Prediction and control of nonlinear dynamical systems using machine learning

Alexander Haluszczynski

München, 2021

Prediction and control of nonlinear dynamical systems using machine learning

Dissertation an der Fakultät für Physik
der
Ludwig-Maximilians-Universität München

vorgelegt von
Alexander Haluszczynski
aus München

München, den 29. Oktober 2021

Erstgutachter: PD Dr. habil. Christoph Räth
Zweitgutachter: Prof. Dr. Erwin Frey
Tag der mündlichen Prüfung: 14.12.2021

Contents

Zusammenfassung **VII**

Abstract **IX**

1 Introduction **1**

1.1 Prediction of nonlinear dynamical systems 1

1.2 Chaos and control of nonlinear dynamical systems 3

1.3 Aim of this thesis 7

2 Methods **9**

2.1 Reservoir Computing 9

2.2 Measuring the climate of a dynamical system 12

3 Prediction variability and topology **15**

3.1 Objectives 15

3.2 Methods 16

3.3 Results 19

3.4 Conclusions 22

4 Improving prediction stability **23**

4.1 Objectives 23

4.2 Methods 24

4.3 Results 25

4.4 Conclusions 32

5 Flexible control mechanism **33**

5.1 Objectives 33

5.2 Methods 34

5.3 Results 35

5.4 Conclusions 39

6 System Causalities and Outlook **41**

Publications **47**

A Good and bad predictions: Assessing and improving the replication of chaotic attractors by means of reservoir computing 48

B Reducing network size and improving prediction stability of reservoir computing 59

C Controlling nonlinear dynamical systems into arbitrary states using machine learning 71

Acknowledgement **81**

Bibliography **83**

Zusammenfassung

Künstliche Intelligenz und Machine Learning erfreuen sich in Folge der rapide gestiegenen Rechenleistung immer größerer Popularität. Sei es autonomes Fahren, Gesichtserkennung, bildgebende Diagnostik in der Medizin oder Robotik – die Anwendungsvielfalt scheint keine Grenzen zu kennen. Um jedoch systematischen Bias und irreführende Ergebnisse zu vermeiden, ist ein tiefes Verständnis der Methoden und ihrer Sensitivitäten von Nöten. Anhand der Vorhersage chaotischer Systeme mit Reservoir Computing – einem künstlichen rekurrenten neuronalem Netzwerk – wird im Rahmen dieser Dissertation beleuchtet, wie sich verschiedene Eigenschaften des Netzwerks auf die Vorhersagekraft und Robustheit auswirken. Es wird gezeigt, wie sich die Variabilität der Vorhersagen – sowohl was die exakte zukünftige Trajektorie betrifft als auch das statistische Langzeitverhalten (das "Klima") des Systems – mit geeigneter Parameterwahl signifikant reduzieren lässt. Die Nichtlinearität der Aktivierungsfunktion spielt hierbei eine besondere Rolle, weshalb ein Skalierungsparameter eingeführt wird, um diese zu kontrollieren. Des Weiteren werden differenzielle Eigenschaften des Netzwerkes untersucht und gezeigt, wie ein kontrolliertes Entfernen der "richtigen" Knoten im Netzwerk zu besseren Vorhersagen führt und die Größe des Netzwerkes stark reduziert werden kann bei gleichzeitig nur moderater Verschlechterung der Ergebnisse. Dies ist für Hardware Realisierungen von Reservoir Computing wie zum Beispiel Neuromorphic Computing relevant, wo möglichst kleine Netzwerke von Vorteil sind. Zusätzlich werden unterschiedliche Netzwerktopologien wie Small World Netzwerke und skalenfreie Netzwerke beleuchtet.

Mit den daraus gewonnenen Erkenntnissen für bessere Vorhersagen von nichtlinearen dynamischen Systemen wird eine neue Kontrollmethode entworfen, die es ermöglicht, dynamische Systeme flexibel in verschiedenste Zielzustände zu lenken. Hierfür wird – anders als bei vielen bisherigen Ansätzen – keine Kenntnis der zugrundeliegenden Gleichungen des Systems erfordert. Ebenso wird nur eine begrenzte Datenmenge verlangt, um Reservoir Computing hinreichend zu trainieren. Zudem ist es nicht nur möglich, chaotisches Verhalten in einen periodischen Zustand zu zwingen, sondern auch eine Kontrolle auf komplexere Zielzustände wie intermittentes Verhalten oder ein spezifischer anderer chaotischer Zustand. Dies ermöglicht eine Vielzahl neuer potenzieller realer Anwendungen, von personalisierten Herzschrittmachern bis hin zu Kontrollvorrichtungen für Raketentriebwerke zur Unterbindung von kritischen Verbrennungsinstabilitäten.

Als Schritt zur Weiterentwicklung von Reservoir Computing zu einem verbesserten hybriden System, das nicht nur rein datenbasiert arbeitet, sondern auch physikalische Zusammenhänge berücksichtigt, wird ein Ansatz vorgestellt, um lineare und nichtlinearen Kausalitätsstrukturen zu separieren. Dies kann verwendet werden, um Systemgleichungen oder Restriktionen für ein hybrides System zur Vorhersage oder Kontrolle abzuleiten.

Abstract

Artificial intelligence and machine learning are becoming increasingly popular as a result of the rapid increase in computing power. Be it autonomous driving, facial recognition, medical imaging diagnostics or robotics – the variety of applications seems to have no limits. However, to avoid systematic bias and misleading results, a deep understanding of the methods and their sensitivities is needed. Based on the prediction of chaotic systems with reservoir computing – an artificial recurrent neural network – this dissertation sheds light on how different properties of the network affect the predictive power and robustness. It is shown how the variability of the predictions – both in terms of the exact short-term predictions and the long-term statistical behaviour (the "climate") of the system – can be significantly reduced with appropriate parameter choices. The nonlinearity of the activation function plays a special role here, thus a scaling parameter is introduced to control it. Furthermore, differential properties of the network are investigated and it is shown how a controlled removal of the right nodes in the network leads to better predictions, whereas the size of the network can be greatly reduced while only moderately degrading the results. This is relevant for hardware realizations of reservoir computing such as neuromorphic computing, where networks that are as small as possible are advantageous. Additionally, different network topologies such as small world networks and scale-free networks are investigated.

With the insights gained for better predictions of nonlinear dynamical systems, a new control method is designed that allows dynamical systems to be flexibly forced into a wide variety of dynamical target states. For this – unlike many previous approaches – no knowledge of the underlying equations of the system is required. Further, only a limited amount of data is needed to sufficiently train reservoir computing. Moreover, it is possible not only to force chaotic behavior to a periodic state, but also to control for more complex target states such as intermittent behavior or a specific different chaotic state. This enables a variety of new potential real-world applications, from personalized cardiac pacemakers to control devices for rocket engines to suppress critical combustion instabilities.

As a step toward advancing reservoir computing to an improved hybrid system that

is not only purely data-based but also takes into account physical relationships, an approach is presented to separate linear and nonlinear causality structures. This can be used to derive system equations or constraints for a hybrid prediction or control system.

Chapter 1

Introduction

Reality is usually not linear - rather, the dynamics and interrelationships in nature, as well as in technical systems, are often complex, nonlinear and even chaotic. While simplification and linearization are sufficient for many modeling tasks, real-world applications typically require consideration and understanding of the full dynamics. Physics has a long tradition of developing the right tools for this. While Newton succeeded in solving the two-body problem in 1687, the extension to a three-body problem made it impossible to find a general closed-form solution. The reason is that the system exhibits sensitive dependence on initial conditions as shown by Henry Poincaré in the 1890s leading to chaotic dynamics. Lacking analytical solutions to most complex problems, physicists came up with approximate techniques and topological approaches to understand and describe the dynamics of the system such as the Poincaré map. However, with the advent of machine learning, it became possible to capture and predict nonlinear dynamical systems in their entire complexity.

1.1 Prediction of nonlinear dynamical systems

For many practical applications, a good prediction of the system under consideration is valuable. Without knowledge of the underlying equations, this is typically a difficult task and requires a reasonable model. Most models are based on historical observations of the system in the form of time series. A time series $\mathcal{S} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ describes the time evolution of a dynamical system, where $\boldsymbol{x}_t$ denotes the respective state variable at time t. In classical statistics, various properties of the time series are analyzed and inferred such as the distribution and its moments, stationarity or dynamical features such as autocorrelation. If multiple variables of the same system are captured, also multivariate effects such as correlations are of interest.

For numerous problems in science and applications it is very useful to be able to generate predictions about the future development of the system under investigation. In the following, the way from a naive model to a sophisticated nonlinear approach is presented [1]. At first, the situation is investigated where there is no sequential order or temporal structure in the data and thus $\mathcal{S}$ is strictly speaking not a time series but a collection of data. The task is to create a model $\boldsymbol{y} = f(\mathcal{S}_x)$ that is able to predict the output of an variable $\mathcal{S}_y$ depending on another variable $\mathcal{S}_x$ given training data $\{\boldsymbol{x}_j, \boldsymbol{y}_j\}$. This is called supervised machine learning and the easiest model is simple linear regression

$$f(\mathcal{S}_x) = \sum_{i=1}^{d} w^{(i)} \boldsymbol{x}^{(i)} + w_0 \,, \tag{1.1}$$

where $\{w^{(i)}\}$ and w_0 are determined via minimizing the error between $f(\mathcal{S}^x)$ and the training data $\mathcal{S}_y$. This can be achieved e.g. by solving the quadratic loss function

$$min \sum_j (\sum_{i=1}^{d} w_i \boldsymbol{x}_j^{(i)} - \boldsymbol{y}_j^{(i)})^2 \,. \tag{1.2}$$

By construction, this model can only learn linear dependency and thus will lead to poor performance for more complex systems and real world applications. This is because any nonlinearity in the system will deteriorate the predictive power of the model. To tackle this problem, one has to introduce nonlinearity into the model. However, it is not as straightforward to choose a nonlinear functional form that represents the data well.

A possible solution to this is the use of an artificial neural network (ANN) to model the complex and potentially nonlinear dependencies. The underlying idea of ANNs came up first in the 1940s when researchers developed a mathematical model for neurons in a biological brain [2] and theories for the process of learning [3]. It consists of nodes (neurons) that are connected by edges (synapses). In the 1950s, a computer was used to simulate such a network [4] followed by the invention of the first ANN – the perceptron [5]. With the rise of computational power in the previous decade, evolution in the area of ANNs led to groundbreaking advances such as facial recognition and autonomous driving [6] or the dominance of ANNs in the complex game Go [7]. Related to the time series prediction problem, the idea is that the d-dimensional input data $\boldsymbol{x}_j$ is fed into an artificial neural network $\boldsymbol{A}$ with N nodes via an $N \times d$ dimensional input mapping $\boldsymbol{W}_{in}$. This is done using an activation function $\sigma(\boldsymbol{W}_{in}\boldsymbol{x}_j, \boldsymbol{A})$, which is typically a nonlinear function such as *tanh* or *sigmoid*. This is important because depending on the network architecture, the activation function is often the only part that introduces nonlinearity into the model. A prediction $\boldsymbol{y}_j$ can then be obtained by applying an output function $\boldsymbol{W}_{out}$ such that

$$\boldsymbol{y}_j = \boldsymbol{W}_{out}(\sigma(\boldsymbol{W}_{in}\boldsymbol{x}_j, \boldsymbol{A}) \,. \tag{1.3}$$

Above simple architecture can be extended by concatenating a number of l multiple layers

$$\boldsymbol{y}_j = \boldsymbol{W}_{out}(\sigma(\boldsymbol{W}^l(\; ... \; \sigma(\boldsymbol{W}^1\sigma(\boldsymbol{W}_{in}\boldsymbol{x}_j, \boldsymbol{A}))) \, . \tag{1.4}$$

This is then called a deep neural network. As stated initially, input data is fed in not sequentially but simultaneously, which is typically the case for e.g. pattern recognition tasks. For time series prediction, a temporal structure has to be imposed. Data $\mathcal{S}^x$ is now a time series and the goal is to predict a future value $\boldsymbol{y}_{t+1}$ based on past values of the time series $\{\boldsymbol{x}_j\}_{j=t-k}^t$, where k denotes the number of observations taken into account. In this case, the simplest approach would be to set up a naive model $\boldsymbol{y}_{t+1} = f(\boldsymbol{x}_t) = \boldsymbol{x}_t$ which assumes that the predicted next state $\boldsymbol{y}_{t+1}$ is equal to the last observed state $\boldsymbol{x}_t$. An apparent extension is to not only base the forecast on the last value, but on the weighted average of multiple past observations such that

$$\boldsymbol{y}_{t+1} = f(\{\boldsymbol{x}_j\}_{j=t-k}^t) = \boldsymbol{c} + \sum_{j=t-k}^{t} \alpha_j \boldsymbol{x}_j + \epsilon_t \, , \tag{1.5}$$

where α_j are the weighting coefficients, ϵ_t is a random process generating noise and $\boldsymbol{c}$ denotes a constant that regulates drift. This is referred to as an autoregressive model. Analogous to the simple linear regression model in the non-temporal case, this approach suffers from a lack of nonlinearity and thus fails to well describe more complex dynamics. However, the above introduced ANN framework can be extended to capture temporal structure. To do so, the input data has to be entered in a sequential way and the ANN has to feed its past state back into the ANN in order to allow for memory. This is then called a recurrent neural network, where the state of the ANN $\boldsymbol{r}(t)$ evolves as

$$\boldsymbol{r}(t+1) = \sigma(\boldsymbol{W}_{in}\boldsymbol{x}_t, \boldsymbol{A}, \boldsymbol{r}(t)) \, . \tag{1.6}$$

Output is then generated by applying the output function to the current state of the ANN such that $\boldsymbol{y}(t) = \boldsymbol{W}_{out}(\boldsymbol{r}(t))$. The learning task is to find suitable parameters for $\boldsymbol{W}_{in}$, $\boldsymbol{W}_{out}$ and $\boldsymbol{A}$ by minimizing the error between predictions $\{\boldsymbol{y}(t)\}$ and training data $\{\bar{\boldsymbol{y}}(t)\}$. A concrete architecture is presented in Section 2.1. As for the non-temporal case, the layout can be extended to multiple layers to obtain a deep recurrent neural network.

1.2 Chaos and control of nonlinear dynamical systems

Having invented chaos theory in 1963 [8], Lorenz summarized it later as *"Chaos: When the present determines the future, but the approximate present does not approximately determine the future"* [9]. He presented a finite system of ordinary nonlinear

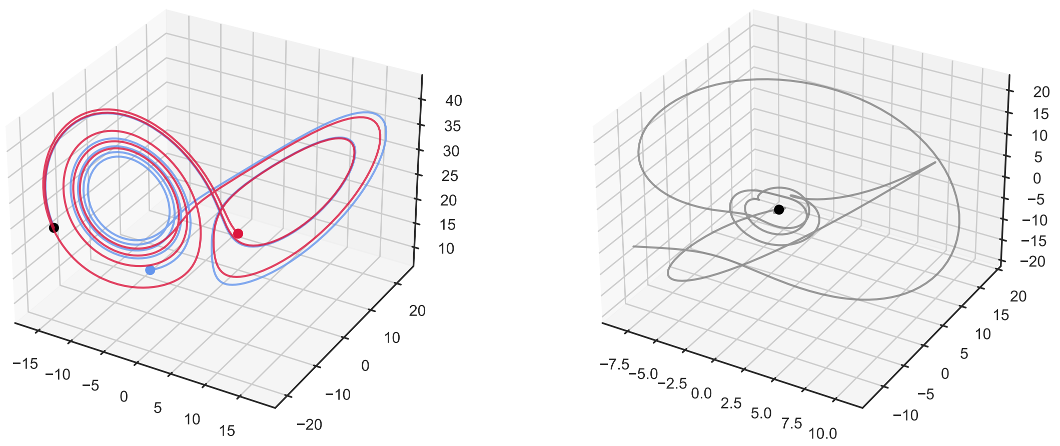

Figure 1.1: The left figure shows trajectories of the Lorenz system with standard parameters for chaotic behavior $\sigma = 10$, $\beta = 8/3$ and $\rho = 28$. The blue trajectory corresponds to initial conditions (-14.0, -20.0, 25.0), while the red trajectory starts slightly shifted at (-14.0, -20.1, 25.0) corresponding to the black dot. The colored dots denote the end points of both trajectories after 1000 simulated time steps with $\Delta t = 0.005$. The right figure shows the difference between the blue and red trajectory and the black represents the starting point.

differential equations modelling atmospheric convection and found that non-periodic solutions are unstable with respect to small disturbances. This leads to the common definition of chaos as sensitive dependence on initial conditions. As a consequence, even tiny rounding or measurement errors of chaotic systems can lead to completely different trajectories and thus chaotic behavior although the system is deterministic. The equations of the Lorenz system read

$$\begin{aligned}\dot{x} &= \sigma(y - x)\\ \dot{y} &= x(\rho - z) - y\\ \dot{z} &= xy - \beta z\ ,\end{aligned} \tag{1.7}$$

where σ, ρ and β are parameters that determine whether the system exhibits periodic, intermittent or chaotic behavior and $\dot{x}$ denotes the time derivative of the variable x. Figure 1.1 shows the sensitive dependence on initial conditions. By only slightly shifting the y-coordinate of the starting point from -20.0 to -20.1 the resulting trajectories quickly separate and end up on different sides of the attractor as denoted by the blue and red dots, respectively. The typical chaotic pattern of the single coordinates of the trajectory can be seen in Fig. 1.2. Concerning numerous real application systems, the question therefore arises whether it is possible to eliminate

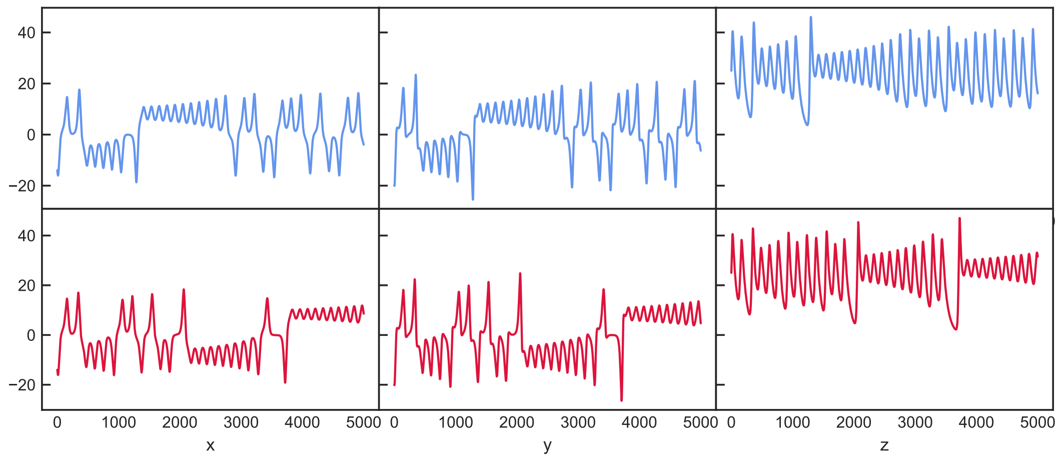

Figure 1.2: The top plots show the x, y and z coordinates of the Lorenz system with standard parameters for chaotic behavior $\sigma = 10$, $\beta = 8/3$ and $\rho = 28$ and initial conditions (-14.0, -20.0, 25.0), while the lower plots correspond to slightly shifted initial conditions at (-14.0, -20.1, 25.0). The trajectories are simulated for 5000 time steps with $\Delta t = 0.005$.

this irregularity and force the system into a periodic state. Chaos control relies on the idea that unstable periodic orbits can be stabilized by small perturbations of the system, which are achieved by applying an external force. Among several approaches there are two basic methods for this: OGY control [10] and Pyragas' [11] time delayed feedback control. The former relies on the Poincaré map, which is a useful tool for the study of dynamical systems [12]. Consider an n-dimensional dynamical system $\dot{\boldsymbol{x}} = f(\boldsymbol{x})$ – with $n = 3$ in the case of Lorenz – where S is an $n-1$ dimensional surface of section transverse to the flow. Then the Poincaré map P is the mapping that describes where a trajectory crossing S will intersect S the next time such that

$$\boldsymbol{x}_{k+1} = P(\boldsymbol{x}_k, p) \ , \tag{1.8}$$

where $\boldsymbol{x}_k$ denotes the k-th intersection with S and p is a controllable parameter of the system. If $\boldsymbol{x}^*$ is a fixed point of P, then it follows $P(\boldsymbol{x}^*) = \boldsymbol{x}^*$. Therefore, a trajectory starting at $\boldsymbol{x}^*$ returns to $\boldsymbol{x}^*$ after some time and thus results in a periodic orbit. Analyzing the behavior of P near the fixed point exhibits then the stability of the periodic orbit. The idea is now to apply a small feedback by changing p if the trajectory is close to a fixed point $\boldsymbol{x}^*$ of P. When this is the case, the system can be linearized around the fixed point

$$\boldsymbol{x}_{k+1} - \boldsymbol{x}^* = A(\boldsymbol{x}_k - \boldsymbol{x}^*) + B(p - \bar{p}) \ , \tag{1.9}$$

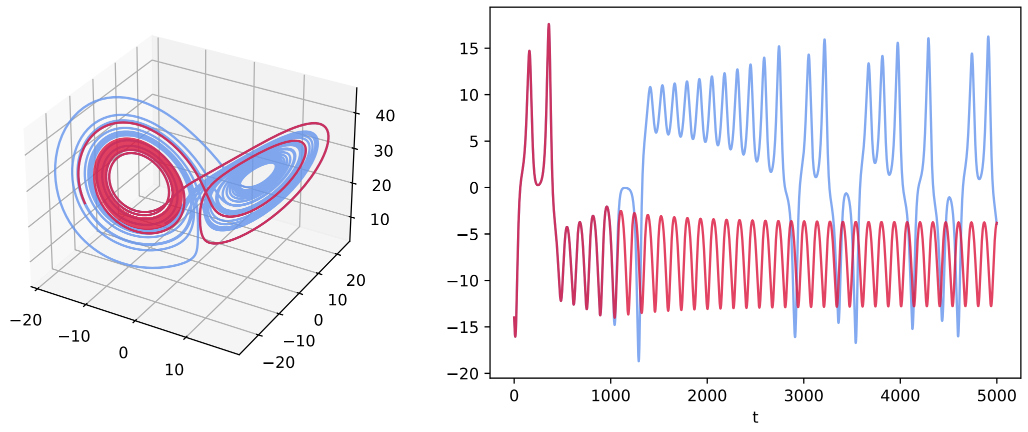

Figure 1.3: The left plot shows the Lorenz system with standard parameters for chaotic behavior $\sigma = 10$, $\beta = 8/3$ and $\rho = 28$. The blue trajectory represents the original attractor, while the red trajectory shows the attractor after time delayed feedback control has been switched on in timestep $t = 1000$. The right plot shows the x coordinate, accordingly. Time resolution is $\Delta t = 0.005$.

where A is the Jacobian of the Poincaré Map, B describes the sensitivities with respect to changing the parameter p and $\bar{p}$ denotes the original parameter value:

$$A = \frac{\partial P}{\partial \boldsymbol{x}}(\boldsymbol{x}^*, \bar{p}) \ , \quad B = \frac{\partial P}{\partial \boldsymbol{p}}(\boldsymbol{x}^*, \bar{p}) \ . \tag{1.10}$$

Then the linear feedback is

$$p - \bar{p} = -K(\boldsymbol{x}_k - \boldsymbol{x}^*) \ , \tag{1.11}$$

where K is yet to be determined. By substituting Eq. 1.11 into the equation for the linearized dynamics around $\boldsymbol{x}^*$ Eq. 1.9 one obtains

$$\boldsymbol{x}_{k+1} - \boldsymbol{x}^* = (A - BK)(\boldsymbol{x}_k - \boldsymbol{x}^*) \ , \tag{1.12}$$

where stability depends on the eigenvalues of $(A - BK)$ and the system is stable if $\mid eig(A - BK) \mid < 1$. The task is therefore to chose K such that this condition holds. This can be done using the pole placement approach [13], where the unstable poles are set to zero and the stable poles remain unchanged. The difficulty in practice, however, is to evaluate A and B which typically requires large amounts of data for a sufficiently accurate state space reconstruction.

In contrast to the OGY control, the time delayed feedback control by Pyragas [11] gives a continuous feedback into the system. The underlying idea is that using the concept of delay coordinates, a large number of unstable periodic orbits can be obtained from a measured scalar signal of the attractor [14, 15], where $\boldsymbol{y}(t) = \boldsymbol{y}(t+\tau)$ and τ is the period of the unstable periodic orbit. Then, a feedback force can be applied which is defined as

$$F = K(\boldsymbol{x}(t+\tau) - \boldsymbol{x}(t)) \ , \tag{1.13}$$

where $K > 0$ is an adjustable parameter. The force is then continuously applied to the system such that

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}) + F(t, \tau) \ . \tag{1.14}$$

Figure 1.3 shows the Pyragas control on the example of the Lorenz system being controlled from a chaotic state into a periodic state. For the time delay, $\tau = 120$ is used, while the coupling parameter has been empirically set to $K = 0.3$. After the control force is turned on in timestep $t = 1000$, the system is kept in a periodic behavior as denoted by the red trajectory. In contrast, the uncontrolled trajectory exhibits chaotic behavior. While there are several extensions of the above introduced classical control schemes, most approaches require either knowledge about the underlying equations or large amounts of data as they rely on phase space methods.

1.3 Aim of this thesis

The aim of this thesis is a step towards de-blackboxing recurrent neural network based predictions of nonlinear dynamical systems. While there have been great successes like the prediction of chaotic dynamics, it is not sufficiently understood which features or topological aspects of the recurrent neural networks lead to good or bad predictions. In the first step, a statistical approach is taken in order to evaluate variability of the outcomes depending on key parameters of the network as well as different network topologies. Here, the goal is to find a way to reduce variability and make forecasts more stable in terms of their dynamical properties called *climate* in the following. Furthermore, the question of whether exact short-term predictions coincide with a good reproduction of the long-term statistical climate of the system and vice versa is illuminated. Based on these findings, the next step is to gain an understanding how differential properties of the system are related to the goodness of the prediction. These include direct manipulations of the network by taking out certain nodes. In addition, the influence of nonlinearity in the activation function is investigated. These insights lead to the ability of producing reliable predictions of chaotic system, which are then used to develop a flexible novel control scheme. By applying an external feedback to the system, it can be forced from its original state into into another dynamical target state. Since previous approaches either rely on the knowledge of the underlying equations – which for most real world systems are unknown – or on large amounts of data, the idea is to develop a new machine learning based approach with only moderate data requirements. Furthermore, a greater flexibility regarding the dynamical target state is aimed for as most existing techniques can only control the system into simple dynamics or rather unspecified complex dynamics. To gain a better understanding of the dynamics of a given system, also the causality structure is analyzed. The aim is to separate linear and nonlinear contributions to the system causality, which could be useful to build hybrid machine learning models to further improve prediction quality.

Chapter 2

Methods

2.1 Reservoir Computing

Reservoir computing (RC) [16–18] is an artificial recurrent neural network (RNN) based method relying on a static internal network called *reservoir*. The term static means that – unlike other RNN approaches – the reservoir is kept fixed once the network has been initially constructed. The same holds for the input function $\mathbf{W}_{in}$ and therefore the RC system is computationally very efficient as the training process only involves optimizing the output layer. As a result, fast training and high model dimensionality is computationally feasible making the model well suited for complex real-world applications.

Typically, the reservoir $\mathbf{A}$ is constructed as a sparse Erdös-Renyi random network [19] with dimensionality and thus number of nodes D_r, which are connected with a probability p leading to an average unweighted degree of d. Initially, the weights of the edges are randomly drawn from an uniform distribution within the interval $[-1, 1]$. The scaling of the weights affects one of the key hyperparameters of RC being the spectral radius ρ of the reservoir $\mathbf{A}$. It is defined as its largest absolute eigenvalue

$$\rho(\mathbf{A}) = max\{|\lambda_1|, ..., |\lambda_{D_r}|\} \ , \tag{2.1}$$

and reflects the average degree of the network. By its very definition, the eigenvalues of a matrix can be rescaled by applying a scalar factor to the reservoir $\mathbf{A}$

$$\mathbf{A}^* = \frac{\mathbf{A}}{\rho(\mathbf{A})}\rho^* \ , \tag{2.2}$$

where ρ^* is the targeted spectral radius. To feed the D dimensional input data $\mathbf{u}(t)$ into the reservoir $\mathbf{A}$, a $D_r \times D$ input mapping matrix $\mathbf{W}_{in}$ is set up, which defines

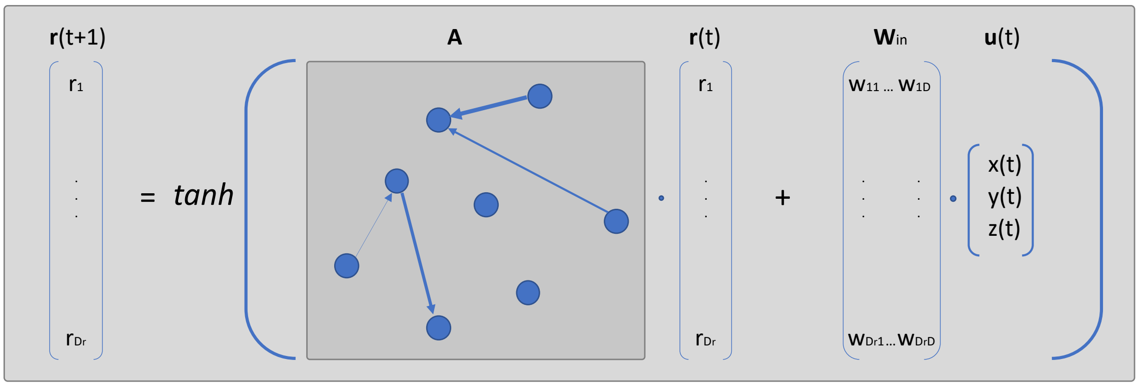

Figure 2.1: Schematic illustration of the evolution of reservoir states $\mathbf{r}$(t).

how strongly each input dimension influences every single node. The elements of $\mathbf{W}_{in}$ are chosen to be uniformly distributed random numbers within the interval $[-\omega, \omega]$. In general, nodes can be fed by multiple input dimensions. However, throughout this thesis we construct $\mathbf{W}_{in}$ such that every node has only one nonzero entry among the D input dimensions. The dynamics of the RC system are contained in its $D_r \times 1$ dimensional reservoir states $\mathbf{r}(t)$. Being initially set to $r_i(t_0) = 0$ for all nodes, the evolution over time is according to the recurrent equation

$$\mathbf{r}(t+1) = \alpha\mathbf{r}(t) + (1-\alpha)tanh(\mathbf{A}\mathbf{r}(t) + \mathbf{W}_{in}\mathbf{u}(t)) \ . \tag{2.3}$$

The parameter α defines to what extend the updated reservoir states are mixed with those from the previous timestep. The activation function is chosen to be the hyperbolic tangent and serves as the only nonlinear component in the standard RC setup. An extensive overview and explanation of all mentioned components is given in [20]. The updating process of the reservoir states $\mathbf{r}$(t) is visualized in Fig. 2.1. Here, the reservoir $\mathbf{A}$ is schematically represented and the bubbles correspond to nodes, while arrows denote directional connections between the nodes. The width of the arrow stands for the respective edge weight. In the adjacency matrix representation of $\mathbf{A}$, the respective edge weights between nodes are then off-diagonal matrix elements which are nonzero when nodes are connected. For most RC applications, the reservoir is a sparse matrix, which corresponds to a small probability p of connecting nodes when constructing $\mathbf{A}$ as an Erdös-Renyi random network. To obtain D dimensional output from the reservoir states $\mathbf{r}(t)$, an output mapping function $\mathbf{W}_{out}$ is required. It linearly depends on the training matrix $\mathbf{P}$

$$\mathbf{v}(t) = \mathbf{W}_{out}(\mathbf{r}(t), \mathbf{P}) = \mathbf{P}\tilde{\mathbf{r}}(t) \ , \tag{2.4}$$

where $\tilde{\mathbf{r}}(t)$ is a function of $\mathbf{r}(t)$ that is often chosen as $\tilde{\mathbf{r}}(t) = \mathbf{r}(t)$. However, this can lead to severe problems due to the antisymmetry of the hyperbolic tangent as explained in [21]. To break this symmetry, a quadratic readout $\tilde{\mathbf{r}} = \{\mathbf{r}, \mathbf{r}^2\}$ can be added. This means that the squared elements of the reservoir states $\mathbf{r}^2 = \{r_1^2, r_2^2, ..., r_{D_r}^2\}$ are appended and therefore the output mapping matrix $\mathbf{P}$ now contains $2D_r \times D$ degrees of freedom. Determining its coefficients is called *training*. This is accomplished by acquiring a sufficient number of reservoir states $\mathbf{r}(t_w ... t_{w+T})$ and then choosing $\mathbf{P}$ such that the output $\mathbf{v}$ of the reservoir is as close as possible to the known test data $\mathbf{v}_R(t_w ... t_{w+T})$. Before the training process is started, the RC system should be initialized during a washout phase of t_w time steps in order synchronize the reservoir states $\mathbf{r}(t)$ with the dynamics of the input signal $\mathbf{u}$. Then the training can be executed by using Ridge regression, which minimizes

$$\sum_{w \le t \le T} \parallel \mathbf{W}_{out}(\tilde{\mathbf{r}}(t), \mathbf{P}) - \mathbf{v}_R(t) \parallel^2 - \beta \parallel \mathbf{P} \parallel^2 , \tag{2.5}$$

where β is the regularization constant that prevents from overfitting by penalizing large values of the elements of $\mathbf{P}$. This can be rewritten in matrix form [22] such that:

$$\mathbf{P} = \left(\mathbf{r}^T\mathbf{r} + \beta \mathbb{1}\right)^{-1} \mathbf{r}^T \mathbf{v}_R \ . \tag{2.6}$$

After the training, the predicted state $\mathbf{v}(t)$ can be fed back in the activation function as input instead of the actual data $\mathbf{u}(t)$ by combining Eq 2.3 and Eq 2.4. The resulting recursive form of the equation for the reservoir states $\mathbf{r}(t)$ allows to create predicted trajectories of arbitrary length:

$$\begin{aligned} \mathbf{r}(t+1) &= tanh(\mathbf{A}\mathbf{r}(t) + \mathbf{W}_{in}\mathbf{W}_{out}(\tilde{\mathbf{r}}(t), \mathbf{P})) \qquad (2.7) \\ &= tanh(\mathbf{A}\mathbf{r}(t) + \mathbf{W}_{in}\mathbf{P}\tilde{\mathbf{r}}(t)) \ . \qquad (2.8) \end{aligned}$$

Figure 2.2 presents a schematic representation of the reservoir computing framework.

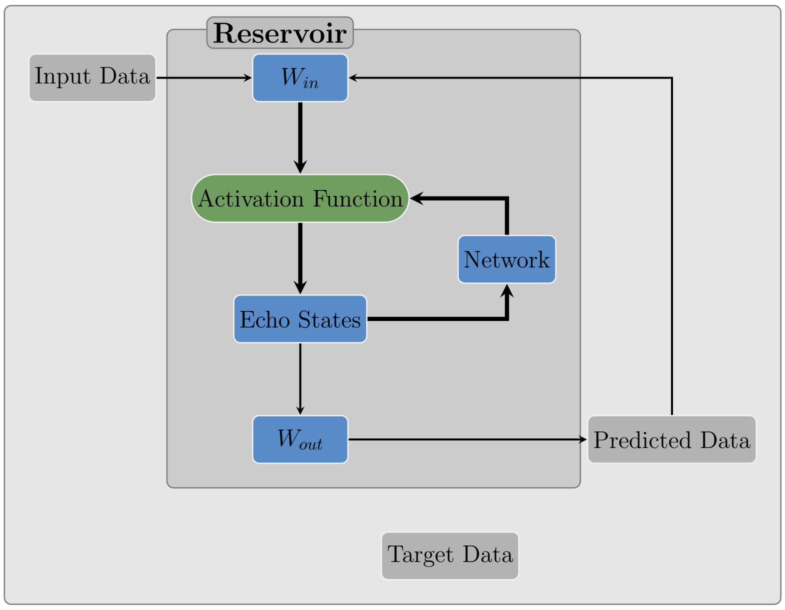

Figure 2.2: Schematic illustration of reservoir computing.

2.2 Measuring the climate of a dynamical system

When forecasting nonlinear dynamical systems such as chaotic attractors, the goal of the predictions is not only to hit the actual short-time trajectory exactly but rather to reproduce the long-term statistical properties of the system called *climate*. This is important because by its definition, chaotic systems exhibit sensitive dependence on initial conditions and therefore small disturbances grow fast. Consequently, even if at first the short term prediction is perfect, at some stage already numerical inaccuracies can lead to the separation of the predicted and actual trajectories. For many applications, however, this is not a problem as long as the predicted trajectory sill leads to the same attractor. To quantify this behavior, quantitative measures are needed that grasp the complex dynamics of the system.

To assess the structural complexity of an attractor, its correlation dimension is calculated, which measures the dimensionality of the space populated by the trajectory [23]. It belongs to the measures for fractal dimensionality, which have been brought up by Mandelbrot in 1967 [24]. The correlation dimension is based on the correlation integral

$$\begin{aligned} C(r) &= \lim_{N\to\infty} \frac{1}{N^2} \sum_{i,j=1}^{N} \theta(r - |\mathbf{x}_i - \mathbf{x}_j|) \\ &= \int_0^r d^3 r' c(\mathbf{r}') \ , \end{aligned} \tag{2.9}$$

where θ is the Heaviside function and $c(\mathbf{r}')$ is the standard correlation function. The correlation integral represents the mean probability that two states in phase space are close to each other at different time steps. This is the case if the distance between the two states is less than the threshold distance r. The correlation dimension ν is then defined by the power-law relationship

$$C(r) \propto r^{\nu} \ . \tag{2.10}$$

For self-similar strange attractors, this relationship holds for a certain range of r, which therefore needs to be properly calibrated. The calculation of the correlation dimension is done using the Grassberger Procaccia algorithm [25] and therefore is purely data based and does not require any knowledge of the underlying equations of the system. One advantage of the correlation dimension over other fractal measures is that it can be calculated having a comparably small number of data points available. Throughout this thesis, mainly the relative comparison among various predictions and actual trajectories is of interest and therefore the accuracy of the absolute values is not the highest priority.

Besides the fractal dimensionality, the statistical climate of an attractor is also characterized by its temporal complexity as measured by the Lyapunov exponents. They describe the average rate of divergence of nearby points in phase space, and thus measure sensitivity to initial conditions. There is one exponent for each dimension in phase space. If the system exhibits at least one positive Lyapunov exponent $\lambda_i > 0$, it is classified as chaotic. The magnitudes of λ_i quantify the time scale on which the system becomes unpredictable [26, 27]. Since at least one positive exponent is the requirement for being classified as chaotic, it is sufficient for the purposes in this thesis to calculate only the largest Lyapunov exponent λ_{max}

$$d(t) = Ce^{\lambda_{max} t} \ . \tag{2.11}$$

This makes the task computationally much easier than determining the full Lyapunov spectrum and describes the overall system behavior to a large extent. We use the Rosenstein algorithm [28] to obtain it. In essence, it tracks the distance $d(t)$ of two initially nearby states in phase space. The constant C normalizes the initial separation. As for the correlation dimension, mainly a relative comparison is of interest to characterize states of the system rather than the exact absolute values. Again, no model or knowledge of the underlying equations is required.

Chapter 3

Prediction variability of Reservoir Computing and the effect of network topology

This chapter is based on the following paper, which is listed in this thesis as Ref [29].

> A. Haluszczynski and C. Räth, "Good and bad predictions: assessing and improving the replication of chaotic attractors by means of reservoir computing", Chaos: An Interdisciplinary Journal of Nonlinear Science **29**, 103143 (2019)

3.1 Objectives

The prediction of spatiotemporal dynamics of nonlinear systems using reservoir computing (RC) has attracted much attention [30–37] in the recent years. Many of these results are impressive, but are based on single executions of reservoir computing and therefore do not allow conclusions about robustness due to the computing model being based on random numbers. The objective of this paper is on the first hand to conduct a thorough statistical analysis by using multiple random realizations of a RC setup with same hyperparameters to analyze the prediction robustness both in terms of exact short-term predictions and the long-term statistical properties of the system. Further, besides the standard Erdös-Renyi random network architecture [19] also scale free [38] networks and small world networks [39] are evaluated. It is shown in the following that the prediction variability and hence robustness of RC strongly differs depending on the parameter choice and does not depend much on network topology.

3.2 Methods

Analogous to Pathak et al. [31] and Lu et al. [36] the focus lies on low-dimensional chaotic systems and therefore the Lorenz system [8] is used as an example. It has been developed as a simplified model for atmospheric convection and exhibits chaos for certain parameter ranges, while other parameter choices also lead to intermittent and periodic behavior. The equations read

$$\begin{aligned}\dot{x} &= \sigma(y-x)\\ \dot{y} &= x(\rho - z) - y\\ \dot{z} &= xy - \beta z + x\end{aligned} \tag{3.1}$$

where the x term in the equation for $\dot{z}$ is added in order to break the symmetry in x and y of the standard Lorenz system with respect to the transformation $x \to -x$ and $y \to -y$. This system is referred to as modified Lorenz system. As reported by Lu et al. [40] such a symmetry can be an issue when inferring the x and y dimension from knowledge of the z dimension. The standard parameters for chaotic behavior $\sigma = 10, \beta = 8/3$ and $\rho = 28$ are used. In addition to the Lorenz system the analysis is also carried out for the Rössler system [41]

$$\begin{aligned}\dot{x} &= -y - z\\ \dot{y} &= x + ay\\ \dot{z} &= b + z(x-c)\ ,\end{aligned} \tag{3.2}$$

where parameters $a = 0.5$, $b = 2.0$ and $c = 4.0$ lead to chaotic behavior. It is regarded as less chaotic than the Lorenz system because its equations have only one quadratic nonlinearity in the z dimension. Therefore, it is interesting to compare its short-term predictability with that of the Lorenz system. Both sets of differential equations are solved using the 4th order Runge-Kutta method [42] with a time resolution $\Delta t = 0.02$ for the Lorenz system and $\Delta t = 0.05$ for the Rössler system. To predict those chaotic systems, a reservoir computing approach is used as introduced in Section 2.1. The reservoir is first constructed as a sparse Erdös-Renyi random network with $D_r = 300$ nodes and connection probability $p = 0.02$ such that the unweighted average degree is $d = 6$. Besides the Erdös-Renyi random network, also random small world networks as a choice for the reservoir $\mathbf{A}$ are investigated. Those are of potential interest, as many real world phenomena such as social networks, electric power grids, chemical reaction networks and neuronal networks exhibit the small world property [43]. Its characteristic is that the average distance between any pair of nodes is small, while at the same time the network exhibits a high average clustering coefficient. To have comparable sparsity and thus average degree d as the Erdös-Renyi network, each node is connected with its six nearest neighbours implying periodic boundary conditions. Nearest neighbour connections are defined as nonzero

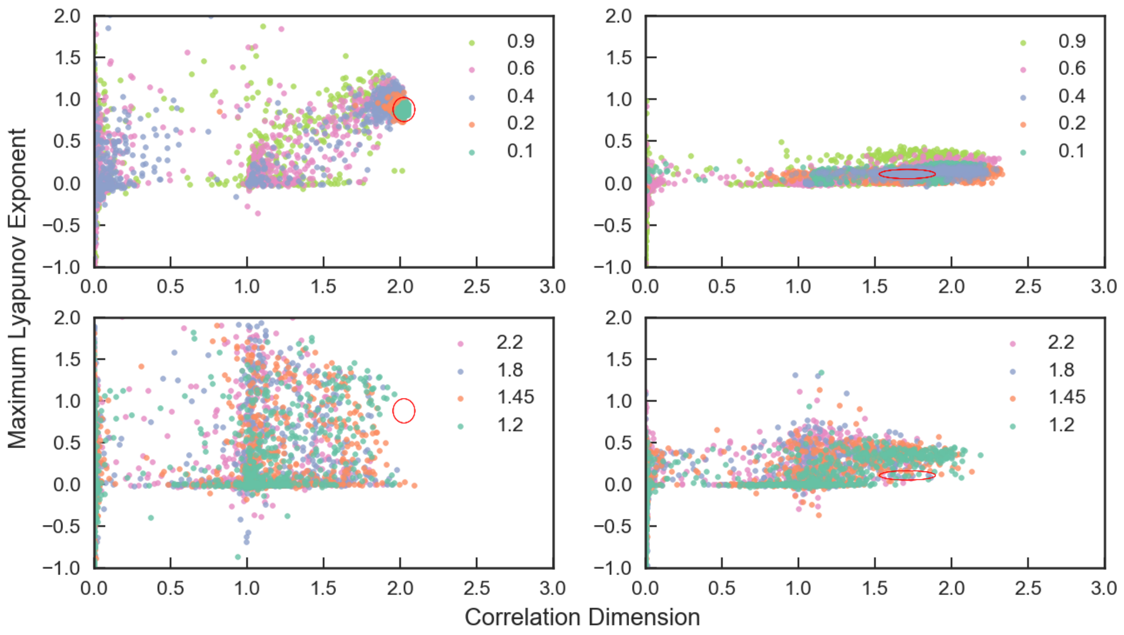

Figure 3.1: Largest Lyapunov exponent scattered against the correlation dimension for each of the $N = 1000$ predictions per spectral radius. Results are shown for the Lorenz (left) and Rössler system (right). Different spectral radii are differentiated by colours. The red object represents the five sigma error ellipse of both measures calculated based on 1000 simulated true trajectories. Random Erdös-Renyi networks are used for the reservoir $\mathbf{A}$.

elements of the adjacency matrix of $\mathbf{A}$ which are directly located next to the diagonal elements. This now gives a network with high clustering coefficient but does not fulfil the condition of small average distances between two nodes. This can be achieved by looping over each edge $x - y$ and rewire it to $x - z$ with probability $p = 0.2$, where node z is randomly chosen. Furthermore, also scale free network topology is tested. Those are graphs where the distribution of the number of edges per node decays with a power law. Many real world networks exhibit not only the small world property but are also scale free. These include the previously mentioned electric power grid networks and neuronal networks and furthermore the world wide web or networks of citations of scientific papers [43]. To construct the reservoir this way, the scale free graph generator of the *NetworkX* package [44] is used with parameters $\alpha = 0.285, \beta = 0.665, \gamma = 0.05, \delta_{in} = 0.2, \delta_{out} = 0$ again leading to an average degree of $d = 6$. For training, 5000 time steps are used, of which 100 are run as washout phase for initialization. For the assessment of short-term prediction quality the

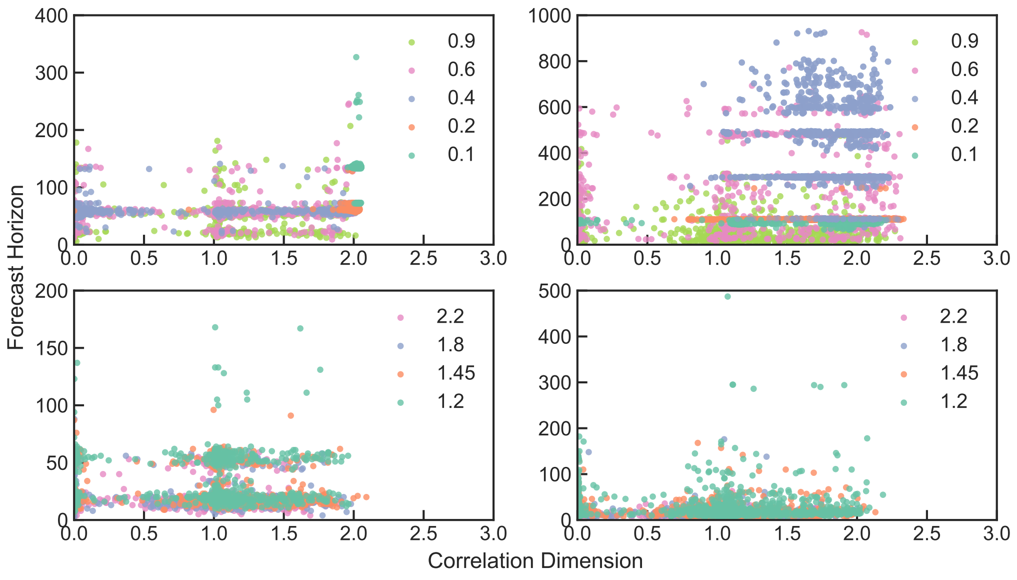

Figure 3.2: Forecast horizon scattered against the correlation dimension for each of the $N = 1000$ predictions per spectral radius. Results are shown for the Lorenz (left) and Rössler system (right). Different spectral radii are differentiated by colours. Random Erdös-Renyi networks are used for the reservoir $\mathbf{A}$.

forecast horizon is chosen as evaluation measure. It tracks for how long the distance between the predicted and actual trajectory lies below a certain threshold value

$$|\mathbf{v}(t) - \mathbf{v}_R(t)| < \boldsymbol{\delta} \ , \tag{3.3}$$

where the thresholds are $\boldsymbol{\delta} = (5, 10, 5)^T$ for the Lorenz system and $\boldsymbol{\delta} = (2.5, 2.5, 4)^T$ for the Rössler system. The values are chosen this way due to the different ranges of the state variables in both systems. The aim is that small fluctuations around the actual trajectory do not immediately lead to the prediction being classified as not matching anymore. Empirically it was found that distances between the trajectories become much larger than the threshold values as soon as short-term prediction collapses. To quantify the long-term statistical climate, the correlation dimension and largest Lyapunov exponent are calculated as introduced in Section 2.2.

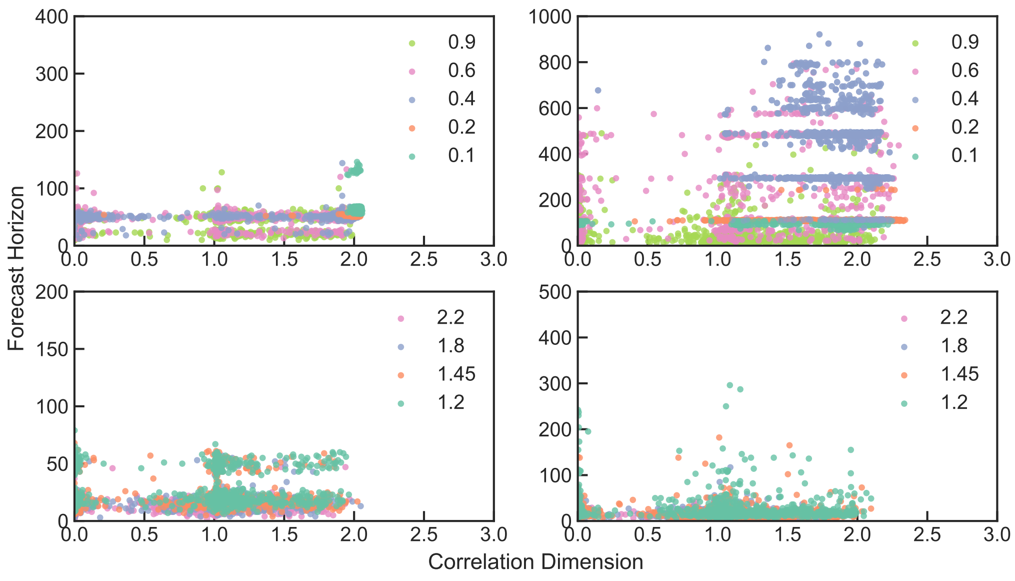

Figure 3.3: Forecast horizon scattered against the correlation dimension for each of the $N = 1000$ predictions per spectral radius. Results are shown for the Lorenz (left) and Rössler system (right). Different spectral radii are differentiated by colours. Small world networks are used for the reservoir $\mathbf{A}$.

3.3 Results

For the assessment of prediction variability depending on the choice for the key parameter spectral radius ρ, $N = 1000$ different random realizations of the RC system have been simulated for each parameter value. Figure 3.1 shows the results evaluated in terms of the statistical climate as expressed by the correlation dimension and largest Lyapunov exponent of the resulting attractor. The red ellipse encloses the five sigma errors of both measures calculated from trajectories based on the actual equations of the Lorenz and Rössler system using 1000 random initial conditions. It is clearly visible that in particular larger values for the spectral radius lead to very high prediction variability with most results being outside the five sigma error ellipse. The left side presents the results for the Lorenz system. Here, for the smallest spectral radius $\rho = 0.1$ the variability is also the smallest but still significant given that five sigma is a large error. For the larger spectral radii shown in the bottom left plot - including the values $\rho = 1.2$ and $\rho = 1.45$ as used in Ref. [31] - there is not a single point within the ellipse. This indicates that the prediction completely

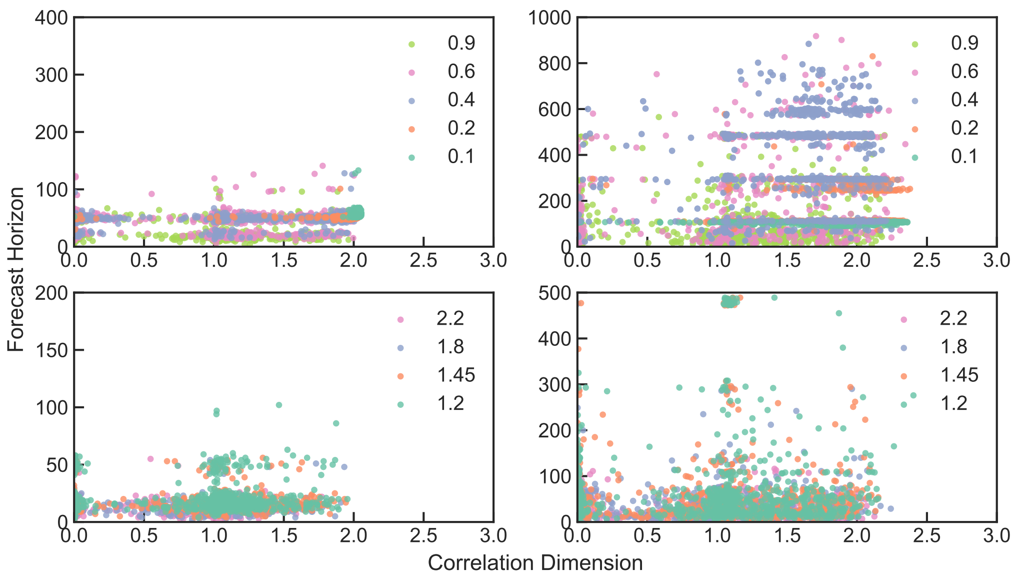

Figure 3.4: Forecast horizon scattered against the correlation dimension for each of the $N = 1000$ predictions per spectral radius. Results are shown for the Lorenz (left) and Rössler system (right). Different spectral radii are differentiated by colours. Scale free networks are used for the reservoir $\mathbf{A}$.

fails to reproduce the long term climate for those cases. The results for the Rössler system on the right side of the plot give a similar picture. In this case even for the best working spectral radius of $\rho = 0.4$ there are many points outside of the $\sigma = 5$ error ellipse.

Besides the statistical climate also the short-term prediction capabilities are of interest. For this, the forecast horizon is plotted against the correlation dimension in Fig. 3.2. The first observation is that the forecast horizon for the Rössler system is significantly longer than for the Lorenz system. This is in line with the initial claim that the Rössler system is less chaotic and thus easier to predict. The second observation is that a good reproduction of the forecast horizon also tends to coincide with a better reproduction of the correlation dimension. Similar to Figure 3.1, the best working spectral radius is $\rho = 0.1$ for the Lorenz system and $\rho = 0.4$ for the Rössler.

The same analysis has been applied to the other network topologies as shown in Fig. 3.3 and Fig. 3.4. Overall the results are very similar among all network topologies. However, scale free networks tend to perform worse for short-term predictions of

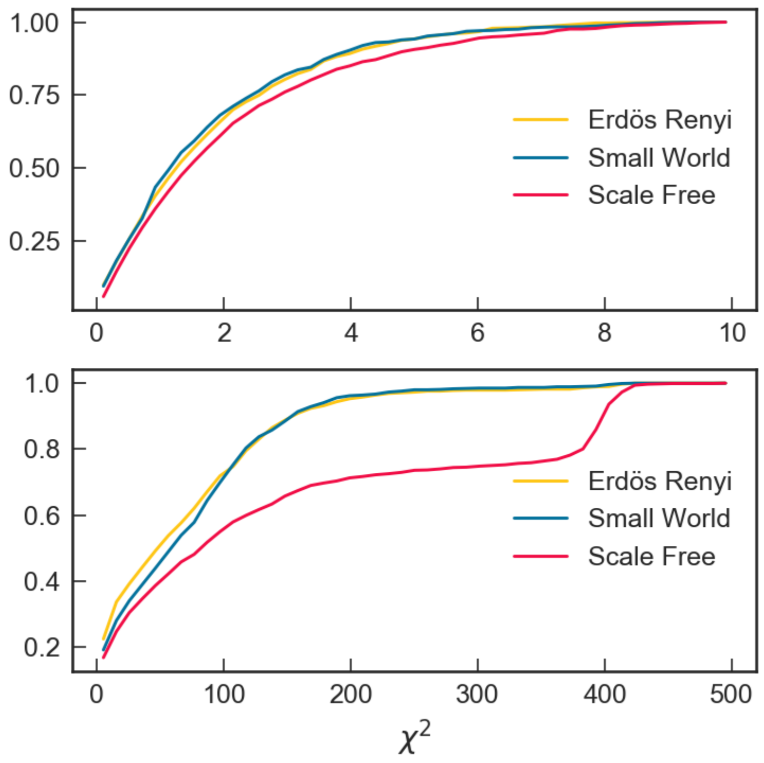

Figure 3.5: Top plot: Cumulative distribution of χ^2 for the best working spectral radius $\rho = 0.1$ of the Lorenz system calculated for values between 0 and 10. Bottom plot: Same for the Rössler system with $\rho = 0.4$ and values between 0 and 500

the Lorenz system with low spectral radii, while there is a number of points for the Rössler system based on high spectral radii that lead to a longer forecast horizon. To compare the performance of the three different network topologies on a statistical level, a χ^2 analysis of the long term climate prediction is performed. For this,

$$\chi^2(i,\rho) = \sum_{j=1}^{2}\left[\frac{X_j(i,\rho) - \langle X_j\rangle}{\sigma_{X_j}}\right]^2 \tag{3.4}$$

is calculated, where i is the $i - th$ random number seed and ρ indexes the spectral radius. The sum goes over the correlation dimension ($j = 1$) and the largest Lyapunov exponent ($j = 2$). $\langle X_j\rangle$ represents the average value derived from 1000 simulated actual Lorenz trajectories and σ_{X_j} is the corresponding standard deviation. Therefore, the resulting χ^2 describes how strong the predicted results deviate from the actual values weighted by the errors of the actual values.

Figure 3.5 shows the cumulative distribution of the χ^2 for all topologies. The top plot contains the results for the Lorenz system with $\rho = 0.1$ and evaluation values of χ^2 for 0 and 10. It is visible that all topologies show comparable performance as already indicated by Fig. 3.3 and Fig. 3.4. However, for the Rössler system there is an underperformance of the scale free network evident. For the bottom plot, the cumulative distribution for values of χ^2 is plotted between 0 and 500 based on $\rho = 0.4$. The reason for the wider range is that even for the best working spectral radius the variability is significantly higher as compared to the Lorenz system. This leads to higher values of χ^2 with only very few data points in the 0 to 10 range.

3.4 Conclusions

This chapter addressed the prediction variability of Reservoir Computing analyzed for different choices of the hyperparameters. In addition, alternative network topologies such as small world and scale free networks have been investigated. The variability is characterized in terms of the forecast horizon and the long term statistical climate expressed as the correlation dimension and the largest Lyapunov exponent of the trajectories. It has been shown that for common choices of the spectral radius ρ, prediction variability is rather high. By choosing appropriate values for ρ – in this case smaller values – variability can be significantly reduced. The reason for this finding lies in the response of the hyperbolic tangent activation function as presented in Ref [45] and Section 4. Furthermore, it turned out that network topology only has minor influence on the prediction variability. As a conclusion, Reservoir Computing users should not only carry out a proper hyperparameter optimization, but also validate their parameter choice by running a statistical analysis over multiple random realizations of the reservoir.

Chapter 4

Improving prediction stability of Reservoir Computing

This chapter is based on the following paper, which is listed in this thesis as Ref [45].

> A. Haluszczynski et al., “Reducing network size and improving prediction stability of reservoir computing”, Chaos: An Interdisciplinary Journal of Nonlinear Science **30**, 063136 (2020)

4.1 Objectives

After it has been shown in Chapter 3 that Reservoir Computing can exhibit high variability in prediction quality depending on the parameter choice, the natural question is where this variability comes from and if prediction quality can be related to differential properties of RC. Those include nodes and edges of the reservoir as well as the input and output weights. First attempts regarding the former were made by Caroll and Pecora [46], who demonstrated that symmetries in a simple reservoir setup with unweighted edges do have a considerable effect on the prediction quality of reservoir computing. In the following it is shown how a controlled removal of nodes influences predictions. Furthermore, the nonlinearity of the activation function is altered by introducing a nonlinear scaling factor in the argument of the hyperbolic tangent. This then explains the high variability for high spectral radii as observed in Chapter 3.

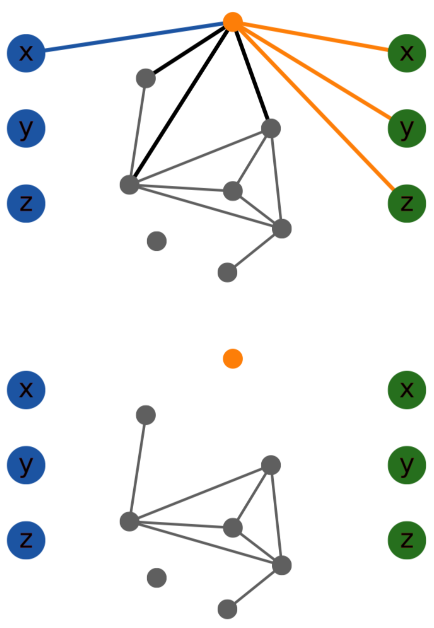

Figure 4.1: Schematic illustration of the controlled node removal procedure. The top graphic shows the initial network before the removal. Input is on the left side, while the right side represents output. Here, the orange example node is being removed. Input/output interactions of the other nodes are not shown here. In the bottom plot the example node has been removed and therefore all connections and interactions vanished.

4.2 Methods

The analysis is mainly based on the Lorenz system as introduced in Section 3.2, as well as a number of other nonlinear dynamical systems [41, 47–52] from the class of autonomous dissipative flows such as the Rössler system [41], Rabinovich-Fabrikant equations [49], Rucklidge system [52] and the Chua circuit [50]. All these systems are three-dimensional but differ in properties like Lyapunov exponents, correlation dimension, size of the attractor and the nature of their nonlinearity. The parameters for all systems except Lorenz and Rössler are taken from the textbook *Chaos and Time-Series Analysis* by Sprott [53]. In addition, also blended systems – a linear combination of two different dynamical systems – have been investigated.

The implementation of Reservoir Computing is based on the setup presented in Section 3.2, however, using different parameters. The reservoir is constructed as a sparse Erdös-Renyi random network with $D_r = 200$ nodes and connection probability

$p = 0.02$, therefore exhibiting an unweighted average degree of $d = 4$. In order to determine suitable parameters for the spectral radius of the reservoir $\rho(\mathbf{A})$, the scale for $\mathbf{W}_{in}$ and the regularization constant β, a random search hyperparameter optimization is carried out. This leverages on the simple architecture of RC allowing for quick training and thus efficient parameter optimization. The forecast horizon as also introduced in 3.2 averaged over 30 random realizations of RC serves as objective function and results in the values $\rho(\mathbf{A}) = 0.17$, $\mathbf{W}_{in}$ scale $= 0.17$ and $\beta = 1.9 \times 10^{-11}$.

To reduce variability and improve computational efficiency, alterations to the reservoir are made by removing nodes including their respective edges from both the reservoir $\mathbf{A}$ and $\mathbf{W}_{in}$. This procedure is inspired by the concept of *attack tolerance* [54] in complex networks. Here, the aim is to investigate the effect of removing nodes on the prediction capabilities of the system. The different magnitudes of the output weights $\mathbf{W}_{out}$ assigned in the training process suggest that their corresponding nodes are either more or less important. Intuitively one would suggest that higher weights correspond to higher importance of the node. The approach is motivated by the assumption that there is a relationship between the importance of each node and its output weights $\mathbf{W}_{out}$ assigned in the training process. To investigate this potential relationship, a fraction p of all nodes is removed that correspond to the largest or smallest elements of $\mathbf{W}_{out}$. However, each node is affiliated with D output weights where D denotes the dimensionality of the system that is being predicted. Therefore, $\mathbf{W}_{out}$ is sorted based on the largest absolute value of all D output weights for each node in order to determine which nodes should be removed. After removal, the newly obtained reduced network is trained again leading to a new set of $\mathbf{W}_{out}$. The node removal process is illustrated in Fig 4.1.

Apart from manipulating the network structure as explained above, also the effect of tuning the nonlinearity of the activation function is studied. This has well-known effects on the memory of the reservoir [55–58]. To tune the nonlinearity, a scaling factor a is introduced in the argument of the hyperbolic tangent such that the update equation for $\mathbf{r}(t)$ now reads:

$$\mathbf{r}(t + \Delta t) = tanh(a[\mathbf{A}\mathbf{r}(t) + \mathbf{W}_{in}\mathbf{P}\mathbf{r}(t)]) \ . \tag{4.1}$$

With the reservoir itself as well as $\mathbf{W}_{in}$ and $\mathbf{W}_{out}$ being linear, the activation function is the only source of nonlinearity in the standard reservoir computing setup. Introducing the nonlinear scaling factor a can be seen as simply tuning the scale for $\mathbf{W}_{in}$ and the spectral radius of $\mathbf{A}$ simultaneously.

4.3 Results

To investigate the effect of the controlled node removal, $N = 500$ random realizations of the reservoir $\mathbf{A}$ and $\mathbf{W}_{in}$ have been simulated. The parameters are those obtained

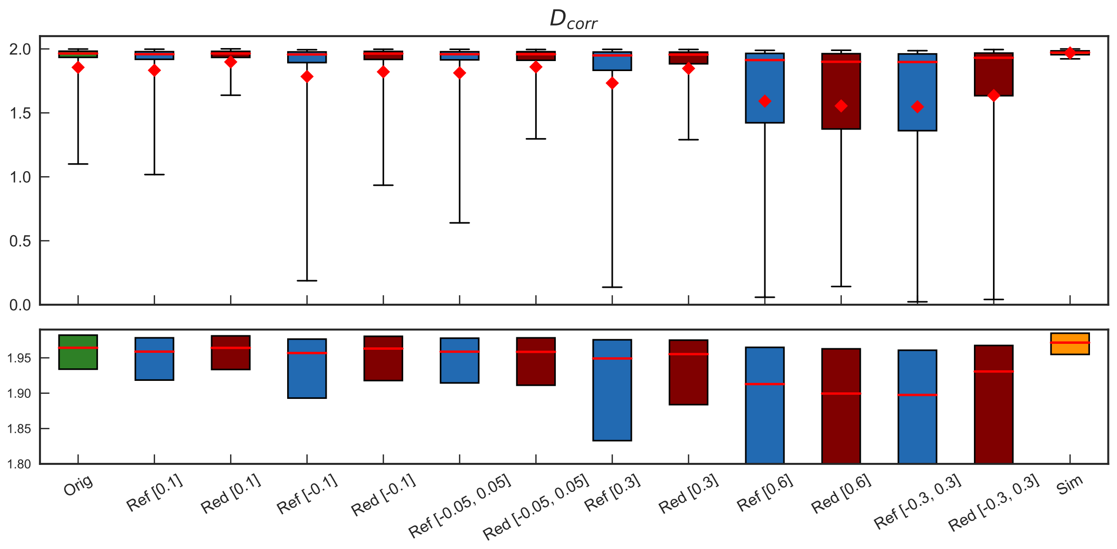

Figure 4.2: Boxplot of the correlation dimension for $N = 500$ realizations for the original setup (green – left) and different percentages of nodes removed. Positive numbers (e.g. Red [0.1]) represent a removal of the 10% largest $\mathbf{W}_{out}$ nodes, while negative numbers (e.g. Red [-0.1]) denote a removal of the 10% smallest $\mathbf{W}_{out}$ nodes. Consequently, Red [-0.05,0.05] stands for the nodes with the 5% largest and smallest $\mathbf{W}_{out}$ values removed symmetrically. *Red* are the results for the system after node removal, while *Ref* represents a smaller reference network. The yellow box on the right represents the error of the correlation dimension calculated from $N = 500$ simulated trajectories. The boxes represent the 25%–75% percentile range while the extended lines denote the 5% and 95% percentile, respectively. Red bars are indicating the mean values and red dots show the median. In order to make a comparison easier, the bottom plot gives a zoomed in view of the 25%–75% percentile boxes and the respective median values.

from the hyperparameter optimization explained above, while the nonlinear scaling factor is set to $a = 1$. Figure 4.2 shows a boxplot representing the distribution of the correlation dimension of the different random realizations for different levels of node removal. The green box on the left represents the results for the original network before node removal, whereas the yellow box on the right quantifies the error of the correlation dimension calculation based on $N = 500$ simulated trajectories by using the Lorenz equations with different initial conditions. The labels are defined the following way: *Red [x]* denotes the results for the reduced system after removing the nodes corresponding to the largest x% of the output weights if $x > 0$ and smallest x% if $x < 0$. Both positive and negative values at the same time mean that nodes are symmetrically removed from both ”sides”. In contrast, the results shown for the *Ref [x]* labels are reference reservoirs, which are initially constructed and trained

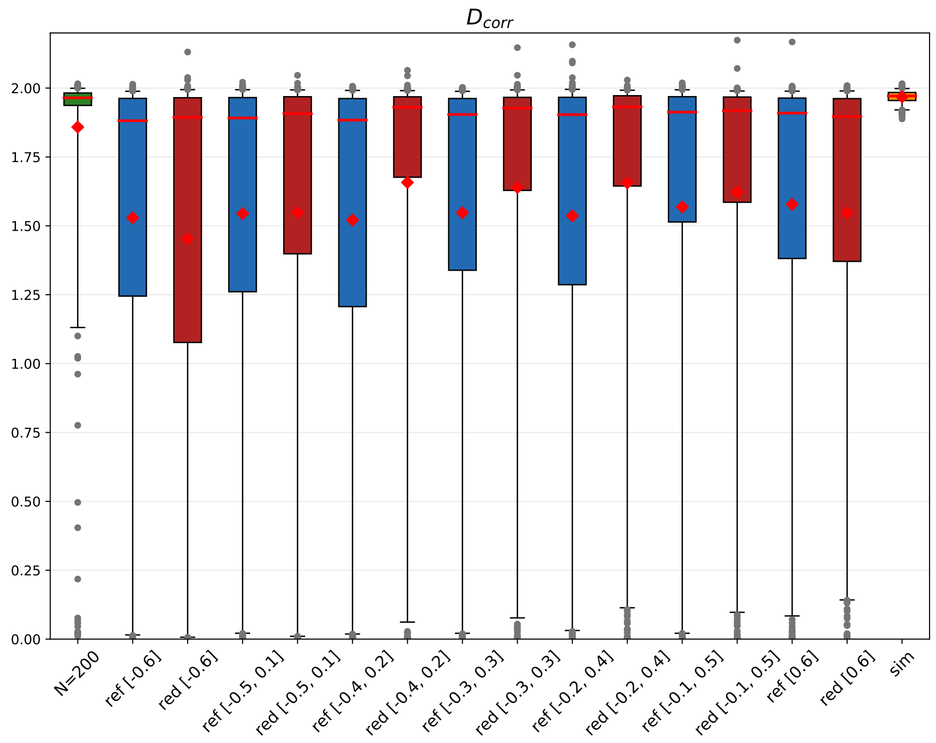

Figure 4.3: Boxplot of the correlation dimension for $N = 500$ realizations for the original setup (green – left) and different percentages of nodes removed. Positive numbers (e.g. Red [0.6]) represent a removal of the 60% largest $\mathbf{W}_{out}$ nodes, while negative numbers (e.g. Red [-0.6]) denote a removal of the 60% smallest $\mathbf{W}_{out}$ nodes. Consequently, Red [-0.3,0.3] stands for the nodes with the 30% largest and smallest $\mathbf{W}_{out}$ values removed symmetrically. *Red* are the results for the system after node removal, while *Ref* represents a smaller reference network. The yellow box on the right represents the error of the correlation dimension calculated from $N = 500$ simulated trajectories. The boxes represent the 25%–75% percentile range while the extended lines denote the 5% and 95% percentile, respectively. Red bars are indicating the mean values and red dots show the median

with the same number of nodes as the reduced systems and calibrated to the same spectral radius that the respective reservoirs have after the node removal procedure. The results show that removing the nodes corresponding to the largest 10% of the output weights – *Red [0.1]* – improves the prediction quality in terms of resembling the correlation dimension compared to the original setup – *Orig.* In particular, the mean of the correlation dimension improves from 1.85 to 1.89, whereas the median stays at 1.96. The values of the simulated system are 1.97 and 1.97, respectively. However, a stronger effect can be observed for bad predictions as denoted by the

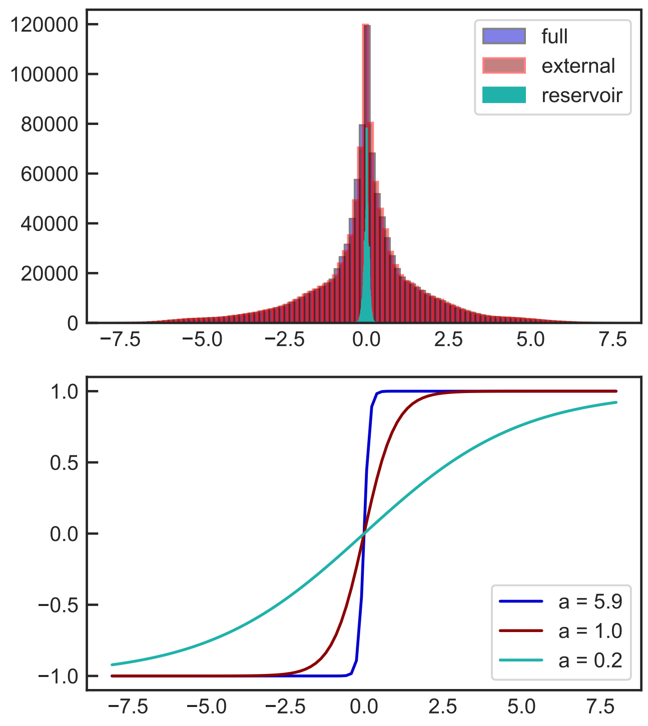

Figure 4.4: Top plot: Distribution of the arguments of the activation function during training period split into the contribution from the reservoir (green), the input term (red) and total (blue). Bottom plot: Hyperbolic tangent for different nonlinear scaling factors - x-axis represents values of its argument and y-axis the function value.

lower bars in the top plot. Here, the 5% percentile significantly increases from around 1 to 1.6 indicating a lower number of outliers, where the predictions did not resemble the statistical climate well. Comparing the reduced reservoir after node removal with now only $N = 180$ nodes to a reservoir initially setup and trained with $N = 180$ nodes shows that the improved prediction quality is not due to the changed reservoir size but driven by the altered properties of the system. In contrast, just reducing the number of nodes leads to more outliers and thus worse reproduction of the correlation dimension. If instead the nodes corresponding to the smallest 10% of the output weights are removed, again the prediction quality deteriorates. This is contrary to the intuitive expectation that small output weights point to unimportant nodes and thus removing them would be beneficial. Finally, a symmetrical removal of the smallest and largest 5% slightly improves prediction quality. Removing even larger amounts of nodes generally increases prediction variability. However, the results for excluding the largest 30% indicate comparable prediction quality to the original setup and

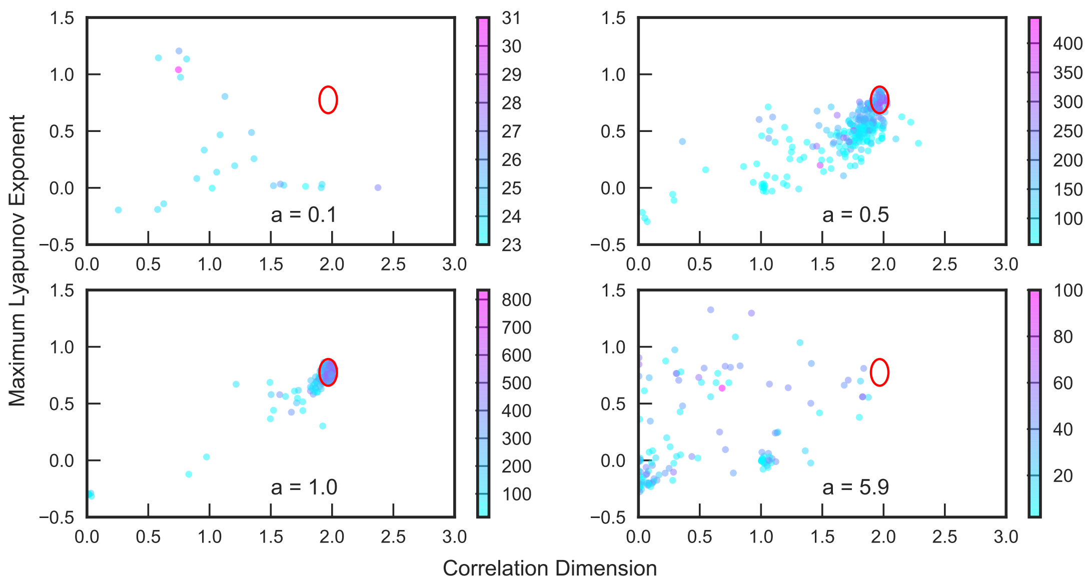

Figure 4.5: Largest Lyapunov exponent scattered against correlation dimension for different values of the nonlinear scaling parameter a based on $N = 300$ realizations each. The colors denote the forecast horizon of the predictions and the red ellipses show the three σ errors of the correlation dimension ($\sigma = 0.024$) and the largest Lyapunov exponent ($\sigma = 0.039$) calculated from simulations of the actual system.

thus may add value in applications where a downscaled system is beneficial due to computational challenges, e.g. when dimensionality is high or in the case of hardware implementations of reservoir computing, such as neuromorphic computing [59–61]. If in the latter case even more downscaling is necessary, Fig. 4.3 shows how removing 60% and therefore a large part of the nodes affects the outcome. The results suggest that in those cases where much downscaling is needed, an asymmetrical removal of nodes from both "sides" leads to the best results. In this case this is achieved by removing the nodes corresponding to the 40% smallest and 20% largest output weights. Although there are significantly more outliers compared to the original setup, the median is still comparably close to the original setup.

In the following, the effect of varying the nonlinear scaling parameter a is investigated. As shown in the bottom plot of Fig. 4.4, the profile of the hyperbolic tangent is such that it is saturating for large positive or negative values and approximately linear for small values. The nonlinear scaling factor a thus controls when the saturation regime begins and consequently the nonlinearity of the response. If it is chosen small,

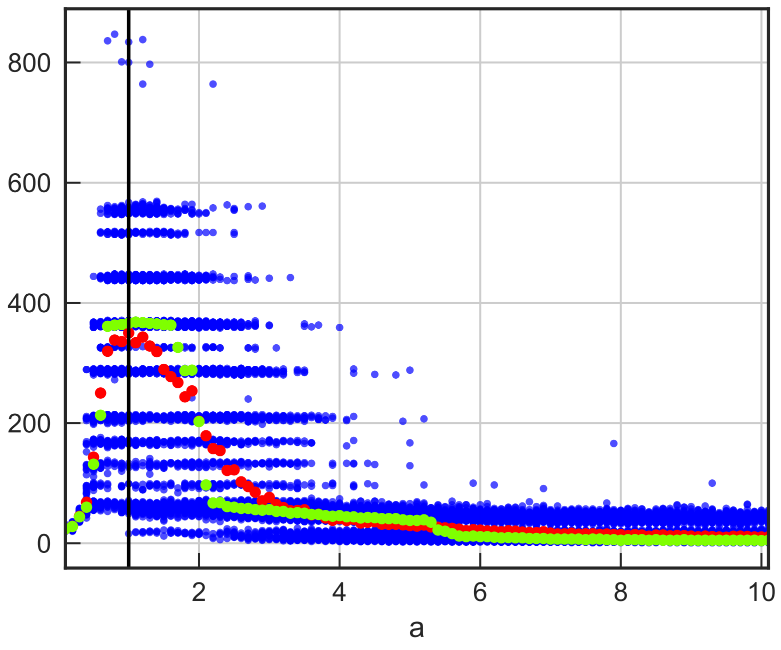

Figure 4.6: Forecast horizon of the modified Lorenz system plotted for different values of the nonlinear scaling parameter a with $N = 300$ realizations for each a (blue). Furthermore, the average (red) as well as the median (green) value are shown for each value of a. The black horizontal line marks the optimal choice for a.

e.g. $a = 0.2$ as denoted by the green line in the bottom plot, the distribution of the input argument lies predominantly within the linear regime of the hyperbolic tangent. The other way around, for large choices like $a = 5.9$ most inputs are located in the saturation regime. Here, the value 5.9 comes from the hyperparameter optimization mentioned above which led to $\mathbf{W}_{in}$ scale $= 0.17$. In commonly used standard parameterizations the $\mathbf{W}_{in}$ scale is 1 – approximately it can be stated that $\mathbf{W}_{in}$ scale of 0.17 in this setup with $a = 5.9$ is equivalent to $\mathbf{W}_{in}$ scale of 1, ignoring the comparably small influence of the reservoir term. Figure 4.5 shows the results based on $N = 300$ random realizations of the reservoir for different values of a. As one would intuitively expect, the above mentioned extreme cases lead to a bad reproduction of the statistical climate of the Lorenz system. In both cases, not a single point is located inside the three sigma error ellipse. This is in particular an interesting result for the commonly used standard setup. Results become significantly better when the choice for a moves towards $a = 1$, where the majority of points is within the error ellipse. Hence it can be stated that a reasonable choice for a plays

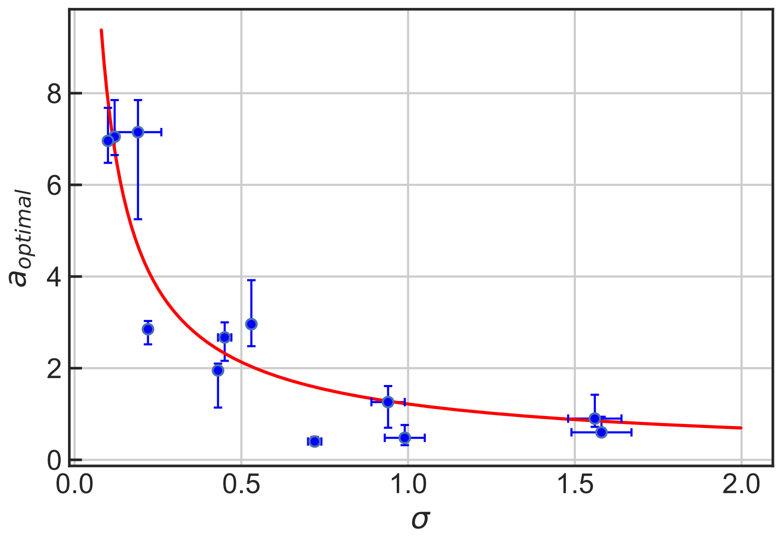

Figure 4.7: Blue dots: Optimal value for the nonlinear scaling parameter a – based on the maximum of the average forecast horizon over all realizations for a given a – plotted against the standard deviation σ of the input data. The vertical bars denote the range of values for a, where the average forecast horizon is up to 10% lower than for the optimal a, while the horizontal bars represent the standard error. The red line represents a fit through the points. Dynamical systems from left to right: Rabinovich-Fabrikant equations, Chua circuit, Complex Butterfly attractor, Thomas' cyclically symmetric attractor, Rössler system, Rucklidge system, Halvorsen model, *blended system:* 0.4*Modified Lorenz system + Halvorsen Model, *blended system:* 0.5*Modified Lorenz system + 3*Rabinovich-Fabrikant equations, *blended system:* Rössler + 2*Rucklidge system, Modified Lorenz system and Chen system

a key role for prediction quality and variability. In addition, Fig. 4.5 also suggests that for reasonable choices of a, realizations with a well resembled statistical climate coincide with longer forecast horizons. To confirm the optimal value for a, again $N = 300$ random realizations for different values of a between 0 and 10 are analyzed. The results are shown in Fig 4.6, where the blue points correspond to the forecast horizon of the single realizations. In addition, the red and green dots represent the average and median value across all realizations for a given value of a. The optimal value for a is then chosen such that the average over all realizations is maximized. This leads to a value around $a = 1.0$, as one would expect given the hyperparameter optimization at the beginning. Moreover, this also agrees with the findings in Fig 4.5 and confirms that nonlinearity in the activation function is essential for predicting complex nonlinear systems.

In addition to the Lorenz system, the same analysis is done for other nonlinear complex systems such as the Chua circuit, the Rössler system and other autonomous dissipative flows as summarized in section 4.2. Figure 4.7 shows the results for their optimal values of a scattered against the standard deviation of the input. In addition, combinations of the nonlinear systems are constructed in order to fill the gap in between the standard deviations of the Halvorsen model (0.53) and the modified Lorenz system (1.56). The results indicate a strong relationship between the optimal choice for a and the input standard deviations, which intuitively makes sense given the above discussion of Fig. 4.4. Surprisingly, this seems to dominate effects of other system-specific properties. As a rule of thumb, the optimal value for the nonlinear scaling parameter can be derived as $a_{opt} = c/\sigma(\mathbf{W}_{in}\mathbf{u})^b$ with $b = 0.80$ and $c = 1.22$ determined by the fitted red curve. This provides a good starting point for the hyperparameter optimization. However, it is always recommended to run a system specific analysis as shown in Fig 4.6. Equivalent results for a_{opt} are obtained by carrying out the same analysis using correlation dimension as an evaluation measure instead of forecast horizon.

4.4 Conclusions

This chapter investigated the influence of alterations to the reservoir network structure by carrying out a controlled node removal. This led to two main insights: First, removing nodes corresponding to the largest 10% of the output weights improves prediction quality and reduces outliers. This is somewhat counter-intuitive, as large weights in the output function suggest a strong influence of the respective node in the construction of the output signal. Second, the network size can be reduced by more than 30% at comparable prediction quality and up to 60% while still delivering reasonable performance. This could be helpful for hardware implementations or improved computational efficiency for high dimensional applications of reservoir computing.

Furthermore, the influence of tuning the nonlinearity of the hyperbolic tangent activation function was investigated by introducing a nonlinear scaling factor. A reasonable choice for the scaling factor reduces variability significantly and leads to better predictions. Moreover, a relationship between the optimal choice of the scaling factor and the standard deviation of the input was found, thus providing a good starting point for a complete hyperparameter optimization. Overall, the results demonstrate that a large optimization potential lies in a systematical refinement of the differential reservoir properties for a given data set.

Chapter 5

A flexible control mechanism for nonlinear dynamical systems

This chapter is based on the following paper, which is listed in this thesis as Ref [62].

> A. Haluszczynski and C. Räth, "Controlling nonlinear dynamical systems into arbitrary states using machine learning", Scientific Reports **11**, 1–8 (2021)

5.1 Objectives

With robust and high quality predictions at hand, far-reaching perspectives open up for applications in dynamical systems. In particular, the control of nonlinear dynamical systems is a key task in many different areas of science and engineering. While the possibility of controlling chaotic systems has been a remarkable discovery [10, 11, 63], previous approaches, however, require knowledge of the underlying equations or large data sets as they rely on phase space methods. Yet, for many real-world applications, the equations are not known or only limited data is available. Exploiting the improved prediction capabilities of reservoir computing as shown in Chapter 3 and Chapter 4, a novel and fully data driven approach is introduced in the following, which generalizes control techniques of chaotic systems. In this way, the system can be brought not only into periodic states, but even very accurately into intermittent and different chaotic behavior. This goes well beyond existing approaches that so far only managed to break up periodic or synchronized motion [64, 65] through 'chaotification' but did not allow to control the system into well-specified, yet more complex target states.

5.2 Methods

The proposed control mechanism relies on a machine learning based prediction of the desired dynamics of the system. To obtain robust predictions, a reservoir computer with quadratic readout is used as introduced in Section 2.1. The term *control* is defined the following way: A dynamical system with trajectory $\mathbf{u}$ originally is in state $\mathbf{X}$, which may represent e.g. periodic, intermittent or chaotic behavior. Then, the system behavior changes into another state $\mathbf{Y}$ as a consequence of order parameter changes or some uncontrollable external force. The aim is now to control the system back into its original state $\mathbf{X}$, while the cause for the initial change in state is still present. This can be achieved by deriving a suitable control force $\mathbf{F}(t)$, which is applied while the system is still in state $\mathbf{Y}$. Deriving $\mathbf{F}(t)$ requires the knowledge of how the trajectory $\mathbf{u}(t)$ would have evolved if the system was still in state $\mathbf{X}$ instead. This 'what if' scenario can be obtained by reservoir computing or in general any other machine learning approach that is capable to predict the dynamics well and is denoted as $\mathbf{v}(t)$. Then, the control force $\mathbf{F}(t)$ is derived as

$$\mathbf{F}(t) = K(\mathbf{u}(t) - \mathbf{v}(t)) \ , \tag{5.1}$$

with K scaling the magnitude of the force. Therefore, the force only depends on the distance between the measured coordinates $\mathbf{u}(t)$ and the machine learning prediction of the desired dynamics $\mathbf{v}(t)$. The control mechanism is investigated based on the Lorenz and Rössler system as introduced in Chapter 3.2. Here, different dynamical states $\mathbf{X}$ and $\mathbf{Y}$ correspond to different parameter sets $\boldsymbol{\pi}$ and $\boldsymbol{\pi}^*$. Thus, if the parameters changed now to $\boldsymbol{\pi}^*$ and, as a consequence, the system exhibits the unwanted dynamical state $\mathbf{Y}$, it can be forced back into $\mathbf{X}$ by applying the control force $\mathbf{F}(t)$

$$\mathbf{u}(t + \Delta t) = \int_t^{t+\Delta t} (\dot{f}(\mathbf{u}(\tilde{t}), \boldsymbol{\pi}^*) + \mathbf{F}(\tilde{t}))d\tilde{t} \ , \tag{5.2}$$

where $\dot{f}$ denotes the time derivatives of the respective system. Since this example is based on a simulation, the known equations are used to simulate the trajectories. In a real world application where the equations are not known, $\mathbf{u}(t)$ would be measured instead of simulated and $\mathbf{F}(t)$ would be directly applied to the system. To evaluate if the controlled dynamics matches the desired state, the correlation dimension and largest Lyapunov exponent are used as evaluation measure similarly to the previous chapters.

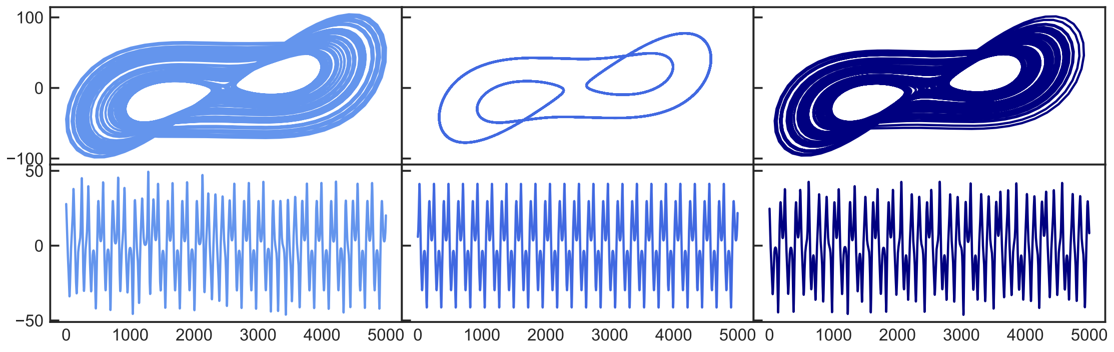

Figure 5.1: Periodic to chaotic control. Top: 2D attractor representation in the x-y plane. Bottom: X coordinate time series. Left plots show the original chaotic state which changes to a periodic state (middle) after tuning the order parameter. After applying the control mechanism, the system is forced into a chaotic state again (right).

5.3 Results

Figure 5.1 shows the results for the Lorenz system initially (left side) being in a chaotic state $\mathbf{X}$ ($\boldsymbol{\pi} = [\sigma = 10.0, \rho = 167.2, \beta = 8/3]$), which then changes to periodic behavior $\mathbf{Y}$ (middle) after ρ is changed to $\rho = 166$. The scaling constant is empirically set to $K = 25$. It has been found that the method works for a wide range of choices and thus an optimization was not needed at that stage. In the left plot the control scheme is activated and the attractor again looks chaotic and similar to the original one. Therefore, the method is able to control a periodic state into a well-defined chaotic target state. For an additional quantitative assessment, $N = 100$ random realizations of the system are evaluated in terms of correlation dimension and largest Lyapunov exponent. The first line in Table 5.1 confirms that the control mechanism robustly re-establishes the statistical climate of the desired state. The next step is to investigate if also intermittent dynamics can be successfully forced, in particular when coming from in this case undesired chaotic dynamics and thus involving complex initial as well as target states. This situation is shown in Figure 5.2. Here, intermittent dynamics correspond to parameters $\boldsymbol{\pi} = [\sigma = 10.0, \rho = 166.15, \beta = 8/3]$. Then, ρ is changed to $\rho = 167.2$ resulting in a chaotic state (middle plots). Again, it can be seen that the control succeeds. In particular, the lower plots indicate that the controlled system in the left plot exhibits the same characteristics — periodic behavior interrupted by irregular bursts — as the right plot representing the original dynamics. Also in this case the statistics in Table 5.1 confirm the observation. It is remarkable that bursts do not seem to occur more often given the chaotic dynamics

Table 5.1: Statistical simulation over $N = 100$ random realizations of the systems evaluated in terms of the mean values of the largest Lyapunov exponent and the correlation dimension with corresponding standard deviations. The subscript *orig* denotes the initial state of the system, while *changed* refers to the new state after parameters changed and *controlled* means the system controlled back into the original state. The description left to the arrow is the original state that also will be achieved again after controlling the system whereas the state written right to the arrow corresponds to the changed condition.

	Largest Lyapunov Exponent λ			Correlation Dimension ν		
	λ_{orig}	$\lambda_{changed}$	$\lambda_{controlled}$	ν_{orig}	$\nu_{changed}$	$\nu_{controlled}$
$Periodic \rightarrow Chaotic$	0.851 $\pm$0.070	0.080 $\pm$0.075	0.841 $\pm$0.074	1.700$\pm$0.065	1.052 $\pm$0.071	1.700 $\pm$0.061
$Chaotic \rightarrow Intermittent$	0.571 $\pm$0.096	0.853 $\pm$0.053	0.614 $\pm$0.101	1.321 $\pm$0.086	1.678 $\pm$0.055	1.351 $\pm$0.091
$Chaotic_B \rightarrow Chaotic_A$	0.479 $\pm$0.060	0.643 $\pm$0.075	0.478 $\pm$0.067	1.941 $\pm$0.038	1.948 $\pm$0.047	1.933 $\pm$0.040
$Chaotic_D \rightarrow Chaotic_C$	0.819 $\pm$0.092	0.884 $\pm$0.058	0.822 $\pm$0.052	1.855 $\pm$0.069	1.959 $\pm$0.037	1.866 $\pm$0.050
$Periodic \leftarrow Chaotic$	-0.003 $\pm$0.012	0.844 $\pm$0.059	0.028 $\pm$0.110	1.001 $\pm$0.065	1.700 $\pm$0.071	1.001 $\pm$0.061
$Chaotic \leftarrow Intermittent$	0.851 $\pm$0.070	0.550 $\pm$0.094	0.828 $\pm$0.067	1.700 $\pm$0.086	1.326 $\pm$0.055	1.698 $\pm$0.091
$Chaotic_B \leftarrow Chaotic_A$	0.629 $\pm$0.069	0.446 $\pm$0.068	0.629 $\pm$0.066	1.948 $\pm$0.037	1.939 $\pm$0.049	1.956 $\pm$0.037
$Chaotic_D \leftarrow Chaotic_C$	0.881 $\pm$0.092	0.836 $\pm$0.058	0.880 $\pm$0.052	1.958 $\pm$0.069	1.864 $\pm$0.038	1.951 $\pm$0.050

of the underlying equations and parameter setup in the left plot. Instead, the control works so well that it exactly enforces the desired dynamics. Finally, the goal is to control from one specific chaotic state to another specific chaotic state. Two examples are shown in Fig. 5.3 and Fig. 5.4. In the first, initial chaotic state $Chaotic_A$ corresponds to parameters $\boldsymbol{\pi} = [\sigma = 10.0, \rho = 28.0, \beta = 8/3]$. When changing ρ to $rho = 50.0$ a different chaotic attractor $Chaotic_B$ is obtained. In the latter, $Chaotic_A$ corresponds to $\boldsymbol{\pi} = [\sigma = 10.0, \rho = 102.0, \beta = 8/3]$ leading to $Chaotic_B$ after this time σ is changed to $\sigma = 20.0$. The examples aim at two different effects: In the first case, the size of the attractor significantly increases after the parameter change, while in the second case the attractor takes on a different shape. The latter is particularly visible in the different dynamics of the X coordinate shown in the lower plots of Fig. 5.4. However, both examples have in common that the control mechanism works excellently. It is visible in Fig. 5.3 that besides the dynamics also the original size of the attractor is restored. Also for the chaos-to-chaos control, the exemplary observations are confirmed by the statistics in Table 5.1.

Furthermore, the bottom half of the table shows the results for the opposite direction for each example given above. Therefore, $Periodic \rightarrow Chaotic$ in the upper half of the table means, that an initially chaotic system changed into a periodic state and then gets controlled back into its initial chaotic state. In contrast, $Periodic \leftarrow Chaotic$ in the lower half now means that the system initially is in the periodic state. It then shows chaotic behavior after the parameter change and finally is controlled back into the original periodic state - thus the opposite direction as above. It is evident that all cases also succeed in the opposite direction. This supports the claim that the prediction based control mechanism works for arbitrary states. To

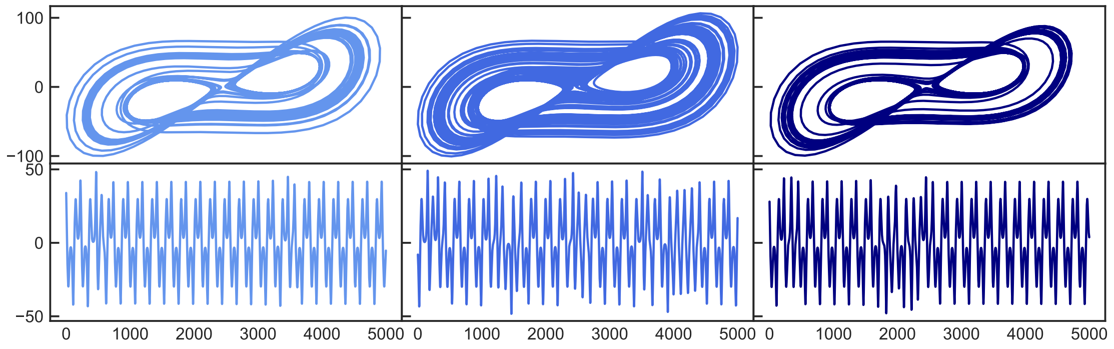

Figure 5.2: Chaotic to intermittent control. Top: 2D attractor representation in the x-y plane. Bottom: X coordinate time series. Left plots show the original intermittent state which changes to a chaotic state (middle) after tuning the order parameter. After applying the control mechanism, the system is forced into an intermittent state again (right).

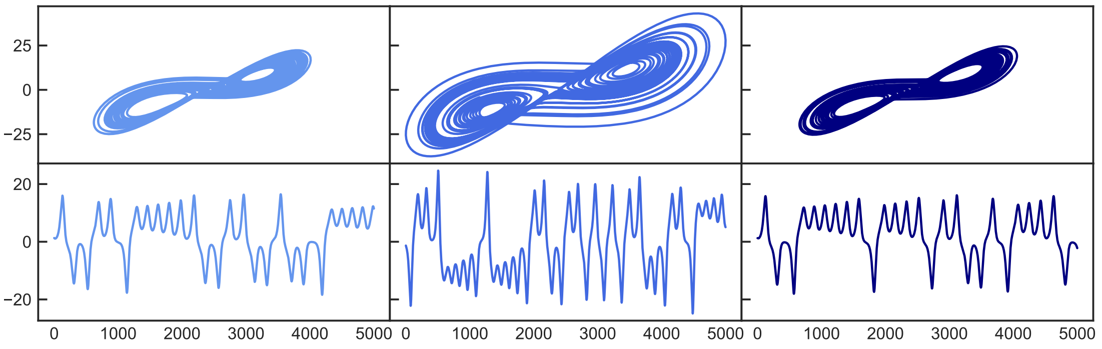

Figure 5.3: Chaotic to chaotic control. Top: 2D attractor representation in the x-y plane. Bottom: X coordinate time series. Left plots show the original chaotic state which changes to a different chaotic state (middle) after tuning the order parameter ρ. After applying the control mechanism, the system is forced into the initial chaotic state again (right).

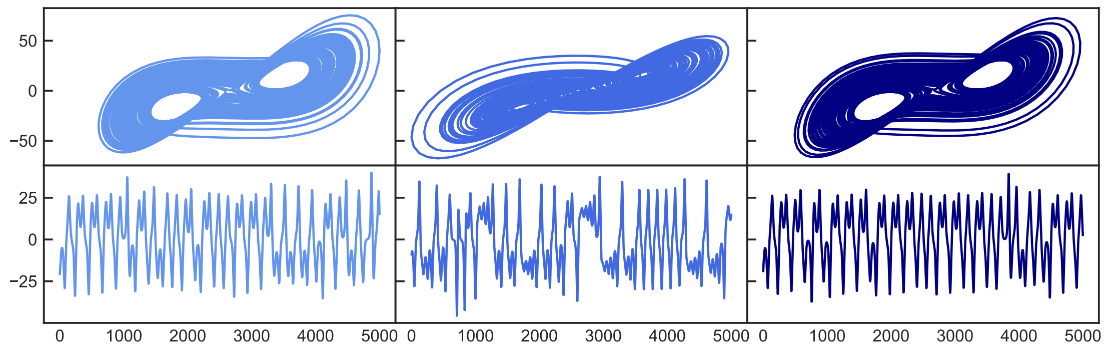

Figure 5.4: Chaotic to chaotic control. Top: 2D attractor representation in the x-y plane. Bottom: X coordinate time series. Left plots show the original chaotic state which changes to a different chaotic state (middle) after tuning the order parameter σ. After applying the control mechanism, the system is forced into the initial chaotic state again (right).

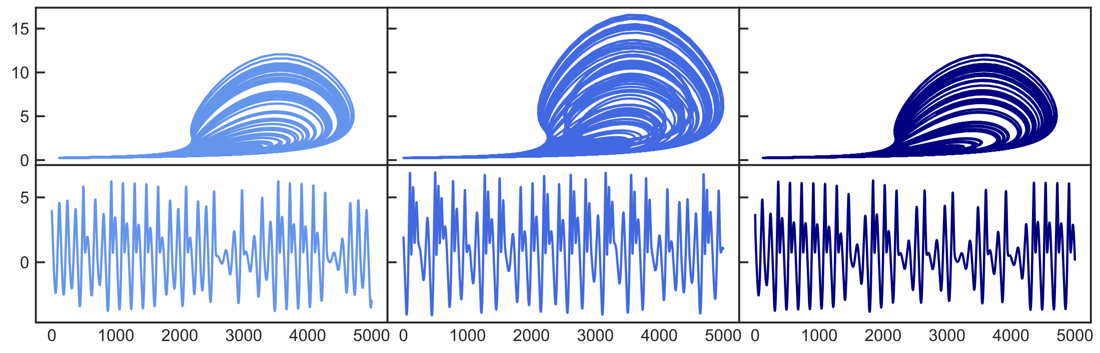

Figure 5.5: Chaotic to chaotic control for the Rössler system. Top: 2D attractor representation in the x-z plane. Bottom: X coordinate time series. Left plots show the original chaotic state which changes to a different chaotic state (middle) after tuning the order parameter. After applying the control mechanism, the system is forced into the initial chaotic state again (right).

verify that this claim does not only hold for the Lorenz system, the same approach of controlling one chaotic state to another chaotic state is tested with the Rössler system. Parameters $\boldsymbol{\pi} = [a = 0.5, b = 2.0, c = 4.0]$ correspond to the state $Chaotic_A$, changing to $Chaotic_B$ after a is changed to $a = 0.55$. It becomes clear in Fig. 5.5 that the control mechanism is successful again.

5.4 Conclusions

In this chapter, a flexible new approach for the control of nonlinear dynamical systems has been presented. In contrast to existing methods, where either knowledge of the underlying equations or large amount of data for phase space methods is required, it is fully data driven and only relies on limited data. With sufficiently good machine learning predictions at hand, it has been shown that the systems can be controlled to different dynamical target states, even from one chaotic state to another specific chaotic state. This opens up numerous possible applications ranging from engineering to medicine. For example, a rocket engine is a nonlinear system that can exhibit critical combustion instabilities [66, 67]. Their onset could be detected in time using reservoir computing and the further course could be prevented with the help of the control mechanism by pushing the engine in a stable state by applying the control force via its pressure valves. Another example would be the development of personalized pacemakers, since the heart of a healthy human does not beat in a purely periodic fashion but rather shows features being typical for chaotic systems like multifractality [68] that vary significantly among individuals. However, pacing protocols developed so far aim at keeping the diastolic interval constant [69–72]. As reservoir computing can learn the individual dynamics, the control scheme could emulate the patient-specific full behavior of the heart in healthy conditions and therefore serve as personalized pacemaker. In conclusion, this machine learning enhanced method allows for an unprecedented flexible control of dynamical systems and has thus the potential solve a plethora of new real-world problems.

Chapter 6

System Causalities and Outlook

It has been shown in the previous chapters that reservoir computing is capable to accurately predict nonlinear dynamical and even chaotic systems in a fully data driven manner without knowledge of the underlying equations. Furthermore, it has been demonstrated that there lies a large optimization potential in a systematical refinement of the differential properties of the reservoir computing system. However, reservoir computing or machine learning methods in general can also be extended by incorporating knowledge about the system in the form of constraints, causal relations or even governing equations to build a so called *hybrid* system [33, 73–76]. Building on the concept of surrogate data [77–79] for the separation of linear and nonlinear effects [80] in time series, the causal structure of dynamical systems is analyzed using different causal inference techniques: Granger causality [81], convergent cross mapping [82] and transfer entropy [83, 84].

In order to separate linear and nonlinear effects, a linearized version of the time series of the respective system is needed. By Fourier transforming a time series $\boldsymbol{x}$, its linear properties are separated into the amplitudes and the nonlinear ones into the phases. Fourier transform (FT) surrogates [85] are then created by adding uniformly distributed numbers $\boldsymbol{\phi}_k$ to the phases and thus destroying the encoded nonlinear properties, while the linear properties in the amplitudes remain unchanged. As a result, the back-transformation is a linear version of the original time series:

$$\tilde{\boldsymbol{x}}^{(k)} = \mathcal{F}^{-1}\big\{\mathcal{F}\{\boldsymbol{x}\} \cdot e^{i\boldsymbol{\phi}_k}\big\} \, . \tag{6.1}$$

To investigate how the linear or nonlinear part of a time series affects another time series, one can construct surrogate based bivariate measures, where either all time series or only one time series is surrogated. A bivariate measure $\psi\big(\boldsymbol{x}, \boldsymbol{y}\big)$ is a function which maps two time series to a real number. Its corresponding surrogate measure is then defined as the average over K surrogate realizations of both time series:

$$\psi^{surro}\big(\boldsymbol{x}, \boldsymbol{y}\big) \equiv \frac{1}{K} \sum_{k=1}^{K} \psi\big(\tilde{\boldsymbol{x}}^{(k)}, \tilde{\boldsymbol{y}}^{(k)}\big) , \tag{6.2}$$

where the superscript k indicates that the same random phases were added to both time series. Linear correlation measures such as Pearson correlation only depend on the phase differences and therefore, linear cross dependencies are also conserved. A number of K different random phases is used in order to obtain stable results. Furthermore, *cross measures* are defined by only surrogating the first time series and leaving the second untouched

$$\psi^{cross}\big(\boldsymbol{x}, \boldsymbol{y}\big) \equiv \frac{1}{K} \sum_{k=1}^{K} \psi\big(\tilde{\boldsymbol{x}}^{(k)}, \boldsymbol{y}\big) , \tag{6.3}$$

or vice versa which is then called $\psi^{anti}\big(\boldsymbol{x}, \boldsymbol{y}\big)$. Now, nonlinearity measures can be derived as

$$\psi^{nlin} \equiv \frac{\max\big\{0, \psi - \psi^{surro}\big\}}{\psi} , \tag{6.4}$$

where negative nonlinearities are attributed to spurious effects and thus ruled out. By interchanging surro-, cross- and anti-measures, further nonlinearity measures can be derived. For the measure ψ, different causal inference techniques are analyzed. Granger causality (GC) [81] was one of the first causal inference approaches and tests whether the prediction error of the next time step of a time series $\boldsymbol{y}$ can be decreased by including the history of another time series $\boldsymbol{x}$. If this is the case, $\boldsymbol{x}$ is said to Granger cause $\boldsymbol{y}$. It compares the prediction error of the autoregression model

$$\hat{y}_t = \sum_{\tau=1}^{\tau_{max}} \alpha_\tau \cdot y_{t-\tau} + \epsilon_t \tag{6.5}$$

to an augmented model including the history of $\boldsymbol{x}$:

$$\hat{y}_t = \sum_{\tau=1}^{\tau_{max}} \alpha_\tau \cdot y_{t-\tau} + \beta_\tau \cdot x_{t-\tau} + \eta_t , \tag{6.6}$$

where α_τ and β_τ are coefficients at lag τ, and ϵ_t and η_t denote independent error terms. Relying on autoregression, GC is only capable of capturing linear causality. Therefore, it is well suited as a verification, since GC of a surrogated time series should be equal to GC of the original one. While GC is mostly used as a binary statistical hypothesis test [86], the strength of the causal coupling can be quantified using the following normalization:

$$\psi_{GC}\big(\boldsymbol{x}, \boldsymbol{y}\big) \equiv 1 - \max\left\{1, \left(\frac{\mathrm{RSS}_{aug}}{\mathrm{RSS}_{rest}}\right)^2\right\} \in [0,1] , \tag{6.7}$$

where RSS_{rest} and RSS_{aug} are the residual sum of squares (RSS) of the initial and augmented model, respectively. To capture also nonlinear effects, convergent cross mapping (CCM) [82, 87] is used as a measure in addition. It builds on Takens' theorem [88], which states that the full state-space can be reconstructed from a single delay embedded coordinate of the system, also called shadow manifold. Due to transitivity, two coordinates within one system can then be mapped to each other through neighboring states in their respective shadow manifolds - this enables a cross prediction. Hence if $\boldsymbol{x}$ causes $\boldsymbol{y}$, the prediction of the future $\hat{y}_t$ using the shadow manifold $\mathcal{M}_{\boldsymbol{x}}$ should be identical to the actual value y_t. As a next step, the prediction is extended from a single point to a time series. For this, both time series are divided into training and test sets, where the former are used to construct the shadow manifolds and the latter serve as benchmarks to evaluate the prediction performance. While CCM is originally defined as the Pearson correlation ρ between the prediction $\hat{\boldsymbol{y}}|\mathcal{M}_x$ and the test set of $\boldsymbol{y}$, the correlation distance $d = \sqrt{2(1-\rho)}$ is proposed as evaluation measures in order to make results more comparable to transfer entropy, which will be introduced shortly. This entire procedure is repeated for an increasing training set fraction, which delivers a series $\boldsymbol{d}$ consisting of D correlation distances. This series should theoretically converge to a maximum since the prediction is enhanced for finer resolutions of the shadow manifolds. To assess convergence, an algorithmic approach using overlapping sliding windows of $\boldsymbol{d}$ is introduced. The convergence is fulfilled if the standard deviation decreases continuously and falls below the preset threshold of 0.1. If $\boldsymbol{d}$ converges, the mean of its last 5 values is calculated in order to smooth outliers. If $\boldsymbol{d}$ does not converge, CCM causality is set to 0:

$$\psi_{CCM}(\boldsymbol{x}, \boldsymbol{y}) \equiv \begin{cases} \frac{1}{5}\sum_{i=1}^{5} d_{D-5+i} & \text{if } \boldsymbol{d} \text{ converges} \\ 0 & \text{else} \end{cases} \in [0,1]\,. \tag{6.8}$$

Finally, transfer entropy [83] is used as an alternative method for quantifying both linear and nonlinear dependencies. It has been proven that for Gaussian variables, Granger causality and transfer entropy are equivalent [89] and thus transfer entropy can be seen as information-theoretical extension of Granger causality which is also capable of detecting nonlinear causality. It represents the reduction of uncertainty on future values of $\boldsymbol{y}$ given the history of $\boldsymbol{y}$ due to the knowledge of past values of $\boldsymbol{x}$ and can be expressed as conditional mutual information:

$$\begin{aligned} \psi_{TE}(\boldsymbol{x}, \boldsymbol{y}) &\equiv I(\boldsymbol{y}; \boldsymbol{x}_{t-1:} \mid \boldsymbol{y}_{t-1:}) \\ &= H(\boldsymbol{y} \mid \boldsymbol{y}_{t-1:}) - H(\boldsymbol{y} \mid \boldsymbol{y}_{t-1:}; \boldsymbol{x}_{t-1:})\,, \end{aligned} \tag{6.9}$$

where $H(\boldsymbol{x} \mid \boldsymbol{y})$ is the conditional entropy

$$H(\boldsymbol{x} \mid \boldsymbol{y}) = \sum_{i,j} p(x_i, y_j) log(\frac{p(x_i, y_j)}{p(y_j)}), \tag{6.10}$$

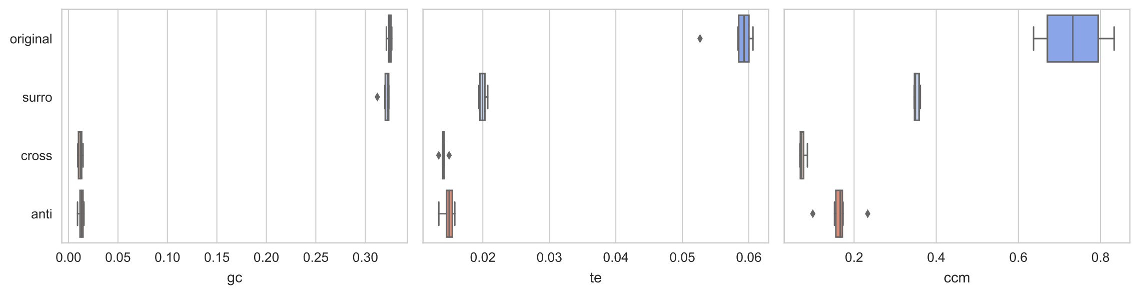

Figure 6.1: System causality box plots of the Lorenz system. The system means of the original-, surro, cross-, and anti-matrices for GC, TE, and CCM are computed, respectively. The sample consists of 50 simulations under the standard configuration. The surrogate-based measures are averaged over $K = 10$ surrogate realizations.

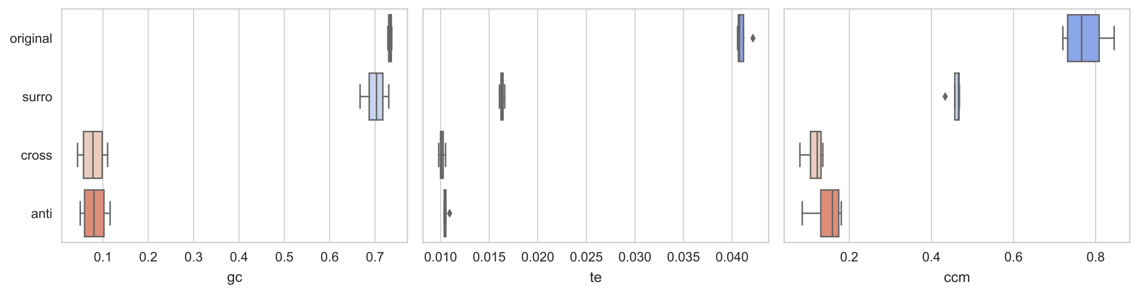

Figure 6.2: System causality box plots of the Halvorsen system. The same configurations as in previous figure are used.

with $p(x_i, y_j)$ being the probability that $\boldsymbol{x} = x_i$ and $\boldsymbol{y} = y_j$. The causality measures are tested on the Lorenz system with standard parameters $\sigma = 10$, $\rho = 28$, and $\beta = 8/3$. In addition, the Halvorsen system [90] is investigated, as it contains quadratic nonlinearities:

$$\frac{dx}{dt} = ax - 4y - 4z - y^2 \,, \tag{6.11}$$

where $a = 1.3$. The equations for y and z are constructed through rotating the three variables. If not specified otherwise, the differential equations are solved for $T = 10000$ steps and discretization $dt = 0.1$. The analysis on the Lorenz and Halvorsen attractors indicate that the system causality is mainly driven by nonlinear properties. This can be seen in Figures 6.1 and 6.2. The box plots show that both transfer entropy (TE) and convergent cross mapping (CCM) indicate significantly lower system causality for the surrogated time series compared to the original system time series. Since nonlinear properties are destroyed in the surrogated time series, these differences prove that nonlinearity play a significant role for causality, where approximately 70% of the measured transfer entropy and 50% of CCM can be attributed to nonlinear properties. Furthermore, the results for Granger causality

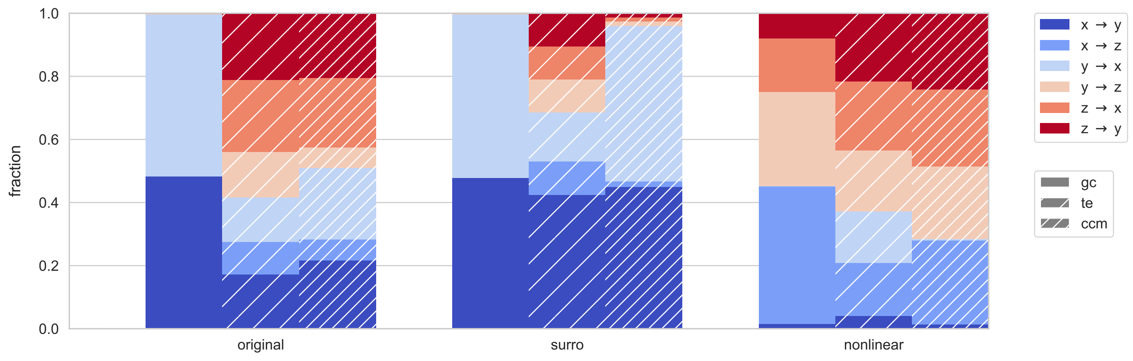

Figure 6.3: Causality decomposition of the Lorenz system. For GC, TE, and CCM the original-, surrogate-, and nonlinear- causality are computed, respectively. The contributions of the individual causal flows to the system causality are mapped by color, while the different inference techniques are indicated by white stripes. The individual fractions are averaged over 50 simulations under the standard configuration. The surrogate-based measures are averaged over $K = 10$ surrogate realizations.

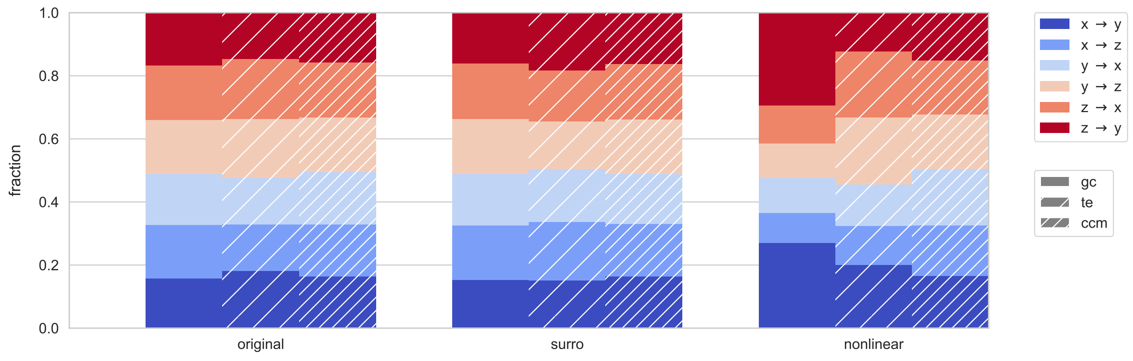

Figure 6.4: Causality decomposition of the Halvorsen system. The same configurations as in the previous figure are used.

(GC) confirm that it is restricted to capturing only linear causality, since original and surrogate based results are both on the same scale. The small deviations stem from the inaccuracies of the linear regression required for the calculation of GC. Furthermore, the newly introduced anti- and cross-causalities, which measure the causal flow from the linear properties from one time series to both the linear and nonlinear properties of another, vanish for all three inference methods. This further underpins the observation that causal flows are mainly dominated by nonlinearity.
On a finer scale, it becomes clear that the causal structure in terms of linear and nonlinear effects differs significantly among the coordinates of the Lorenz system, as illustrated in Fig. 6.3. The x and y pair is mainly driven by linear properties as it dominates the surrogate-causalities of GC and CCM - with both directions

contributing equal amounts. In contrast, the surrogate TE indicates that the direction x to y dominates the linear causality with a fraction of around 41%. This result is in line with the governing equations as the equation for x contains a linear contribution from y, while the equation for y contains a linear and nonlinear contribution from x. The remaining causal dependencies are split evenly across the respective flows. In comparison, all flows in the Halvorsen attractor contribute approximately equally to the system causality across all causality types and inference techniques, as shown in Fig. 6.4. This causal structure is expected due to the circulant nature of the governing equations.

Current work is exploring the derivation of the governing equations based on the system causality [91]. Feeding this information into the reservoir computing system could then restrict predictions to meaningful paths by constraining the allowed dynamics of the system and leveraging the additional information. Finding a suitable layout for this is subject to further research.

Publications

A Good and bad predictions: Assessing and improving the replication of chaotic attractors by means of reservoir computing

A. Haluszczynski and C. Räth, “Good and bad predictions: assessing and improving the replication of chaotic attractors by means of reservoir computing”, Chaos: An Interdisciplinary Journal of Nonlinear Science **29**, 103143 (2019)

Good and bad predictions: Assessing and improving the replication of chaotic attractors by means of reservoir computing

Cite as: Chaos **29**, 103143 (2019); https://doi.org/10.1063/1.5118725
Submitted: 04 July 2019 . Accepted: 09 October 2019 . Published Online: 25 October 2019

Alexander Haluszczynski, and Christoph Räth

COLLECTIONS

Paper published as part of the special topic on When Machine Learning Meets Complex Systems: Networks, Chaos and Nonlinear Dynamics
Note: This paper is part of the Focus Issue, "When Machine Learning Meets Complex Systems: Networks, Chaos and Nonlinear Dynamics."

Good and bad predictions: Assessing and improving the replication of chaotic attractors by means of reservoir computing

Cite as: Chaos **29**, 103143 (2019); doi: 10.1063/1.5118725
Submitted: 4 July 2019 · Accepted: 9 October 2019 ·
Published Online: 25 October 2019

View Online Export Citation CrossMark

Alexander Haluszczynski[1,2,a)] and Christoph Räth[3,b)]

AFFILIATIONS

[1] Department of Physics, Ludwig-Maximilians-Universität, Schellingstraße 4, 80799 Munich, Germany
[2] risklab GmbH, Seidlstraße 24, 80335 Munich, Germany
[3] Deutsches Zentrum für Luft- und Raumfahrt, Institut für Materialphysik im Weltraum, Münchner Str. 20, 82234 Wessling, Germany

Note: This paper is part of the Focus Issue, "When Machine Learning Meets Complex Systems: Networks, Chaos and Nonlinear Dynamics."
[a)] **Electronic mail:** alexander.haluszczynski@gmail.com
[b)] **Electronic mail:** christoph.raeth@dlr.de

ABSTRACT

The prediction of complex nonlinear dynamical systems with the help of machine learning techniques has become increasingly popular. In particular, reservoir computing turned out to be a very promising approach especially for the reproduction of the long-term properties of a nonlinear system. Yet, a thorough statistical analysis of the forecast results is missing. Using the Lorenz and Rössler system, we statistically analyze the quality of prediction for different parametrizations—both the exact short-term prediction as well as the reproduction of the long-term properties (the "climate") of the system as estimated by the correlation dimension and largest Lyapunov exponent. We find that both short- and long-term predictions vary significantly among the realizations. Thus, special care must be taken in selecting the good predictions as realizations, which deliver better short-term prediction also tend to better resemble the long-term climate of the system. Instead of only using purely random Erdös-Renyi networks, we also investigate the benefit of alternative network topologies such as small world or scale-free networks and show which effect they have on the prediction quality. Our results suggest that the overall performance with respect to the reproduction of the climate of both the Lorenz and Rössler system is worst for scale-free networks. For the Lorenz system, there seems to be a slight benefit of using small world networks, while for the Rössler system, small world and Erdös-Renyi networks performed equivalently well. In general, the observation is that reservoir computing works for all network topologies investigated here.

The application of machine learning techniques to various fields in science and technology yields very promising and fast advancing results. However, the robustness of these methods is a critical aspect that is often not adequately addressed. Particularly when trying to predict complex nonlinear systems—here by using a recurrent neural network based approach called reservoir computing—it is very useful to know how likely it is to end up with a "good" prediction and how different the results can be in terms of quality. In our context, a good prediction is achieved when the predicted trajectory matches those of the actual system in the short-term, while reproducing its statistical properties in the long-term. In order to thoroughly investigate the prediction quality, we run our analysis not only using a single prediction but on many realizations, which are based on the same parameters but different random number seeds. As a result, we find strong variability among the quality of the predictions, indicating that robustness seems to be an issue, and show the effect of varying the network topology of the reservoir.

I. INTRODUCTION

In the recent years, the use of machine learning (ML) techniques has not only become increasingly important in research but also popular in media, public perception, and businesses. The euphoric application in all possible areas, however, carries the risk of misinterpreting the results if deeper methodological knowledge is lacking. This is reminiscent of the situation in the late 1980s and early 1990s when chaos was a hot topic in the scientific community. In the absence of adequate statistical analysis, many systems have been erroneously categorized as being chaotic on the basis of, e.g., assessing the attractor dimensions by single measurements of short and noisy time series. Only after Theiler *et al.*[1] introduced the concept of surrogate data, the errors of the nonlinear measures for a given data set could be assessed, and it turned out that many claims of chaos had to be rejected. The lesson learned is that in the absence of a proper (linear) model of the underlying process, credible results can only be obtained by applying thorough statistical analyses involving averaging over a large number of realizations of simulations or surrogates. In recent years, the use of reservoir computing for quantifying and predicting the spatiotemporal dynamics of nonlinear systems has attracted much attention.[2–9] Many of the achievements—be it, e.g., the crossprediction of variables in two-dimensional excitable media[6] or the reproduction of the spectrum of Lyapunov exponents in lower dimensional (Lorenz or Rössler) and higher dimensional (Kuramoto-Sivashinsky) systems[2–4]—are impressive and guide the way to a range of new applications of ML in complex systems research. However, all results shown until now are based on a single or few realizations of reservoir computing. It is thus so far impossible to judge the robustness of the results on, e.g., variations of the set of random variables specifying the reservoir. Here, we perform the first thorough statistical analysis of predicting short- and long-term behavior of nonlinear time series by means of reservoir computing.

The heart of reservoir computing is—as the name already says—a so-called reservoir, which consists of D_r nodes that are sparsely connected with each other. The nodes are supposed to yield a proper "echo state" to a given input, which is then transferred to the output layer. This is why most types of reservoir networks are often called "echo state networks (ESN)."Beginning with the first introduction of ESNs by Maass and Jaeger,[10,11] the reservoir has typically been modeled as a random Erdös Rényi network, where two nodes are connected with a certain probability p. However, the groundbreaking works of Watts and Strogatz,[12] Albert and Barabási,[13] and many others have shown that random networks are far from being common in physics, biology, finance, or sociology. Rather, more complex networks like scale-free, small world, or intermediate forms of networks[14,15] with intriguing new properties are most often found in real world applications. Having this in mind, it seems natural to ask whether also for reservoir computing the topology of the network has a significant influence on the prediction results.[16] As a first step, we use the three aforementioned prototypical classes of networks as a reservoir and compare them regarding their ability of short- and long-term prediction of time series.

The paper is organized as follows: Sec. II introduces reservoir computing and the methods used in our study. In Sec. III, we present the main results obtained from the statistical analysis of the prediction results as well as studying different reservoir topologies. Our summary and the conclusions are given in Sec. IV.

II. METHODS

A. Lorenz and Rössler system

As in Pathak *et al.*[3] and Lu *et al.*,[8] we use the Lorenz system[17] as an example for replicating chaotic attractors using reservoir computing. It has been developed as a simplified model for atmospheric convection and it exhibits chaos for certain parameter ranges. The standard Lorenz system, however, is symmetric in x and y with respect to the transformation $x \to -x$ and $y \to -y$. This can be an issue, for example, when trying to infer the x and y dimension from knowledge of the z dimension as outlined in Lu *et al.*[18] In order to study a more general example, we would like to modify the Lorenz system such that this symmetry is broken. This can be achieved by adding the term x to the z-component, which then reads

$$\begin{aligned}\dot{x} &= \sigma(y - x),\\ \dot{y} &= x(\rho - z) - y,\\ \dot{z} &= xy - \beta z + x.\end{aligned} \tag{1}$$

We use the standard parameters $\sigma = 10, \beta = 8/3$, and $\rho = 28$. This system is referred to as a modified Lorenz system. The equations are solved using the 4th order Runge-Kutta method with a time resolution $\Delta t = 0.02$.

In addition to the Lorenz system, we ran the same analysis also on the Rössler system,[19] which equations read

$$\begin{aligned}\dot{x} &= -y - z,\\ \dot{y} &= x + ay,\\ \dot{z} &= b + z(x - c),\end{aligned} \tag{2}$$

where we use the parameters $a = 0.5$, $b = 2.0$, and $c = 4.0$. Again, this is a $D = 3$ dimensional chaotic system but said to be less chaotic than the Lorenz attractor. Thus, it is an interesting object to study, in particular, when it comes to the short-term prediction capabilities of reservoir computing. For the Rössler system, the time resolution is chosen to be $\Delta t = 0.05$ in order to ensure a sufficient manifestation of the attractor in the $t_{train} = 5000$ training time steps.

B. Reservoir computing

Reservoir computing is a machine learning technique that falls into the category of artificial recurrent neural networks. The core of the model is a network called "reservoir," which—in contrast to feedforward neural networks—exhibits loops. This means that past values feed back into the system and thus allow for dynamics.[20,21] In order to complete the task of predicting time series, the ability to capture dynamics is essential. Moreover, reservoir computing has a powerful advantage: While in other methods the network itself is dynamical, here the training is based only on the linear output layer and, therefore, allows for large reservoir dimensionality while still being computationally feasible.[8]

In our implementation, we stick to the setup used by Pathak *et al.*,[3] which works as follows. We have an input signal $\mathbf{u}(t)$ with dimension D that we would like to feed into a reservoir $\mathbf{A}$. The reservoir is chosen to be a sparse Erdös-Renyi random network with $D_r = 300$ nodes and $p = 0.02$.[22] Here, p describes the probability of connecting two nodes, which then leads to an unweighted average degree of $d = 6$. To obtain the weighted network, we then replace all nonzero elements of the adjacency matrix by independently and uniformly drawn numbers from $[-1, 1]$. It is important to highlight that the network itself is static and thus does not change over time. In order to feed the lower dimensional input signal $\mathbf{u}(t)$ into the reservoir, an $D_r \times D$ input function $\mathbf{W}_{in}$ is required. The entries of $\mathbf{W}_{in}$ are chosen here to be uniformly distributed random numbers within the range of the nonzero elements of the reservoir.

A key property of the system are its $D_r \times 1$ reservoir states $\mathbf{r}(t)$, which represent the scalar states of the nodes of the reservoir network. We initially assume $r_i(t_0) = 0$ for all nodes and update the reservoir states in each time step according to the equation

$$\mathbf{r}(t + \Delta t) = \alpha \mathbf{r}(t) + (1 - \alpha)\tanh(\mathbf{A}\mathbf{r}(t) + \mathbf{W}_{in}\mathbf{u}(t)). \tag{3}$$

As in Pathak *et al.*,[3] we set $\alpha = 0$ and, therefore, do not mix the input function with past reservoir states. Now, we have a fully dynamical system where the network edges are constant and the states of the nodes are time dependent.

The next step is to map the reservoir states $\mathbf{r}(t)$ back to the D dimensional output $\mathbf{v}$ given by

$$\mathbf{v}(t + \Delta t) = \mathbf{W}_{out}(\mathbf{r}(t + \Delta t), \mathbf{P}). \tag{4}$$

Here, we assume that $\mathbf{W}_{out}$ depends linearly on $\mathbf{P}$ and reads $\mathbf{W}_{out}(\mathbf{r}, \mathbf{P}) = \mathbf{P}\mathbf{r}$. This means that the output depends only on the reservoir states $\mathbf{r}(t)$ and the output matrix $\mathbf{P}$, which contains a large number of adjustable parameters—all its elements. Therefore, after acquiring a sufficient number of reservoir states $\mathbf{r}(t)$, we have to choose $\mathbf{P}$ such that the output $\mathbf{v}$ of the reservoir is as close as possible to the known real output $\mathbf{v}_R$. This process is called training. In general, the task is to find an output matrix $\mathbf{P}$ using Ridge regression, which minimizes

$$\sum_{-T \le t \le 0} \| \mathbf{W}_{out}(\mathbf{r}(t), \mathbf{P}) - \mathbf{v}_R(t) \|^2 - \beta \| \mathbf{P} \|^2, \tag{5}$$

where β denotes the regularization constant, which prevents from overfitting by penalizing large values of the fitting parameter. In this study, we choose $\beta = 0.01$. The notation, $\| \mathbf{P} \|$, describes the sum of the square elements of the matrix $\mathbf{P}$. For solving this problem, we are applying the matrix form of the Ridge regression,[23] which leads to

$$\mathbf{P} = (\mathbf{r}^T\mathbf{r} + \beta \mathbb{1})^{-1}\mathbf{r}^T\mathbf{v}_R. \tag{6}$$

The notion $\mathbf{r}$ and $\mathbf{v}_R$ without the time indexing denotes matrices where the columns are the vectors $\mathbf{r}(t)$ and $\mathbf{v}_R(t)$, respectively, in each time step. In our implementation, we chose $t_{train} = 5000$ training time steps while allowing for a washout or initialization phase of $t_{init} = 100$. During this time, we do not "record" the reservoir states $\mathbf{r}(t)$, which means that only 4900 time steps are used for the regression. In order to ensure that 100 time steps washout is sufficient, we also ran the analysis with 1000 time steps washout and found both results to be equivalent.

After $\mathbf{P}$ is determined, we can now switch to the prediction mode by giving the predicted state $\mathbf{v}(t)$ as input instead of the actual data $\mathbf{u}(t)$. The update equation for the network states $\mathbf{r}(t)$ then reads

$$\begin{aligned}\mathbf{r}(t + \Delta t) &= \tanh(\mathbf{A}\mathbf{r}(t) + \mathbf{W}_{in}\mathbf{W}_{out}(\mathbf{r}(t), \mathbf{P})) \\ &= \tanh(\mathbf{A}\mathbf{r}(t) + \mathbf{W}_{in}\mathbf{P}\mathbf{r}(t)).\end{aligned} \tag{7}$$

This allows us to produce a predicted trajectory of any length by just applying Eq. (4).

C. Alternative network topologies

So far, it has been standard practice to use purely random Erdös-Renyi networks for the reservoir $\mathbf{A}$. However, there is a variety of conceivable network topologies that may have an influence on the results. In this study, we investigate the use of "small world"[12] and "scale-free"[24] networks as an alternative.

Small world networks are graphs where the distance—in terms of steps via other nodes—between any pair of nodes is small. At the same time, the clustering coefficient is relatively high, which means that neighboring nodes tend to be connected. This so-called "small world property" is observed in many real world networks such as social networks, electric power grids, chemical reaction networks, and neuronal networks.[25] In order to have the same average degree $d = 6$ as the random Erdös-Renyi networks, we construct the small world networks in the following way: First, we connect each node with its six nearest neighbors implying periodic boundary conditions. This is equivalent to arranging all nodes as a ring. Then, we loop over each edge $x - y$ and rewire it to $x - z$ with probability $p = 0.2$, where node z is randomly chosen.

Scale-free networks are graphs where the distribution of the number of edges per node decays with a power-law tail. This is again a property, which is observed in many real world networks. For example, the abovementioned electric power grid networks and neuronal networks exhibit not only the small world property but are also scale free. Other examples include the World Wide Web or networks of citations of scientific papers.[25] Again, the network is constructed such that its average degree is $d = 6$. For this, we use the scale-free graph generator of the "NetworkX" package[26] with parameters $\alpha = 0.285, \beta = 0.665, \gamma = 0.05, \delta_{in} = 0.2, \delta_{out} = 0$. Note that in this case, the graph is directed, while the other two network topologies result in undirected graphs. Here, α sets the probability of adding a new node, which is connected to an already existing node, which is chosen randomly according to the in-degree distribution, while γ does the same except that the node is chosen according to the out-degree distribution. In addition, β regulates the probability of creating an edge between two existing nodes where one is chosen according to the in-degree distribution and the other node according to the out-degree distribution.[27]

D. Measures of the system

In order to assess the quality of the prediction, we are mainly using three different measures. The goal is to adequately address both the exact short-term prediction as well as the long-term reproduction of the statistical properties of the system—its so-called climate.

1. Forecast horizon

To quantify the quality and duration of the exact prediction of the trajectory, we use a fairly simple measure, which we call "forecast horizon." Here, we track the number of time steps during which the predicted and the actual trajectory are matching. As soon as one of the three coordinates exceeds certain deviation thresholds, we consider the trajectories as not matching anymore. Throughout our study, we use

$$|\mathbf{v}(t) - \mathbf{v}_R(t)| > \boldsymbol{\delta}, \tag{8}$$

where the thresholds are $\boldsymbol{\delta} = (5, 10, 5)^T$ for the Lorenz system and $\boldsymbol{\delta} = (2.5, 2.5, 4)^T$ for the Rössler system. The values are chosen this way due to the different ranges of the state variables in both systems. The aim is that small fluctuations around the actual trajectory as well as minor detours do not exceed the threshold. Empirically, we found that distances between the trajectories become much larger than the threshold values as soon as short-term prediction collapses.

2. Correlation dimension

One important characteristic of the long-term properties of the system is its structural complexity. This can be quantified by calculating the correlation dimension, which measures the dimensionality of the space populated by the trajectory.[28] It is based on the correlation integral,

$$\begin{aligned} C(r) &= \lim_{N\to\infty} \frac{1}{N^2} \sum_{i,j=1}^{N} \theta(r - |\mathbf{x}_i - \mathbf{x}_j|) \\ &= \int_0^r d^3r' c(\mathbf{r}'), \end{aligned} \tag{9}$$

which describes the mean probability that two states in the phase space are close to each other at different time steps. The condition "close to" is met if the distance between the two states is less than the threshold distance r. θ represents the Heaviside function, while $c(\mathbf{r}')$ denotes the standard correlation function. For self-similar strange attractors, the following power-law relationship holds in a range of r,

$$C(r) \propto r^{\nu}. \tag{10}$$

The "correlation dimension" is then measured by the scaling exponent ν. We use the Grassberger Procaccia algorithm[29] to calculate the correlation dimension of our trajectories. This approach is purely data driven and, therefore, does not require any knowledge about the system.

3. Largest Lyapunov exponent

Another aspect of the long-term behavior is the temporal complexity of the system. When dealing with chaotic systems, looking at its Lyapunov exponents is an obvious choice. The Lyapunov exponents λ_i describe the average rate of divergence of nearby states in the phase space and thus measure sensitivity to initial conditions. For each dimension in the phase space, there is one exponent. If the system exhibits at least one positive Lyapunov exponent, it is classified as chaotic, while the magnitude of the exponent quantifies the time scale on which the system becomes unpredictable.[30,31] Therefore, it is sufficient for our analysis to determine only the largest Lyapunov exponent λ_1,

$$d(t) = Ce^{\lambda_1 t}, \tag{11}$$

which makes the task computationally easier. Here, $d(t)$ is the average distance or separation of the initially nearby states at time t and C is a constant that normalizes the initial separation. To calculate the largest Lyapunov exponent, we use the Rosenstein algorithm.[32]

III. RESULTS

Although machine learning techniques and reservoir computing, in particular, have become increasingly popular, a thorough statistical analysis of the forecast results is yet missing. Therefore, we found it insightful to not only perform one single prediction where "prediction" means forecasting the trajectory for $t_{prediction} = 10\,000$ time steps. As there are random numbers involved in the construction of the reservoir $\mathbf{A}$ as well as the input function $\mathbf{W}_{in}$, we can run the prediction with $N = 1000$ different random number seeds while keeping all other parameters of the network constant. Therefore, for different seeds, both $\mathbf{A}$ and $\mathbf{W}_{in}$ will vary. This allows us to gain a statistical view on the quality of the prediction instead of analyzing only single realizations. In addition to the parameters mentioned in Sec. II, there is one more parameter that we can tune. This is the spectral radius ρ of the adjacency matrix of the reservoir $\mathbf{A}$, which is defined as its largest absolute eigenvalue,

$$\rho(\mathbf{A}) = \max\{|\lambda_1|, \ldots, |\lambda_{D_r}|\}, \tag{12}$$

and reflects some kind of average degree of the network. We can adjust the spectral radius by

$$\mathbf{A}^* = \frac{\mathbf{A}}{\rho(\mathbf{A})}\rho^*, \tag{13}$$

where ρ^* is the desired spectral radius. Note that λ_i here denote the eigenvalues of the adjacency matrix of the reservoir $\mathbf{A}$—not to be confused with the Lyapunov exponent in Eq. (11), which is commonly called λ as well. Other studies showed results for particular values of ρ such as Pathak *et al.*,[3] e.g., claiming that the prediction using a spectral radius of $\rho = 1.2$ accurately resembles the long-term climate of the system, while the same setup with $\rho = 1.45$ does not. To possibly reproduce these results and to assess the best ranges for the spectral radius, we ran the reservoir computing with $N = 1000$ different random number seeds for each spectral radius $\rho_i^* \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.3, 1.45, 1.6, 1.8, 2.0, 2.2, 2.4\}$, while all other parameters remain constant. We simulate one trajectory, which is used for the training of the network as well as for the comparison of the predicted trajectory with the actual one. Furthermore, we simulated additional 1000 trajectories of the actual Lorenz and Rössler system with different randomly chosen initial conditions in order to investigate the statistical error when calculating the correlation dimension and the largest Lyapunov exponent from the time series with limited length.

Table I shows the means and standard deviations for both measures indicating that the error is reasonably small. As we use "only" 10 000 data points, our results are slightly below the expected values of around 2.04 for the correlation dimension and 0.89 for the largest Lyapunov exponent of the Lorenz system. For the Rössler

TABLE I. Mean and standard deviation σ of the two measures calculated from 1000 simulated trajectories with different initial conditions.

Lorenz	Mean	σ
Correlation dimension	2.026	0.014
Largest Lyapunov exponent	0.878	0.029
Rössler	Mean	σ
Correlation dimension	1.713	0.037
Largest Lyapunov exponent	0.107	0.011

system, the results for the correlation dimension are significantly below the desired value of around 2 because the Grassberger Procaccia algorithm is slower converging as compared to the Lorenz system when using only 10 000 data points. This is also reflected in the higher standard deviation σ of the correlation dimension. However, we verified through increasing the number of data points that our calculations converge to the expected results.

Figure 1 shows two examples of predicted trajectories using reservoir computing in the setup described above with a spectral radius of $\rho = 0.3$. Although we ran the prediction over $n = 10\,000$ time steps, we plotted the results for $n = 2000$ time steps for the sake of clarity. On the left side of the plot, one can see the trajectory of the X coordinate for the predicted system using reservoir computing (blue) and the simulated system based on the Lorenz differential equations (green). In the upper plot, both trajectories are overlapping for around 200 time steps and then deviate while still showing the characteristic pattern of the Lorenz system. However, in the lower plot, both trajectories already separate after less than 100 time steps leading to a pattern, which looks completely different.

This is remarkable given the fact that the setup for both cases is identical except for a different random number seed, which results in different realizations of the input function $\mathbf{W}_{in}$ and the reservoir $\mathbf{A}$. Since looking solely at the X coordinate yields insufficient information about the overall prediction quality, it is meaningful to investigate the whole attractor as plotted on the right side of Fig. 1. Here, we can see that the Lorenz attractor is reconstructed very well by the upper prediction, while the lower prediction has nothing to do with the butterfly-shaped Lorenz attractor. Instead, the trajectory quickly runs into a fixed point after detaching and partly forming some kind of a mirrored Lorenz attractor. The difference in prediction quality is not only reflected in the ability to match the original trajectory in the short-term but also with respect to the correlation dimension and the largest Lyapunov exponent. While in the upper case the resulting values of $\nu = 1.992$ and $\lambda = 0.851$ are well within the expectations for the Lorenz system, the lower realization

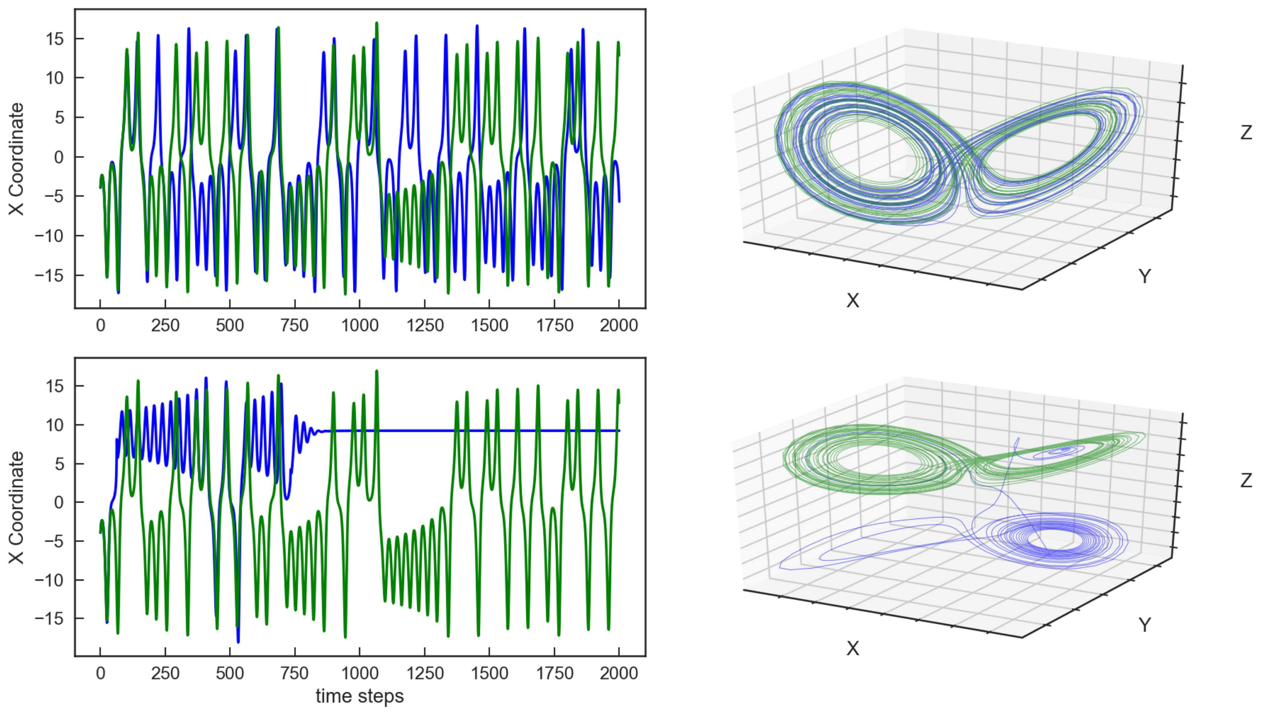

FIG. 1. Left: X coordinate of two predicted (blue) trajectories of the Lorenz system plotted over $n = 2000$ time steps. The results are compared against the trajectory of the simulated actual Lorenz system (green). The upper plot shows a good realization where both trajectories are partly overlapping, while the lower part shows a bad prediction. Right: Three dimensional attractor for both cases. The spectral radius is $\rho = 0.3$, and random Erdös-Renyi networks are used for the reservoir **A**. The correlation dimension is $\nu = 1.992$ for the upper and $\nu = 0.007$ for the lower realization, while the largest Lyapunov exponents are $\lambda_1 = 0.851$ (upper) and $\lambda_1 = 0.420$ (lower).

completely misses to reconstruct the long-term climate. Immediately, the question arises if this observation is an exception or whether the prediction quality is not robust with respect to different random number seeds. Therefore, we systematically investigated this effect by running the same setup with $N = 1000$ different random number seeds. Since it would be laborious to visually inspect the trajectories and attractors of all realizations, we rely on the measures introduced in Sec. II in order to assess if a prediction is good or bad.

Figure 2 shows a scatter plot where the forecast horizon is plotted against the correlation dimension for all realizations. The colors represent different spectral radii, and for each spectral radius, there is one point for each of the $N = 1000$ random number seeds. In order to make the results better readable, we divided the plot into four sections where we grouped the results for four to five different spectral radii. The first thing we can observe is that the prediction of the Lorenz system using reservoir computing works better for smaller spectral radii. However, more importantly, one can also see that the prediction quality significantly varies even when using the same spectral radius—as already indicated by Fig. 1. This becomes not only evident by the results for the forecast horizon but especially when considering the correlation dimension. Its values quickly spread when increasing the spectral radius indicating that the resulting predicted trajectories do not resemble the long-term climate of the system well in many cases. Figure 3 shows the same results for the Rössler system. In contrast to the Lorenz system, not the smallest spectral radius but a choice of $\rho = 0.4$ leads to the best results. In addition, the short-term prediction ability measured by the forecast horizon is significantly better with a number of predictions matching the original trajectory for 500 to almost 1000 time steps. However, the spread of the results for the correlation dimension seems to be larger as compared to the Lorenz system with high variability even for the best working spectral radii. This is partly due to the fact that the numerical calculation of the correlation dimension converges slower for the Rössler system as mentioned in Sec. III. As in Fig. 2, there seems to be some "quantization" of the forecast horizon for both systems. The reason is that the predicted trajectory typically detaches from the actual one after completing a loop around the attractor.

The variability becomes even clearer when looking at Fig. 4. Here, we can see another scatter plot where the largest Lyapunov exponent is plotted against the correlation dimension, and thus, both components of assessing the long-term climate are reflected. The left plots show the results for the Lorenz system whereas those for the Rössler system are shown on the right side. In addition, the red ellipse shows the five sigma errors of the correlation dimension and the largest Lyapunov exponent calculated from 1000 simulations using the actual equations of the Lorenz and Rössler system as shown in Table I. When studying the left side, it becomes clear that even for the smallest and for the Lorenz system the best working spectral radius being $\rho = 0.1$ (top left plot, dark green), the resulting "bubble" of points is of the same size as the $\sigma = 5$ error ellipse. This indicates strong variability given that $\sigma = 5$ is a large error. For the larger spectral radii shown in the bottom left plot—including the values $\rho = 1.2$ and $\rho = 1.45$ as used in Ref. 3—there is not a single point within the ellipse. This indicates that the prediction completely fails

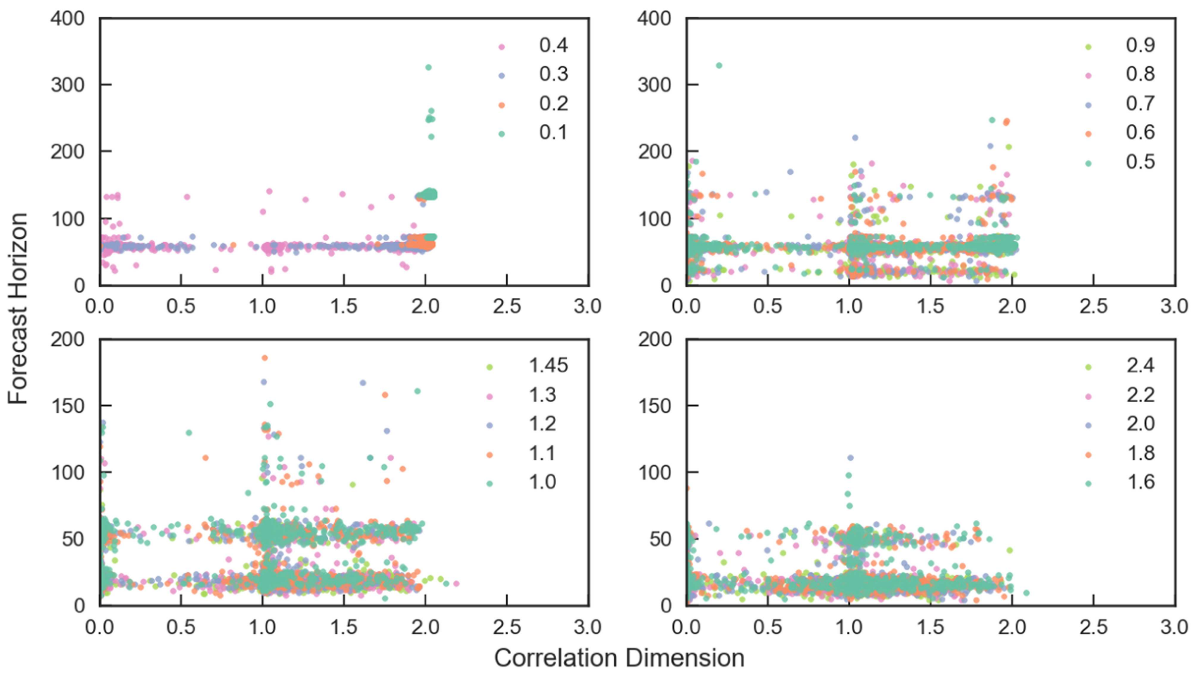

FIG. 2. Forecast horizon scattered against the correlation dimension for each of the $N = 1000$ predictions of the Lorenz system per spectral radius. Different spectral radii are differentiated by colors. Random Erdös-Renyi networks are used for the reservoir **A**.

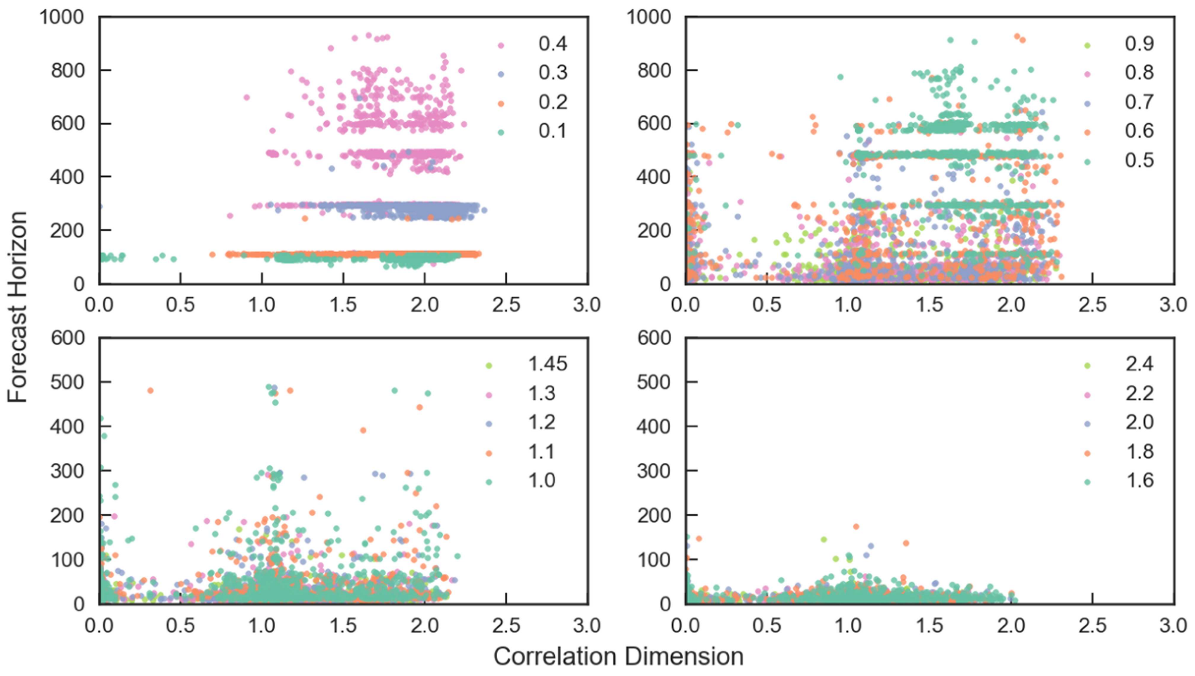

FIG. 3. Forecast horizon scattered against the correlation dimension for each of the $N = 1000$ predictions of the Rössler system per spectral radius. Different spectral radii are differentiated by colors. Random Erdös-Renyi networks are used for the reservoir **A**.

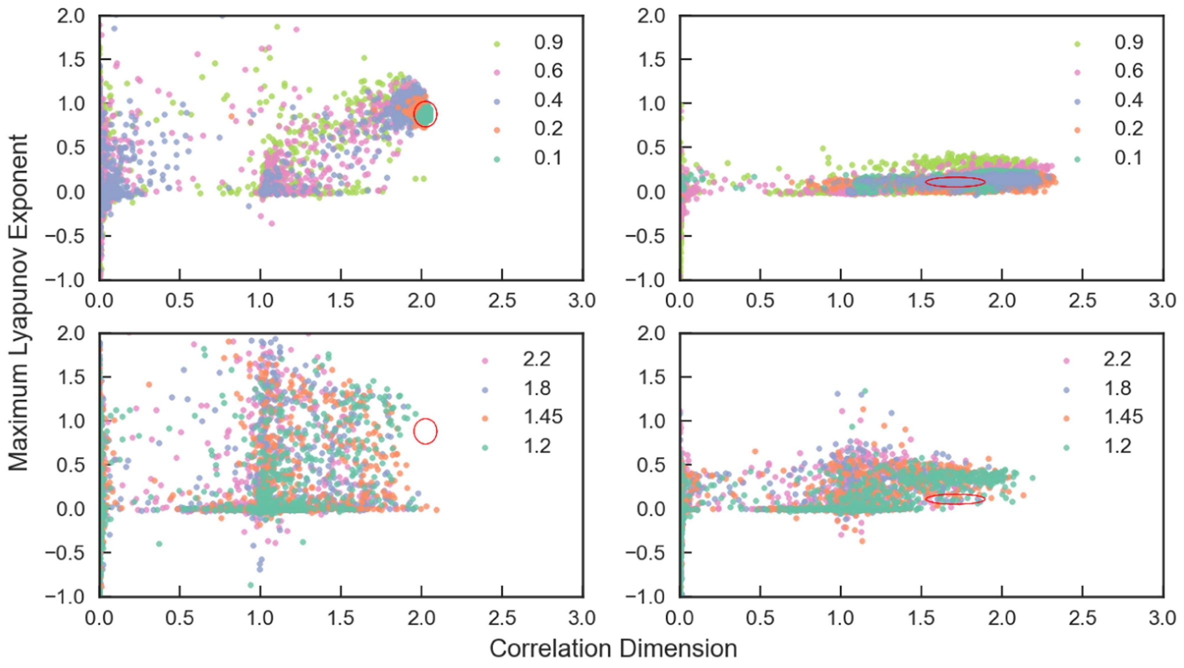

FIG. 4. Largest Lyapunov exponent scattered against the correlation dimension for each of the $N = 1000$ predictions per spectral radius. Results are shown for the Lorenz (left) and Rössler system (right). Different spectral radii are differentiated by colors. The red object represents the five sigma error ellipse of both measures calculated based on 1000 simulated true trajectories. Random Erdös-Renyi networks are used for the reservoir **A**.

to reproduce the long-term climate for those cases. The results for the Rössler system on the right side of the plot give a similar picture. However, even for the best working spectral radius of $\rho = 0.4$, there are many points outside of the $\sigma = 5$ error ellipse. In addition, one can also see from the upper plots of Fig. 4 that a good reproduction of the correlation dimension also tends to coincide with a better reproduction of the largest Lyapunov exponent, although this effect is not very significant.

So far, we only looked at the results based on the random Erdös-Renyi networks. In order to compare the performance of the three different network topologies on a statistical level, we perform a χ^2 analysis of the long-term climate prediction. This means that we calculate

$$\chi^2(i,\rho) = \sum_{j=1}^{2}\left[\frac{X_j(i,\rho) - \langle X_j\rangle}{\sigma_{X_j}}\right]^2, \tag{14}$$

where i is the ith random number seed and ρ indexes the spectral radius. The sum goes over the correlation dimension ($j = 1$) and the largest Lyapunov exponent ($j = 2$). $\langle X_j\rangle$ represents the average value derived from 1000 simulated actual Lorenz trajectories—as shown in Table I, while σ_{X_j} is the corresponding standard deviation. Therefore, the resulting χ^2 describes how strong the predicted results deviate from the actual values weighed by the errors of the actual values.

Figure 5 shows the cumulative distribution of the χ^2 for the three network topologies. The top plot contains the results for the Lorenz system using the spectral radius $\rho = 0.1$ and evaluation values of χ^2 for 0 and 10. We can see that scale-free networks (red line) tend to work worse, while small world networks (blue line) slightly outperform the Erdös-Renyi networks (yellow line). However, it becomes clear that the method of reservoir computing works for all networks topologies tested here. In contrast, there is a difference between scale-free networks and the other topologies in the case of the Rössler system (bottom plot). Here, we calculated the cumulative distribution for values of χ^2 between 0 and 500 based on $\rho = 0.4$. The reason is that even for the best working spectral radius, the variability is significantly higher as compared to the Lorenz system, which leads to higher values of χ^2 with only very few data points in the 0–10 range. It is interesting to note that the performance of scale-free networks is now strongly below the other two networks, while Erdös-Renyi networks are slightly leading overall. Therefore, in contrast to the Lorenz system, network topology seems to matter in this case.

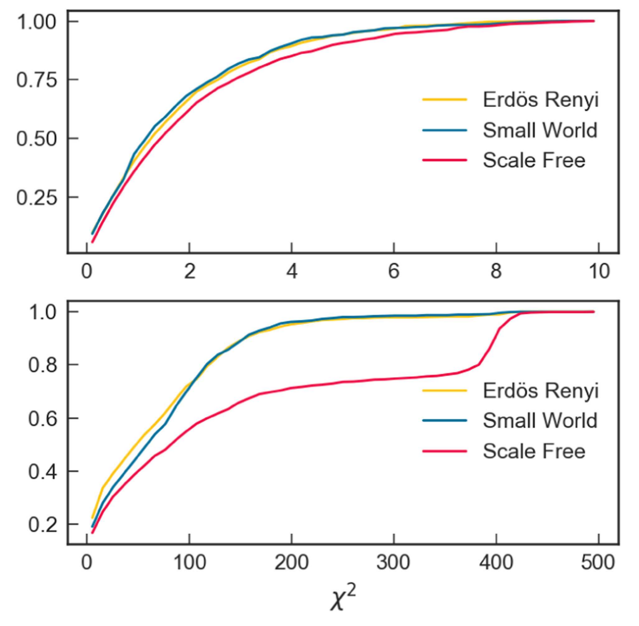

FIG. 5. Top plot: Cumulative distribution of χ^2 for the best working spectral radius $\rho = 0.1$ of the Lorenz system calculated for values between 0 and 10. Bottom plot: Same for the Rössler system with $\rho = 0.4$ and values between 0 and 500.

IV. CONCLUSIONS AND OUTLOOK

In this paper, we investigated the prediction of chaotic attractors by using reservoir computing from a statistical perspective. Instead of only predicting one trajectory, we simulated 1000 realizations each—where each realization corresponds to another random number seed—for a number of different spectral radii ρ. Analyzing both the Lorenz and Rössler system, we found that the ability to exactly forecast the correct trajectory as well as the reconstruction of the long-term climate measured by the correlation dimension and largest Lyapunov exponent strongly varies. Even for the exact same parameter setup, there can be very good results that match the true trajectory for a large number of time steps and nicely reconstruct the attractor. On the other hand, there can be results that completely fail in either one or both dimensions and are not reflecting the desired properties of the system. The results suggest that smaller spectral radii than typically used work better for both systems, while in the case of the Lorenz system, even the smallest spectral radius $\rho = 0.1$ performed best. However, even in this case, results show strong variability as they completely fill the five sigma error ellipse of the correlation dimension and largest Lyapunov exponent. For the Rössler system, there are several predictions exceeding the $\sigma = 5$ error ellipse for the best working spectral radius of $\rho = 0.4$, and thus, variability is even stronger as compared to the Lorenz system. This is an interesting observation given that the Rössler system is considered less nonlinear. Overall, special care must be taken in selecting the good predictions as realizations, which deliver better short time prediction also tend to better resemble the long-term climate of the system.

Furthermore, we ran the same analysis using two other network topologies: Small world and scale free networks. In essence, they produced comparable results with a slight outperformance of small world networks and underperformance of scale-free networks for the Lorenz system. For the Rössler system, the picture is different with a slight outperformance of Erdös-Renyi networks, while scale-free networks are showing worst results in terms of the χ^2 analysis. A tentative explanation for the lower performance of scale-free networks could be the following: According to Singh *et al.*,[33] the more capable a brain or neuronal system is, the less scaling is present in its degree distribution. Overall, it is important to point out that despite the differences presented here, the general methodology of reservoir computing works for different network topologies.

However, even after trying different parameters and alternatively a setup where also the input states are going into the regression as described by Lukoševičius and Jaeger,[20] the variability can always be observed.

Once the network is trained, the prediction is deterministic and depends strongly on the weights and only weak on the topology. It should thus be possible to associate good and bad predictions with differential properties of the respective realization of the reservoir in a systematic way. First attempts in this direction for a reservoir with unweighted edges have recently been reported in Carroll and Pecora.[34] Once based on those insights, more stable predictions are possible, a more precise analysis of the attractor properties, e.g., with the spectrum of Lyapunov exponents could be useful and necessary. Furthermore, the role of the network size is also an interesting aspect. Current and future work is dedicated to the investigation of these questions—not the least because the answers to them will shed new light on the complexity of the underlying dynamical system.

ACKNOWLEDGMENTS

We wish to acknowledge useful discussions and comments from Jonas Aumeier, Ingo Laut, and Mierk Schwabe.

REFERENCES

[1] J. Theiler, S. Eubank, A. Longtin, B. Galdrikian, and J. Doyne Farmer, "Testing for nonlinearity in time series: The method of surrogate data," Physica D **58**, 77–94 (1992).
[2] Z. Lu, J. Pathak, B. Hunt, M. Girvan, R. Brockett, and E. Ott, "Reservoir observers: Model-free inference of unmeasured variables in chaotic systems," Chaos **27**, 041102 (2017).
[3] J. Pathak, Z. Lu, B. R. Hunt, M. Girvan, and E. Ott, "Using machine learning to replicate chaotic attractors and calculate Lyapunov exponents from data," Chaos **27**, 121102 (2017).
[4] J. Pathak, B. Hunt, M. Girvan, Z. Lu, and E. Ott, "Model-free prediction of large spatiotemporally chaotic systems from data: A reservoir computing approach," Phys. Rev. Lett. **120**, 024102 (2018).
[5] J. Pathak, A. Wikner, R. Fussell, S. Chandra, B. R. Hunt, M. Girvan, and E. Ott, "Hybrid forecasting of chaotic processes: Using machine learning in conjunction with a knowledge-based model," Chaos **28**, 041101 (2018); e-print arXiv:1803.04779.
[6] R. S. Zimmermann and U. Parlitz, "Observing spatio-temporal dynamics of excitable media using reservoir computing," Chaos **28**, 043118 (2018).
[7] T. L. Carroll, "Using reservoir computers to distinguish chaotic signals," Phys. Rev. E **98**, 052209 (2018); e-print arXiv:1810.04574.
[8] Z. Lu, B. R. Hunt, and E. Ott, "Attractor reconstruction by machine learning," Chaos **28**, 061104 (2018).
[9] P. Antonik, M. Gulina, J. Pauwels, and S. Massar, "Using a reservoir computer to learn chaotic attractors, with applications to chaos synchronization and cryptography," Phys. Rev. E **98**, 012215 (2018).
[10] W. Maass, T. Natschlaeger, and H. Markram, "Real-time computing without stable states: A new framework for neural computation based on perturbations," Neural Comput. **14**, 2531–2560 (2002).
[11] H. Jaeger and H. Haas, "Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication," Science **304**, 78–80 (2004).
[12] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," Nature **393**, 440 (1998).
[13] R. Albert and A.-L. Barabási, "Statistical mechanics of complex networks," Rev. Mod. Phys. **74**, 47–97 (2002); e-print arXiv:cond-mat/0106096.
[14] A. D. Broido and A. Clauset, "Scale-free networks are rare," Nature Commun. **10**, 1017 (2019).
[15] M. Gerlach and E. G. Altmann, "Testing statistical laws in complex systems," Phys. Rev. Lett. **122**, 168301 (2019).
[16] H. Cui, X. Liu, and L. Li, "The architecture of dynamic reservoir in the echo state network," Chaos **22**, 033127 (2012).
[17] E. N. Lorenz, "Deterministic nonperiodic flow," J. Atmos. Sci. **20**, 130–141 (1963).
[18] Z. Lu, J. Pathak, B. Hunt, M. Girvan, R. Brockett, and E. Ott, "Reservoir observers: Model-free inference of unmeasured variables in chaotic systems," Chaos **27**, 041102 (2017).
[19] O. E. Rössler, "An equation for continuous chaos," Phys. Lett. A **57**, 397–398 (1976).
[20] M. Lukoševičius and H. Jaeger, "Reservoir computing approaches to recurrent neural network training," Comput. Sci. Rev. **3**, 127–149 (2009).
[21] R. Gençay and T. Liu, "Nonlinear modelling and prediction with feedforward and recurrent networks," Physica D **108**, 119–134 (1997).
[22] P. Erdos, "On random graphs," Publ. Math. **6**, 290–297 (1959).
[23] A. E. Hoerl and R. W. Kennard, "Ridge regression: Biased estimation for nonorthogonal problems," Technometrics **12**, 55–67 (1970).
[24] A.-L. Barabási and E. Bonabeau, "Scale-free networks," Sci. Am. **288**, 60–69 (2003).
[25] L. Amaral, A. Scala, M. Barthélémy, and H. Stanley, "Classes of small-world networks," Proc. Natl. Acad. Sci. U.S.A. **97**, 11149–11152 (2000).
[26] A. Hagberg, P. Swart, and D. S Chult, "Exploring network structure, dynamics, and function using NetworkX," Technical Report, Los Alamos National Laboratory (LANL), Los Alamos, NM, 2008.
[27] B. Bollobás, C. Borgs, J. Chayes, and O. Riordan, "Directed scale-free graphs," in *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms* (Society for Industrial and Applied Mathematics, 2003), pp. 132–139.
[28] P. Grassberger and I. Procaccia, "Measuring the strangeness of strange attractors," Physica D **9**, 189–208 (1983).
[29] P. Grassberger, "Generalized dimensions of strange attractors," Phys. Lett. A **97**, 227–230 (1983).
[30] A. Wolf, J. B. Swift, H. L. Swinney, and J. A. Vastano, "Determining Lyapunov exponents from a time series," Physica D **16**, 285–317 (1985).
[31] R. Shaw, "Strange attractors, chaotic behavior, and information flow," Z. Naturforsch. A **36**, 80–112 (1981).
[32] M. T. Rosenstein, J. J. Collins, and C. J. De Luca, "A practical method for calculating largest Lyapunov exponents from small data sets," Physica D **65**, 117–134 (1993).
[33] S. S. Singh, B. Khundrakpam, A. T. Reid, J. D. Lewis, A. C. Evans, R. Ishrat, B. I. Sharma, and R. B. Singh, "Scaling in topological properties of brain networks," Sci. Rep. **6**, 24926 (2016).
[34] T. L. Carroll and L. M. Pecora, "Network structure effects in reservoir computers," Chaos **29**, 083130 (2019).

B Reducing network size and improving prediction stability of reservoir computing

A. Haluszczynski et al., “Reducing network size and improving prediction stability of reservoir computing”, Chaos: An Interdisciplinary Journal of Nonlinear Science **30**, 063136 (2020)

Reducing network size and improving prediction stability of reservoir computing

Cite as: Chaos **30**, 063136 (2020); https://doi.org/10.1063/5.0006869
Submitted: 06 March 2020 . Accepted: 05 June 2020 . Published Online: 16 June 2020

Alexander Haluszczynski, Jonas Aumeier, Joschka Herteux, and Christoph Räth

Reducing network size and improving prediction stability of reservoir computing

Cite as: Chaos **30**, 063136 (2020); doi: 10.1063/5.0006869
Submitted: 6 March 2020 · Accepted: 5 June 2020 ·
Published Online: 16 June 2020

View Online Export Citation CrossMark

Alexander Haluszczynski,[1,2,a)] Jonas Aumeier,[3,b)] Joschka Herteux,[3,c)] and Christoph Räth[3,d)]

AFFILIATIONS

[1] Department of Physics, Ludwig-Maximilians-Universität, Schellingstraße 4, 80799 Munich, Germany
[2] risklab GmbH, Seidlstraße 24, 80335 Munich, Germany
[3] Institut für Materialphysik im Weltraum, Deutsches Zentrum für Luft- und Raumfahrt, Münchner Str. 20, 82234 Wessling, Germany

[a)] **Author to whom correspondence should be addressed:** alexander.haluszczynski@gmail.com
[b)] **Electronic mail:** jonas.aumeier@dlr.de
[c)] **Electronic mail:** joschka.herteux@dlr.de
[d)] **Electronic mail:** christoph.raeth@dlr.de

ABSTRACT

Reservoir computing is a very promising approach for the prediction of complex nonlinear dynamical systems. Besides capturing the exact short-term trajectories of nonlinear systems, it has also proved to reproduce its characteristic long-term properties very accurately. However, predictions do not always work equivalently well. It has been shown that both short- and long-term predictions vary significantly among different random realizations of the reservoir. In order to gain an understanding on when reservoir computing works best, we investigate some differential properties of the respective realization of the reservoir in a systematic way. We find that removing nodes that correspond to the largest weights in the output regression matrix reduces outliers and improves overall prediction quality. Moreover, this allows to effectively reduce the network size and, therefore, increase computational efficiency. In addition, we use a nonlinear scaling factor in the hyperbolic tangent of the activation function. This adjusts the response of the activation function to the range of values of the input variables of the nodes. As a consequence, this reduces the number of outliers significantly and increases both the short- and long-term prediction quality for the nonlinear systems investigated in this study. Our results demonstrate that a large optimization potential lies in the systematical refinement of the differential reservoir properties for a given dataset.

A pervasive stigma of common machine learning (ML) methods is that they are considered an inscrutable black box. This is problematic for many practical applications, since a precise understanding of the tool is necessary to correctly assess uncertainties and sensitivities. Knowing that there is often significant variability in the prediction quality, the natural question arises how one can identify good predictions and prevent outliers that do not adequately resemble the targeted data or system. In contrast to many other neural network based approaches, reservoir computing (RC) makes it possible to bring light into the dark. Its comparably simple architecture allows for a systematic analysis of the differential properties of the reservoir realizations, leading to good or bad predictions. In the context of nonlinear dynamical systems, a good prediction should not only be able to match the actual short-term trajectory but also needs to recreate its statistical long-term characteristics. To investigate the connection between properties of the reservoir and prediction quality, we remove certain nodes from the reservoir network and analyze how this impacts predictions. We find that a controlled node removal of the "right" nodes not only leads to less variability, and thus better predictions, but also allows to reduce network size noticeably. Furthermore, we turn from the reservoir itself to the activation function and show how rescaling of the argument gives rise to better results.

I. INTRODUCTION

The remarkable rise of machine learning (ML) techniques during the recent years has made it inevitable for researchers to dig deeper into the mechanisms and properties of the methods. This is required to fundamentally understand how, when, and why they are working well. Otherwise, the application of machine learning techniques to various fields in business and science carries the risk of misinterpreting the results if deeper methodological knowledge is lacking.

In the context of complex systems research, the use of reservoir computing (RC)[1]—also known as *Echo State Networks*[2,3]—for quantifying and predicting the spatiotemporal dynamics of nonlinear systems has attracted much attention recently.[4–11] RC represents a special kind of recurrent neural networks (RNNs). The core of the model is a neural network called reservoir, which is a complex network with loops. Input data are fed into the nodes of the reservoir, which are connected according to a predefined network topology (mostly random networks). Only the weights of the linear output layer, transforming the reservoir response to output variables, are subject to optimization via linear regression. This makes the learning extremely fast, comparatively transparent, and omits the vanishing gradient problem of other RNN training methods. The reservoir is kept fixed and only the weights constituting the output layer are optimized in a deterministic and non-iterative manner. Therefore, RC allows for a controlled differential manipulation of the properties of the neural network and to identify, how those changes are associated with the prediction quality.

Many of the achievements obtained with RC—be it, e.g., the cross-prediction of variables in two-dimensional excitable media,[8] the reproduction of the spectrum of Lyapunov exponents in lower dimensional (Lorenz or Rössler) and higher dimensional (Kuramoto–Sivashinsky) systems,[4–6] or the prediction of extreme events[12]—are impressive and significantly extend the possibilities to predict future states of high dimensional, nonlinear systems. While the results reported in the works mentioned above are mainly based on a single or few realizations of reservoir computing, we showed, however, in an earlier study[13] that there is a strong variability in prediction quality by running multiple realizations of the reservoir. The natural question that arises is where this variability comes from and whether one can associate good and bad predictions with differential properties of the reservoir. Based on a reservoir with unweighted edges, first attempts in this direction have been made by Carroll and Pecora.[14] They showed that symmetries in the network do have a considerable effect on the prediction quality of RC. In this work, we investigate the effect of two methods to manipulate reservoirs with weighted edges, since those are typically used in time-series prediction. First, we decrease the reservoir size by applying pruning techniques. Thinning out a (deep) neural network is a classical technique for enhancing its generalization ability. However, pruning mostly refers to the removal of synapses, i.e., edges, in a network. More rarely, pruning refers to the removal of neurons, i.e., nodes. So far, only few studies have investigated the effects of a controlled removal of edges or nodes in reservoir computing (see, e.g., Refs. 15–17). Pruning of the reservoir network is a new optimization approach for the prediction of the long-term behavior of chaotic systems using RC. We propose and discuss a novel scheme for the controlled removal of nodes that relies on ideas stemming from network science. In addition, we vary the nonlinearity of the hyperbolic tangent activation function with a scaling factor. The paper is organized as follows: Sec. II introduces reservoir computing and the reservoir manipulation methods used in our study. In Sec. III, we present the main results obtained from the statistical analysis of the prediction results and its associated differential properties of the reservoir realizations. The summary and an outlook are given in Sec. IV.

II. METHODS

A. Reservoir computing

Within the class of artificial recurrent neural networks, reservoir computing is a promising approach for predicting complex nonlinear dynamical systems. The model is based on a static network called *reservoir*, whose nodes and edges are kept fixed after it has initially been set up. In contrast to feedforward type neural networks, the reservoir is allowed to have loops, and, therefore, past values are fed back into the system allowing for dynamics.[18,19] As opposed to other neural network based machine learning approaches, the training process of reservoir computing alters only the linear output layer. This allows for large model dimensionality while still being computationally feasible.[10]

This implementation is mainly based on the setup of our previous study[13] and works in the following way. First, we set up the reservoir $\mathbf{A}$, which has dimensionality D_r, and is constructed as a sparse Erdös–Renyi random network.[20] In our study, we chose $D_r = 200$ nodes that are connected with a probability of $p = 0.02$. This results in an unweighted average degree of $d = 4$, while the weights of the edges are determined by independently drawn and uniformly distributed numbers within the interval $[-1, 1]$. Once created, the reservoir is fixed and does not change over time. The next task is to feed the D dimensional input data $\mathbf{u}(t)$ into the reservoir $\mathbf{A}$. This requires an $D_r \times D$ input matrix $\mathbf{W}_{in}$ that defines for every node the excitation by each dimension of the input signal. The entries of $\mathbf{W}_{in}$ are chosen to be uniformly distributed random numbers within a certain range to be defined later.

A key element of the system are its $D_r \times 1$ reservoir states $\mathbf{r}(t)$, which represent the scalar states of the nodes of the reservoir. We initially set $r_i(t_0) = 0$ for all nodes and update the reservoir states in each time step according to the equation

$$\mathbf{r}(t + \Delta t) = \alpha \mathbf{r}(t) + (1 - \alpha)\tanh(\mathbf{A}\mathbf{r}(t) + \mathbf{W}_{in}\mathbf{u}(t)). \tag{1}$$

As in Pathak *et al.*,[5] we set $\alpha = 0$ and, therefore, do not mix the input function with past reservoir states. Now, we have a dynamical system, where the reservoir $\mathbf{A}$ itself is static and its scalar states $\mathbf{r}(t)$ change over time.

The next step is to map the reservoir states $\mathbf{r}(t)$ back to the D dimensional output $\mathbf{v}$ through an output function $\mathbf{W}_{out}$

$$\mathbf{v}(t + \Delta t) = \mathbf{W}_{out}(\mathbf{r}(t + \Delta t), \mathbf{P}). \tag{2}$$

Here, we assume that $\mathbf{W}_{out}$ depends linearly on a matrix $\mathbf{P}$ and reads $\mathbf{W}_{out}(\mathbf{r}, \mathbf{P}) = \mathbf{P}\mathbf{r}$. This means that the output of the system depends only on the reservoir states $\mathbf{r}(t)$ and the output matrix $\mathbf{P}$, which contains $D_r \times D$ degrees of freedom. Therefore, after acquiring a sufficient number of reservoir states $\mathbf{r}(t)$, we have to choose $\mathbf{P}$ such that the output $\mathbf{v}$ of the reservoir is as close as possible to the known real data $\mathbf{v}_R$. This process is called training. Specifically, the task is to find an output matrix **P** using Ridge regression, which minimizes

$$\sum_{-T\leq t\leq 0} \| \mathbf{W}_{out}(\mathbf{r}(t),\mathbf{P}) - \mathbf{v}_R(t) \|^2 - \beta \| \mathbf{P} \|^2, \tag{3}$$

where β is the regularization constant. Setting $\beta > 0$ prevents from overfitting by penalizing large values of the fitting parameters. The notation $\| \mathbf{P} \|$ describes the sum of the square elements of the matrix **P**. For solving this problem, we are using the matrix form of the Ridge regression,[21] which leads to

$$\mathbf{P} = (\mathbf{r}^T\mathbf{r} + \beta\mathbb{1})^{-1}\mathbf{r}^T\mathbf{v}_R. \tag{4}$$

The notion **r** and $\mathbf{v}_R$ without the time indexing t denotes matrices, where the columns are the vectors $\mathbf{r}(t)$ and $\mathbf{v}_R(t)$, respectively, in each time step. In our implementation, we chose $t_{train} = 5000$ training time steps while allowing for a washout or initialization phase of $t_{init} = 5000$. During this time, we do not "record" the reservoir states $\mathbf{r}(t)$ in order to allow the system to sufficiently synchronize with the dynamics of the input signal.

Now that **P** is determined, we can feed the predicted state $\mathbf{v}(t)$ back in as input instead of the actual data $\mathbf{u}(t)$ by combining Eqs. (1) and (2). This allows to create predicted trajectories of arbitrary length due to the recursive equation for the reservoir states $\mathbf{r}(t)$,

$$\begin{aligned}\mathbf{r}(t+\Delta t) &= \tanh(\mathbf{A}\mathbf{r}(t) + \mathbf{W}_{in}\mathbf{W}_{out}(\mathbf{r}(t),\mathbf{P}))\\ &= \tanh(\mathbf{A}\mathbf{r}(t) + \mathbf{W}_{in}\mathbf{P}\mathbf{r}(t)).\end{aligned} \tag{5}$$

An illustration of this reservoir computing framework is given in Fig. 1.

To find the most suitable parameter values for the spectral radius of the reservoir $\rho(\mathbf{A})$, the scale for $\mathbf{W}_{in}$ and the regularization constant β, we carried out a hyperparameter optimization. As reservoir computing system can be trained very quickly, we use a simple random search procedure with uniform sampling from the parameter space.[22] The objective function is the forecast horizon, as defined in Sec. II C, averaged over $N = 30$ realizations. The term *realizations* means running reservoir computing with the exact same parameters but different random realizations of the reservoir **A** and the input function $\mathbf{W}_{in}$. Each of the realizations is starting from randomly chosen coordinates obtained from simulating a very long trajectory of the Lorenz system. In order to assure independent trajectories, small scale uniform noise is added. The optimal values are shown in Table I.

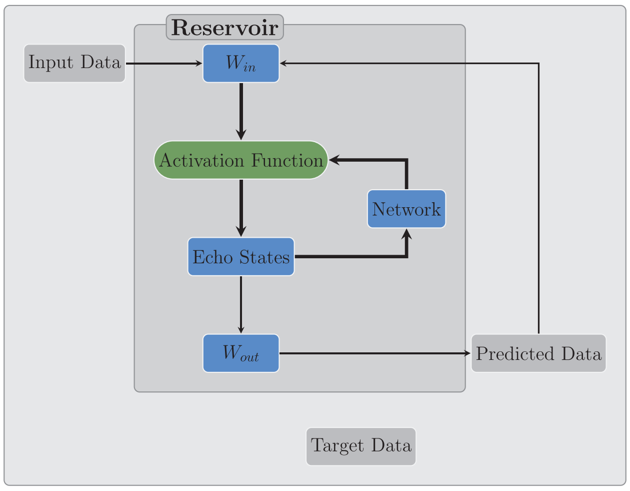

FIG. 1. Schematic illustration of reservoir computing.

TABLE I. Results of the hyperparameter optimization.

ρ	0.17
$\mathbf{W}_{in}$ scale	0.17
β	1.9×10^{-11}

B. Controlled node removal and activation function adjustment

The standard approach to reservoir computing exhibits a strong variability in prediction quality as shown in Haluszczynski and Räth.[13] In order to reduce this variability, we make alterations to the reservoir structure by removing nodes and their respective edges from both the reservoir **A** and $\mathbf{W}_{in}$. This is inspired by the concept of *attack tolerance*[23] in complex networks and the aim is to investigate the effect of removing nodes on the prediction capabilities of the system. The approach is motivated by the assumption that there is a relationship between the importance of each node and its output weights $\mathbf{W}_{out}$ assigned in the training process. Following the findings of Albert *et al.*[23] for networks, one would assume that the removal ("attack") of important nodes (with high $\mathbf{W}_{out}$ values) has a large negative impact on the "response" of the reservoir to input data, i.e., on the prediction quality. On the other hand, the removal of unimportant nodes (with low $\mathbf{W}_{out}$ values) should not alter the prediction too much. To test this assumption, we remove a fraction p of the $N = 200$ nodes, which correspond to certain values—e.g., the largest or smallest—of $\mathbf{W}_{out}$. However, each node is affiliated not only with one but D output weights, where D denotes the dimensionality of the system that is being predicted. Hence, we sort $\mathbf{W}_{out}$ based on the largest absolute value of all D output weights for each node in order to determine which nodes should be removed. After removal, we train the newly obtained reduced network again. This leads to a new set of $\mathbf{W}_{out}$. As a consequence, the new reservoir is not only reduced in size but also altered in its spectral radius, degree distribution, and the distribution of $\mathbf{W}_{in}$. The node removal process is illustrated in Fig. 2.

In addition to changes to the structure of the reservoir network outlined above, we study the effect of nonlinearity of the activation function. This has well-known effects on the memory of the reservoir.[24–27] However, in the present study, we focus on systems where the role of memory is small. To do this, we introduce a nonlinear scaling factor a in the hyperbolic tangent of the activation function to further improve prediction quality. This changes the

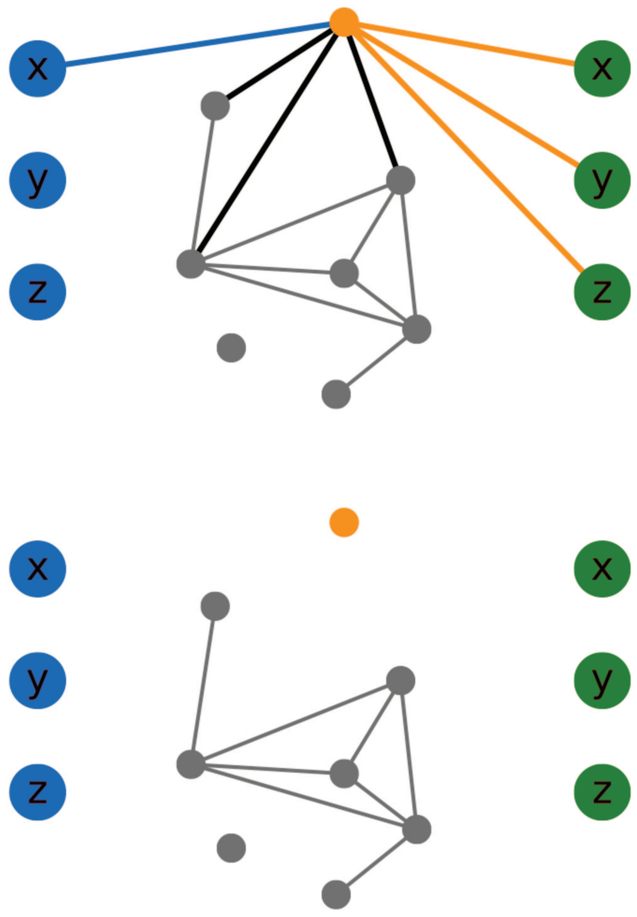

FIG. 2. Schematic illustration of the controlled node removal. The top graphic shows the initial network before the removal procedure. Here, the orange example node is fed only with input (blue) from the x dimension of the system. The orange lines denote its contribution to the output values of all three dimensions. We assign the relevant weight by taking the maximum of the absolute value of these three weights. The black lines represent connections to other nodes. Input/output interactions of the other nodes are not shown here. In the bottom plot, the example node has been removed, and, therefore, all connections and interactions vanished.

update equation for $\mathbf{r}(t)$ to

$$\mathbf{r}(t+\Delta t) = \tanh(a[\mathbf{A}\mathbf{r}(t) + \mathbf{W}_{in}\mathbf{P}\mathbf{r}(t)]). \qquad (6)$$

The nonlinearity of the activation function is a crucial property for reservoir computing. Because both the reservoir itself and the output function are linear, the activation function is the only source of nonlinearity in the system. The introduction of a scaling factor in the argument can be interpreted as varying the degree of this nonlinearity. Equivalently, it can be seen as simply tuning the scale for $\mathbf{W}_{in}$ and the spectral radius of $\mathbf{A}$ simultaneously. Thus, the effective number of parameters stays the same. However, due to its relation to the activation function, it is interesting to study the isolated effect of the scaling, while fixing the other parameters.

C. Measures and system characteristics

1. Forecast horizon

To quantify the quality and duration of the exact prediction of the trajectory, we use a fairly simple measure, which we call *forecast horizon*. It is defined as the number of time steps while the predicted and the actual trajectory are matching. As soon as one of the D coordinates exceeds certain deviation thresholds, we consider the trajectories as not matching anymore. Throughout our study we use

$$|\mathbf{v}(t) - \mathbf{v}_R(t)| > \boldsymbol{\delta}, \qquad (7)$$

where the thresholds are $\boldsymbol{\delta} = (5.8, 8.0, 6.9)^T$ for the Lorenz system. In general, the values are chosen based on the different ranges of the state variables and correspond to 15% of the spatial extent of the attractor. The aim is that small fluctuations around the actual trajectory as well as minor detours do not exceed the threshold. Empirically, we found that distances between the trajectories become much larger than the threshold values as soon as short-term prediction collapses. A similar measure has been proposed using a normalized L2 norm.[7] However, when dealing with data, which show significant differences in spatial extent between dimensions (e.g., the Chua circuit), this weighted approach is advantageous.

2. Correlation dimension

The structural complexity of a dynamical system is an important characteristic of its long-term properties. This can be quantified by its correlation dimension, where we measure the dimensionality of the space populated by the trajectory.[28] The correlation dimension is based on the correlation integral

$$C(r) = \lim_{N\to\infty} \frac{1}{N^2} \sum_{i,j=1}^{N} \theta(r - |\mathbf{x}_i - \mathbf{x}_j|)$$
$$= \int_0^r d^3r' c(\mathbf{r}'), \qquad (8)$$

which describes the mean probability that two states in phase space are close to each other at different time steps. The condition *close to* is met if the distance between the two states is less than the threshold distance r. θ represents the Heaviside function while $c(\mathbf{r}')$ denotes the standard correlation function. For self-similar strange attractors, the following power-law relationship holds in a range of r:

$$C(r) \propto r^{\nu}. \qquad (9)$$

The *correlation dimension* is then measured by the scaling exponent ν. We use the Grassberger Procaccia algorithm[29] to calculate the correlation dimension of our trajectories. The scaling region is derived from the data itself as $r \in [0.5, 2.5] * s_r$, where the trajectory dependent scaling factor s_r is defined as $s_r = \overline{\sigma(\mathbf{u})}/8.5$. Thus, the scaling region depends on the standard deviation σ of the input data $\mathbf{u}$. This approach is purely data driven and, therefore, does not require any knowledge about the system.

3. Largest Lyapunov exponent

Apart from the structural properties, the temporal complexity of a system is another crucial feature of its so-called long-term climate. For chaotic systems, the analysis of the Lyapunov exponents is the most suitable approach for quantifying this. The Lyapunov exponents λ_i describe the average rate of divergence of nearby points in phase space and thus measure sensitivity to initial conditions. For each dimension in phase space, there is one exponent. If the system exhibits at least one positive Lyapunov exponent, it is classified as chaotic, while the magnitude of the exponent quantifies the time scale on which the system becomes unpredictable.[30,31] Therefore, it is sufficient for our analysis to determine only the largest Lyapunov exponent λ_1,

$$d(t) = C e^{\lambda_1 t}. \tag{10}$$

This makes the task computationally much easier than calculating the full Lyapunov spectrum. Here, $d(t)$ is the average distance or separation of the initially nearby states at time t and C is a constant that normalizes this initial separation. To calculate the largest Lyapunov exponent, we use the Rosenstein algorithm without requiring temporal separation of neighbors.[32]

D. Modified Lorenz system

As an example for replicating chaotic attractors, we apply reservoir computing to the Lorenz system.[33] It has been developed as a simplified model for atmospheric convection and exhibits chaos for certain parameter ranges. The standard Lorenz system, however, is symmetric in x and y with respect to the transformation $x \to -x$ and $y \to -y$. In order to study a more general example, we would like to modify the Lorenz system such that this symmetry is broken. This can be achieved by adding the term x to the z-component such that the equations read as

$$\begin{aligned}\dot{x} &= \sigma(y - x),\\ \dot{y} &= x(\rho - z) - y,\\ \dot{z} &= xy - \beta z + x.\end{aligned} \tag{11}$$

This system is called the modified Lorenz system. We utilize the standard parameters for its chaotic regime $\sigma = 10$, $\beta = 8/3$, and $\rho = 28$ and solve the equations using the fourth order Runge–Kutta method with a time resolution $\Delta t = 0.02$.

In addition to the Lorenz system, we run the analysis in Sec. III B also for a number of other nonlinear dynamical systems[34–40] from the class of autonomous dissipative flows, such as the Rössler system,[34] Rabinovich–Fabrikant equations,[37] Rucklidge system,[40] Halvorsen cyclically symmetric system,[41] and the Chua circuit.[38] All these systems are $D = 3$ dimensional but differ in properties like Lyapunov exponents, correlation dimension, size of the attractor, and the nature of their nonlinearity. The parameters for all systems except Lorenz and Rössler are taken from the textbook *Chaos and Time-Series Analysis* by Sprott.[41]

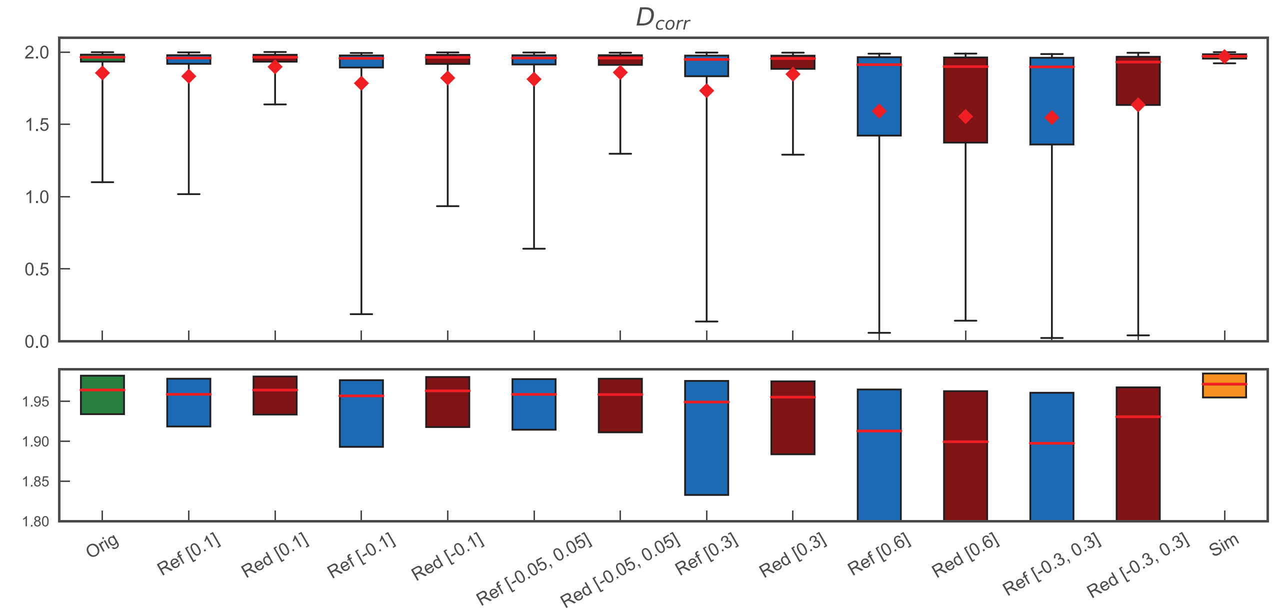

FIG. 3. Boxplot of the correlation dimension for $N = 500$ realizations for the original setup (green—left) and different percentages of nodes removed. Positive numbers (e.g., Red [0.1]) represent a removal of the 10% largest $\mathbf{W}_{out}$ nodes, while negative numbers (e.g., Red [−0.1]) denote a removal of the 10% smallest $\mathbf{W}_{out}$ nodes. Consequently, Red [−0.05, 0.05] stands for the nodes with the 5% largest and smallest $\mathbf{W}_{out}$ values removed symmetrically. *Red* are the results for the system after node removal, while *Ref* represents a smaller reference network. The yellow box on the right represents the error of the correlation dimension calculated from $N = 500$ simulated trajectories. The boxes represent the 25%–75% percentile range, while the extended lines denote the 5% and 95% percentile, respectively. Red bars are indicating the mean values and red dots show the median. In order to make a comparison easier, the bottom plot gives a zoomed in view of the 25%–75% percentile boxes and the respective median values.

III. RESULTS

In our previous study,[13] we showed that there is a strong variability in prediction quality by running the same setup with multiple different random realizations of the reservoir. In order to quantify the quality of a prediction, we analyzed both the exact short-term prediction horizon and the reproduction of the long-term climate of the system as measured by the correlation dimension and the largest Lyapunov exponent. Our aim is now to reduce this variability by applying the controlled node removal procedure and introducing an optimal choice for the nonlinear scaling parameter a in the activation function as introduced in Sec. II B.

A. Controlled node removal

After showing that changing the overall network topology, e.g., by using small-world or scale-free networks, does not lead to better predictions,[13] we now focus on the differential properties of the reservoir. To do this, we carry out the controlled node removal procedure as introduced in Sec. II B. Here, we stick to the default setup by setting the nonlinear scaling parameter to $a = 1$ and, therefore, do not rescale the activation function in order to separate effects.

Figure 3 shows a boxplot of the correlation dimension for $N = 500$ realizations and compares the results for the original system (green box on the left) to those after different steps of node removal. In addition, the yellow box on the right shows the error of the correlation dimension calculated from $N = 500$ simulated trajectories with different initial conditions. The boxes represent the 25%–75% percentile range while the extended lines denote the 5% and 95% percentile, respectively. Furthermore, the median values are indicated by the red bars while the red dots show the mean values. The labels on the x axis are defined in the following way: *Red [x]* denotes the results for the system after removing the nodes corresponding to the largest x% of the output weights if $x > 0$ and smallest x% if $x < 0$. Both positive and negative values at the same time mean that we symmetrically remove nodes from both "sides." In contrast, the results shown for the *Ref [x]* labels are reference reservoirs, which are initially constructed and trained with less nodes and calibrated to the same spectral radius as the reservoirs after the node removal procedure.

The results indicate that removing the nodes that correspond to the largest 10% of the output weights—*Red [0.1]*—improves the prediction quality compared to the original setup—*Orig*. In particular, the mean of the correlation dimension improves to 1.89 compared to 1.85 in the default setup, while the median stays at 1.96. The values of the simulated system are 1.97 and 1.97. This means that predominantly bad predictions have been enhanced. Moreover, the 5% percentile significantly increases from around 1 to 1.6. This indicates a lower number of outliers, where the reproduction of the correlation dimension did not work. As this reduced reservoir now effectively only has 180 nodes, it is interesting to analyze how a reservoir computing setup performs, which is initialized with only 180 nodes. We can see in Fig. 3 that for *Ref [0.1]*, the reproduction of the correlation dimension becomes slightly worse as compared to the default setup with $D_r = 200$ nodes. This means that the improvement due to the controlled node removal is not due to the changed reservoir size but driven by the altered properties of the system. In contrast, removing nodes corresponding to the smallest 10% of the output weights has a slightly negative effect on the prediction quality. However, the results are still better than those of its reference system with the same spectral radius and only 180 nodes initially. Finally, we symmetrically removed the nodes corresponding to the smallest and largest 5% of the output weights. As for the first case, the prediction quality improves compared to the default system and the performance is again better than its reference system.

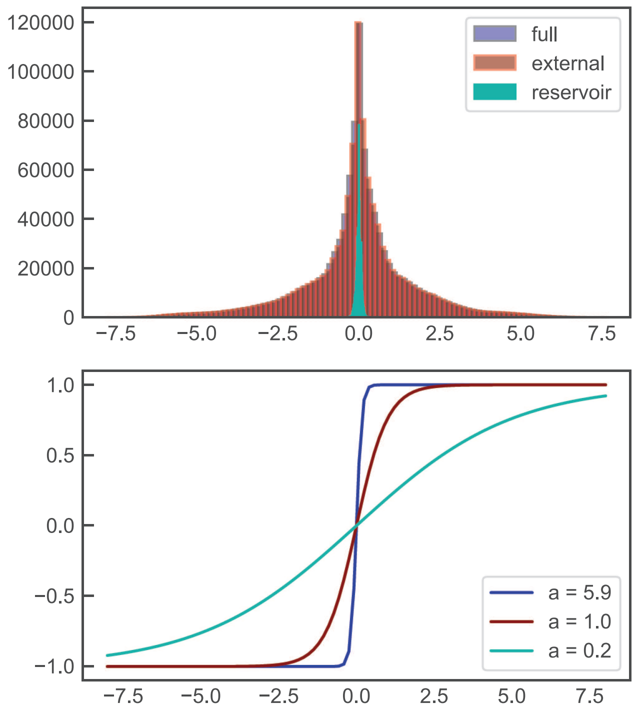

FIG. 4. Top plot: Distribution of the arguments of the activation function during training period split into the contribution from the reservoir (green), the input term (red), and total (blue). Bottom plot: Hyperbolic tangent for different nonlinear scaling factors.

Naturally the question arises, how results change if we remove more than 10% of the nodes and if it is possible to achieve comparable performance for smaller reservoir computing systems than the original setup with $D_r = 200$ nodes. Thus, we calculated the results for removing up to 60% of the nodes and, therefore, significantly reduced network sizes. As a first step, we increase the percentage of removed nodes to those associated with the largest 30% of the output weights. We can see that the performance is comparable to the larger original system while the number of outliers is still reduced. This can be observed by the shorter length of the black line. Moreover, this also holds for the mean and median values. Those are 1.85 and 1.96, respectively, for the reduced system and 1.86 and 1.96 for the original system. In addition, we also ran the same analysis for the nodes belonging to the smallest 30% of output weights. Again, this leads to significantly worse results than excluding nodes with large weights.

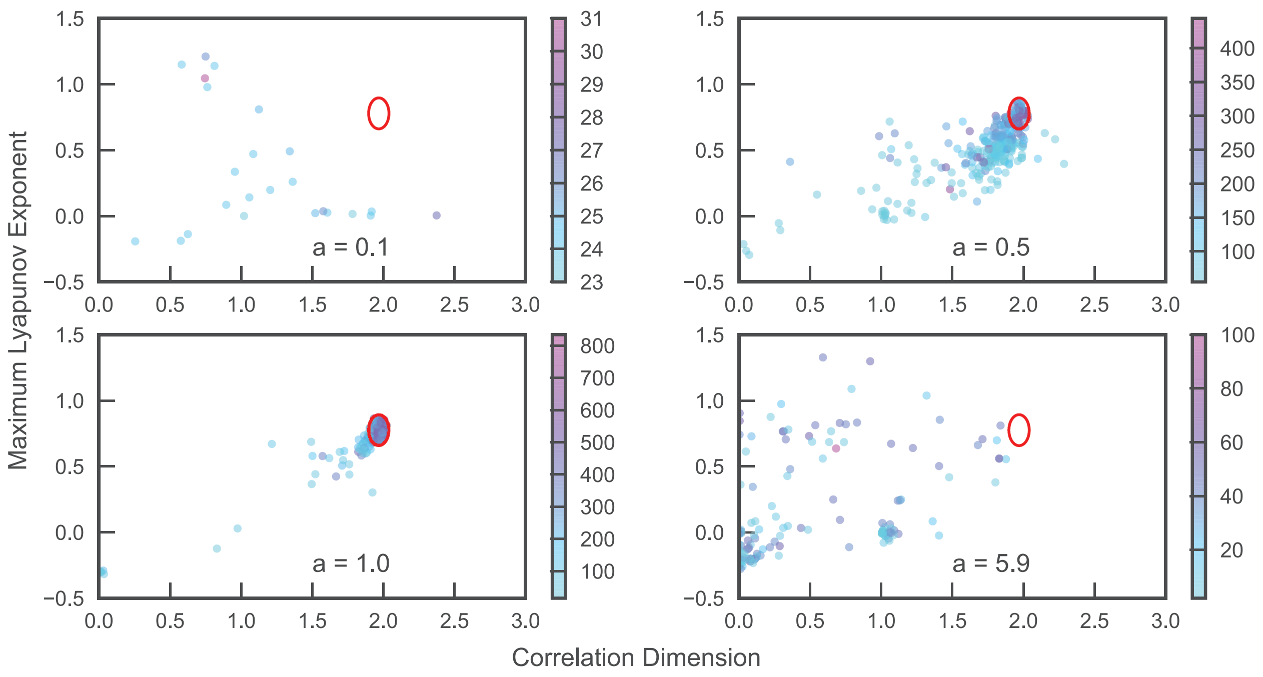

FIG. 5. Largest Lyapunov exponent scattered against correlation dimension for different values of the nonlinear scaling parameter *a* based on $N = 300$ realizations each. The colors denote the forecast horizon of the predictions and the red ellipses show the three σ errors of the correlation dimension ($\sigma = 0.024$) and the largest Lyapunov exponent ($\sigma = 0.039$) calculated from simulations of the actual system.

In contrast to the improvements in reproducing the correlation dimension seen for the 10% and 30% cases, removing the nodes corresponding to the largest 60% of the output weights clearly leads to lower prediction quality and a higher number of bad realizations. The same can be observed for removing those nodes based on the smallest 60% of the output weights, which is not shown here. It is interesting to note that in both cases, the initially reduced reference system now performs better than in those cases, where a lower percentage of nodes has been removed. Furthermore, symmetrically removing the nodes reflecting both the largest and smallest 30% of the output weights leads to better results than removing 60% of either the largest or smallest. In addition, the results now outperform those of the reference system. While the overall prediction quality is notably worse than for the default system, it is very interesting to notice that it is possible to still achieve good prediction results with a significantly downscaled system. This can be very beneficial when applications are computationally more challenging, e.g., when the dimensionality of the dynamical system is high or the trajectories are very long. Moreover, reducing the network size is important when it comes to hardware implementations of reservoir computing, such as neuromorphic computing.[42] To make this more practicable, Griffith *et al.*[43] proposed very simple reservoir designs with low connectivity. In contrast, our approach reduces the number of nodes D_r and, therefore, could add additional benefits for hardware implementations.

Instead of calculating the correlation dimension, we ran the same analysis also based on the forecast horizon of the predictions. As the results look very similar to those of the correlation dimension, they are not shown here.

B. Prediction variability and nonlinear scaling parameter

As a next step, we focus on the activation function and examine the effect of different choices for the nonlinear scaling parameter *a*. The upper plot of Fig. 4 shows the distribution of the arguments of the hyperbolic tangent activation function during the training period. While the green area shows the influence of the reservoir term $\mathbf{A}\mathbf{r}(t)$, the red area represents the impact of the external input $\mathbf{W}_{in}\mathbf{u}(t)$. One can clearly see that the values of the reservoir term are very small compared to those of the external input. In commonly used parameterizations of reservoir computing, the value for the $\mathbf{W}_{in}$ scale is 1—this means that the weights of the input function are uniformly distributed between -1 and 1. However, our hyperparameter optimization in Sec. II A led to a $\mathbf{W}_{in}$ scale of 0.17, and, therefore, we can approximately say that the input scale of 1 in the standard parameterization is equivalent to a value of $a = 5.9$ in our setup, ignoring the comparably small influence of the reservoir term. If we compare the scale of the distribution to the functional form of the hyperbolic tangent in the lower plot, it becomes clear that for $a = 5.9$, the majority of points lies in the saturation regime of the function. Intuitively, one can expect that the best results can be achieved, if *a* is chosen such that a large part of the distribution of the input arguments lies within the “dynamical” range of the hyperbolic tangent rather than in its saturation regime. Low values of a, however, would lead to an approximately linear behavior of the function and would thus not allow the system to adequately capture the nonlinear dynamics of the input data.

In order to test this assumption empirically, we simulated $N = 300$ realizations for different values of a. We then evaluated the forecast horizon as well as the long-term climate for each realization. The bottom right plot in Fig. 5 shows the largest Lyapunov exponent scattered against the correlation dimension for the modified Lorenz system. The results are based on the above described default setup with the nonlinear scaling factor set to $a = 5.9$. The red ellipse shows the three σ errors of the correlation dimension and the largest Lyapunov exponent. Those are calculated from simulations of the actual equations of the Lorenz system for $N = 500$ different initial conditions. We can clearly see that for $a = 5.9$, all points are widely spread outside this error ellipse and are, therefore, to be classified as bad predictions. This is because they do not well resemble the long-term climate. While some realizations lead to meaningful values for the largest Lyapunov exponent, the correlation dimension is badly reconstructed in particular.

To find the optimal value for a, we systematically analyzed multiple realizations for a number of different values of a between 0 and 10. This is shown in Fig. 6, where the blue points correspond to the forecast horizon of the single realizations. In addition, the red and green dots represent the average and median value across all realizations for a given value of a. We then determine the optimal value for a such that the average is maximized. This leads to an optimal value of around $a = 1.0$, which is in line with our expectation, given that we carried out a hyperparameter optimization in the beginning. For validating the above arguments, we turn back to Fig. 5.

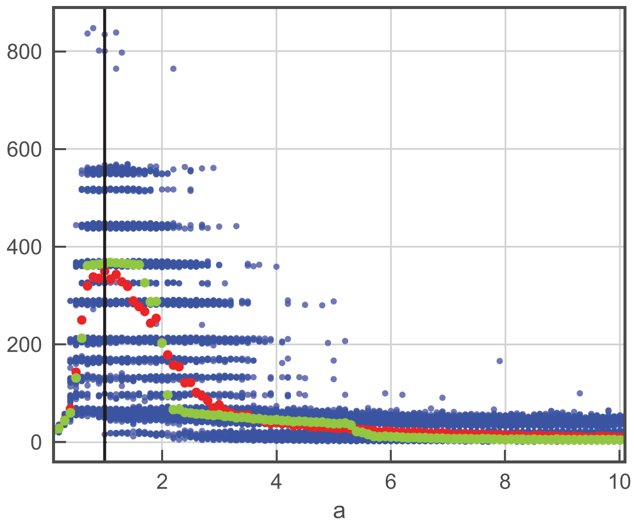

FIG. 6. Forecast horizon of the modified Lorenz system plotted for different values of the nonlinear scaling parameter a with $N = 300$ realizations for each a (blue). Furthermore, the average (red) and the median (green) values are shown for each value of a. The black horizontal line marks the optimal choice for a.

The bottom left plot shows the results for the optimal choice of a, where many outliers and thus bad predictions disappeared. Moreover, there is now a compact cloud of points around the error ellipse, and, therefore, the overall prediction quality is significantly better as compared to the case $a = 5.9$ in the bottom right plot. In contrast, setting $a = 0.1$ and $a = 0.5$ as shown in the top plots leads to a complete breakdown of the prediction ability of the system. The reason that one can see only a few points in the top left plot is the following. The prediction quality for $a = 0.1$ completely collapses in most cases such that we obtain *NaN* results for our calculations of the largest Lyapunov exponent. This happens in cases where the prediction jumps between multiple points in a cyclical fashion. Consequently, this leads to division by zero and generally only occurs for unsuitable parameter choices—in this case for too small values of a. As both examples in the top plots correspond to arguments of the activation function being in the linear regime of the hyperbolic tangent, this demonstrates that nonlinearity in the activation function is essential for predicting complex nonlinear systems. Besides the results for the reproduction of the long-term climate, we also show the forecast horizon encoded in the colors of the points. Equivalently, the longest forecast horizon can be achieved by choosing the optimal value for a, whereas smaller or larger values both lead to worse results. Another interesting result is that realizations, which well resemble the long-term climate, have a higher forecast horizon than those failing to properly reconstruct the climate.

In addition to the Lorenz system, we carried out the same analysis for other nonlinear complex systems such as the Chua circuit,

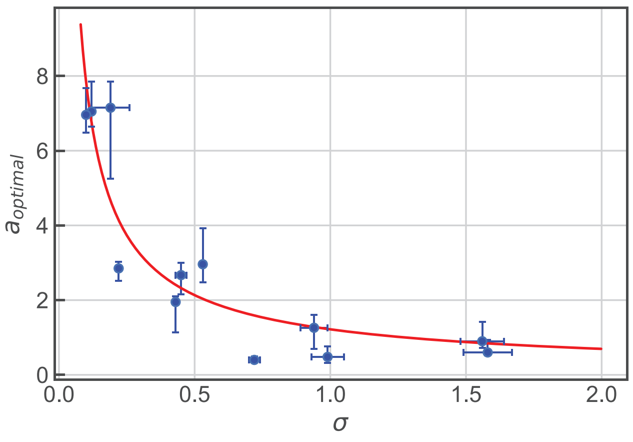

FIG. 7. Blue dots: Optimal value for the nonlinear scaling parameter a—based on the maximum of the average forecast horizon over all realizations for a given a—plotted against the standard deviation σ of the input data. The vertical bars denote the range of values for a, where the average forecast horizon is up to 10% lower than for the optimal a, while the horizontal bars represent the standard error. The red line represents a fit through the points. Dynamical systems from left to right: Rabinovich–Fabrikant equations, Chua circuit, Complex Butterfly attractor, Thomas' cyclically symmetric attractor, Rössler system, Rucklidge system, Halvorsen model, *blended system:* 0.4 * Modified Lorenz system + Halvorsen Model, *blended system:* 0.5 * Modified Lorenz system + 3 * Rabinovich–Fabrikant equations, *blended system:* Rössler + 2*Rucklidge system, Modified Lorenz system, and Chen system.

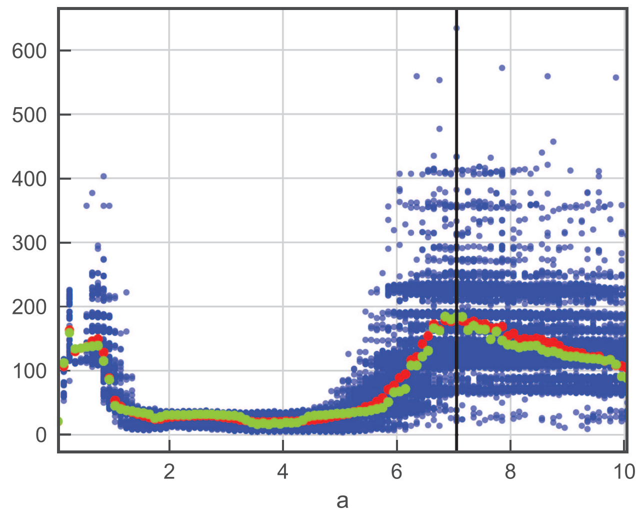

FIG. 8. Forecast horizon of the Chua circuit plotted for different values of the nonlinear scaling parameter *a* with $N = 300$ realizations for each *a* (blue). Furthermore, the average (red) and the median (green) values are shown for each value of *a*. The black horizontal line marks the optimal choice for *a*.

the Rössler system, and other autonomous dissipative flows as summarized in Sec. III B. Figure 7 shows the results for their optimal values of *a* scattered against the standard deviation of the input. In addition, we also constructed combinations of the systems used, in order to fill the gap in between the standard deviations of the Halvorsen model (0.53) and the modified Lorenz system (1.56). We can clearly see that there is a relationship between the optimal *a* and the input standard deviations. This makes intuitively sense, since the dynamical regime of the hyperbolic tangent needs to be at a different range for different distributions. Surprisingly, this seems to dominate effects of other system-specific properties. Therefore, as a rule of thumb, the optimal value for the nonlinear scaling parameter is given by $a_{opt} = c/\sigma(\mathbf{W}_{in}\mathbf{u})^b$ with $b = 0.80$ and $c = 1.22$ determined by the fitted red curve. This provides a good starting point for the hyperparameter optimization. However, it is always recommended to run a system-specific analysis as shown in Fig. 6. On the example of the Chua circuit, it turns out that good predictions cannot only be achieved by values for *a* close to the result given by the above formula for a_{opt}. However, among those systems, the Chua circuit yields optimal predictions not only for $a = 7.05$ following the above introduced rule of thumb, but also shows another peak for small values of around $a = 0.75$ as shown in Fig. 8. This might be related to the fact that the equations of the Chua attractor only have a local nonlinearity at $x = \pm 1$, making the linear regime very successful anywhere else. We also looked at a larger parameter range for *a* and found that the average forecast horizon is monotonically declining for values of $a > 10$, which are not shown here. Equivalent results for a_{opt} are gained by carrying out the same analysis for the above mentioned systems based on the reproduction of the correlation dimension. In particular, the results for the Chua circuit indicate that there is a significant potential for system specific optimizations.

IV. CONCLUSIONS AND OUTLOOK

In this paper, we used reservoir computing to predict and reconstruct attractors for chaotic systems such as the Lorenz system. While other recurrent neural network based approaches often tend to be a black box, the architecture of reservoir computing is simple enough that a systematic analysis of driving properties for good predictions should be possible. The reason is that the reservoir network itself is static, and, therefore, predictions are deterministic and depend strongly on output weights once trained. Knowing this, we made alterations to the reservoir network structure by removing nodes and their respective edges based on their weights in the output function. This was motivated by two aims: first, understanding how the prediction quality depends on differential properties of the system and, second, investigating by how much a reservoir computing setup can be reduced while still delivering sufficient prediction performance. We found that removing the nodes associated with the largest 10% of the output weights improves the replication of the climate of the Lorenz system and reduces variability in prediction quality. This is somewhat counterintuitive, as large weights in the output function suggest a strong influence of the respective node in the aggregation of the (correct) output signal. These findings have to be rather interpreted in the sense that some connections from the nodes with the largest output weights obviously impede the reservoir operations and lead to worse predictions. Further research is needed to unveil the relevance and the impact of connections within the reservoir on the prediction results. Furthermore, it turned out that by applying the node removal framework, the network size can be reduced by more than 30% at comparable prediction quality and up to 60% while still delivering reasonable performance. This could be helpful when it comes to hardware implementations of the reservoir, as, for example, neuromorphic computing.[42]

Moreover, we varied the scaling of the hyperbolic tangent activation function. We showed that for widely used parameterizations of reservoir computing, a high fraction of the arguments of the activation function is in the saturation regime of the hyperbolic tangent. This leads to high variability and bad prediction quality, as the system cannot adequately grasp the input dynamics. By tuning the scale of the activation function, this problem can be addressed much more conveniently and intuitively than by varying the spectral radius and W_{in} scale separately. We found a relationship between the optimal choice of *a* and the standard deviation of the input, that can serve as a rule of thumb and provide a good starting point for a complete hyperparameter optimization. At the same time, a system-specific analysis and optimization of the nonlinear scaling parameter can unveil interesting results. An example for this was presented for the Chua circuit, where we found not only one peak for the optimal value of *a* but another—much smaller—regime where good predictions can be achieved. We showed that a description of the dependency of the optimal *a* on the standard deviation of the input of the activation function does not only hold for the Lorenz system but for other complex nonlinear systems as well.

Our results demonstrate that a large optimization potential lies in a systematical refinement of the differential reservoir properties for a given dataset. This was outlined on the examples of controlled node removal and introduction of a scaling factor in the activation function. Future research will focus on deepening the understanding of how other differential properties of the reservoir affect the quality of the predictions, with the aim to identify an optimal reservoir in terms of (minimal) size, (best) prediction quality, and (highest) statistical robustness.

ACKNOWLEDGMENTS

We wish to acknowledge useful discussions and comments from Sebastian Baur, Youssef Mabrouk, and Mierk Schwabe.

DATA AVAILABILITY

The data that support the findings of this study are available from the corresponding author upon reasonable request.

REFERENCES

[1] H. Jaeger and H. Haas, "Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication," Science **304**, 78–80 (2004).
[2] H. Jaeger, "The "echo state" approach to analysing and training recurrent neural networks-with an erratum note," German National Research Centre for Information Technology Report No. 138, Bonn, Germany, 2001, p. 13.
[3] W. Maass, T. Natschläger, and H. Markram, "Real-time computing without stable states: A new framework for neural computation based on perturbations," Neural Comput. **14**, 2531–2560 (2002).
[4] Z. Lu, J. Pathak, B. Hunt, M. Girvan, R. Brockett, and E. Ott, "Reservoir observers: Model-free inference of unmeasured variables in chaotic systems," Chaos **27**, 041102 (2017).
[5] J. Pathak, Z. Lu, B. R. Hunt, M. Girvan, and E. Ott, "Using machine learning to replicate chaotic attractors and calculate lyapunov exponents from data," Chaos **27**, 121102 (2017).
[6] J. Pathak, B. Hunt, M. Girvan, Z. Lu, and E. Ott, "Model-free prediction of large spatiotemporally chaotic systems from data: A reservoir computing approach," Phys. Rev. Lett. **120**, 024102 (2018).
[7] J. Pathak, A. Wikner, R. Fussell, S. Chandra, B. R. Hunt, M. Girvan, and E. Ott, "Hybrid forecasting of chaotic processes: Using machine learning in conjunction with a knowledge-based model," Chaos **28**, 041101 (2018).
[8] R. S. Zimmermann and U. Parlitz, "Observing spatio-temporal dynamics of excitable media using reservoir computing," Chaos **28**, 043118 (2018).
[9] T. L. Carroll, "Using reservoir computers to distinguish chaotic signals," Phys. Rev. E **98**, 052209 (2018).
[10] Z. Lu, B. R. Hunt, and E. Ott, "Attractor reconstruction by machine learning," Chaos **28**, 061104 (2018).
[11] P. Antonik, M. Gulina, J. Pauwels, and S. Massar, "Using a reservoir computer to learn chaotic attractors, with applications to chaos synchronization and cryptography," Phys. Rev. E **98**, 012215 (2018).
[12] N. A. K. Doan, W. Polifke, and L. Magri, "A physics-aware machine to predict extreme events in turbulence," arXiv:1912.10994 (2019).
[13] A. Haluszczynski and C. Räth, "Good and bad predictions: Assessing and improving the replication of chaotic attractors by means of reservoir computing," Chaos **29**, 103143 (2019).
[14] T. L. Carroll and L. M. Pecora, "Network structure effects in reservoir computers," arXiv:1903.12487 (2019).
[15] X. Dutoit, B. Schrauwen, J. Van Campenhout, D. Stroobandt, H. Van Brussel, and M. Nuttin, "Pruning and regularization in reservoir computing," Neurocomputing **72**, 1534–1546 (2009).
[16] S. Scardapane, G. Nocco, D. Comminiello, M. Scarpiniti, and A. Uncini, "An effective criterion for pruning reservoir's connections in echo state networks," in *2014 International Joint Conference on Neural Networks (IJCNN)* (IEEE, 2014), pp. 1205–1212.
[17] S. Scardapane, D. Comminiello, M. Scarpiniti, and A. Uncini, "Significance-based pruning for reservoir's neurons in echo state networks," in *Advances in Neural Networks: Computational and Theoretical Issues* (Springer, 2015), pp. 31–38.
[18] M. Lukoševičius and H. Jaeger, "Reservoir computing approaches to recurrent neural network training," Comput. Sci. Rev. **3**, 127–149 (2009).
[19] R. Gençay and T. Liu, "Nonlinear modelling and prediction with feedforward and recurrent networks," Physica D **108**, 119–134 (1997).
[20] P. Erdos, "On random graphs," Publ. Math. **6**, 290–297 (1959).
[21] A. E. Hoerl and R. W. Kennard, "Ridge regression: Biased estimation for nonorthogonal problems," Technometrics **12**, 55–67 (1970).
[22] The search ranges for the hyperparameter optimization are 0–2.5 for $\rho(\mathbf{A})$, 0–2.0 for $\mathbf{W}_in$, and $\log_e 10^-11$–$\log_e 0.1$ for β, which has been scaled onto a logarithmic scale for better coverage of small values.
[23] R. Albert, H. Jeong, and A.-L. Barabási, "Error and attack tolerance of complex networks," Nature **406**, 378–382 (2000).
[24] M. Inubushi and K. Yoshimura, "Reservoir computing beyond memory-nonlinearity trade-off," Sci. Rep. **7**, 1–10 (2017).
[25] A. Goudarzi, A. Shabani, and D. Stefanovic, "Exploring transfer function nonlinearity in echo state networks," in *2015 IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA)* (IEEE, 2015), pp. 1–8.
[26] D. Verstraeten, B. Schrauwen, M. d'Haene, and D. Stroobandt, "An experimental unification of reservoir computing methods," Neural Netw. **20**, 391–403 (2007).
[27] P. Verzelli, C. Alippi, and L. Livi, "Echo state networks with self-normalizing activations on the hyper-sphere," Sci. Rep. **9**, 1–14 (2019).
[28] P. Grassberger and I. Procaccia, "Measuring the strangeness of strange attractors," Physica D **9**, 189–208 (1983).
[29] P. Grassberger, "Generalized dimensions of strange attractors," Phys. Lett. A **97**, 227–230 (1983).
[30] A. Wolf, J. B. Swift, H. L. Swinney, and J. A. Vastano, "Determining lyapunov exponents from a time series," Physica D **16**, 285–317 (1985).
[31] R. Shaw, "Strange attractors, chaotic behavior, and information flow," Z. Naturforsch. A **36**, 80–112 (1981).
[32] M. T. Rosenstein, J. J. Collins, and C. J. De Luca, "A practical method for calculating largest lyapunov exponents from small datasets," Physica D **65**, 117–134 (1993).
[33] E. N. Lorenz, "Deterministic nonperiodic flow," J. Atmos. Sci. **20**, 130–141 (1963).
[34] O. E. Rössler, "An equation for continuous chaos," Phys. Lett. A **57**, 397–398 (1976).
[35] A. S. Elwakil, S. Ozoguz, and M. P. Kennedy, "Creation of a complex butterfly attractor using a novel Lorenz-type system," IEEE Trans. Circuits Syst. I Fundam. Theory Appl. **49**, 527–530 (2002).
[36] G. Chen and T. Ueta, "Yet another chaotic attractor," Int. J. Bifurcation Chaos **9**, 1465–1466 (1999).
[37] M. Rabinovich and A. Fabrikant, "Stochastic self-modulation of waves in nonequilibrium media," J. Exp. Theor. Phys. **77**, 617–629 (1979).
[38] T. Matsumoto, L. Chua, and M. Komuro, "The double scroll," IEEE Trans. Circuits Syst. **32**, 797–818 (1985).
[39] R. Thomas, "Deterministic chaos seen in terms of feedback circuits: Analysis, synthesis, "Labyrinth chaos"," Int. J. Bifurcation Chaos **9**, 1889–1905 (1999).
[40] A. M. Rucklidge, "Chaos in models of double convection," J. Fluid Mech. **237**, 209–229 (1992).
[41] J. C. Sprott and J. C. Sprott, *Chaos and Time-Series Analysis* (CiteSeerX, 2003), Vol. 69.
[42] G. Tanaka, T. Yamane, J. B. Héroux, R. Nakane, N. Kanazawa, S. Takeda, H. Numata, D. Nakano, and A. Hirose, "Recent advances in physical reservoir computing: A review," Neural Netw. **115**, 100–123 (2019).
[43] A. Griffith, A. Pomerance, and D. J. Gauthier, "Forecasting chaotic systems with very low connectivity reservoir computers," Chaos **29**, 123108 (2019).

C Controlling nonlinear dynamical systems into arbitrary states using machine learning

A. Haluszczynski and C. Räth, “Controlling nonlinear dynamical systems into arbitrary states using machine learning”, Scientific Reports **11**, 1–8 (2021)

scientific reports

OPEN

Controlling nonlinear dynamical systems into arbitrary states using machine learning

Alexander Haluszczynski[1,2]✉ & Christoph Räth[3]

Controlling nonlinear dynamical systems is a central task in many different areas of science and engineering. Chaotic systems can be stabilized (or chaotified) with small perturbations, yet existing approaches either require knowledge about the underlying system equations or large data sets as they rely on phase space methods. In this work we propose a novel and fully data driven scheme relying on machine learning (ML), which generalizes control techniques of chaotic systems without requiring a mathematical model for its dynamics. Exploiting recently developed ML-based prediction capabilities, we demonstrate that nonlinear systems can be forced to stay in arbitrary dynamical target states coming from any initial state. We outline and validate our approach using the examples of the Lorenz and the Rössler system and show how these systems can very accurately be brought not only to periodic, but even to intermittent and different chaotic behavior. Having this highly flexible control scheme with little demands on the amount of required data on hand, we briefly discuss possible applications ranging from engineering to medicine.

The possibility to control nonlinear chaotic systems into stable states has been a remarkable discovery[1,2]. Based on the knowledge of the underlying equations, one can force the system from a chaotic state into a fixed point or periodic orbit by applying an external force. This can be achieved based on the pioneering approaches by Ott et al.[1] or Pyragas[3]. In the former, a parameter of the system is slightly changed when it is close to an unstable periodic orbit in phase space, while the latter continuously applies a force based on time delayed feedback. There have been many extensions of those basic approaches (see e.g. Boccaletti et al.[4] and references therein) including "anti-control" schemes[5], that break up periodic or synchronized motion. However, all of them do not allow to control the system into well-specified, yet more complex target states such as chaotic or intermittent behavior. Further, these methods either require exact knowledge about the system, i.e. the underlying equations of motion, or rely on phase space techniques for which very long time series are necessary.

In recent years, tremendous progress has been made in the prediction of nonlinear dynamical systems by means of machine learning (ML). It has been demonstrated that not only exact short-term predictions over several Lyapunov times become possible, but also the long-term behavior of the system (its "climate") can be reproduced with unexpected accuracy[6–12]—even for very high-dimensional systems[13–15]. While several ML techniques have successfully been applied to time series prediction, reservoir computing (RC)[16,17] can be considered as the so far best approach, as it combines often superior performance with intrinsic advantages like smaller network size, higher robustness, fast and comparably transparent learning[18] and the prospect of highly efficient hardware realizations[19–21].

Combining now ML-based predictions of nonlinear systems with manipulation steps, we propose in this study a novel, fully data-driven approach for controlling nonlinear dynamical systems. In contrast to previous methods, this allows to obtain a variety of target states including periodic, intermittent and chaotic ones. Furthermore, we do not require the knowledge of the underlying equations. Instead, it is sufficient to record some history of the system that allows the machine learning method to be sufficiently trained. As previously outlined[22], an adequate learning requires orders of magnitude less data than phase space methods.

[1]Department of Physics, Ludwig-Maximilians-Universität, Schellingstraße 4, 80799 Munich, Germany. [2]Allianz Global Investors, risklab, Seidlstraße 24, 80335 Munich, Germany. [3]Institut für Materialphysik im Weltraum, Deutsches Zentrum für Luft- und Raumfahrt, Münchner Str. 20, 82234 Wessling, Germany. ✉email: alexander.haluszczynski@gmail.com

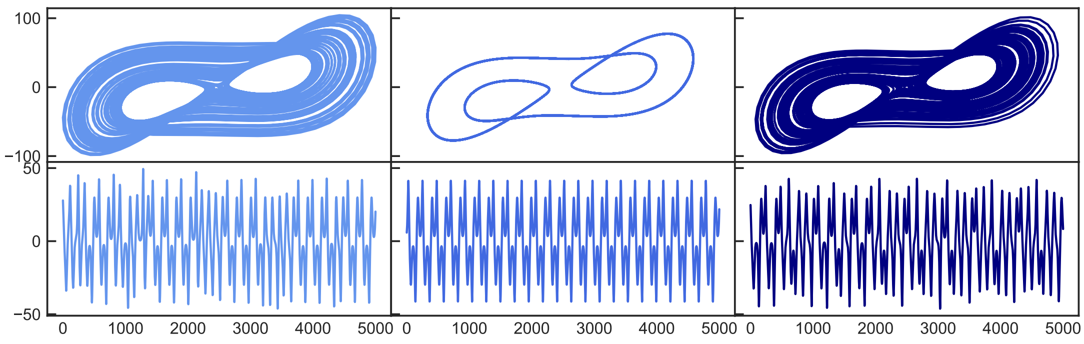

Figure 1. Periodic to chaotic control. Top: 2D attractor representation in the x–y plane. Bottom: X coordinate time series. Left plots show the original chaotic state which changes to a periodic state (middle) after tuning the order parameter. After applying the control mechanism, the system is forced into a chaotic state again (right).

Results

We define the situation that requires to be controlled in the following way: A dynamical system with trajectory **u** is in state **X**, which may represent e.g. periodic, intermittent or chaotic behavior. Then, the system behavior changes into another state **Y** as a consequence of order parameter changes or some uncontrollable external force. The aim of a control mechanism is now to push the system back into its original state **X**, while the cause for the initial change in state is still present. This can be achieved by deriving a suitable control force $\mathbf{F}(t)$ which is applied while the system is in state **Y**. Deriving $\mathbf{F}(t)$ requires the knowledge of how the trajectory $\mathbf{u}(t)$ of the system would have evolved if the system was still in state **X** instead. This 'what if' scenario can be obtained by training a suitable machine learning technique on past observations of the system while being in state **X**. In this study, this is achieved by using reservoir computing[23], which is a recurrent neural network based approach. In principle, any other prediction method could be used instead as long as it is able to deliver good predictions. Once trained and synchronized, it can create predictions $\mathbf{v}(t)$ of arbitrary length from which the control force $\mathbf{F}(t)$ is derived as

$$\mathbf{F}(t) = K(\mathbf{u}(t) - \mathbf{v}(t)), \tag{1}$$

where K scales the magnitude of the force. Since **F**(t) only depends on the (measured) coordinates **u**(t) and the ML prediction **v**(t), no mathematical model is required to control the system and thus the method is generally applicable as long as good predictions are available. The definition of the control force being dependent on the distance between the actual coordinate and a target coordinate is similar to what has been originally proposed by Pyragas[3]. However, in our case the control is not limited to periodic orbits but can achieve a variety of dynamical target states. A step by step description of the method is given in Section 0.2. The control of nonlinear dynamical system is studied on the example of the Lorenz system[24], which is a model for atmospheric convection. Depending on the choice of parameters, the system exhibits e.g. periodic, intermittent or chaotic behavior. The equations read

$$\dot{x} = \sigma(y - x); \quad \dot{y} = x(\rho - z) - y; \quad \dot{z} = xy - \beta z, \tag{2}$$

and $\boldsymbol{\pi} \equiv (\sigma, \rho, \beta)$ are the order parameters that lead to a certain state and the trajectory is thus described by $\mathbf{u}(t) = (x(t), y(t), z(t))^T$. First, we simulate the Lorenz system with parameters $\boldsymbol{\pi}$ such that we obtain the desired initial state **X**. Second, we train reservoir computing on the resulting trajectory until time step t_{train}. Then, the parameters are shifted to $\boldsymbol{\pi}^*$ such that the system behavior changes to state **Y** at time step t_{shift}. If $t_{shift} \geq t_{train}$, the RC system is synchronized accordingly with the trajectory since t_{train}. Synchronization means that the scalar states of the reservoir (see Eq. 5) are updated but the system is not re-trained. To control the system now back into state **X**, the correction force $\mathbf{F}(t)$ is derived in each time step based on the prediction $\mathbf{v}(t)$ and applied to the system by solving the differential equations of the system for the next time step including $\mathbf{F}(t)$

$$\mathbf{u}(t + \Delta t) = \int_t^{t+\Delta t} (\dot{f}(\mathbf{u}(\tilde{t}), \boldsymbol{\pi}^*) + \mathbf{F}(\tilde{t}))d\tilde{t}, \tag{3}$$

where $\dot{f}$ is defined in Eq. (2). The knowledge of $\dot{f}$ is only required for the model system examples in this study but not for real world applications. The equations are solved using the 4th order Runge–Kutta method with a time resolution $\Delta t = 0.02$. Since still the parameters $\boldsymbol{\pi}^*$ are used, the system would continue to exhibit the undesired state **Y** if the control force was 0. For the Lorenz system, the scaling constant set to $K = 25$. We did not optimize for K and empirically found that our method works for a wide range of choices. It is important to emphasize that a smaller choice for K does not necessarily mean that a smaller force is needed, because smaller values may allow for more separation of **u**(t) and **v**(t).

Figure 1 shows the results for the Lorenz system originally (left side) being in a chaotic state **X** ($\boldsymbol{\pi} = [\sigma = 10.0, \rho = 167.2, \beta = 8/3]$), which then changes to periodic behavior (middle) **Y** after ρ is changed to $\rho = 166$. Then, the control mechanism is activated and the resulting attractor again resembles the original chaotic state (left). While 'chaotification' of periodic states has been achieved in the past, the resulting attractor generally did not correspond to a certain specified target state but just exhibited some chaotic behavior. Since we would like to not only rely on a visual assessment, we characterize the attractors using quantitative

	Largest Lyapunov exponent λ			Correlation dimension ν		
	λ_{orig}	$\lambda_{changed}$	$\lambda_{controlled}$	ν_{orig}	$\nu_{changed}$	$\nu_{controlled}$
Periodic $\rightarrow$ *Chaotic*	0.851 ± 0.070	0.080 ± 0.075	0.841 ± 0.074	1.700 ± 0.065	1.052 ± 0.071	1.700 ± 0.061
Chaotic $\rightarrow$ *Intermittent*	0.571 ± 0.096	0.853 ± 0.053	0.614 ± 0.101	1.321 ± 0.086	1.678 ± 0.055	1.351 ± 0.091
$Chaotic_B \rightarrow Chaotic_A$	0.479 ± 0.060	0.643 ± 0.075	0.478 ± 0.067	1.941 ± 0.038	1.948 ± 0.047	1.933 ± 0.040
$Chaotic_D \rightarrow Chaotic_C$	0.819 ± 0.092	0.884 ± 0.058	0.822 ± 0.052	1.855 ± 0.069	1.959 ± 0.037	1.866 ± 0.050
Periodic $\leftarrow$ *Chaotic*	- 0.003 ± 0.012	0.844 ± 0.059	0.028 ± 0.110	1.001 ± 0.065	1.700 ± 0.071	1.001 ± 0.061
Chaotic $\leftarrow$ *Intermittent*	0.851 ± 0.070	0.550 ± 0.094	0.828 ± 0.067	1.700 ± 0.086	1.326 ± 0.055	1.698 ± 0.091
$Chaotic_B \leftarrow Chaotic_A$	0.629 ± 0.069	0.446 ± 0.068	0.629 ± 0.066	1.948 ± 0.037	1.939 ± 0.049	1.956 ± 0.037
$Chaotic_D \leftarrow Chaotic_C$	0.881 ± 0.092	0.836 ± 0.058	0.880 ± 0.052	1.958 ± 0.069	1.864 ± 0.038	1.951 ± 0.050

Table 1. Statistical simulation over $N = 100$ random realizations of the systems evaluated in terms of the mean values of the largest Lyapunov exponent and the correlation dimension with corresponding standard deviations. The subscript *orig* denotes the initial state of the system, while *changed* refers to the new state after parameters changed and *controlled* means the system controlled back into the original state. The description left to the arrow is the original state that also will be achieved again after controlling the system whereas the state written right to the arrow corresponds to the changed condition.

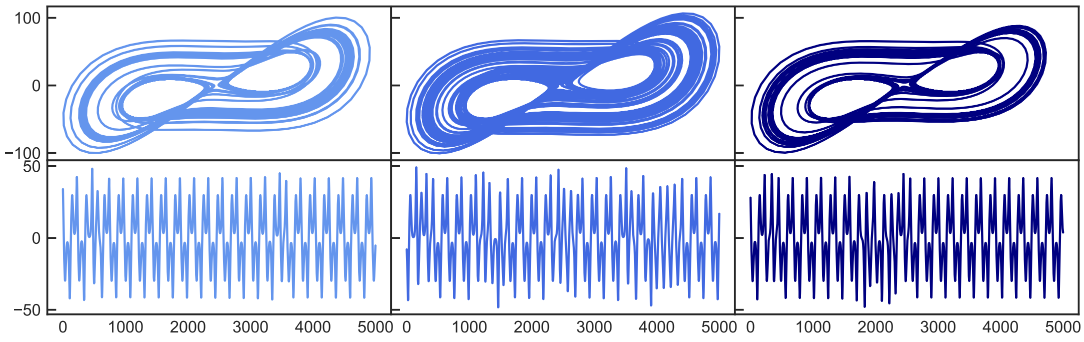

Figure 2. Chaotic to intermittent control. Top: 2D attractor representation in the x–y plane. Bottom: X coordinate time series. Left plots show the original intermittent state which changes to a chaotic state (middle) after tuning the order parameter. After applying the control mechanism, the system is forced into an intermittent state again (right).

measures. First, we calculate the largest Lyapunov exponent, which quantifies the temporal complexity of the trajectory, where a positive value indicates chaotic behavior. Second, we use the correlation dimension to assess the structural complexity of the attractor. Based on the two measures, the dynamical state of the system can be sufficiently specified for our analysis. Both techniques are described in the supporting information. Because a single example is not sufficiently meaningful, we perform our analysis statistically by evaluating 100 random realizations of the system at a time. The term 'random realization' refers to different random drawings of the reservoir **A** and the input mapping $\mathbf{W}_{in}$, as well as the initial conditions for the Lorenz system. The first line in Table 1 shows the respective statistical results for the setup shown in Figure 1. The largest Lyapunov exponent of the original chaotic system $\lambda_{orig} = 0.851$ significantly reduces to $\lambda_{changed} = 0.080$ when the parameter change drives the system into a periodic state. After the control mechanism is switched on, the value for the resulting attractor moves back to $\lambda_{controlled} = 0.0841$ and thus is within one standard deviation from its original value. Same applies to the correlation dimension, which resembles its original value after control very well.

Since there is a clear distinction between the chaotic- and the periodic state, with the latter being simple in terms of its dynamics, the next step is to control the system between more complex dynamics. Therefore, we start simulate the Lorenz system again with parameters $\boldsymbol{\pi} = [\sigma = 10.0, \rho = 166.15, \beta = 8/3]$ that lead to intermittent behavior[25]. This is shown in Fig. 2 on the left. Now ρ is changed to $\rho = 167.2$, which results in a chaotic state (middle plots). The control mechanism is turned on and the resulting state shows again the intermittent behavior (right plots) as in the initial state. This is particularly visible in the lower plots where only the X coordinate is shown. While the trajectory mostly follows a periodic path, it is interrupted by irregular burst that occur from time to time. It is remarkable that bursts do not seem to occur more often given the chaotic dynamics of the underlying equations and parameter setup. However, the control works so well that it exactly enforces the desired dynamics. This observation can again be confirmed by looking at the statistical results in Table 1.

Just like in the first two examples, it was not possible before to control a system from one chaotic state to another particular chaotic state. To do this, we start with the parameter set ($\boldsymbol{\pi} = [\sigma = 10.0, \rho = 28.0, \beta = 8/3]$) leading to a chaotic attractor which we call $Chaotic_A$. When changing ρ to $rho = 50.0$ we obtain a different chaotic attractor $Chaotic_B$. This time we use a different range of values for ρ compared to the previous examples in order to present a situation where not only the chaotic dynamics change, but also the size of the attractor significantly

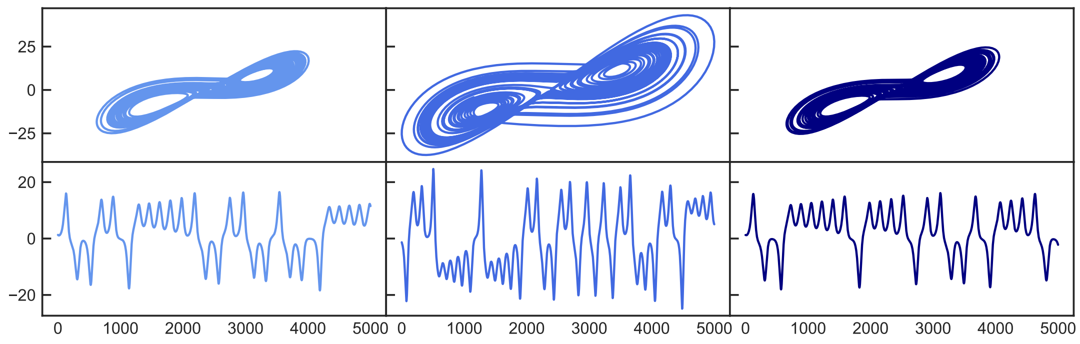

Figure 3. Chaotic to chaotic control. Top: 2D attractor representation in the x–y plane. Bottom: X coordinate time series. Left plots show the original chaotic state which changes to a different chaotic state (middle) after tuning the order parameter. After applying the control mechanism, the system is forced into the initial chaotic state again (right).

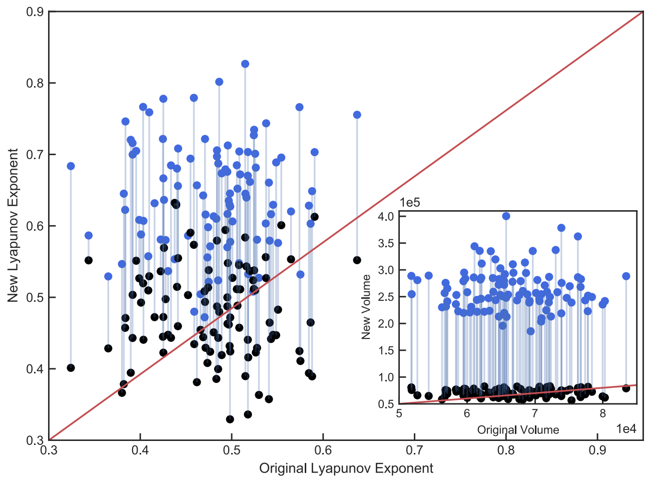

Figure 4. Chaotic to chaotic control (ρ changed). Values on the x-axis denote the largest Lyapunov exponent λ_{max} of the original system state before parameter change for $N = 100$ random realizations. Y-axis reflects the values for λ_{max} after parameters changed from $\rho = 28$ to $\rho = 50$. The blue dots correspond to the uncontrolled systems, while the black dots represent the controlled systems. Inlay plot shows the same for the volume of the attractor.

varies between the two states. The goal of the control procedure now is to not only force the dynamics of the system back to the behavior of the initial state $Chaotic_A$, but also to return the attractor to its original size. Figure 3 shows that both goals succeed. This is also confirmed by the statistical results, indicating that the largest Lyapunov exponent of the controlled system is perfectly close to the one of the uncontrolled original state. For the correlation dimension, however, there are no significant deviations between the two chaotic states. To give a more striking illustration of the statistical analysis, we show the results for each of the 100 random realizations in Fig. 4. The main plot scatters the largest Lyapunov exponents as measured for the original parameter set $\boldsymbol{\pi}$ against those measured after the parameters have been changed to $\boldsymbol{\pi}^*$. While the blue dots represent the situation where the control mechanism is not active, the control has been switched on for the black dots. Furthermore, each pair of points is connected with a line that belongs to the same random realization. It is clearly visible that the control leads to a downwards shift of the cloud of points towards the diagonal, which is consistent to the respective average values of the largest Lyapunov exponent shown in Table 1. In addition, the inlay plot shows the same logic but for the volume of the attractors being measured in terms of the smallest cuboid that covers the attractor. The control mechanism consistently works for every single realization and reduces the volume of the attractor back towards the initially desired state. We successfully applied our approach to other examples of controlling a chaotic state to another chaotic state, e.g. by varying the parameter σ as shown in the supporting information.

The bottom half of Table 1 proves that our statements are also valid if one reverses the direction in the examples. For example, *Periodic* $\rightarrow$ *Chaotic* in the upper half of the table means, that an initially chaotic system changed into a periodic state and then gets controlled back into its initial chaotic state. In contrast, *Periodic* $\leftarrow$ *Chaotic* in the lower half now means that the system initially is in the periodic state. It then shows chaotic behavior after the parameter change and finally is controlled back into the original periodic state—thus the opposite direction as above. It is evident that all examples also succeed in the opposite direction. This supports our claim that the prediction based control mechanism works for arbitrary states.

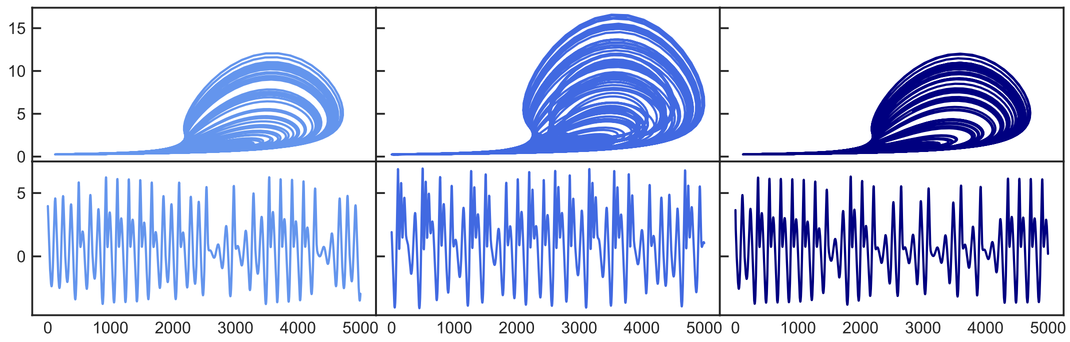

Figure 5. Chaotic to chaotic control for the Roessler system. Top: 2D attractor representation in the x–z plane. Bottom: X coordinate time series. Left plots show the original chaotic state which changes to a different chaotic state (middle) after tuning the order parameter. After applying the control mechanism, the system is forced into the initial chaotic state again (right).

In addition to the Lorenz system we also applied the method to another popular chaotic attractor: the Roessler system[26]. The equations read

$$\dot{x} = -(y + z); \quad \dot{y} = x + ay; \quad \dot{z} = b + (x - c)z \tag{4}$$

and we use parameters $\boldsymbol{\pi} = [a = 0.5, b = 2.0, c = 4.0]$ leading to a chaotic behavior. This serves as our initial state and the dynamics change to another chaotic state after the parameters are changed to $\boldsymbol{\pi}^* = [a = 0.55, b = 2.0, c = 4.0]$. For the Roessler system, we use a time resolution of $\Delta t = 0.05$ and $K = 20$. It can be seen in Fig. 5 that the control mechanism is successful. Again, the left plots represent the initial attractor resulting from the parameter set $\boldsymbol{\pi}$. Switching to $\boldsymbol{\pi}^*$ (middle plots) not only increases the size of the attractor in the x–z plane, but also significantly changes the pattern of the x-coordinate time series. Both, the appearance of the attractor and its x-coordinate pattern become similar to the initial attractor again after the control mechanism is active (right plots). The initial state with parameters $\boldsymbol{\pi}$ has properties $[\lambda_{max} = 0.13, \nu = 1.59]$, which become $[\lambda_{max} = 0.14, \nu = 1.75]$ after parameters have been changed to $\boldsymbol{\pi}^*$. Turning on the control mechanism leads to $[\lambda_{max} = 0.12, \nu = 1.64]$.

Discussion

Our method has a wide range of potential applications in various areas. For example, in nonlinear technical systems such as rocket engines it can be used to prevent the engine from critical combustion instabilities[27,28]. This could be achieved by detecting them based on the reservoir computing predictions (or any other suitable ML technique) and subsequently controlling the system into a more stable state. Here, the control force can be applied to the engine via its pressure valves. Another example would be medical devices such as pacemakers. The heart of a healthy human does not beat in a purely periodic fashion but rather shows features being typical for chaotic systems like multifractality[29] that vary significantly among individuals. While pacing protocols developed so far aim at keeping the diastolic interval constant[30], our general control scheme will emulate the patient-specific full behavior of the heart in healthy conditions. The control scheme could therefore be used to develop personalized pacemakers that do not just stabilize the heartbeat to periodic behavior[31–33], but may rather adjust the heartbeat to the individual needs of the patients.

In conclusion, our machine learning enhanced method allows for an unprecedented flexible control of dynamical systems and has thus the potential to extend the range of applications of chaos inspired control schemes to a plethora of new real-world problems.

Methods

Reservoir computing.

RC or echo state networks[17,34,35] is an artificial recurrent neural network based approach, which builds on a static internal network called *reservoir* **A**. Static means that the nodes and edges are kept fixed once the network has been initially created. This property makes RC computationally very efficient, as only its linear output layer is being optimized in the training process. The reservoir **A** is constructed as a sparse Erdös–Renyi random network[36] with $D_r = 300$ nodes that are connected with a probability $p = 0.02$. In order to feed the $D = 3$ dimensional input data $\mathbf{u}(t)$ into the reservoir **A**, we set up an $D_r \times D$ input mapping matrix $\mathbf{W}_{in}$, which defines how strongly each input dimension influences every single node. The dynamics of the network are represented by its $D_r \times 1$ dimensional scalar states $\mathbf{r}(t)$ evolving according to the recurrent equation

$$\mathbf{r}(t + \Delta t) = tanh(\mathbf{A}\mathbf{r}(t) + \mathbf{W}_{in}\mathbf{u}(t)). \tag{5}$$

Output $\mathbf{v}(t + \Delta t)$ is created by mapping back $\mathbf{r}(t)$ using a linear output function $\mathbf{W}_{out}$ such that

$$\mathbf{v}(t) = \mathbf{W}_{out}(\tilde{\mathbf{r}}(t), \mathbf{P}) = \mathbf{P}\tilde{\mathbf{r}}(t), \tag{6}$$

where $\tilde{\mathbf{r}} = \{\mathbf{r}, \mathbf{r}^2\}$. The matrix **P** is determined in the training process. This is done by acquiring a sufficient number of reservoir states $\mathbf{r}(t_w \dots t_w + t_T)$ and then choosing **P** such that the output **v** of the reservoir is as close as possible to the known real data $\mathbf{v}(t_w \dots t_w + t_T)$. For this we use Ridge regression, which minimizes

$$\sum_{-T \le t \le 0} \| \mathbf{W}_{out}(\tilde{\mathbf{r}}(t), \mathbf{P}) - \mathbf{v}_R(t) \|^2 - \beta \| \mathbf{P} \|^2, \tag{7}$$

where β is the regularization constant that prevents from overfitting by penalizing large values of the fitting parameters. The training process only involves the linear output layer and therefore is fast compared to other ML methods. Replacing $\mathbf{u}(t)$ in the *tanh* activation function above by $\mathbf{P}\tilde{\mathbf{r}}(t)$ allows to create predictions of arbitrary length due to the recursive equation for the reservoir states $\mathbf{r}(t)$:

$$\begin{aligned} \mathbf{r}(t + \Delta t) &= tanh(\mathbf{A}\mathbf{r}(t) + \mathbf{W}_{in}\mathbf{W}_{out}(\tilde{\mathbf{r}}(t), \mathbf{P})) \\ &= tanh(\mathbf{A}\mathbf{r}(t) + \mathbf{W}_{in}\mathbf{P}\tilde{\mathbf{r}}(t)). \end{aligned} \tag{8}$$

Further details including the choices for the hyperparameters are presented in the supporting information. We use a washout phase of 1000 time steps, a training period of 5000 time steps and let the parameter change of the dynamical system from $\boldsymbol{\pi}$ to $\boldsymbol{\pi}^*$ happen immediately after the training period and thus the prediction is needed from this moment on. However, it is not necessary that the network is trained on the full history until the parameter change happened. In general, it needs to be sufficiently trained and can then be synchronized based on the recorded trajectory after the training ended. The prediction is carried out for 10,000 time steps.

It has been shown by Bompas et al.[18] that the performance of reservoir computing does not strongly depend on the precision of the data. Hence, measurement noise and sensitive dependence on initial conditions for chaotic systems is not a problem when it comes to real world applications of the proposed method.

Control mechanism. The concrete steps of the application of the control mechanism to the examples in our study are shown in Algorithm 1. This is the simplest setup possible, where only one long prediction for **v**(t) is performed before the control force is activated. We also successfully tested multiple more complicated setups, e.g. where the control force is not immediately switched on and the system is running on the new parameters $\boldsymbol{\pi}^*$ (and thus state **Y**) for a while, where the reservoir computing prediction is updated after synchronizing the RC model with the realized trajectory since the last training or where the force is not applied in every time step. The control phase is run for 10,000 time steps.

These steps also apply for real world systems, where no mathematical model is available. The only requirement is sufficient data of the system recorded while being in the desired dynamical state **X**.

Algorithm 1: Control mechanism

```
1 Simulate the system with initial parameters $\boldsymbol{\pi}$ such that the system is in a certain dynamical state $\mathbf{X}$
2 Train reservoir computing onto this data
3 Change the parameters to $\boldsymbol{\pi}^*$ such that the system is now in dynamical state $\mathbf{Y}$ with coordinates $\mathbf{u}(t_0)$
4 Predict how $\mathbf{u}(t_0)$ would evolve if the system was still in state $\mathbf{X}$ using RC from Step 2 - this yields $\mathbf{v}$(t)
5 For $t > t_0$
6 Measure $\mathbf{u}$(t) and determine $\mathbf{v}$(t)
7 Compute the control force $\mathbf{F(t)}$,
      $\mathbf{F(t)} = K(\mathbf{u}(t) - \mathbf{v}(t))$
8 Apply the control force when simulating
      $\mathbf{u}(t+\Delta t) = \int_t^{t+\Delta t} (\dot{f}(\mathbf{u}(\tilde{t}), \boldsymbol{\pi}^*) + \mathbf{F}(\tilde{t}))d\tilde{t}$ ,
9 Repeat Steps 6 to 8 in every time step
```

Correlation dimension. To characterize the attractor and therefore its dynamical state we rely on quantitative measures. For this, we are looking at the long-term properties of the attractor rather than its short-term trajectory. One important aspect of the long-term behavior is the structural complexity. This can be assessed by calculating the correlation dimension of the attractor, where we measure the dimensionality of the space populated by the trajectory[37]. The correlation dimension is based on the correlation integral

$$
\begin{aligned}
C(r) &= \lim_{N\to\infty} \frac{1}{N^2} \sum_{i,j=1}^{N} \theta(r - |\mathbf{x}_i - \mathbf{x}_j|) \\
&= \int_0^r d^3 r' c(\mathbf{r}'),
\end{aligned} \tag{9}
$$

where θ is the Heaviside function and $c(\mathbf{r}')$ denotes the standard correlation function. The correlation integral represents the mean probability that two states in phase space are close to each other at different time steps. This is the case if the distance between the two states is less than the threshold distance r. The correlation dimension ν is then defined by the power-law relationship

$$C(r) \propto r^{\nu}. \tag{10}$$

For self-similar strange attractors, this relationship holds for a certain range of r, which therefore needs to be properly calibrated. As we are finally only interested in comparisons, precision with regards to absolute values is not essential here. We use the Grassberger Procaccia algorithm[38] to calculate the correlation dimension.

Lypunov exponents. The temporal complexity of a system can be measured by its Lyapunov exponents λ_i, which describe the average rate of divergence of nearby points in phase space, and thus measure sensitivity to initial conditions. There is one exponent for each dimension in phase space. If the system exhibits at least one positive Lyapunov exponent, it is classified as chaotic. The magnitudes of λ_i quantify the time scale on which the system becomes unpredictable[39,40]. Since at least one positive exponent is the requirement for being classified as chaotic, it is sufficient for our analysis to calculate only the largest Lyapunov exponent λ_{max}

$$d(t) = Ce^{\lambda_{max} t}. \tag{11}$$

This makes the task computationally much easier than determining the full Lyapunov spectrum. We use the Rosenstein algorithm[41] to obtain it. In essence, we track the distance $d(t)$ of two initially nearby states in phase space. The constant C normalizes the initial separation. As for the correlation dimension, we are interested in a relative comparison that characterizes states of the system rather than the exact absolute values. It is important to point out that both measures—the correlation dimension and the largest Lyapunov exponent—are calculated purely based on data and do not require any knowledge of the underlying equations.

Data availability

The data that support the findings of this study are available from the corresponding author upon reasonable request.

Received: 4 April 2021; Accepted: 7 June 2021
Published online: 21 June 2021

References

1. Ott, E., Grebogi, C. & Yorke, J. A. Controlling chaos. *Phys. Rev. Lett.* **64**, 1196 (1990).
2. Shinbrot, T., Grebogi, C., Yorke, J. A. & Ott, E. Using small perturbations to control chaos. *Nature* **363**, 411–417 (1993).
3. Pyragas, K. Continuous control of chaos by self-controlling feedback. *Phys. Lett. A* **170**, 421–428 (1992).
4. Boccaletti, S., Grebogi, C., Lai, Y.-C., Mancini, H. & Maza, D. The control of chaos: Theory and applications. *Phys. Rep.* **329**, 103–197 (2000).
5. Schiff, S. J. *et al.* Controlling chaos in the brain. *Nature* **370**, 615–620 (1994).
6. Chattopadhyay, A., Hassanzadeh, P. & Subramanian, D. Data-driven predictions of a multiscale lorenz 96 chaotic system using machine-learning methods: Reservoir computing, artificial neural network, and long short-term memory network. *Nonlinear Process. Geophys.* **27**, 373–389 (2020).
7. Vlachas, P. R. *et al.* Backpropagation algorithms and reservoir computing in recurrent neural networks for the forecasting of complex spatiotemporal dynamics. *Neural Netw.* **126**, 191–217 (2020).
8. Sangiorgio, M. & Dercole, F. Robustness of LSTM neural networks for multi-step forecasting of chaotic time series. *Chaos Solitons Fractals* **139**, 110045 (2020).
9. Herteux, J. & Räth, C. Breaking symmetries of the reservoir equations in echo state networks. *Chaos Interdiscip. J. Nonlinear Sci.* **30**, 123142 (2020).
10. Haluszczynski, A. & Räth, C. Good and bad predictions: Assessing and improving the replication of chaotic attractors by means of reservoir computing. *Chaos Interdiscip. J. Nonlinear Sci.* **29**, 103143 (2019).
11. Griffith, A., Pomerance, A. & Gauthier, D. J. Forecasting chaotic systems with very low connectivity reservoir computers. *Chaos Interdiscip. J. Nonlinear Sci.* **29**, 123108 (2019).
12. Lu, Z., Hunt, B. R. & Ott, E. Attractor reconstruction by machine learning. *Chaos Interdiscip. J. Nonlinear Sci.* **28**, 061104 (2018).
13. Pathak, J., Hunt, B., Girvan, M., Lu, Z. & Ott, E. Model-free prediction of large spatiotemporally chaotic systems from data: A reservoir computing approach. *Phys. Rev. Lett.* **120**, 024102 (2018).
14. Zimmermann, R. S. & Parlitz, U. Observing spatio-temporal dynamics of excitable media using reservoir computing. *Chaos Interdiscip. J. Nonlinear Sci.* **28**, 043118 (2018).
15. Baur, S. & Räth, C. Predicting high-dimensional heterogeneous time series employing generalized local states (2021). arXiv:2102.12333.
16. Maass, W., Natschläger, T. & Markram, H. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Comput.* **14**, 2531–2560 (2002).
17. Jaeger, H. & Haas, H. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *Science* **304**, 78–80 (2004).
18. Bompas, S., Georgeot, B. & Guéry-Odelin, D. Accuracy of neural networks for the simulation of chaotic dynamics: Precision of training data vs precision of the algorithm. *Chaos Interdiscip. J. Nonlinear Sci.* **30**, 113118 (2020).

19. Marcucci, G., Pierangeli, D. & Conti, C. Theory of neuromorphic computing by waves: Machine learning by rogue waves, dispersive shocks, and solitons. *Phys. Rev. Lett.* **125**, 093901 (2020).
20. Tanaka, G. *et al.* Recent advances in physical reservoir computing: A review. *Neural Netw.* **115**, 100–123 (2019).
21. Carroll, T. L. Adding filters to improve reservoir computer performance. *Physica D* **416**, 132798 (2021).
22. Pathak, J., Lu, Z., Hunt, B. R., Girvan, M. & Ott, E. Using machine learning to replicate chaotic attractors and calculate Lyapunov exponents from data. *Chaos Interdiscip. J. Nonlinear Sci.* **27**, 121102 (2017).
23. Lukoševičius, M. & Jaeger, H. Reservoir computing approaches to recurrent neural network training. *Comput. Sci. Rev.* **3**, 127–149 (2009).
24. Lorenz, E. N. Deterministic nonperiodic flow. *J. Atmos. Sci.* **20**, 130–141 (1963).
25. Pomeau, Y. & Manneville, P. Intermittent transition to turbulence in dissipative dynamical systems. *Commun. Math. Phys.* **74**, 189–197 (1980).
26. Rössler, O. E. An equation for continuous chaos. *Phys. Lett. A* **57**, 397–398 (1976).
27. Kabiraj, L., Saurabh, A., Wahi, P. & Sujith, R. Route to chaos for combustion instability in ducted laminar premixed flames. *Chaos Interdiscip. J. Nonlinear Sci.* **22**, 023129 (2012).
28. Nair, V., Thampi, G. & Sujith, R. Intermittency route to thermoacoustic instability in turbulent combustors. *J. Fluid Mech.* **756**, 470 (2014).
29. Ivanov, P. C. *et al.* Multifractality in human heartbeat dynamics. *Nature* **399**, 461–465 (1999).
30. Kulkarni, K., Walton, R. D., Armoundas, A. A. & Tolkacheva, E. G. Clinical potential of beat-to-beat diastolic interval control in preventing cardiac arrhythmias. *J. Am. Heart Assoc.* e020750 (2021).
31. Garfinkel, A., Spano, M. L., Ditto, W. L. & Weiss, J. N. Controlling cardiac chaos. *Science* **257**, 1230–1235 (1992).
32. Hall, K. *et al.* Dynamic control of cardiac alternans. *Phys. Rev. Lett.* **78**, 4518 (1997).
33. Christini, D. J. *et al.* Nonlinear-dynamical arrhythmia control in humans. *Proc. Natl. Acad. Sci.* **98**, 5827–5832 (2001).
34. Jaeger, H. The, "echo state" approach to analysing and training recurrent neural networks-with an erratum note. *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report* **148**, 13 (2001).
35. Maass, W., Natschlaeger, T. & Markram, H. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Comput.* **14**, 2531–2560. https://doi.org/10.1162/089976602760407955 (2002).
36. Erdos, P. On random graphs. *Publicationes mathematicae* **6**, 290–297 (1959).
37. Grassberger, P. & Procaccia, I. Measuring the strangeness of strange attractors. In *The Theory of Chaotic Attractors*, 170–189 (Springer, 2004).
38. Grassberger, P. Generalized dimensions of strange attractors. *Phys. Lett. A* **97**, 227–230 (1983).
39. Wolf, A., Swift, J. B., Swinney, H. L. & Vastano, J. A. Determining Lyapunov exponents from a time series. *Physica D* **16**, 285–317 (1985).
40. Shaw, R. Strange attractors, chaotic behavior, and information flow. *Zeitschrift für Naturforschung A* **36**, 80–112 (1981).
41. Rosenstein, M. T., Collins, J. J. & De Luca, C. J. A practical method for calculating largest Lyapunov exponents from small data sets. *Physica D* **65**, 117–134 (1993).

Acknowledgements
We would like to thank Y. Mabrouk, J. Herteux, S. Baur and J. Aumeier for helpful discussions.

Author contributions
C.R. initiated the research. A.H. and C.R. designed the study. A.H. performed the calculations. A.H. and C.R. interpreted the results and wrote the manuscript.

Funding
Open Access funding enabled and organized by Projekt DEAL.

Competing interests
The authors declare no competing interests.

Additional information
Supplementary Information The online version contains supplementary material available at https://doi.org/10.1038/s41598-021-92244-6.

Correspondence and requests for materials should be addressed to A.H.

Acknowledgement

First and foremost I would like to express my sincere gratitude to my supervisor Christoph Räth for his invaluable advice, great flexibility and continuous support. Without you and your impressive intuition for the right topics this journey of discovery would not have been possible. I would also like to express my sincere gratitude to Sven Treu and *risklab* for giving me the flexibility and opportunity to do such a large undertaking alongside my *normal* work. I wish to thank all the people whose assistance was crucial in the completion of this PhD work, in particular Jonas Aumeier, Joschka Herteux, Youssef Mabrouk and Haochun Ma. Last but not least, I would like to offer my special thanks to my girlfriend Isabel Linner for her support and for putting up with me spending a lot of time doing simulations or writing papers during our beautiful vacations.

Bibliography

[1]K. Fujii and K. Nakajima, “Quantum reservoir computing: a reservoir approach toward quantum machine learning on near-term quantum devices”, in *Reservoir computing* (Springer, 2021), pp. 423–450.

[2]W. S. McCulloch and W. Pitts, “A logical calculus of the ideas immanent in nervous activity”, The bulletin of mathematical biophysics **5**, 115–133 (1943).

[3]D. O. Hebb, *The organisation of behaviour: a neuropsychological theory* (Science Editions New York, 1949).

[4]B. Farley and W. Clark, “Simulation of self-organizing systems by digital computer”, Transactions of the IRE Professional Group on Information Theory **4**, 76–84 (1954).

[5]F. Rosenblatt, “The perceptron: a probabilistic model for information storage and organization in the brain.”, Psychological review **65**, 386 (1958).

[6]J. Schmidhuber, “Deep learning in neural networks: an overview”, Neural networks **61**, 85–117 (2015).

[7]D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, et al., “Mastering the game of go without human knowledge”, nature **550**, 354–359 (2017).

[8]E. N. Lorenz, “Deterministic nonperiodic flow”, Journal of atmospheric sciences **20**, 130–141 (1963).

[9]C. Danforth, “3.2 lorenz’s discovery of chaos”, Mathematics of Planet Earth: Mathematicians Reflect on How to Discover, Organize, and Protect Our Planet **140**, 39 (2015).

[10]E. Ott, C. Grebogi, and J. A. Yorke, “Controlling chaos”, Physical review letters **64**, 1196 (1990).

[11]K. Pyragas, “Continuous control of chaos by self-controlling feedback”, Physics letters A **170**, 421–428 (1992).

[12]S. H. Strogatz, *Nonlinear dynamics and chaos with student solutions manual: with applications to physics, biology, chemistry, and engineering* (CRC press, 2018).

[13]F. J. Romeiras, C. Grebogi, E. Ott, and W. Dayawansa, "Controlling chaotic dynamical systems", Physica D: Nonlinear Phenomena **58**, 165–192 (1992).

[14]W. Ditto, S. Rauseo, R. Cawley, C. Grebogi, G.-H. Hsu, E. Kostelich, E. Ott, H. Savage, R. Segnan, M. Spano, et al., "Experimental observation of crisis-induced intermittency and its critical exponent", Physical review letters **63**, 923 (1989).

[15]D. P. Lathrop and E. J. Kostelich, "Characterization of an experimental strange attractor by periodic orbits", Physical Review A **40**, 4028 (1989).

[16]H. Jaeger, "The "echo state" approach to analysing and training recurrent neural networks-with an erratum note", Bonn, Germany: German National Research Center for Information Technology GMD Technical Report **148**, 13 (2001).

[17]W. Maass, T. Natschläger, and H. Markram, "Real-time computing without stable states: a new framework for neural computation based on perturbations", Neural computation **14**, 2531–2560 (2002).

[18]H. Jaeger and H. Haas, "Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless communication", science **304**, 78–80 (2004).

[19]P. Erdos, "On random graphs", Publicationes mathematicae **6**, 290–297 (1959).

[20]M. Lukoševičius and H. Jaeger, "Reservoir computing approaches to recurrent neural network training", Computer Science Review **3**, 127–149 (2009).

[21]J. Herteux and C. Räth, "Breaking symmetries of the reservoir equations in echo state networks", Chaos: An Interdisciplinary Journal of Nonlinear Science **30**, 123142 (2020).

[22]A. E. Hoerl and R. W. Kennard, "Ridge regression: biased estimation for nonorthogonal problems", Technometrics **12**, 55–67 (1970).

[23]P. Grassberger and I. Procaccia, "Measuring the strangeness of strange attractors", in *The theory of chaotic attractors* (Springer, 2004), pp. 170–189.

[24]B. Mandelbrot, "How long is the coast of britain? statistical self-similarity and fractional dimension", science **156**, 636–638 (1967).

[25]P. Grassberger, "Generalized dimensions of strange attractors", Physics Letters A **97**, 227–230 (1983).

[26]A. Wolf, J. B. Swift, H. L. Swinney, and J. A. Vastano, "Determining lyapunov exponents from a time series", Physica D: Nonlinear Phenomena **16**, 285–317 (1985).

[27]R. Shaw, "Strange attractors, chaotic behavior, and information flow", Zeitschrift für Naturforschung A **36**, 80–112 (1981).

[28]M. T. Rosenstein, J. J. Collins, and C. J. De Luca, "A practical method for calculating largest lyapunov exponents from small data sets", Physica D: Nonlinear Phenomena **65**, 117–134 (1993).

[29]A. Haluszczynski and C. Räth, “Good and bad predictions: assessing and improving the replication of chaotic attractors by means of reservoir computing”, Chaos: An Interdisciplinary Journal of Nonlinear Science **29**, 103143 (2019).

[30]Z. Lu, J. Pathak, B. Hunt, M. Girvan, R. Brockett, and E. Ott, “Reservoir observers: Model-free inference of unmeasured variables in chaotic systems”, Chaos **27**, 041102, 041102 (2017).

[31]J. Pathak, Z. Lu, B. R. Hunt, M. Girvan, and E. Ott, “Using machine learning to replicate chaotic attractors and calculate lyapunov exponents from data”, Chaos: An Interdisciplinary Journal of Nonlinear Science **27**, 121102 (2017).

[32]J. Pathak, B. Hunt, M. Girvan, Z. Lu, and E. Ott, “Model-free prediction of large spatiotemporally chaotic systems from data: a reservoir computing approach”, Physical review letters **120**, 024102 (2018).

[33]J. Pathak, A. Wikner, R. Fussell, S. Chandra, B. R. Hunt, M. Girvan, and E. Ott, “Hybrid forecasting of chaotic processes: Using machine learning in conjunction with a knowledge-based model”, Chaos **28**, 041101, 041101 (2018).

[34]R. S. Zimmermann and U. Parlitz, “Observing spatio-temporal dynamics of excitable media using reservoir computing”, Chaos: An Interdisciplinary Journal of Nonlinear Science **28**, 043118 (2018).

[35]T. L. Carroll, “Using reservoir computers to distinguish chaotic signals”, Physical Review E **98**, 052209, 052209 (2018).

[36]Z. Lu, B. R. Hunt, and E. Ott, “Attractor reconstruction by machine learning”, Chaos: An Interdisciplinary Journal of Nonlinear Science **28**, 061104 (2018).

[37]P. Antonik, M. Gulina, J. Pauwels, and S. Massar, “Using a reservoir computer to learn chaotic attractors, with applications to chaos synchronization and cryptography”, Physical Review E **98**, 012215, 012215 (2018).

[38]A.-L. Barabási and E. Bonabeau, “Scale-free networks”, Scientific american **288**, 60–69 (2003).

[39]D. J. Watts and S. H. Strogatz, “Collective dynamics of ‘small-world’networks”, nature **393**, 440 (1998).

[40]Z. Lu, J. Pathak, B. Hunt, M. Girvan, R. Brockett, and E. Ott, “Reservoir observers: model-free inference of unmeasured variables in chaotic systems”, Chaos: An Interdisciplinary Journal of Nonlinear Science **27**, 041102 (2017).

[41]O. E. Rössler, “An equation for continuous chaos”, Physics Letters A **57**, 397–398 (1976).

[42]C. Runge, “Über die numerische auflösung von differentialgleichungen”, Mathematische Annalen **46**, 167–178 (1895).

[43]L. Amaral, A. Scala, M. Barthélémy, and H. Stanley, “Classes of small-world networks”, PNAS; Proceedings of the National Academy of Sciences **97**, 11149–11152 (2000).

[44]A. Hagberg, P. Swart, and D. S Chult, *Exploring network structure, dynamics, and function using networkx*, tech. rep. (Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008).

[45]A. Haluszczynski, J. Aumeier, J. Herteux, and C. Räth, “Reducing network size and improving prediction stability of reservoir computing”, Chaos: An Interdisciplinary Journal of Nonlinear Science **30**, 063136 (2020).

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

[47]A. S. Elwakil, S. Ozoguz, and M. P. Kennedy, “Creation of a complex butterfly attractor using a novel lorenz-type system”, IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications **49**, 527–530 (2002).

[48]G. Chen and T. Ueta, “Yet another chaotic attractor”, International Journal of Bifurcation and chaos **9**, 1465–1466 (1999).

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

[50]T. Matsumoto, L. Chua, and M. Komuro, “The double scroll”, IEEE Transactions on Circuits and Systems **32**, 797–818 (1985).

[51]R. Thomas, “Deterministic chaos seen in terms of feedback circuits: analysis, synthesis,"labyrinth chaos
”, International Journal of Bifurcation and Chaos **9**, 1889–1905 (1999).

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

[53]J. C. Sprott and J. C. Sprott, *Chaos and time-series analysis*, Vol. 69 (Citeseer, 2003).

[54]R. Albert, H. Jeong, and A.-L. Barabási, “Error and attack tolerance of complex networks”, Nature **406**, 378–382 (2000).

[55]M. Inubushi and K. Yoshimura, “Reservoir computing beyond memory-nonlinearity trade-off”, Scientific reports **7**, 1–10 (2017).

[56]A. Goudarzi, A. Shabani, and D. Stefanovic, “Exploring transfer function nonlinearity in echo state networks”, in 2015 ieee symposium on computational intelligence for security and defense applications (cisda) (IEEE, 2015), pp. 1–8.

[57]D. Verstraeten, B. Schrauwen, M. d’Haene, and D. Stroobandt, “An experimental unification of reservoir computing methods”, Neural networks **20**, 391–403 (2007).

[58]P. Verzelli, C. Alippi, and L. Livi, “Echo state networks with self-normalizing activations on the hyper-sphere”, Scientific reports **9**, 1–14 (2019).

[59]G. Marcucci, D. Pierangeli, and C. Conti, “Theory of neuromorphic computing by waves: machine learning by rogue waves, dispersive shocks, and solitons”, Physical Review Letters **125**, 093901 (2020).

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

[61]T. L. Carroll, “Adding filters to improve reservoir computer performance”, Physica D: Nonlinear Phenomena **416**, 132798 (2021).

[62]A. Haluszczynski and C. Räth, “Controlling nonlinear dynamical systems into arbitrary states using machine learning”, Scientific Reports **11**, 1–8 (2021).

[63]T. Shinbrot, C. Grebogi, J. A. Yorke, and E. Ott, “Using small perturbations to control chaos”, nature **363**, 411–417 (1993).

[64]S. J. Schiff, K. Jerger, D. H. Duong, T. Chang, M. L. Spano, and W. L. Ditto, “Controlling chaos in the brain”, Nature **370**, 615–620 (1994).

[65]S. Boccaletti, C. Grebogi, Y.-C. Lai, H. Mancini, and D. Maza, “The control of chaos: theory and applications”, Physics reports **329**, 103–197 (2000).

[66]L. Kabiraj, A. Saurabh, P. Wahi, and R. Sujith, “Route to chaos for combustion instability in ducted laminar premixed flames”, Chaos: An Interdisciplinary Journal of Nonlinear Science **22**, 023129 (2012).

[67]V. Nair, G. Thampi, and R. Sujith, “Intermittency route to thermoacoustic instability in turbulent combustors”, Journal of Fluid Mechanics **756**, 470 (2014).

[68]P. C. Ivanov, L. A. N. Amaral, A. L. Goldberger, S. Havlin, M. G. Rosenblum, Z. R. Struzik, and H. E. Stanley, “Multifractality in human heartbeat dynamics”, Nature **399**, 461–465 (1999).

[69]K. Kulkarni, R. D. Walton, A. A. Armoundas, and E. G. Tolkacheva, “Clinical potential of beat-to-beat diastolic interval control in preventing cardiac arrhythmias”, Journal of the American Heart Association, e020750 (2021).

[70]A. Garfinkel, M. L. Spano, W. L. Ditto, and J. N. Weiss, “Controlling cardiac chaos”, Science **257**, 1230–1235 (1992).

[71]K. Hall, D. J. Christini, M. Tremblay, J. J. Collins, L. Glass, and J. Billette, “Dynamic control of cardiac alternans”, Physical Review Letters **78**, 4518 (1997).

[72]D. J. Christini, K. M. Stein, S. M. Markowitz, S. Mittal, D. J. Slotwiner, M. A. Scheiner, S. Iwai, and B. B. Lerman, “Nonlinear-dynamical arrhythmia control in humans”, Proceedings of the National Academy of Sciences **98**, 5827–5832 (2001).

[73]A. Wikner, J. Pathak, B. Hunt, M. Girvan, T. Arcomano, I. Szunyogh, A. Pomerance, and E. Ott, "Combining machine learning with knowledge-based modeling for scalable forecasting and subgrid-scale closure of large, complex, spatiotemporal systems", Chaos: An Interdisciplinary Journal of Nonlinear Science **30**, 053111 (2020).

[74]R. Maulik, B. Lusch, and P. Balaprakash, "Reduced-order modeling of advection-dominated systems with recurrent neural networks and convolutional autoencoders", Physics of Fluids **33**, 037106 (2021).

[75]A. Wikner, J. Pathak, B. R. Hunt, I. Szunyogh, M. Girvan, and E. Ott, "Using data assimilation to train a hybrid forecast system that combines machine-learning and knowledge-based components<? a3b2 show [editpick]?>", Chaos: An Interdisciplinary Journal of Nonlinear Science **31**, 053114 (2021).

[76]N. A. K. Doan, W. Polifke, and L. Magri, "Short-and long-term prediction of a chaotic flow: a physics-constrained reservoir computing approach", arXiv preprint arXiv:2102.07514 (2021).

[77]J. Theiler, S. Eubank, A. Longtin, B. Galdrikian, and J. D. Farmer, "Testing for nonlinearity in time series: the method of surrogate data", Physica D: Nonlinear Phenomena **58**, 77–94 (1992).

[78]D. Prichard and J. Theiler, "Generating surrogate data for time series with several simultaneously measured variables", Physical review letters **73**, 951 (1994).

[79]C. Räth, M. Gliozzi, I. Papadakis, and W. Brinkmann, "Revisiting algorithms for generating surrogate time series", Physical review letters **109**, 144101 (2012).

[80]A. Haluszczynski, I. Laut, H. Modest, and C. Räth, "Linear and nonlinear market correlations: characterizing financial crises and portfolio optimization", Physical Review E **96**, 062315 (2017).

[81]C. W. Granger, *Essays in econometrics: collected papers of clive wj granger*, Vol. 32 (Cambridge University Press, 2001).

[82]G. Sugihara, R. May, H. Ye, C.-h. Hsieh, E. Deyle, M. Fogarty, and S. Munch, "Detecting causality in complex ecosystems", science **338**, 496–500 (2012).

[83]T. Schreiber, "Measuring information transfer", Physical review letters **85**, 461 (2000).

[84]J. Runge, "Causal network reconstruction from time series: from theoretical assumptions to practical estimation", Chaos: An Interdisciplinary Journal of Nonlinear Science **28**, 075310 (2018).

[85]C. Räth and R. Monetti, "Surrogates with random fourier phases", in Topics on chaotic systems: selected papers from chaos 2008 international conference (World Scientific, 2009), pp. 274–285.

[86]S. L. Bressler and A. K. Seth, “Wiener–granger causality: a well established methodology”, Neuroimage **58**, 323–329 (2011).

[87]J. M. McCracken and R. S. Weigel, “Convergent cross-mapping and pairwise asymmetric inference”, Physical Review E **90**, 062903 (2014).

[88]F. Takens, “Detecting strange attractors in turbulence”, in *Dynamical systems and turbulence, warwick 1980* (Springer, 1981), pp. 366–381.

[89]L. Barnett, A. B. Barrett, and A. K. Seth, “Granger causality and transfer entropy are equivalent for gaussian variables”, Physical review letters **103**, 238701 (2009).

[90]S. Vaidyanathan and A. T. Azar, “Adaptive control and synchronization of halvorsen circulant chaotic systems”, in *Advances in chaos theory and intelligent control* (Springer, 2016), pp. 225–247.

[91]H. Ma, A. Haluszczynski, and C. Räth, “Identifying causality drivers and deriving governing equations of nonlinear complex systems”, to be submitted.