


# Breaking symmetries of the reservoir equations in echo state networks

Cite as: Chaos **30**, 123142 (2020); <https://doi.org/10.1063/5.0028993>

Submitted: 08 September 2020 . Accepted: 03 December 2020 . Published Online: 22 December 2020

 Joschka Herteux, and Christoph R ath



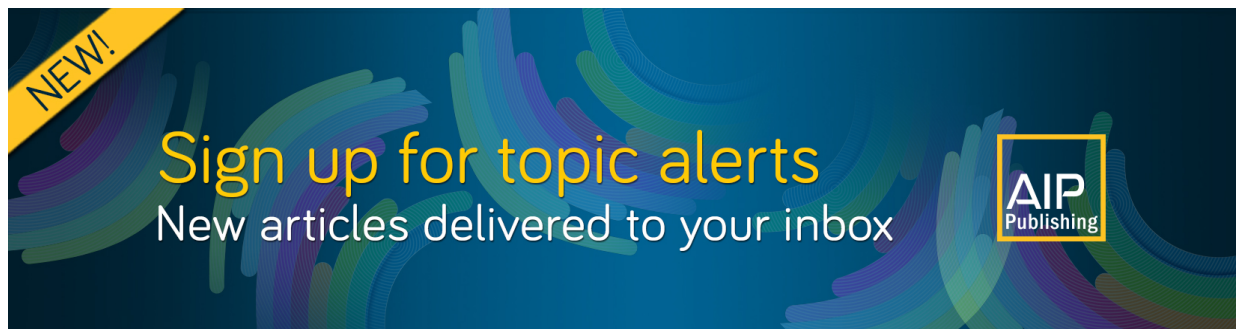
View Online



Export Citation



CrossMark



**NEW!**

Sign up for topic alerts  
New articles delivered to your inbox

AIP  
Publishing



# Breaking symmetries of the reservoir equations in echo state networks

Cite as: Chaos 30, 123142 (2020); doi: 10.1063/5.0028993

Submitted: 8 September 2020 · Accepted: 3 December 2020 ·

Published Online: 22 December 2020



View Online



Export Citation



CrossMark

Joschka Herteux<sup>a)</sup>  and Christoph R ath<sup>b)</sup>

## AFFILIATIONS

Institut f ur Materialphysik im Weltraum, Deutsches Zentrum f ur Luft- und Raumfahrt, M unchner Str. 20, 82234 Wessling, Germany

<sup>a)</sup> Author to whom correspondence should be addressed: [joschka.herteux@dlr.de](mailto:joschka.herteux@dlr.de)

<sup>b)</sup> Electronic mail: [christoph.raeth@dlr.de](mailto:christoph.raeth@dlr.de)

## ABSTRACT

Reservoir computing has repeatedly been shown to be extremely successful in the prediction of nonlinear time-series. However, there is no complete understanding of the proper design of a reservoir yet. We find that the simplest popular setup has a harmful symmetry, which leads to the prediction of what we call *mirror-attractor*. We prove this analytically. Similar problems can arise in a general context, and we use them to explain the success or failure of some designs. The symmetry is a direct consequence of the hyperbolic tangent activation function. Furthermore, four ways to break the symmetry are compared numerically: A bias in the output, a shift in the input, a quadratic term in the readout, and a mixture of even and odd activation functions. First, we test their susceptibility to the mirror-attractor. Second, we evaluate their performance on the task of predicting Lorenz data with the mean shifted to zero. The short-time prediction is measured with the forecast horizon while the largest Lyapunov exponent and the correlation dimension are used to represent the climate. Finally, the same analysis is repeated on a combined dataset of the Lorenz attractor and the Halvorsen attractor, which we designed to reveal potential problems with symmetry. We find that all methods except the output bias are able to fully break the symmetry with input shift and quadratic readout performing the best overall.

Published under license by AIP Publishing. <https://doi.org/10.1063/5.0028993>

Reservoir computing describes a kind of recurrent neural network, which has been very successful in the prediction of chaotic systems. However, the details of its inner workings have yet to be fully understood. One important aspect of any neural network is the activation function. Even though its effects have been extensively studied in other machine learning techniques, there are still open questions in the context of reservoir computing. Our research aims to fill this gap. We prove analytically that an antisymmetric activation function like the hyperbolic tangent leads to a disastrous symmetry in a popular setup we call simple ESN. This leads the reservoir to learn an inverted version of the training data we call *mirror-attractor*, which we demonstrate numerically. This heavily perturbs any prediction, especially if the mirror-attractor overlaps with the real attractor. Furthermore, we compare four different ways to break the symmetry. We test numerically whether they tend to learn the mirror-attractor and test their performance on two tasks where the simple ESN fails. We find that three of them are able to fully break the symmetry.

## I. INTRODUCTION

Machine learning (ML) has shown to be tremendously successful in categorization and recognition tasks, and the use of ML algorithms has become common in technical devices of daily living. But the application of ML also pervades more and more areas of science including research on complex systems. For a very recent collection, see Ref. 1 and references therein.

In nonlinear dynamics, ML-based methods have recently attracted a lot of attention, because it was demonstrated that the exact short-term prediction of nonlinear system can be significantly improved. Furthermore, it was shown that ML techniques also allow for a very accurate reproduction of the long-term properties (“the climate”) of complex systems.<sup>2,3</sup> Several ML methods like deep feed-forward artificial neural network (ANN), recurrent neural network (RNN) with long short-term memory (LSTM), and reservoir computing (RC) fulfill the prediction tasks.<sup>4,5</sup> RC has attracted most attention. It is a machine learning method that has been independently discovered as liquid state machines (LSMs) by Maass<sup>6</sup> and as

echo state networks (ESN) by Jaeger.<sup>7</sup> We focus here on the ESN approach, which falls under the category of recurrent neural networks (RNNs). The main difference to other RNNs such as LSTMs is that in RC only the last layer is explicitly trained via linear regression. Instead of hidden layers, it uses a so-called *reservoir*, which in the case of the ESN is typically a network with recurrent connections. The popularity of RC has several reasons. First, RC often shows superior performance. Second, ESNs offer conceptual advantages. As only the output layer is explicitly trained, the number of weights to be adjusted is very small. Thus, the training of ESNs is comparably transparent, extremely CPU-efficient (orders of magnitude faster than for ANNs), and the vanishing-gradient-problem is circumvented. Furthermore, small, smart, and energy-efficient hardware implementations using photonic systems,<sup>8</sup> spintronic systems,<sup>9</sup> and many more are conceivable and being developed<sup>10</sup> (and references therein). Ongoing research is focused on identifying the necessary conditions for a good reservoir. Recent studies focused mainly on the influence of the size and topology of the reservoir on the prediction capabilities.<sup>11–15</sup> Less attention has so far been paid on the role of activation and onto the overall performance of RC.

In this paper, we study in detail the sensitivity of RC to symmetries in the activation function. We reveal that previously reported shortcomings for simple ESNs can unambiguously be attributed to symmetry properties of the activation function (and not of the input signal). We propose and assess four different methods to break the symmetries that were developed for obtaining more reliable prediction results.

The paper is organized as follows: Section II first discusses the different measures used and the two test systems: the Lorenz and the Halvorsen equations. Afterward, the different ESN designs used in this study are introduced, and the symmetry of the simple ESN is proven. In Sec. III, the three numerical experiments we conducted and their results are presented. Finally, we discuss our findings in Sec. IV.

## II. METHODS

### A. Measures and system characteristics

#### 1. Forecast horizon

As in Refs. 13 and 14, we use the *forecast horizon* to measure the quality of short-time predictions. It is defined as the time between the start of a prediction and the point where it deviates from the test data more than a fixed threshold. The exact condition reads

$$|\mathbf{v}(t) - \mathbf{v}_R(t)| > \delta, \quad (1)$$

where the norm is taken elementwise. Due to the chaotic nature of our training data, any small perturbation will usually grow exponentially with time. Thus, this indicates the end of a reliable prediction of the actual trajectory. The measure is not very sensitive to the exact value of the threshold for the same reason.

The threshold generally depends on the direction, and we use  $\delta = (5.8, 8.0, 6.9)^T$  for the Lorenz system without preprocessing. In general, the values of  $\delta$  are chosen to be approximately 15% of the spatial extent of the respective attractor in the given direction. This is useful if the dynamics of a system takes place on different lengthscales.

### 2. Correlation dimension

To evaluate the climate of a prediction, we use two measures. To understand the structural complexity of the attractor, it is interesting to look at the correlation dimension. This is a way to quantify the dimensionality of the space populated by the trajectory.<sup>16</sup> The correlation dimension is based on the discrete form of the correlation integral

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

which returns the fraction of pairs of points that are closer than the threshold distance  $r$ .  $\theta$  represents the Heaviside function. The correlation dimension is then defined by the relation

$$C(r) \propto r^\nu \quad (3)$$

as the scaling exponent  $\nu$ . For a self-similar strange attractor, this relation holds in some range of  $r$ , which needs to be properly calibrated. Here, we adjusted it for every given problem beforehand on simulated data, which is not used for training or testing. We note that precision is not essential here, since we are only interested in comparisons and not in absolute values.

To get the correlation dimension for a given dataset, we use the Grassberger Procaccia algorithm.<sup>17</sup>

### 3. Largest Lyapunov exponent

The second measure we use to evaluate the climate is the largest Lyapunov exponent. In contrast to the correlation dimension, it is indicative of the development of the system in time. A  $d$ -dimensional chaotic system is characterized by  $d$  Lyapunov exponents of which at least one is positive. They describe the average rate of exponential growth of a small perturbation in each direction in phase space. The largest Lyapunov exponent  $\lambda$  is the one associated with the direction of the fastest divergence,

$$d(t) = C e^{\lambda t}. \quad (4)$$

Since it dominates the dynamics, it has a special significance. It can be calculated from data with relative ease by using the Rosenstein algorithm.<sup>18</sup> It is also possible to determine the complete Lyapunov spectrum from the equations, which we have access to for our testdata as well as for our ESNs.<sup>3,19</sup> However, we found that the comparison is clearer in our case with the data-driven approach, because it is completely independent of details of the system, e.g., the question if it is discrete or continuous. This method is also computationally less costly.

We can further define the *Lyapunov time*  $\tau_\lambda = \frac{1}{\lambda}$  as characteristic timescale of a system. We use  $\tau_\lambda = \frac{1}{0.87} \approx 1.15$  for the Lorenz system and  $\tau_\lambda = \frac{1}{0.74} \approx 1.35$  for the Halvorsen system, based on our measurements in Table II.

### B. Lorenz and Halvorsen systems

A standard example of a chaotic attractor is provided by the Lorenz system.<sup>20</sup> It is widely used as a test case for the prediction of

such systems with RC. It is defined by the equations

$$\begin{aligned}\dot{x} &= \sigma(y - x), \\ \dot{y} &= x(\rho - z) - y, \\ \dot{z} &= xy - \beta z + x,\end{aligned}\quad (5)$$

where we use the standard parameters  $\sigma = 10$ ,  $\beta = 8/3$ , and  $\rho = 28$ . We simulate the dynamics by integrating these equations using the Runge–Kutta method with time steps of  $\Delta t = 0.02$ . We use varying starting points on the attractor.

The equations are symmetric under the transformation  $(x, y, z) \rightarrow (-x, -y, z)$ . Thus the mirror-attractor differs only in the  $z$ -coordinate. Furthermore, the mean of the attractor in the  $z$ -direction is far away from the origin at  $\bar{z} \approx 23.5$ . This makes it an especially useful example for breaking the symmetry in the reservoir.

As a secondary test case, we use the Halvorsen equations<sup>21</sup>

$$\begin{aligned}\dot{x} &= -\sigma x - 4y - 4z - y^2, \\ \dot{y} &= -\sigma y - 4z - 4x - z^2, \\ \dot{z} &= -\sigma z - 4x - 4y - x^2,\end{aligned}\quad (6)$$

with  $\sigma = 1.3$ . We simulate the dynamics in the same way as for the Lorenz equations. This system also exhibits chaotic behavior but does not have any symmetries under inversion of its coordinates. It has a cyclic symmetry, which should, however, not be relevant in this context.

Both Lorenz and Halvorsen systems are three-dimensional autonomous dissipative flows.

### C. Reservoir computing and simple ESN

There is a multitude of ways to design an ESN. In this paper, we use several different variants, which will be introduced in the following sections. In general, the input is fed into the reservoir and influences its dynamics. A usually linear readout is trained to translate the state of the reservoir into the desired output. The reservoir state is then a random, high-dimensional, nonlinear transformation of all previous input data. This naturally gives it a kind of memory.

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

$$\mathbf{r}_{t+1} = f(\mathbf{A}\mathbf{r}_t, \mathbf{W}_{in}\mathbf{x}_t). \quad (7)$$

Here,  $f$  is called activation function, the adjacency matrix  $\mathbf{A} \in \mathbb{R}^{N \times N}$  represents the network,  $\mathbf{x}_t \in \mathbb{R}^{d_x}$  is the input fed into the reservoir, and  $\mathbf{W}_{in} \in \mathbb{R}^{d_x \times N}$  is the input matrix. There are many possible choices for  $f$  and ways to construct  $\mathbf{A}$  and  $\mathbf{W}_{in}$ . Here, we always create  $\mathbf{A}$  as an Erdős–Renyi random network. Sparse networks have been found to be advantageous.<sup>11</sup> The weights of the network are then drawn uniformly from  $[-1, 1]$  and afterward rescaled to fix the spectral radius  $\rho$  to some fixed value.  $\rho$  is a free hyperparameter.

We chose  $\mathbf{W}_{in}$  to be also sparse, in the sense that every row has only one nonzero element. This means every reservoir node is only connected to one degree of freedom of the input.<sup>2</sup> We fixed the number of nodes per dimension to be the same plus or minus one. The nonzero elements are drawn uniformly from the interval  $[-1, 1]$  and then rescaled with a factor  $s_{input}$ , which is another free hyperparameter.

From this, we can then compute the output  $\mathbf{y}_t \in \mathbb{R}^{d_y}$ . Here, we are interested in the prediction case, where we train the ESN to approximate  $\mathbf{y}_t \approx \mathbf{x}_{t+1}$ . Thus, the dimension of input and output are the same, so we use  $d_x = d_y := d$ . The readout is characterized by

$$\mathbf{y}_t = \mathbf{W}_{out}\tilde{\mathbf{r}}_t, \quad (8)$$

where typically  $\tilde{\mathbf{r}}_t = \mathbf{r}_t$ , but  $\tilde{\mathbf{r}}_t \in \mathbb{R}^{\tilde{N}}$  can also be some nonlinear transformation or extension of  $\mathbf{r}_t$ . The readout matrix  $\mathbf{W}_{out} \in \mathbb{R}^{d \times \tilde{N}}$  is the only part of the ESN that is trained. This is typically done via simple Ridge Regression.<sup>22</sup>

To train the reservoir, the training data  $\mathbf{x}^{train} = \{\mathbf{x}_0, \dots, \mathbf{x}_{T_{train}}\}$  is fed into the reservoir to get the sequence  $\mathbf{r}^{train} = \{\mathbf{r}_0, \dots, \mathbf{r}_{T_{train}+1}\}$ . The first  $T_{sync}$  time steps of  $\mathbf{r}$  are then discarded. This transient period is only used to synchronize the reservoir with the training data. This frees the ESN of any influence of the reservoir's initial condition thanks to the *fading memory property*. The state of a properly designed reservoir continuously loses its dependence on past states over time. For a detailed description, see e.g., Refs. 6, 23, and 24.

Now the readout matrix can be calculated by minimizing

$$\sum_{T_{sync} \leq t \leq T_{train}} \|\mathbf{W}_{out}\tilde{\mathbf{r}}_t - \mathbf{v}_t\|^2 - \beta \|\mathbf{W}_{out}\|^2, \quad (9)$$

where we get another hyperparameter  $\beta$  from regularization. The target output  $\mathbf{v}_t$  is in the case of prediction just  $\mathbf{x}_{t+1}$ . We, thus, get<sup>22</sup>

$$\mathbf{W}_{out} = \left(\tilde{\mathbf{r}}^T\tilde{\mathbf{r}} + \beta\mathbb{1}\right)^{-1}\tilde{\mathbf{r}}^T\mathbf{v}, \quad (10)$$

where  $\mathbf{r}$  is  $\mathbf{r}^{train}$  in the matrix form after discarding the synchronization steps and  $\mathbf{v}$  is analogous.

In the following, we always use a network with  $N = 200$ ,  $T_{train} = 10\,500$ , and  $T_{sync} = 500$  and average degree  $k = 4$  unless otherwise stated. The spectral radius  $\rho$ , the regularization parameter  $\beta$ , and the input scaling  $s_{input}$  are optimized for specific problems.

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

$$\mathbf{r}_{t+1} = \tanh(\mathbf{A}\mathbf{r}_t + \mathbf{W}_{in}\mathbf{x}_t), \quad (11)$$

$$\mathbf{y}_t = \mathbf{W}_{out}\mathbf{r}_t. \quad (12)$$

The activation function is a sigmoidal function, specifically a hyperbolic tangent, which is the typical choice. The reservoir states are not transformed before the readout. With this setup, successful predictions of different datasets have been made in many cases. However, we can sometimes see very specific ways in which they fail as illustrated in Figs. 1 and 2. When predicting the Lorenz attractor (see Sec. II B), the prediction sometimes jumps to an inverted version of the training dataset, which we call *mirror-attractor*. To investigate the severity of the problem, we created 1000 realizations with predictions of 500 000 time steps. We found that in 98.5% of cases, where the prediction crossed the zero in the  $z$ -direction, this led to a jump to the other attractor. 18% of the predictions did not exhibit a jump at any point. Among those that did, the average time of the



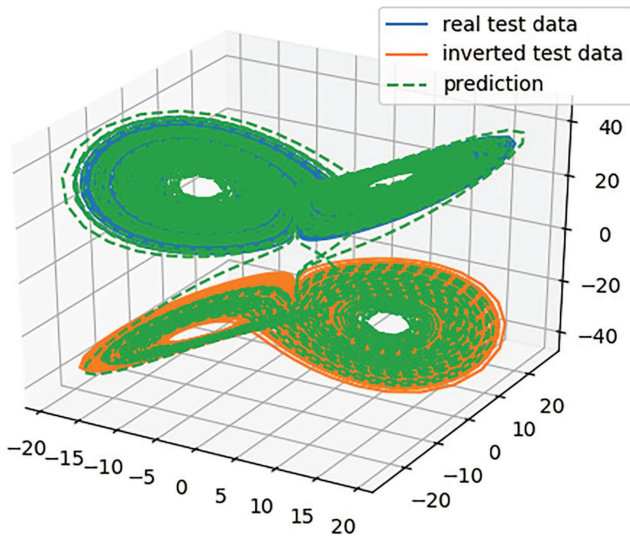


FIG. 1. Failed prediction of the Lorenz system with a simple ESN. The trajectory jumps down to the mirror-attractor.

first jump was about 31 000 time steps (539 Lyapunov times). Typically, the trajectory changed from one attractor to the other multiple times.

While this is concerning, it still allows for decent short-time predictions. We were able to reach average forecast horizons of about 400 time steps (7 Lyapunov times) after hyperparameter

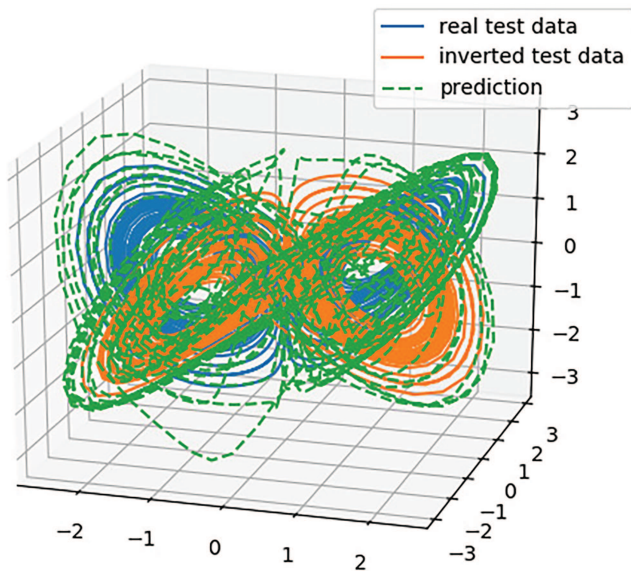


FIG. 2. Failed prediction of zero-mean, normalized Lorenz data. The trajectory jumps frequently between the original and the mirror-attractor.

optimization. However, when the data are brought to zero-mean, the ability to make accurate predictions largely breaks down. After hyperparameter optimization, we get a forecast horizon of about 90 time steps (1.6 Lyapunov times). Since the two attractors overlap, the prediction jumps between them very frequently as we can see in Fig. 2. Sometimes, it even travels outside of both for a short time. Since this kind of preprocessing is considered good practice in machine learning and usually leads to better results, this shows a severe failure of the method.

Difficulties in predicting the Lorenz system, which is a widespread test case for ESNs, when using this kind of setup have been noted by previous studies. They were linked to the symmetry of the Lorenz equation under transformation  $(x, y, z) \rightarrow (-x, -y, z)$ . However, we observe these problems with other datasets as well. We can largely explain this phenomenon by a mathematical analysis independent of the input data.

To prove this, we do the following: Assume  $\mathbf{r}_0 = \mathbf{0}$  w.l.o.g. because of the fading memory property. Let us now analyze what happens, when instead of the original training sequence  $\mathbf{x}^{train} = \{\mathbf{x}_0, \dots, \mathbf{x}_{T_{train}}\}$  we use its inverted version  $-\mathbf{x}^{train} = \{-\mathbf{x}_0, \dots, -\mathbf{x}_{T_{train}}\}$  to train the readout matrix,

$$\mathbf{r}_0(-\mathbf{x}^{train}) = \mathbf{0} = -\mathbf{r}_0(\mathbf{x}^{train}), \tag{13}$$

$$\begin{aligned} \mathbf{r}_1(-\mathbf{x}^{train}) &= \tanh(-\mathbf{W}_{in}\mathbf{x}_n), \\ &= -\tanh(\mathbf{W}_{in}\mathbf{x}_n), \end{aligned} \tag{14, 15}$$

$$= -\mathbf{r}_1(\mathbf{x}^{train}). \tag{16}$$

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

$$\mathbf{r}_t(-\mathbf{x}^{train}) = -\mathbf{r}_t(\mathbf{x}^{train}). \tag{17}$$

Then,

$$\mathbf{r}_{t+1}(-\mathbf{x}^{train}) = \tanh(\mathbf{A}\mathbf{r}_t(-\mathbf{x}^{train}) - \mathbf{W}_{in}\mathbf{x}_t) \tag{18}$$

$$= -\tanh(\mathbf{A}\mathbf{r}_t(\mathbf{x}^{train}) + \mathbf{W}_{in}\mathbf{x}_t) \tag{19}$$

$$= -\mathbf{r}_{t+1}(\mathbf{x}^{train}). \tag{20}$$

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

Obviously, because of the linearity of the readout, we also get

$$\mathbf{y}_t(-\mathbf{x}^{train}) = -\mathbf{W}_{out}\mathbf{r}_t(\mathbf{x}^{train}) = -\mathbf{y}_t(\mathbf{x}^{train}). \tag{21}$$

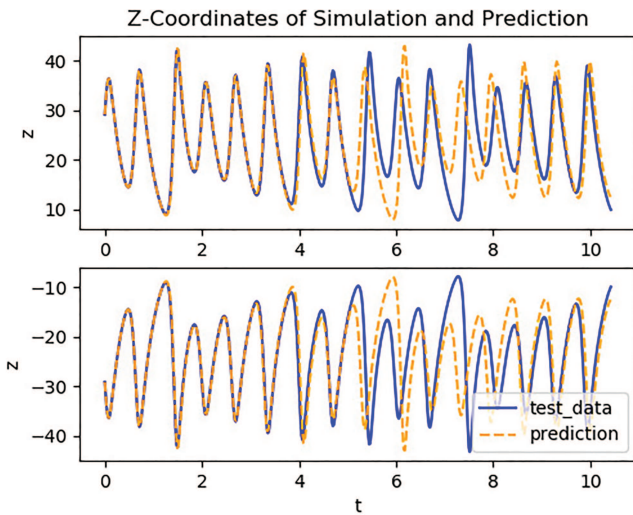
Finally,

$$\mathbf{W}_{out}(-\mathbf{x}) = (\mathbf{r}^T(-\mathbf{x})\mathbf{r}(-\mathbf{x}) + \beta\mathbb{1})^{-1} \mathbf{r}^T(-\mathbf{x})(-\mathbf{x}) \tag{22}$$

$$= (\mathbf{r}^T(\mathbf{x})\mathbf{r}(\mathbf{x}) + \beta\mathbb{1})^{-1} \mathbf{r}^T(\mathbf{x})\mathbf{x} \tag{23}$$

$$= \mathbf{W}_{out}(\mathbf{x}), \tag{24}$$

so training the simple ESN with inverted data is equivalent to training on the original data, and both lead to learning the exact same parameters. Thus, it can never map these sequences to either the same output or any output that differs by anything other than the sign. It is, therefore, not universal. It can only fully learn the dynamics of systems that are themselves point symmetric at the origin.



**FIG. 3.** Prediction of the z-coordinate after synchronization with original data (upper) and inverted data (lower) for simple ESN. The  $t$ -axis is given in units of Lyapunov times.

Furthermore, when we use a simple ESN for prediction, it is now obvious that it learns to replicate the inverted mirror-attractor and the real attractor. In cases where they overlap, they are however incompatible. When they do not overlap but are close enough to each other, this makes jumps possible.

In Fig. 3, this is demonstrated by comparing the predictions of an already trained simple ESN after being synchronized with additional Lorenz data either unchanged or inverted. We can see that the prediction of inverted data is simply the inversion of the prediction of the original data, just as expected from theory.

For a jump to happen, the reservoir has to arrive at a state that matches better with the mirror-attractor than with the real one. Since the reservoir has memory, it is not obvious how fast input data from the phase-space region of the mirror-attractor can actually make that happen. Empirically, we found that crossing the zero in the  $z$ -direction leads to a jump in 98.5% of cases. We further observed that inverting the input in a single time step was reliably enough to push the prediction on the mirror-attractor. This implies a strong sensitivity to the input data with regards to inducing jumps.

It is clear that this kind of symmetry creates a significant limitation for the simple ESN. We can, therefore, easily explain the previous problems with this kind of approach as well as the so far mostly empirical success of some methods combating them. In many recent publications, the readout was extended with some kind of nonlinear transformation. The empirical advantage of this has been explored without theoretical explanation by Chattopadhyay *et al.*<sup>5</sup> Typically, quadratic terms are included in the readout (see Sec. II D 2). This was to our knowledge originally introduced by Lu *et al.*<sup>25</sup> in order to specifically solve a problem relating to the symmetry of the Lorenz equations, when using the ESN as an observer. It has since been used successfully in many more general cases without theoretical explanation. From our analysis, it is now clear that

this readout breaks the antisymmetry of the ESN as a whole, which is completely independent of any symmetries of the input data.

A similar analysis can be fruitful on many different designs of ESN. For example in a recent Paper by Carroll and Pecora,<sup>12</sup> the following update equation was used:

$$r_i(t + 1) = \alpha r_i(t) + (1 - \alpha) \tanh, \tag{25}$$

$$\left( \sum_{j=1}^M A_{ij} r_j(t) + w_i x(t) + 1 \right), \tag{26}$$

where  $w_i$  are the elements of what they call *input vector*. Empirically, they found the performance suffered for  $w_i = 1 \forall i$  compared to  $w_i \in \{+1, -1\}$ . We can explain this with a symmetry under the transformation  $s(t) \rightarrow -s(t) - \frac{2}{w}$  for any constant  $w_i = w \forall i$  similar to what we found for the simple ESN. As soon as  $w_i$  takes on different values for different  $i$ , this symmetry is broken.

### D. Breaking symmetry

To better understand what is the best way to break the harmful antisymmetry in the simple ESN, we test four different designs. There are two main ways of approaching the problem. We can either break the symmetry in the reservoir, e.g. by changing the activation function, or in the readout. When choosing the latter option, Eq. (17) still holds. The dynamics of the reservoir still do not change meaningfully with the sign of the training data. Only during prediction does the influence of the readout actually come into play.

We use two designs following each approach.

#### 1. Output bias

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

$$\mathbf{y}_t = \mathbf{W}_{out} \tilde{\mathbf{r}}_t = \tilde{\mathbf{W}}_{out} \mathbf{r}_t + \mathbf{b}, \tag{27}$$

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

$$\mathbf{y}_t(-\mathbf{r}_t) = -\tilde{\mathbf{W}}_{out} \mathbf{r}_t + \mathbf{b} = -\mathbf{y}_t(\mathbf{r}_t) + 2\mathbf{b}. \tag{28}$$

We note, however, that this way there are only  $d$  parameters to represent the difference under sign-change.

#### 2. Lu readout

As previously mentioned, a quadratic extension of the readout has recently become popular after its introduction by Lu *et al.*<sup>25</sup> We use two different variants of this approach. In its most powerful version, it consists of using  $\tilde{\mathbf{r}} = \{r_1, r_2, \dots, r_N, r_1^2, r_2^2, \dots, r_N^2\}$  in the readout.

Effectively, we get

$$\mathbf{y}_t = \mathbf{W}_{out}\tilde{\mathbf{r}}_t = \mathbf{W}_{out}^1\mathbf{r}_t + \mathbf{W}_{out}^2\mathbf{r}_t^2, \tag{29}$$

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

$$\mathbf{y}_t(-\mathbf{r}_t) = -\mathbf{W}_{out}^1\mathbf{r}_t + \mathbf{W}_{out}^2\mathbf{r}_t^2 \tag{30}$$

$$= -\mathbf{y}_t(\mathbf{r}_t) + 2\mathbf{W}_{out}^2\mathbf{r}_t^2. \tag{31}$$

The number of parameters that represent the difference under sign-change is  $d \times N$ . In the following, we will call this *extended Lu readout*. We use this in order to test the full potential of this approach. However, the higher number of parameters in the output matrix makes a quantitative comparison to other approaches unfair. For this purpose, we also test a second version.

In the original work by Lu *et al.*, only half of the nodes are squared in the readout, and each is only used either in its linear or in its quadratic form. This can be achieved with a transformation of the reservoir state like  $\tilde{\mathbf{r}} = \{r_1, r_2^2, r_3, r_4^2, \dots, r_{N-1}, r_N^2\}$ , where we assumed  $N$  to be even. We call this as *Lu readout or regular Lu readout*. This way the number of parameters is unaffected, which makes a fair comparison with the other methods possible.

### 3. Input shift

This design for an ESN has been proven to be universal by Grigoryeva and Ortega.<sup>23</sup> Specifically, this means it can approximate any causal and time-invariant filter with the fading-memory property, which naturally excludes any problems with symmetry. This is also recommended in “A practical guide to applying echo state networks” by Lukoševičius.<sup>26</sup>

For this design, the activation function of the simple ESN is extended by including a random bias term in every node of the reservoir. It can be written as a random vector  $\gamma \in \mathbb{R}^N$  and gives the following new update equation:

$$\mathbf{r}_{t+1} = \tanh(\mathbf{A}\mathbf{r}_t + \mathbf{W}_{in}\mathbf{x}_t + \gamma). \tag{32}$$

The readout is unchanged from the simple ESN. Unlike the first two methods, this breaks the symmetry in the reservoir itself.

We draw the elements of  $\gamma$  uniformly from  $[-s_\gamma, s_\gamma]$ , where  $s_\gamma$  is a new hyperparameter to be optimized. This was a somewhat arbitrary choice for simplicity. In principle, we could instead use a normal distribution, the distribution of the training data, etc.

### 4. Mixed activation functions

Another way of breaking the symmetry directly in the reservoir is to replace some of the odd tanh activation functions with even functions. This was inspired by a different framing of the problem: Every node in the network can be understood as a function of the concatenation of input datum and reservoir state  $\tilde{\mathbf{x}} = \{r_1, \dots, r_N, x_1, \dots, x_d\}$ . The readout is then simply a linear combination of these functions, where the weights are optimized to approximate the output. The nodes differ by the random parameters introduced through the elements of  $\mathbf{W}_{in}$ ,  $\mathbf{A}$  and  $\gamma$ , if input shift is included. In general, we want this set of functions to approximate a basis in the corresponding function space to be as powerful as possible. In the case of the simple ESN, these functions are all odd and any

linear combination of them will still be odd. Mixing in even functions gives us access to the whole function space as any function can be divided in an even and an odd part.

As even function, we simply used  $\tanh^2$ . We assigned half of the nodes connected to each input dimension to be even nodes, where this activation function is used.

## III. RESULTS

### A. Predicting the mirror-attractor

To test the ability of the four methods to break the symmetry of the simple ESN, we tried to force them to predict the mirror-attractor of the Lorenz equations after being trained with regular data. If the symmetry is truly broken, we expect this prediction to fail completely, indicating that the ESN did not learn anything about the mirror-attractor.

To accomplish this, we trained our ESNs with regular Lorenz data and then synchronized it with the inverted next 500 time steps of the simulation. We then measured the forecast horizon of the prediction in regards to the (also inverted) test data. For comparison, we also looked at the prediction after synchronization with the same data without inversion.

In Fig. 4, we see the behavior of the ESN with output bias. It differs from before in that the prediction of the mirror-attractor is not simply the inversion of the regular prediction as for the simple ESN in Fig. 3. However, even though it is generally a worse prediction, it clearly follows the inverted trajectory. The ESN has still learned a slightly perturbed version of the mirror-attractor.

When using input shift, Lu readout or mixed activations, we never observed a prediction staying in the vicinity of the mirror-attractor. Most of them instead leave it immediately and quickly

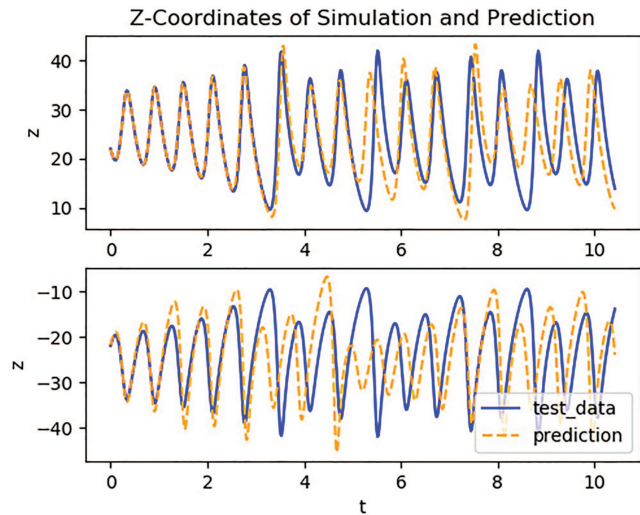


FIG. 4. Prediction of the z-coordinate after synchronization with original data (upper) and inverted data (lower) for the ESN with output bias. The  $t$ -axis is given in units of Lyapunov times. The prediction of inverted data is perturbed but still in the mirror-attractor.



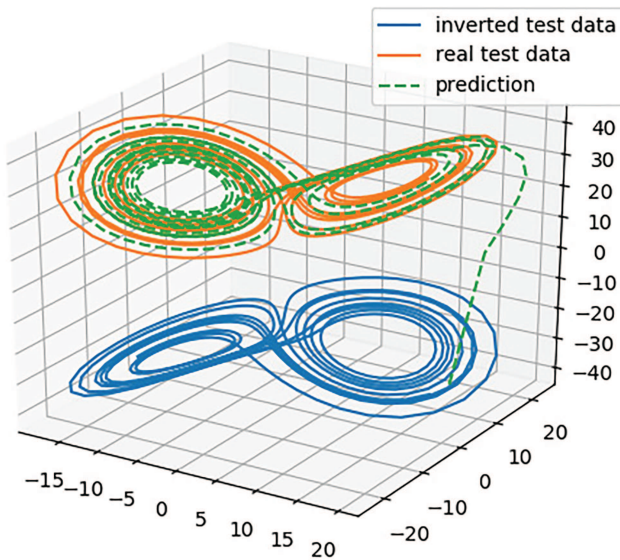


FIG. 5. Prediction after synchronizing with inverted data for the ESN with Lu readout.

converge to the real Lorenz attractor as in Fig. 5. In some cases, the trajectory finds some other fixed point instead, but it never stays in the mirror-attractor. Qualitatively, we get the same behavior when using mixed activations or input shift instead.

Furthermore, we made 1000 predictions of the mirror-attractor with all four designs while varying the network and the starting point of the training data. The distribution of forecast horizons is shown in Fig. 6. The output bias clearly sticks out as the only method showing the ability to predict the mirror-attractor. Some realizations reach forecast horizons up to one Lyapunov time, while the other methods never go beyond 0.1 Lyapunov times corresponding to only  $O(1)$  time steps. An exception is the regular Lu readout, where the prediction tends to stay on the mirror-attractor slightly longer than for input shift, extended Lu readout, and mixed activation functions. However, all of these trajectories converge to the original attractor afterward.

We also tested the rate of jumps between the attractors for the ESN with output bias analogously to the simple ESN in Sec. II C by making 1000 predictions with 500 000 time steps. Network and training data were varied for each realization. This time 30.5% of them did not show any jump. For the others, the average time of the first jump was after about 33 000 time steps (574 Lyapunov times). 95.2% of times when the  $z$ -coordinate crossed zero, it lead to a jump. This might indicate a small improvement, but the fundamental problem has not been solved by the output bias.

We did not observe any jumps when we used the other methods.

### B. Zero-mean Lorenz

To further compare the performance of the different ESNs, we test the ability to learn and predict Lorenz data where the mean has

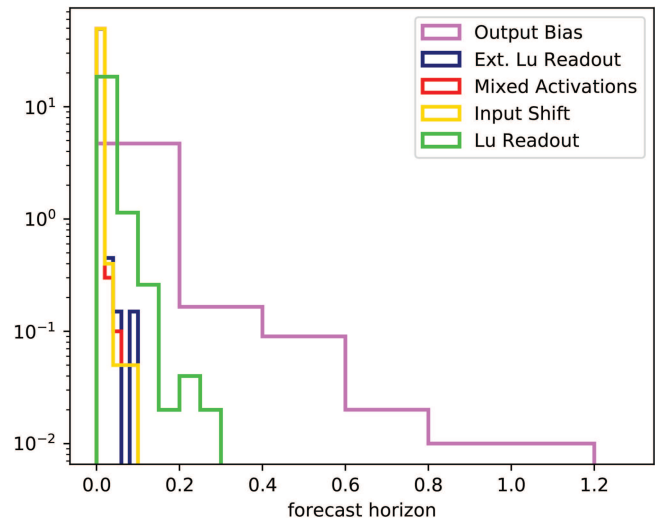


FIG. 6. Normalized histogram of the forecast horizon (in units of Lyapunov times) with respect to the inverted test data after synchronizing with inverted training data for the four different symmetry-breaking designs.

been shifted to the origin. As already discussed, this leads to overlap between the real and the mirror-attractor. So even though this preprocessing is usually preferred, it makes the simple ESN's problems with antisymmetry, especially severe. We also rescaled all data to have a standard deviation of 1.

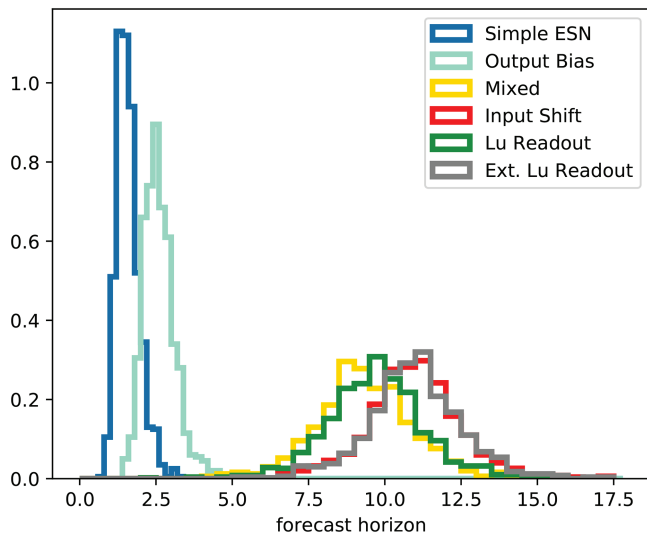
To get reliable quantitative results, we first carried out a hyperparameter optimization on this task for every design used. We used a grid search with 100 realizations for each point in parameter space. The best parameters are chosen on the basis of the highest average forecast horizon. Further details and results can be found in Appendix.

With the optimized hyperparameters, we created 1000 realizations for each design and measured forecast horizon, largest Lyapunov exponent, and correlation dimension. To accurately represent the climate, we used predictions with a length of 20 000 time steps. The results are compiled in Table I and Figs. 7 and 8.

TABLE I. Performance of the different ESN designs on zero-mean Lorenz data. Comparison to the original Lorenz data in last row. Forecast horizon (F.H.) given in units of time steps and Lyapunov times in brackets.

ESN design	F.H. in $\Delta t (\tau_\lambda)$	$\lambda \pm \sigma$	$\nu \pm \sigma$
Simple ESN	90.9(1.6)	$0.2 \pm 0.2$	$1.6 \pm 0.5$
Output bias	149.7(2.6)	$0.3 \pm 0.2$	$2.0 \pm 0.5$
Mixed activations	538.2(9.4)	$0.87 \pm 0.03$	$1.97 \pm 0.13$
Input shift	629.3(11.0)	$0.87 \pm 0.02$	$1.978 \pm 0.008$
Lu readout	558.4(9.7)	$0.87 \pm 0.04$	$1.96 \pm 0.17$
Ext. Lu readout	631.3(11.0)	$0.87 \pm 0.02$	$1.978 \pm 0.008$
Test data	$\infty$	$0.87 \pm 0.02$	$1.978 \pm 0.008$



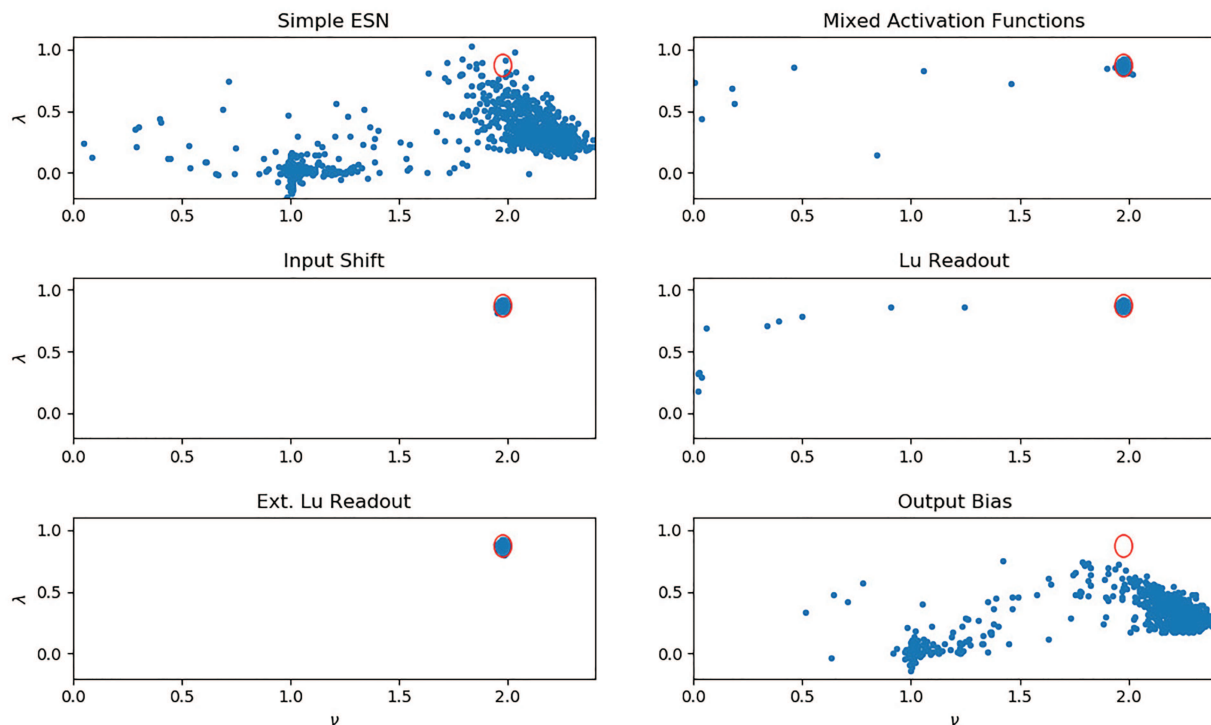


**FIG. 7.** Normalized histogram of forecast horizons (in units of Lyapunov times) for normalized, zero-mean Lorenz data with all five variants of ESN. Based on 1000 realizations (varying training data, network, and starting point of prediction) for each.

The simple ESN’s forecast horizon consistently lies in a region below 3.5 Lyapunov times with a mean of 1.6, and similarly to the first task, the output bias offers a noticeable but small improvement with a mean of 2.6 Lyapunov times. The other four methods to break the symmetry all seem to work in principle. Their forecast horizons mostly lie between 7 and 14 Lyapunov times. Extended Lu readout and input shift show no significant difference with an average forecast horizon of about 11 Lyapunov times, while mixed activation functions and regular Lu readout show a lower average forecast horizon of 9.4 and 9.6, respectively.

In agreement with the results for the forecast horizon, the simple ESN’s climate produces values of largest Lyapunov exponent and correlation dimension far away from the desired region. Again the output bias is only a small improvement. All results for extended Lu readout and input shift lie in the direct vicinity of the target, and the mean values and standard deviations match those of the test data within uncertainty. Regular Lu readout and mixed activation functions also reproduce the correct mean values but with much higher variance. This can be attributed to the clear outliers visible in the plots. We note that the results for the climate are less reliable than those for the forecast horizon since we did not optimize the hyperparameters for this task.

In some cases, the predicted trajectory diverged completely or got stuck in a fixed point. This made the proper calculation of the



**FIG. 8.** Scatterplot of the largest Lyapunov exponent against the correlation dimension when predicting zero-mean Lorenz data. Red ellipse corresponds to five times the standard deviation of the test data. Based on 1000 realizations for each setup as in Fig. 7.

largest Lyapunov exponent impossible, and thus, these values are excluded from that statistic. This happened 6 times with the simple ESN setup, 30 times with mixed activation functions, and 5 times when using a regular Lu readout. Input shift, extended Lu readout, and even output bias did not show this behavior in any prediction in this experiment.

Overall using a regular Lu readout or mixed activation functions did not perform quite as well on this standard task as input shift and extended Lu readout.

### C. Halvorsen and Lorenz

Finally, we compare the five different ESNs on a task that we specifically designed to test their symmetry breaking abilities. For this goal, we create a dataset by simulating both the Lorenz and the Halvorsen systems. The mean of the Lorenz data is shifted to  $(1, 1, 1)$ , and the mean of the Halvorsen data is shifted to  $(-1, -1, -1)$ . Both are rescaled so that no datapoint has a distance higher than 1 from the mean in any dimension. This ensures that the two attractors do not overlap while lying completely in the region of each others' mirror-attractor. To train the ESN to simultaneously be able to predict both systems, we use the following trick. First, we synchronize it with the Lorenz data and record  $\mathbf{r}_{Lorenz}^{train}$  after the initial transient period. We do, however not calculate  $W_{out}$  yet. Instead, we repeat the process with the Halvorsen data. Now the transient period has the additional use of letting the reservoir forget about the Lorenz system. This way we get  $\mathbf{r}_{Lorenz}^{train}$  and  $\mathbf{r}_{Halvorsen}^{train}$ . We simply concatenate them to get a single dataset  $\mathbf{r}^{train}$ , from which we finally compute the readout matrix. As desired output, we use an analogous concatenation of the Lorenz data and the Halvorsen data. We note that, since the linear readout is in no way sensitive to the causal relationship between the reservoir states and the transient period at the second training stage was discarded, the transition between the two systems in itself does not influence training.

This way the ESN has to learn dynamics that are governed by a completely different set of equations instead of the mirror-attractor. In the end, it should be able to predict both attractors depending on the starting point of the reservoir states. Since this is a more difficult task and to make sure that possible failures are not just due to a lack of nodes, we use  $N = 500$  in this experiment. However, we made similar qualitative observations for smaller and larger networks.

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

Unsurprisingly, we observe that the simple ESN is not able to master this task (see Fig. 9). Most predictions completely diverge from both attractors. In the handful of cases where one of the predictions actually reproduced the climate of one attractor, the other one was always a complete failure. Qualitatively, we see the same results when including an output bias.

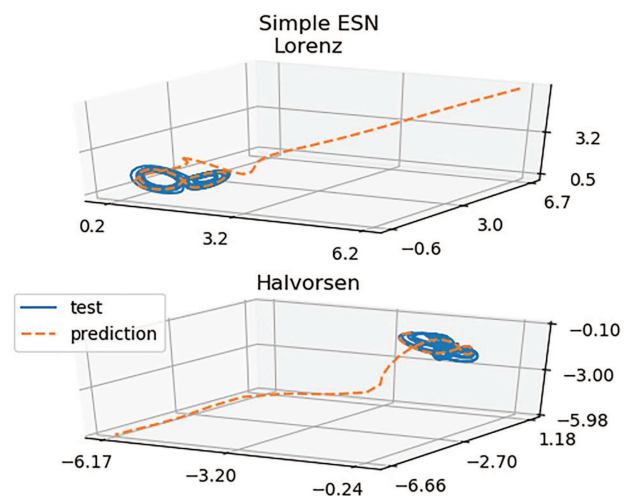
Since all predictions with the simple ESN and almost all with the ESN with output bias either diverged or converged to some fixed point, we could not provide meaningful results for the climate. We

**TABLE II.** Performance of the different ESN designs on combined Lorenz and Halvorsen data. Upper value is always Lorenz and lower Halvorsen. Comparison to the original data in last row. Forecast horizon (F.H.) given in units of time steps and Lyapunov times in brackets.

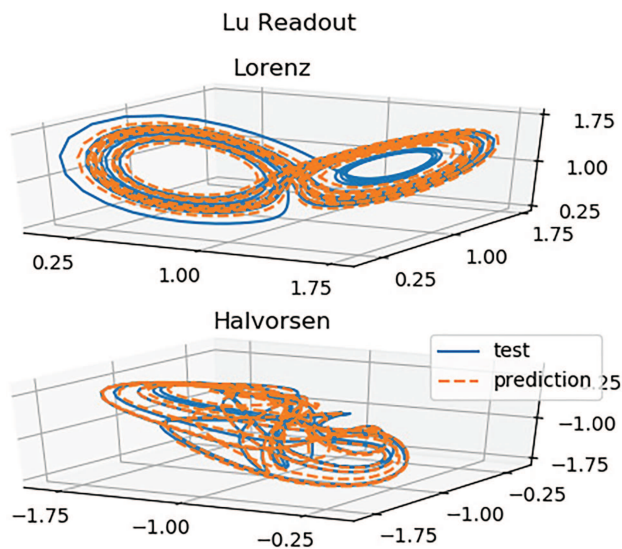
ESN design	F.H. in $\Delta t (\tau_\lambda)$	$\lambda \pm \sigma$	$\nu \pm \sigma$
Simple ESN	183.2(3.2)	...	...
	54.2(0.8)	...	...
Output bias	226.3(3.9)	...	...
	57.7(0.9)	...	...
Mixed activations	563.4(9.8)	$0.87 \pm 0.03$	$1.99 \pm 0.02$
	709.4(10.5)	$0.74 \pm 0.03$	$1.88 \pm 0.04$
Input shift	570.5(9.9)	$0.87 \pm 0.02$	$1.992 \pm 0.007$
	721.4(10.7)	$0.74 \pm 0.03$	$1.88 \pm 0.04$
Lu readout	561.8(9.7)	$0.87 \pm 0.02$	$1.97 \pm 0.18$
	719.4(10.7)	$0.74 \pm 0.03$	$1.87 \pm 0.04$
Ext. Lu readout	610.9(10.6)	$0.87 \pm 0.05$	$1.93 \pm 0.35$
	784.1(11.6)	$0.74 \pm 0.03$	$1.87 \pm 0.03$
Test data	$\infty$	$0.87 \pm 0.02$	$1.993 \pm 0.007$
	$\infty$	$0.74 \pm 0.03$	$1.87 \pm 0.04$

note, however, that the short-term prediction of the Lorenz system was actually significantly better for both than in Sec. III B. We attribute this to the higher number of nodes. Still the inability to reproduce long-term behavior indicates that this kind of problem can only be solved with a properly broken symmetry as we assumed.

Again the other methods to break the symmetry are all successful (see as an example Fig. 10) and predict both attractors with the same training quite well. The results in the forecast horizon for input shift, mixed activation functions, and the regular Lu readout are very similar with an average of about 9.8 Lyapunov times (Lorenz) and



**FIG. 9.** Example of predictions after training the same network on Lorenz and Halvorsen data simultaneously as described in Sec. III C. Here, we use the simple ESN setup.



**FIG. 10.** Example prediction of Lorenz and Halvorsen attractors with a single ESN with extended Lu readout.

10.7 Lyapunov times (Halvorsen). However, only the input shift was able to reproduce the climate with the same accuracy as the test data.

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

#### IV. DISCUSSION

In the present work, we showed a mathematical proof for the antisymmetry of the simple ESN with regards to changing the sign of the input. This is a consequence of the antisymmetry of the activation function. It makes it impossible to fully learn the dynamics of any attractor that is not point symmetric around the origin. In practice, we observed that the prediction jumps to an inverted version of the real attractor we call the mirror-attractor. This is especially disastrous if the two overlap. From this we conclude that this setup is not suitable for general tasks and should not be used.

Furthermore, we note that the sensitivity to this kind of symmetries with regards to the input is a universal property of ESNs and reservoir computers in general. This is in no way limited to the

specifics of the simple ESN. It must be kept in mind in every reservoir design and can explain the empirical success or failure of some of them.

In our experiments with the output bias, we found that formally breaking the symmetry alone is not enough to solve the problems associated with it. It was only able to improve the performance marginally and we still observed the appearance of an only slightly perturbed mirror-attractor. This might be due to the fact that the number of parameters representing the symmetry break in this approach is too low to accurately model the difference.

We were, however, able to successfully break the symmetry and solve the problem with three other approaches: Introducing an input shift in the activation function using a mixture of even and odd activation functions and including squared nodes in the readout. All of them were able to eliminate the mirror-attractor and make qualitatively good predictions even for zero-mean Lorenz data, where the overlap with the mirror-attractor is a severe problem for the simple ESN. They were further all able to master the task of predicting a dataset made of Lorenz and Halvorsen data, where it was necessary to learn completely different dynamics in the regime of the mirror-attractor.

The input shift proved in our test as a very useful and reliable tool to break the symmetry. The performance in all tasks was consistently good in short-time prediction. It successfully reproduced the climate statistics of the test data for the zero-mean Lorenz dataset and even for the combined Lorenz and Halvorsen data. This method was the only one to never produce outliers of completely failed predictions. It is also worth stressing that to our knowledge universality is only proven for ESNs with input shift. One disadvantage of this approach is the additional hyperparameter, which has to be optimized.

Even though the Lu readout in its regular form also seems to have broken the symmetry successfully, the results were generally not quite on par with the input shift. However, this gap was bridged by the extended Lu readout, which poses a convenient way to add more parameters to the model without increasing the size of the reservoir. For the zero-mean Lorenz data, this leads to a performance essentially identical with that of the input shift. In the case of the combined dataset, the short-time prediction surpassed the input shift, while the climate was not reproduced with the same accuracy. This improvement is likely due to the higher number of parameters and thus higher complexity in the readout. While this is of course accompanied by an increase in computational time, it demonstrates the general possibility to enhance the prediction abilities of a given reservoir by extending the readout. Even though the regular Lu readout seems to be enough to break the symmetry, using even higher order nonlinear transformations of nodes and an even bigger output matrix could further increase the performance. An application of this could be found in physical RC, where the dynamics of the reservoir might be inaccessible. One might however consider making the dynamics of the reservoir more complex, while keeping the simple linear readout, to be more in line with the philosophy of RC.

The ESN with mixed activation functions performed similarly well to the ESN with regular Lu readout. This is interesting, because the former can be understood as using squared nodes not only in the readout, but also in the dynamics of the reservoir. The results imply that this does not lead to a meaningful improvement. At least

for our implementation, we also found it to have a higher time cost. Thus, we do not recommend its use in the given form. However, the usage of different functions, different ratios, etc., could lead to better performance. Further research in this direction is needed.

In light of these results, we recommend both the input shift and the Lu readout as methods to break the symmetry.

**ACKNOWLEDGMENTS**

We wish to acknowledge useful discussions and comments from Jonas Aumeier, Sebastian Baur, Youssef Mabrouk, Alexander Haluszczynski, and Hubertus Thomas. We also want to thank our anonymous reviewers for their helpful suggestions.

**APPENDIX: HYPERPARAMETER OPTIMIZATION**

The hyperparameter optimization was carried out as a simple grid search with the aim to maximize the forecast horizon. For the simple ESN, we searched over  $a$  and  $\epsilon$ , with

$$s_{input} = a(1 - \epsilon), \tag{A1}$$

$$\rho = a\epsilon. \tag{A2}$$

The same was done for the ESN with output bias and the ESN with Lu readout. For the ESN with the input shift, we additionally varied the scale  $s_\gamma$ , and for the mixed activations, we replaced  $a$  with  $a_1$  and  $a_2$ , which were optimized for the tanh-nodes and the tanh<sup>2</sup>-nodes separately.

**1. Predicting the mirror-attractor**

Since we were less interested in quantitative results in this case, we did not perform a real hyperparameter optimization procedure. Instead, we used the parameters from our previous work<sup>14</sup> for the simple ESN and manually searched the parameters for the others to reproduce the Lorenz attractor reasonably well. This left us with the parameters in Table III.

**2. Zero-mean Lorenz**

In the case of the zero-mean Lorenz data, we simulated 100 trajectories of training data and 100 trajectories of test data. At every point in hyperparameter space, we generate a new network and a new  $W_m$  for each trajectory. We then choose the hyperparameters with the highest average forecast horizon.

**TABLE III.** Hyperparameter choices for the prediction of the mirror-attractor.

	Simple	Out. bias	Lu readout	Mixed	Inp. shift
$a$	0.32	0.32	0.32	...	0.32
$\epsilon$	0.5	0.5	0.5	0.5	0.5
$\beta$	$1.9 \times 10^{-11}$	$1.9 \times 10^{-11}$	$1.9 \times 10^{-11}$	$1.9 \times 10^{-11}$	$1.9 \times 10^{-11}$
$s_\gamma$	...	...	...	...	13.
$a_1$	...	...	...	0.32	...
$a_2$	...	...	...	0.32	...

**TABLE IV.** Hyperparameter range and results for the simple ESN on zero-mean Lorenz data.

Hyperparameter	Min	Max	Step size	Optimal
$a$	0.1	3.0	0.1	1.0
$\epsilon$	0.0	1.0	0.05	0.7

**TABLE V.** Hyperparameter range and results for the ESN with output bias on zero-mean Lorenz data.

Hyperparameter	Min	Max	Step size	Optimal
$a$	0.1	3.0	0.1	1.0
$\epsilon$	0.0	1.0	0.05	0.7

**TABLE VI.** Hyperparameter range and results for the ESN with regular Lu readout on zero-mean Lorenz data.

Hyperparameter	Min	Max	Step size	Optimal
$a$	0.1	3.0	0.1	0.9
$\epsilon$	0.0	1.0	0.05	0.55

**TABLE VII.** Hyperparameter range and results for the ESN with extended Lu readout on zero-mean Lorenz data.

Hyperparameter	Min	Max	Step size	Optimal
$a$	0.1	3.0	0.1	1.3
$\epsilon$	0.0	1.0	0.05	0.4

**TABLE VIII.** Hyperparameter range and results for the ESN with input shift on zero-mean Lorenz data.

Hyperparameter	Min	Max	Step size	Optimal
$a$	0.2	3.0	0.2	1.2
$\epsilon$	0.0	1.0	0.1	0.6
$s_\gamma$	0.3	3.0	0.3	1.5

**TABLE IX.** Hyperparameter range and results for the ESN with mixed activation functions on zero-mean Lorenz data.

Hyperparameter	Min	Max	Step size	Optimal
$a_1$	0.2	3.0	0.2	0.6
$a_2$	0.2	3.0	0.2	0.8
$\epsilon$	0.1	1.0	0.1	0.6



**TABLE X.** Hyperparameter range and results for the simple ESN on combined Lorenz and Halvorsen data.

Hyperparameter	Min	Max	Step Size	Optimal
$a$	1.1	4.0	0.1	2.0
$\epsilon$	0.0	1.0	0.05	0.4

**TABLE XI.** Hyperparameter range and results for the ESN with output bias on combined Lorenz and Halvorsen data.

Hyperparameter	Min	Max	Step size	Optimal
$a$	1.1	4.0	0.1	1.9
$\epsilon$	0.0	1.0	0.05	0.4

**TABLE XII.** Hyperparameter range and results for the ESN with regular Lu readout on combined Lorenz and Halvorsen data.

Hyperparameter	Min	Max	Step size	Optimal
$a$	1.1	4.0	0.1	1.9
$\epsilon$	0.0	1.0	0.05	0.3

**TABLE XIII.** Hyperparameter range and results for the ESN with extended Lu readout on combined Lorenz and Halvorsen data.

Hyperparameter	Min	Max	Step size	Optimal
$a$	1.1	4.0	0.1	2.3
$\epsilon$	0.0	1.0	0.05	0.45

**TABLE XIV.** Hyperparameter range and results for the ESN with input shift on combined Lorenz and Halvorsen data.

Hyperparameter	Min	Max	Step size	Optimal
$a$	1.2	4.0	0.2	3.0
$\epsilon$	0.0	1.0	0.1	0.1
$s_\gamma$	0.3	3.0	0.3	1.5

**TABLE XV.** Hyperparameter range and results for the ESN with mixed activation functions on combined Lorenz and Halvorsen data.

Hyperparameter	Min	Max	Step size	Optimal
$a_1$	1.2	4.0	0.2	2.8
$a_2$	1.2	4.0	0.2	2.4
$\epsilon$	0.1	1.0	0.1	0.2

Since it did not seem to depend strongly on the other hyperparameters, the problem or the specific design, we simply set  $\beta = 1.9 \times 10^{-11}$  as in the first task to save time with the already very costly grid search.

The results are compiled in [Tables IV–IX](#).

### 3. Halvorsen and Lorenz

For the combined dataset of Halvorsen and Lorenz attractors, we simulate 100 training and test trajectories of each system and use them as described in [Sec. III C](#). As before, we train a completely new reservoir on each trajectory for every point in parameter space. For every realization, the product of the two forecast horizons is calculated. The optimal hyperparameters are chosen to maximize this product averaged over the trajectories.

For this task, we did include the regularization parameter  $\beta$  in the search with a logarithmic scale from  $10^{-13}$  to 0.001 in 11 steps. We consistently found  $\beta = 10^{-10}$  to be the best choice for all designs.

The results are compiled in [Tables X–XV](#).

### DATA AVAILABILITY

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

### REFERENCES

- Y. Tang, J. Kurths, W. Lin, E. Ott, and L. Kocarev, "Introduction to focus issue: When machine learning meets complex systems: Networks, chaos, and nonlinear dynamics," *Chaos* **30**, 063151 (2020).
- Z. Lu, B. R. Hunt, and E. Ott, "Attractor reconstruction by machine learning," *Chaos* **28**, 061104 (2018).
- 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).
- P. R. Vlachas, J. Pathak, B. R. Hunt, T. P. Sapsis, M. Girvan, E. Ott, and P. Koumoutsakos, "Backpropagation algorithms and reservoir computing in recurrent neural networks for the forecasting of complex spatiotemporal dynamics," [arXiv:1910.05266 \[eess.SP\]](#) (2019).
- A. Chattopadhyay, P. Hassanzadeh, and D. Subramanian, "Data-driven predictions of a multi-scale Lorenz 96 chaotic system using machine-learning methods: Reservoir computing, artificial neural network, and long short-term memory network," *Nonlin. Processes Geophys.* **27**, 373–389 (2020).
- 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).
- H. Jaeger, "The 'echo state' approach to analysing and training recurrent neural networks—with an erratum note," German National Research Center for Information Technology GMD Technical Report No. 148. Jg., Nr. 34, S. 13 (2001).
- G. Van der Sande, D. Brunner, and M. C. Soriano, "Advances in photonic reservoir computing," *Nanophotonics* **6**, 561–576 (2017).
- D. Prychynenko, M. Sitte, K. Litzius, B. Krüger, G. Bourianoff, M. Kläui, J. Sinova, and K. Everschor-Sitte, "Magnetic skyrmion as a nonlinear resistive element: A potential building block for reservoir computing," *Phys. Rev. Appl.* **9**, 014034 (2018).
- 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).
- A. Griffith, A. Pomerance, and D. J. Gauthier, "Forecasting chaotic systems with very low connectivity reservoir computers," *Chaos* **29**, 123108 (2019).
- T. L. Carroll and L. M. Pecora, "Network structure effects in reservoir computers," [arXiv:1903.12487](#) (2019).

- <sup>13</sup>A. Haluszczyński 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).
- <sup>14</sup>A. Haluszczyński, J. Aumeier, J. Herteux, and C. R ath, “Reducing network size and improving prediction stability of reservoir computing,” *Chaos* **30**, 063136 (2020).
- <sup>15</sup>T. Carroll, “Path length statistics in reservoir computers,” *Chaos* **30**, 083130 (2020).
- <sup>16</sup>P. Grassberger and I. Procaccia, “Measuring the strangeness of strange attractors,” *Physica D* **9**, 189–208 (1983).
- <sup>17</sup>P. Grassberger, “Generalized dimensions of strange attractors,” *Phys. Lett. A* **97**, 227–230 (1983).
- <sup>18</sup>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).
- <sup>19</sup>M. Sandri, “Numerical calculation of Lyapunov exponents,” *Math. J.* **6**, 78–84 (1996).
- <sup>20</sup>E. N. Lorenz, “Deterministic nonperiodic flow,” *J. Atmos. Sci.* **20**, 130–141 (1963).
- <sup>21</sup>J. C. Sprott and J. C. Sprott, *Chaos and Time-Series Analysis* (Citeseer, 2003), Vol. 69.
- <sup>22</sup>A. E. Hoerl and R. W. Kennard, “Ridge regression: Biased estimation for nonorthogonal problems,” *Technometrics* **12**, 55–67 (1970).
- <sup>23</sup>L. Grigoryeva and J.-P. Ortega, “Echo state networks are universal,” *Neural Netw.* **108**, 495–508 (2018).
- <sup>24</sup>S. Boyd and L. Chua, “Fading memory and the problem of approximating non-linear operators with volterra series,” *IEEE Trans. Circuits Syst.* **32**, 1150–1161 (1985).
- <sup>25</sup>Z. Lu, J. Pathak, B. Hunt, M. Girvan, R. Broomhead, and E. Ott, “Reservoir observers: Model-free inference of unmeasured variables in chaotic systems,” *Chaos* **27**, 041102 (2017).
- <sup>26</sup>M. Luko evi cius, “A practical guide to applying echo state networks,” in *Neural Networks: Tricks of the Trade*, 2nd ed., edited by G. Montavon, G. B. Orr, and K.-R. M uller (Springer, Berlin, 2012), pp. 659–686.