

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 R ath



View Online



Export Citation



CrossMark



 **NEW: TOPIC ALERTS**
Explore the latest discoveries in your field of research

SIGN UP TODAY!



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 R ath^{3,d)}

AFFILIATIONS

¹Department of Physics, Ludwig-Maximilians-Universit t, Schellingstra e 4, 80799 Munich, Germany

²risklab GmbH, Seidlstra e 24, 80335 Munich, Germany

³Institut f r Materialphysik im Weltraum, Deutsches Zentrum f r Luft- und Raumfahrt, M nchner 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.

Published under license by AIP Publishing. <https://doi.org/10.1063/5.0006869>

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 events¹²—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 study¹³ 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 study¹³ and works in the following way. First, we set up the reservoir **A**, which has dimensionality D_r , and is constructed as a sparse Erdős–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 **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

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

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 **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}

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

Here, we assume that \mathbf{W}_{out} depends linearly on a matrix **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 **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 **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 \mathbf{P} using Ridge regression, which minimizes

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

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 \mathbf{P} . For solving this problem, we are using the matrix form of the Ridge regression,²¹ which leads to

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

The notion \mathbf{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 \mathbf{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 $\mathbf{r}(t)$,

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

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

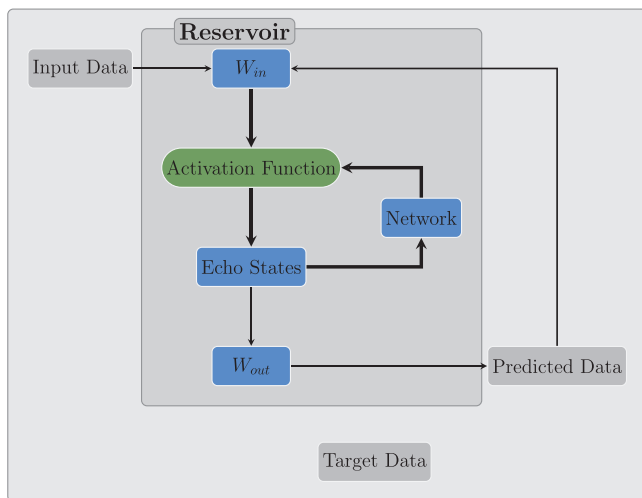


FIG. 1. Schematic illustration of reservoir computing.

TABLE I. Results of the hyperparameter optimization.

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

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 \mathbf{A} and the input function \mathbf{W}_{in} . Each of the realizations is starting from randomly chosen coordinates obtained from simulating a very long trajectory of the Lorenz system. In order to assure independent trajectories, small scale uniform noise is added. The optimal values are shown in Table I.

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 R ath.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 \mathbf{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 non-linear scaling factor a in the hyperbolic tangent of the activation function to further improve prediction quality. This changes the

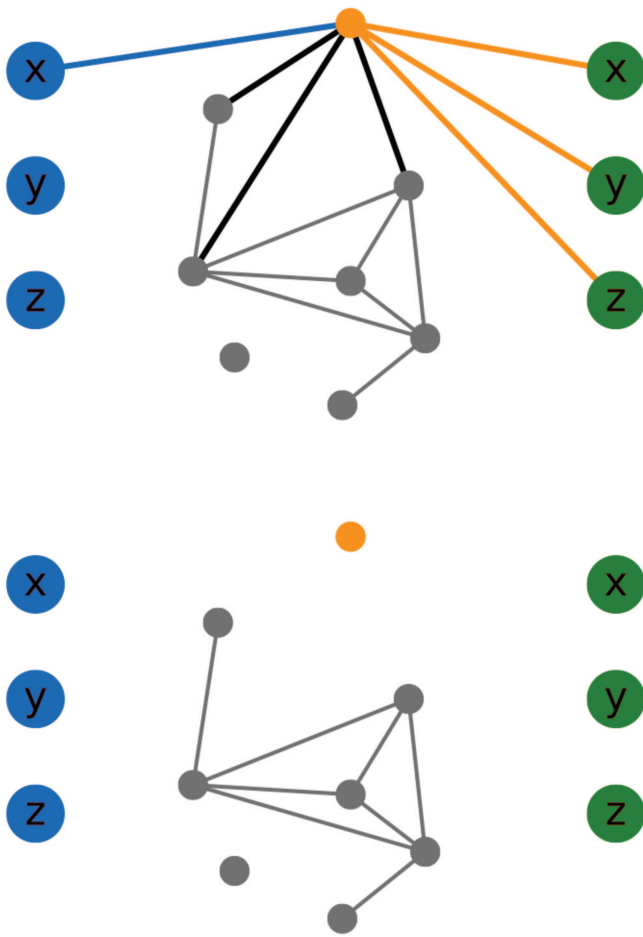


FIG. 2. Schematic illustration of the controlled node removal. The top graphic shows the initial network before the removal procedure. Here, the orange example node is fed only with input (blue) from the x dimension of the system. The orange lines denote its contribution to the output values of all three 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

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

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

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

where the thresholds are $\delta = (5.8, 8.0, 6.9)^T$ 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

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

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 :

$$C(r) \propto r^\nu. \quad (9)$$

The *correlation dimension* is then measured by the scaling exponent ν . We use the Grassberger Procaccia algorithm²⁹ 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 $s_r = \sigma(\mathbf{u})/8.5$. 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 $x \rightarrow -x$ and $y \rightarrow -y$. 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

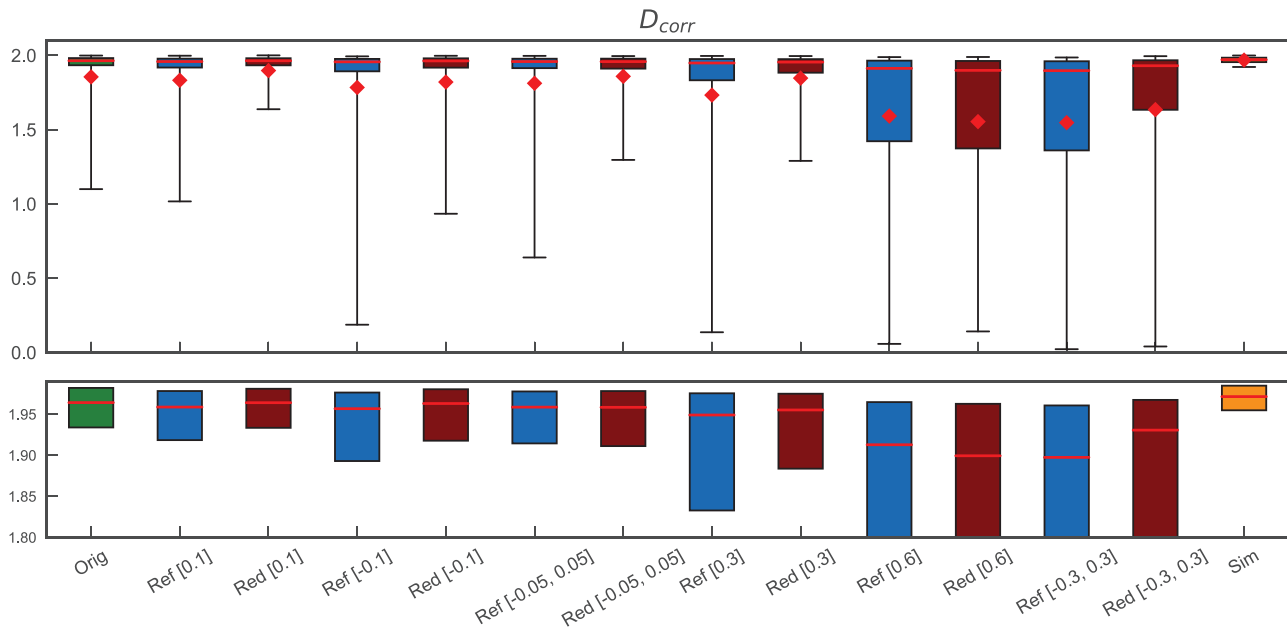


FIG. 3. Boxplot of the correlation dimension for $N = 500$ realizations for the original setup (green—left) and different percentages of nodes removed. Positive numbers (e.g., Red [0.1]) represent a removal of the 10% largest \mathbf{W}_{out} nodes, while negative numbers (e.g., Red [−0.1]) denote a 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

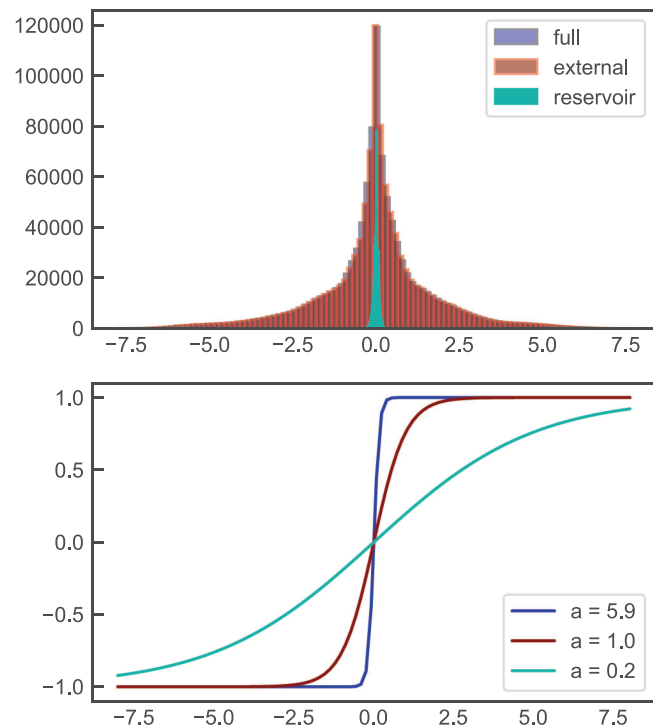


FIG. 4. Top plot: Distribution of the arguments of the activation function during training period split into the contribution from the reservoir (green), the input term (red), and total (blue). Bottom plot: Hyperbolic tangent for different nonlinear scaling factors.

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.

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.

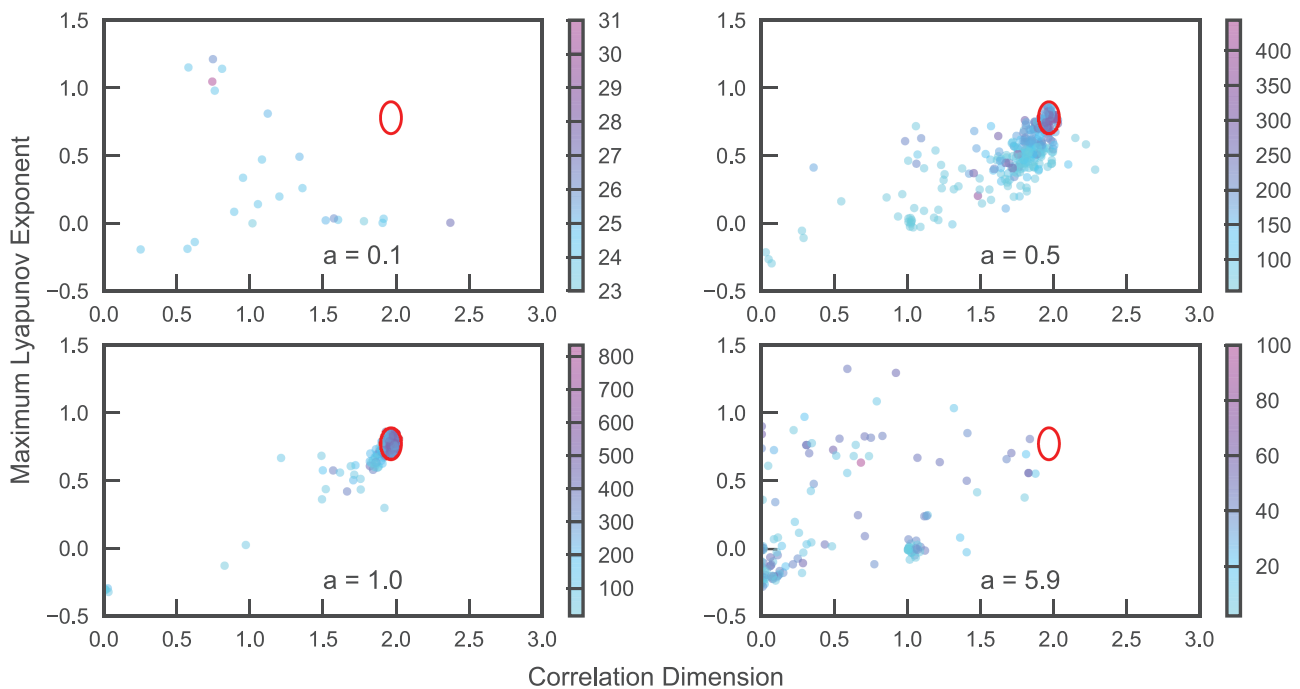


FIG. 5. Largest Lyapunov exponent scattered against correlation dimension for different values of the nonlinear scaling parameter a based on $N = 300$ realizations each. The colors denote the forecast horizon of the predictions and the red ellipses show the three σ errors of the correlation dimension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

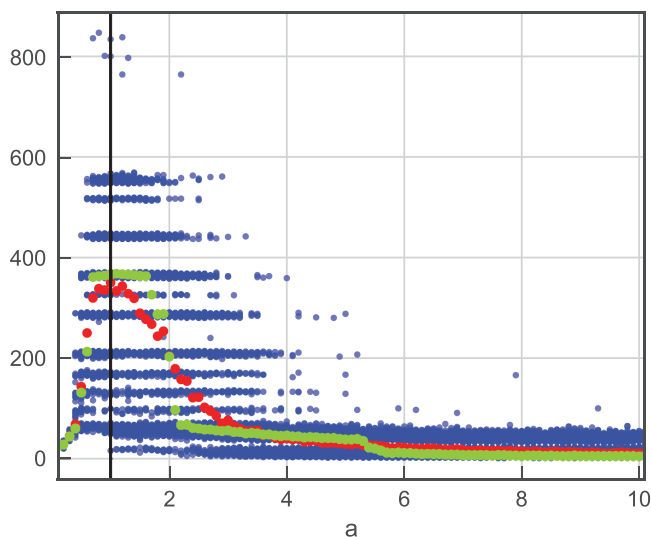


FIG. 6. Forecast horizon of the modified Lorenz system plotted for different values of the nonlinear scaling parameter a with $N = 300$ realizations for each a (blue). Furthermore, the average (red) and the median (green) values are shown for each value of a . The black horizontal line marks the optimal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

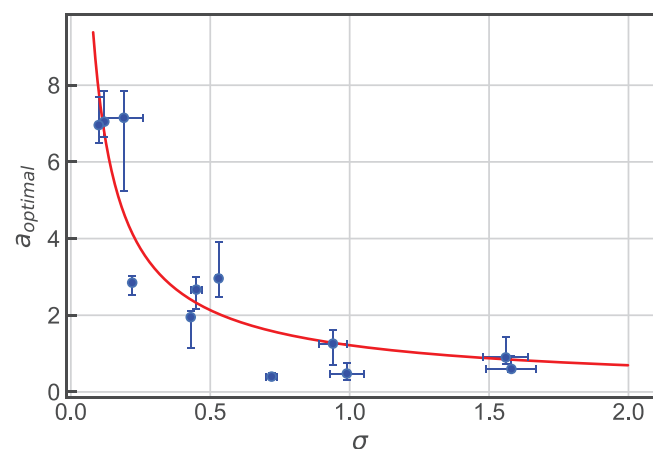


FIG. 7. Blue dots: Optimal value for the nonlinear scaling parameter a —based on the maximum of the average forecast horizon over all realizations for a given a —plotted against the standard deviation σ of the input data. The vertical bars denote the range of values for a , where the average forecast 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.

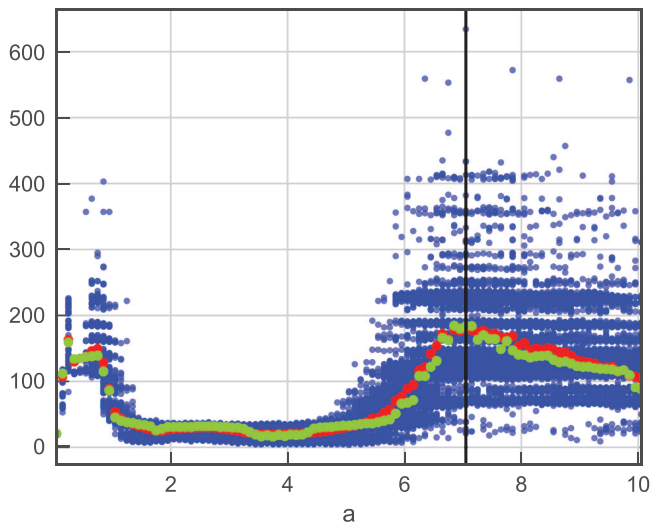


FIG. 8. Forecast horizon of the Chua circuit plotted for different values of the nonlinear scaling parameter a with $N = 300$ realizations for each a (blue). Furthermore, the average (red) and the median (green) values are shown for each value of a . The black horizontal line marks the optimal choice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

- ¹H. Jaeger and H. Haas, "Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication," *Science* **304**, 78–80 (2004).
- ²H. 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.
- ³W. 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).
- ⁴Z. 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).
- ⁵J. 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).
- ⁶J. 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).
- ⁷J. 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).
- ⁸R. S. Zimmermann and U. Parlitz, "Observing spatio-temporal dynamics of excitable media using reservoir computing," *Chaos* **28**, 043118 (2018).
- ⁹T. L. Carroll, "Using reservoir computers to distinguish chaotic signals," *Phys. Rev. E* **98**, 052209 (2018).
- ¹⁰Z. Lu, B. R. Hunt, and E. Ott, "Attractor reconstruction by machine learning," *Chaos* **28**, 061104 (2018).
- ¹¹P. 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).
- ¹²N. A. K. Doan, W. Polifke, and L. Magri, "A physics-aware machine to predict extreme events in turbulence," [arXiv:1912.10994](https://arxiv.org/abs/1912.10994) (2019).
- ¹³A. Haluszczynski and C. R ath, "Good and bad predictions: Assessing and improving the replication of chaotic attractors by means of reservoir computing," *Chaos* **29**, 103143 (2019).
- ¹⁴T. L. Carroll and L. M. Pecora, "Network structure effects in reservoir computers," [arXiv:1903.12487](https://arxiv.org/abs/1903.12487) (2019).
- ¹⁵X. 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).
- ¹⁶S. 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.
- ¹⁷S. 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.
- ¹⁸M. Luko evi cius and H. Jaeger, "Reservoir computing approaches to recurrent neural network training," *Comput. Sci. Rev.* **3**, 127–149 (2009).
- ¹⁹R. Gen ay and T. Liu, "Nonlinear modelling and prediction with feedforward and recurrent networks," *Physica D* **108**, 119–134 (1997).
- ²⁰P. Erdos, "On random graphs," *Publ. Math.* **6**, 290–297 (1959).
- ²¹A. E. Hoerl and R. W. Kennard, "Ridge regression: Biased estimation for nonorthogonal problems," *Technometrics* **12**, 55–67 (1970).
- ²²The search ranges for the hyperparameter optimization are 0–2.5 for $\rho(A)$, 0–2.0 for 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.
- ²³R. Albert, H. Jeong, and A.-L. Barab asi, "Error and attack tolerance of complex networks," *Nature* **406**, 378–382 (2000).
- ²⁴M. Inubushi and K. Yoshimura, "Reservoir computing beyond memory–nonlinearity trade-off," *Sci. Rep.* **7**, 1–10 (2017).
- ²⁵A. 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.
- ²⁶D. Verstraeten, B. Schrauwen, M. d'Haene, and D. Stroobandt, "An experimental unification of reservoir computing methods," *Neural Netw.* **20**, 391–403 (2007).
- ²⁷P. Verzelli, C. Alippi, and L. Livi, "Echo state networks with self-normalizing activations on the hyper-sphere," *Sci. Rep.* **9**, 1–14 (2019).
- ²⁸P. Grassberger and I. Procaccia, "Measuring the strangeness of strange attractors," *Physica D* **9**, 189–208 (1983).
- ²⁹P. Grassberger, "Generalized dimensions of strange attractors," *Phys. Lett. A* **97**, 227–230 (1983).
- ³⁰A. Wolf, J. B. Swift, H. L. Swinney, and J. A. Vastano, "Determining lyapunov exponents from a time series," *Physica D* **16**, 285–317 (1985).
- ³¹R. Shaw, "Strange attractors, chaotic behavior, and information flow," *Z. Naturforsch. A* **36**, 80–112 (1981).
- ³²M. 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).
- ³³E. N. Lorenz, "Deterministic nonperiodic flow," *J. Atmos. Sci.* **20**, 130–141 (1963).
- ³⁴O. E. R ossler, "An equation for continuous chaos," *Phys. Lett. A* **57**, 397–398 (1976).
- ³⁵A. 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).
- ³⁶G. Chen and T. Ueta, "Yet another chaotic attractor," *Int. J. Bifurcation Chaos* **9**, 1465–1466 (1999).
- ³⁷M. Rabinovich and A. Fabrikant, "Stochastic self-modulation of waves in nonequilibrium media," *J. Exp. Theor. Phys.* **77**, 617–629 (1979).
- ³⁸T. Matsumoto, L. Chua, and M. Komuro, "The double scroll," *IEEE Trans. Circuits Syst.* **32**, 797–818 (1985).
- ³⁹R. Thomas, "Deterministic chaos seen in terms of feedback circuits: Analysis, synthesis, "Labyrinth chaos," *Int. J. Bifurcation Chaos* **9**, 1889–1905 (1999).
- ⁴⁰A. M. Rucklidge, "Chaos in models of double convection," *J. Fluid Mech.* **237**, 209–229 (1992).
- ⁴¹J. C. Sprott and J. C. Sprott, *Chaos and Time-Series Analysis* (CiteSeerX, 2003), Vol. 69.
- ⁴²G. Tanaka, T. Yamane, J. B. H eroux, 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).
- ⁴³A. Griffith, A. Pomerance, and D. J. Gauthier, "Forecasting chaotic systems with very low connectivity reservoir computers," *Chaos* **29**, 123108 (2019).