ARTICLE

https://doi.org/10.1038/s41467-021-25801-2 OPEN

Next generation reservoir computing

Daniel J. Gauthier[1,2], Erik Bollt[3,4], Aaron Griffith[1] & Wendson A. S. Barbosa[1]

Reservoir computing is a best-in-class machine learning algorithm for processing information generated by dynamical systems using observed time-series data. Importantly, it requires very small training data sets, uses linear optimization, and thus requires minimal computing resources. However, the algorithm uses randomly sampled matrices to define the underlying recurrent neural network and has a multitude of metaparameters that must be optimized. Recent results demonstrate the equivalence of reservoir computing to nonlinear vector autoregression, which requires no random matrices, fewer metaparameters, and provides interpretable results. Here, we demonstrate that nonlinear vector autoregression excels at reservoir computing benchmark tasks and requires even shorter training data sets and training time, heralding the next generation of reservoir computing.

[1] The Ohio State University, Department of Physics, 191 West Woodruff Ave., Columbus, OH 43210, USA. [2] ResCon Technologies, LLC, PO Box 21229 Columbus, OH 43221, USA. [3] Clarkson University, Department of Electrical and Computer Engineering, Potsdam, NY 13669, USA. [4] Clarkson Center for Complex Systems Science (C3S2), Potsdam, NY 13699, USA. ✉email: gauthier.51@osu.edu

A dynamical system evolves in time, with examples including the Earth's weather system and human-built devices such as unmanned aerial vehicles. One practical goal is to develop models for forecasting their behavior. Recent machine learning (ML) approaches can generate a model using only observed data, but many of these algorithms tend to be data hungry, requiring long observation times and substantial computational resources.

Reservoir computing[1,2] is an ML paradigm that is especially well-suited for learning dynamical systems. Even when systems display chaotic[3] or complex spatiotemporal behaviors[4], which are considered the hardest-of-the-hard problems, an optimized reservoir computer (RC) can handle them with ease.

As described in greater detail in the next section, an RC is based on a recurrent artificial neural network with a pool of interconnected neurons—the reservoir, an input layer feeding observed data $\mathbf{X}$ to the network, and an output layer weighting the network states as shown in Fig. 1. To avoid the vanishing gradient problem[5] during training, the RC paradigm randomly assigns the input-layer and reservoir link weights. Only the weights of the output links $\mathbf{W}_{\text{out}}$ are trained via a regularized linear least-squares optimization procedure[6]. Importantly, the regularization parameter α is set to prevent overfitting to the training data in a controlled and well understood manner and makes the procedure noise tolerant. RCs perform as well as other ML methods, such as Deep Learning, on dynamical systems tasks but have substantially smaller data set requirements and faster training times[7,8].

Using random matrices in an RC presents problems: many perform well, but others do not all and there is little guidance to select good or bad matrices. Furthermore, there are several RC metaparameters that can greatly affect its performance and require optimization[9–13]. Recent work suggests that good matrices and metaparameters can be identified by determining whether the reservoir dynamics r synchronizes in a generalized sense to $\mathbf{X}$[14,15], but there are no known design rules for obtaining generalized synchronization.

Recent RC research has identified requirements for realizing a general, universal approximator of dynamical systems. A universal approximator can be realized using an RC with nonlinear activation at nodes in the recurrent network and an output layer (known as the feature vector) that is a weighted linear sum of the network nodes under the weak assumptions that the dynamical system has bounded orbits[16].

Less appreciated is the fact that an RC with linear activation nodes combined with a feature vector that is a weighted sum of nonlinear functions of the reservoir node values is an equivalently powerful universal approximator[16,17]. Furthermore, such an RC is mathematically identical to a nonlinear vector autoregression (NVAR) machine[18]. Here, no reservoir is required: the feature vector of the NVAR consists of k time-delay observations of the dynamical system to be learned and nonlinear functions of these observations, as illustrated in Fig. 1, a surprising result given the apparent lack of a reservoir!

These results are in the form of an existence proof: There exists an NVAR that can perform equally well as an optimized RC and, in turn, the RC is implicit in an NVAR. Here, we demonstrate that it is easy to design a well-performing NVAR for three challenging RC benchmark problems: (1) forecasting the short-term dynamics; (2) reproducing the long-term climate of a

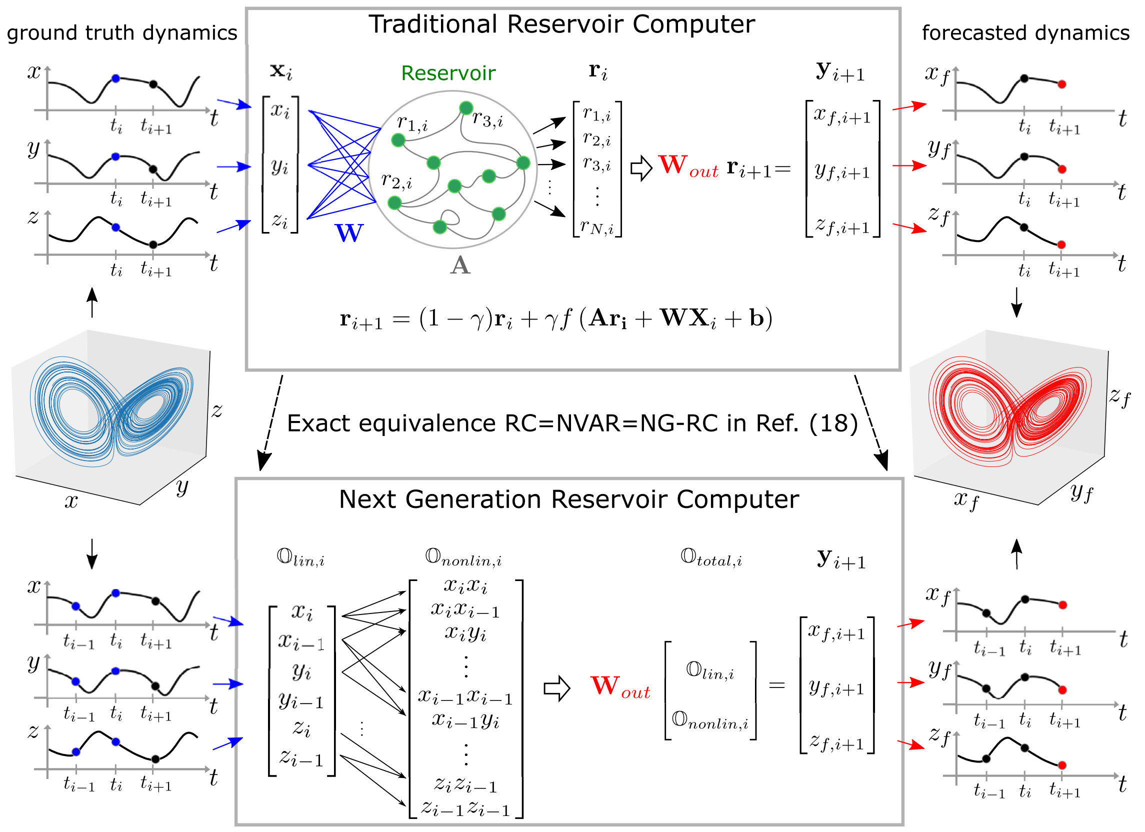
ground truth dynamics
Traditional Reservoir Computer
forecasted dynamics
Reservoir
$\mathbf{x}_i$ $\mathbf{r}_i$ $\mathbf{y}_{i+1}$
$r_{1,i}$ $r_{2,i}$ $r_{3,i}$ $r_{N,i}$
W
A
$\mathbf{W}_{out}\,\mathbf{r}_{i+1}=$
$x_{f,i+1}$ $y_{f,i+1}$ $z_{f,i+1}$
$\mathbf{r}_{i+1} = (1-\gamma)\mathbf{r}_i + \gamma f(\mathbf{A}\mathbf{r_i} + \mathbf{W}\mathbf{X}_i + \mathbf{b})$
Exact equivalence RC=NVAR=NG-RC in Ref. (18)
Next Generation Reservoir Computer
$\mathbb{O}_{lin,i}$ $\mathbb{O}_{nonlin,i}$ $\mathbb{O}_{total,i}$
x_i x_{i-1} y_i y_{i-1} z_i z_{i-1}
x_ix_i x_ix_{i-1} x_iy_i $x_{i-1}x_{i-1}$ $x_{i-1}y_i$ z_iz_{i-1} $z_{i-1}z_{i-1}$
$\mathbf{W}_{out}$
t_{i-1} t_i t_{i+1} t
x y z x_f y_f z_f

Fig. 1 A traditional RC is implicit in an NG-RC. (top) A traditional RC processes time-series data associated with a strange attractor (blue, middle left) using an artificial recurrent neural network. The forecasted strange attractor (red, middle right) is a linear weight of the reservoir states. (bottom) The NG-RC performs a forecast using a linear weight of time-delay states (two times shown here) of the time series data and nonlinear functionals of this data (quadratic functional shown here).

chaotic system (that is, reconstructing the attractors shown in Fig. 1); and (3) inferring the behavior of unseen data of a dynamical system.

Predominantly, the recent literature has focused on the first benchmark of short-term forecasting of stochastic processes time-series data[16], but the importance of high-accuracy forecasting and inference of unseen data cannot be overstated. The NVAR, which we call the next generation RC (NG-RC), displays state-of-the-art performance on these tasks because it is associated with an implicit RC, and uses exceedingly small data sets and side-steps the random and parametric difficulties of directly implementing a traditional RC.

We briefly review traditional RCs and introduce an RC with linear reservoir nodes and a nonlinear output layer. We then introduce the NG-RC and discuss the remaining metaparameters, introduce two model systems we use to showcase the performance of the NG-RC, and present our findings. Finally, we discuss the implications of our work and future directions.

The purpose of an RC illustrated in the top panel of Fig. 1 is to broadcast input data $\mathbf{X}$ into the higher-dimensional reservoir network composed of N interconnected nodes and then to combine the resulting reservoir state into an output $\mathbf{Y}$ that closely matches the desired output $\mathbf{Y}_d$. The strength of the node-to-node connections, represented by the connectivity (or adjacency) matrix $\mathbf{A}$, are chosen randomly and kept fixed. The data to be processed $\mathbf{X}$ is broadcast into the reservoir through the input layer with fixed random coefficients $\mathbf{W}$. The reservoir is a dynamic system whose dynamics can be represented by

$$\mathbf{r}_{i+1} = (1-\gamma)\mathbf{r}_i + \gamma f\left(\mathbf{A}\mathbf{r}_i + \mathbf{W}\mathbf{X}_i + \mathbf{b}\right), \tag{1}$$

where $\mathbf{r}_i = \left[r_{1,i}, r_{2,i}, \ldots, r_{N,i}\right]^T$ is an N-dimensional vector with component $r_{j,i}$ representing the state of the jth node at the time t_i, γ is the decay rate of the nodes, f an activation function applied to each vector component, and $\mathbf{b}$ is a node bias vector. For simplicity, we choose γ and $\mathbf{b}$ the same for all nodes. Here, time is discretized at a finite sample time dt and i indicates the ith time step so that $dt = t_{i+1}$-t_i. Thus, the notations $\mathbf{r}_i$ and $\mathbf{r}_{i+1}$ represent the reservoir state in consecutive time steps. The reservoir can also equally well be represented by continuous-time ordinary differential equations that may include the possibility of delays along the network links[19].

The output layer expresses the RC output $\mathbf{Y}_{i+1}$ as a linear transformation of a feature vector $\mathbb{O}_{\mathrm{total},i+1}$, constructed from the reservoir state $\mathbf{r}_{i+1}$, through the relation

$$\mathbf{Y}_{i+1} = \mathbf{W}_{\mathrm{out}}\mathbb{O}_{\mathrm{total},i+1}, \tag{2}$$

where $\mathbf{W}_{\mathrm{out}}$ is the output weight matrix and the subscript total indicates that it can be composed of constant, linear, and nonlinear terms as explained below. The standard approach, commonly used in the RC community, is to choose a nonlinear activation function such as $f(x) = \tanh(x)$ for the nodes and a linear feature vector $\mathbb{O}_{\mathrm{total},i+1} = \mathbb{O}_{\mathrm{lin},i+1} = \mathbf{r}_{i+1}$ in the output layer. The RC is trained using supervised training via regularized least-squares regression. Here, the training data points generate a block of data contained in $\mathbb{O}_{\mathrm{total}}$ and we match $\mathbf{Y}$ to the desired output $\mathbf{Y}_d$ in a least-square sense using Tikhonov regularization so that $\mathbf{W}_{\mathrm{out}}$ is given by

$$\mathbf{W}_{\mathrm{out}} = \mathbf{Y}_d\mathbb{O}_{\mathrm{total}}{}^T\left(\mathbb{O}_{\mathrm{total}}\mathbb{O}_{\mathrm{total}}{}^T + \alpha\mathbf{I}\right)^{-1}, \tag{3}$$

where the regularization parameter α, also known as ridge parameter, is set to prevent overfitting to the training data and $\mathbf{I}$ is the identity matrix.

A different approach to RC is to move the nonlinearity from the reservoir to the output layer[16,18]. In this case, the reservoir nodes are chosen to have a linear activation function $f(\mathbf{r}) = \mathbf{r}$, while the feature vector $\mathbb{O}_{\mathrm{total}}$ becomes nonlinear. A simple example of such RC is to extend the standard linear feature vector to include the squared values of all nodes, which are obtained through the Hadamard product $\mathbf{r} \odot \mathbf{r} = \left[r_1^2, r_2^2, \ldots, r_N^2\right]^T$[18]. Thus, the nonlinear feature vector is given by

$$\mathbb{O}_{\mathrm{total}} = \mathbf{r} \oplus (\mathbf{r} \odot \mathbf{r}) = \left[r_1, r_2, \ldots, r_N, r_1^2, r_2^2, \ldots, r_N^2\right]^T, \tag{4}$$

where $\oplus$ represents the vector concatenation operation. A linear reservoir with a nonlinear output is an equivalently powerful universal approximator[16] and shows comparable performance to the standard RC[18].

In contrast, the NG-RC creates a feature vector directly from the discretely sample input data with no need for a neural network. Here, $\mathbb{O}_{\mathrm{total}} = c \oplus \mathbb{O}_{\mathrm{lin}} \oplus \mathbb{O}_{\mathrm{nonlin}}$, where c is a constant and $\mathbb{O}_{\mathrm{nonlin}}$ is a nonlinear part of the feature vector. Like a traditional RC, the output is obtained using these features in Eq. 3. We now discuss forming these features.

The linear features $\mathbb{O}_{\mathrm{lin},i}$ at time step i is composed of observations of the input vector $\mathbf{X}$ at the current and at k-1 previous times steps spaced by s, where (s-1) is the number of skipped steps between consecutive observations. If $\mathbf{X}_i = \left[x_{1,i}, x_{2,i}, \ldots, x_{d,i}\right]^T$ is a d-dimensional vector, $\mathbb{O}_{\mathrm{lin},i}$ has $d\,k$ components, and is given by

$$\mathbb{O}_{\mathrm{lin},i} = \mathbf{X}_i \oplus \mathbf{X}_{i-s} \oplus \mathbf{X}_{i-2s} \oplus \ldots \oplus \mathbf{X}_{i-(k-1)s}. \tag{5}$$

Based on the general theory of universal approximators[16,20], k should be taken to be infinitely large. However, it is found in practice that the Volterra series converges rapidly, and hence truncating k to small values does not incur large error. This can also be motivated by considering numerical integration methods of ordinary differential equations where only a few subintervals (steps) in a multistep integrator are needed to obtain high accuracy. We do not subdivide the step size here, but this analogy motivates why small values of k might give good performance in the forecasting tasks considered below.

An important aspect of the NG-RC is that its warm-up period only contains (sk) time steps, which are needed to create the feature vector for the first point to be processed. This is a dramatically shorter warm-up period in comparison to traditional RCs, where longer warm-up times are needed to ensure that the reservoir state does not depend on the RC initial conditions. For example, with $s = 1$ and $k = 2$ as used for some examples below, only two warm-up data points are needed. A typical warm-up time in traditional RC for the same task can be upwards of 10^3 to 10^5 data points[12,14]. A reduced warm-up time is especially important in situations where it is difficult to obtain data or collecting additional data is too time-consuming.

For the case of a driven dynamical system, $\mathbb{O}_{\mathrm{lin}}(t)$ also includes the drive signal[21]. Similarly, a system in which one or more accessible system parameters are adjusted, $\mathbb{O}_{\mathrm{lin}}(t)$ also includes these parameters[21,22].

The nonlinear part $\mathbb{O}_{\mathrm{nonlin}}$ of the feature vector is a nonlinear function of $\mathbb{O}_{\mathrm{lin}}$. While there is great flexibility in choosing the nonlinear functionals, we find that polynomials provide good prediction ability. Polynomial functionals are the basis of a Volterra representation for dynamical systems[20] and hence they are a natural starting point. We find that low-order polynomials are enough to obtain high performance.

All monomials of the quadratic polynomial, for example, are captured by the outer product $\mathbb{O}_{\mathrm{lin}} \otimes \mathbb{O}_{\mathrm{lin}}$, which is a symmetric matrix with $(dk)^2$ elements. A quadratic nonlinear feature vector $\mathbb{O}^{(2)}_{\mathrm{nonlinear}}$, for example, is composed of the (dk) $(dk+1)/2$ unique monomials of $\mathbb{O}_{\mathrm{lin}} \otimes \mathbb{O}_{\mathrm{lin}}$, which are given by the upper triangular elements of the outer product tensor. We define $\lceil\otimes\rceil$ as the operator that collects the unique monomials in a vector. Using this notation, a p-order polynomial feature vector is given by

$$\mathbb{O}^{(p)}_{\text{nonlinear}} = \mathbb{O}_{\text{lin}} \lceil \otimes \rceil \mathbb{O}_{\text{lin}} \lceil \otimes \rceil \ldots \lceil \otimes \rceil \mathbb{O}_{\text{lin}} \qquad (6)$$

with $\mathbb{O}_{\text{lin}}$ appearing p times.

Recently, it was mathematically proven that the NVAR method is equivalent to a linear RC with polynomial nonlinear readout[18]. This means that every NVAR implicitly defines the connectivity matrix and other parameters of a traditional RC described above and that every linear polynomial-readout RC can be expressed as an NVAR. However, the traditional RC is more computationally expensive and requires optimizing many meta-parameters, while the NG-RC is more efficient and straightforward. The NG-RC is doing the same work as the equivalent traditional RC with a full recurrent neural network, but we do not need to find that network explicitly or do any of the costly computation associated with it.

We now introduce models and tasks we use for showcasing the performance of NG-RC. For one of the forecasting tasks and the inference task discussed in the next section, we generate training and testing data by numerically integrating a simplified model of a weather system[23] developed by Lorenz in 1963. It consists of a set of three coupled nonlinear differential equations given by

$$\dot{x} = 10(y - x),\ \dot{y} = x(28 - z) - y,\ \dot{z} = xy - 8z/3, \qquad (7)$$

where the state $\mathbf{X}(t) \equiv [x(t), y(t), z(t)]^T$ is a vector whose components are Rayleigh–Bénard convection observables. It displays deterministic chaos, sensitive dependence to initial conditions—the so-called butterfly effect—and the phase space trajectory forms a strange attractor shown in Fig. 1. For future reference, the Lyapunov time for Eq. 7, which characterizes the divergence timescale for a chaotic system, is 1.1-time units. Below, we refer to this system as Lorenz63.

We also explore using the NG-RC to predict the dynamics of a double-scroll electronic circuit[24] whose behavior is governed by

$$\begin{aligned} \dot{V}_1 &= V_1/R_1 - \Delta V/R_2 - 2I_r \sinh(\beta \Delta V), \\ \dot{V}_2 &= \Delta V/R_2 + 2I_r \sinh(\beta \Delta V) - I, \\ \dot{I} &= V_2 - R_4 I \end{aligned} \qquad (8)$$

in dimensionless form, where $\Delta V = V_1 - V_2$. Here, we use the parameters $R_1 = 1.2$, $R_2 = 3.44$, $R_4 = 0.193$, $\beta = 11.6$, and $I_r = 2.25 \times 10^{-5}$, which give a Lyapunov time of 7.81-time units.

We select this system because the vector field is not of a polynomial form and ΔV is large enough at some times that a truncated Taylor series expansion of the sinh function gives rise to large differences in the predicted attractor. This task demonstrates that the polynomial form of the feature vector can work for nonpolynomial vector fields as expected from the theory of Volterra representations of dynamical systems[20].

In the two forecasting tasks presented below, we use an NG-RC to forecast the dynamics of Lorenz63 and the double-scroll system using one-step-ahead prediction. We start with a listening phase, seeking a solution to $\mathbf{X}(t + dt) = \mathbf{W}_{\text{out}} \mathbb{O}_{\text{total}}(t)$, where $\mathbf{W}_{\text{out}}$ is found using Tikhonov regularization[6]. During the forecasting (testing) phase, the components of $\mathbf{X}(t)$ are no longer provided to the NG-RC and the predicted output is fed back to the input. Now, the NG-RC is an autonomous dynamical system that predicts the systems' dynamics if training is successful.

The total feature vector used for the Lorenz63 forecasting task is given by

$$\mathbb{O}_{\text{total}} = c \oplus \mathbb{O}_{\text{lin}} \oplus \mathbb{O}^{(2)}_{\text{nonlinear}}, \qquad (9)$$

which has $[1 + d\,k + (d\,k)\,(d\,k+1)/2]$ components.

For the double-scroll system forecasting task, we notice that the attractor has odd symmetry and has zero mean for all variables for the parameters we use. To respect these characteristics, we take

$$\mathbb{O}_{\text{total}} = \mathbb{O}_{\text{lin}} \oplus \mathbb{O}^{(3)}_{\text{nonlinear}} \qquad (10)$$

which has $[d\,k + (d\,k)\,(d\,k+1)\,(d\,k+2)/6]$ components.

For these forecasting tasks, the NG-RC learns simultaneously the vector field and an efficient one-step-ahead integrator to find a mapping from one time to the next without having to learn each separately as in other nonlinear state estimation approaches[25–28]. The one-step-ahead mapping is known as the flow of the dynamical system and hence the NG-RC learns the flow. To allow the NG-RC to focus on the subtle details of this process, we use a simple Euler-like integration step as a lowest-order approximation to a forecasting step by modifying Eq. 2 so that the NG-RC learns the difference between the current and future step. To this end, Eq. 2 is replaced by

$$\mathbf{X}_{i+1} = \mathbf{X}_i + \mathbf{W}_{\text{out}} \mathbb{O}_{\text{total},i}. \qquad (11)$$

In the third task, we provide the NG-RC with all three Lorenz63 variables during training with the goal of inferring the next-step-ahead prediction of one of the variables from the others. During testing, we only provide it with the x and y variables and infer the z variable. This task is important for applications where it is possible to obtain high-quality information about a dynamical variable in a laboratory setting, but not in field deployment. In the field, the observable sensory information is used to infer the missing data.

Results

For the first task, the ground-truth Lorenz63 strange attractor is shown in Fig. 2a. The training phase uses only the data shown in Fig. 2b–d, which consists of 400 data points for each variable with $dt = 0.025$, $k = 2$, and $s = 1$. The training compute time is <10 ms using Python running on a single-core desktop processor (see Methods). Here, $\mathbb{O}_{\text{total}}$ has 28 components and $\mathbf{W}_{\text{out}}$ has dimension (3×28). The set needs to be long enough for the phase-space trajectory to explore both wings of the strange attractor. The plot is overlayed with the NG-RC predictions during training; no difference is visible on this scale.

The NG-RC is then placed in the prediction phase; a qualitative inspection of the predicted (Fig. 2e) and true (Fig. 2a) strange attractors shows that they are very similar, indicating that the NG-RC reproduces the long-term climate of Lorenz63 (benchmark problem 2). As seen in Fig. 2f–h, the NG-RC does a good job of predicting Lorenz63 (benchmark 1), comparable to an optimized traditional RC[3,12,14] with 100s to 1000s of reservoir nodes. The NG-RC forecasts well out to ~5 Lyapunov times.

In Supplementary Note 1, we give other quantitative measurements of the accuracy of the attractor reconstruction and the values of $\mathbf{W}_{\text{out}}$ in Supplementary Note 2; there are many components that have substantial weights and that do not appear in the vector field of Eq. 7, where the vector field is the right-hand-side of the differential equations. This gives quantitative information regarding the difference between the flow and the vector field.

Because the Lyapunov time for the double-scroll system is much longer than for the Lorenz63 system, we extend the training time of the NG-RC from 10 to 100 units to keep the number of Lyapunov times covered during training similar for both cases. To ensure a fair comparison to the Lorenz63 task, we set $dt = 0.25$. With these two changes and the use of the cubic monomials, as given in Eq. 10, with $d = 3$, $k = 2$, and $s = 1$ for a total of 62 features in $\mathbb{O}_{\text{total}}$, the NG-RC uses 400 data points for each variable during training, exactly as in the Lorenz63 task.

Other than these modifications, our method for using the NG-RC to forecast the dynamics of this system proceeds exactly as for

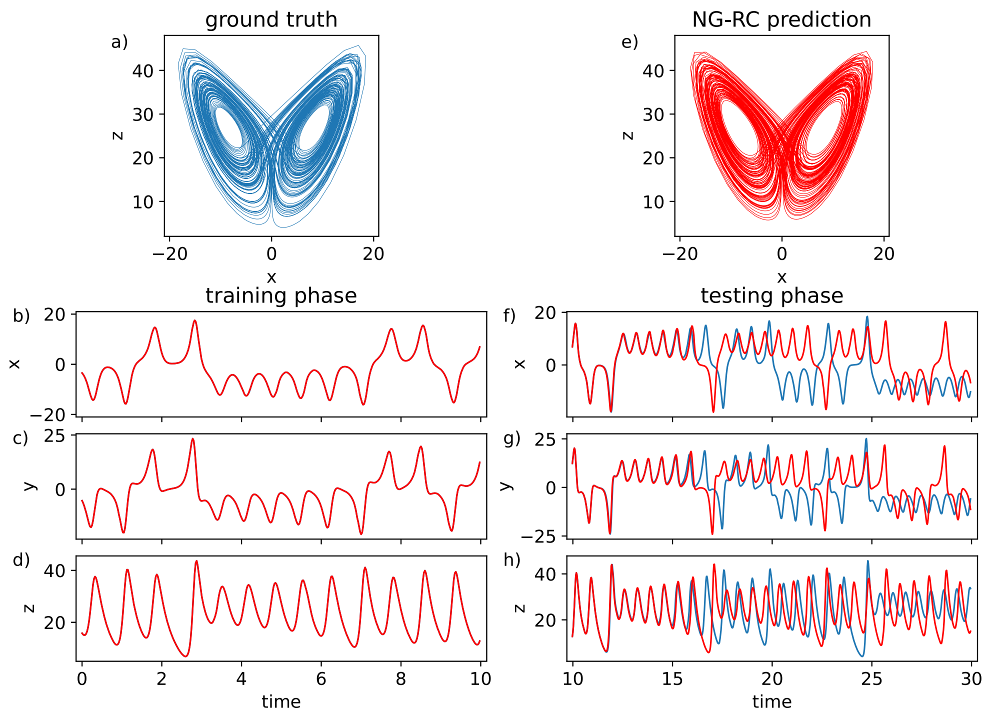

Fig. 2 Forecasting a dynamical system using the NG-RC. True (**a**) and predicted (**e**) Lorenz63 strange attractors. **b**–**d** Training data set with overlayed predicted behavior with $\alpha = 2.5 \times 10^{-6}$. The normalized root-mean-square error (NRMSE) over one Lyapunov time during the training phase is $1.06 \pm 0.01 \times 10^{-4}$, where the uncertainty is the standard error of the mean. **f**–**h** True (blue) and predicted datasets during the forecasting phase (NRMSE $= 2.40 \pm 0.53 \times 10^{-3}$).

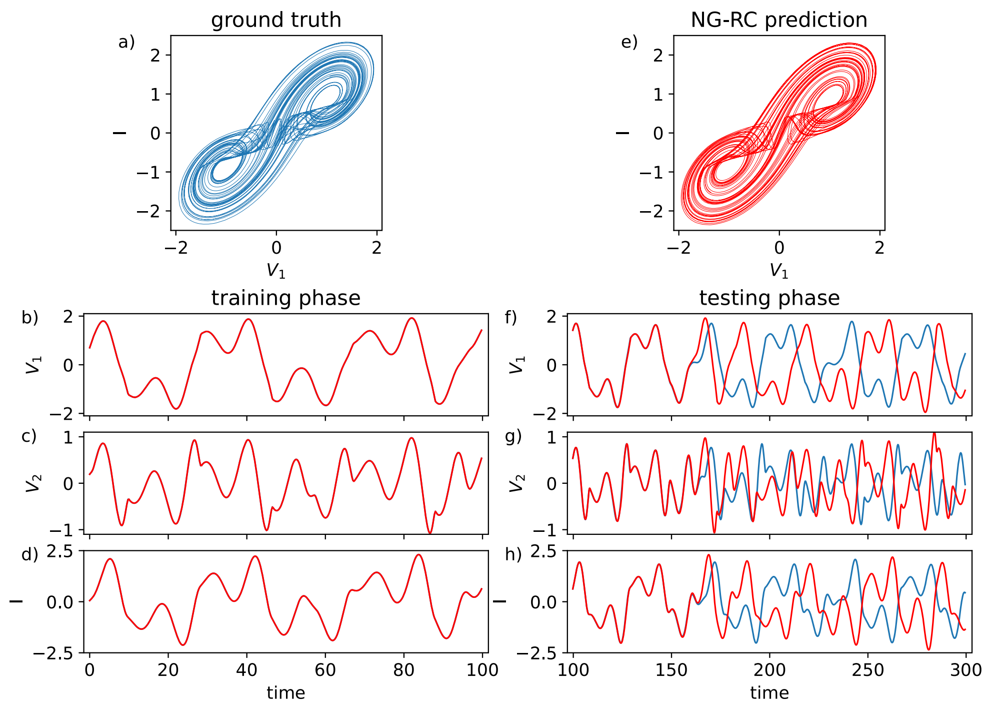

Fig. 3 Forecasting the double-scroll system using the NG-RC. True (**a**) and predicted (**e**) double-scroll strange attractors. **b**–**d** Training data set with overlayed predicted behavior. **f**–**h** True (blue) and predicted datasets during the forecasting phase (NRMSE $= 4.5 \pm 1.0 \times 10^{-3}$).

the Lorenz63 system. The results of this task are displayed in Fig. 3, where it is seen that the NG-RC shows similar predictive ability on the double-scroll system as in the Lorenz63 system, where other quantitative measures of accurate attractor reconstruction is given in Supplementary Note 1 as well as the components of $\mathbf{W}_{\text{out}}$ in Supplementary Note 2.

In the last task, we infer dynamics not seen by the NG-RC during the testing phase. Here, we use $k = 4$ and $s = 5$ with $dt = 0.05$ to generate an embedding of the full attractor to infer the other component, as informed by Takens' embedding theorem[29]. We provide the x, y, and z variables during training and we again observe that a short training data set of only 400

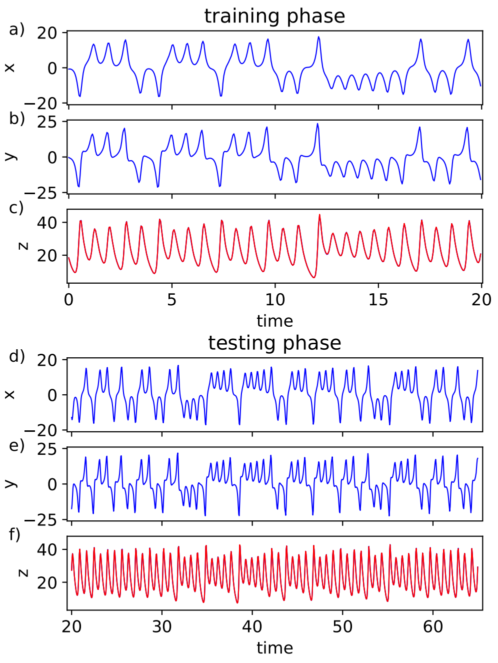

Fig. 4 Inference using an NG-RC. a–c Lorenz63 variables during the training phase (blue) and prediction (**c**, red). The predictions overlay the training data in (**c**), resulting in a purple trace (NRMSE = $9.5 \pm 0.1 \times 10^{-3}$ using $\alpha = 0.05$). **d–f** Lorenz63 variables during the testing phase, where the predictions overlay the training data in (**f**), resulting in a purple trace (NRMSE = $1.75 \pm 0.3 \times 10^{-2}$).

points is enough to obtain good performance as shown in Fig. 4c, where the training data set is overlayed with the NG-RC predictions. Here, the total feature vector has 45 components and hence $\mathbf{W}_{\mathrm{out}}$ has dimension (1×45). During the testing phase, we only provide the NG-RC with the x and y components (Fig. 4d, e) and predict the z component (Fig. 4f). The performance is nearly identical during the testing phase. The components of $\mathbf{W}_{\mathrm{out}}$ for this task are given in Supplementary Note 2.

Discussion

The NG-RC is computationally faster than a traditional RC because the feature vector size is much smaller, meaning there are fewer adjustable parameters that must be determined as discussed in Supplementary Notes 3 and 4. We believe that the training data set size is reduced precisely because there are fewer fit parameters. Also, as mentioned above, the warmup and training time is shorter, thus reducing the computational time. Finally, the NG-RC has fewer metaparameters to optimize, thus avoiding the computational costly optimization procedure in high-dimensional parameter space. As detailed in Supplementary Note 3, we estimate the computational complexity for the Lorenz63 forecasting task and find that the NG-RC is ~33–162 times less costly to simulate than a typical already efficient traditional RC[12], and over 10^6 times less costly for a high-accuracy traditional RC[14] for a single set of metaparameters. For the double-scroll system, where the NG-RC has a cubic nonlinearity and hence more features, the improvement is a more modest factor of 8–41 than a typical efficient traditional RC[12] for a single set of metaparameters.

The NG-RC builds on previous work on nonlinear system identification. It is most closely related to multi-input, multiple-output nonlinear autoregression with exogenous inputs (NARX) studied since the 1980s[21]. A crucial distinction is that Tikhonov regularization is not used in the NARX approach and there is no theoretical underpinning of a NARX to an implicit RC. Our NG-RC fuses the best of the NARX methods with modern regression methods, which is needed to obtain the good performance demonstrated here. We mention that Pyle et al.[30] recently found good performance with a simplified NG-RC but without the theoretical framework and justification presented here.

In other related work, there has been a revival of research on data-driven linearization methods[31] that represent the vector field by projecting onto a finite linear subspace spanned by simple functions, usually monomials. Notably, ref. [25] uses least-square while recent work uses LASSO[26,27] or information-theoretic methods[32] to simplify the model. The goal of these methods is to model the vector field from data, as opposed to the NG-RC developed here that forecasts over finite time steps and thus learns the flow of the dynamical system. In fact, some of the large-probability components of $\mathbf{W}_{\mathrm{out}}$ (Supplementary Note 2) can be motivated by the terms in the vector field but many others are important, demonstrating that the NG-RC-learned flow is different from the vector field.

Some of the components of $\mathbf{W}_{\mathrm{out}}$ are quite small, suggesting that several features can be removed using various methods without hurting the testing error. In the NARX literature[21], it is suggested that a practitioner start with the lowest number of terms in the feature vector and add terms one-by-one, keeping only those terms that reduce substantially the testing error based on an arbitrary cutoff in the observed error reduction. This procedure is tedious and ignores possible correlations in the components. Other theoretically justified approaches include using the LASSO or information-theoretic methods mentioned above. The other approach to reducing the size of the feature space is to use the kernel trick that is the core of ML via support vector machines[20]. This approach will only give a computational advantage when the dimension of $\mathbb{O}_{\mathrm{total}}$ is much greater than the number of training data points, which is not the case in our studies here but may be relevant in other situations. We will explore these approaches in future research.

Our study only considers data generated by noise-free numerical simulations of models. It is precisely the use of regularized regression that makes this approach noise-tolerant: it identifies a model that is the best estimator of the underlying dynamics even with noise or uncertainty. We give results for forecasting the Lorenz63 system when it is strongly driven by noise in the Supplementary Note 5, where we observe that the NG-RC learns the equivalent noise-free system as long as α is increased demonstrating the importance of regularization.

We also only consider low-dimensional dynamical systems, but previous work forecasting complex high-dimensional spatial-temporal dynamics[4,7] using a traditional RC suggests that an NG-RC will excel at this task because of the implicit traditional RC but using smaller datasets and requiring optimizing fewer metaparameters. Furthermore, Pyle et al.[30] successfully forecast the behavior of a multi-scale spatial-temporal system using an approach similar to the NG-RC.

Our work has important implications for learning dynamical systems because there are fewer metaparameters to optimize and the NG-RC only requires extremely short datasets for training. Because the NG-RC has an underlying implicit (hidden) traditional RC, our results generalize to any system for which a standard RC has been applied previously. For example, the NG-RC can be used to create a digital twin for dynamical systems[33] using only observed data or by combining approximate models with observations for data assimilation[34,35]. It can also be used for nonlinear control of dynamical systems[36], which can be quickly adjusted to account for changes in the system, or for speeding up the simulation of turbulence[37].

Methods

The exact numerical results presented here, such as unstable steady states (USSs) and NRMSE, will vary slightly depending on the precise software used to calculate them. We calculate the results for this paper using Python 3.7.9, NumPy 1.20.2, and SciPy 1.6.2 on an x86-64 CPU running Windows 10.

Data availability

The data generated in this study can be recreated by running the publicly available code as described in the Code availability statement.

Code availability

All code is available under an MIT License on Github (https://github.com/quantinfo/ng-rc-paper-code)[38].

Received: 14 June 2021; Accepted: 1 September 2021;
Published online: 21 September 2021

References

1. Jaeger, H. & Haas, H. Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless communication. *Science* **304**, 78–80 (2004).
2. Maass, W., Natschläger, T. & Markram, H. Real-time computing without stable states: a new framework for neural computation based on perturbations. *Neural Comput.* **14**, 2531–2560 (2002).
3. 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* **27**, 121102 (2017).
4. Pathak, J., Hunt, B., Girvan, M., Lu, Z. & Ott, E. Model-free prediction of large spatiotemporally chaotic systems from data: a reservoir computing approach. *Phys. Rev. Lett.* **120**, 24102 (2018).
5. Bengio, Y., Boulanger-Lewandowski, N. & Pascanu, R. Advances in optimizing recurrent networks. *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013, pp. 8624–8628 https://doi.org/10.1109/ICASSP.2013.6639349 (2013).
6. Vogel, C. R. *Computational Methods for Inverse Problems* (Society for Industrial and Applied Mathematics, 2002).
7. Vlachas, P. R. et al. Backpropagation algorithms and reservoir computing in recurrent neural networks for the forecasting of complex spatiotemporal dynamics. *Neural Netw.* **126**, 191–217 (2020).
8. Bompas, S., Georgeot, B. & Guéry-Odelin, D. Accuracy of neural networks for the simulation of chaotic dynamics: precision of training data vs precision of the algorithm. *Chaos* **30**, 113118 (2020).
9. Yperman, J. & Becker, T. Bayesian optimization of hyper-parameters in reservoir computing. Preprint at *arXiv:1611.0519* (2016).
10. Livi, L., Bianchi, F. M. & Alippi, C. Determination of the edge of criticality in echo state networks through fisher information maximization. *IEEE Trans. Neural Netw. Learn. Syst.* **29**, 706–717 (2018).
11. Thiede, L. A. & Parlitz, U. Gradient based hyperparameter optimization in echo state networks. *Neural Netw.* **115**, 23–29 (2019).
12. Griffith, A., Pomerance, A. & Gauthier, D. J. Forecasting chaotic systems with very low connectivity reservoir computers. *Chaos* **29**, 123108 (2019).
13. Antonik, P., Marsal, N., Brunner, D. & Rontani, D. Bayesian optimisation of large-scale photonic reservoir computers. *Cogn. Comput.* **2021**, 1–9 (2021).
14. Lu, Z., Hunt, B. R. & Ott, E. Attractor reconstruction by machine learning. *Chaos* **28**, 061104 (2018).
15. Platt, J. A., Wong, A. S., Clark, R., Penny, S. G. & Abarbanel, H. D. I. Robust forecasting through generalized synchronization in reservoir computing. Preprint at *arXiv:2103.0036* (2021).
16. Gonon, L. & Ortega, J. P. Reservoir computing universality with stochastic inputs. *IEEE Trans. Neural Netw. Learn. Syst.* **31**, 100–112 (2020).
17. Hart, A. G., Hook, J. L. & Dawes, J. H. P. Echo state networks trained by Tikhonov least squares are L2(μ) approximators of ergodic dynamical systems. *Phys. D. Nonlinear Phenom.* **421**, 132882 (2021).
18. 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. *Chaos* **31**, 013108 (2021).
19. Gauthier, D. J. Reservoir computing: harnessing a universal dynamical system. *SIAM News* **51**, 12 (2018).
20. Franz, M. O. & Schölkopf, B. A unifying view of Wiener and Volterra theory and polynomial kernel regression. *Neural. Comput.* **18**, 3097–3118 (2006).
21. Billings, S. A. *Nonlinear System Identification* (John Wiley & Sons, Ltd., 2013).
22. Kim, J. Z., Lu, Z., Nozari, E., Papas, G. J. & Bassett, D. S. Teaching recurrent neural networks to infer global temporal structure from local examples. *Nat. Mach. Intell.* **3**, 316–323 (2021).
23. Lorenz, E. N. Deterministic nonperiodic flow. *J. Atmos. Sci.* **20**, 130–141 (1963).
24. Chang, A., Bienfang, J. C., Hall, G. M., Gardner, J. R. & Gauthier, D. J. Stabilizing unstable steady states using extended time-delay autosynchronization. *Chaos* **8**, 782–790 (1998).
25. Crutchfield, J. P. & McNamara, B. S. Equations of motion from a data series. *Complex Sys.* **1**, 417–452 (1987).
26. Wang, W.-X., Lai, Y.-C., Grebogi, C. & Ye, J. Network reconstruction based on evolutionary-game data via compressive sensing. *Phys. Rev. X* **1**, 021021 (2011).
27. Brunton, S. L., Proctor, J. L., Kutz, J. N. & Bialek, W. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proc. Natl Acad. Sci. USA* **113**, 3932–3937 (2016).
28. Lai, Y.-C. Finding nonlinear system equations and complex network structures from data: a sparse optimization approach. Chaos **31**, 082101 (2021).
29. Takens, F. Detecting strange attractors in turbulence. In *Dynamical Systems and Turbulence, Warwick 1980* (eds Rand, D. & Young, L. S.) 366–381 (Springer, 1981).
30. Pyle, R., Jovanovic, N., Subramanian, D., Palem, K. V. & Patel, A. B. Domain-driven models yield better predictions at lower cost than reservoir computers in Lorenz systems. *Philos. Trans. R. Soc. A Math. Phys. Eng. Sci.* **379**, 24102 (2021).
31. Carleman, T. Application de la théorie des équations intégrales linéares aux d'équations différentielles non linéares. *Acta Math.* **59**, 63–87 (1932).
32. Almomani, A. A. R., Sun, J. & Bollt, E. How entropic regression beats the outliers problem in nonlinear system identification. *Chaos* **30**, 013107 (2020).
33. Grieves, M. W. Virtually Intelligent Product Systems: Digital and Physical Twins. In *Complex Systems Engineering: Theory and Practice* (eds Flumerfelt, S., et al.) 175–200 (American Institute of Aeronautics and Astronautics, Inc., 2019).
34. Wikner, A. et al. Combining machine learning with knowledge-based modeling for scalable forecasting and subgrid-scale closure of large, complex, spatiotemporal systems. *Chaos* **30**, 053111 (2020).
35. Wikner, A. et al. Using data assimilation to train a hybrid forecast system that combines machine-learning and knowledge-based components. *Chaos* **31**, 053114 (2021).
36. Canaday, D., Pomerance, A. & Gauthier, D. J. Model-free control of dynamical systems with deep reservoir computing. to appear in *J. Phys. Complex.* http://iopscience.iop.org/article/10.1088/2632-072X/ac24f3 (2021).
37. Li, Z. et al. Fourier neural operator for parametric partial differential equations. Preprint at *arXiv*:2010.08895 (2020). In The International Conference on Learning Representations (ICLR 2021).
38. Gauthier, D. J., Griffith, A. & de sa Barbosa, W. ng-rc-paper-code repository. https://doi.org/10.5281/zenodo.5218954 (2021).

Acknowledgements

We gratefully acknowledge discussions with Henry Abarbanel, Ingo Fischer, and Kathy Lüdge. D.J.G. is supported by the United States Air Force AFRL/SBRK under Contract No. FA864921P0087. E.B. is supported by the ARO (N68164-EG) and DARPA.

Author contributions

D.J.G. optimized the NG-RC, performed the simulations in the main text, and drafted the manuscript. E.B. conceptualized the connection between an RC and NVAR, helped interpret the data and edited the manuscript. A.G. and W.A.S.B. helped interpret the data and edited the manuscript.

Competing interests

D.J.G. has financial interests as a cofounder of ResCon Technologies, LCC, which is commercializing RCs. The remaining authors declare no competing interests.

Additional information

Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s41467-021-25801-2.

Correspondence and requests for materials should be addressed to Daniel J. Gauthier.

Peer review information *Nature Communications* thanks Serhiy Yanchuk and the other anonymous reviewer(s) for their contribution to the peer review of this work. Peer reviewer reports are available.