scientific reports

Check for updates

OPEN

A novel approach to minimal reservoir computing

Haochun Ma[1], Davide Prosperino[1] & Christoph Räth[2]✉

Reservoir computers are powerful machine learning algorithms for predicting nonlinear systems. Unlike traditional feedforward neural networks, they work on small training data sets, operate with linear optimization, and therefore require minimal computational resources. However, the traditional reservoir computer uses random matrices to define the underlying recurrent neural network and has a large number of hyperparameters that need to be optimized. Recent approaches show that randomness can be taken out by running regressions on a large library of linear and nonlinear combinations constructed from the input data and their time lags and polynomials thereof. However, for high-dimensional and nonlinear data, the number of these combinations explodes. Here, we show that a few simple changes to the traditional reservoir computer architecture further minimizing computational resources lead to significant and robust improvements in short- and long-term predictive performances compared to similar models while requiring minimal sizes of training data sets.

The prediction of complex dynamic systems is a key challenge across various disciplines in science, engineering, and economics[1]. While machine learning approaches, like generative adversarial networks, can provide sensible predictions[2], difficulties with vast data requirements, the large number of hyperparameters, and lack of interpretability limit their usefulness in some scientific applications[3]. However, it is required to fundamentally understand how, when, and why the models are working to prevent the risk of misinterpreting the results if deeper methodological knowledge is missing[4].

In the context of complex systems research, reservoir computers (RCs)[5,6] have emerged for predicting the dynamics of chaotic systems. The core of the model is a fixed reservoir, which is usually constructed randomly[7–9]. The input data is fed into the nodes of the reservoir and solely the weights of the readout layer, which transform the reservoir response to output variables, are subject to optimization via linear regression. This makes the learning extremely fast and comparatively transparent. However, this approach can be hit-or-miss, and it is hardly possible to know a priori how the topology of the reservoir will affect the performance[10–12].

Recent research has emerged on algorithms which do not require randomness. They are built around regressions[13] on large libraries of linear and nonlinear combinations constructed from the data observations and their time lags, such as next generation reservoir computers (NG-RCs)[14] or sparse identification of nonlinear dynamics (SINDy)[15]. These algorithms are built around nonlinear vector autogression (NVAR)[16] and the mathematical fact that a powerful universal approximator can be constructed by using an RC with a linear activation function[17,18].

The model we present in this paper is based on the same mathematical principles — but instead of getting rid of the traditional reservoir architecture altogether, we take an intermediate step and make only a few simple changes: we restructure the input weights so that all coordinate combinations are fed separately into the reservoir. Additionally, we remove the randomness of the reservoir by replacing it with a block-diagonal matrix of blocks of ones. Instead of introducing the nonlinearity in the activation function, we add higher orders of the reservoir states in the readout.

Using the example of synthetic, chaotic systems, and in particular the Lorenz system, we show that these alterations lead to excellent short- and long-term predictions that significantly outperform traditional RC, NG-RC, and SINDy. While prediction performance is often evaluated visually, we use three quantitative measures: the largest Lyapunov exponent, the correlation dimension, and the forecast horizon. We also validate the robustness of our results by using multiple attractor starting points, different training data sizes and discretizations.

[1]Department of Physics, Ludwig-Maximilians-Universität, Schellingstraße 4, 80799 Munich, Germany. [2]Deutsches Zentrum für Luft- und Raumfahrt (DLR), Institut für KI Sicherheit, Wilhelm-Runge-Straße 10, 89081 Ulm, Germany. ✉email: christoph.raeth@dlr.de

Results

In this work, we show how small changes to the traditional RC architecture can significantly improve its prediction capability of chaotic systems especially for low data requirements. Therefore, similar to Gauthier et al.,[14] we use the minimal data setup for the Lorenz system with a discretization of dt=0.025 and T_{train}=400 training data points.

The minimal possible architecture would be a spectral radius ρ^*=0 and block-size b=1, for which our RC reduces to the case described by Gonon and Ortega.[17] Here, we do not have a reservoir and directly feed the input data to the readout and perform a Ridge regression. While we find this parametrization to be capable of reasonable predictions, a few minor alterations increase the performance significantly.

The standard RC architecture used in this work has block-size b=3, spectral radius ρ^*=0.1, and a nonlinearity degree η=2. This equals 36 variables per coordinate. The results of this setup are illustrated in Fig. 1.

In order to obtain robust results we repeat the analysis for 1000 different starting points on the attractor and compare the prediction performance to the other models. In Fig. 2 we see that the novel RC architecture

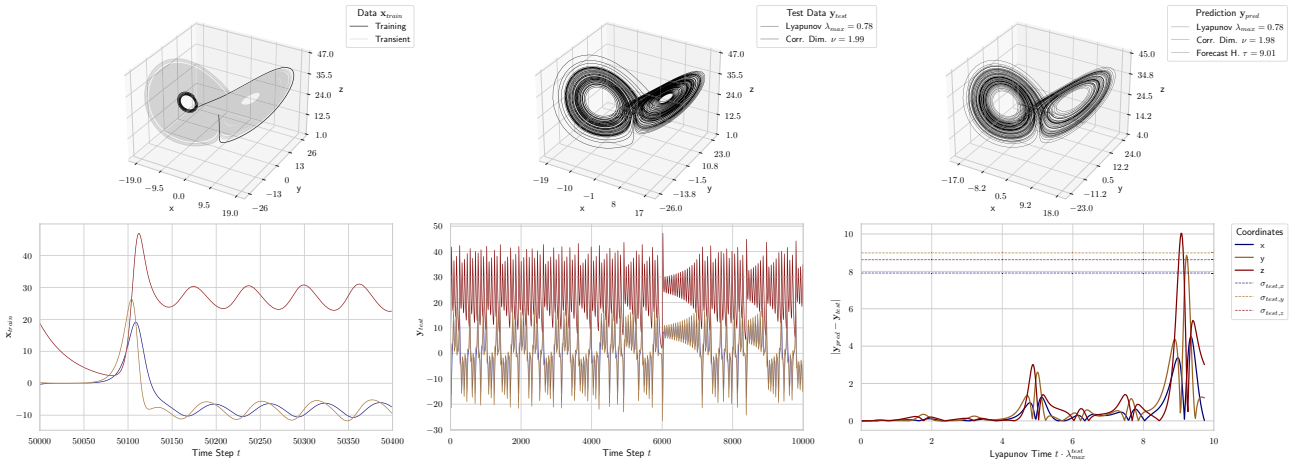

Figure 1. Prediction on a minimal training data set of the Lorenz system. The first column shows the attractor (top) and the trajectories (bottom) of the 400 training data points (and the discarded transient). The second column shows the attractor and the trajectories of the test data. The third column shows the attractor (top) and the absolute prediction error (bottom) of the prediction. The dashed lines indicate the standard deviations of the three components of the test data.

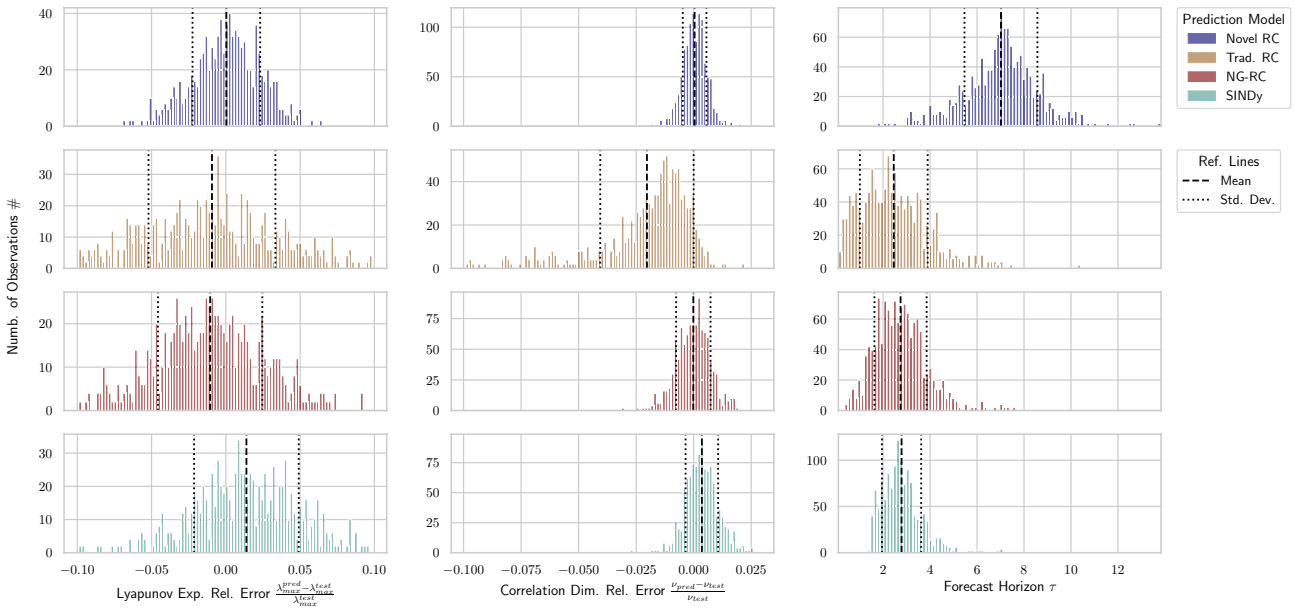

Figure 2. Prediction measures (columns) of different models (rows) for 1000 different starting points on the Lorenz attractor (dt=0.025, T_{train}=400). For the correlation dimension and the Lyapunov exponent we calculate the relative error to the respective test data. The mean and standard deviation of each distribution is denoted by a dashed and dotted black line, respectively.

significantly outperforms them with regards to short-term predictions with an average forecast horizon of ∼7.0 Lyapunov times — this is ∼2.5 times more than the averages of the other models. The long-term prediction is also slightly better as the average relative errors of the correlation dimension and the Lyapunov exponent are $\sim 3.5 \cdot 10^{-4}$, respectively — this is ∼9.0 and ∼39.7 times smaller than the averages of other models. The traditional RC has generally more widely distributed errors due to its randomness.

We verify the robustness of our novel RC to variations in discretization and length of training data. In Fig. 3 we observe that it is quite robust and as expected, performs significantly better than comparable models especially with regards to short-term prediction. Here, we only see a decline in prediction performance for coarse discretizations $dt > 0.045$. The robustness of the long-term prediction is similar to traditional RC and SINDy. Interestingly, we see a decline in performance of NG-RC for larger training lengths $T_{train} > 700$ and finer discretizations $dt < 0.02$. Furthermore, we find out model to be reasonably robust to changes in hyperparameters and noise up to a signal-to-noise ratio of ∼38dB.

Furthermore, we analyze the prediction performance of our model on different chaotic systems, which have different nonlinear behavior. We choose the models so that we can understand the inner workings of our RC better. For example, the Halvorsen system has only quadratic nonlinearities with no interacting coordinates and hence the input matrix only needs the first three blocks (which represent the distinct coordinates). Another example to point out is the Rabinovich-Fabrikant system, which has cubic nonlinearities. Here, we see that a nonlinearity degree of $\eta \geq 3$ is necessary for a reasonable prediction. The model parameters and the prediction measures for the different systems are illustrated in Table 1.

Discussion

In this work, we present a novel RC architecture that outperforms comparable methods in terms of short- and long-term predictions while requiring similarly minimal training datasets and computational power. The architecture is modified by restructuring the input weights and reservoir such that combinations of input data coordinates are fed separately into the reservoir. Therefore, we use a block-diagonal matrix of ones as the reservoir, which acts as an averaging operator for the reservoir states at each update step. Similar to average pooling layers in other machine learning methods, this can be interpreted as a way to primarily "extract" features that are more robust[19]. It also takes out the randomness of traditional RC. Instead of using a nonlinear activation function to create the reservoir states, we capture the nonlinearity of the data in the readout layer by appending higher orders of the reservoir states before the Ridge regression. We find that these changes lead to a significant improvement in the short- and long-term predictions of chaotic systems in comparison to models such as the traditional RC, NG-RC, and SINDy. In order to evaluate the prediction performance, we compute the largest Lyapunov exponent, the correlation dimension, and the prediction horizon.

This work can be extended in many directions. For example, the generation of the reservoir states can be explored to understand what the RC actually learns. In our modified architecture, the states are constructed

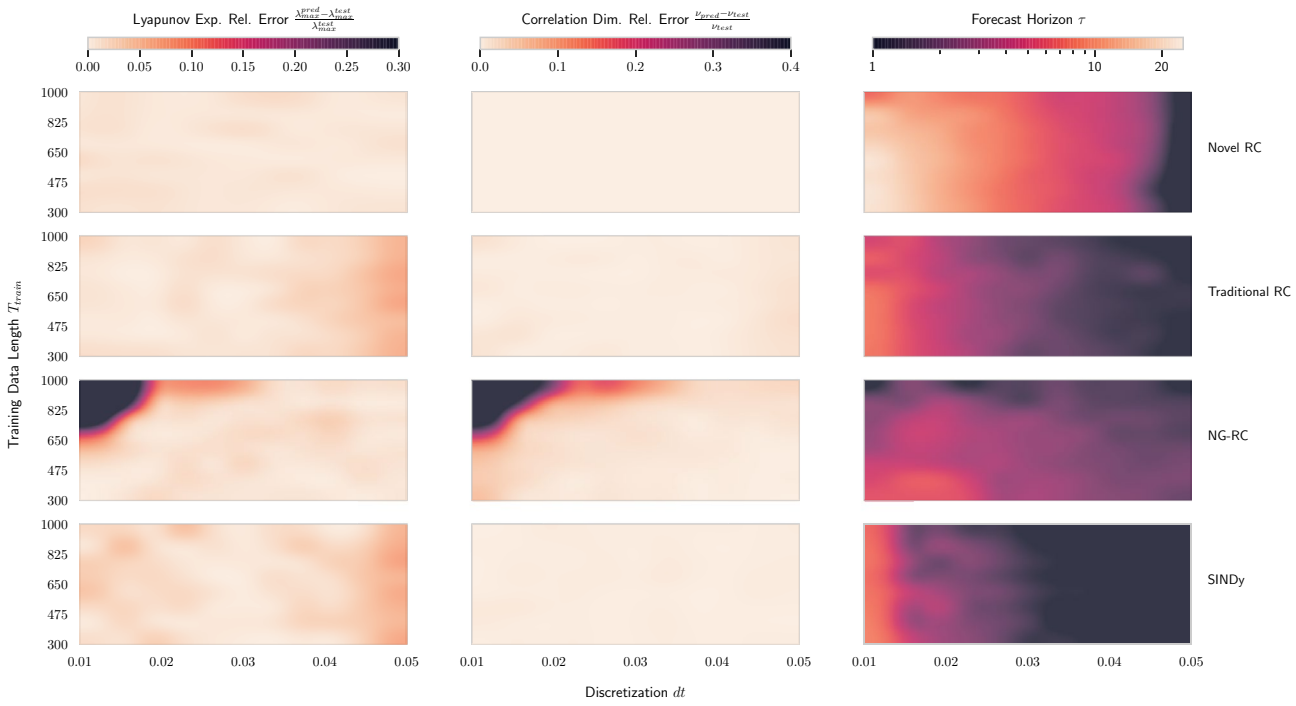

Figure 3. Prediction measures (columns) of different models (rows) for different discretizations and lengths of training data of the Lorenz system. We vary the discretization (x-axis) and the length (y-axis) of the training data between (0.01, 0.05) and (300, 1000), respectively. For the correlation dimension and the Lyapunov exponent we calculate the relative error to the respective test data. Note that the forecast horizon has a logarithmic color scale. Each value in the heatmaps is the average over 100 variations of attractor starting points.

	Training Data		Novel Architecture			Forecast Horizon τ			
System	T_{train}	dt	b	ρ^*	η	Novel	Trad.	NG-RC	SINDy
Halvorsen	300	0.01	3	0.1	2	498±34	231±47	249±27	335±37
Rabi.-Fabr.	300	0.01	3	0.1	3	261±23	168±36	89±12	107±11
Aizawa ([43])	300	0.01	3	0.1	4	193±16	131±27	76±9	65±7
Dadras-Momeni ([44])	300	0.01	3	0.1	2	423±25	228±41	259±19	248±21
Rössler ([45])	300	0.01	3	0.1	2	781±51	301±72	332±40	401±55
Four wing ([46])	300	0.01	3	0.1	2	1497±39	1135±68	659±28	712±31
Chen ([47])	300	0.01	3	0.1	2	922±41	880±72	750±36	812±41

Table 1. Minimal setup for different chaotic systems. We vary the parameters of the training data and the RC architecture to find the minimal setup for different chaotic systems. To do this, we compute the relative errors of the Lyapunov exponent and the correlation dimension for 100 different attractor starting points. The minimum setup is defined as the setup where the average relative errors of the Lyapunov exponent and the correlation dimension are both $<10^{-2}$. This ensures that the long-term climate of the chaotic system is reliably reproduced. In this table, we denote the parameters of the data setup (columns 1−2) and RC architecture (columns 3−5). The last 3 columns denote the mean and standard deviations of the forecast horizon for the different prediction models. The governing equations can be found in the respective references.

by mixing the average of the past data with the new data with different "proportions". Therefore, methods for constructing the reservoir states, such as the exponentially weighted moving average (EWMA) of the data[20], should be explored. Related to this, the design of the readout is also an interesting topic to look into. Similarly to NG-RC and SINDy, nonlinear functions could be applied and appended to the reservoir states in order to capture more complex structures in the data.

Another study can be conducted on how the elimination of randomness from RC-like models affects their capabilities, e.g., information processing capacity[21] or multifunctionality[22].

Furthermore, the applicability to high-dimensional and highly nonlinear data can be analyzed and compared with models relying on large feature libraries, such as NG-RC and SINDy. Since the number of variables scales less rapidly in our architecture, it would be relevant to see how much computational power can be saved, especially for hardware RCs.

Moreover, the model can be tested on real-world examples from different disciplines to produce reliable short- and long-term predictions, especially in cases where training data is scarce and expensive.

Methods

Reservoir computers. A reservoir computer (RC)[5,23,24] is an artificial recurrent neural network (RNN) that relies on a static network called *reservoir*. The term static means that, unlike other RNN approaches, the reservoir remains fixed once the network is constructed. The same is true for the input weights. Therefore, the RC is computationally very efficient since the training process only involves optimizing the output layer. As a result, fast training and high model dimensionality are computationally feasible, making RC well suited for complex real-world applications.

In the following we describe the individual components of the architecture and the modifications that we propose. To make the following section more understandable we introduce them in a high-level summary:

1. *Input weights*: the input weights $\mathbf{W}_{in}$ are designed so that each combination of the coordinates of the data is fed into the reservoir separately.
2. *Reservoir*: the reservoir $\mathbf{A}$ is chosen as a block-diagonal matrix consisting of matrices of ones with size b.
3. *Reservoir states*: we do not use a nonlinear activation function in order to construct the reservoir states $\mathbf{r}(t)$. Hence the iterative update equation reduces to:

$$\mathbf{r}(t+1) = \mathbf{A} \cdot \mathbf{r}(t) + \mathbf{W}_{in} \cdot \mathbf{u}(t)\,, \tag{1}$$

   where $\mathbf{u}(t)$ denotes the training data at time t.
4. *Readout*: instead of only inserting the squared reservoir states[25], our generalized states $\tilde{\mathbf{r}}$ contain all orders up to a nonlinearity degree η:

$$\tilde{\mathbf{r}} = \left\{\mathbf{r}, \mathbf{r}^2, \ldots, \mathbf{r}^{\eta-1}, \mathbf{r}^{\eta}\right\}. \tag{2}$$

5. *Training and Prediction*: as in traditional RCs, we stack the training data $\mathbf{u}$ and the corresponding reservoir states $\tilde{\mathbf{r}}$ to matrices $\mathbf{U}$ and $\tilde{\mathbf{R}}$ respectively. We then solve the equation $\mathbf{W}_{out} \cdot \tilde{\mathbf{R}} = \mathbf{U}$ by using Ridge regression[26] resulting in:

$$\mathbf{W}_{out} = \mathbf{U} \cdot \tilde{\mathbf{R}}^T \left(\tilde{\mathbf{R}} \cdot \tilde{\mathbf{R}}^T + \beta\, \mathbf{I}\right)^{-1}, \tag{3}$$

where $\beta=10^{-5}$ is the regularization constant that prevents overfitting and $\mathbf{I}$ denotes the identity matrix. The prediction procedure of the reservoir states also stays the same (with the adjusted update equation):

$$\mathbf{r}(t+1) = \mathbf{A}\cdot\mathbf{r}(t) + \mathbf{W}_{in}\cdot\mathbf{W}_{out}\cdot\tilde{\mathbf{r}}(t)\,. \tag{4}$$

Note that the reservoir $\mathbf{A}$ only acts on the "simple" reservoir state $\mathbf{r}$, while the second summand acting on $\mathbf{W}_{out}$ is $\tilde{\mathbf{r}}$ containing all the nonlinear powers. The predicted time series $\mathbf{y}(t)$ can then be obtained by the multiplication:

$$\mathbf{y}(t) = \mathbf{W}_{out}\cdot\tilde{\mathbf{r}}(t)\,. \tag{5}$$

Input weights. In order to feed the input data $\mathbf{u}(t)$ into the reservoir, an input weights matrix $\mathbf{W}_{in}$ is defined, which determines how strongly each coordinate influences every single node of the reservoir network. In a traditional RC, the elements of $\mathbf{W}_{in}$ are chosen to be uniformly distributed random numbers within the interval $[-1,1]$.

In our novel framework we do not choose the elements randomly, but follow a structured approach. Firstly, in order to remove the randomness, for a block-size of b we take b equally spaced values between $[1, 0]$.

$$\mathbf{w} = (w_1, w_2, \ldots, w_b)^T = \left(1, \frac{b-2}{b-1}, \ldots, \frac{1}{b-1}, 0\right)^T \tag{6}$$

To avoid non-invertible matrices for ridge regression, we take the square root values of all weights $\mathbf{w} = \left(\sqrt{w_1}, \ldots, \sqrt{w_b}\right)^T$. Then we specifically structure the input matrix so that the different combinations of input data coordinates, also called *features*, are fed separately into the reservoir. In the case of a 3-dimensional system with coordinates $\mathbf{u}(t) = (x, y, z)^T(t)$ and a nonlinearity order η=2, the input matrix (multiplication) looks like:

$$\mathbf{W}_{in}\cdot\mathbf{u}(t) = \begin{pmatrix} \mathbf{w} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{w} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{w} \\ \mathbf{w} & \mathbf{w} & \mathbf{0} \\ \mathbf{w} & \mathbf{0} & \mathbf{w} \\ \mathbf{0} & \mathbf{w} & \mathbf{w} \end{pmatrix} \cdot \mathbf{u}(t) = \begin{pmatrix} x \\ y \\ z \\ x+y \\ x+z \\ y+z \end{pmatrix}(t) \otimes \mathbf{w} \tag{7}$$

where $\otimes$ denotes the tensor product and hence each block represents one feature f. For n-dimensional data the feature space contains $n_f = 2^n - 2$ elements. Thus, the dimension of the reservoir is $d = n_f \cdot b$.

Reservoir. The core of an RC, the reservoir $\mathbf{A}$, is usually constructed as a sparse Erdős-Rényi random network[27] with number of nodes d. As for the choice of input weights, we choose the reservoir in such a way that each feature remains separate. Therefore, we use a block diagonal matrix J consisting of ones $\mathbf{J}$ with block size b. Thus, each block $\mathbf{J}_i$ can be directly mapped to a particular feature:

$$\mathbf{J} = \begin{pmatrix} \mathbf{J}_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{J}_2 & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{J}_{n_f} \end{pmatrix} \longmapsto \begin{pmatrix} \mathbf{J}_x & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{J}_y & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{J}_{y+z} \end{pmatrix}. \tag{8}$$

Similar to a traditional RC, we scale the spectral radius $\rho(\mathbf{J})$ of the reservoir to a target spectral radius ρ^*. While the computation of the spectral radius is usually a computationally expensive task[28] that scales with $\mathcal{O}(d^3)$, the computation is no longer necessary for block diagonal matrices of ones $\mathbf{J}$. This is because the eigenvalues of the matrix are equal to the block size b. Thus, the rescaled reservoir is given by:

$$\mathbf{A} \equiv \frac{\rho^*}{\rho(\mathbf{J})}\mathbf{J} \longrightarrow \frac{\rho^*}{b}\mathbf{J}\,. \tag{9}$$

Our default target spectral radius is ρ^*=0.1.

Reservoir states. As in traditional RC, we use a recurrent update equation to capture the dynamics of the system in the so-called reservoir states $\mathbf{r}(t)$. This would normally require a bounded nonlinear activation function $g(\cdot)$ that captures the nonlinear properties of the data. The activation function (usually the hyperbolic tangent) is applied on an element-by-element basis.

However, as mentioned earlier, we shift the nonlinearity entirely to the readout. Therefore, the time evolution of the states is iteratively determined via:

$$\mathbf{r}(t+1) = g\big(\mathbf{A}\cdot\mathbf{r}(t) + \mathbf{W}_{in}\cdot\mathbf{u}(t)\big) \tag{10}$$

$$\longrightarrow \mathbf{A}\cdot\mathbf{r}(t) + \mathbf{W}_{in}\cdot\mathbf{u}(t)\,. \tag{11}$$

Due to our choice of architecture, the reservoir states for each feature can be obtained separately:

$$\mathbf{r}(t) = \begin{pmatrix} \mathbf{r}_x \\ \mathbf{r}_y \\ \mathbf{r}_z \\ \mathbf{r}_{x+y} \\ \mathbf{r}_{x+z} \\ \mathbf{r}_{y+z} \end{pmatrix}(t) = \begin{pmatrix} \mathbf{r}_{x,1} \\ \mathbf{r}_{x,2} \\ \vdots \\ \mathbf{r}_{x,b} \\ \mathbf{r}_{y,1} \\ \vdots \\ \mathbf{r}_{y+z,b} \end{pmatrix}(t)\,. \tag{12}$$

Hence, we can take all reservoir states belonging to one feature $\mathbf{r}_f(t)$ — we call them *feature states* — and analyze them separately. This helps us to understand that the reservoir acts as an averaging operator on the feature states:

$$\mathbf{A}_f \cdot \mathbf{r}_f(t) = \left(\frac{\rho^*}{b}\sum_{i=1}^{b}\mathbf{r}_{f,i}(t)\right)\cdot \mathbf{I}_b = \rho^* \cdot \bar{\mathbf{r}}_f(t)\,, \tag{13}$$

where $\mathbf{I}_b$ is a vector of ones of size b. Thus, in each iteration step, the feature states are "normalized" to the average of the past feature states $\bar{\mathbf{r}}_f(t)$ and a varying strength (determined by the input weight) is added to the new feature data $f(t)$:

$$\mathbf{r}_f(t+1) = \rho^* \cdot \bar{\mathbf{r}}_f(t) + \mathbf{w}\cdot f(t)\,. \tag{14}$$

where f can be replaced by any other feature without loss of validity. The average, or "memory", of each feature is tracked in the last row of the feature states since w_b=0. Furthermore, this implies that target spectral radius ρ^* determines how strongly the memory of the data is kept in each iteration step.

Readout. While a quadratic readout, i.e., the squared reservoir states $\mathbf{r}^2$, is often added to a traditional RC to break the symmetry of the activation function[25], we need the readout to capture the nonlinearity of the data. Therefore, we add even higher orders of nonlinearity to the so-called generalized states $\tilde{\mathbf{r}}$. For a given degree of nonlinearity η they look like the following:

$$\tilde{\mathbf{r}} = \left\{\mathbf{r}, \mathbf{r}^2, \ldots, \mathbf{r}^{\eta-1}, \mathbf{r}^{\eta}\right\}. \tag{15}$$

Hence, for a degree of nonlinearity η and a block-size of b, the number of elements in the readout, which is also the number of variables to be optimized, is:

$$n_{out} = \left(2^n - 2\right)\cdot\eta\cdot b = \left(\sum_{k=1}^{n}\binom{n}{k} - 1\right)\cdot\eta\cdot b\,, \tag{16}$$

which we rewrite to binomial coefficients for better comparison. For high-dimensional data with high nonlinearity, the number of variables to be optimized is much smaller than for comparable predictive models such as NG-RC[14] or SINDy[15]. This is because NG-RC and SINDy require combinations with recurrences. Therefore, the size of their feature space for a nonlinearity degree is η (at least):

$$n_f = \sum_{k=1}^{\eta}\binom{n+k+1}{k} = \binom{n+\eta}{n} - 1\,, \tag{17}$$

which grows much faster for larger n and η than the expression for n_{out} in Eq. 16.

Prediction performance measures. When forecasting nonlinear dynamical systems, the goal is not only to exactly replicate the short-time trajectory, but also to reproduce the long-term statistical properties, or climate, of the system.

Correlation dimension. To assess the structural complexity of an attractor, we calculate its correlation dimension ν, which measures the fractal dimensionality of the space populated by its trajectory[29,30]. The correlation dimension is implicitly defined by the power-law relationship based on the correlation integral:

$$C(r) = \int_0^r d^n r' c(\mathbf{r}') \longrightarrow C(r) \propto r^{\nu} \tag{18}$$

where n is the dimension of the data and $c(\mathbf{r}')$ is the standard correlation function. The integral represents the mean probability that two states in the phase space are close to each other at different time steps. This is the case if the distance between the two states is smaller than the threshold distance r. For self-similar, strange attractors, this power-law relationship holds for a certain range of r, which can be calibrated using the *Grassberger-Procaccia* algorithm[31]. The benefits of this measure are that it is purely data-based, it only needs a small number of data points, and it does not require any knowledge of the underlying governing equations of the system.

Lyapunov exponents. Besides its fractal dimensionality, the statistical climate of an attractor is also characterized by its temporal complexity represented by the *Lyapunov* exponents[32]. They describe the average rate of

divergence of nearby points in the phase space, and thus measure sensitivity with respect to initial conditions[33]. There is one exponent for each dimension in the phase space. If the system has at least one positive Lyapunov exponent, it is classified as chaotic. Thus, it is sufficient for the purposes in this work to calculate only the largest Lyapunov exponent λ_{max}:

$$d(t) = C \cdot e^{\lambda_{max} \cdot t} \tag{19}$$

where $d(t)$ denotes the distance of two initially nearby states in phase space and the constant C is the normalization constant at the initial separation. Thus, instead of determining the full Lyapunov spectrum, we only need to find the largest one as it describes the overall system behavior to a large extent. Therefore we use the *Rosenstein* algorithm[34].

Forecast horizon. To quantify the quality and duration of the short-term prediction of the trajectory we use the forecast horizon τ[12]. It tracks the number of time steps for which the absolute error between each coordinate of the predicted $\mathbf{y}_{pred}(t)$ and test $\mathbf{y}_{test}(t)$ data does not exceed the standard deviation of the test data $\boldsymbol{\sigma}\left(\mathbf{y}_{test}(t)\right)$:

$$\left|\mathbf{y}_{pred}(t) - \mathbf{y}_{test}(t)\right| < \boldsymbol{\sigma}\left(\mathbf{y}_{test}(t)\right). \tag{20}$$

We express the forecast horizon in units of the Lyapunov time by multiplying it with the discretization and maximum Lyapunov exponent of the test data $\tau \cdot dt \cdot \lambda_{max}^{test}$. This measure is intended to evaluate how long a prediction can reproduce the actual trajectory before the chaotic nature of the system leads to an exponential divergence.

Dynamical systems. We apply our model to a number of synthetic chaotic systems. In our analyses, we focus on the following three due to their specific properties in terms of nonlinearity.

Lorenz. As it is common in RC research[35,36] we use the Lorenz system which was initially proposed for modeling atmospheric convection[37]:

$$\begin{aligned} \frac{dx}{dt} &= \sigma \cdot (y - x) \\ \frac{dy}{dt} &= x \cdot (\rho - z) - y \\ \frac{dz}{dt} &= x \cdot y - \beta \cdot z\,, \end{aligned} \tag{21}$$

where the standard parametrization for chaotic behavior is σ=10, ρ=28, and β=8/3.

Halvorsen. As in Hertreux and Räth[25] we use the Halvorsen system[38] for our analyses, which has a cyclic symmetry and, unlike to the Lorenz system, only has nonlinearities without interaction of coordinates:

$$\begin{aligned} \frac{dx}{dt} &= a \cdot x - b \cdot y - b \cdot z - y^2 \\ \frac{dy}{dt} &= a \cdot y - b \cdot z - b \cdot x - z^2 \\ \frac{dz}{dt} &= a \cdot z - b \cdot x - b \cdot y - x^2\,, \end{aligned} \tag{22}$$

where a=1.3 and b=4 are the standard parameters.

Rabinovich–Fabrikant. In order to test whether our model works also for systems entailing cubic nonlinearities, we analyze the Rabinovich-Fabrikant system[39]:

$$\begin{aligned} \frac{dx}{dt} &= y \cdot (z - 1 + x^2) + \gamma \cdot x \\ \frac{dy}{dt} &= x \cdot (3 \cdot z + 1 - x^2) + \gamma \cdot y \\ \frac{dz}{dt} &= -2 \cdot z \cdot (\alpha + x \cdot y)\,, \end{aligned} \tag{23}$$

where α=0.14 and γ=0.1 are the standard parameters.

Simulating and splitting data. Since we compare our model with NG-RC and SINDy, we use the same data setup as the original works[14,15]. Hence, we solve the differential equations of the systems using the *Runge-Kutta* method[40] with a discretization of dt=0.025 in order to ensure a sufficient manifestation of the attractor.

We discard the initial transient of $T_{transient}$=50000, use the next T_{train}=400 steps for training, then skip T_{skip}=10000 steps, and use the remaining T_{test}=10000 for testing the prediction. Hence in total we simulate T=70400 steps.

To get robust results, we also vary the starting points on the attractor by using the rounded last point of one data sample as the starting point for the next. The initial starting points for the Lorenz, Halvorsen, and Rabinovich-Fabrikant systems are $(-14, -20, 25)$, $(-6.4, 0, 0)$, and $(-0.4, 0.1, 0.7)$, respectively.

Comparable prediction models. We compare our novel RC to other models designed for predicting dynamical systems. We briefly describe them in the following.

Traditional reservoir computer. For the traditional RC architecture we choose an Erdős-Rényi network of dimension d=600 with a target spectral radius ρ^* and a quadratic readout. This equals 1200 variables per coordinate to be optimized. In order to get robust results we repeat the prediction for 1000 realizations and take the average of the prediction measures.

Next generation reservoir computer. The next generation reservoir computer (NG-RC) developed by Gauthier et al.[14] is a so-called nonlinear vector autoregression (NVAR) machine and thus, does not require a reservoir. It solely needs the feature vector, which consists of time-delay observations of the data and nonlinear functions of these observations. The resulting output weights can be used to construct the governing equations of the data. We use the standard setting with time delays k=2 and skips s=1.

Sparse identification of nonlinear dynamics. Sparse identification of nonlinear dynamics (SINDy)[15] discovers the underlying dynamical system of data by learning its governing equations through sparse regression. It is similar to NG-RC, but uses an iterative approach to filter only relevant features. We use the standard parametrization and the official Python package PySINDy[41,42].

Data availability

The datasets used and/or analyzed during the current study available from the corresponding author on reasonable request.

Received: 1 May 2023; Accepted: 1 August 2023
Published online: 10 August 2023

References

1. S. L. Brunton and J. N. Kutz, *Data-driven Science and Engineering: Machine Learning, Dynamical Systems, and Control* (Cambridge University Press, 2022)
2. Creswell, A. *et al.* Generative adversarial networks: An overview. *IEEE Signal Process.Mag.* **35**, 53 (2018).
3. Zhang, J., Wang, Y., Molino, P., Li, L. & Ebert, D. S. Manifold: A model-agnostic framework for interpretation and diagnosis of machine learning models. *IEEE Trans. Vis. Comput. Graph.* **25**, 364 (2018).
4. Roscher, R., Bohn, B., Duarte, M. F. & Garcke, J. Explainable machine learning for scientific insights and discoveries. *IEEE Access* **8**, 42200 (2020).
5. Jaeger, H. The "echo state" approach to analysing and training recurrent neural networks-with an erratum note. *Bonn Ger. Ger. Natl. Res. Center Inf. Technol. GMD Tech. Rep.* **148**, 13 (2001).
6. D. Prokhorov, Echo state networks: appeal and challenges, in *Proc. 2005 IEEE International Joint Conf. on Neural Networks, 2005.*, Vol. 3 (IEEE, 2005) pp. 1463–1466
7. Broido, A. D. & Clauset, A. Scale-free networks are rare. *Nat. Commun.* **10**, 1017 (2019).
8. Gerlach, M. & Altmann, E. G. Testing statistical laws in complex systems. *Phys. Rev. Lett.* **122**, 168301 (2019).
9. G. Holzmann, Reservoir computing: a powerful black-box framework for nonlinear audio processing, in *International Conf. on Digital Audio Effects (DAFx)* (Citeseer, 2009)
10. J. Platt, H. Abarbanel, S. Penny, A. Wong, and R. Clark, Robust forecasting through generalized synchronization in reservoir computing, in *AGU Fall Meeting Abstracts*, Vol. 2021 (2021) pp. NG25B–0522
11. Tanaka, G. *et al.* Recent advances in physical reservoir computing: A review. *Neural Netw.* **115**, 100 (2019).
12. Haluszczynski, A. & Räth, C. Good and bad predictions: Assessing and improving the replication of chaotic attractors by means of reservoir computing. *Chaos Interdisc. J. Nonlinear Sci.* **29**, 103143 (2019).
13. Bollt, E. On explaining the surprising success of reservoir computing forecaster of chaos? the universal machine learning dynamical system with contrast to var and dmd a3b2 show [feature]. *Chaos Interdisc. J. Nonlinear Sci.* **31**, 013108 (2021).
14. Gauthier, D. J., Bollt, E., Griffith, A. & Barbosa, W. A. Next generation reservoir computing. *Nat. Commun.* **12**, 5564 (2021).
15. Brunton, S. L., Proctor, J. L. & Kutz, J. N. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proc. Natl. Acad. Sci.* **113**, 3932 (2016).
16. Weise, C. L. The asymmetric effects of monetary policy: A nonlinear vector autoregression approach. *J. Money Credit Bank.* **85**, 25468 (1999).
17. Gonon, L. & Ortega, J.-P. Reservoir computing universality with stochastic inputs. *IEEE Trans. Neural Netw. Learn. Syst.* **31**, 100 (2019).
18. Hart, A. G., Hook, J. L. & Dawes, J. H. Echo state networks trained by tikhonov least squares are l2 (μ) approximators of ergodic dynamical systems. *Phys. D Nonlinear Phenom.* **421**, 132882 (2021).
19. D. Yu, H. Wang, P. Chen, and Z. Wei, Mixed pooling for convolutional neural networks, in *Rough Sets and Knowledge Technology: 9th International Conf., RSKT 2014, Shanghai, China, October 24-26, 2014, Proc. 9* (Springer, 2014) pp. 364–375
20. Hunter, J. S. The exponentially weighted moving average. *J. Qual. Technol.* **18**, 203 (1986).
21. Dambre, J., Verstraeten, D., Schrauwen, B. & Massar, S. Information processing capacity of dynamical systems. *Sci. Rep.* **2**, 1 (2012).
22. Flynn, A., Tsachouridis, V. A. & Amann, A. Multifunctionality in a reservoir computer. *Chaos Interdisc. J. Nonlinear Sci.* **31**, 013125 (2021).
23. Maass, W., Natschläger, T. & Markram, H. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Comput.* **14**, 2531 (2002).
24. Jaeger, H. & Haas, H. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *Science* **304**, 78 (2004).
25. J. Herteux and C. Räth, Breaking symmetries of the reservoir equations in echo state networks. Chaos Interdisc. J. Nonlinear Sci. **30**, 123142 (2020)

26. Hoerl, A. E. & Kennard, R. W. Ridge regression: Applications to nonorthogonal problems. *Technometrics* **12**, 69 (1970).
27. Erdos, P. *et al.* On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci* **5**, 17 (1960).
28. Paige, C. C. Bidiagonalization of matrices and solution of linear equations. *SIAM J. Numer. Anal.* **11**, 197 (1974).
29. P. Grassberger and I. Procaccia, Measuring the strangeness of strange attractors, in *The Theory of Chaotic Attractors* (Springer, 2004) pp. 170–189
30. Mandelbrot, B. How long is the coast of britain? Statistical self-similarity and fractional dimension. *Science* **156**, 636 (1967).
31. Grassberger, P. Generalized dimensions of strange attractors. *Phys. Lett. A* **97**, 227 (1983).
32. Wolf, A., Swift, J. B., Swinney, H. L. & Vastano, J. A. Determining lyapunov exponents from a time series. *Phys. D Nonlinear Phenom.* **16**, 285 (1985).
33. Shaw, R. Strange attractors, chaotic behavior, and information flow. *Z. Naturforschung A* **36**, 80 (1981).
34. Rosenstein, M. T., Collins, J. J. & De Luca, C. J. A practical method for calculating largest lyapunov exponents from small data sets. *Phys. D Nonlinear Phenom.* **65**, 117 (1993).
35. Pathak, J., Lu, Z., Hunt, B. R., Girvan, M. & Ott, E. Using machine learning to replicate chaotic attractors and calculate lyapunov exponents from data. *Chaos Interdisc. J. Nonlinear Sci.* **27**, 121102 (2017).
36. Lu, Z., Hunt, B. R. & Ott, E. Attractor reconstruction by machine learning. *Chaos Interdisc. J. Nonlinear Sci.* **28**, 061104 (2018).
37. Lorenz, E. N. Deterministic nonperiodic flow. *J. Atmos. Sci.* **20**, 130 (1963).
38. S. Vaidyanathan and A. T. Azar, Adaptive control and synchronization of halvorsen circulant chaotic systems, In *Advances in chaos theory and intelligent control* (Springer, 2016) pp. 225–247
39. Rabinovich, M. I., Fabrikant, A. L. & Tsimring, L. S. Finite-dimensional spatial disorder. *Soviet Phys. Usp.* **35**, 629 (1992).
40. E. Hairer, S. P. Nørsett, and G. Wanner, *Solving Ordinary Differential Equations I* (Springer, 1993)https://doi.org/10.1007/978-3-540-78862-1
41. B. de Silva, K. Champion, M. Quade, J.-C. Loiseau, J. Kutz, and S. Brunton, Pysindy: A python package for the sparse identification of nonlinear dynamical systems from data. J. Open Source Softw. **5**, 2104 (2020)https://doi.org/10.21105/joss.02104
42. A. A. Kaptanoglu, B. M. de Silva, U. Fasel, K. Kaheman, A. J. Goldschmidt, J. Callaham, C. B. Delahunt, Z. G. Nicolaou, K. Champion, J.-C. Loiseau, J. N. Kutz, and S. L. Brunton, Pysindy: A comprehensive python package for robust sparse system identification. J. Open Source Softw. 7, 3994 (2022) https://doi.org/10.21105/joss.03994
43. Aizawa, Y. *et al.* Stagnant motions in hamiltonian systems. *Progr. Theor. Phys. Suppl.* **98**, 36 (1989).
44. Dadras, S. & Momeni, H. R. A novel three-dimensional autonomous chaotic system generating two, three and four-scroll attractors. *Phys. Lett. A* **373**, 3637 (2009).
45. Rossler, O. An equation for hyperchaos. *Phys. Lett. A* **71**, 155 (1979).
46. Qi, G., Chen, G., van Wyk, M. A., van Wyk, B. J. & Zhang, Y. A four-wing chaotic attractor generated from a new 3-d quadratic autonomous system. *Chaos Solitons Fractals* **38**, 705 (2008).
47. Chen, G. & Ueta, T. Yet another chaotic attractor. *Int. J. Bifurc. Chaos* **9**, 1465 (1999).

Author contributions

H.M conducted the research and wrote the main manuscript text. C.R. and D.P. assisted in the research. All authors reviewed the manuscript.

Funding

Open Access funding enabled and organized by Projekt DEAL.

Competing interests

The authors declare no competing interests.

Additional information

Correspondence and requests for materials should be addressed to C.R.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.