Chaos
An Interdisciplinary Journal of Nonlinear Science

RESEARCH ARTICLE | JUNE 12 2023

Efficient forecasting of chaotic systems with block-diagonal and binary reservoir computing

Haochun Ma; Davide Prosperino; Alexander Haluszczynski; Christoph Räth

Check for updates

Chaos 33, 063130 (2023)
https://doi.org/10.1063/5.0151290

View Online
Export Citation
CrossMark

Downloaded from http://pubs.aip.org/aip/cha/article-pdf/doi/10.1063/5.0151290/17996145/063130_1_5.0151290.pdf

AIP Publishing

Efficient forecasting of chaotic systems with block-diagonal and binary reservoir computing

Cite as: Chaos **33**, 063130 (2023); doi: 10.1063/5.0151290
Submitted: 20 March 2023 · Accepted: 12 May 2023 ·
Published Online: 12 June 2023

View Online Export Citation CrossMark

Haochun Ma,[1,2] Davide Prosperino,[1,2] Alexander Haluszczynski,[2] and Christoph Räth[3,a)]

AFFILIATIONS

[1]Department of Physics, Ludwig-Maximilians-Universität, Schellingstraße 4, 80799 Munich, Germany
[2]Allianz Global Investors, risklab, Seidlstraße 24, 80335 Munich, Germany
[3]Deutsches Zentrum für Luft- und Raumfahrt (DLR), Institut für KI Sicherheit, Wilhelm-Runge-Straße 10, 89081 Ulm, Germany

[a)]**Author to whom correspondence should be addressed:** christoph.raeth@dlr.de

ABSTRACT

The prediction of complex nonlinear dynamical systems with the help of machine learning has become increasingly popular in different areas of science. In particular, reservoir computers, also known as echo-state networks, turned out to be a very powerful approach, especially for the reproduction of nonlinear systems. The reservoir, the key component of this method, is usually constructed as a sparse, random network that serves as a memory for the system. In this work, we introduce block-diagonal reservoirs, which implies that a reservoir can be composed of multiple smaller reservoirs, each with its own dynamics. Furthermore, we take out the randomness of the reservoir by using matrices of ones for the individual blocks. This breaks with the widespread interpretation of the reservoir as a single network. In the example of the Lorenz and Halvorsen systems, we analyze the performance of block-diagonal reservoirs and their sensitivity to hyperparameters. We find that the performance is comparable to sparse random networks and discuss the implications with regard to scalability, explainability, and hardware realizations of reservoir computers.

Published under an exclusive license by AIP Publishing. https://doi.org/10.1063/5.0151290

The application of reservoir computers to various fields in science and technology yields very promising and fast advancing results due to their capabilities in forecasting chaotic attractors, inferring unmeasured values in systems, and recognizing speech. While the construction of a reservoir computer is rather simple in comparison to other machine learning techniques, the architecture and functionality of them is in many regards still a black-box since at its core, the reservoir is usually still chosen as a random network. Thus, we replace the network with block-diagonal matrices dividing it into multiple smaller reservoirs, take out the randomness, and show that these alterations still deliver an equal quality of short- and long-term predictions. This architecture breaks with the common interpretation of the reservoir as a single network and may prove to be more scalable and easier to implement in hardware than their more complex variants while still performing as well.

I. INTRODUCTION

The analysis and modeling 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 excellent predictions on dynamical systems,[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 in order to prevent the risk of misinterpreting the results if deeper methodological knowledge is missing.[4]

In the context of complex system research, reservoir computers (RCs)[5] have emerged for quantifying and predicting the spatiotemporal dynamics of chaotic nonlinear systems. They represent a special kind of recurrent neural networks (RNNs) and are often referred to as echo-state networks (ESNs).[6] The core of the model is a fixed reservoir, which is a complex network with connections according to a predefined network topology. The input data are 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, comparatively transparent, and prevents the vanishing gradient problem of other RNN methods.[7]

The topology of a reservoir, or the arrangement of the nodes and connections within it, can have a significant impact on the

Downloaded from http://pubs.aip.org/aip/cha/article-pdf/doi/10.1063/5.0151290/17996145/063130_1_5.0151290.pdf

Published under an exclusive license by AIP Publishing

performance of a reservoir computing system.[8] In current state-of-the-art models, the topology of the reservoir is often chosen randomly,[9] with the hope that the resulting dynamics will be sufficiently complex to allow for good performance on a given task. However, this approach can be hit-or-miss,[10,11] and it is impossible to know a priori how the topology of the reservoir will affect the performance of the system.

While Maass *et al.*[12] and Jaeger[13] introduced ESNs with reservoirs being modeled as a random Erdös–Renyi network, Watts and Strogatz,[14] Albert and Barabási,[15] and others have shown that random networks are far from being common in physics, biology, or sociology. Instead, more complex networks like scale-free, small-world, or intermediate forms of networks[16,17] are most often found in real-world applications. Further approaches to make reservoir computing more explainable have been made in recent years, with, e.g., Haluszczynski and Räth[11] comparing different network construction algorithms, Griffith *et al.*[18] introducing very low connectivity networks, and Carroll and Pecora[10] analyzing the effect of network symmetries on prediction performance. However, there still remain open questions about the functionality of reservoir computers, which need to be answered for developing new algorithms, for fine-tuning the system for specific applications, or building efficient hardware realizations of RCs.

In this work, we break with the interpretation of the reservoir as a single network by deliberately using block-diagonal matrices as reservoirs. This implies that we decompose the reservoir into multiple smaller reservoirs as outlined in Sec. II B. Furthermore, we use matrices of ones as the blocks, which take out the randomness of the network completely. We assess the ability of block-diagonal reservoirs for short- and long-term predictions by comparing the measures discussed in Sec. II C to the standard RC setup.

II. METHODS

We structure our methods section into three different parts: benchmark models, reservoir computing, and prediction performance measures.

A. Benchmark models

We perform our analyses on two synthetic example models, which exhibit chaotic behavior and are three-dimensional autonomous, dissipative flows.

1. Lorenz system

As in Pathak *et al.*[19] and Lu *et al.*,[20] we use the Lorenz system, which was initially used for modeling atmospheric convection,[21] as an example for replicating chaotic attractors using reservoir computing,

$$\begin{aligned}\frac{dx}{dt} &= \sigma(y-x),\\ \frac{dy}{dt} &= x(\rho-z)-y,\\ \frac{dz}{dt} &= xy-\beta z,\end{aligned} \tag{1}$$

where the standard parameterization for chaotic behavior is $\sigma = 10$, $\rho = 28$, and $\beta = 8/3$.

2. Halvorsen system

As in Herteux and Räth,[22] we use the Halvorsen system for our analyses, which has a cyclic symmetry. While the nonlinear terms of the Lorenz system are mixed products of two different variables, the Halvorsen system[23] entails only non-mixed quadratic nonlinearities,

$$\begin{aligned}\frac{dx}{dt} &= ax-by-bz-y^2,\\ \frac{dy}{dt} &= ay-bz-bx-z^2,\\ \frac{dz}{dt} &= az-bx-by-x^2,\end{aligned} \tag{2}$$

where $a = 1.3$ and $b = 4$ are the standard parameter choice.

3. Simulating and splitting data

If not stated otherwise, we solve the differential equations of the synthetic system using the Runge–Kutta method[24] for $T = 70\,000$ steps and a discretization of $dt = 0.02$ in order to ensure a sufficient manifestation of the attractor. We discard the initial transient of $T = 50\,000$ steps and use the remaining steps for training $T_{train} = 10\,000$ and testing $T_{test} = 10\,000$ of the RCs. In order to get robust results, we 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 and Halvorsen systems are $(-14, -20, 25)$ and $(-6.4, 0, 0)$, respectively. This setting is comparable to the ones used by Griffith *et al.*[18] and Haluszczynski and Räth.[11] Figure 1 illustrates the attractors and trajectories of the simulated data.

B. Reservoir computing

A reservoir computer (RC)[12,13,25,26] is an artificial recurrent neural network (RNN) that relies on a static internal 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 system 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.

1. Algorithm

The reservoir $\mathbf{A}$ is usually constructed as a sparse Erdös-Rényi random network[27] with dimensionality or number of nodes d. However, in this paper, we replace the network structure of the reservoir with a block-diagonal matrix of ones. This breaks with the widespread interpretation of the reservoir as a single network.

To feed the n-dimensional input data $\mathbf{u}(t)$ into reservoir $\mathbf{A}$, a $d \times n$ input matrix $\mathbf{W}_{in}$ is constructed, which defines how strongly each input dimension influences every single node. The elements of $\mathbf{W}_{in}$ are chosen to be uniformly distributed random numbers within the interval $[-1, 1]$.

Downloaded from http://pubs.aip.org/aip/cha/article-pdf/doi/10.1063/5.0151290/17996145/063130_1_5.0151290.pdf

Published under an exclusive license by AIP Publishing

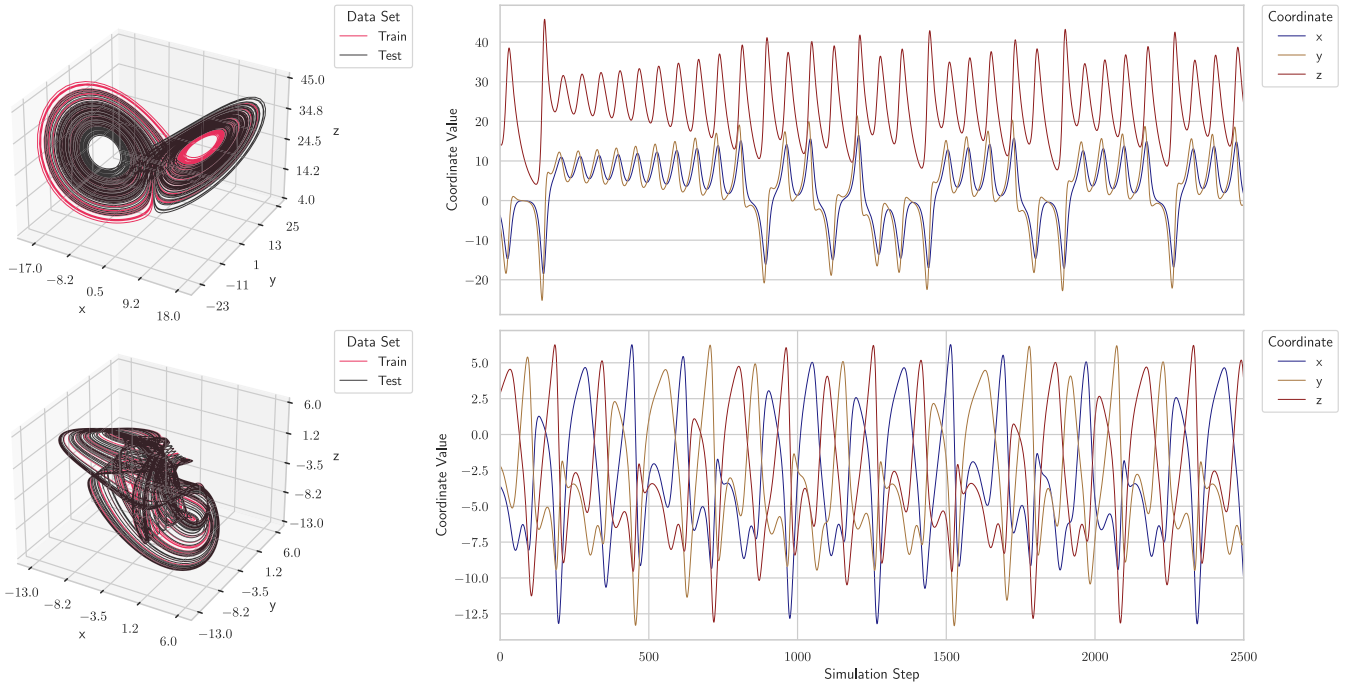

FIG. 1. Attractor (left) and coordinate trajectories (right) of the Lorenz (top) and Halvorsen (bottom) system. The left columns show the attractors of the two systems, where the first $T_{train} = 10\,000$ steps are used for training (red) and the subsequent $T_{test} = 10\,000$ steps are used for testing the prediction (black). The right columns illustrate the coordinate trajectories of both systems for the first $T = 2500$ steps. The parameters of the systems and their simulations are described in Sec. II A.

The dynamics of the RC system are contained in its d-dimensional reservoir states $\mathbf{r}(t)$. Being initially set to $r_i(0) = 0$ for all nodes, the time evolution can be defined using a recurrent formulation,

$$\mathbf{r}(t+1) = f(\mathbf{A} \cdot \mathbf{r}(t) + \mathbf{W}_{in} \cdot \mathbf{u}(t)), \tag{3}$$

where f is a limited, nonlinear function—as is common, we use the hyperbolic tangent. Before the training process is started, the RC system should be initialized during a washout phase of t_w time steps in order synchronize the reservoir states $\mathbf{r}(t)$ with the dynamics of the input signal $\mathbf{u}(t)$. Furthermore, in order to break potential problems arising from the anti-symmetry of the hyperbolic tangent, we use a quadratic readout as explained by Herteux and Räth.[22] This means that the squared elements of the reservoir states are appended $\mathbf{r} \mapsto \{\mathbf{r}, \mathbf{r}^2\}$.

To obtain n-dimensional output from the (matrix of) reservoir states $\mathbf{r}(t)$, an output-mapping function $\mathbf{W}_{out}$ is needed. This is accomplished by acquiring a sufficient number of reservoir states $\mathbf{r}(t_w, \ldots, t_w + t_{T_{train}})$ and then choosing an output-mapping matrix $\mathbf{W}_{out}$ such that the output of the reservoir is as close as possible to the known real data (matrix) $\mathbf{u}(t_w, \ldots, t_w + t_{T_{train}})$. Then, the training can be executed by using Ridge regression,[28]

$$\mathbf{W}_{out} = (\mathbf{r}^T \cdot \mathbf{r} + \beta \cdot \mathbf{I})^{-1} \mathbf{r}^T \cdot \mathbf{u}, \tag{4}$$

where β is the regularization constant that prevents overfitting and $\mathbf{I}$ denotes a identity matrix. The predicted state $\mathbf{v}(t)$ can be obtained by multiplying the output matrix with the reservoir state $\mathbf{r}(t)$,

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

After training, the predicted state $\mathbf{v}(t)$ can be fed back in the activation function as input instead of the actual data $\mathbf{u}(t)$ by combining Eqs. (3) and (5). The resulting recursive form of the equation for the reservoir states $\mathbf{r}(t)$ allows us to create predicted trajectories of arbitrary length,

$$\mathbf{r}(t+1) = f(\mathbf{A} \cdot \mathbf{r}(t) + \mathbf{W}_{in} \cdot \mathbf{W}_{out} \cdot \mathbf{r}(t)). \tag{6}$$

2. Block-diagonal reservoir

The main focus of this work is to verify that a reservoir can be divided into multiple smaller reservoirs without limiting its prediction performance. Therefore, we choose our $d \times d$ dimensional reservoir topology to be a block-diagonal matrix with blocks $\mathbf{J}_i$ of size $b \times b$, where $i \in \{1, 2, \ldots, \lfloor \frac{d}{b} \rfloor\}$. Each of the blocks essentially represents a separate smaller reservoir,

$$\mathbf{J} = \begin{pmatrix} \mathbf{J}_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{J}_2 & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{J}_{\lfloor \frac{d}{b} \rfloor} \end{pmatrix}. \tag{7}$$

As usual, the reservoir topology is rescaled to a target spectral radius ρ^*. Therefore, the spectral radius of the matrix $\mathbf{J}$ needs to

Downloaded from http://pubs.aip.org/aip/cha/article-pdf/doi/10.1063/5.0151290/17996145/063130_1_5.0151290.pdf

Published under an exclusive license by AIP Publishing

be determined first, which is the largest absolute eigenvalue of the matrix,

$$\rho(\mathbf{J}) = \max\{|\lambda_1|, \ldots, |\lambda_d|\}. \tag{8}$$

The time complexity of obtaining the eigenvalues of a d-dimensional matrix using bi-diagonalization is $\mathcal{O}(d^3)$.[29] We can speed up the calculation by a factor of

$$\frac{d^3}{b^3 \cdot \lfloor \frac{d}{b} \rfloor} \approx \left(\frac{d}{b}\right)^2, \tag{9}$$

since the eigenvalues of a bock-diagonal matrix is the list of the eigenvalues of the blocks,

$$\rho(\mathbf{J}) = \max\left\{\rho(\mathbf{J_1}), \ldots, \rho\left(\mathbf{J}_{\lfloor \frac{d}{b} \rfloor}\right)\right\}. \tag{10}$$

Then, the scaled reservoir $\mathbf{A}$, which is finally used in the RC, is given by

$$\mathbf{A} = \frac{\rho^*}{\rho(\mathbf{J})}\mathbf{J}. \tag{11}$$

In Secs. II B 3 and II B 4, we describe the different types of blocks $\mathbf{J}_i$ that we study in this work: first, where the blocks are Erdös–Renyi networks, and second, where the blocks are matrices of ones.

3. Blocks of Erdös–Rényi networks

First, we choose the individual blocks $\mathbf{J}_i$ to be Erdös–Rényi networks with a connection probability of $p = 0.02$.[14] In our analyses, we distinguish between two cases:

1. *Individual blocks*: Each block $\mathbf{J}_i$ is constructed separately with a different random seed.
2. *Equal blocks*: All the blocks are equal to each other and thus, the Erdös–Rényi network only needs to be constructed once,

$$\mathbf{J}_1 = \mathbf{J}_2 = \cdots = \mathbf{J}_{\lfloor \frac{d}{b} \rfloor}.$$

Furthermore, this case delivers another increase in eigenvalue decomposition speed by a factor of $\lfloor \frac{d}{b} \rfloor$.

4. Blocks of matrices of ones

In order to take out the randomness of the reservoir, we construct it so that each block $\mathbf{J}_i$ is a matrix full of ones. This has several special implications: first, we do not need to calculate the spectral radius of the reservoir $\rho(\mathbf{J})$ anymore, since it is equal to the block-size b,

$$\mathbf{A} = \frac{\rho^*}{b}\mathbf{J}. \tag{12}$$

Furthermore, this reservoir architecture implies that in every iteration, each block $\mathbf{J}_i$ acts as an averaging operator on the reservoir states, as explained in the following. This is comparable to average pooling layers of other machine learning techniques since the averaging reduces the dimensionality of the reservoir states and "extracts" primarily features that are more robust.[31] We denote the mean between the ith and jth row of reservoir state $\mathbf{r}(t)$ as

$$\bar{r}_{i:j}(t) \equiv \frac{1}{j-i+1}\sum_{i}^{j} r_i(t). \tag{13}$$

Then, each block yields a vector of size $b \times 1$, which has equal values. For example, the first row of the multiplication $\mathbf{J} \cdot \mathbf{r}(t)$ reads

$$[\overbrace{1, \ldots, 1}^{b}, \overbrace{0, \ldots, 0}^{d-b}] \cdot \mathbf{r}(t) = \sum_{i=1}^{b} r_i(t) = b \cdot \bar{r}_{1:b}(t). \tag{14}$$

This is repeated exactly for the first b rows. Consequently, the reservoir multiplication $\mathbf{A} \cdot \mathbf{r}(t)$ from Eq. (3) yields

$$\mathbf{A} \cdot \mathbf{r}(t) = \frac{\rho^*}{b}\mathbf{J} \cdot \mathbf{r}(t) = \rho^* \begin{pmatrix} \bar{r}_{1:b}(t) \\ \bar{r}_{1:b}(t) \\ \vdots \\ \bar{r}_{(i-1)\cdot b+1:i\cdot b:k\cdot b}(t) \\ \bar{r}_{(i-1)\cdot b+1:i\cdot b:k\cdot b}(t) \\ \vdots \\ \bar{r}_{d-b+1:d}(t) \\ \bar{r}_{d-b+1:d}(t) \\ \vdots \end{pmatrix}. \tag{15}$$

Therefore, the reservoir multiplication contribution is identical for each block, which means that at each training step, the reservoir memory is the same for each block. This directly implies lower computational costs.

5. Implications for hardware reservoir computers

By separating the reservoir into multiple smaller reservoirs and by taking out the randomness we significantly reduce the complexity of the architecture, especially with regards to hardware implementations. Examples of the reservoir topologies and their respective spring-layouts according to the *Fruchterman–Reingold* force-directed algorithm[30] are illustrated in Fig. 2. We observe that the networks constructed fully or partly with Erdös–Rényi Networks have more complex interconnected network structures, while the networks with blocks of ones are ordered and have clear separate unified reservoirs.

This leads us to make the following assumptions on potential implications for hardware RCs:

- *Improved generalization*: Similar to ensemble methods[32] from other machine learning methods, the use of multiple smaller reservoirs can lead to a more diverse representation of the inputs. This potentially reduces the risk of overfitting and improves generalization to unseen data.
- *Enhanced robustness*: The failure of one reservoir unit can be potentially compensated by the other units, which can still contribute to the processing of the inputs. This makes the system more robust to noise and errors, especially in hardware implementations.

Downloaded from http://pubs.aip.org/aip/cha/article-pdf/doi/10.1063/5.0151290/17996145/063130_1_5.0151290.pdf

Published under an exclusive license by AIP Publishing

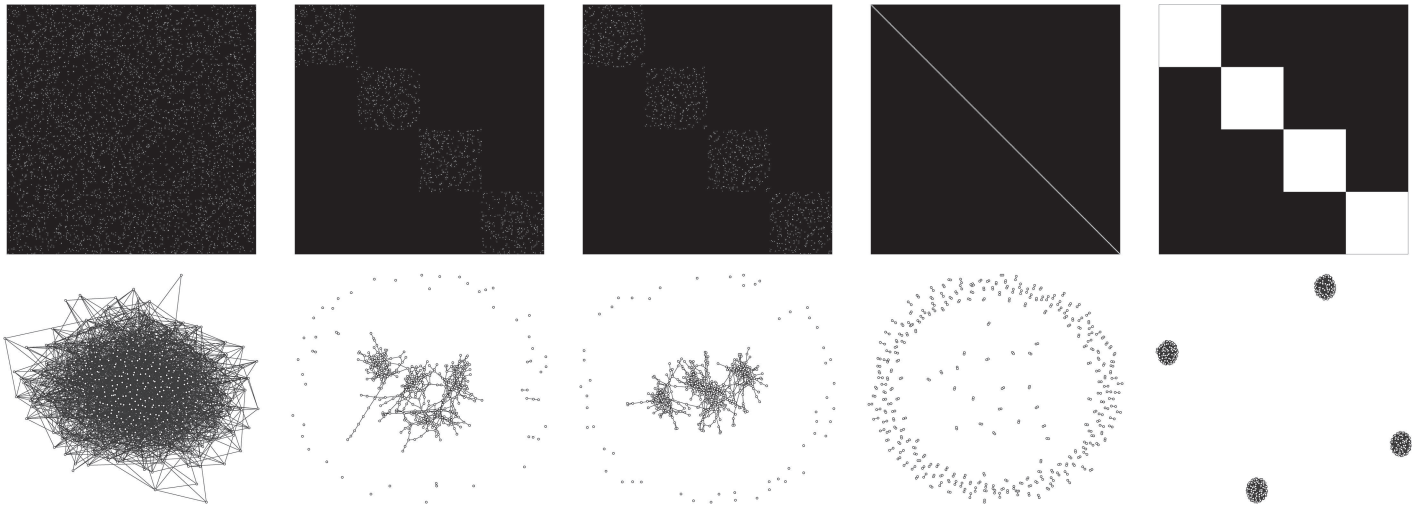

FIG. 2. Reservoir topologies (top) and corresponding spring-layouts[30] (bottom). The number of nodes is $d = 500$ for all reservoir topologies. In the top row, the white entries denote the presence of a network connection. In the bottom row, each node is represented by a white circle with black edges, while the connections are represented by black lines. From left to right, the illustrated network topologies are: (1) ordinary topology using a Erdös–Rényi graph, (2) block-diagonal topology with different Erdös–Rényi graphs of size $b = 125$ as blocks, (3) block-diagonal topology with equal Erdös–Rényi graphs of size $b = 125$ as blocks, (4) block-diagonal topology with matrices of ones of size $b = 2$ as blocks, and (5) block-diagonal topology with matrices of ones of size $b = 125$ as blocks.

- *Better scalability*: The computational and memory requirements can be reduced by parallelization, making the system more scalable and suitable for deployment on high-dimensional data.

Each of these assumptions requires different experimental setups—hence validating these potential implications is beyond the scope of this paper and is the subject of further research.

C. Measuring prediction performance

When forecasting nonlinear dynamical systems such as chaotic attractors, the goal of the predictions is not only to exactly replicate the actual short-time trajectory but also to reproduce the long-term statistical properties of the system called *climate*. This is important because by definition chaotic systems exhibit sensitive dependence on initial conditions and therefore small disturbances grow exponentially fast. Consequently, even if at first the short-term prediction is perfect, at some stage already numerical inaccuracies lead to the separation of the predicted and actual trajectories. However, for many applications, this is not a problem as long as the predicted trajectory still leads to the same attractor. In order to quantify this behavior, quantitative measures are needed that grasp the complex dynamics of the system. Therefore, we use the measures applied in the paper by Haluszczynski and Räth[11] and Haluszczynski.[26]

1. Forecast horizon

To quantify the quality and duration of the short-term prediction of the trajectory, we use a fairly simple measure, which we call forecast horizon, as used by Haluszczynski and Räth.[11] For that, we track the number of time steps during which the predicted $\mathbf{v}(t)$ and the actual trajectory $\mathbf{v}_R(t)$ are matching. As soon as one of the three coordinates exceeds certain deviation thresholds we consider the trajectories as not matching anymore. Throughout our study, we use

$$\tau = |\mathbf{v}(t) - \mathbf{v}_R(t)| > \delta, \tag{16}$$

where we define the thresholds as the standard deviation of the real data $\mathbf{v}_R(t)$,

$$\delta = \boldsymbol{\sigma}\,(\mathbf{v}_R(t)). \tag{17}$$

The aim of this measure is that small fluctuations around the actual trajectory, as well as minor deviations do not exceed the threshold. A higher value means that the prediction is close to the "true" trajectory over a longer period of time and has not deviated yet, although the underlying system is chaotic.

2. Correlation dimension

To assess the structural complexity of an attractor, we calculate its correlation dimension, which measures the dimensionality of the space populated by the trajectory.[33] It belongs to the measures for fractal dimensionality, which have been proposed by Mandelbrot[34] in 1967. The correlation dimension is based on the correlation integral,

$$C(r) = \lim_{N\to\infty} \frac{1}{N^2} \sum_{i,j=1}^{N} \theta\left(r - |\mathbf{x}_i - \mathbf{x}_j|\right) = \int_0^r d^3r' c(\mathbf{r}'), \tag{18}$$

where θ denotes the *Heaviside* function 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.

Downloaded from http://pubs.aip.org/aip/cha/article-pdf/doi/10.1063/5.0151290/17996145/063130_1_5.0151290.pdf

Published under an exclusive license by AIP Publishing

TABLE I. Benchmark measures calculated on the predictions using the traditional reservoir architecture and on the test data. Using the traditional architecture with reservoir dimension $d = 500$ and target spectral radius $\rho^* = 0.1$, we vary the random seeds and attractor starting points, make predictions, and evaluate the quality of the predictions. This yields a total of $n = 10\,000$ different realizations of forecast horizons, correlation dimensions, and largest Lyapunov exponents. The average and standard deviation over these prediction measures for the respective systems are denoted in the columns "Lorenz traditional" and "Halvorsen traditional." For each attractor starting point, we have a different test dataset. We calculate the correlation dimension and largest Lyapunov exponent of all test datasets and denote the average and standard deviation in the columns "Lorenz test" and "Halvorsen test."

Measure/System	Lorenz traditional	Lorenz test	Halvorsen traditional	Halvorsen test
Forecast horizon τ	219 ± 10	∞	382 ± 12	∞
Correlation dimension ν	2.01 ± 0.09	2.02 ± 0.02	1.98 ± 0.07	1.99 ± 0.02
Lyapunov exponent $\lambda_{\max}$	0.86 ± 0.05	0.87 ± 0.03	0.72 ± 0.06	0.74 ± 0.03

The correlation dimension ν is then defined by the power-law relationship,

$$C(r) \propto r^{\nu}. \tag{19}$$

For self-similar, strange attractors, this relationship holds for a certain range of r, which needs to be properly calibrated. The calculation of the correlation dimension is done using the *Grassberger Procaccia* algorithm.[35] It is purely data-based and does not require any knowledge of the underlying governing equations of the system. One advantage of the correlation dimension over other fractal measures is that it can be calculated having a comparably small number of data points available. In the context of this work, mainly the relative comparison among various predictions and actual trajectories is of interest and, therefore, the accuracy of the absolute values is not the highest priority.

3. Lyapunov exponent

Besides the fractal dimensionality, the statistical climate of an attractor is also characterized by its temporal complexity represented by the *Lyapunov* exponents.[36] They describe the average rate of divergence of nearby points in the phase space, and thus measure sensitivity with respect to initial conditions. 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. The magnitudes of λ_i quantify the time scale for which the system becomes unpredictable.[37] Since at least one positive exponent is the requirement for being classified as chaotic, it is sufficient for the purposes in this work to calculate only the largest Lyapunov exponent $\lambda_{\max}$:

$$d(t) = Ce^{\lambda_{\max} t}, \tag{20}$$

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. Here, we use the *Rosenstein* algorithm.[38] As mentioned for the correlation dimension, mainly a relative comparison is of interest in order to characterize states of the system rather than determine the exact absolute values. Again, for this measure, no model or knowledge of the underlying governing equations is required.

D. Benchmarks

In order to evaluate whether the predictions using the modified RC architecture are on par with the traditional setup, we run multiple predictions for different reservoir dimensions, target spectral radii, attractor starting points, and random seeds.

We do not observe significant changes for network dimensions $d \geq 400$ and find a target spectral radius of $\rho^* = 0.1$ to have the best overall prediction results. This is consistent to the findings by Haluszczynski and Räth.[11]

Thus, using the traditional architecture with reservoir dimension $d = 600$ and target spectral radius $\rho^* = 0.1$, we vary the random seeds and attractor starting points, make predictions, and evaluate the quality of the predictions. This yields a total of $n = 10\,000$ different realizations of forecast horizons, correlation dimensions, and largest Lyapunov exponents. The average and standard deviation over these prediction measures for the respective systems are denoted in the columns "Lorenz traditional" and "Halvorsen traditional" in Table I. We use these values as benchmarks for our analysis so that we can compare the prediction performance of our modified architecture to the traditional RC setup.

Furthermore, for each attractor starting point, we have a different test dataset. We calculate the correlation dimension and largest Lyapunov exponent of all test datasets and denote the average and standard deviation in the columns "Lorenz test" and "Halvorsen test" in Table I. These values can be seen as the "true" correlation dimension and largest Lyapunov exponent.

III. RESULTS

In the following, we present the results of our analyses for variations of different parameters to demonstrate the robustness of the modified architecture. The varied parameters are:

- *Network dimension d*: We vary the network dimension between $d \in \{400, 450, \ldots, 600\}$ as these are sensible values in RC research. We specifically choose multiples of 50 in order to have a decent number $\lfloor \frac{d}{b} \rfloor$ of block-sizes b.
- *Block-size b*: We vary the block-size b and set it to all divisors of the network dimension d. We exclude the divisor $b = 1$ as it essentially takes out the reservoir. Also, we do not use $b = d$ since it represents again a single network as in the traditional architecture.

Downloaded from http://pubs.aip.org/aip/cha/article-pdf/doi/10.1063/5.0151290/17996145/063130_1_5.0151290.pdf

Published under an exclusive license by AIP Publishing

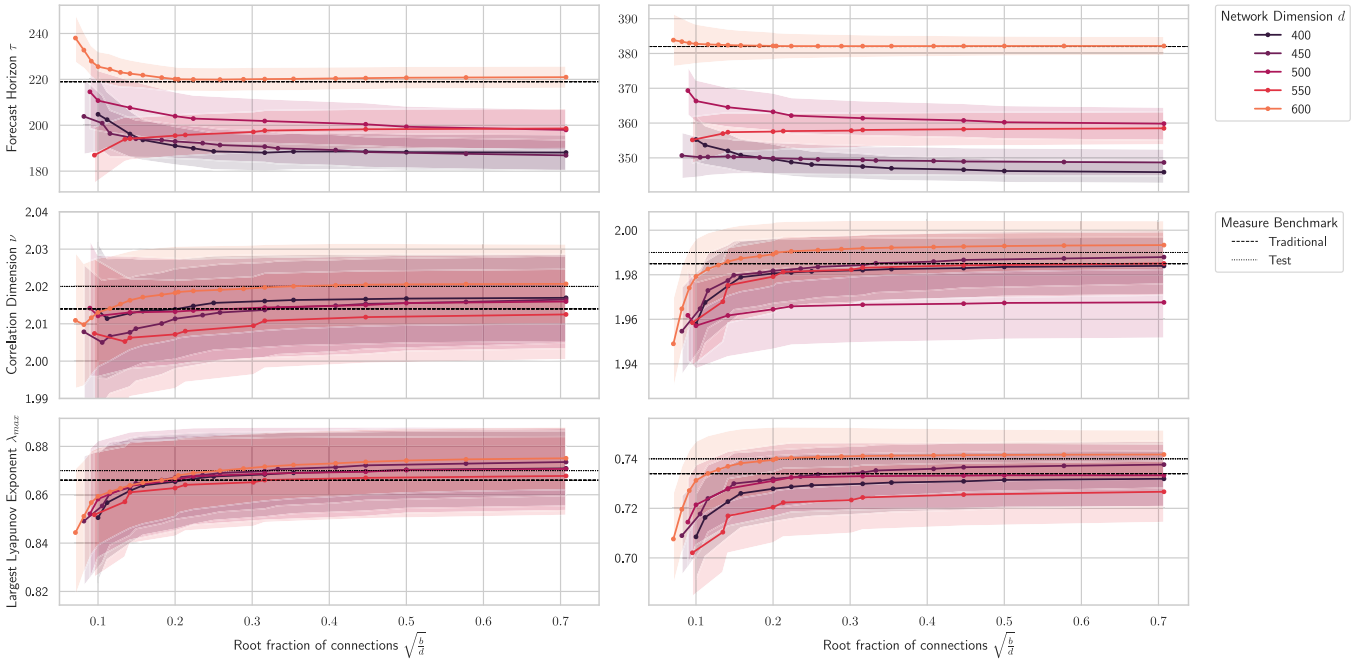

FIG. 3. Prediction measures for the Lorenz (left column) and Halvorsen systems (right column) for different random seeds of individual Erdös–Rényi blocks. The displayed prediction measures from top to bottom are: forecast horizon τ, correlation dimension ν, and the largest Lyapunov exponent λ_{max}. The differently colored lines represent different network dimensions with the corresponding shadowed area denoting the standard deviation over the variations. The dashed and dotted black horizontal lines represent the prediction measure benchmarks specified in Table I. The dashed line (lower) represents the average prediction performance of the traditional RC architecture, while the dotted line (higher) represents the average correlation dimension and largest Lyapunov exponent of the test data.

- *Target spectral radius* ρ^*: We vary the spectral radius from $\rho^* \in \{0.1, 0.2, \ldots, 2.0\}$ and find that, similarly to the traditional architecture, the target spectral radius of $\rho^* = 0.1$ yields the most robust prediction results. Henceforth, we set $\rho^* = 0.1$ as default.
- *Attractor starting points*: We choose 500 different starting points on the attractor as explained in Sec. II A 3.
- *Random seeds*: We choose 100 different random seeds across all components of the RC architecture which have randomness: input weights $\mathbf{W}_{in}$ and the reservoir $\mathbf{A}$ for block-diagonal Erdös–Rényi networks.

In order to make the figures easier to visualize, we calculate the fraction of connection and use the root of it as the x axis,

$$\sqrt{\frac{b}{d}}. \tag{21}$$

Thus, the higher the fraction, the bigger the blocks and the number of connections in the network. This is necessary because the number of divisors for each network size d is different and the divisors are not equally spaced.

A. Blocks of Erdös–Rényi networks

As mentioned before, we distinguish between two cases for the Erdös–Rényi blocks. First, where all the blocks are individual networks and second, where all blocks are equal.

- *Individual blocks*: We find that for both the Lorenz and the Halvorsen systems, all prediction measures are close to the respective benchmark values in Table I and even surpass them for a network size of 600. Generally, we observe that small block-sizes have a worse long-term prediction quality with regard to the correlation dimension and the largest Lyapunov exponent. However, they appear to have a better short-term forecast horizon. The standard deviation over the variation of random seeds is comparable to the benchmarks. The results are illustrated in Fig. 3. Furthermore, we find the variation for different input weights and starting points to be similarly robust.
- *Equal blocks*: Similar results can be observed for the equal blocks (Fig. 4)—however, the standard deviation is slightly lower. This can be explained by the reduced level in randomness since only one block is randomly constructed.

Generally, we find that the prediction performances stabilize for $\sqrt{\frac{b}{d}} > 0.3$ for both individual and equal blocks. As explained in Eq. (9), this speeds up the calculation of the reservoir spectral radius by a factor of ≈123 for individual blocks and ≈412 for equal blocks.

Furthermore, we observe that the modified architecture outperforms traditional RC with regard to long-term predictions while it performs slightly worse for short-term predictions. This can be inferred by looking at the correlation dimensions and largest Lyapunov exponents, which are higher than the respective values of

Downloaded from http://pubs.aip.org/aip/cha/article-pdf/doi/10.1063/5.0151290/17996145/063130_1_5.0151290.pdf

Published under an exclusive license by AIP Publishing

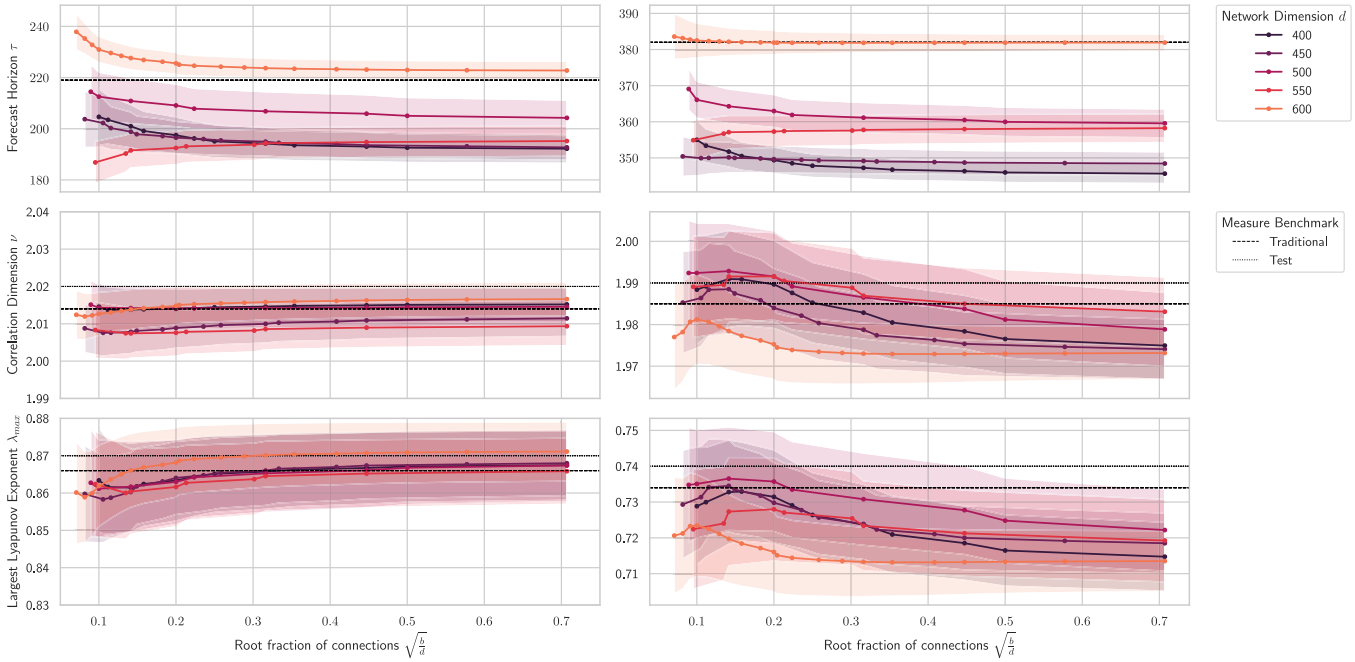

FIG. 4. Prediction measures for the Lorenz (left column) and Halvorsen systems (right column) for different random seeds of equal Erdös–Rényi blocks. The setup of this figure is similar to Fig. 3.

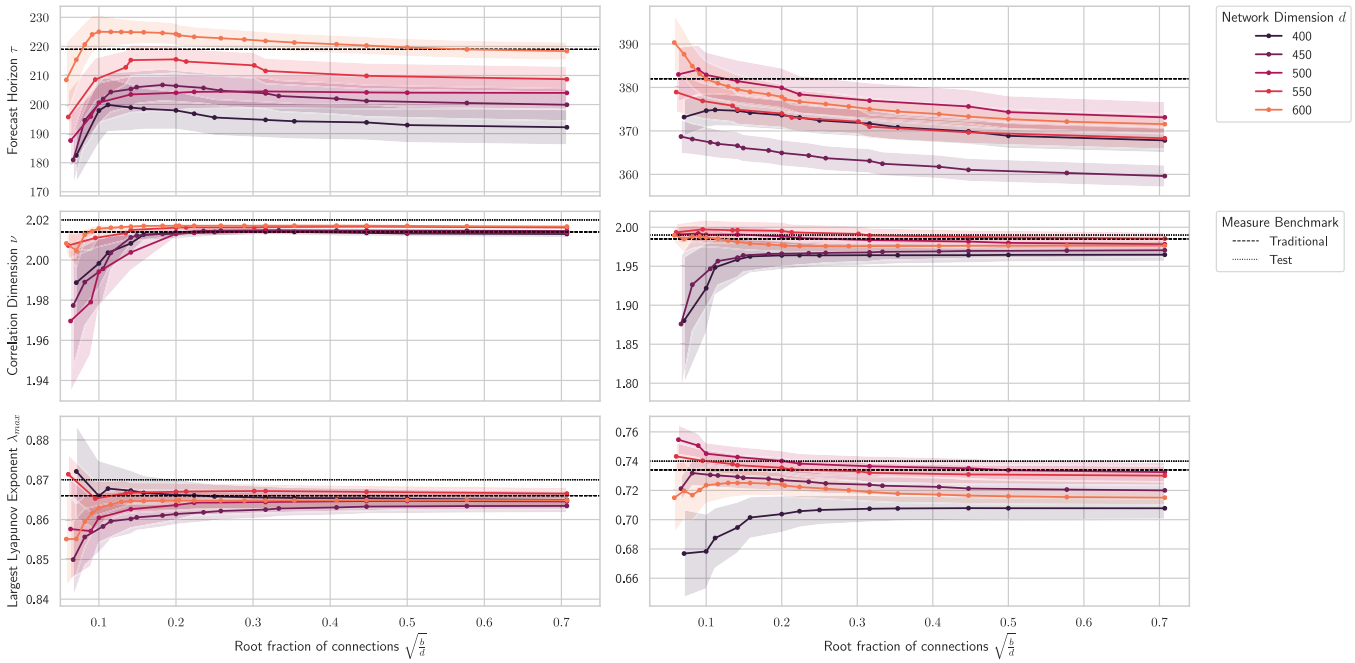

FIG. 5. Prediction measures for the Lorenz (left column) and Halvorsen systems (right column) for different input weights. The setup of this figure is similar to Fig. 3.

Downloaded from http://pubs.aip.org/aip/cha/article-pdf/doi/10.1063/5.0151290/17996145/063130_1_5.0151290.pdf

Published under an exclusive license by AIP Publishing

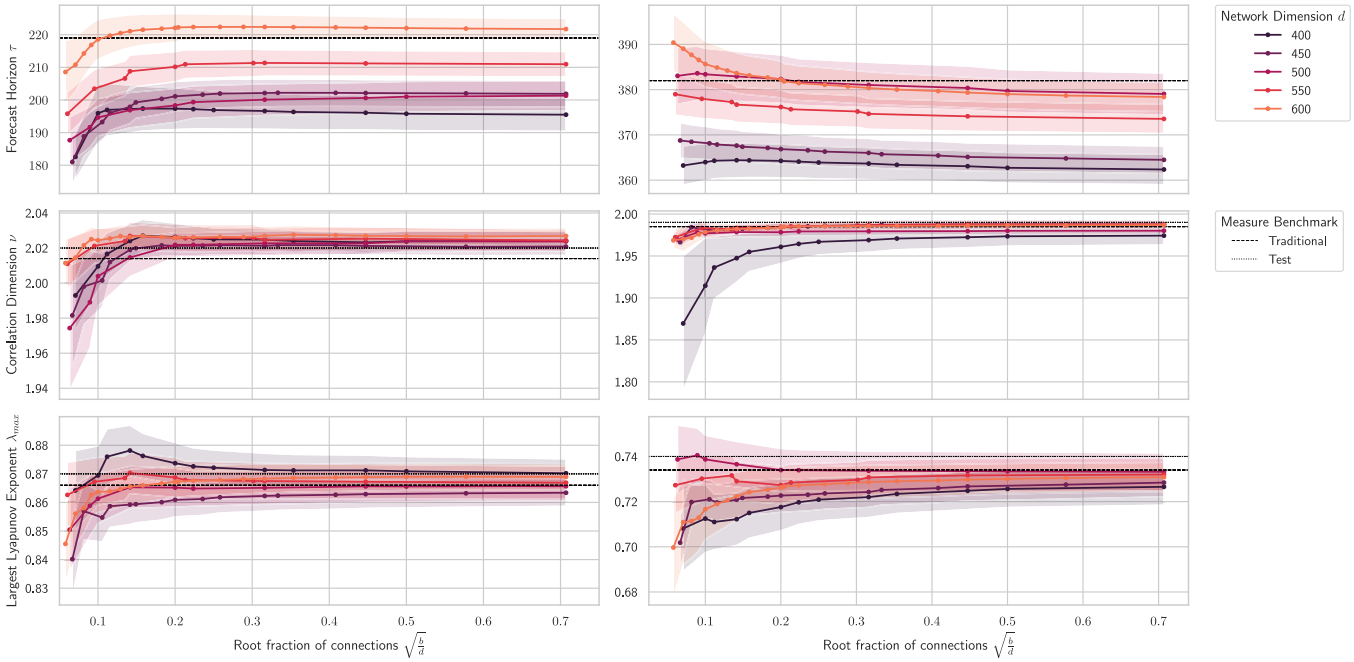

FIG. 6. Prediction measures for the Lorenz (left column) and Halvorsen systems (right column) for different attractor starting points. The setup of this figure is similar to Fig. 3.

the traditional RC architecture and closer to the "true" values of the test data. A comparable short-term prediction performance can be achieved by increasing the network dimension to $d = 600$. There we see that the forecast horizon is able to match the value of the traditional architecture and can even slightly outperform it for small block-sizes of $\sqrt{\frac{b}{d}} \approx 0.05$. In this specific case, the calculation speed of the spectral radius is increased by a factor of 160 000 and 3 200 000 for individual and equal blocks, respectively.

This behavior holds true for both the Lorenz and Halvorsen systems.

B. Blocks of matrices of ones

For the blocks of matrices of ones, the randomness of the reservoirs is taken out. Thus, we focus on the remaining randomness, which is present in the input weights $\mathbf{W}_{in}$, and the variation of the attractor starting point.

- *Input weights* $\mathbf{W}_{in}$: We find that for both the Lorenz and the Halvorsen systems, all prediction measures are close to the respective benchmark values and even surpass them for some network sizes. Generally, we observe that the performance is slightly worse than using blocks of Erdös–Rényi networks and the benchmarks. As expected, the standard deviation over the variation of input weights is comparable to the benchmarks and lower than for randomly constructed reservoirs. The results are illustrated in Fig. 5.
- *Attractor starting points*: A similar behavior can be observed for the variation in attractor starting points. However, we observe that network dimensions $d = 600$ even outperform the benchmarks for some block-sizes. The results are illustrated in Fig. 6.

In general, we find that the prediction quality for using blocks of ones as the reservoir is stable for a variation in input weights and attractor starting points. For certain block-sizes, the modified architecture is able to surpass the benchmarks for both short- and long-term predictions. Finding the best performing instance of these reservoirs can be done in a few iterations by varying the block-size for a large enough network dimension. Since the computationally expensive task of calculating the spectral radius of the reservoir is not necessary in this setup, fine-tuning the RC architecture with a parameter scan is fast and scalable.

IV. CONCLUSION AND OUTLOOK

In this paper, we introduce an alternative approach to constructing reservoir computers by replacing the reservoir, which is traditionally a single random network, with a block-diagonal matrix. This implies that the reservoir can be composed of multiple smaller reservoirs, which breaks with the common understanding of the reservoir as a single network.

Furthermore, we remove the randomness of the reservoir by using matrices of ones for the individual blocks.

We evaluate the short- and long-term prediction performance of block-diagonal reservoirs for two nonlinear chaotic systems: the Lorenz and Halvorsen systems. For that, we use three measures:

Downloaded from http://pubs.aip.org/aip/cha/article-pdf/doi/10.1063/5.0151290/17996145/063130_1_5.0151290.pdf

Published under an exclusive license by AIP Publishing

forecast horizon, correlation dimension, and the largest Lyapunov exponent. We find that—overall—the quality of the predictions is comparable to classical random networks. Although the block-diagonal reservoirs tend to perform slightly worse than the traditional architecture on average, some block-diagonal reservoirs with appropriate size of the blocks perform as well and sometimes even better than the conventional network reservoirs.

We find the result to be robust over variations in network dimensions, block-sizes, target spectral radii, attractor starting points, input weights, and random seeds.

This modified reservoir architecture not only has immediate large benefits regarding the computational effort but also the great potential for simple and fast hardware implementations of reservoir computers becomes obvious. Following this line of research is subject to further research.

We discover many interesting lines of future research. Further directions we find promising to take a look at are: understanding whether block-diagonal reservoirs improve generalization, enhance robustness, or increase scalability.

Current and future work is dedicated to the investigation of these questions—not the least because the answers to them will shed new light on the complexity of the underlying dynamical system.

ACKNOWLEDGMENTS

We would like to acknowledge the DLR for providing code and computational resources.

AUTHOR DECLARATIONS

Conflict of Interest

The authors have no conflicts to disclose.

Author Contributions

Haochun Ma: Conceptualization (equal); Formal analysis (equal); Investigation (equal); Methodology (equal); Software (equal); Validation (equal); Visualization (equal); Writing – original draft (equal); Writing – review & editing (equal). **Davide Prosperino:** Formal analysis (equal); Software (equal); Writing – original draft (equal); Writing – review & editing (equal). **Alexander Haluszczynski:** Formal analysis (equal); Validation (equal); Visualization (equal); Writing – original draft (equal); Writing – review & editing (equal). **Christoph Räth:** Conceptualization (equal); Supervision (equal); Writing – original draft (equal); Writing – review & editing (equal).

DATA AVAILABILITY

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

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]A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath, "Generative adversarial networks: An overview," IEEE Signal Process. Mag. **35**, 53–65 (2018).
[3]J. Zhang, Y. Wang, P. Molino, L. Li, and D. S. Ebert, "Manifold: A model-agnostic framework for interpretation and diagnosis of machine learning models," IEEE Trans. Visualiz. Comp. Graphics **25**, 364–373 (2018).
[4]R. Roscher, B. Bohn, M. F. Duarte, and J. Garcke, "Explainable machine learning for scientific insights and discoveries," IEEE Access **8**, 42200–42216 (2020).
[5]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).
[6]D. Prokhorov, "Echo state networks: Appeal and challenges," in *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005* (IEEE, 2005), Vol. 3, pp. 1463–1466.
[7]S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," Int. J. Uncertain. Fuzziness Knowledge-Based Syst. **6**, 107–116 (1998).
[8]A. Haluszczynski, J. Aumeier, J. Herteux, and C. Räth, "Reducing network size and improving prediction stability of reservoir computing," Chaos **30**, 063136 (2020).
[9]G. Holzmann, "Reservoir computing: A powerful black-box framework for nonlinear audio processing," in *International Conference on Digital Audio Effects (DAFx)* (Citeseer, 2009).
[10]T. L. Carroll and L. M. Pecora, "Network structure effects in reservoir computers," Chaos **29**, 083130 (2019).
[11]A. Haluszczynski and C. Räth, "Good and bad predictions: Assessing and improving the replication of chaotic attractors by means of reservoir computing," Chaos **29**, 103143 (2019).
[12]W. Maass, T. Natschläger, and H. Markram, "Real-time computing without stable states: A new framework for neural computation based on perturbations," Neural Comput. **14**, 2531–2560 (2002).
[13]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, Bonn, Germany, No. 148, p. 13 (2001).
[14]D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," Nature **393**, 440–442 (1998).
[15]R. Albert and A.-L. Barabási, "Statistical mechanics of complex networks," Rev. Mod. Phys. **74**, 47 (2002).
[16]A. D. Broido and A. Clauset, "Scale-free networks are rare," Nat. Commun. **10**, 1017 (2019).
[17]M. Gerlach and E. G. Altmann, "Testing statistical laws in complex systems," Phys. Rev. Lett. **122**, 168301 (2019).
[18]A. Griffith, A. Pomerance, and D. J. Gauthier, "Forecasting chaotic systems with very low connectivity reservoir computers," Chaos **29**, 123108 (2019).
[19]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).
[20]Z. Lu, B. R. Hunt, and E. Ott, "Attractor reconstruction by machine learning," Chaos **28**, 061104 (2018).
[21]E. N. Lorenz, "Deterministic nonperiodic flow," J. Atmos. Sci. **20**, 130–141 (1963).
[22]J. Herteux and C. Räth, "Breaking symmetries of the reservoir equations in echo state networks," Chaos **30**, 123142 (2020).
[23]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.
[24]E. Hairer, S. P. Nørsett, and G. Wanner, *Solving Ordinary Differential Equations I* (Springer, 1993).
[25]H. Jaeger and H. Haas, "Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication," Science **304**, 78–80 (2004).
[26]A. Haluszczynski, "Prediction and control of nonlinear dynamical systems using machine learning," Ph.D, dissertation (Ludwig-Maximilians-Universität München, 2021).
[27]P. Erdös, A. Rényi *et al.*, "On the evolution of random graphs," Publ. Math. Inst. Hung. Acad. Sci. **5**, 17–60 (1960), https://static.renyi.hu/~p_erdos/1960-10.pdf

Downloaded from http://pubs.aip.org/aip/cha/article-pdf/doi/10.1063/5.0151290/17996145/063130_1_5.0151290.pdf

Published under an exclusive license by AIP Publishing

[28]A. E. Hoerl and R. W. Kennard, "Ridge regression: Applications to nonorthogonal problems," Technometrics **12**, 69–82 (1970).
[29]C. C. Paige, "Bidiagonalization of matrices and solution of linear equations," SIAM J. Numer. Anal. **11**, 197–209 (1974).
[30]S. G. Kobourov, "Spring embedders and force directed graph drawing algorithms," arXiv:1201.3011 (2012).
[31]D. Yu, H. Wang, P. Chen, and Z. Wei, "Mixed pooling for convolutional neural networks," in *Rough Sets and Knowledge Technology: 9th International Conference, RSKT 2014, Shanghai, China, October 24–26, 2014, Proceedings 9* (Springer, 2014), pp. 364–375.
[32]T. G. Dietterich, "Ensemble methods in machine learning," in *Multiple Classifier Systems: First International Workshop, MCS 2000 Cagliari, Italy, June 21–23, 2000 Proceedings 1* (Springer, 2000), pp. 1–15.
[33]P. Grassberger and I. Procaccia, "Measuring the strangeness of strange attractors," in *The Theory of Chaotic Attractors* (Springer, 2004), pp. 170–189.
[34]B. Mandelbrot, "How long is the coast of Britain? Statistical self-similarity and fractional dimension," Science **156**, 636–638 (1967).
[35]P. Grassberger, "Generalized dimensions of strange attractors," Phys. Lett. A **97**, 227–230 (1983).
[36]A. Wolf, J. B. Swift, H. L. Swinney, and J. A. Vastano, "Determining Lyapunov exponents from a time series," Physica D **16**, 285–317 (1985).
[37]R. Shaw, "Strange attractors, chaotic behavior, and information flow," Z. Naturforsch. A **36**, 80–112 (1981).
[38]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).

Downloaded from http://pubs.aip.org/aip/cha/article-pdf/doi/10.1063/5.0151290/17996145/063130_1_5.0151290.pdf

Published under an exclusive license by AIP Publishing