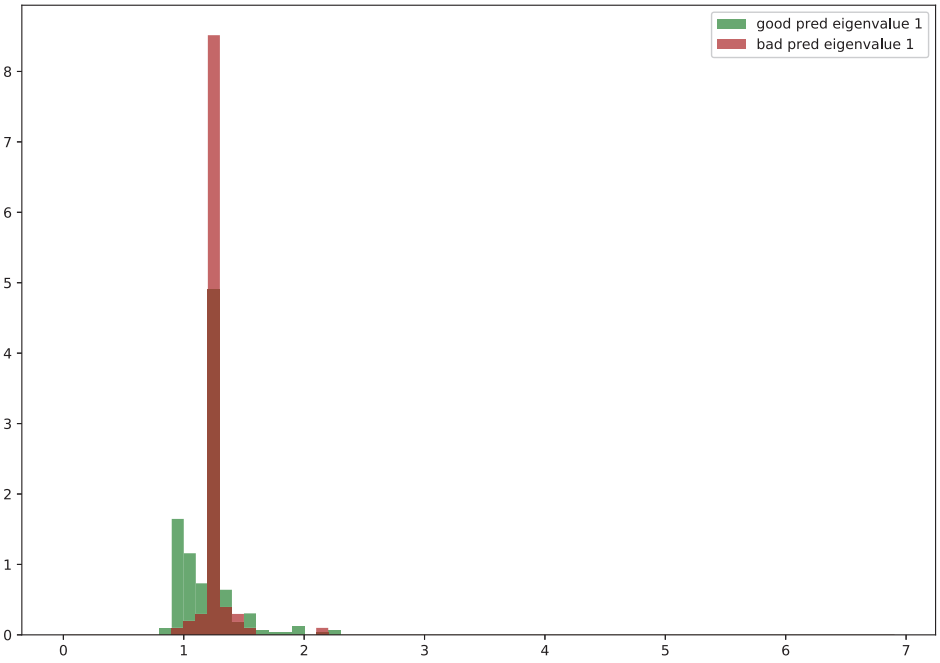


Figure 5.28: Second largest eigenvalues of W_{all} in $[-0.1, 0.3]$ split
Normalized histograms of the second largest real part of the eigenvalues of W_{all} , again compared between good [green] and bad [red] predictions.

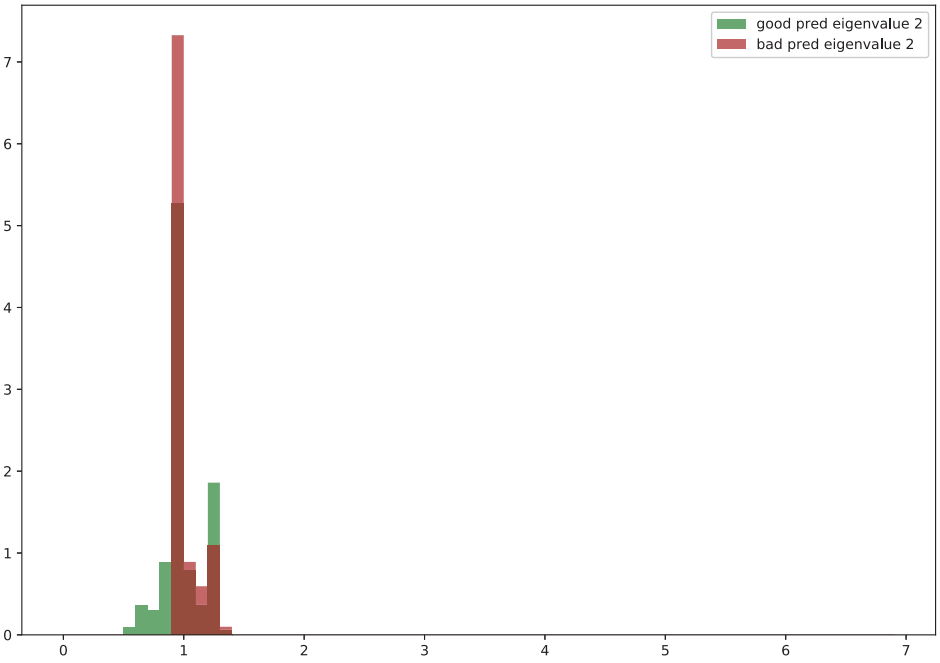


Figure 5.29: Third largest eigenvalues of W_{all} in $[-0.1, 0.3]$ split
Normalized histograms of the third largest real part of the eigenvalues of W_{all} , again compared between good [green] and bad [red] predictions.

Echo state rotation and error In the next step, we split the echo state update equation Equation 2.4 into a linear and a nonlinear part and investigate how they alter the echo state, separately. This makes it necessary to take a single realization perspective, instead of a statistical. While the change in magnitude during the two substeps did not show any noticeable features, the change in echo state vector $\vec{r}(t)$ angle displays a correlation with the error at that time step.

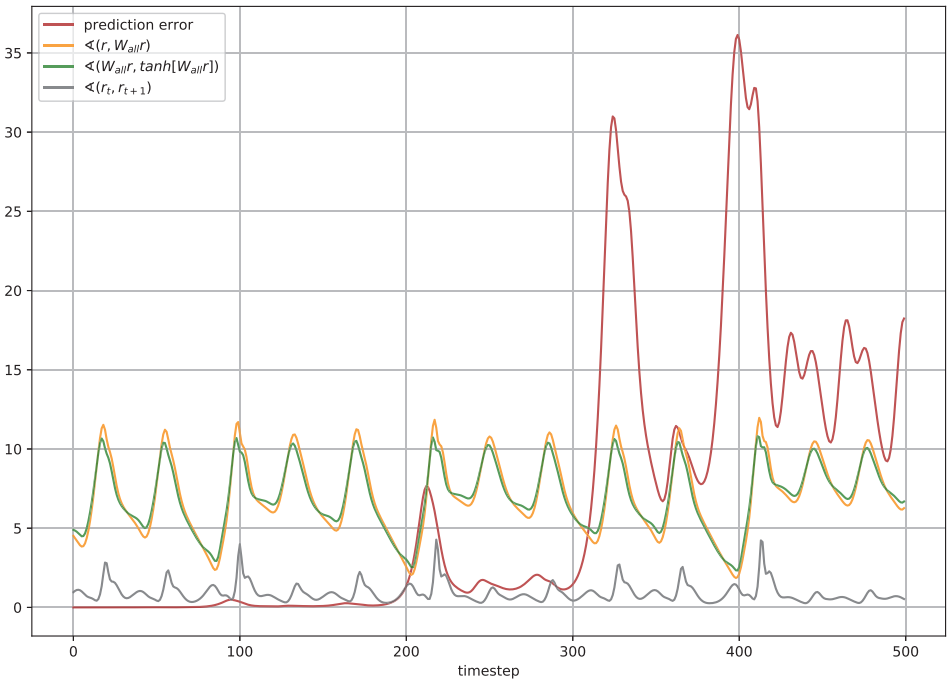


Figure 5.30: Echo state update rotation and error
The prediction error is plotted in red for each time step, orange shows the angle difference between the echo state vector before and after the linear part of a time step. The angle difference caused by the nonlinear transformation is plotted in green. The change in angle caused by a full time step is shown in gray.

In Figure 5.30 there are two peaks in the error of increasing intensity until the prediction deviates from the test trajectory at 300 time steps Each peak at 95 and 215 steps is preceded by an anomaly in both, the rotation by the linear part from r to $W_{all}r$ and the nonlinear part form $W_{all}r$ to $tanh[W_{all}r]$. The same kind of anomaly correlating with a peak in error can be observed at 400 steps, where the error is already high.

The same correlation between change in angle dynamics and significant increase in error can be observed in Figure 5.31 at 200 steps. In this case the error grew already from the beginning of the prediction. It has to be noted, that there are examples not showing this kind of connection between rotation angle and error. But since the rotation caused by the temporal substeps is a measure that can be calculated while predicting without further knowledge, it might still be a valuable precursor of failing predictions in some cases. A deeper analysis of when this observed relation holds and how it

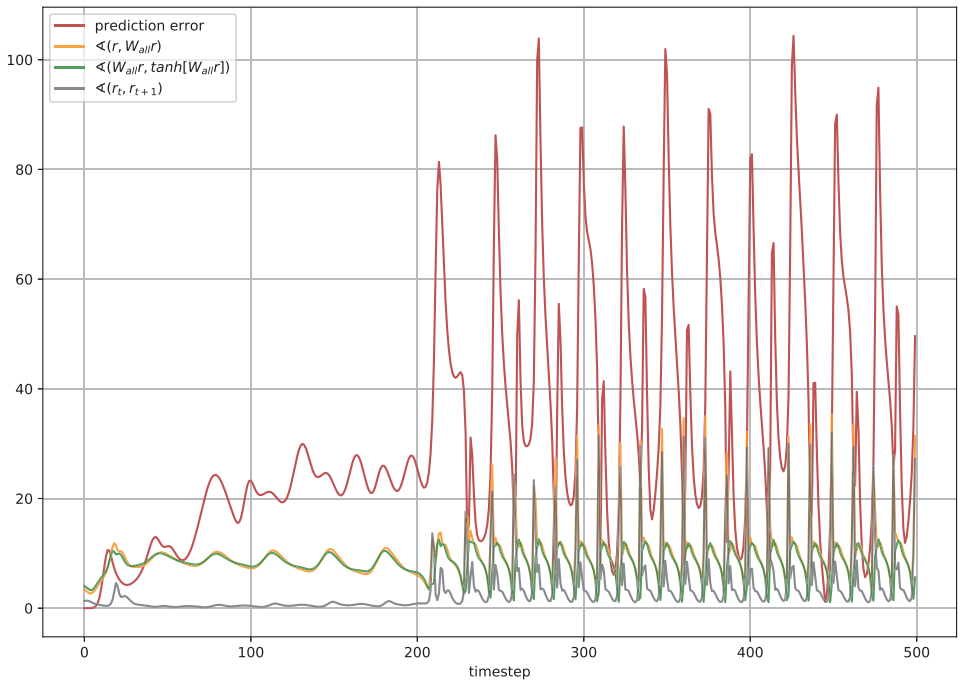


Figure 5.31: Echo state update rotation and error
See Figure 5.30 for explanation of the colors.

could be exploited is beyond the scope of this thesis.

Chapter 6

Conclusion and Outlook

The purpose of the thesis was to find modifications to reservoir computing in order to reduce the spread in prediction accuracy, measured by correlation dimension, Lyapunov exponent and demerge time. This has been achieved by removing nodes according to their learned output weight.

The main results of the conducted experiments are given in the following. We have seen that network size and topology are crucial for attractor reconstruction accuracy. Small world and random networks with $N = 300$ exhibit little spread in correlation dimension. Scale free networks are less suited for reservoir computing, as reported earlier [17]. This might be due to high degree nodes, obtaining lower weights during training, see Figure 5.8. The small world property, adjusted by rewiring coefficient p_{sw}, has little impact on long- and short-term prediction quality. Densities between 2 and 4 % facilitate structural attractor reconstruction, while short-term accuracy slightly improves with higher network densities.

We were able to improve long-term prediction by removing 10% of nodes that obtained large magnitude weights. The short-term prediction is affected only marginally, even if network size plays an important role there. Networks further reduced about 30% of nodes outperform same size random networks in terms of replicating the attractor, dynamically and spatially. The prediction accuracy decreases with decreasing network size. For even smaller networks, such as $N = 80$, removing highest and lowest weighted nodes evenly yields the best predictions comparing short and long-term prediction measures. We conclude, that some of the nodes in the network impede the dynamics leading to accurate predictions. Therefore, node removal can be considered a proper method for yielding more efficient networks.

In the last part, we analyzed the linear part of the closed loop setup. Its largest eigenvalues and its rotation angle in updating echo state vectors appears to be linked to success in predicting the future states of the dynamical system at hand. Too large first and second eigenvalues seem to increase the likelihood of failing prediction according to Lyapunov exponent and correlation dimension. In some cases, the change in angle correlates strongly with the prediction error.

Finally, promising fields of further research are outlined. Analyzing static and dynamical properties of the closed loop setup seems to be a promising line research, as it makes knowledge of the true future of the system unnecessary. The reason why slightly reduced networks, using the proposed method, improves predictive power, has the be investigated further. Therefore, it might be helpful to get a deeper understanding of the correspondence between the two dynamical systems, namely the closed loop reservoir computer and the target system. This could lead to a deeper understanding of both, reservoir computing and chaotic systems in general.

The proposed method of node removal shows potential to increase prediction accuracy also for other recent developments, such as feeding true states in order to extend accurate prediction time in low dimensional systems [10] or using reservoir computing extreme event prediction [38]. The

increased efficiency of our reduced networks can also be beneficial for physical reservoir implementations [16, 39, 46].

Appendix A

List of Reservoir Computing Objects and Notation

- W network, see section 2.3 Network
- W_{out} , see section 2.3 Training
- W_{in} , see section 2.3 W_{in}
- $W_{all} = W_{in} \cdot W_{out} + W$, see section 5.4 Structural Analysis of Reservoir Computing Building Blocks
- $\vec{x}(t)$ input time series, temporal dependency and vector notation may be omitted in the future
- $\vec{y}_{target}(t)$ target time series
- $\vec{y}(t)$ predicted time series
- $\vec{r}(t)$ echo state vector
- spectral radius r, see section 2.3 Spectral Radius
- size of network N, see section 2.3 Network
- node density ρ
- small world rewiring parameter p_{sw}
- degree of a node d
- strength of a node s
- regression regularization parameter β, see Equation 2.3 Regularized linear regression - Training II
- demerge time τ, see section 4.1 Demerge Time
- correlation dimension D_{cor}, see section 4.3 Correlation Dimension D_{cor}
- lyapunov exponent λ, see section 4.2 Lyapunov Exponent λ

Appendix B

Rosenstein Kantz Algorithm

The lyapunov exponent quantifies the divergence of nearby points over time. The distance of the jth pair of neighbors diverges roughly as

$$d_j(i) = C_j e^{\lambda_{max}(i \cdot \Delta t)}, \tag{B.1}$$

where C_j is the initial distance. Taking the logarithm yields similar slopes but different offsets for different pairs j. By determining the slope of the average over j we get a precise value for the largest Lyapunov Exponent [42]:

$$y(i) = \frac{1}{\Delta t} \langle \log [d_j(i)] \rangle \tag{B.2}$$

The slope is approximated at two points, τ_{min} and τ_{max}. In determining suitable next neighbor pairs one only keeps pairs where the future in τ_{max} time steps is part of the trajectory and neglects all the pairs that are not separated enough temporally [1] . The values for τ_{min} and τ_{max} are set to 25 and 190 by visual inspection. The former corresponds to the average time scale that the separation vector needs to turn into the direction of the largest Lyapunov Exponent. The latter is the number of steps at which the separation reaches a plateau due to the finite size of the attractor.

[1] Since temporal neighbors stay close as time evolves, we introduce minimal temporal separation threshold of 10 time steps. The result without threshold is almost identical, as seen when comparing Figure B.1 Lyapunov Exponents for different Trajectory Lengths without threshold to Figure 4.2 Lyapunov exponents for different trajectory lengths

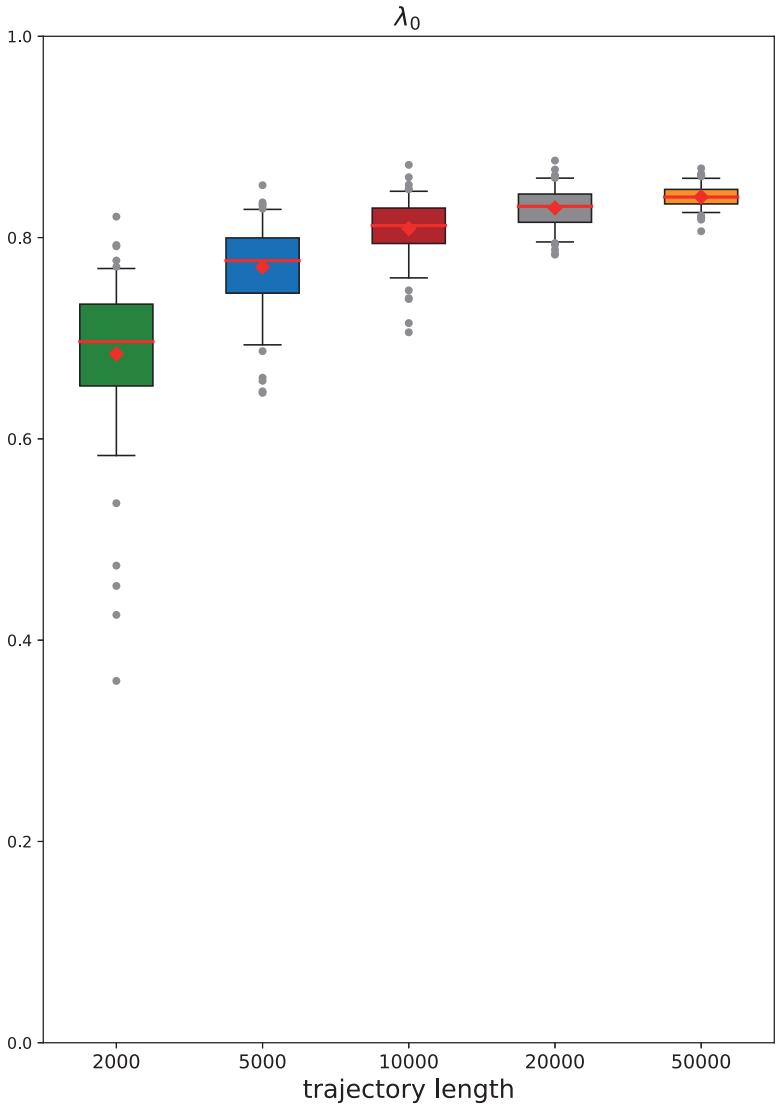


Figure B.1: Lyapunov Exponents for different Trajectory Lengths without threshold
Each box shows the distribution of Lyapunov Exponents for 100 trajectories. The boxes correspont to different trajectory lengths. The temporal threshold for identifying next neighbors is 0 time steps.

Appendix C

Further Gridsearch Results

To complement the Figure 5.1 Gridsearch results for random, $N = 100$ networks the according plots for different network sizes and topologies are shown. Even if the prediction for scale free, $N = 100$ networks shown in Figure C.1 is better for $W_{in} = 0.3$, the same overall behaviour of better predictions for smaller values of r and W_{in} scale is visible.

C.1 $N = 100$

As the small network size of only 100 nodes shows a larger spread the differences become clear.

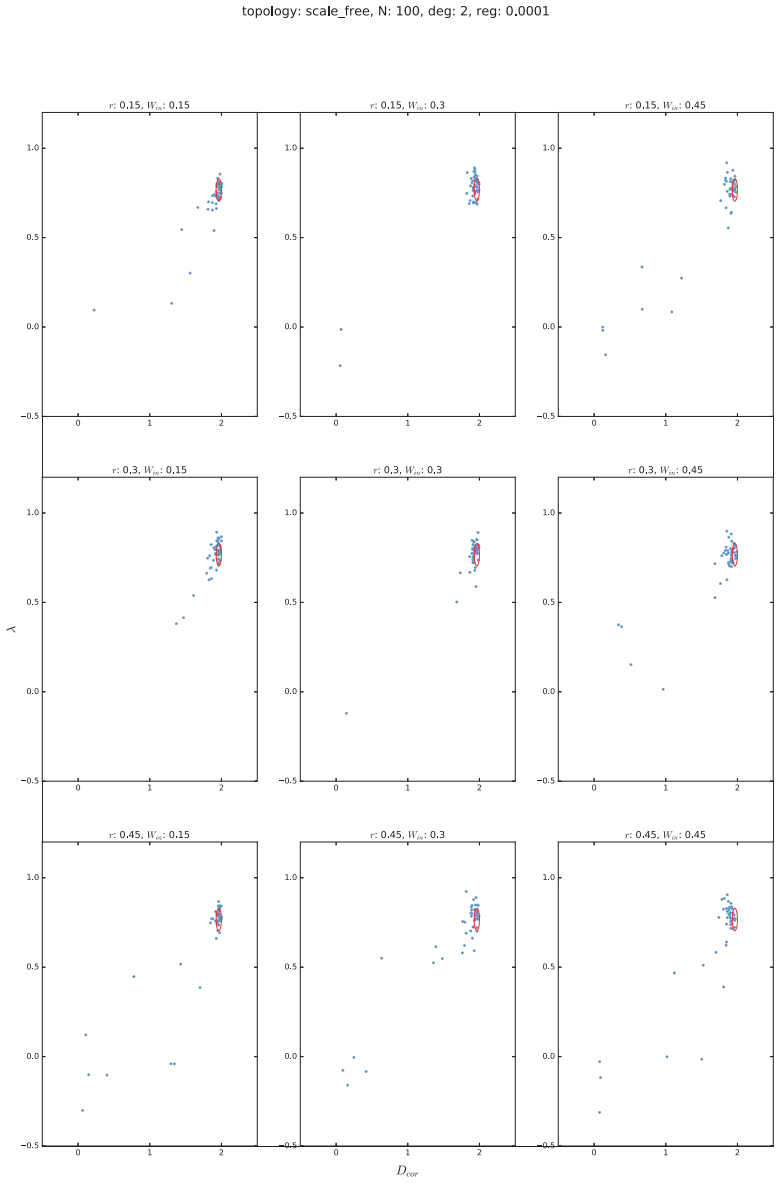


Figure C.1: Gridsearch results for scale free, $N = 100$ networks
Each plot shows the correlation dimension versus the lyapunov exponent for varying spectral radius r and W_{in} scale. The network topology is scale free, the size $N = 100$ with a density of 0.02. The red ellipse marks the 5th and 95th percentile of the respective measure for the simulated trajectories.

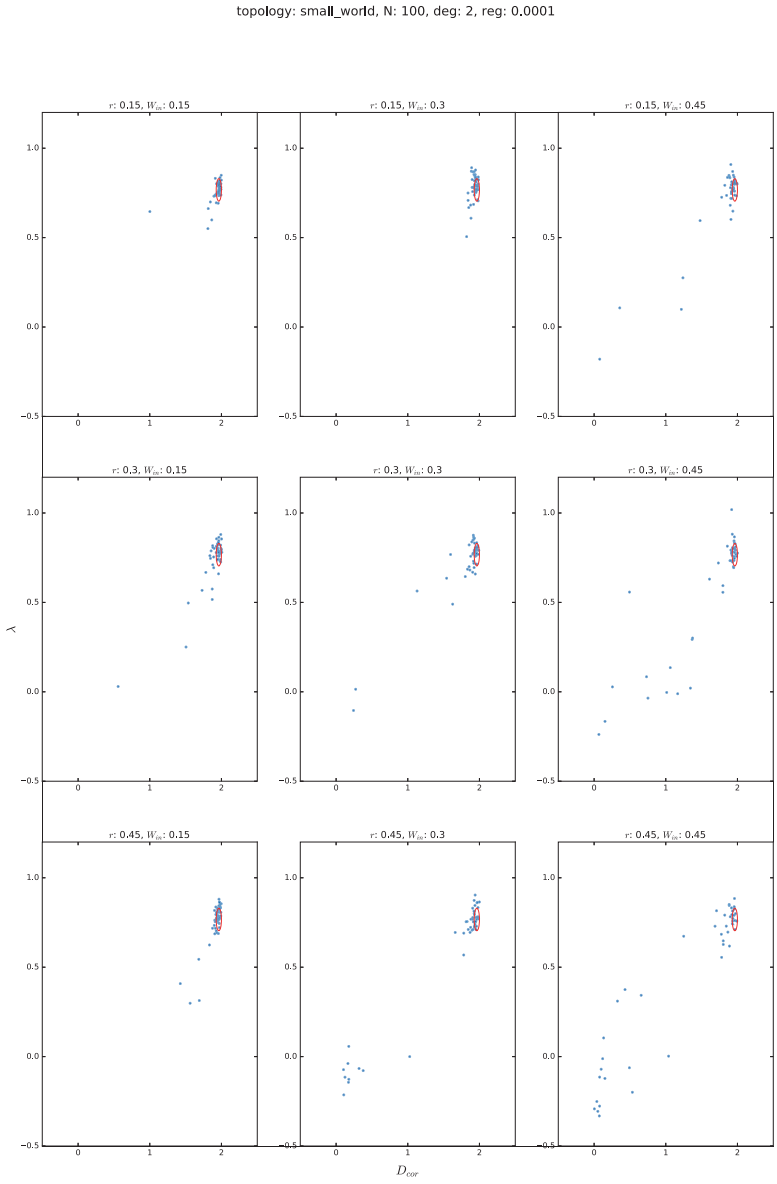


Figure C.2: Gridsearch results for small world, $N = 100$ networks
Each plot shows the correlation dimension versus the lyapunov exponent for varying spectral radius r and W_{in} scale. The network topology is small world, the size $N = 100$ with a density of 0.02. The red ellipse marks the 5th and 95th percentile of the respective measure for the simulated trajectories.

C.2 $N = 500$

For larger networks the sensitivity to the spectral radius and W_{in} scale seem to be reduced, as the results mostly agree with the ones of the simulated trajectories. The topology dependence is invisible in this representation for networks of this size.

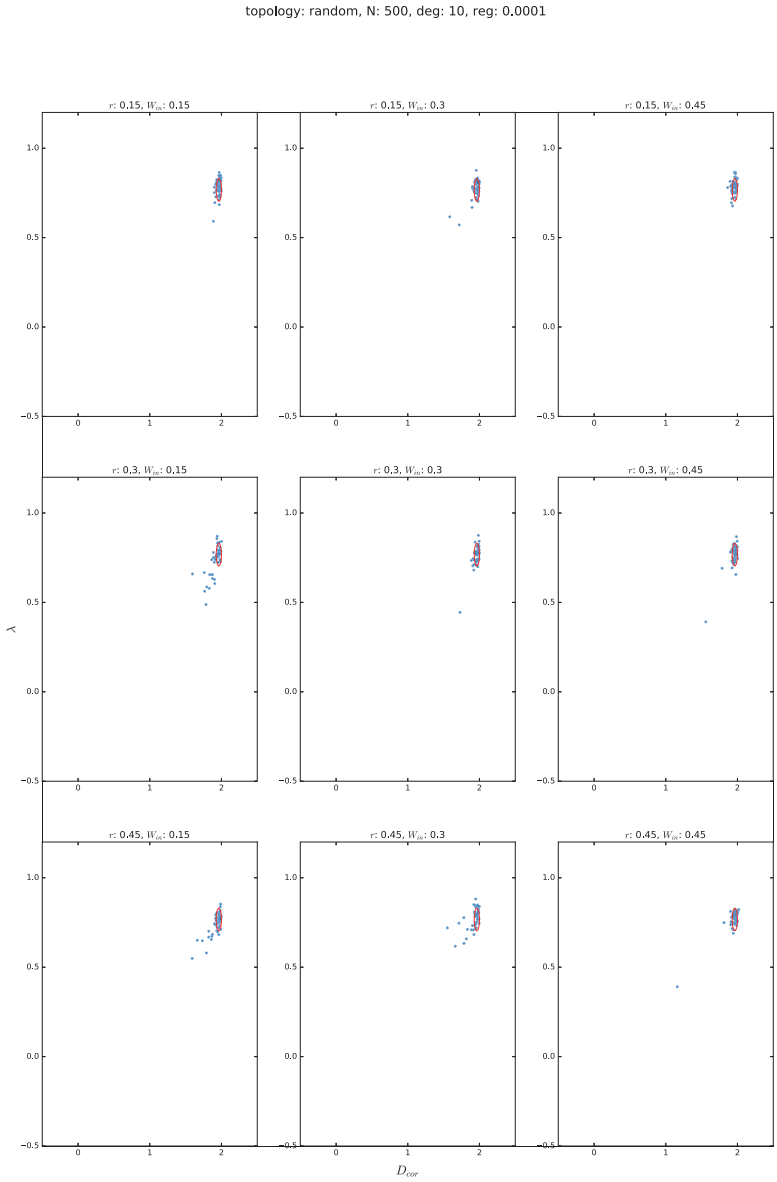


Figure C.3: Gridsearch results for random, $N = 500$ networks
Each plot shows the correlation dimension versus the lyapunov exponent for varying spectral radius r and W_{in} scale. The network topology is random, the size $N = 500$ with a density of 0.02. The red ellipse marks the 5th and 95th percentile of the respective measure for the simulated trajectories.

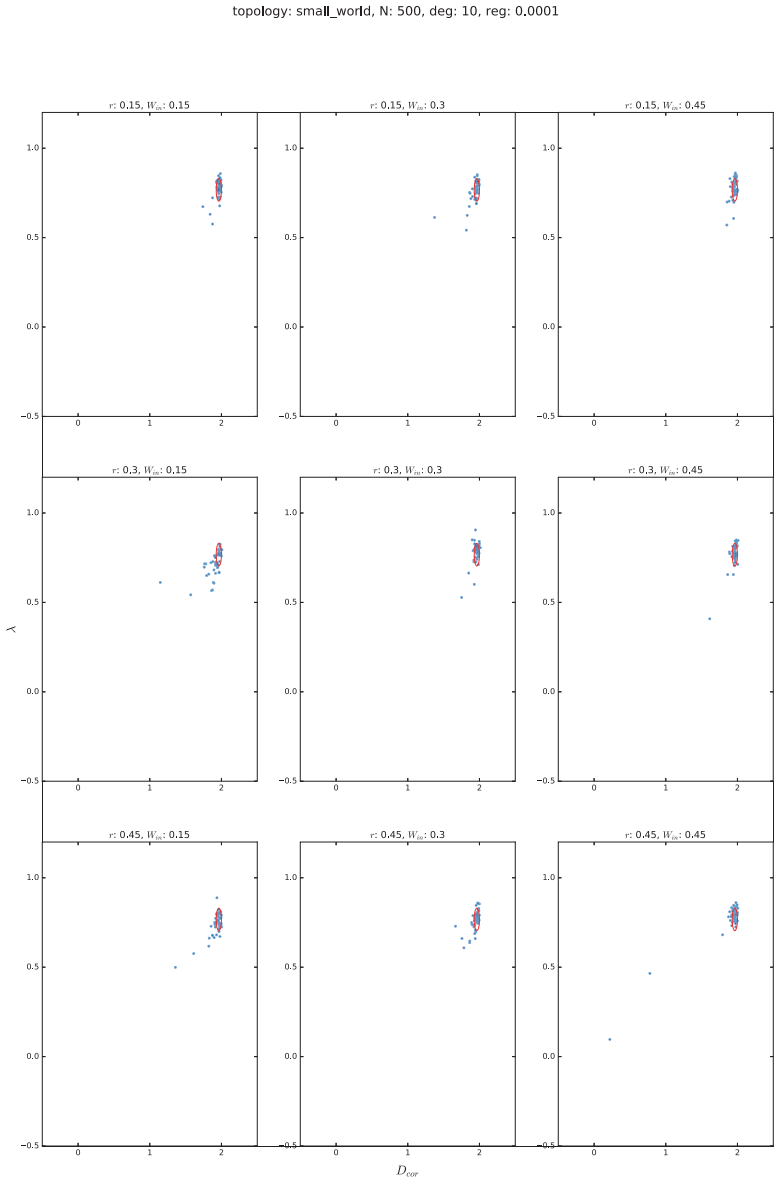


Figure C.4: Gridsearch results for small world, $N = 500$ networks
Each plot shows the correlation dimension versus the lyapunov exponent for varying spectral radius r and W_{in} scale. The network topology is small world, the size $N = 500$ with a density of 0.02. The red ellipse marks the 5th and 95th percentile of the respective measure for the simulated trajectories.

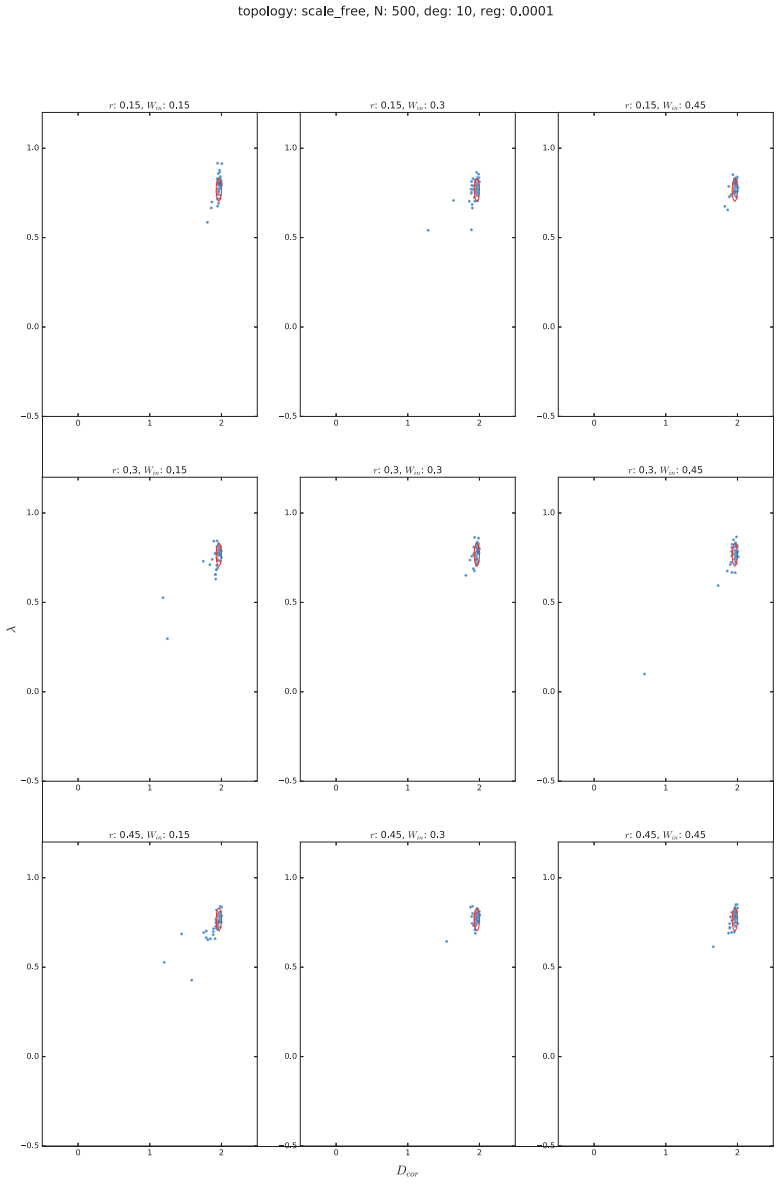


Figure C.5: Gridsearch results for scale free, $N = 500$ networks
Each plot shows the correlation dimension versus the lyapunov exponent for varying spectral radius r and W_{in} scale. The network topology is scale free, the size $N = 500$ with a density of 0.02. The red ellipse marks the 5th and 95th percentile of the respective measure for the simulated trajectories.

Appendix D

Additional Results of Node Removal

D.1 Lyapunov Exponents 10 Percent

The boxplot and label explaination can be found in Figure 5.16 Correlation dimension 10%. The behavior is similar to the one shown for demerge time in Figure 5.17

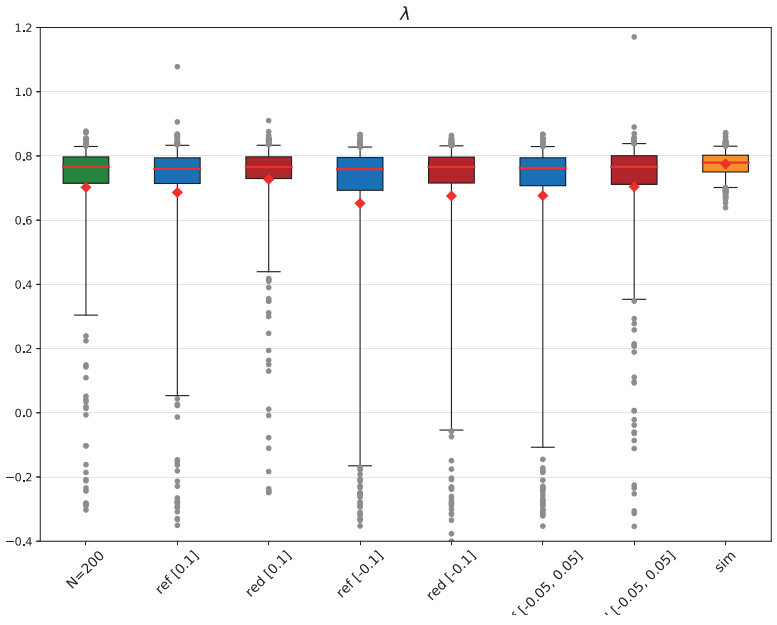


Figure D.1: Lyapunov Exponent 10
2 removed trajectories.

D.2 Lyapunov Exponents 30 Percent

Again the situation is analog to correlation dimension shown in Figure 5.18

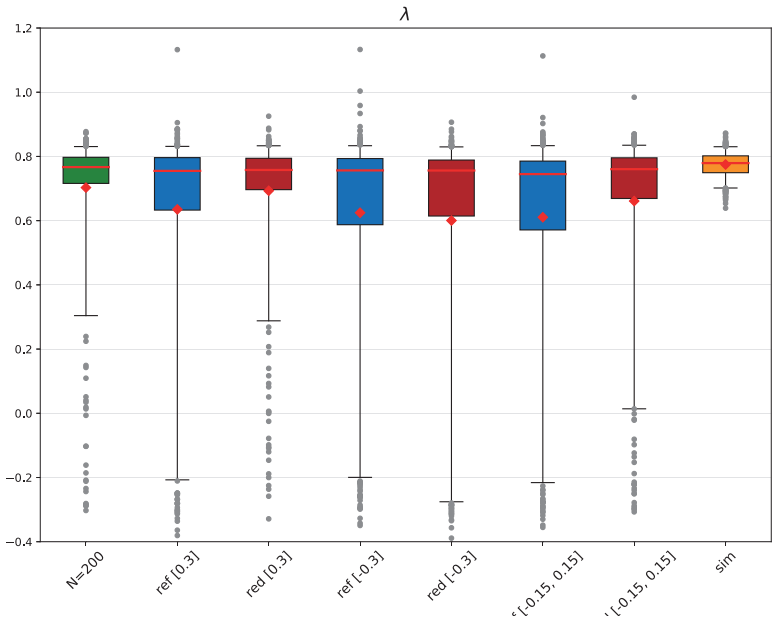


Figure D.2: Lyapunov Exponent 30
3 removed trajectories.

D.3 Lyapunov Exponents 40 Percent

Figure D.3 Lyapunov Exponent 40% reproduces the overall behavior of Figure 5.21 Demerge time 40%, showing the best performance for [0.4] and [-0.1, 0.3] compared to other splits and their respective reference reservoirs and the worst performance for [-0.4]. In the lyapunov exponent case the good performing splits show approximately twice the spread of the large reservoirs, when comparing the 25th to 75th percentile boxes.

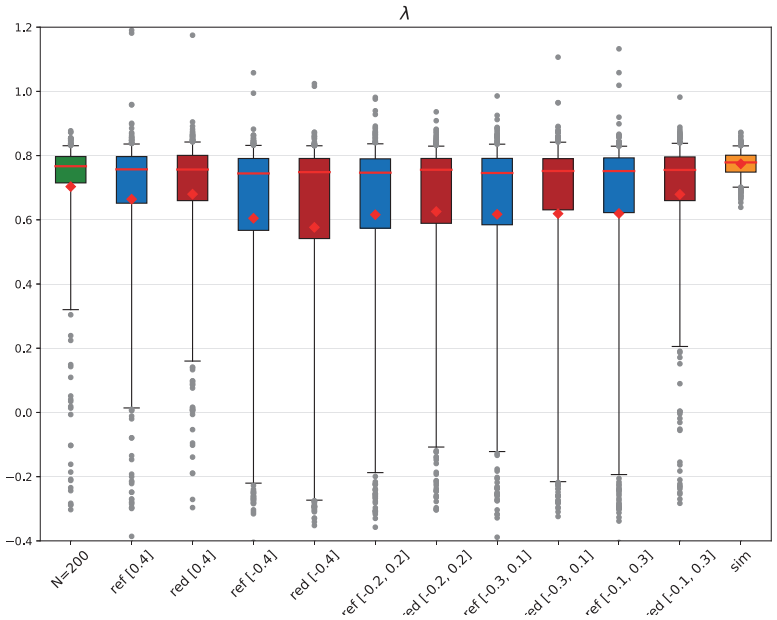


Figure D.3: Lyapunov Exponent 40%
The splits are labeled as follows: top [0.4], bottom [-0.4], symmetric [-0.2, 0.2], bottom 30% and top 10% [-0.3, 0.1] and vice versa 10% bottom with 30% top [-0.1, 0.3]. Removed 12 trajectories.

D.4 Lyapunov Exponents 60 Percent

As for demerge time and correlation dimension, the best splits are around the symmetrical, outperforming their reference networks by far.

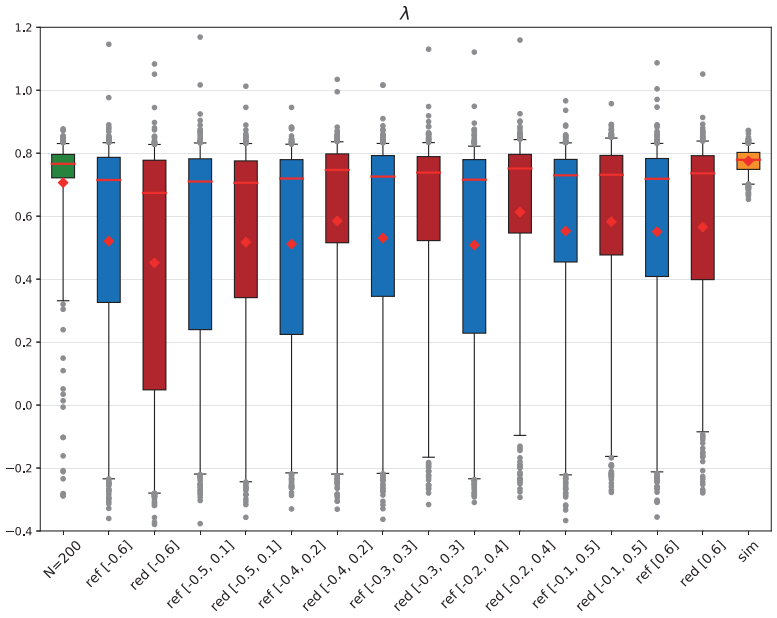


Figure D.4: Lyapunov Exponent 60%
Removed 114 trajectories.

Bibliography

[1] Henry D. I. Abarbanel, Reggie Brown, John J. Sidorowich, and Lev Sh. Tsimring. The analysis of observed chaotic data in physical systems. *Rev. Mod. Phys.*, 65:1331–1392, Oct 1993.

[2] Réka Albert, Hawoong Jeong, and Albert-László Barabási. Error and attack tolerance of complex networks. *Nature*, 406(6794):378–382, Jul 2000.

[3] L. A. N. Amaral, A. Scala, M. Barthélémy, and H. E. Stanley. Classes of small-world networks. *Proceedings of the National Academy of Sciences*, 97(21):11149–11152, 2000.

[4] Albert-László Barabási and Bonabeau Eric. Scale-free networks. *Scientific American*, 2003.

[5] Albert-László Barabási and Albert Réka. Emergence of scaling in random networks. *Nature*, 1999.

[6] T. L. Carroll. Using reservoir computers to distinguish chaotic signals. *Phys. Rev. E*, 98:052209, Nov 2018.

[7] T. L. Carroll and L. M. Pecora. Network structure effects in reservoir computers. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 29(8):083130, 2019.

[8] Hongyan Cui, Xiang Liu, and Lixiang Li. The architecture of dynamic reservoir in the echo state network. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 22(3):033127, 2012.

[9] Kenji Doya. Bifurcations in the learning of recurrent neural networks. 1992.

[10] Huawei Fan, Junjie Jiang, Chun Zhang, Xingang Wang, and Ying-Cheng Lai. Long-term prediction of chaotic systems with machine learning. *Phys. Rev. Research*, 2:012080, Mar 2020.

[11] J. Doyne Farmer, Edward Ott, and James A. Yorke. The dimension of chaotic attractors.

[12] J. Doyne Farmer and John J. Sidorowich. Predicting chaotic time series. *Phys. Rev. Lett.*, 59:845–848, Aug 1987.

[13] Ken-ichi Funahashi and Yuichi Nakamura. Approximation of dynamical systems by continuous time recurrent neural networks. *Neural Networks*, 6(6):801 – 806, 1993.

[14] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[15] Peter Grassberger and Itamar Procaccia. Characterization of strange attractors. *Phys. Rev. Lett.*, 50:346–349, Jan 1983.

[16] Aaron Griffith, Andrew Pomerance, and Daniel J. Gauthier. Forecasting chaotic systems with very low connectivity reservoir computers. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 29(12):123108, 2019.

[17] Alexander Haluszczynski and Christoph 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*, 29(10):103143, 2019.

[18] Herbert Jaeger. The echo state approach to analysing and training recurrent neural networks. *GMD-Report 148, German National Research Institute for Computer Science*, 01 2001.

[19] Herbert Jaeger. Tutorial on training recurrent neural networks, covering bppt, rtrl, ekf and the echo state network approach. *GMD-Forschungszentrum Informationstechnik, 2002.*, 5, 01 2002.

[20] Herbert Jaeger. A tutorial on training recurrent neural networks, covering bppt, rtrl, ekf and the "echo state network" approach. 2013.

[21] Mehra Jagdish and Rechenberg Helmut. *The Completion of Quantum Mechanics 1926-1941 - Part 1: The Probability Interpretation and the Statistical Transformation Theory, the Physical Interpretation and the Empirical and Mathematical Foundations of Quantum Mechanics 1926-1932*. The historical development of quantum theory. 2000.

[22] Holger Kantz. A robust method to estimate the maximal lyapunov exponent of a time series. *Physics Letters A*, 185(1):77 – 87, 1994.

[23] Edward N. Lorenz. Deterministic nonperiodic flow. *Journal of the Atmospheric Sciences*, 20, 1963.

[24] Zhixin Lu, Brian R. Hunt, and Edward Ott. Attractor reconstruction by machine learning. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 28(6):061104, 2018.

[25] Zhixin Lu, Jaideep Pathak, Brian Hunt, Michelle Girvan, Roger Brockett, and Edward Ott. Reservoir observers: Model-free inference of unmeasured variables in chaotic systems. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 27(4):041102, 2017.

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

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

[28] Sieniek M. Godbole V. et al. McKinney, S.M. International evaluation of an ai system for breast cancer screening. *Nature*, 577, 2020.

[29] Jo Merchant. Powerful antibiotics discovered using ai, 2020.

[30] Aleksandra Mojsilovic. The age of data and opportunities, 2014.

[31] Thalaiyasingam Ajanthan Namhoon Lee and Philip H. S. Torr. Singe-shot network pruning based on connection sensitivity.

[32] V. I. Oseledec. A multiplicative ergodic theorem. characteristic ljapunov, exponents of dynamical systems.

[33] A. Rényi P. Erdös. On random graphs i. *Publ. Math. Debrecen*, 6:290–297, 1959.

[34] Jaideep Pathak, Zhixin Lu, Brian R. Hunt, Michelle Girvan, and Edward Ott. Using machine learning to replicate chaotic attractors and calculate lyapunov exponents from data. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 27(12):121102, 2017.

[35] Henri. Poincaré. *Science and Method*. Cosimo Classics, 2010.

[36] H. Poincaré. Mémoire sur les courbes définies par une équation différentielle (ii). *Journal de Mathématiques Pures et Appliquées*, 8:251–296, 1882.

[37] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C - The Art of Scientific Computing*. Cambridge University Press, 1992.

[38] Viktoras Pyragas and Kestutis Pyragas. Using reservoir computer to predict and prevent extreme events. *arXiv e-prints*, page arXiv:2002.09279, feb 2020.

[39] Mushegh Rafayelyan, Jonathan Dong, Yongqi Tan, Florent Krzakala, and Sylvain Gigan. Large-scale optical reservoir computing for spatiotemporal chaotic systems prediction.

[40] Sebastian Raschka and Vahid Mirjalili. *Python Machine Learning - Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow 2*. Packt, 2019.

[41] René Felix Reinhart and Jochen Jakob Steil. *Reservoir regularization stabilizes learning of Echo State Networks with output feedback*. PhD thesis, Technische Fakultät Universität Bielefeld, 2011.

[42] Michael T. Rosenstein, James J. Collins, and Carlo J. De Luca. A practical method for calculating largest lyapunov exponents from small data sets. *Physica D: Nonlinear Phenomena*, 1993.

[43] David Ruelle and Floris Takens. On the nature of turbulence.

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

[45] Steven H. Strogatz. *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry and Engineering*. Westview Press, 2000.

[46] Gouhei Tanaka, Toshiyuki Yamane, Jean Benoit Héroux, Ryosho Nakane, Naoki Kanazawa, Seiji Takeda, Hidetoshi Numata, Daiju Nakano, and Akira Hirose. Recent advances in physical reservoir computing: A review. *Neural Networks*, 115:100 – 123, 2019.

[47] James Theiler. Estimating fractal dimension. *J. Opt. Soc. Am. A*, 7(6):1055–1073, Jun 1990.

[48] Pietro Verzelli, Cesare Alippi, and Lorenzo Livi. Echo state networks with self-normalizing activations on the hyper-sphere. *Scientific Reports*, 2019.

[49] Pantelis R. Vlachas, Jaideep Pathak, Brian R. Hunt, Themistoklis P. Sapsis, Michelle Girvan, Edward Ott, and Petros Koumoutsakos. Backpropagation algorithms and reservoir computing in recurrent neural networks for the forecasting of complex spatiotemporal dynamics, 2019.

[50] Duncan J. Watts and Steven H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 1998.

[51] Matthew Shunshi Zhang and Bradly C. Stadle. One-shot pruning of recurrent neural networks by jacobian spectrum evaluation.

[52] Roland S. Zimmermann and Ulrich Parlitz. Observing spatio-temporal dynamics of excitable media using reservoir computing. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 28(4):043118, 2018.

Acknowledgement

I would like to thank Joschka, Youssef and Sebastian, as well as Daniel and Peter for uncounted answered questions and helpful discussions about physics and coding. Further, I would like to gratefully acknowledge the unwavering support my family gave me. Most importantly, special thanks to Hubertus and Christoph for enabling and supervising this thesis.

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, April 14, 2020,

Unterschrift: