Product expansion of reservoir states for robust chaotic time series prediction

Kaiwen Jiang,[1] Michael Small,[2,3] and Yong Zou[1,*]
[1]*School of Physics, East China Normal University, Shanghai 200241, China*
[2]*The Complex Systems Group, Department of Mathematics and Statistics, University of Western Australia, Crawley, Australia*
[3]*UWA Data Institute, University of Western Australia, Crawley, Australia*

(Received 1 July 2026; accepted 21 October 2026; published 6 November 2026)

Reservoir computing (RC) excels at chaotic time series prediction but suffers from high sensitivity to hyperparameters (spectral radius, sparsity) and requires large reservoirs, limiting its utility. To overcome these limitations, we propose an enhanced RC framework that introduces time-shift transformations on the reservoir states to embed temporal information, followed by explicit Volterra-style nonlinear feature expansions—quadratic polynomial expansion (QT-RC) or hierarchical product expansion (HP-RC). These methods capture multiplicative nonlinearities and historical information while preserving the efficiency of linear ridge regression. Using the Lorenz and Rössler systems as benchmarks, we show that our methods achieve prediction accuracy comparable to traditional RC with hundreds of nodes while using only five to 10 reservoir nodes, and drastically reduce sensitivity to spectral radius and connectivity. Phase diagram analysis reveals complementary roles: nonlinear transformations enhance expressiveness and reduce spectral radius dependence, whereas time-shift transformations improve robustness under dense connectivity. This work offers a pathway toward efficient, robust, and hardware-friendly reservoir computing, grounded in Takens' embedding theorem and Volterra series theory, which provides a universal framework for nonlinear systems with fading memory.

DOI: 10.1103/v6gn-wncs

I. INTRODUCTION

Reservoir computing (RC) is a computational framework suited for temporal data processing and is a type of recurrent neural network model [1–3]. An RC system consists of a reservoir that maps input signals into a high-dimensional space and a readout layer that combines these high-dimensional reservoir states [2]. The crucial point of this framework is that only the readout connections are trained; the internal connections of the reservoir are sparse, their weights are randomly chosen initially and then kept fixed, which yields fast learning and low training cost [4]. As a result, RC has been successfully used to predict time series of various low-dimensional and spatiotemporal systems [5], including chaotic dynamics [6,7], laser pulses [8], epidemic spreading [9], the El Niño-Southern Oscillation [10], global atmospheric circulations [11], tipping points [12–14], and extreme events [15,16]. More recently, RC has expanded into nonlinear dynamical systems theory, where large-scale collections of chaotic time series from continuous systems have been proposed to benchmark machine learning methods [17,18]. By design, RC is powerful for time series prediction because the reservoir encodes temporal information; the prediction horizon achieved by most RC algorithms varies drastically, ranging from several Lyapunov times [5,19] to, in some cases, arbitrarily long scales [20,21]. Despite these successful applications to a wide array of dynamical systems, very few theoretical explanations have been proposed toward a theory of reservoir computing [2,22].

Despite the proven effectiveness of RC in modeling complex dynamical systems, its practical deployment is often hindered by a fundamental issue: prediction instability [3,23]. Traditional RC models exhibit high sensitivity to the choice of hyperparameters, such as the spectral radius and sparsity. Consequently, two nominally identical RC architectures trained on the same data can yield substantially different predictive performances—one achieving accurate long-term forecasts while another fails rapidly. Improving prediction stability is therefore not merely a technical refinement but a necessary condition for transitioning RC from a powerful theoretical tool to a trustworthy and deployable solution in real-world, high-stakes environments. Moreover, stable RC models reduce the need for expensive hyperparameter tuning and ensemble averaging, making them more efficient and accessible for resource-constrained applications.

The output of reservoir computing is often a function of the reservoir's internal states. These states represent the accumulated information from the input data, processed through the nonlinear dynamics of the reservoir. One of the common approaches is to take a linear combination of reservoir states, which is represented by $\mathbf{W}_{\text{out}}$. This involves multiplying each state by a corresponding weight and summing the results. The weights of the linear combination are learned through training to optimize the output for a specific task, i.e., time series prediction. The beauty of $\mathbf{W}_{\text{out}}$ is that it makes training incredibly cheap since we typically use ridge regression.

A key limitation of traditional RC is its reliance on large, high-dimensional reservoirs, which poses challenges for hard-

*Contact author: yzou@phy.ecnu.edu.cn

 ©2026 American Physical Society

ware implementation, training stability, and hyperparameter sensitivity. A powerful strategy to address this limitation is the deliberate introduction of time delays into the RC architecture. It has been demonstrated that a single nonlinear node with delayed feedback could create a rich set of "virtual nodes," achieving performance comparable to hundreds of physical neurons [24]. This concept was later extended to the photonic domain [25], which achieved ultrafast speech recognition at rates of one million words per second using an electro-optic phase-delay system. Marquez *et al*. [26] demonstrated that a small reservoir, when augmented with a fixed delay line set to the Takens' embedding delay, could match the performance of a much larger traditional RC. Therefore, time delays provide a form of temporal expansion, effectively projecting the dynamics of a small network's output into a high-dimensional feature space without increasing the physical network size. From a theoretical perspective, Duan *et al*. [27] provided a rigorous foundation for these empirical successes, proving that the reservoir states form a topological embedding of the input dynamics and establishing a trade-off between the number of physical neurons and the number of time delays. Subsequent work has shown that time shifts can also be applied directly to the readout layer: Del Frate *et al*. [28] demonstrated that applying different time shifts to individual node readouts breaks undesired synchronization among neurons, dramatically reducing prediction error. Carroll and Hart further advanced this line of research by showing that a small nonlinear core combined with time-shifted output can match the performance of much larger reservoirs [29]. These references establish time delay as a fundamental design principle that can create high-dimensional dynamics from minimal hardware, reducing the physical reservoir size.

While Takens' theorem provides a rigorous justification for using time-delayed copies of a single observable to reconstruct the attractor of a dynamical system [30], the Volterra series, in turn, offers a universal framework for capturing higher-order nonlinearities in nonlinear systems with fading memory as an infinite sum of polynomial functionals of its input history [31]. Note that time-delay embedding alone is a linear operation, and it concatenates delayed copies without introducing new nonlinear interactions. In contrast, a Volterra series expansion explicitly constructs the higher-order terms (quadratic, cubic, etc.) needed to capture the multiplicative couplings that are ubiquitous in chaotic systems.

Building on these insights, we propose an enhanced RC framework that integrates time-shift concatenation with explicit nonlinear feature expansion. Specifically, we introduce quadratic transformed reservoir computing (QT-RC), which applies a second-order Volterra kernel to the time-delayed reservoir states, and hierarchical product reservoir computing (HP-RC), which extends this to higher-order and multiscale interactions. Our methods operate on the reservoir states $\mathbf{r}(t)$ rather than the raw input, allowing the model to capture nonlinearities that are already embedded in the reservoir's dynamics. This design preserves the computational efficiency of linear ridge regression while endowing low-dimensional reservoirs (e.g., five to 10 nodes) with the expressive power required for accurate long-term chaotic prediction. Furthermore, it dramatically reduces the sensitivity to hyperparameters such as the spectral radius, offering a compelling path toward robust, interpretable, and hardware-friendly reservoir computing.

The remainder of this paper is organized as follows: Section II presents the basic settings of the reservoir computing model, in particular introducing delayed embedding for the reservoir states and two ways of nonlinear transformations. The results are shown in Sec. V. The advantages will be shown in Sec. VI by comparing our methods with other methods. The conclusions are provided in Sec. VII.

II. SETTINGS FOR RESERVOIR COMPUTING

Reservoir computing (RC) is a simplified version of a recurrent neural network often used for time series prediction. There are three layers: an input layer $\mathbf{W}_{in}$, a fixed random recurrent hidden layer (the reservoir) $\mathbf{W}_r$, and a trainable output layer $\mathbf{W}_{\text{out}}$.

The hidden layer of the reservoir consists of M network nodes that are connected by an $M \times M$ weighted adjacency matrix $\mathbf{W}_r$. There are two hyperparameters that must be properly chosen for $\mathbf{W}_r$, namely, the average connectivity d and the spectral radius ρ. Both $\mathbf{W}_{in}$ and $\mathbf{W}_r$ are initially randomly chosen and then fixed in the training process.

The reservoir states are represented by the vector $\mathbf{r}(t) \in \mathbb{R}^M$, i.e., $\mathbf{r}(t) = [r_1(t), \ldots r_n(t), \ldots, r_M(t)]^\top$, where $r_n(t), n \in [1, M], t \in [0, l]$ represents the state of the n-th node in the reservoir at time t. In addition, the state updating rule of the reservoir is expressed by

$$\mathbf{r}(t+1) = (1-\alpha)\mathbf{r}(t) + \alpha \tanh[\mathbf{W}_r \mathbf{r}(t) + \mathbf{W}_{in}\mathbf{u}(t)], \quad (1)$$

where $\alpha \in (0, 1)$ is the reservoir leakage rate and $\mathbf{u}(t)$ is the input signal.

Only the output weights $\mathbf{W}_{\text{out}}$ are trained using ridge regression, minimizing the error between the target $\mathbf{v}_g(t)$ and the output $\mathbf{v}(t) = \mathbf{W}_{\text{out}}\mathbf{r}(t)$. For example, the estimation of $\mathbf{W}_{\text{out}}$ is achieved by minimizing the following loss function:

$$\text{Loss} = \sum \|\mathbf{v}_g(t) - \mathbf{W}_{\text{out}}\mathbf{r}(t)\| + \beta \|\mathbf{W}_{\text{out}}\|. \quad (2)$$

For autonomous prediction, the output $\mathbf{v}(t)$ is fed back as the input, forming a closed-loop system. This framework enables efficient and stable learning of complex temporal dynamics.

A. Time-delayed coordinates of the reservoir

In order to improve the nonlinear property of the reservoir, several transformations of $\mathbf{r}(t)$ are introduced before feeding $\mathbf{r}(t)$ to the output layer, allowing the reservoir computer to learn more complex patterns. For example, Pathak proposed in Ref. [5] to square half of the node states in the reservoir $\mathbf{r}(t)$, i.e., $\tilde{r}_j = r_j^2$ for odd terms while leaving even terms unchanged $\tilde{r}_j = r_j$. This transformation is now commonly performed in the literature, leading to better performance in the regression task in the output layer. It is certain that different nonlinear transformations explore different types of relationships in the data. Other options, like polynomial libraries, have recently been proposed in Ref. [32] to transform the state space $\mathbf{r}(t)$ into an ensemble of state spaces that contain different higher-order terms of $\mathbf{r}(t)$, i.e., $f(\mathbf{r}(t)) =$

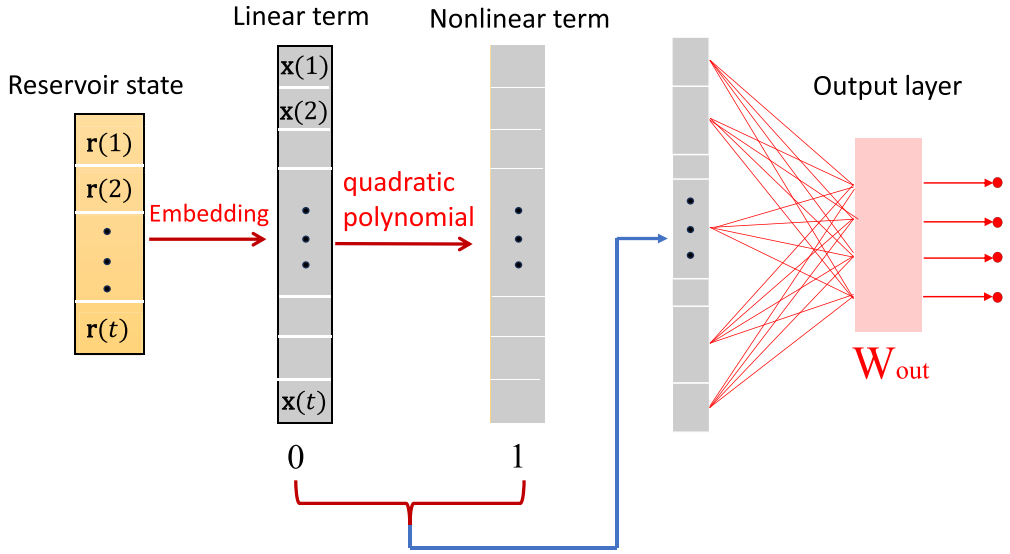


FIG. 1. Schematic of quadratic transformed reservoir computing (QT-RC). The reservoir state $\mathbf{r}(t)$ is first embedded into a higher-dimensional space via time delays, resulting in $\mathbf{x}_0(t)$ (Layer 0). A quadratic polynomial expansion is then applied to $\mathbf{x}_0(t)$ to generate nonlinear features (Layer 1). Both the linear and quadratic layers are concatenated and fed into a linear readout $\mathbf{W}_{\text{out}}$ trained via ridge regression. This design preserves convexity while enhancing the model's expressive power.

$\text{vec}(\mathbf{r}(t), \mathbf{r}^2(t), \mathbf{r}^3(t), \ldots, \mathbf{r}^n(t))$. This polynomial library of $f(\mathbf{r}(t))$ to the reservoir's state vectors has been shown to capture more intricate patterns and dependencies in input data [32]. In both cases, we still use ridge regression to obtain $\mathbf{W}_{\text{out}}$ in a linear manner, but the mapping itself contains nonlinear information.

We first introduce a time-shifted transformation to the reservoir's states by time delay embedding, which further encompasses a nonlinear transformation. More specifically, motivated by Takens' embedding theorem in dynamical system theory, the reservoir's states in the embedded space are

$$\mathbf{x}(t) = [\mathbf{r}(t); \mathbf{r}(t-\tau); \ldots; \mathbf{r}(t-(m-1)\tau)], \tag{3}$$

where τ and m are the embedding delay and dimension, respectively. There are $L = l - (m-1)\tau$ vectors in the embedded space. The delay τ is chosen by the decorrelation time of the input time series, i.e., the first time when the autocorrelation function drops to $1/\tau_e$. The number of delays m is called the embedding dimension, which is often guided by the false nearest neighbors. The computational complexity of the reservoir is greatly increased for large values of m. We show below numerically that $m = 3$ is generally sufficient for most oscillatory time series.

Note that this step expands the feature space linearly, but does not introduce multiplicative interactions among the components. To capture nonlinear couplings, we draw on the Volterra series [31], which represents any nonlinear system with fading memory as a sum of polynomial functionals of its input history. We apply this principle to construct explicit Volterra-style features from $\mathbf{x}(t)$. Specifically, we propose the following two complementary approaches, both of which can be mathematically viewed as a localized approximation of a Volterra filter series [31], capturing higher-order cross-correlations without sacrificing linear readout convexity.

B. Quadratic transformed reservoir computing

Next, we define a transformation function $f(\mathbf{x})$, which is schematically shown in Fig. 1.We use the outer product $\mathbf{x} \otimes \mathbf{x}$ to obtain all quadratic polynomials of an L-dimensional vectors $\mathbf{x}$, resulting in a symmetric matrix of $L \times L$ elements. We further define an operator $\hat{\otimes}$ that collects unique monomials within the symmetric matrix. Using this notation, all p-order polynomial transformations are expressed as

$$f^{(p)}(\mathbf{x}) = \mathbf{x}\hat{\otimes}\mathbf{x}\hat{\otimes}\ldots\hat{\otimes}\mathbf{x}. \tag{4}$$

In this work, we mainly consider the quadratic operation $p = 2$.

The above two steps result in a time-shifted reservoir computer with quadratic transformed coordinates, which are denoted as

$$\mathbf{r}_{qt}(t) = \Psi(\mathbf{r}(t)). \tag{5}$$

We show below that quadratic nonlinearity is sufficient to improve the time series prediction task.

Unlike the traditional way of a linear output as $\mathbf{W}_{\text{out}}\mathbf{v}(t)$, we introduce the output layer that transforms the high-dimensional time-shifted reservoir's states $\mathbf{r}_{qt}(t)$ into the output signal $\mathbf{v}(t)$ via $\mathbf{W}_{\text{out}}$ as

$$\mathbf{v}(t) = \mathbf{W}_{\text{out}}\mathbf{r}_{qt}(t). \tag{6}$$

Then, the reservoir state $\mathbf{r}(t)$ in the remaining steps [Eqs. (1)–(6)] is replaced by $\mathbf{r}_{qt}(t)$. We denote the above algorithm as quadratic transformed reservoir computing (QT-RC) in the following.

C. Hierarchical product reservoir computing

We show that a hierarchical product expansion in time-delayed coordinates [Eqs. (3)] further improves the prediction performance of RC, which is denoted as HP-RC in the

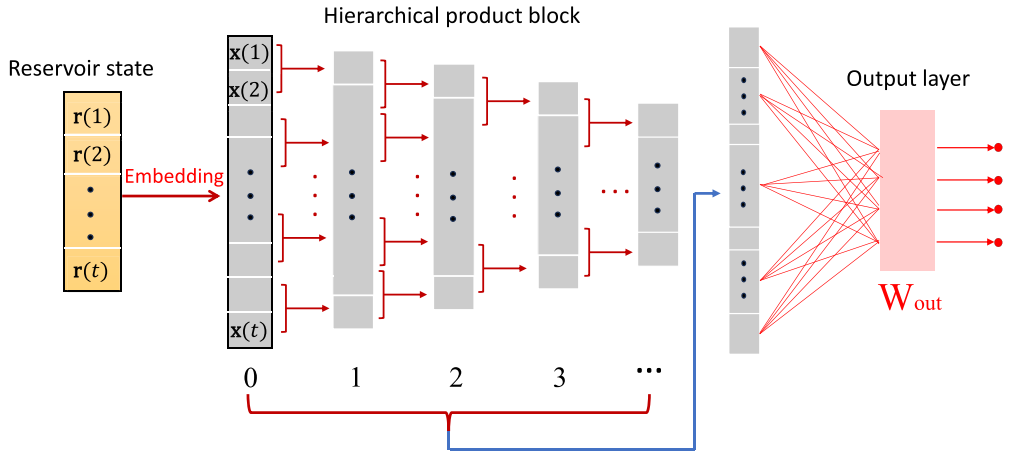


FIG. 2. Schematic of hierarchical product reservoir computing (HP-RC). Reservoir states $\mathbf{r}(t)$ are time-shifted and concatenated to form the zero-layer embedding $\mathbf{x}_0(t)$. Subsequent layers apply element-wise products of selected elements from the previous layer, producing higher-order and multiscale features. In the example, $p = 2$ and $\Delta = 1$, namely, $\mathbf{x}_{i-1}^{j}(t) \odot \mathbf{x}_{i-1}^{j+1}(t)$. The concatenated feature vector is fed into a linear readout $\mathbf{W}_{\text{out}}$ trained via ridge regression.

following. More specifically, some steps are the following, as shown in Fig. 2:

(i) Denote the layer without product operation as the zero-layer $\mathbf{x}_0(t)$. We first explicitly define the zero-layer vector as follows:

$$\mathbf{x}_0(t) = [r_1(t), \ldots, r_M(t),\ r_1(t-\tau), \ldots, r_M(t-\tau), \times \ldots,\ r_1(t-(m-1)\tau), \ldots, r_M(t-(m-1)\tau)]^{\top}. \tag{7}$$

That is, the component j of $\mathbf{x}_0(t)$ with $j = (q-1)M + n$ corresponds to the reservoir node n and the delay level $q-1$ ($n = 1, \ldots, M$, $q = 1, \ldots, m$). This ordering is a convention and any consistent ordering is equally valid.

(ii) Each subsequent product layer i is obtained by taking the element-wise Hadamard product of selected components from the previous layer $i-1$:

$$\mathbf{x}_i^j(t) = \prod_{k=0}^{p-1} \mathbf{x}_{i-1}^{j+k\Delta}(t), \tag{8}$$

where p is the order of the product (e.g., $p = 2$ for quadratic and $p = 3$ for cubic), Δ is the step size in the feature index, and j indexes the starting position in the ordered feature vector.

With the ordering defined above, $j \mapsto j+1$ moves to the next component (the next node within the same delay block, or the first node of the next delay block at block boundaries). Thus, $\Delta = 1$ combines adjacent components, that is, mostly different nodes within the same delay block, or cross-delay pairs at block boundaries. In contrast, $\Delta = M$ leads to products of the same reservoir node at neighboring delays, i.e., a purely temporal product. Therefore, this formulation of Eq. (8) allows for flexible construction of higher-order, multiscale interactions across feature vectors. Unlike temporal convolutions with learned kernels, our product expansion is deterministic and fixed, preserving convexity and enabling efficient ridge regression.

There are three algorithmic parameters in Eq. (8): (1) the number of product layers i, (2) the step size of the feature index Δ, and (3) the order of the product p. For the base case $p = 2$ and $\Delta = 1$, the product is taken between adjacent components in the feature vector:

$$\mathbf{x}_i^j(t) = \mathbf{x}_{i-1}^j(t) \odot \mathbf{x}_{i-1}^{j+1}(t), \tag{9}$$

where $\odot$ denotes the element-wise Hadamard product. More generally, when $p > 2$ or $\Delta > 1$, the product can incorporate multiple components with variable intervals. For example, the product

$$\mathbf{x}_i^j(t) = \mathbf{x}_{i-1}^j(t) \odot \mathbf{x}_{i-1}^{j+2}(t) \odot \mathbf{x}_{i-1}^{j+4}(t), \tag{10}$$

produces a cubic product ($p = 3$) with feature index step $\Delta = 2$. In most of our numerical experiments, we choose $i = 3$ layers, $p = 2$ (quadratic), and $\Delta = 1$ (adjacent components). The effects of varying the number of layers and the choice of Δ and p are discussed in the following sections.

(iii) The final reservoir states are obtained by concatenating all product layers:

$$\mathbf{x}_{\text{total}}(t) = [\mathbf{x}_0(t); \mathbf{x}_1(t); \ldots; \mathbf{x}_s(t)], \tag{11}$$

where s denotes the total number of hierarchical layers.

(iv) Then, we apply the standard linear readout:

$$\mathbf{v}(t) = \mathbf{W}_{\text{out}}\, \mathbf{x}_{\text{total}}(t). \tag{12}$$

HP-RC can be regarded as a structured, hierarchical nonlinear transformation method. It is intrinsically related to polynomial transformations, but may capture different patterns of feature interaction. Note that the hierarchical product operation is performed for the embedded reservoir state $\mathbf{x}(t)$, instead of the raw input signals as used in Ref. [33].

III. VALID PREDICTION TIME

In order to compare the prediction skills of the nonlinear time-delayed RC with the traditional RC, we introduce the

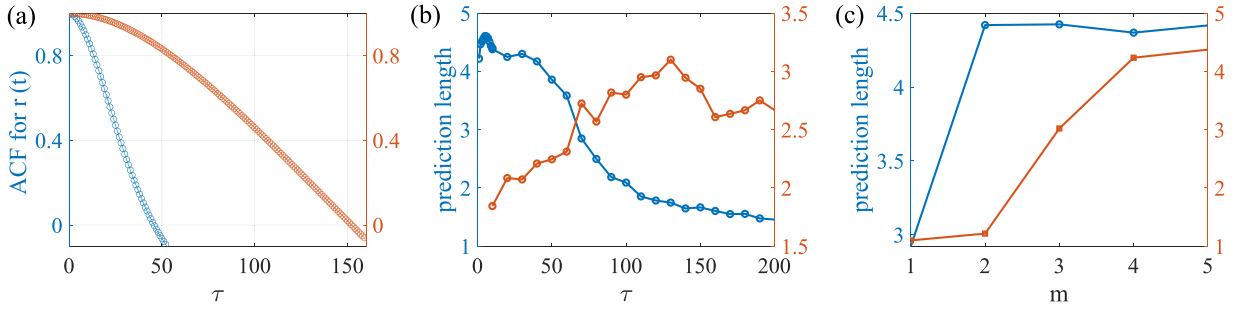


FIG. 3. Determination of embedding parameters for $\mathbf{x}_0(t)$. (a) Autocorrelation function of the reservoir state $\mathbf{r}(t)$; the first minimum (decorrelation time) is used to select the embedding delay τ (Lorenz: $\tau = 30$; Rössler $\tau = 120$). (b) Valid prediction time t_p as a function of τ, showing optimal performance near the decorrelation time. (c) t_p as a function of the embedding dimension m; $m = 3$ (Lorenz) and $m = 4$ (Rössler) are chosen as they yield the longest predictions.

following valid effective prediction time:

$$t_p = \min\left\{t : \frac{\|\mathbf{v}_p(t) - \mathbf{v}_g(t)\|}{\|\mathbf{v}_g(t)\|} > \varepsilon\right\}, \tag{13}$$

where $\|\cdot\|$ denotes the L_2 norm, $\mathbf{v}_p(t)$ and $\mathbf{v}_g(t)$ are the predicted and ground truth time series, respectively. The effective predictable time describes the time after which the prediction differs substantially from the true time series, which is based on the root mean square error (RMSE), which measures the distance between the predicting signal and the ground truth time series. Throughout this work, the error tolerance ε is chosen as 0.5.

Note that the effective predictable time is correlated with the largest Lyapunov exponent and that we normalize it in terms of the Lyapunov time.

Considering that both the spectral radius ρ and the average connectivity density d of matrix $\mathbf{W}_r$ play the dominant roles in determining the prediction skills of RC, we present a phase diagram in the plane (ρ, d). This diagram is color-coded to represent the valid prediction time, while ρ and d are varied within empirically observed intervals.

IV. MODEL TIME SERIES

We show the prediction capabilities of our methods for both chaotic time series from Lorenz and Rössler systems [34], which contain all three-dimensional ordinary differential equations with quadratic nonlinearities. In particular, we consider the following two cases:

(1) Lorenz system:

$$\begin{aligned}\dot{x} &= 10(y - x),\\ \dot{y} &= x(28 - z) - y,\\ \dot{z} &= xy - 8/3z.\end{aligned} \tag{14}$$

(2) Rössler system:

$$\begin{aligned}\dot{x} &= -(y + z),\\ \dot{y} &= x + 0.2y,\\ \dot{z} &= 0.2 + (x - 5.7)z.\end{aligned} \tag{15}$$

These chaotic systems are numerically integrated by a fourth-order Runge-Kutta integrator, and the time step for the integrator is $\Delta t = 0.01$. We use all the (x, y, z) components of the system as the input signal. For the Lorenz system, the training and warm-up lengths are 5000 and 500 steps, respectively; for the Rössler system, they are 10 000 and 1000 steps. All results in the following are averaged over 20 independent random realizations.

The choice of τ and m: Following Takens' embedding theorem [30,35], the delay τ in Eq. (3) is chosen by the decorrelation time of the input time series, i.e., the first time when the autocorrelation function drops to $1/\tau_e$. Therefore, we choose $\tau = 30$ and $\tau = 120$ for the Lorenz and Rössler cases, respectively, as guided by Fig. 3(a) showing the decorrelation time of the reservoirs' state $\mathbf{r}(t)$. Figure 3(b) shows that the mean prediction time reaches the maximum at $\tau = 30$ for the Lorenz case, while $\tau = 120$ for the Rössler case, suggesting a reasonable choice for τ. Concerning the embedding dimension m, we choose $m = 3$ for the Lorenz case, while $m = 4$ for the Rössler case, which provides the maximal mean prediction time, as shown in Fig. 3(c).

V. RESULTS

A. Nonlinear transformations increase effective prediction length

Compared to the results by the traditional RC [in Fig. 4(a)], we first show that the nonlinear transformations to the delayed coordinates of the RC states significantly improve the effective prediction length, as shown in Figs. 4(b) and 4(c). More specifically, Fig. 4(a) shows that the traditional RC leads to a valid prediction length of 5–6 Lyapunov times, while our new

TABLE I. Choices for hyperparameters for both QT-RC and HP-RC in the single-prediction task (Fig. 4). The embedding parameters are $m = 3$, $\tau = 30$ for the Lorenz and $m = 4$ and $\tau = 120$ for the Rössler system, as summarized in Fig. 3.

Hyperparameter	Description	QT-RC	HP-RC
σ	connection strength of $\mathbf{W}_{in}$	1	1
ρ	spectral radius of $\mathbf{W}_r$	0.2	−0.48
d	average connectivity of $\mathbf{W}_r$	3	10
α	leakage rate	0.1	0.1
β	Tikhonov regularization parameter	10^{-6}	10^{-6}

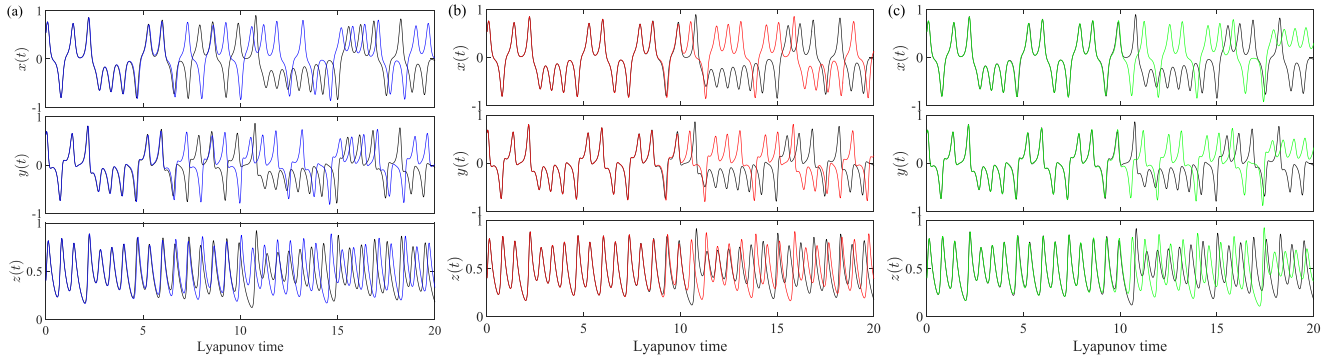


FIG. 4. Exemplary time series prediction for the Lorenz system using (a) traditional RC, (b) QT-RC, and (c) HP-RC. The proposed methods achieve significantly longer valid prediction times (10–12 Lyapunov times) compared to traditional RC (5–6 Lyapunov times). The reservoir dimensions are $M = 100$ for traditional RC and HP-RC, and $M = 10$ for QT-RC. The hyperparameters used are listed in Table I.

methods (both quadratic transformed and hierarchical product RC) yield 10–12 Lyapunov times [Figs. 4(b) and 4(c)].

For the single prediction task as shown in Fig. 4, the hyperparameters for both QT-RC and HP-RC methods are summarized in Table I. These values are obtained by the Bayesian optimization algorithm that has been widely used in the literature [36,37], and are not optimized for each individual case. The size of the reservoir is explicitly given below.In addition, three parameters are fixed throughout all experiments: $\sigma = 1$, $\alpha = 0.1$, and $\beta = 10^{-6}$. The spectral radius ρ and the average connectivity d are chosen from the phase diagrams (Fig. 5) as the grid points that maximize the mean valid prediction time averaged over independent reservoir realizations. The embedding parameters m and τ are determined by autocorrelation and false nearest neighbor analyses (Fig. 3). Note that our results do not depend on fine-tuning at a single optimal point, and the proposed methods maintain high performance over a wide range of ρ and d (Figs. 5 and 12).

B. Much freedom in choosing hyperparameters

The phase diagram in (d, ρ) for the traditional RC is shown in Fig. 5(a), which is color-coded by the valid prediction time t_p. In particular, for each pair (d, ρ), we compute the mean prediction length by the average value of t_p over 20 independent runs. It can be observed that the performance of traditional RC is largely dependent on the hyperparameters: it performs well only within a narrow region where ρ is close to $\rho = \pm 1$ and d is small (sparse connectivity). When the absolute value of ρ is less than 1, the prediction performance drops dramatically. This aligns with the theoretical expectation of traditional RC: when the spectral radius is close to 1, the reservoir possesses moderate memory capacity and dynamical stability.

In contrast to Fig. 5(a), the quadratic transformed RC (QT-RC) based on delayed coordinates with only a reservoir of $M = 10$ already exhibits considerable predictive capability across most of the parameter region [Fig. 5(b)]. Specifically, QT-RC maintains a high and stable t_p throughout the parameter space, including the region where $|\rho| < 1$ and the region with large d. This indicates that QT-RC not only significantly reduces the requirement for reservoir dimensionality, but also substantially enhances the robustness to reservoir topological parameters. Qualitatively similar results are obtained by hierarchical product RC (HP-RC) as shown in Fig. 5(c).

We perform a similar comparison on the Rössler system. Figure 6 shows the phase diagrams of (d, ρ) color-coded

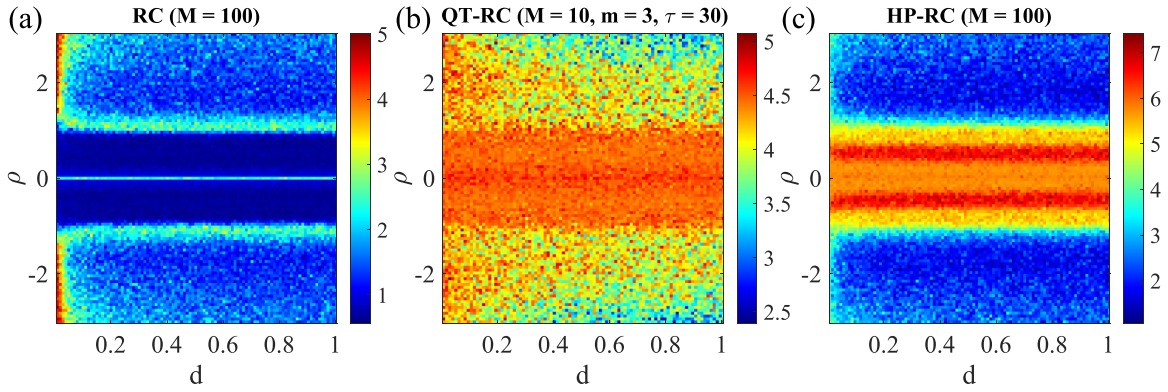


FIG. 5. Phase diagram (d, ρ) for the Lorenz system, color-coded by the mean valid prediction time (in Lyapunov times). (a) traditional RC with $M = 100$: high performance is confined to a narrow region near $\rho = \pm 1$ and low d, (b) quadratic transformed RC $M = 10$, and (c) hierarchical product RC $M = 100$ both achieve high t_p across the entire parameter space, including $|\rho| < 1$ and dense connectivity (d large).

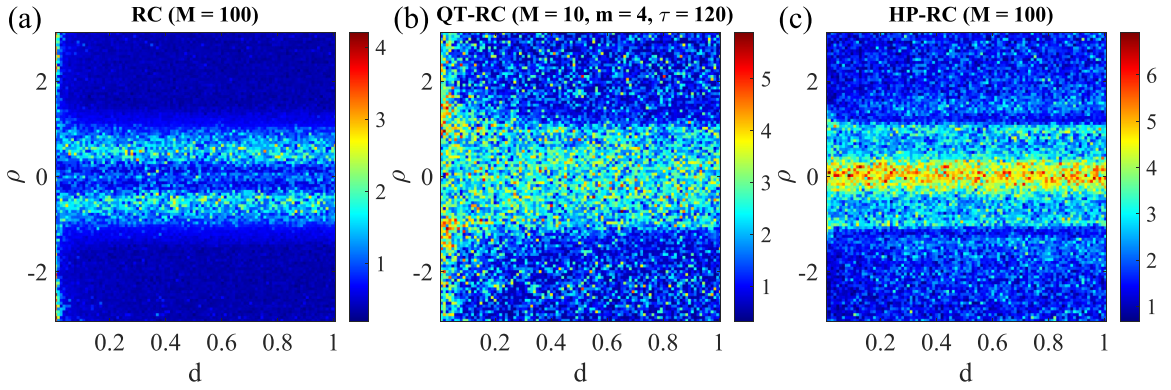


FIG. 6. Phase diagram (d, ρ) for the Rössler system, color-coded by the mean prediction time (in Lyapunov times). (a) traditional RC with $M = 100$ again shows a narrow high-performance region, (b) quadratic transformed RC $M = 10$, and (c) hierarchical product RC $M = 100$ maintain robust performance over wide parameter ranges.

by t_p, which are, respectively, by RC ($M = 100$), QT-RC ($M = 10$), and HP-RC ($M = 100$), respectively. The results are consistent with those of the Lorenz system: the ten-dimensional QT-RC performs comparably to, or even better than, the 100-dimensional RC. In particular, in regions where the absolute value of ρ is small, both QT-RC and HP-RC still maintain a strong predictive capability.

C. Important roles of time-delayed coordinates and nonlinear transformation

To isolate the contributions of time-shift and nonlinear transformations, we performed a controlled ablation study at $M = 100$ reservoir nodes, while all hyperparameters remain identical. In particular, we compare the following cases: traditional RC, RC shifted in time (TS-RC) only, quadratic operation in the reservoir state $\mathbf{r}(t)$ only, and hierarchical product in the reservoir state $\mathbf{r}(t)$ directly. The results are shown in Fig. 7 and summarized in the following. Note that the ablation study at a smaller reservoir size $M = 10$ is shown in Fig. 16, showing qualitatively consistent results.

(i) RC with time-shifted states alone: In this case, we only perform a time-shift concatenation on the reservoir states [Eq. (3)] without applying nonlinear polynomial transformations, directly producing a linear output. Figure 7(b) shows the prediction phase diagram of a 100-dimensional RC on the Lorenz system. Compared with Fig. 5, it can be seen that time-shifted RC improves performance in the region where $|\rho| > 1$ and d are large, but offers limited improvement in the region where $|\rho| < 1$. This behavior is consistent with the interpretation that the time-shift transformation primarily provides additional temporal information, but may be insufficient in capturing certain nonlinear features.

(ii) Quadratic transformed RC: In this case, the output is obtained by directly applying a second-order polynomial transformation to the current reservoir state $\mathbf{r}(t)$. Figure 7(c) shows the phase diagram of the prediction color-coded by t_p of the Lorenz system. Remarkably, the quadratic transformation alone leads to excellent predictive performance in a wide region where $|\rho| < 1$. This indicates that explicitly introducing quadratic terms greatly enhances the expressive power of low-dimensional reservoirs, allowing them to effectively model the nonlinearity of chaotic systems even without strong dynamical memory (i.e., $|\rho| < 1$).

(iii) Hierarchical product in $\mathbf{r}(t)$ directly: The hierarchical operation of a direct pairwise product on the reservoir state $\mathbf{r}(t)$ results in excellent predictive performance, where $|\rho| < 1$, as shown in Fig. 7(d), is similar to Fig. 7(b). This suggests that product operation serves as an alternative way to enhance the nonlinear property of the chaotic system.

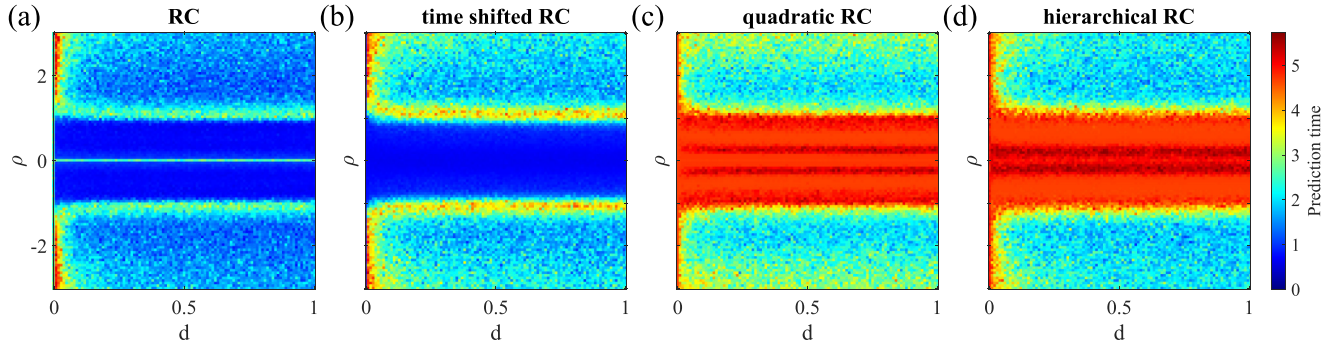


FIG. 7. Decomposition analysis revealing the complementary roles of time-shift and nonlinear operations. (a) traditional RC, (b) RC with time-shifted states alone on $\mathbf{r}(t)$ improves performance for $|\rho| > 1$, (c) RC with quadratic expansion on $\mathbf{r}(t)$ alone yields excellent performance in the $|\rho| < 1$ region, showing that explicit nonlinearity reduces spectral-radius dependence, and (d) RC with hierarchical product operation on $\mathbf{r}(t)$ alone. The same data are used for training for all cases, and the same parameters are chosen: $M = 100$, $\sigma = 1$, $\alpha = 0.1$, $\beta = 10^{-6}$.

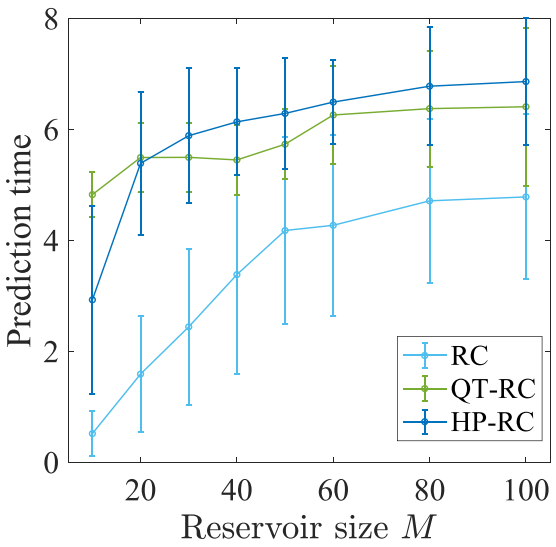


FIG. 8. Valid prediction time as a function of reservoir size M for RC, QT-RC, and HP-RC, using each method's optimal (ρ, d) setting. Error bars indicate standard deviations over 100 independent realizations.

Based on Fig. 7, both time-shift and nonlinear operations on $\mathbf{r}(t)$ (quadratic and hierarchical product) significantly outperform traditional RC. The two transformations affect different regions of the parameter space: time-shifting primarily improves performance for $|\rho| > 1$ and large d, while nonlinear operation achieves optimal performance in the $|\rho| < 1$ region.

In summary, the three main advantages of QT-RC and HP-RC are summarized as follows:

(1) Reduced dimensionality requirement: consistent with the enhanced expressiveness provided by nonlinear transformations.

(2) Performance maintained in dense connectivity (large d): consistent with the expansion of state information through the time-shift transformation.

(3) Performance maintained in the region of $|\rho| < 1$: consistent with the explicit modeling of nonlinear features enabled by nonlinear transformations.

Thus, time-shift and nonlinear transformations play complementary roles: time-shift primarily provides additional temporal information and improves robustness in the $|\rho| > 1$ region, while nonlinear transformations enhance expressiveness and reduce spectral-radius dependence in the $|\rho| < 1$ region. Their combination in HP-RC yields a synergistic improvement, collectively enhancing predictive performance and robustness across the entire parameter space.

Finally, to compare the achievable prediction horizon as a function of reservoir size, Fig. 8 shows the valid prediction time as a function of M. Since the three methods have different optimal working regions in the (ρ, d) plane, each method uses its own optimal (ρ, d) setting that remains in the high-performance region across the full range of M (RC: $\rho = -2.4$, $d = 1$; QT-RC: $\rho = 0.3$, $d = 20$; HP-RC: $\rho = -0.6$, $d = 20$). All other hyperparameters ($\tau = 30$, $m = 3$, $\alpha = 0.1$, $\beta = 10^{-6}$, $\sigma = 1$) and training protocols are kept identical. All three methods improve with increasing reservoir size: QT-RC shows a clear advantage for small reservoirs, HP-RC achieves a higher upper bound for large reservoirs, and RC exhibits a monotonic improvement with size. These results further confirm that the proposed methods provide robust and scalable performance across both hyperparameter space and reservoir dimension.

D. Characterization of memory capacity

The phase-diagram analysis in Sec. V C suggests that time-shift transformations and nonlinear feature expansions play complementary roles: the former primarily enhance temporal memory, while the latter increase nonlinear expressiveness and reduce sensitivity to the spectral radius. We use memory capacity to quantitatively support this interpretation with quantitative evidence.

First, we compute the linear memory capacity of the feature set following Jaeger [38]: for each delay $q \in [1, K_1]$, a linear readout is trained to recall the input $u(t-q)$, and the capacity coefficient is the squared correlation coefficient between the target input $u(t-q)$ and its readout $\hat{z}(t)$. More specifically, we compute

$$c_q = \frac{\mathrm{COV}^2(u(t-q), \hat{z}(t))}{\mathrm{VAR}(u(t)) \cdot \mathrm{VAR}(\hat{z}(t))}. \tag{16}$$

The total linear memory capacity is then simply the sum of these individual capacities for all delays, i.e., $MC = \sum_{q=1}^{K_1} c_q$, which represents the total amount of linearly accessible short-term memory the reservoir possesses.

We further adopt the information processing capacity (IPC) framework of Dambre *et al.* [39], which decomposes the total capacity into linear memory and higher-order nonlinear contributions. More specifically, capacity is computed over the target space spanned by input delay monomials. Here, "linear" and "quadratic" refer to the degree of the monomial: first-order linear monomials are single delayed inputs $u(t-q)$ for $q = 1, \ldots, K_1$; second-order quadratic monomials are products of two delayed inputs $u(t-q_1)u(t-q_2)$ for $1 \leqslant q_1 \leqslant q_2 \leqslant K_2$. The capacity coefficient is computed for each orthogonal target and summed; namely, the first-order contributions define the linear memory capacity, the second-order contributions define the quadratic nonlinear capacity, and their sum is the total capacity. Physically, the quadratic capacity measures the nonlinear expressiveness of the feature set—its ability to linearly reconstruct quadratic functionals of the input history.

Note that, when computing memory capacity, we drive the reservoir with a random, independent and identically distributed (i.i.d.) input sequence, e.g., uniformly distributed input $u(t) \in [-1, 1]$.

Figure 9 summarizes the results for RC, TS-RC, Quadratic-RC, QT-RC, and HP-RC (all with reservoir nodes $M = 10$). Figure 9(a) shows that time-shift transformations significantly increase linear memory capacity. Panel (b) shows that nonlinear expansions increase quadratic (nonlinear) capacity while largely preserving linear capacity. Panel (c) shows that nonlinear expansions increase total capacity (linear + quadratic).

These metrics provide quantitative, consistent results for the complementary roles discussed in Sec. V C, namely, time-shift transformations primarily increase linear memory, while

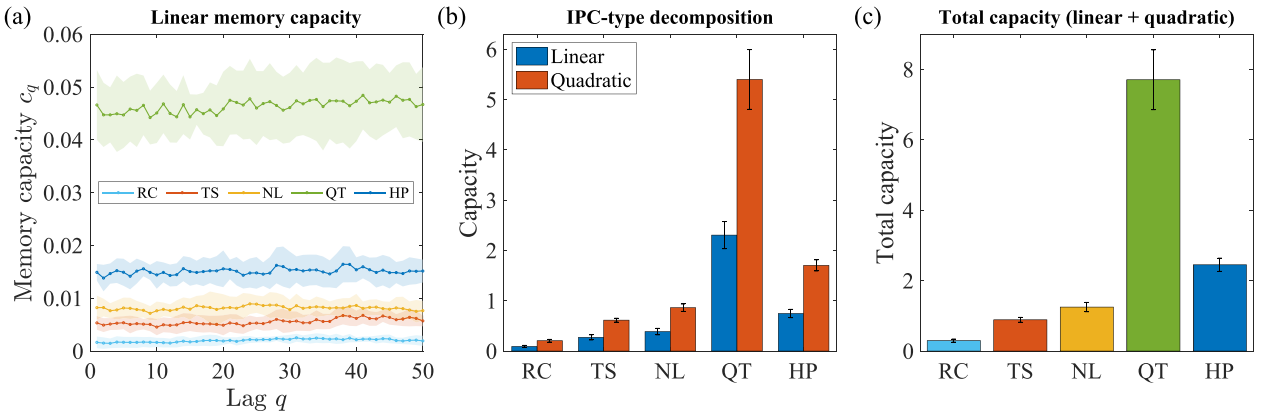


FIG. 9. Quantitative characterization of linear and nonlinear memory capacity. (a) Linear memory capacity as a function of lag q. (b) IPC decomposition: linear (memory) capacity versus quadratic (nonlinear) capacity. (c) Total capacity (linear + quadratic). All variants use the same parameters: $M = 10$, $\rho = 0.9$, $d = 0.1$, $\alpha = 0.1$, $\beta = 10^{-6}$, $m = 3$, and $\tau = 10$.

quadratic expansions substantially increase quadratic and total capacity. The combination of both operations in QT-RC and HP-RC yields the largest overall capacity.

E. Algorithmic parameter choice for hierarchical product RC

In this section, we discuss the choices for three algorithmic parameters in the HP-RC [Eq. (9)]: (i) the number of hierarchical layers, (ii) the product interval (time difference) between points, and (iii) the order of the hierarchical product. In addition, we focus on the Lorenz example only. The following conclusions hold for the Rössler system.

Figure 10(a) shows phase diagrams (d, ρ) color-coded by t_p obtained by HP-RC under different hierarchical layers (note that we focus on the region of sparse connectivity $d \in (0, 0.1)$ for the reservoir). From left to right, the results are shown for layers 0 to 3, where layer 0 represents the initial state without product. We find from the phase diagrams that the predictive performance of layer 1 is the most prominent. This is because, after a single product operation, the extracted features mainly take the form of quadratic nonlinearities, which match the leading-order nonlinear terms in the Lorenz equations. We note that the quadratic expansion is not intended to represent

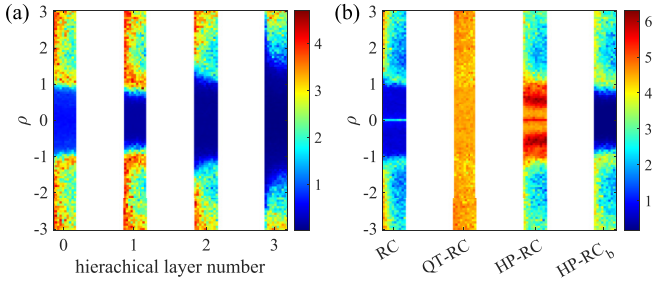


FIG. 10. Effect of hierarchical depth and layer combination on HP-RC performance ($d \in (0, 0.1)$). (a) Phase diagrams for different product layers $i = 0, 1, 2, 3$; layer 1 (quadratic) yields the strongest improvement, matching the quadratic nonlinearity of the Lorenz system. (b) Comparison of full HP-RC with using only the first product layer shows that including linear information from layer 0 is essential; full HP-RC outperforms both QT-RC and the single-layer variant, capturing both linear and quadratic features.

the full flow ϕ^t of the Lorenz system for finite t. Rather, with the small leakage rate $\alpha = 0.1$ used here, it provides a leading-order approximation of the single-step reservoir map. The empirical evidence [Figs. 11(c) and 11(d)] shows that this approximation, when iterated in closed loop, is sufficient for accurate long-term prediction.

In Fig. 10(b), we further compare the predictive performance of traditional RC, QT-RC, full HP-RC, and HP-RC using only the output of the first product layer. It can be seen that using only the first-layer product states for prediction yields performance significantly worse than that of QT-RC or full HP-RC, and is even comparable to that of traditional RC. This is because the dynamics of the Lorenz system is not entirely composed of quadratic terms; they also include important linear components. Relying solely on quadratic nonlinear features is therefore insufficient. This indirectly explains why QT-RC and full HP-RC outperform traditional RC: their final feature representations incorporate both the linear information from the original reservoir states and the intrinsic quadratic nonlinearities of the system, thereby capturing the dynamical structure more completely.

From Figs. 11(a) and 11(b), we find that HP-RC with quadratic product significantly outperforms HP-RC with cubic product in predicting performance. This result is consistent with our previous comparison of performance across different hierarchical layers and is primarily attributed to the fact that the nonlinear features introduced by quadratic product better match the dominant quadratic nonlinearity inherent in the Lorenz system. Furthermore, under the same number of hierarchical layers, variations in the product interval within a small range (e.g., time difference of one to three steps) have no significant impact on predictive performance. The comparison between the quadratic and quartic products in Fig. 11(c) further supports this conclusion, indicating again that a moderate degree of nonlinear expansion (up to second order) is sufficient to effectively capture the system's dynamical characteristics, while higher-order layers yield no substantial additional benefit.

To further investigate this point, we also evaluated the single-step prediction error of polynomial maps of order $p = 1, \ldots, 4$ fitted to the same training data [Fig. 11(d)]. As expected, higher-order polynomials yield smaller single-step

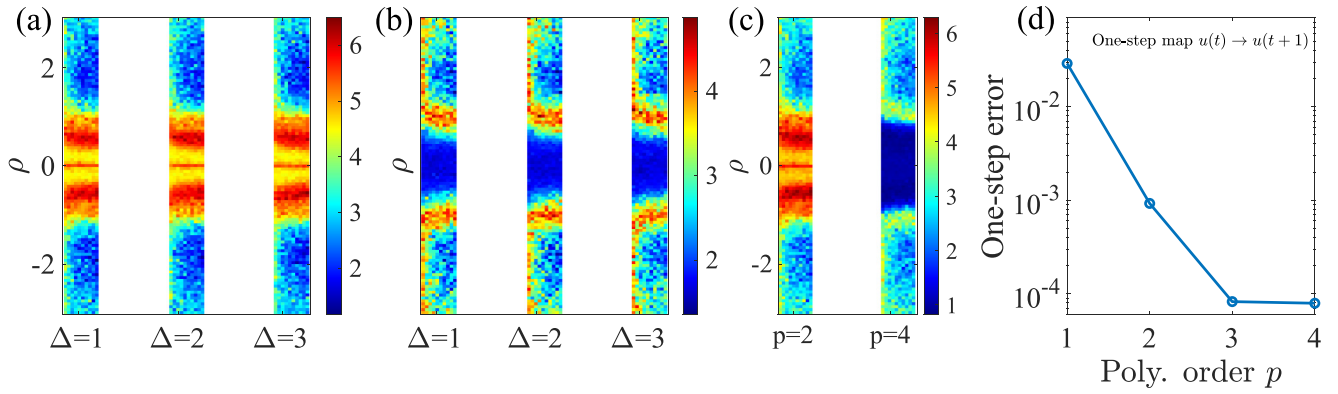


FIG. 11. Influence of product order p and time interval Δ on HP-RC performance. (a) Quadratic products ($p = 2$) with different intervals $\Delta = 1, 2, 3$ yield similar high performance. (b) Cubic products ($p = 3$) with intervals $\Delta = 1, 2, 3$ are less effective, indicating that higher-order products are not necessary for Lorenz dynamics. (c) Quadratic ($p = 2$) clearly outperforms quartic ($p = 4$) products, confirming that a moderate nonlinear expansion (second order) is sufficient for this system. (d) Single-step prediction error of polynomial maps of order $p = 1, \ldots, 4$.

errors. However, this single-step advantage does not translate into better closed-loop chaotic prediction: in Figs. 11(a)–11(c), the quadratic product ($p = 2$) consistently outperforms higher orders. This indicates that for closed-loop chaotic prediction, the relevant criterion is not the accuracy of a single-step approximation but rather the ability of the iterated map to sustain the correct attractor dynamics. The quadratic product captures the dominant nonlinear interactions relevant for maintaining the correct attractor dynamics in the examined parameter range.

F. Robust accuracy

Because the RC algorithm involves two random matrices W_{in}, W_r, the prediction phase diagrams and the single-prediction results shown above cannot fully reflect the algorithm's performance. The phase diagram represents the average prediction time; the best approach would be to plot the statistical distribution of the predicted times, but this can only display the results under specific hyperparameters. To address this limitation, we further use two metrics—accuracy and robustness—to compensate for the insufficient characterization of the algorithm's prediction stability.

Specifically, taking the Lorenz system as an example, accuracy is measured by the root mean square error (RMSE) within a time horizon of 400 steps (approximately 3.5 Lyapunov times). Robustness is quantified by the ratio of predictions whose VPT are above 90% of the mean value that is obtained from an ensemble of realizations. In addition, the standard deviation (std) of the prediction time and the std of the RMSE are also presented as complementary indicators.

In the following, we fix each spectral radius ρ and select the corresponding optimal average degree d and perform 1000 random realizations for each configuration. For each, we compute: (i) the average VPT, (ii) the prediction accuracy RMSE, and (iii) the ratio of predictions with VPT above 90% of the mean. Figure 12(a) shows the mean VPT and standard deviation (error bars) of traditional RC ($M = 100$), QT-RC ($M = 10$), and HP-RC (100-dimensional), respectively, under different ρ. In general, QT-RC and HP-RC achieve not only a higher average VPT but also a smaller standard deviation for most values of ρ, demonstrating superior accuracy and stability.

For a more intuitive comparison, Fig. 13 presents a scatter plot of the mean VPT and RMSE versus the standard deviation obtained by the three algorithms under different random realizations. The cluster of points for QT-RC and HP-RC is more concentrated in the upper right corner (high mean VPT) and the bottom (low standard deviation), further confirming their advantages in accuracy and robustness.

G. Partial state information experiments

Following the suggestion of Fleddermann *et al.* [40], we also test our methods under partial state information, using only the $x(t)$ component of the Lorenz and Rössler systems as input. The protocol is identical to the full-state experiments ($x(t), y(t), z(t)$), except that only the scalar $x(t)$ is injected into the reservoir and used as a training target.

We compare traditional RC ($M = 100$), QT-RC ($M = 10$), and HP-RC ($M = 100$). The (ρ, d) values used here are selected from the phase diagrams at the optimal grid points (Figs. 5 and 6), following the same protocol as in Table I. The results in Table II show that QT-RC and HP-RC consistently outperform traditional RC, even under partial observation. For the Lorenz system, QT-RC achieves a mean valid prediction time of 0.73 Lyapunov times, compared to 0.46 for traditional RC (a 59% improvement). For the Rössler system, HP-RC achieves the best performance (2.92 Lyapunov times), compared to 1.71 for traditional RC. These results confirm that the

TABLE II. Valid prediction time t_p (mean $\pm$ standard deviation, in Lyapunov times) for partial state information (only the $x(t)$ component is observed). Results are obtained at the optimal (ρ, d) settings from the phase diagrams. The number of independent realizations is 200 for the Lorenz system and 100 for the Rössler system.

System	Traditional RC	QT-RC	HP-RC
Lorenz (only x)	0.46 ± 0.35	0.73 ± 0.81	0.60 ± 0.51
Rössler (only x)	1.71 ± 0.45	2.15 ± 0.65	2.92 ± 0.90

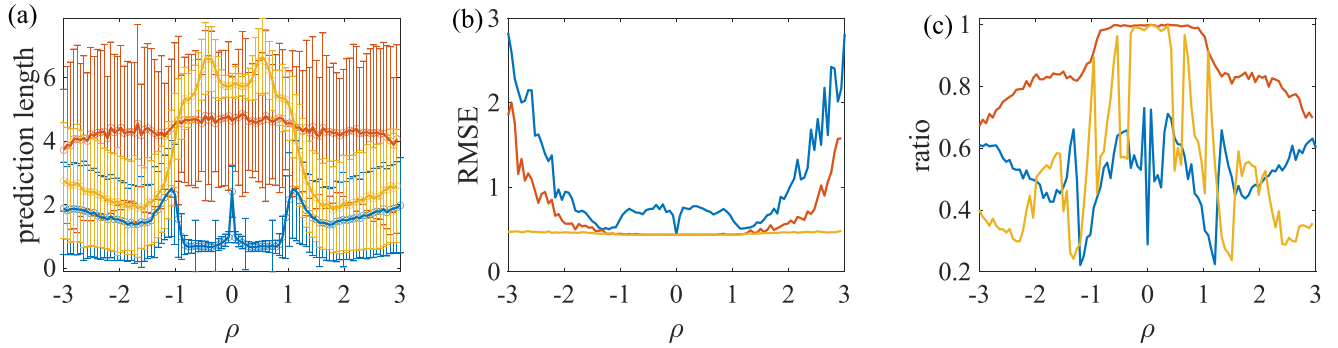


FIG. 12. Quantitative comparison of accuracy and robustness over 1000 independent realizations. (a) Mean valid prediction time t_p with error bars (standard deviation) as a function of spectral radius ρ. (b) Prediction error (RMSE) averaged over the same realizations. (c) Ratio of realizations achieving t_p above 90% of the mean value.

benefits of time-shift and nonlinear transformations are robust to the amount of available state information.

Our results are consistent with the observations of Ref. [40]: they found that for the Lorenz-96 system under partial observation, the mean valid prediction time was mostly below 2 Lyapunov times, even when using reservoirs with up to $M = 1500$ nodes and extensive hyperparameter optimization. In our setting, with much smaller sizes ($M = 10$–100) and a simpler hyperparameter protocol, QT-RC and HP-RC achieve comparable performance, highlighting the efficiency of our proposed methods.

H. Comparison of feature expansion locations

To empirically justify the choice of applying the nonlinear expansion to the reservoir state $\mathbf{r}(t)$ rather than to the raw input $\mathbf{u}(t)$, we performed a controlled comparison between two otherwise comparable constructions that place the same expansion at different stages, as illustrated schematically in Fig. 14. In the first construction ("input-feature RC"), the expansion is applied directly to the raw input $\mathbf{u}(t)$, and the expanded features are used as the driving signal for a standard reservoir; the readout is linear in $\mathbf{r}(t)$, without further expansion. In the second construction (QT-RC or HP-RC), the raw input drives the reservoir, and the expansion is applied to the reservoir state $\mathbf{r}(t)$ before the readout.

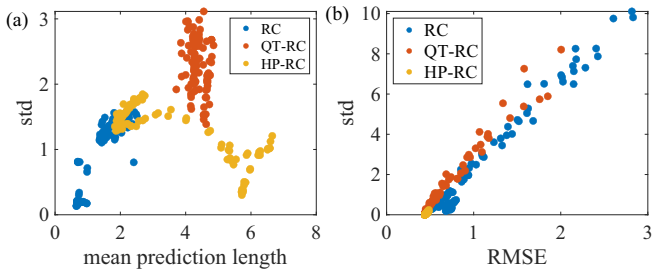


FIG. 13. Scatter plots illustrating the accuracy-robustness tradeoff. (a) Mean valid prediction time versus its standard deviation; QT-RC and HP-RC points cluster in the lower-right (high mean, low std). (b) Prediction error (RMSE) versus its standard deviation; our methods cluster in the lower-left (low error, low variance).

Under otherwise identical settings, the results are shown in Fig. 14. For the quadratic expansion, the input-feature construction achieves 1.48 ± 1.04 Lyapunov times, whereas QT-RC achieves 4.77 ± 0.64 Lyapunov times. For the hierarchical product expansion, the input-feature construction achieves 1.21 ± 0.65 Lyapunov times, whereas HP-RC achieves 5.08 ± 0.64 Lyapunov times. In both cases, placing the expansion on the reservoir states yields substantially longer valid prediction times (approximately three- to four-fold improvements), with smaller variance. The evaluation time (3000 steps) is much longer than the observed prediction lengths, so the reported distributions are not truncated by the test length. These results confirm that the nonlinear expansion should be applied to the reservoir states rather than to the raw inputs.

VI. COMPARISON WITH THE NEXT-GENERATION RC

Here, we explicitly compare the performance of our method with the so-called Next Generation RC (NG-RC) algorithm [41]. This method abandons the random reservoir in normal RC, and directly applies time-delay embedding to the input time series, constructs polynomial features, and then uses ridge regression for linear output. The feature vector of NG-RC often consists of a constant term, linear delay terms (current and past k moments with interval s), and higher-order polynomial terms (usually quadratic or cubic).

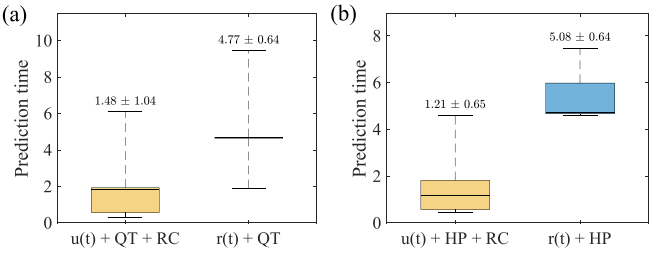


FIG. 14. Comparison of two constructions placing the nonlinear expansion at different stages. (a) Quadratic expansion: input-feature QT (left) vs. QT-RC (right). (b) Hierarchical product expansion: input-feature HP (left) vs. HP-RC (right). In both cases, applying the expansion to the reservoir states $\mathbf{r}(t)$ (right panels) yields substantially longer prediction times than applying it to the raw inputs $\mathbf{u}(t)$ (left panels).

Although NG-RC achieves performance comparable to optimized traditional RC on several benchmark tasks and trains faster, its hyperparameters have a significant impact on prediction performance and lack theoretical guidance. More specifically, while NG-RC eliminates dependence on the hyperparameters of traditional RC (spectral radius, sparsity, leakage rate, etc.), it introduces its own set of hyperparameters: the number of delays (k), the delay interval (s), and the polynomial order (p). The impact of these new hyperparameters on performance is very significant. Currently, methods for choosing them are primarily heuristic or based on grid search, lacking a universal theoretical principle. How to efficiently and robustly select these hyperparameters is a very active research direction in the NG-RC field.

Interestingly, the QT-RC and HP-RC proposed in this paper share certain similarities with NG-RC in their overall design: both methods obtain a certain state representation, then introduce time delays and explicit nonlinear transformations, followed by linear regression for output. However, the key difference lies in the object to which the nonlinear feature construction is applied. NG-RC directly applies delay embedding and polynomial expansion to the input data $\mathbf{u}(t)$, while our method performs delay and quadratic/product operations on the reservoir states $\mathbf{r}(t)$. As will be shown in this section, this distinction leads to significant differences in hyperparameter robustness and generalization performance between the two approaches.

A. Valid prediction time

First, we clarify that NG-RC is deterministic for fixed training data and hyperparameters. Its performance variance arises from the choice of initial condition (equivalently, the training window), since the training segment is taken as a fixed-length window preceding the test start. To provide a fair comparison in the following, NG-RC is evaluated over 500 different initial conditions, yielding a t_p distribution comparable to the 500 independent reservoir realizations used for QT-RC and HP-RC.

Second, QT-RC and NG-RC are optimized using identical random search protocols: 150 candidate settings, each evaluated over 20 independent runs (QT-RC: reservoir realizations; NG-RC: initial conditions), with the objective being the mean valid prediction time. The QT-RC search space covers $\rho \in [-1.5, 1.5]$, $d \in [1, 20]$, $\alpha \in [0.01, 1]$, $\beta \in [10^{-8}, 10^{-2}]$ (log-uniform step), $\tau \in \{10, 20, 30, 40, 50\}$, $m \in \{2, 3, 4\}$, and $W_{\text{in}} \in \{0.5, 1, 2\}$. The optimal setting for QT-RC is $\rho = 0.35$, $d = 8.7$, $\alpha = 0.59$, $\beta = 5.9 \times 10^{-8}$, $\tau = 30$, $m = 4$, $W_{\text{in}} = 0.5$ (with $\tau = 30$ exactly matching the delay in Sec. IV). The NG-RC search space covers $k \in \{1, \ldots, 5\}$, $s \in \{1, \ldots, 10\}$, and $\beta \in [10^{-8}, 10^{-3}]$ (log-uniform step), giving the optimal values ($k = 4$, $s = 9$, $\beta = 4.9 \times 10^{-7}$). We emphasize that QT-RC, HP-RC, and NG-RC are all optimized with the same setting, which are shown in Fig. 15(a) therefore is fair.

Figure 15(a) shows the resulting distributions. Under the equal optimization method, HP-RC achieves the longest mean prediction time (6.23 ± 0.97 Lyapunov times). QT-RC (5.99 ± 1.15) and NG-RC (5.94 ± 1.51) have comparable means, but QT-RC shows a smaller standard deviation (1.15 vs. 1.51) and higher median (6.03 vs. 5.80). The optimized NG-RC reaches this level only at the upper edge of its stable delay window ($s = 9$). Thus, QT-RC offers better run-to-run reliability with similar mean performance. This reliability advantage, together with the broad robustness across the (ρ, d) plane shown in Fig. 5, is the core message of the comparison.

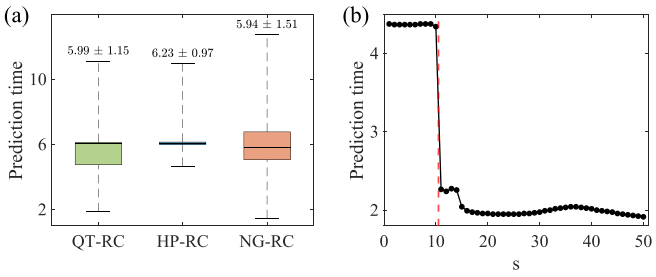


FIG. 15. Comparison with Next-Generation Reservoir Computing (NG-RC). (a) Histograms of t_p from 500 independent reservoirs of QT-RC and HP-RC vs. 500 initial conditions of NG-RC. The mean values of QT-RC (5.99 Lyapunov times) and HP-RC (6.23) exceed the NG-RC result (5.94), and the distributions are narrow. (b) NG-RC performance as a function of its delay parameter s: it peaks only for $1 \leqslant s \leqslant 10$ and collapses for larger delays.

B. Choice of time delay s

As shown in Fig. 3 in Sec. IV, the choice of time delay τ in our methods is guided by the autocorrelation function. The mean valid prediction time t_p reaches a maximum when the delay τ is chosen in this region. In this respect, the embedding for the reservoir state $\mathbf{r}(t)$ is consistent with Takens' embedding theorem.

However, the dependence of NG-RC on delays does not conform to the expectations of Takens' embedding theorem, and its performance drops sharply when the delay becomes even moderately large. In Fig. 15(b), we show the valid prediction time t_p versus time delay s.

The results show that when s is between 1 and 10, t_p remains at a relatively high level (approximately 4–4.5 Lyapunov times); once $s > 10$, the prediction performance immediately drops by about half. It is worth noting that the optimal delay for NG-RC always falls within the 1–10 interval (for other systems), rather than at the dynamical delay determined by the autocorrelation function [for the Lorenz system, the decorrelation time corresponds to a delay of approximately $\tau = 30$ in Fig. 3(a)]. This phenomenon suggests that the delay operation in NG-RC is not intended for topological embedding reconstruction of the attractor in phase space, but rather as a means to fit discrete-time maps (such as numerical integrators of the fourth-order Runge-Kutta method). When the delay becomes too large, the local approximation on which the integrator relies breaks down, leading to prediction collapse. In other words, NG-RC essentially learns the specific numerical integration scheme implicit in the training data, rather than the underlying continuous-time vector field.

In contrast, QT-RC and HP-RC strictly follow embedding theory for delay selection, thereby achieving stable and long-term prediction performance. Our methods successfully extract the dynamics that are independent of discretization

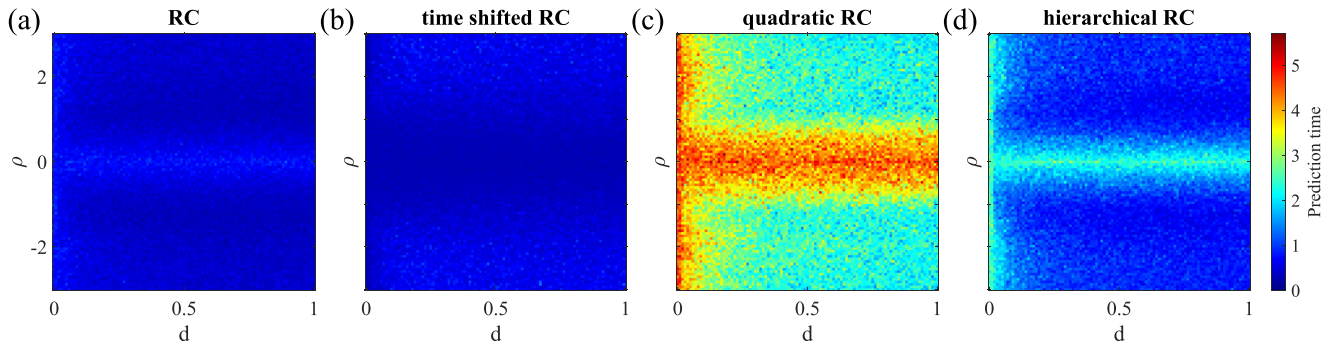


FIG. 16. The caption as Fig. 7 for a smaller reservoir size $M = 10$. Note that quadratic expansion and hierarchical product operation on $\mathbf{r}(t)$ alone, namely in (c), (d), yield excellent performance in the $|\rho| < 1$ region.

through time-delayed reservoir states and explicit nonlinear transformations.

In summary, although NG-RC features a concise design, its high sensitivity to the delay parameter s limits its generalization capability in scenarios with unknown or varying sampling rates. In contrast, our methods, by combining the dynamical evolution of the reservoir with delay transformations that adhere to embedding theory, exhibit better generalization ability and hyperparameter robustness, offering a new pathway toward efficient and hardware-friendly reservoir computing.

C. Comparison with other methods

We note that our implementation of a reservoir has different purposes from time-delayed reservoir computing [27]. In our approach, the traditional update rule of the reservoir is not changed. The delays are applied after the reservoir states have been computed. The reservoir states $\mathbf{r}(t), \mathbf{r}(t-\tau), \ldots, \mathbf{r}(t-(m-1)\tau)$ are combined into a single feature vector, which is then used as input to a subsequent output layer for further analysis. Therefore, time-delayed embedding is used to create new features from the reservoir's states to improve the information content of the output.

Previous work has shown that time delays can create virtual nodes [24], enable ultrafast photonic processing [25], provide theoretical embedding guarantees [27], break node synchronization [28], and systematically reduce reservoir size [29]. However, these methods are based solely on linear time-shifted states. Our QT-RC and HP-RC extend this paradigm by introducing explicit polynomial (quadratic) feature expansion on the time-shifted reservoir states, enabling the capture of multiplicative nonlinearities that pure time-delay methods cannot represent.

Viehweg *et al.* [33] recently proposed a similar product-derived RC, a deterministic alternative to RC that constructs features from pairwise products of time-delayed raw inputs. Their method operates directly on raw inputs and does not include any recurrent reservoir dynamics. In contrast, our HP-RC method applies the product operations to the reservoir state $\mathbf{r}(t)$ rather than raw inputs. The reservoir, though small (e.g., $M = 10$), provides a nonlinear dynamical transformation that embeds the input history into a high-dimensional state space. Hierarchical operations on these $\mathbf{r}(t)$ can thus capture interactions that are not directly present in the raw input-a crucial advantage for multivariate chaotic systems such as Lorenz, where the quadratic terms involve variables that may not be explicitly measured or provided as input. We therefore view Viehweg's approach and ours as complementary: they focus on eliminating randomness entirely in feed-forward architectures, while we retain a minimal reservoir to enhance expressiveness and reduce delay sensitivity.

While most RC systems employ a linear readout for its efficiency and convexity, some studies have explored nonlinear readouts, typically at the cost of increased training complexity. For example, nonlinear readout approaches replace $\mathbf{W}_{\text{out}}$ with a shallow neural network or kernel method, requiring iterative optimization (e.g., backpropagation) instead of closed-form ridge regression. In contrast, our methods retain the standard linear readout $\mathbf{W}_{\text{out}}$ trained via ridge regression; nonlinearity is introduced exclusively through a deterministic transformation applied to the reservoir states $\mathbf{r}(t)$ before the readout, namely, time-shift concatenation followed by polynomial expansion (QT-RC) or hierarchical product (HP-RC). This preserves convexity and computational efficiency while achieving expressive power for chaotic prediction.

VII. CONCLUSIONS

In this work, we have shown that combining Takens'-inspired time-delay embedding with explicit Volterra-style nonlinear feature expansion provides a powerful and theoretically grounded framework for enhancing reservoir computing. Our proposed methods, hierarchical product (HP-RC) and quadratic transformed (QT-RC), overcome the limitations of traditional RC. They reduce reservoir size and hyperparameter sensitivity while preserving the efficiency of linear ridge regression. Time-delay embedding alone, though effective in expanding the feature space and providing historical context [24,27], remains fundamentally a linear operation: concatenating delayed copies of the reservoir state vector does not introduce new nonlinear interactions among its components. By explicitly constructing Volterra-style features, that is, quadratic monomials in QT-RC and hierarchical products in HP-RC, from time-delayed reservoir states, we capture the multiplicative couplings that are ubiquitous in chaotic systems, such as the quadratic terms that govern the leading-order nonlinear interactions in the Lorenz equations.

Our numerical results indicate that an explicit nonlinear transformation substantially improves the prediction performance. By applying a quadratic polynomial expansion

(including all pairwise products of delayed states) or a hierarchical product operation that combines shifted copies, the model gains the ability to represent interactions across different time points. This capability is critical for accurately reproducing the system's attractor geometry. Our two-step design, consisting of temporal expansion via time delays followed by explicit nonlinear feature construction, preserves the computational efficiency of linear ridge regression while endowing low-dimensional reservoirs (e.g., five to ten nodes) with the expressive power required for high-fidelity, long-term chaotic prediction. Crucially, the synergy between linear delay expansion and nonlinear feature engineering dramatically reduces sensitivity to hyperparameters such as the spectral radius and connectivity, offering a compelling path toward robust, hardware-friendly reservoir computing.

Note that we use the standard RC readout, and the nonlinear transformation we propose is a preprocessing step applied to the reservoir states $\mathbf{r}(t)$ before they are fed into the existing linear readout $\mathbf{W}_{\text{out}}$. Therefore, this design preserves convexity, closed-form training, and computational efficiency while achieving or surpassing the prediction accuracy of traditional RC with significantly smaller reservoirs and reduced hyperparameter sensitivity. Consequently, our approach occupies a unique middle ground: it gains the nonlinear expressivity required for chaotic systems without abandoning the simplicity and robustness that make RC attractive in the first place.

A limitation of our approach is the increased feature dimension due to product expansion, which scales with the number of delays and product order. This can become computationally expensive for very large m or p. However, as shown in our results, small m (e.g., 3–4) and $p = 2$ are sufficient for the systems considered, and the deterministic nature of the expansion avoids the need for multiple random initializations. Future work could explore adaptive product selection, alternative nonlinear transformations, or hardware implementations of the hierarchical product structure.

ACKNOWLEDGMENTS

Parts of this work have been financially supported by the National Natural Science Foundation of China (Grant No. 42461144209), the Natural Science Foundation of Shanghai (Grant No. 24ZR1420700), and the STI2030-Major Projects (Grant No. 2021ZD0202600).

The authors declare that they have no conflict of interest.

DATA AVAILABILITY

The data that support the findings of this article are openly available at [42].

[1] H. Jaeger and H. Haas, Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication, Science **304**, 78 (2004).

[2] K. Nakajima and I. Fischer, *Reservoir Computing - Theory, Physical Implementations, and Applications* (Springer Singapore, 2021).

[3] M. Yan, C. Huang, P. Bienstman, P. Tino, W. Lin, and J. Sun, Emerging opportunities and challenges for the future of reservoir computing, Nat. Commun. **15**, 2056 (2024).

[4] 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 Networks **115**, 100 (2019).

[5] J. Pathak, B. Hunt, M. Girvan, Z. Lu, and E. Ott, Model-free prediction of large spatiotemporally chaotic systems from data: A reservoir computing approach, Phys. Rev. Lett. **120**, 024102 (2018).

[6] L. M. Smith, J. Z. Kim, Z. Lu, and D. S. Bassett, Learning continuous chaotic attractors with a reservoir computer, Chaos **32**, 011101 (2022).

[7] L. Shi, Y. Yan, H. Wang, S. Wang, and S.-X. Qu, Predicting chaotic dynamics from incomplete input via reservoir computing with $(d + 1)$-dimension input and output, Phys. Rev. E **107**, 054209 (2023).

[8] P. Amil, M. C. Soriano, and C. Masoller, Machine learning algorithms for predicting the amplitude of chaotic laser pulses, Chaos **29**, 113111 (2019).

[9] S. Ghosh, A. Senapati, A. Mishra, J. Chattopadhyay, S. K. Dana, C. Hens, and D. Ghosh, Reservoir computing on epidemic spreading: A case study on COVID-19 cases, Phys. Rev. E **104**, 014308 (2021).

[10] F. Hassanibesheli, J. Kurths, and N. Boers, Long-term ENSO prediction with echo-state networks, Environmental Research: Climate **1**, 011002 (2022).

[11] T. Arcomano, I. Szunyogh, J. Pathak, A. Wikner, B. R. Hunt, and E. Ott, A machine learning-based global atmospheric forecast model, Geophys. Res. Lett. **47**, e2020GL087776 (2020).

[12] D. Patel and E. Ott, Using machine learning to anticipate tipping points and extrapolate to post-tipping dynamics of non-stationary dynamical systems, Chaos **33**, 023143 (2023).

[13] X. Li, Q. Zhu, C. Zhao, X. Qian, X. Zhang, X. Duan, and W. Lin, Tipping point detection using reservoir computing, Research **6**, 0174 (2023).

[14] T. M. Bury, R. I. Sujith, I. Pavithran, M. Scheffer, T. M. Lenton, M. Anand, and C. T. Bauch, Deep learning for early warning signals of tipping points, Proc. Natl. Acad. Sci. USA **118**, e2106140118 (2021).

[15] J. Jiang, Z.-G. Huang, C. Grebogi, and Y.-C. Lai, Predicting extreme events from data using deep machine learning: When and where, Phys. Rev. Res. **4**, 023028 (2022).

[16] V. A. Pammi, M. G. Clerc, S. Coulibaly, and S. Barbay, Extreme events prediction from nonlocal partial information in a spatiotemporally chaotic microcavity laser, Phys. Rev. Lett. **130**, 223801 (2023).

[17] W. Gilpin, Chaos as an interpretable benchmark for forecasting and data-driven modelling, in *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*, edited by J. Vanschoren and S. Yeung (Neural Information Processing Systems Foundation, 2021).

[18] W. Gilpin, Model scale versus domain knowledge in statistical forecasting of chaotic systems, Phys. Rev. Res. **5**, 043252 (2023).
[19] Z. Lu, J. Pathak, B. Hunt, M. Girvan, R. Brockett, and E. Ott, Reservoir observers: Model-free inference of unmeasured variables in chaotic systems, Chaos **27**, 041102 (2017).
[20] H. Fan, J. Jiang, C. Zhang, X. Wang, and Y.-C. Lai, Long-term prediction of chaotic systems with machine learning, Phys. Rev. Res. **2**, 012080 (2020).
[21] T. Weng, H. Yang, C. Gu, J. Zhang, and M. Small, Synchronization of chaotic systems and their machine-learning models, Phys. Rev. E **99**, 042203 (2019).
[22] E. Bollt, 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).
[23] J. Liu, K. Jiang, J. Fan, and Y. Zou, Improving the prediction stability of reservoir computing by means of dynamic structural bias, Phys. Rev. Res. **8**, 023114 (2026).
[24] L. Appeltant, M. C. Soriano, G. Van der Sande, J. Danckaert, S. Massar, J. Dambre, B. Schrauwen, C. R. Mirasso, and I. Fischer, Information processing using a single dynamical node as complex system, Nat. Commun. **2**, 468 (2011).
[25] L. Larger, A. Baylón-Fuentes, R. Martinenghi, V. S. Udaltsov, Y. K. Chembo, and M. Jacquot, High-speed photonic reservoir computing using a time-delay-based architecture: Million words per second classification, Phys. Rev. X **7**, 011015 (2017).
[26] B. A. Marquez, J. Suarez-Vargas, and B. J. Shastri, Takens-inspired neuromorphic processor: A downsizing tool for random recurrent neural networks via feature extraction, Phys. Rev. Res. **1**, 033030 (2019).
[27] X.-Y. Duan, X. Ying, S.-Y. Leng, J. Kurths, W. Lin, and H.-F. Ma, Embedding theory of reservoir computing and reducing reservoir network using time delays, Phys. Rev. Res. **5**, L022041 (2023).
[28] E. Del Frate, A. Shirin, and F. Sorrentino, Reservoir computing with random and optimized time-shifts, Chaos **31**, 121103 (2021).
[29] T. L. Carroll and J. D. Hart, Time shifts to reduce the size of reservoir computers, Chaos **32**, 083122 (2022).
[30] F. Takens, Detecting strange attractors in turbulence, in *Dynamical Systems and Turbulence, Warwick 1980*, Lecture Notes in Mathematics, edited by D. Rand and L.-S. Young (Springer, New York, 1981), Vol. 898, pp. 366–381.
[31] S. P. Boyd and L. O. Chua, Fading memory and the problem of approximating nonlinear operators with Volterra series, IEEE Trans. Circuit Syst. **32**, 1150 (1985).
[32] H.-H. Ren, Y.-L. Bai, M.-H. Fan, L. Ding, X.-X. Yue, and Q.-H. Yu, Constructing polynomial libraries for reservoir computing in nonlinear dynamical system forecasting, Phys. Rev. E **109**, 024227 (2024).
[33] J. Viehweg, D. Walther, and P. Mäder, Temporal convolution derived multi-layered reservoir computing, Neurocomputing **617**, 128938 (2025).
[34] J. C. Sprott, *Chaos and Time-Series Analysis* (Oxford University Press, Oxford, 2003).
[35] H. D. I. Abarbanel, R. Brown, J. J. Sidorowich, and L. S. Tsimring, The analysis of observed chaotic data in physical systems, Rev. Mod. Phys. **65**, 1331 (1993).
[36] J. A. Platt, S. G. Penny, T. A. Smith, T.-C. Chen, and H. D. Abarbanel, A systematic exploration of reservoir computing for forecasting complex spatiotemporal dynamics, Neural Networks **153**, 530 (2022).
[37] P. Antonik, N. Marsal, D. Brunner, and D. Rontani, Bayesian optimisation of large-scale photonic reservoir computers, Cognit. Comput. **15**, 1452 (2023).
[38] H. Jaeger, Short term memory in echo state networks, Technical Report 152 (GMD - German National Research Institute for Computer Science, 2002).
[39] J. Dambre, D. Verstraeten, B. Schrauwen, and S. Massar, Information processing capacity of dynamical systems, Sci. Rep. **2**, 514 (2012).
[40] L. Fleddermann, S. Herzog, and U. Parlitz, Enhancing reservoir predictions of chaotic time series by incorporating delayed values of input and reservoir variables, Chaos **35**, 053147 (2025).
[41] D. J. Gauthier, E. Bollt, A. Griffith, and W. A. Barbosa, Next generation reservoir computing, Nat. Commun. **12**, 5564 (2021).
[42] https://github.com/yongPIK/QT_TC_RC.