
A SYSTEMATIC EXPLORATION OF RESERVOIR COMPUTING FOR FORECASTING COMPLEX SPATIOTEMPORAL DYNAMICS

A PREPRINT

Jason A. Platt * Stephen G. Penny † Timothy A. Smith † Tse-Chun Chen †
Henry D. I. Abarbanel * ‡

ABSTRACT

A reservoir computer (RC) is a type of recurrent neural network architecture with demonstrated success in the prediction of spatiotemporally chaotic dynamical systems. A further advantage of RC is that it reproduces intrinsic dynamical quantities essential for its incorporation into numerical forecasting routines such as the ensemble Kalman filter—used in numerical weather prediction to compensate for sparse and noisy data. We explore here the architecture and design choices for a “best in class” RC for a number of characteristic dynamical systems. Our analysis points to the importance of large scale parameter optimization. We also note in particular the importance of including input bias in the RC design, which has a significant impact on the forecast skill of the trained RC model. In our tests, the use of a nonlinear readout operator does not affect the forecast time or the stability of the forecast. The effects of the reservoir dimension, spinup time, amount of training data, normalization, noise, and the RC time step are also investigated. Finally, we detail how our investigation leads to optimal design choices for a parallel RC scheme applied to the 40 dimensional spatiotemporally chaotic Lorenz 1996 dynamics.

1 Introduction

Chaotic nonlinear dynamical systems are deterministic, but have the property that even small uncertainties in initial conditions lead to exponentially growing errors that limit long term forecasts. A typical linear systems analysis (*e.g.*, a fourier transform) of such signals would conclude that the signal is noise, thus missing an essential feature of chaos. An illustrative example of this is the error made by Edwin Colpitts [1] in 1918—who designed and built a nonlinear variable frequency circuit—in identifying the irregularity in the output of his circuit as “noise.” 76 years later Michael Kennedy [2] recognized this feature to be deterministic chaos.

The original identification of chaotic motion of nonlinear systems was made by Poincaré in the 19th century while analyzing the three body problem of classical mechanics. A century later, the present era of understanding and insight into the properties of such systems was initiated by Edward Lorenz in 1963 [3]. Lorenz’s simplification of convection in the Earth’s lower atmosphere introduced the idea of deterministic, nonperiodic behavior. The study of these nonlinear dynamical systems typically requires the use of numerical integration methods, as analytic solutions to the differential equations are not generally available.

Many of the most advanced algorithms for predicting chaotic systems are based on the Bayesian optimization framework of data assimilation [4]. For example, the famous algorithm giving the solution to the linear filtering problem with discrete data proposed by Kalman in 1960 [5]—now called the Kalman filter—is one of the foundations for much of the progress in numerical weather prediction (NWP) [6] over the past several decades. While the canonical

*Department of Physics, University of California San Diego

†Cooperative Institute for Research in Environmental Sciences, University of Colorado Boulder, Boulder, CO 80309, Physical Sciences Laboratory, National Oceanic and Atmospheric Administration, Boulder, CO 80305

‡Marine Physical Laboratory, Scripps Institution of Oceanography, University of California San Diego, 9500 Gilman Drive, La Jolla, CA 92093, USA

Kalman filter assumes the linearity/linearization of the dynamics, there are also fully nonlinear developments of data assimilation [7] which have use beyond NWP such as in neurobiological settings [8].

One limitation of data assimilation is that the technique requires a detailed physical model of the underlying process—something that is not generally known for complicated physical systems. To address this issue, there has been an increasing drive towards replacing the physical model altogether by using a “data-driven model” as the predictor. Such data-driven models fall into the category of machine learning (ML); examples include autoregressive methods (ARMA), the multilayer perceptron (MLP), and the recurrent neural network (RNN) [9].

RNNs are a family of artificial neural networks where the internal state of the network depends explicitly both on the previous internal or ‘hidden’ state and an external driving signal. Thus, there is a natural ordering to the data—we call the order label time—which explicitly allows treatment of an RNN as a dynamical system. In contrast, other forms of artificial neural networks impose no ordering on the data and thus their natural mathematical description is as a function [10].

An RNN takes a D -dimensional input sequence $\mathbf{u}(t) \in \mathbb{R}^D$, evolves as the N -dimensional state $\mathbf{r}(t) \in \mathbb{R}^N$, and is asked to predict a value $\mathbf{y}(t) \in \mathbb{R}^Y$. Equation 10.5 in *Deep Learning* by Goodfellow, et al. [9], gives the general discrete time RNN evolution equation

$$\mathbf{r}(t) = F(\mathbf{r}(t-1), \mathbf{u}(t-1); \theta) \quad (1)$$

where θ represents the RNN hyperparameters. This shows the dependence of $\mathbf{r}(t)$ on the previous RNN state $\mathbf{r}(t-1)$ and external driving signal $\mathbf{u}(t-1)$. The output/readout layer of the network then ‘reads’ $\mathbf{r}(t)$ as containing the memory of previous inputs, through the recursive relation, to predict $\mathbf{y}(t)$. In other words, the output of the network $\hat{\mathbf{u}}(t) \equiv \mathbf{W}_{\text{out}}(\mathbf{r}(t))$ is a function \mathbf{W}_{out} of the RNN hidden state $\mathbf{r}(t)$. The operator \mathbf{W}_{out} defined here is a generic function, and could be implemented as a neural network. Later we will take \mathbf{W}_{out} to be a linear operator (a matrix). We can thus use $\hat{\mathbf{u}}(t) \sim \mathbf{y}(t) = \mathbf{u}(t)$ as a forecast; \sim will be used for “approximately equal.”

Long short-term memory (LSTM) networks, introduced by Hochreiter and Schmidhuber (1997) [11] and Gers et al. (1999) [12] have been one of the most useful varieties of RNN, as they are capable of maintaining a representation of long-term dependencies. This design helped LSTMs to circumvent a number of difficulties that plagued RNNs regarding exploding and vanishing gradients during optimization using backpropagation [13–15]. A simplified form of the LSTM was later developed by Kyunghyun Cho et al. in 2014 [16] called the Gated Recurrent Unit (GRU), which has a reduced set of trainable model parameters.

Our purpose here is to systematically explore an alternative form of RNN called a reservoir computer (RC) due to its demonstrated success in the prediction of chaotic dynamics. Through the examination of architectural and procedural choices for the RC design, we present “best in class” results for a wide range of simple dynamical systems. We then explore the method of localization for scaling the RC to larger spatiotemporally chaotic systems.

1.1 Background

RC [17, 18] is a general term that encompasses multiple research tracks that developed in different fields in the early 2000’s, in particular Echo State Networks (ESNs) [19–21] and Liquid State Machines (LSMs) [22]. It is a kind of RNN with a structure consisting of a fixed high-dimensional “reservoir” combined with a single trained output layer. The fixed nature of the reservoir reduces the size of the trained parameter space to a handful of ‘macro-scale’ parameters governing global properties, as well as ‘micro-scale’ parameters that comprise the matrix and vector elements. A key advantage of RC is that the final output or ‘readout’ layer can be formulated as a linear layer and then trained using linear regression. This reduction in the size of the searchable parameter space allows one to easily bypass the exploding/vanishing gradient problems that arise due to the use of backpropagation schemes [13–15].

In 2007, Schrauwen, Verstraeten, and Campenhout [23] drew the connection between the reservoir computing methods first proposed by Jaeger [19] and Maass [22] and the then state-of-the-art backpropagation-decorrelation learning rule for RNNs described by Steil [24]. For example, the recurrent learning rule of Atiya and Parlos (2000) [25] trains the output weights while the internal weights are only globally scaled up or down.

In addition to being easy to use and train, RCs have been shown to be very successful for chaotic time series prediction [26–29]. For NWP specifically, a major use case of multivariate chaotic time series prediction, Arcomano et al. (2020) [30] applied RC to a global atmospheric forecast model. Penny et al. (2021) [31] integrated RC with state-of-the-art data assimilation methods to estimate the forecast error covariance matrix, tangent linear model representation, and other components of the data assimilation process that are typically costly to either formulate or compute. Further computational advantage are afforded by implementing RCs on GPUs or even dedicated hardware [32].

We note that other ML methods have been explored for modeling dynamical systems, with varying degrees of success. While we cannot name them all, we highlight a few examples. Bocquet et al. [33, 34] used NNs and Convolutional NNs [9] to learn ODEs that could then be integrated using traditional numerical methods. As with RC, they have similarly been able to reconstruct the Lyapunov spectrum of simple systems. Convolutional LSTMs have been used in MetNet [35] to predict the evolution of precipitation. Eric Bollt [36] and Gauthier et al. [37] introduced a vector autoregression scheme that has shown remarkable success in the prediction of small systems; see, for example, Clark et al. [38] for an application in neurobiology. Data assimilation has been shown equivalent to feedforward Machine learning [39].

1.2 Why a Systematic Exploration of RC is Needed

Although published guides to the general use of RC exist [40] and there have been multiple studies on specific applications of RC with dynamical systems [41], there has been little guidance on best practices for implementing RC for forecasting chaotic dynamical systems. A practitioner must piece together design decisions by drawing on potentially conflicting information from multiple sources. We intend here to point the researcher or practitioner to strategies that have proven more successful, and to avoid those that we show to be ‘traps’ (*i.e.*, commonly published design decisions that have little or even negative impact on overall performance). Additionally, using ML to forecast chaotic dynamical systems presents a number of its own unique challenges when compared to standard practices in ML. There is a high potential for misunderstanding by practitioners and developers.

We aim to lay out in clear terms the problem being solved by RC for the specific task of forecasting chaotic dynamical systems, guidance on how to train and test the RC models, as well as the lessons we have learned in applying these methods over the last several years on the specifics of implementations for particular problems.

1.3 Structure

We proceed as follows:

1. Section 2 gives an overview of the RC implementation.
2. Section 3 explains how an RC works and why it has properties, such as the reproduction of error growth statistics, that may not be well documented in other ML models.
3. Section 4 gives details of the generative models and advice on how to properly test a new RC implementation.
4. Section 5 details the general training procedure and shows experimental results on experimental design choices.
5. Section 6 discusses scaling up the RC to high dimensional systems.

2 RC Definition

In the context of chaotic time series prediction we apply RC to the temporal ML problem where the task is to predict a time series $\mathbf{u}(t)$ generated from an autonomous dynamical system

$$\dot{\mathbf{u}}(t) = f_{\mathbf{u}}(\mathbf{u}(t)), \quad (2)$$

where the dot denotes a time derivative and $f_{\mathbf{u}}$ denotes the equations of the dynamical system. The dimension of the input system is D . In general $f_{\mathbf{u}}$ is not known, nor does it need to be known for RC to be used. We do assume that $f_{\mathbf{u}}$ exists. The system $f_{\mathbf{u}}$ can describe any physical or mathematical process and therefore these techniques can be applied in fields such as biology, hydrology, meteorology, oceanography, economics, chemistry, and many others.

Here we use an RC model with the specific form of a simplified ‘Elman’ style RNN [42]. The RC consists of three layers: an input layer \mathbf{W}_{in} , the reservoir itself, and an output layer \mathbf{W}_{out} . The reservoir is composed of N nodes that are generally acted upon with simple nonlinear elements *e.g.*, tanh activation functions. The nodes in the network are connected through an $N \times N$ adjacency matrix \mathbf{A} , chosen randomly to have a connection density ρ_A and non-zero elements uniformly chosen between $[-1, 1]$, scaled such that the maximal eigenvalue of \mathbf{A} is a number denoted the spectral radius (ρ_{SR}).

The input layer \mathbf{W}_{in} is an $N \times D$ dimensional matrix that maps the input signal $\mathbf{u}(t)$ from D dimensions into the N dimensional reservoir space. The elements of \mathbf{W}_{in} are chosen uniformly between $[-\sigma, \sigma]$. The matrix \mathbf{W}_{out} provides the mapping from the RC reservoir state to the system state as $\dot{\mathbf{u}}(t) = \mathbf{W}_{\text{out}}\mathbf{r}(t)$. The reservoir state $\mathbf{r}(t)$

can be viewed as embedding the information given in the time series $\mathbf{u}(t), \mathbf{u}(t-1), \dots, \mathbf{u}(0)$ in a higher $N \gg D$ dimensional time delayed space, consistent with Takens theorem of time-delay embedding [43–45]—see Hart et al. for a more in depth discussion [46]. Takens theorem implies that information from unobserved states of the dynamics is contained in the time delay signal, thus allowing an RC to operate even when not all the information is measured.

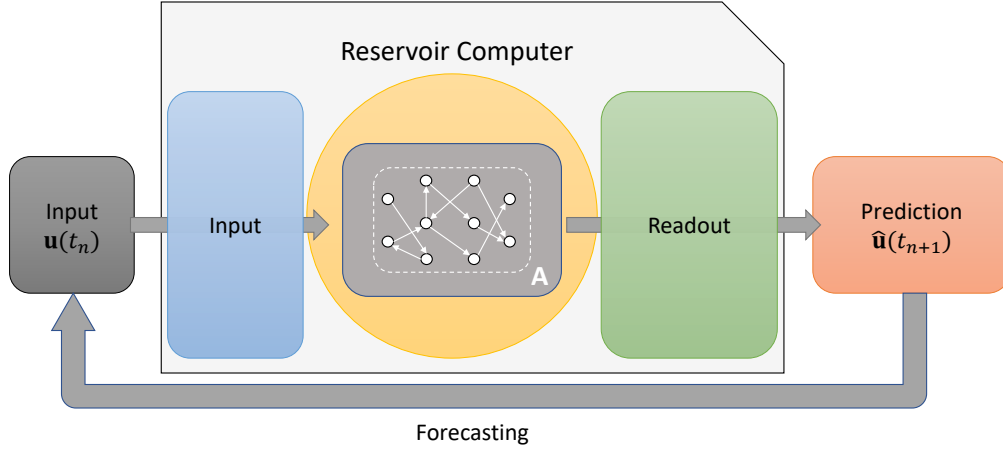


Figure 1: Schematic of information flow in the RC. The input data $\mathbf{u}(t_n) \in \mathbb{R}^D$ is mapped through the input layer $\mathbf{W}_{\text{in}} \in \mathbb{R}^{N \times D}$ into the reservoir $\mathbf{r} \in \mathbb{R}^N$ where the nonlinear activation function (e.g. \tanh) is applied and mixed through a fixed (*i.e.* not trained) adjacency matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$. The readout layer $\mathbf{W}_{\text{out}} \in \mathbb{R}^{D \times N}$ is typically trained using linear regression and gives the one-step-ahead prediction. Longer-term forecasts are achieved by cycling a feedback of the output back to the input.

2.0.1 Training \mathbf{W}_{out}

The output layer $\mathbf{W}_{\text{out}} \in \mathbb{R}^{D \times N}$ is a matrix such that $\mathbf{W}_{\text{out}}\mathbf{r}(t) \equiv \hat{\mathbf{u}}(t) \sim \mathbf{u}(t)$ chosen during the training phase. The elements of \mathbf{W}_{out} are what we call “micro-scale” parameters—in contrast to the “macro-scale” parameters ($\sigma, \rho_{SR}, \rho_A, N, \dots$) that set overall properties of the RC by adjusting multiple matrix elements at once. An analogy would be that setting the macro-scale parameters is similar to specifying the overall temperature or pressure of a gas, the positions and velocities of each individual particle need not be taken into account, greatly simplifying the problem. The micro-scale parameters that comprise the elements of \mathbf{W}_{out} can be trained at minimal cost using linear regression.

The reservoir themselves can either be posed in mapping or differential form. The differential form is

$$\dot{\mathbf{r}}(t) = f_r^d(\mathbf{r}(t), \mathbf{u}(t)) = \gamma[-\mathbf{r}(t) + \tanh(\mathbf{A}\mathbf{r}(t) + \mathbf{W}_{\text{in}}\mathbf{u}(t) + \sigma_b \mathbb{1})] \quad (3)$$

with γ the inverse time constant, σ_b the input bias, $\mathbb{1}$ a vector of 1’s, and $t \in \mathbb{R}$.

The mapping form is

$$\mathbf{r}(t+1) = F_r^d(\mathbf{r}(t), \mathbf{u}(t)) = \alpha \tanh(\mathbf{A}\mathbf{r}(t) + \mathbf{W}_{\text{in}}\mathbf{u}(t) + \sigma_b \mathbb{1}) + (1-\alpha)\mathbf{r}(t) \quad (4)$$

with $t \in \mathbb{Z}$. The notation is as follows; f describes the differential form, while F describes the mapping form. F_r^d denotes the ‘driven’ RC, as opposed to ‘autonomous’ (F_r^a), \mathbf{r} is the reservoir state vector, and \mathbf{u} is the state produced by the true dynamics, which is provided as input data. Eqs.(3, 4) are denoted the “driven RC” because $\mathbf{u}(t)$ enters directly into the reservoir equation of motion.

The mapping form is equivalent to the differential form when the latter is integrated using the Euler method. This can be shown by setting $\alpha = \gamma\Delta t$ and discretizing Eq.(3). While both the mapping and differential forms can be used interchangeably, this detail must be acknowledged as the integration method may affect the RC accuracy. Alternative forms of the RC have been suggested—see [41] for a review—but so far we have not found any advantages in using those forms.

We determine the matrix elements in \mathbf{W}_{out} by optimizing the loss

$$\text{minimize } \|\mathbf{W}_{\text{out}}\mathbf{r} - \mathbf{u}\|^2 + \beta\|\mathbf{W}_{\text{out}}\|^2, \quad (5)$$

where \mathbf{r} is a matrix containing $\mathbf{r}(t)$ and \mathbf{u} is a matrix containing $\mathbf{u}(t)$ for all t in the training dataset, and β is a Tikhonov-Miller regularization hyperparameter [47]. This is also known as ridge regression and is implemented through the `gesv` LAPACK routine from Intel MKL [48]—the results from other solvers are given in the appendix. The solution is

$$\mathbf{W}_{\text{out}} = \mathbf{u}\mathbf{r}^T(\mathbf{r}\mathbf{r}^T + \beta\mathbb{I})^{-1}, \quad (6)$$

where \mathbb{I} is the $N \times N$ identity matrix. There is an option to add nonlinearity to the readout operator, which is addressed in section 5.5.

2.0.2 Forecasting

When forecasting, the external forcing $\mathbf{u}(t)$ is replaced by a feedback loop $\mathbf{W}_{\text{out}}\mathbf{r}(t)$. Thus the equation for the ‘‘autonomous RC’’ is

$$\mathbf{r}(t+1) = F_r^a(\mathbf{r}(t)) = \alpha \tanh[\mathbf{A}\mathbf{r}(t) + \mathbf{W}_{\text{in}}\mathbf{W}_{\text{out}}\mathbf{r}(t) + \sigma_b\mathbb{1}] + (1 - \alpha)\mathbf{r}(t). \quad (7)$$

One can see that $\mathbf{u}(t)$ is replaced by $\mathbf{W}_{\text{out}}\mathbf{r}(t)$ and so in prediction we have $F_r^a(\mathbf{r}(t))$ being an autonomous dynamical system, allowing multi-timestep forecasts via recursion.

Parameter	Description
ρ_{SR}	Spectral radius (max eigenvalue) of the adjacency matrix \mathbf{A}
ρ_A	Density of the adjacency matrix \mathbf{A}
N	Degrees-of-Freedom of the reservoir
α	Leak rate (time constant) of the reservoir computer
σ	Strength of input signal
σ_b	Strength of input bias
β	Tikhonov regularization parameter

Table 1: Global scalar parameters of the RC. These parameters may either be treated as ‘macro-scale’ model parameters to be optimized during training, or may be used as hyperparameters and tuned manually.

3 RC Theory

3.1 Attractor Reconstruction and Lyapunov Exponents

The Lyapunov spectrum, composed of a system’s Lyapunov Exponents (LE), is a tool to probe the fundamental characteristics of a dynamical system [44, 49]. For example, one definition of a chaotic system is that there is at least one positive LE [50]. LEs can also be used to estimate the rate of entropy production (Kolmogorov-Sinai Entropy) and the fractal or information dimension [50].

The LEs describe how two points x_1 and x_2 , separated initially by a distance $\delta x_0 = |x_1 - x_2|$, evolve in time with respect to one another. More specifically,

$$\delta x(t) \approx \delta x_0 e^{\lambda_1 t}$$

with λ_1 being the largest LE⁴. Therefore if λ_1 is positive, over time the two initial conditions that at first only differ by a small amount will diverge away from one another exponentially fast in the linearized regime—the definition of chaos. We give a more rigorous definition of LEs in the appendix along with a discussion of how to calculate them—Appendix A.

When the LEs of the **autonomous** Eq.(7) RC match the LEs of the input data generated from the dynamical system f_u then the RC is said to have ‘‘reconstructed’’ the attractor [27]. A reconstructed attractor is strongly correlated with properties such as the physicality and robustness of forecasts [51]. The LEs of f_u can be estimated from the data with no knowledge of the dynamics [44].

⁴it is important to note that this is only actually true when δx lies exactly along the direction λ_1 , however for large t the largest exponent will dominate.

To calculate the LEs of the autonomous reservoir we note that the Jacobian/linear propagator of Eq.(7) is

$$\mathbf{DF}_r^a(n) = \alpha \cdot \text{diag}(\mathbb{1} - \tanh[\mathbf{W}\mathbf{r}(n) + \sigma_b \mathbb{1}]^2) \mathbf{W} + (1 - \alpha) \mathbb{I} \quad (8)$$

with diag denoting a diagonal matrix and the constant matrix $\mathbf{W} = \mathbf{A} + \mathbf{W}_{\text{in}} \mathbf{W}_{\text{out}}$. This propagator may be used in the algorithms given in Geist et al., [52] using the procedure given in [27, 29] to calculate the LEs.

3.2 Generalized Synchronization

As described in detail by [27, 51], the RC operates under the paradigm of generalized synchronization (GS) [53]. The definition of GS is: for a drive system $\mathbf{u} \in \mathbb{R}^D$ and response system $\mathbf{r} \in \mathbb{R}^N$, they are synchronized if there exists a function ψ such that $\mathbf{r} = \psi(\mathbf{u})$ [53]. Intuitively this implies that the dynamics of the response system are entirely predictable from the history of the input; in RC, for a contracting system, this property is related to the “echo state property” [19] and “fading memory” [54]. The requirement to find a matrix \mathbf{W}_{out} such that $\mathbf{W}_{\text{out}} \mathbf{r}(t) = \mathbf{u}(t)$ is tantamount to finding a linear approximation to the local inverse of the generalized synchronization function—we call this statement predictive generalized synchronization [51]. We would like to note that the statement of the existence of the inverse of ψ is in general not a global, but a local property due to the necessity of ψ being differentiable [55] for ψ^{-1} to exist in the neighborhood of a particular point. Grigoryeva et al. [56, 57] details general conditions for differential GS to exist in an RC. When ψ does not meet these criteria then the RC is untrainable.

The concept of GS is useful for visualizing the operation of the RC. When $\mathbf{u}(t)$ and $\mathbf{r}(t)$ are synchronized, the combined state \mathbb{R}^{N+D} will lie on an invariant *synchronization manifold* \mathcal{M} [58]. This manifold is generally low dimensional [51] and so the dynamics of the RC are taking place on this low dimensional structure in phase space. In addition, the concept of synchronization shows that there is a certain amount of “spinup” time needed for the transient initial conditions to contract to the synchronization manifold. The amount of time needed for spinup is directly related to the strength of the conditional Lyapunov exponents (CLEs) [59] of the synchronization manifold—see appendix A for CLEs. The statement of stability of the motion on the manifold [58, 60] is that the contraction normal to the manifold must be larger than the contraction tangential to it [61], thus giving a statement of the stability of the RC forecast. To relate the concept of GS directly to the echo state property we note that for synchronization to occur between the data and the RC, the CLEs must be negative [51, 58]. Negative CLEs, by definition, imply that different initial conditions of the RC contract towards a single uniform trajectory with the initial conditions forgotten. This is precisely the statement of the echo state property, connecting the dynamical systems view of RC with the information processing formalism. The existence of GS, and by extension the echo state property, can be easily verified by the auxiliary method [62] as explored in [51].

As a last note, it has been observed that RCs often work best when at the edge of stability [63–65], although this may not always be the case [66] and indeed we observe that the optimal spectral radius can be quite close to 0; nevertheless, the edge corresponds to the largest CLE being ~ 0 [51].

4 Data Generation for Numerical Experiments

We begin with a discussion of the training and testing data used in the following numerical experiments. The use of input data is different for dynamical systems forecasting than for the typical ML use case. Typically in ML there are standard training and testing datasets *e.g.*, MNIST [67], that one can acquire in order to directly test newly proposed models against published results from previous models [68]. For dynamical systems forecasting we instead have certain standard models from which one can generate data. Therefore it is important to make sure one is observing best practices when constructing ones own training, validation and testing sets.

We would like to construct independent training, validation and testing data for training and testing our ML models. In order to be confident in our results, the validation/testing data must be independent from both one another and the training data. Furthermore, we would like the testing data to sample as much of the phase space of f_u as possible, so that we can be confident that the RC generalizes to parts of the input with different properties than which it has been trained on. All data have been generated with these principles in mind Fig.(2).

Additionally, we report forecast times in terms of the Lyapunov timescale of the input dynamical system. This timescale $\tau_\lambda = 1/\lambda_1$, gives the natural timescale of error growth in the system and is therefore a useful measure of how “good” the forecast is. Not even a perfect model can predict a chaotic dynamical system forever—numerical errors will eventually cause divergence—and therefore τ_λ gives us a metric of a reasonable timescale for prediction.

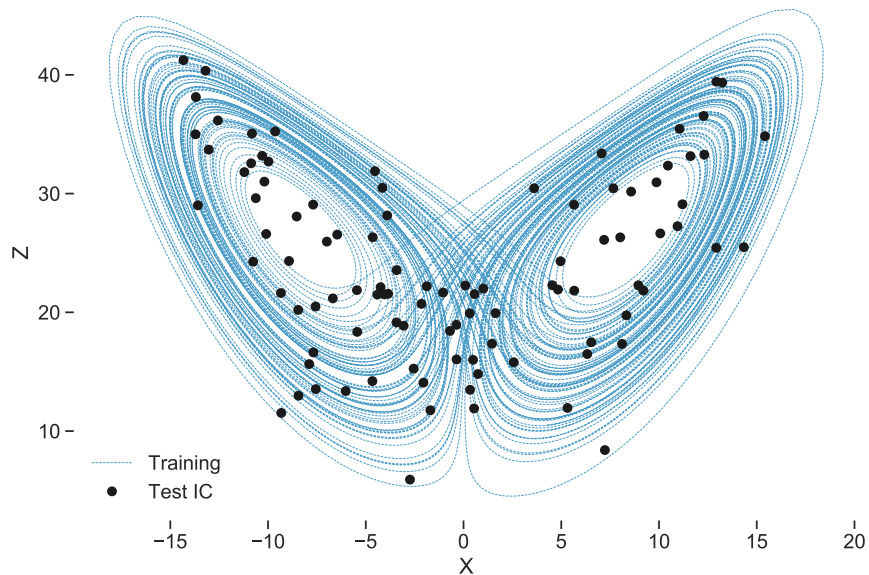


Figure 2: Training data (blue) and test forecast initial conditions (black) for the L63 system. For the training data it is important that the data cover most parts of the input attractor. This ensures that the RC does not overfit a local section of the dynamics and that it can generalize. The difference in dynamics over the attractor can be formalized by the finite time Lyapunov exponents (FTLE) [44, 69], which denote the localized error growth rates. These can be positive or negative for the L63 system depending on the initial condition. Likewise for the testing data, the initial conditions for the test forecasts should widely sample the attractor. It is important that the testing data be long term forecasts and not one-step forecasts in order to capture the broadband frequencies of the chaotic dynamics.

4.1 Generative Models

The dynamical models used for data generation have been chosen for their applicability to certain use cases as well as their ability to test particular abilities of reservoir computing. The Rossler system and Colpitt’s oscillator are both 3 dimensional systems with long time scale dynamics—*i.e.*, τ_λ large, slow error growth rates. The Colpitt’s oscillator is particularly interesting because its dynamical equations of motion include an exponential term, making it the most nonlinear of the models studied here. The degree of nonlinearity of the dynamics can create greater difficulty for prediction with a linear readout operator. The Lorenz attractor (L63) is another 3 dimensional system but with shorter time scale dynamics than Rossler or Collpitts. The Lorenz 1996 system can be defined with arbitrary size so we can use it to look at how our RC scales with system dimension. Finally the Climate Lorenz Model (CL63) is composed of three L63 systems coupled together with multi-timescale dynamics. RCs tend to focus on a particular time scale [17] so it is important to study multi-timescale dynamics for certain applications such as in a weather prediction context [4, 70]. See the appendix for equations and parameters.

Dataset	Largest LE λ_1	Time Scale $\tau_\lambda = 1/\lambda_1$
Rosler	0.065	15.4
Colpitts	0.07	14.3
L63	0.9	1.1
L96-5D	0.4	2.5
L96-10D	1.1	0.9
CL63	0.9	1.1

Table 2: Dataset timescales. All dynamical systems we consider are made dimensionless, so the “time” here is number of dimensionless steps into the future.

Many real world problems have only partially observed dynamics, making the ML problem much more difficult. RC has an intrinsic memory due to the recurrence relation in the update rule [20]—giving it a natural time delay embedding—and propagating information from the unmeasured states to the measured ones. Thus, even though ML

techniques cannot extrapolate to unseen variables, it can use that information to predict forward in time in those variables it has been trained on. When there are too few observations to constrain the model then one can integrate the RC with data assimilation algorithms [4, 31] that facilitate the transfer of information from sparse, noisy and partial observations to unseen dynamics, although this is much more computationally intensive. In weather prediction settings there is plentiful retrospective analysis data that uses physical models to provide a historical reconstruction of the full state of the atmosphere. For simplicity, in this work we assume that all state variables have been measured or reconstructed, allowing us to train the RC on the full state of the system. We expect that many of the lessons learned in the full state case will generally carry over to the partially observed case.

4.2 Forecast Testing Metric

A common ML benchmark for time series prediction is the one-step prediction mean square error Eq.(5). Indeed this is exactly how we train the matrix elements in \mathbf{W}_{out} to obtain the linear map from $\mathbf{u}(t) \rightarrow \mathbf{r}(t) \rightarrow \mathbf{u}(t+1)$. As a method for testing, however, one-step prediction is not an appropriate metric; it overemphasizes the high frequency modes in the data at the expense of the long-term stability of the RC forecast.

As shown by Platt et al. [51], the metric that best represents the ideal behavior of the RC—good short-term predictions coupled with “physical” (the forecast RC stays on the synchronization manifold \mathcal{M}) long-term forecasts—is best represented by the reproduction of the input system’s LEs. In practice the LEs can be difficult to calculate from high-dimensional experimental data and computationally expensive for the RC. Therefore, a proxy for the reproduction of the LEs has in practice been long-term forecasts [51].

Our standard metric for forecast time is the valid prediction time (VPT) [29]. The VPT is the time t when the accuracy of the forecast exceeds a given threshold. For example,

$$\text{RMSE}(t) = \sqrt{\frac{1}{D} \sum_{i=1}^D \left[\frac{u_i^f(t) - u_i(t)}{\sigma_i} \right]^2} > \epsilon, \quad (9)$$

where D is the system dimension, σ is the long term standard deviation of the time series, ϵ is an arbitrary threshold, and u^f is the RC forecast. For the results shown in subsequent section a total of 200 test initial conditions are used to report a distribution of VPTs with ϵ set arbitrarily to 0.3. [29] use $\epsilon \sim 0.5$ in their reported results.

5 RC Training

Training the RC model can be separated into training the macro-scale parameters—*e.g.*, spectral radius, leak rate—and the micro-scale parameters *i.e.*, the elements of \mathbf{W}_{out} . Much of the early guidance on RC training [40] included advice on the intuition behind hand-tuning the macro parameters for a particular problem as hyperparameters. For instance, a widely cited guideline is that the spectral radius must be below 1 but that is based on many assumptions that do not necessarily hold for a particular RC application [71–73]. Modern optimization algorithms and increased computational power have decreased the need to hand-tune all of these macro parameters as hyperparameters. Instead, they can be incorporated directly into the objective function for training via optimization. For intuition on the effect of these parameters on the RC we refer the reader to Lukoševičius [40].

We employ a two step optimization approach, described by [28, 31] where one first fixes the macro-scale parameters, obtains \mathbf{W}_{out} through linear regression Eq.(6) and then tests the completed RC on a series of long forecasts. For the macro loss function we have had success using the scaled error

$$\mathcal{L}_{\text{macro}} = \sum_{i=1}^M \sum_{t=t_i}^{t_f} \|u^f(t) - u(t)\|^2 \exp\left\{-\frac{t-t_i}{t_f-t_i}\right\}, \quad (10)$$

with M being the number of forecasts used for comparison. We reiterate that the training for the macro-scale parameters does not use the one-step-ahead prediction error because that is not a good proxy for the reproduction of the LE spectrum. The exponential weighting in Eq.(10) is added to approximately offset the exponentially growing error in the forecast due to positive LEs.

Penny et al. [31] (Fig.12) examined the effect of increasing numbers of M on the cost function landscape and showed that increasing M smooths out the loss function landscape and enables better convergence to the global minimum. However, a compromise is necessary as increasing M will substantially increase computational complexity. Griffith et al. [28] showed that using $M = 1$ causes the RC to be overly sensitive to unstable sets in the dataset. We have

empirically found that using 15-20 or so forecasts is generally enough for these simple models, provided that those forecasts are made at statistically independent points. Ideally the data would well sample the input system attractor, but in practice that cannot always be the case.

We set the random seed for the reservoir instantiation during the optimization to reduce the noise in the optimization function. After training, however, we have not found the random seed leading to substantially different prediction times over a large number of test samples—see Fig 7 in [51] for an illustration. This does not mean that the prediction for any individual test will be exactly the same but that on aggregate the randomness does change the mean of the distribution substantially. This result may be different in the “small” reservoir regime of $N < \sim 100$.

5.1 Bayesian Optimization

Rather than hand-tune each macro-scale parameter as a hyperparameter, we suggest instead using an optimization procedure such as the two step optimization routine described above. While any global search algorithm can be used *e.g.*, differential evolution, we use a Bayesian optimization technique using surrogate modeling. Bayesian optimization is a protocol for optimizing expensive nonlinear functions that works by “minimizing the expected deviation from the extremum” of a target loss function [74]. We have found success with the efficient global optimization algorithm of Jones et al., [75] as implemented by [76]. This algorithm evaluates the loss function over a number of sampled points and then a Gaussian process regression is fit to the surface produced. One can then optimize over this interpolated “surrogate” model to identify promising regions of parameter space and then iterate by reevaluating the loss function in these regions. Given that we need only train a few macro-scale parameters, this surrogate optimization technique renders the optimization problem computationally tractable.

With these algorithms, one can jettison the hand-tuning of hyperparameters and simply optimize all the macro-scale RC parameters for a given problem. This procedure produces good performance for a wide range of simple dynamical models Fig.(3) in a reasonable amount of time— < 20 minutes for all the simple models on a personal desktop computer, and sometimes as low as 1-2 minutes for small (~ 200) reservoirs. This is much faster than the commensurate grid search over the parameters since the sensitivity of the RC to some of the parameters requires a very fine search.

We will now go into greater detail on the choices that must be made outside of the optimization routine (by definition hyperparameters) and how they can affect the solution to the RC problem. The experiments are conducted by fixing/-varying a particular hyperparameter or macro-scale parameter from table 1 and then optimizing over the rest of the macro-scale parameters. It will be stated if we are not optimizing over the un-varied parameters.

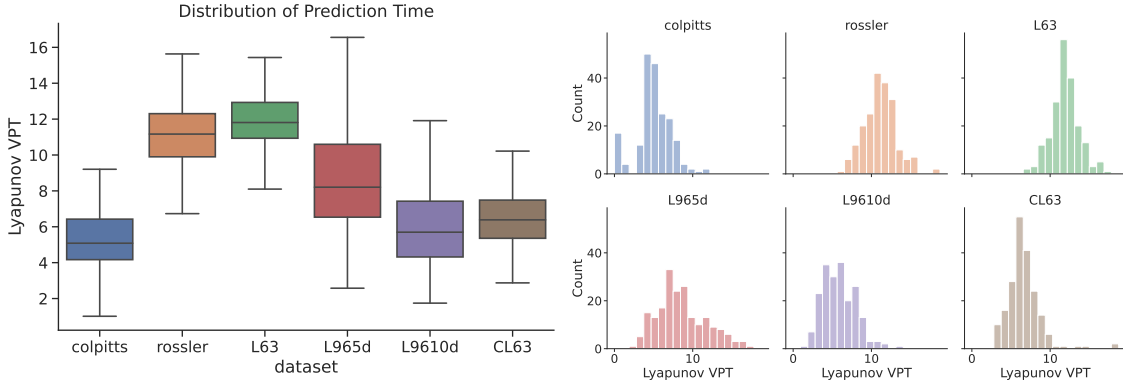


Figure 3: **(left)** Valid prediction time for the best fit parameters of the RC over different datasets. Note the distribution of forecast times. The RC predicts well on all the models given to it. Results are shown for $N = 2000$, results for $N = 250$ are shown in Fig.(4).

(right) Histogram view of the distribution of VPT for the different models. The distributions are close to Gaussian but with heavier tails.

5.2 Reservoir Dimension

The reservoir dimension N of the RC is related both to the memory capacity of the network [20, 40] as well as the high dimensional projection of the nonlinear data from $D \rightarrow N$. Larger reservoirs tend to produce more accurate forecasts

with longer VPTs than smaller networks. However, there are diminishing returns where ever larger reservoirs are needed for ever smaller improvements in predictability. For example, consider the L63 system in Fig.(4). Using a reservoir size of just 250 nodes, we see a VPT only a couple Lyapunov time scales less than when using a reservoir of size 2000. Therefore if our application only requires a few Lyapunov time scales of prediction skill, then it is possible to use a much smaller and more computationally efficient RC.

Because optimization routines are far more computationally intensive than forecasting with the RC, it is practical to ask whether one can optimize/train using a smaller reservoir and then scale up to larger reservoir to generate a forecast. The answer is that one can indeed train a small reservoir and then use those same macro-scale parameters with a larger reservoir to see substantial improvement in prediction skill. This strategy was used by [31]. However, an RC optimized at the larger size will tend to produce more accurate forecasts. If computational resources allow, it is appropriate to perform a final optimization using the reservoir size that will be used for the production RC. The left side of Fig.(4) shows the effect of increasing the reservoir dimension without re-optimizing parameters. The RC performance tends to saturate as the reservoir dimension increases.

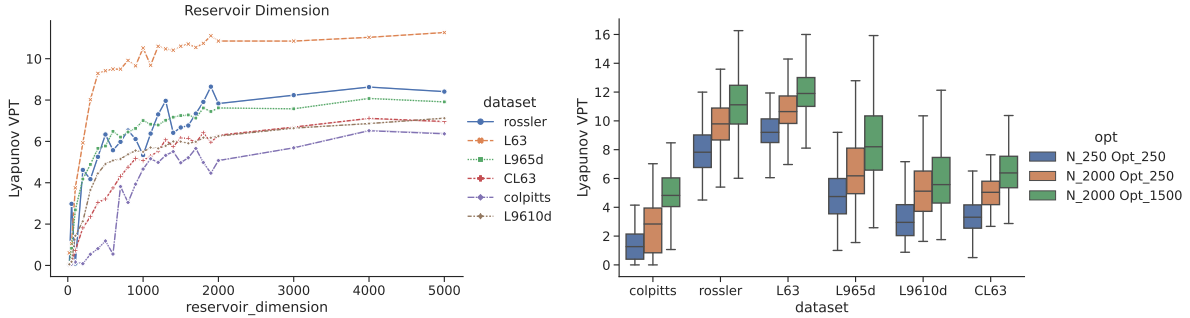


Figure 4: **(left)** The effect of increasing reservoir dimension without reoptimization of the global parameters. RCs optimized for $N = 2000$. There is a clear minimum sized RC for each of the dynamical systems studied as well as a point where the law of diminishing returns makes ever smaller increases in prediction ever more costly.

(right) VPT for a 3 different RCs 1) small $N = 250$ RC optimized at $N = 250$ 2) an RC with identical global parameters to 1, but with a dimension scaled up to $N = 2000$ 3) an RC with $N = 2000$ optimized for $N = 1500$. The optimization of the RC parameters at a larger reservoir size makes the prediction substantially better.

5.3 Spinup Time

The spinup time is the time it takes for a trained RC to converge from its initial condition (usually either set to zero or randomly chosen) onto the synchronization manifold to which it is driven by the input data. The number of steps needed for spinup is related to:

1. How far the initial condition of the RC is from the synchronization manifold.
2. The CLEs of the generalized synchronization manifold; these are related to the ‘‘echo state property’’ and the spectral radius of the adjacency matrix \mathbf{A} .
3. The RC approaches the synchronization manifold asymptotically so while the error tolerance is also important, it probably does not need to converge to within 10^{-12} . Convergence during spinup and error growth during prediction are both governed by $\delta x(t) \approx \delta x_0 e^{\lambda t}$ with λ giving the negative CLEs for spinup or the positive LEs for forecasting.

In practice it is easy to tell empirically how much time is needed—see Fig.(5)—usually for these small models only $\mathcal{O}(10)$ time steps are necessary. For larger systems the spinup time can be significantly higher.

5.4 Input Bias

The input bias σ_b can have a significant impact on RC prediction skill, but it has consistently been left out of most published RC studies focused on forecasting chaotic dynamics. We see in Fig.(6) that if we do not include this term in the RC when optimizing macro-scale parameters then for almost all models the forecast skill is negligible. The input bias sets the extent of the ‘nonlinearity’ in the reservoir; $\sigma_b = 0$ corresponds to the reservoir operating mostly in the

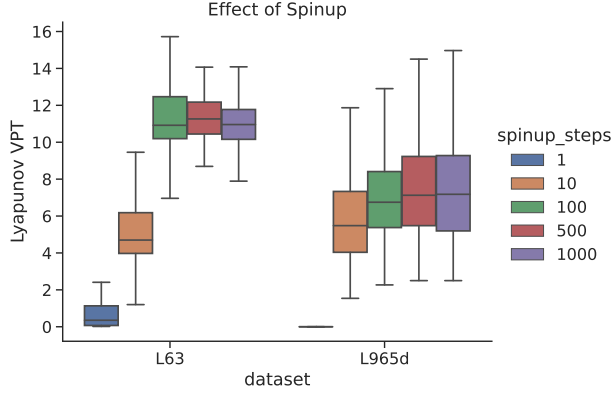


Figure 5: Effect of setting the number of spinup steps on the VPT of the RC for L63 and L96-5D. Longer spinup times reduce the initial error of the forecast as the RC and input data synchronize together.

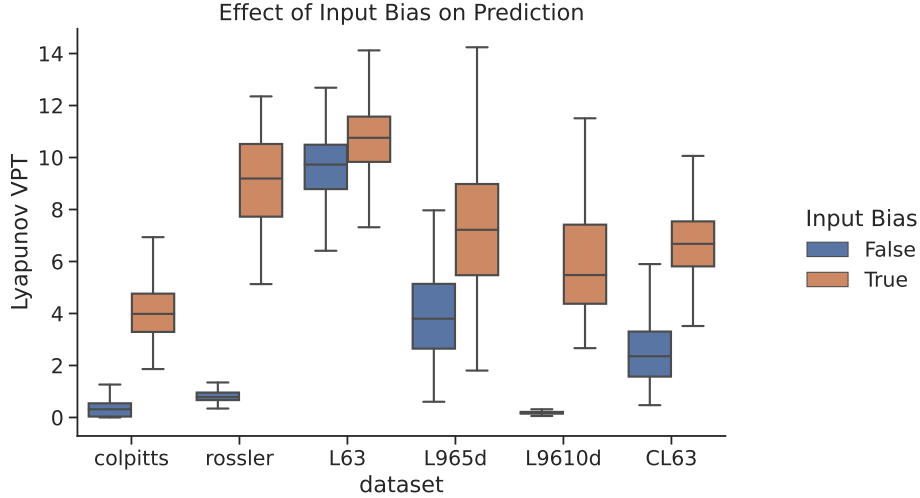


Figure 6: There are two sets of parameters: 1) no input bias 2) input bias. Both sets of parameters are optimized separately but with the same reservoir size. The readout is linear for both. The input bias is a constant σ_b , which is optimized Eq.(4). σ_b sets the nonlinear regime of the \tanh . When $\sigma_b = 0$ the optimization is not able to find any parameters that work for many of the models. However, when σ_b is optimized all models produce accurate and reliable forecasts.

linear part of the \tanh function around 0 when the input data is centered at 0, while a high bias term corresponds to the RC operation being in a more nonlinear part of the regime.

To reiterate the point, the inclusion and optimization of the input bias was found to be extremely necessary for forecasting almost all the dynamical models studied. The inclusion of this term should be made standard for this kind RC architecture.

5.5 Readout Functions: Linear, Extended, Quadratic

We generalize the readout $\mathbf{W}_{\text{out}}\mathbf{r}$ to $\mathbf{W}_{\text{out}}Q(\mathbf{r})$ for some function Q . An example in the literature is by Lu et al., [27] who use a quadratic readout function $Q(\mathbf{r}) = [\mathbf{r}(t), \mathbf{r}^2(t)]$. As another example, [40] recommended an extended output where $Q(\mathbf{r}, \mathbf{u}) = [\mathbf{r}(t), \mathbf{u}(t-1)]$ so the previous input is fed directly into \mathbf{W}_{out} . We have tested a number of options and found that the readout does not have much impact on the quality of the forecasts made by the RC Fig.(7) as long as the parameters are well optimized and the input bias σ_b is not set to 0. While these results seemingly contradict those reported and used in the previously cited studies, we note that when the bias term is not included—see Fig.(6)—then changing the readout layer does indeed improve the predictions. We speculate that the readout in this case is injecting

a certain amount of nonlinearity, essential for good predictions, into a system that is operating around the linear regime of the activation function.

In addition to the injection of nonlinearity into the RC, the improved performance of the RC with input bias may be due to the breaking of the symmetry of the reservoir equations. Herteux et al. [77] showed that for the simple Lorenz 63 system, the symmetry of the equations leads to the formation of a “mirror attractor” that causes divergence of the RC off the synchronization manifold and onto an alternate trajectory. Adding either input bias or a quadratic readout resolved the issue. These results were extended by [78] to show that, especially for small reservoirs, breaking the symmetries of the input data can be extremely important for prediction. Therefore, while we do not see any change in prediction for the different readout functions once input bias is added, perhaps for small reservoirs ($< \sim 100$) a nonlinear readout could become important.

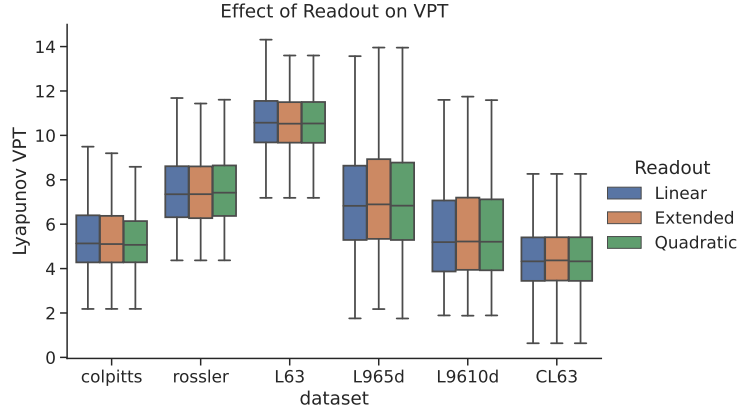


Figure 7: Predictions for 3 different kinds of readout methods. Linear: $\hat{\mathbf{u}}(t) = \mathbf{W}_{\text{out}}\mathbf{r}(t)$, Extended: $\hat{\mathbf{u}}(t) = \mathbf{W}_{\text{out}}[\mathbf{r}(t), \mathbf{u}(t-1)]$, Quadratic: $\hat{\mathbf{u}}(t) = \mathbf{W}_{\text{out}}[\mathbf{r}(t), \mathbf{r}^2(t)]$.

5.6 Amount of Training Data

The data should be separated into the M long forecasts that are used to compute the loss $\mathcal{L}_{\text{macro}}$ and the data used to train \mathbf{W}_{out} in Eq.(6). If needed these datasets can also overlap. More and longer test forecasts for the global loss will help guarantee 1) the generalizability of the RC to segments of the data the RC might not have seen, 2) stability of the trained RC, and 3) the optimality of the global parameters found during training. The amount of data that is used for the linear regression governs the accuracy of the linear map \mathbf{W}_{out} . Experiments indicate diminishing returns, where increasing the size of the training data leads to diminishing improvements in forecast skill Fig.(8).

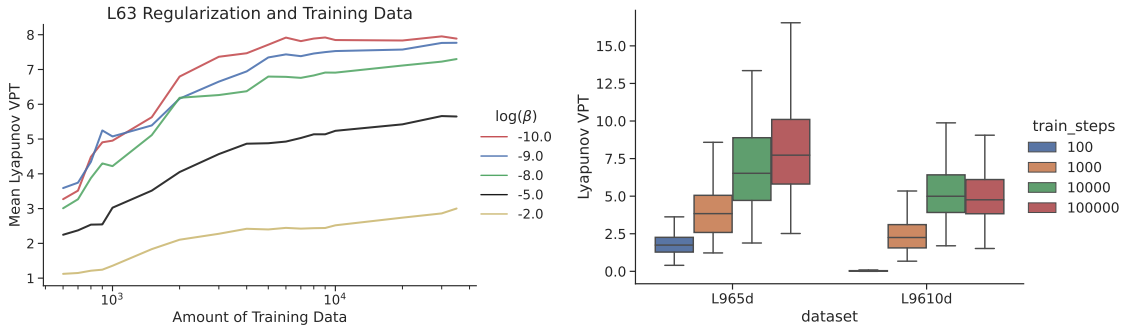


Figure 8: (left) L63, RC $N=250$ for different values of regularization without reoptimizing all the parameters. Diminishing returns with increasing amounts of data. (right) Here we re-optimize the RC for each amount of training data. We see a clear reduction in the rate of increasing skill as the amount of training data increases.

5.7 Normalization

Normalization is a standard practice in ML. Applying normalization is usually among the first suggestions when analyzing a new dataset [9]. We must emphasize, however, that when forecasting dynamical systems—and specifically for the systems and RC architecture studied here—the standard methods for normalizing datasets can have deleterious effects. To illustrate, we introduce two normalization schemes,

1. Normalize each variable separately by subtracting the mean and dividing by the max/min

$$u_i^{\text{norm}} = \frac{u_i - \text{mean}(u_i)}{\max u_i - \min u_i}; i \in [1, 2, \dots, D].$$

2. Normalize the variables by the joint mean and max/min of the variables

$$u_i^{\text{norm}} = \frac{u_i - \text{mean}(\mathbf{u})}{\max \mathbf{u} - \min \mathbf{u}}; i \in [1, 2, \dots, D].$$

We assume that the variables in the dynamical system have been nondimensionalized. The statistical functions mean, max, and min are calculated from all the training data and thus represent “climatological” statistics. The VPTs resulting from both normalization schemes are shown in Fig.(9).

As shown in figure 9, normalization scheme 2 vastly outperforms scheme 1. Our hypothesis is that normalizing each variable separately—particularly subtracting the mean—destroys the cross-variable information necessary to reconstruct the relationships in the dynamical system. There can be a significant amount of information stored in the relationships between the variables as well as their magnitudes. The second scheme compensates for the normalization via σ and σ_b to preserve all the information in the data while the first scheme destroys that information. Instead of dividing by the max/min we also divided by the standard deviation of the variables with the same qualitative results.

It is possible that we could recover the prediction skill by a skillful selection of \mathbf{W}_{in} . If we optimized the inputs (σ) for each column of \mathbf{W}_{in} separately as well as σ_b , that would help the RC to compensate for the normalization. However, this introduces additional global parameters that must be optimized.

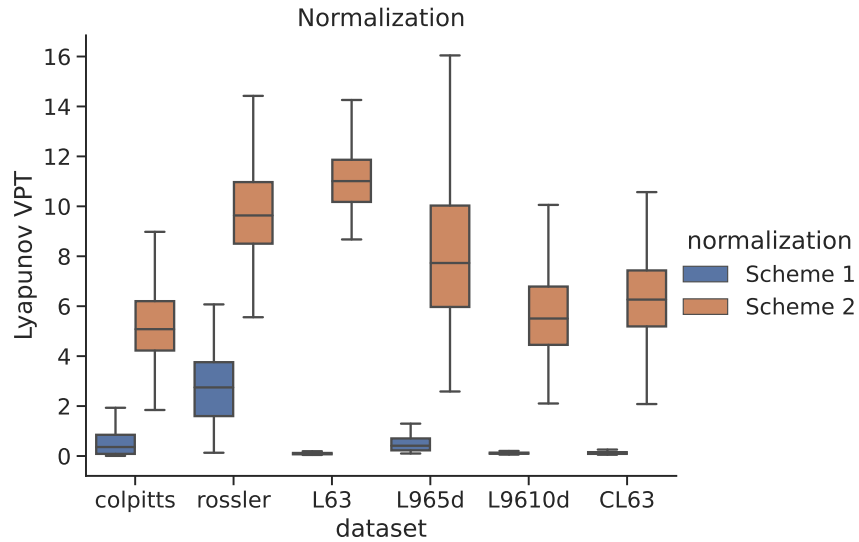


Figure 9: Results from normalizing by the two different schemes. The results from scheme 1 can actually be improved slightly by adding a small $\sim 1 - 2\%$ amount noise to the data—a phenomenon described as well in [29]—but the improvement is marginal. While we are not advocating for a particular normalization scheme, especially for already nondimensionalized data, it is important to keep in mind that there is essential information in the relationships between the time series that can be destroyed by introducing normalization for this RC architecture.

For RC it would be advisable to not normalize the variables when the data is already nondimensionalized, as is the case for the dynamical models examined here. We note that [31] used no normalization and still produced successful predictions with the RC models. In a realistic setting, however, when the data is collected from sensors and there are

many different unit scales, one has to do some kind of normalization in order for the methods to be numerically stable. We discuss this point further with a simple example in appendix B. Of course the normalization scheme appropriate for a particular machine learning task is dependent on the architecture of the network and the specific application so we do not claim any generalizability beyond the direct application of this RC method.

5.8 Effect of Noise

The RC is quite sensitive to additive noise Fig.(10). There is a sharp decrease in prediction quality with even a small amount of noise, meaning that any slight perturbation to the training of \mathbf{W}_{out} causes the errors to multiply rapidly. This is consistent with our observation that even casting double precision floating point numbers in \mathbf{W}_{out} to single precision can cause similar degradation. With a perfect numerical model one would expect a qualitatively similar noise response due to the exponential error growth quantified by the LEs.

We must address a phenomenon reported for RC by [29] as well as in ML models in general, that sometimes a small amount of noise can help predictions. This observation held true when we normalized the data using scheme 1 rather than scheme 2—see the previous section. We note, however, that even with the small observed increase in predictive skill, the VPT was still far below that reported with the scheme 2. While [29] speculated that the noise could regularize the data, we find this is not true in the model explored here.

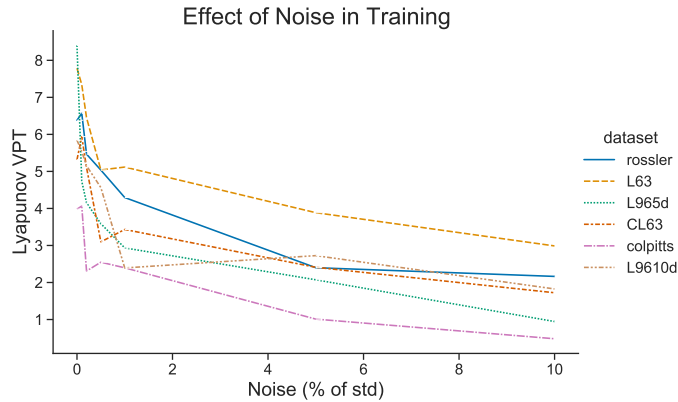


Figure 10: Mean VPT as a function of additive Gaussian noise expressed as a percentage of the long term standard deviation of each dynamical variable. The parameters were optimized for each noise level. We see a sharp decrease in prediction skill for small noise levels. Often in ML settings small amounts of noise can decrease overfitting [29] and improve prediction times. Our hypothesis in this case is that for perfect simulated data the RC is able to almost exactly reconstruct the vector field of the data f_u , with the finite forecast time related to how well the initial state is estimated from spinup. Noise, therefore, introduces errors into the estimation of f_u but also the estimation of the initial state, causing a fall in the (extremely high) prediction times of the zero noise case. We note that the prediction times with small noise are still greater than several Lyapunov timescales.

5.8.1 Sparse in Time

RC can be quite sensitive to the time step of the input data *i.e.*, how sparse the measurements are in time. In Fig.(11) we tested the RC for different time steps Δt , reoptimizing the RC each time. The total time of the training data $T = L\Delta t$ for L number of steps, did not change. We call attention to the result that finer training data does not always increase the prediction time of the RC and that certain models had an optimal time step. In practice one may want to interpolate the input data to try to match the optimal time step.

5.9 Sparsity of the Adjacency Matrix A

The sparsity of the adjacency matrix in general makes very little difference to the predictive capability of the RC—this is described by [40] and matches our experience. The benefit of using a very sparse matrix comes from taking advantage of sparse matrix representations in scientific computing software packages to vastly increase computational speed. We have found a set value of 98% or 99% sparsity, $\rho_A = 0.01$ to be sufficient for all applications, provided the resulting matrix is full rank.

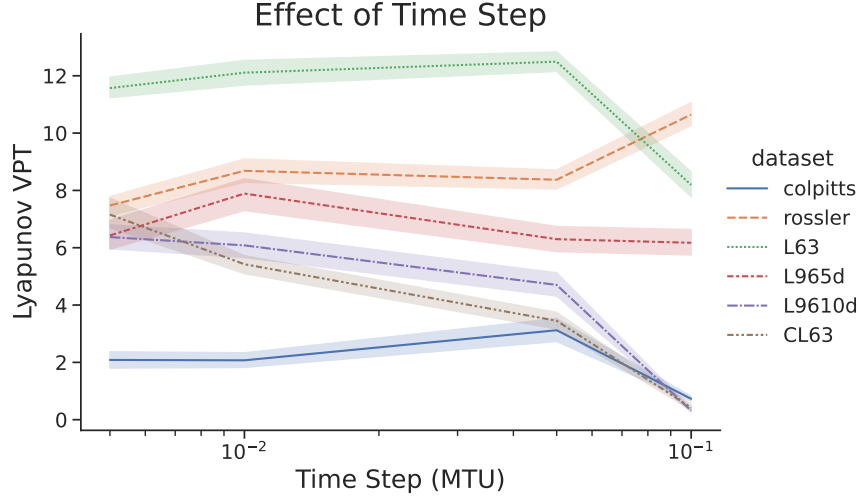


Figure 11: VPT as a function of the time step of the RC. The shading denotes the 95% confidence interval for the mean of the distribution. Note the decrease in VPT as the time step increases for the L9610d and CL63 systems and increase for the Rossler system.

Of course there is a limit to this guideline—if the matrix is too sparse (*i.e.*, has no nonzero values) then the RC will not work. In addition, when using small reservoirs of say less than $N=100$ then the randomness of the selection of \mathbf{A} can have a significant impact on the quality of predictions [17]. Slightly larger reservoirs seem to be more robust to random variations in connectivity.

6 Scaling to Higher Dimensional Systems

Up to this point we have considered systems with a state dimension $D \leq 10$, and a single RC model has been sufficient to predict their time evolution. However, systems with larger state spaces will require larger reservoirs to make adequate predictions. We highlight this situation in Figure 12, which shows the VPT of RC models with increasing reservoir dimension on the Lorenz96 system with 40 nodes. Note that we use a leading Lyapunov exponent of $\lambda_1 \simeq 1.68$ to represent the timescale for this system following [29]. There is no prediction skill until a certain minimum reservoir size is attained, somewhere between 4,800 and 6,000. For systems with even larger state spaces, the reservoir size will have to increase beyond the amount of Random Access Memory (RAM) available and parallelization schemes will have to be considered.

To address these issues, [26] introduced a parallelization strategy which has been used in numerous high dimensional applications, e.g. [29–31]. In this localization scheme, multiple reservoirs work semi-independently, each making predictions of a subset of the system state. More precisely, the elements of the system state $\mathbf{u} \in \mathbb{R}^D$ are split up into N_g groups, such that each group is assigned an individual RC model that predicts N_{output} of the state vector nodes, with $D = N_g N_{\text{output}}$. At each time step, each group receives N_{halo} of the neighboring state vector values, such that a system with a single spatial dimension has an input dimension for each RC model of $N_{\text{input}} = 2N_{\text{halo}} + N_{\text{output}}$. With this architecture one must decide how to choose N_{output} and N_{halo} . Our goal in this section is to explore these choices within the framework outlined in Sections 4 & 5.

6.1 Varying the halo size and output dimension

Figure 13(a) shows the performance of localized RC models with varying output size, N_{output} and halo size, N_{halo} . For output dimensions 2 and 4 (color), we see that the RC model shows its best performance when the halo size is 2. These RC models show no prediction skill with halos smaller than 2, and exhibit diminishing performance as the halo size increases beyond this point. These results suggest that for this particular system, the optimal configuration has a halo size of 2. For the Lorenz96 system, this is unsurprising since we know that the time evolution for each individual node requires information of neighboring states up to 2 nodes away, equation (D.0.4). For halo sizes smaller than 2, the known interactions between neighboring grid points are not represented. As the halo size increases, the RC model must effectively ‘learn’ the length scale across which the underlying dynamical interactions occur.

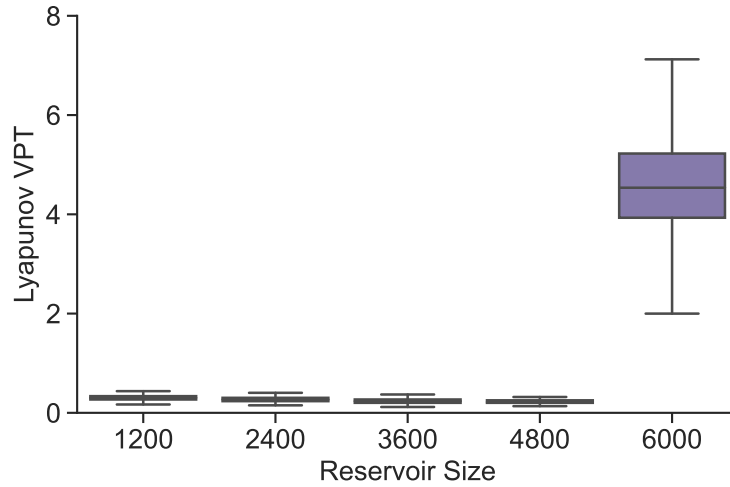


Figure 12: VPT from predictions of the 40D Lorenz96 system with a single RC and no localization. Parameters are optimized for each reservoir size. Parameter values are given in Table 10. The RC fails to predict at all below a certain cutoff size that increases concurrently with the size of the dynamical system.

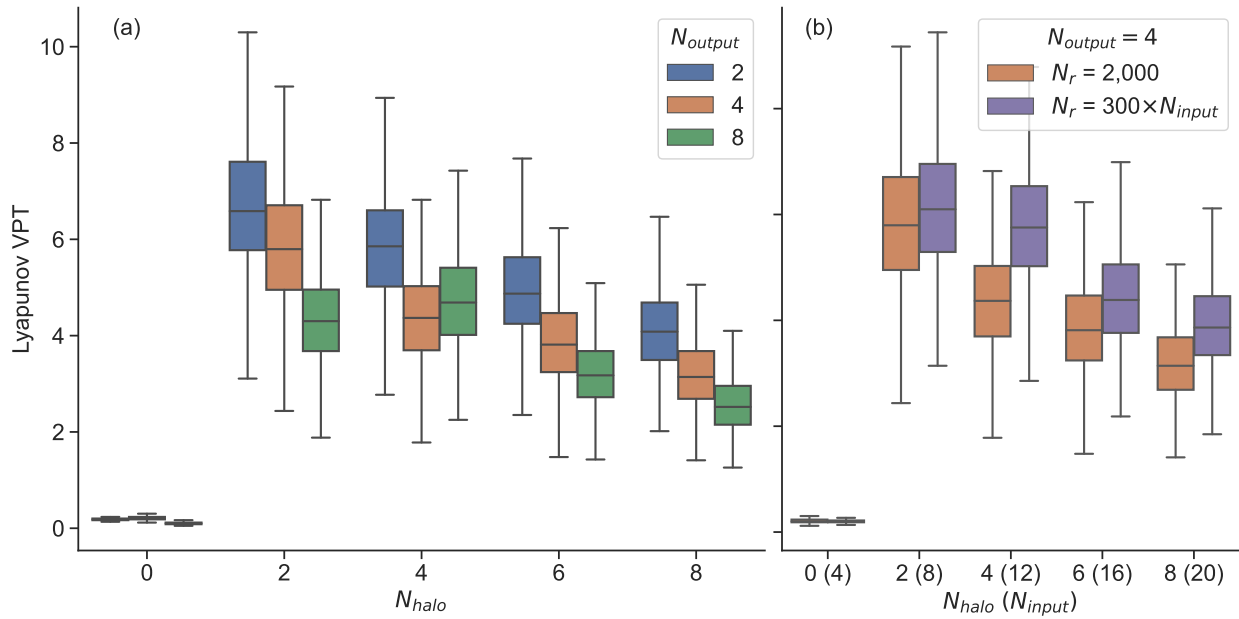


Figure 13: VPT of localized RC model predictions for the 40D Lorenz96 system. (a) Performance across various N_{output} (color) and N_{halo} values (x-axis). The reservoir size is fixed at $N_r = 2,000$. (b) Performance for $N_{output} = 4$, with the reservoir size fixed at 2,000 (orange) and reservoir size increasing with N_{input} (purple). Each histogram is generated from 1,000 samples with parameters optimized for the given reservoir size, output dimension, and halo size. Parameter values are given in Tables 11 & 12.

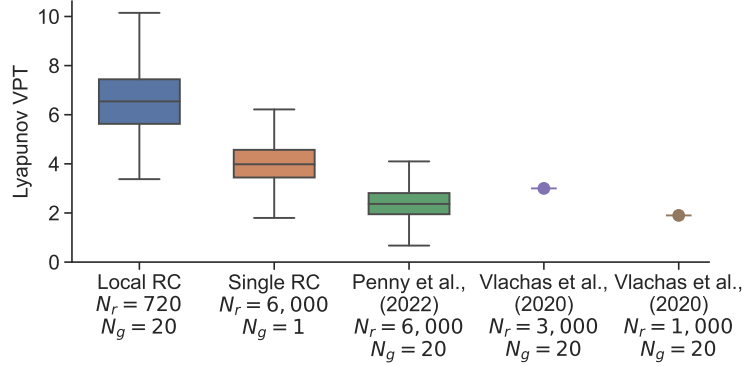


Figure 14: VPT of various RC based predictions of the 40D Lorenz96 system. The Local RC (blue) has the configuration $(N_{\text{output}}, N_{\text{halo}}, N_r) = (2, 2, 720)$. The Single RC (orange) uses $N_r = 6,000$ (Figure 12). The green box plot shows results from the localized RC Model 3 in [31], which uses $(N_{\text{output}}, N_{\text{halo}}, N_r) = (2, 4, 6000)$. The purple and tan dotted lines indicates the VPT from [29], which use $(N_{\text{output}}, N_{\text{halo}}, N_r) = (2, 4, 3000)$ and $(N_{\text{output}}, N_{\text{halo}}, N_r) = (2, 4, 1000)$, respectively. We note that the VPT reported here used a threshold $\epsilon = 0.5$ in order to match the results reported in [29]—the estimation of the results was accomplished by an examination of figure 9a where the VPT is directly reported. Additionally, the models in [29] for the parallel architecture were tuned by hand rather than using an automated parameter optimization procedure. Both the Local RC and Single RC models use an optimized nonzero input bias σ_b , while the other RC designs use $\sigma_b = 0$. Parameter values for the first three models are given in Table 13. The amount of training data for the $N_r = 720$ case is 40,000 with a time step of $dt = 0.01$ which compares to 100,000 data points in [29].

The results suggests that the optimal halo size should be set to a “minimum length scale” relevant to the time evolution of the dynamical system. This value is obviously system dependent, and will require knowledge of how the training data are acquired. If the data arise from a numerical model, the halo size could be related to the numerical stencil used in time integration (e.g. finite difference or finite element scheme). In the case of observed data, approximating the minimum length scale will require knowledge of the system dynamics.

As the output dimension for each local reservoir increases we see a similar effect as increasing the halo size: the prediction skill decreases. Taken together, prediction skill is optimal when the reservoir “stencil” is as small as possible: each local RC model is trained with exactly the amount of information it needs. This result is in some ways a restatement of [26] (their Figure 5b), who show that prediction performance improves as the number of local reservoirs increases with a fixed system dimension. Using more local reservoirs requires more computational resources, and the benefits must be weighed against the increase in computational cost.

6.2 Increasing the reservoir size with input dimension

The previous section showed that prediction skill decreases as the input dimension increases. Here, we test the impact of increasing the reservoir size in order compensate for this reduction in performance. Figure 13(b) shows the result of this test for $N_{\text{output}} = 4$, comparing performance with a fixed reservoir size $N_r = 2,000$ (orange) against models where $N_r = 300N_{\text{input}}$. For $N_{\text{halo}} = 4$, the median VPT increases by 1.4 (32%) as a result of increasing the reservoir size from 2,000 to 3,600 (80%). For larger values of N_{input} , the payoff from increasing the reservoir size is even smaller. For example, $N_{\text{halo}} = 6$, the median VPT increases by 0.6 (15%) but requires a reservoir size that is 4,800 (140% larger). The results highlight what is shown in Section 5.2 (Figure 4). After a certain level of performance is obtained for a given problem size, increasing the reservoir size provides diminishing returns for prediction skill.

6.3 Model Comparison

Finally, in order to emphasize that the optimization and exploration performed here leads to improved predictive ability, we compare our RC architecture with results from the literature Fig.(14). The first model we compare to is from Penny et al. [31], who use $N_g = 20$ local reservoirs, each with an output dimension of 2, halo size 4, and reservoir size of 6,000. We reimplement this model and note that the architecture/optimization framework is similar to that used here; the major difference is the added inclusion of an input bias. Penny et al. found it necessary to increase the local reservoir dimension to 6,000 in order to attain reasonable prediction skill—see Fig.(14). Our speculation

is that this was essentially a brute force solution to overcome an inactive bias term since that is the main difference between the models. This is an important takeaway for forecasting high dimensional systems, which may have such a large number of localized RC models that increasing the reservoir size to overcome this inactive bias term would become prohibitive.

The second model comparison is to Vlachas et al. [29] who compare and contrast a number of RC and RNN architectures and training methods using the Lorenz-96 40D system (as well as the Kuramoto-Sivashinsky equations). They benchmark results from an RC with hand tuned hyperparameters and compare the RC performance in terms of VPT, memory, and training time against LSTMs and GRUs. Their RC model was trained with 2.5 times more data at the same time step and they tune the noise level in their data to give maximal performance. We show in Fig.(14) that our optimized RC gives significantly better performance at a dramatically lower reservoir size. While the time taken to optimize the hyperparameters is significant at several hours wall time, offline training for a smaller and better performing RC is acceptable when training these kind of models for an online prediction task. Additionally the optimization procedure abrogates the need for hand tuning these parameters.

7 Conclusion and Discussion

Reservoir computing is a powerful machine learning method that can be used to successfully predict chaotic time series data. While not as flexible as other RNN methods, the RC has a number of properties that make it a good method of choice for these kinds of tasks. The ability to set the macro-scale properties of the network and the quick training of \mathbf{W}_{out} through linear regression couple together to allow the RC to be trained and deployed quickly and easily. Additionally, the RC has been shown not only to give good predictions but also to react like a physical model to perturbations in the system state (shown as the reproduction of the LEs) [27, 31, 51]. This property is crucial for applications in numerical weather prediction and other fields where it is important to produce both a forecast as well as an uncertainty estimate of that forecast. RCs can also be scaled up to large systems while being accelerated on dedicated hardware for vast speedups [32], thus making them an exciting candidate for the simulation of high dimensional systems.

We presented an exploration of the options needed for RCs to be trained successfully for chaotic time series forecasting. Almost all the results presented here involve the reoptimization of parameters through the outlined Bayesian optimization procedure for every experiment. The complex effects of the macro-scale parameters on the predictability of the RC and the correlations between these parameters preclude drawing many conclusions from simply varying each one of the parameters individually. When the data, RC training method, or one parameter is changed then a full re-optimization is necessary to compensate. This is one reason why the results presented here differ from those presented in previous literature.

The bias term in the RC equation is found to be critically important for all but the L63 system Fig.(6), and we emphasize that this is generally neglected in the current literature. Indeed predictions were practically impossible for many of the models without the addition of this term. We additionally show that the form of the readout has little impact on the skillful forecast time, at least for the given dynamical models. While this may be surprising given that the inclusion of an r^2 term in the readout has become a standard practice [26, 27, 29], we note that without the inclusion of bias, the nonlinear readout does have some positive impact (results not shown).

Our further exploration included showing the deleterious effects of a standard ML normalization procedure where each dimension of the input is recentered and rescaled separately [9]. This procedure is important in deep learning where the stability of the optimization procedure using backpropagation can be compromised. Centering all of the data around 0, however, destroys the important relationships between the variables, leading to a very poor predictive capacity for the RC. The RC is not trained through backpropagation, therefore it is advisable to avoid this kind of normalization when attempting to predict chaotic time series.

We also examine the effect of the reservoir dimension N , the amount of training data as well as the time step of the data on the performance of an RC. A general “law of diminishing returns” is exhibited in regards to RC size and the amount of data. This effect can be partially mitigated by re-optimizing the RC using a larger reservoir size instead of scaling up directly. We suggest that more effort should be dedicated to increasing the robustness of the RC to noise, since in real applications this will be a limitation on the usefulness of the method.

Finally, we show how to scale up the method to larger systems through localization, examining the effect of halo size and output dimension. We presented results on the 40 dimensional Lorenz system. Comparing to other results on this system for the RC [29, 31], the architecture and optimization procedure proposed here increased the VPT while decreasing the reservoir size, albeit at the expense of increased offline training time. Vlachas et al. (2020) [29] noted

“that RC...have slightly lower VPT than GRU and LSTM but require significantly lower training times.” Our results now give a clear advantage of RC over the LSTM and GRU in performance and online training time; additionally the RC has the advantage of reproducing the LEs of the input data, enabling the reproduction of the climatological attractor [27]. We hope that this work, by collecting a number of different characteristic models together, gives a clear set of standards to compare as a benchmark when developing new RC techniques.

8 Acknowledgements

J.A. Platt, S.G. Penny, and H.D.I. Abarbanel acknowledge support from the Office of Naval Research (ONR) grants N00014-19-1-2522 and N00014-20-1-2580. S.G. Penny and T.A. Smith acknowledge support from NOAA grant NA20OAR4600277. T.-C. Chen is supported by the NOAA Cooperative Agreement with CIRES, NA17OAR4320101. S.G. Penny and J.A. Platt jointly proposed and guided the research study. J.A. Platt conducted the majority of the investigation. T.A. Smith conducted the study of localization and scaling to larger systems. Coauthors Penny, Platt, Smith, and Chen all contributed to the development of the software used to conduct the experiments in this study. We thank Daniel Gauthier for his extensive comments.

9 Code Availability

Source code defining a version of the RC, optimization code and calculation of the LEs can be downloaded from github <https://github.com/japlatt/BasicReservoirComputing>. We hope to make the full parallel RC code available in the near future.

References

- [1] Edwin H. Colpitts. “Oscillation Generation”. US 1624537. Feb. 1918.
- [2] M.P. Kennedy. “Chaos in the Colpitts oscillator”. In: *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications* 41.11 (1994), pp. 771–774. DOI: [10.1109/81.331536](https://doi.org/10.1109/81.331536).
- [3] Edward N. Lorenz. “Deterministic Nonperiodic Flow”. In: *Journal of the Atmospheric Sciences* 20.2 (Mar. 1963), pp. 130–141. ISSN: 0022-4928. DOI: [10.1175/1520-0469\(1963\)020<0130:DNF>2.0.CO;2](https://doi.org/10.1175/1520-0469(1963)020<0130:DNF>2.0.CO;2).
- [4] Eugenia Kalnay. *Atmospheric Modeling, Data Assimilation and Predictability*. Cambridge University Press, 2002. DOI: [10.1017/CBO9780511802270](https://doi.org/10.1017/CBO9780511802270).
- [5] R. E. Kalman. “A New Approach to Linear Filtering and Prediction Problems”. In: *Journal of Basic Engineering* 82.1 (Mar. 1960), pp. 35–45. ISSN: 0021-9223. DOI: [10.1115/1.3662552](https://doi.org/10.1115/1.3662552).
- [6] Geir Evensen. *Data Assimilation: The Ensemble Kalman Filter*. Springer, 2009.
- [7] H. D. I. Abarbanel. *The Statistical Physics of Data Assimilation and Machine Learning*. Cambridge University Press, 2022, p. 284.
- [8] Anna Miller et al. “Statistical Data Assimilation: Formulation and Examples From Neurobiology”. In: *Frontiers in Applied Mathematics and Statistics* 4 (2018). ISSN: 2297-4687. DOI: [10.3389/fams.2018.00053](https://doi.org/10.3389/fams.2018.00053).
- [9] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. <http://www.deeplearningbook.org>. MIT Press, 2016.
- [10] Zachary Chase Lipton. “A Critical Review of Recurrent Neural Networks for Sequence Learning”. In: *CoRR* abs/1506.00019 (2015). arXiv: [1506.00019](https://arxiv.org/abs/1506.00019).
- [11] Sepp Hochreiter and Jürgen Schmidhuber. “Long Short-Term Memory”. In: *Neural Computation* 9.8 (1997), pp. 1735–1780.
- [12] F. A. Gers, J. Schmidhuber, and F. Cummins. “Learning to Forget: Continual Prediction with LSTM”. In: *Neural Computation* 12.10 (2000), pp. 2451–2471.
- [13] S. Hochreiter et al. “Gradient flow in recurrent nets: the difficulty of learning long-term dependencies”. In: *A Field Guide to Dynamical Recurrent Neural Networks*. Ed. by S. C. Kremer and J. F. Kolen. IEEE Press, 2001.
- [14] Y. Bengio, P. Frasconi, and P. Simard. “The problem of learning long-term dependencies in recurrent networks”. In: *IEEE International Conference on Neural Networks*. 1993, 1183–1188 vol.3.
- [15] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. “On the difficulty of training recurrent neural networks”. In: ed. by Sanjoy Dasgupta and David McAllester. Vol. 28. *Proceedings of Machine Learning Research* 3. Atlanta, Georgia, USA: PMLR, June 2013, pp. 1310–1318.

- [16] Kyunghyun Cho et al. “Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation”. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1724–1734. DOI: [10.3115/v1/D14-1179](https://doi.org/10.3115/v1/D14-1179).
- [17] M. Lukoševicius and H. Jaeger. “Reservoir computing approaches to recurrent neural network training”. In: *Comput. Sci. Rev.* 3 (2009), pp. 127–149.
- [18] Herbert Jaeger. *Long Short-Term Memory in Echo State Networks: Details of a Simulation Study*. 2012.
- [19] Herbert Jaeger. “The “echo state” approach to analysing and training recurrent neural networks—with an erratum note”. In: *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report 148* (2010), pp. 1–47.
- [20] Herbert Jaeger. “Short term memory in echo state networks. GMD-Report 152”. In: *GMD - German National Research Institute for Computer Science*. 2002.
- [21] Herbert Jaeger and Harald Haas. “Harnessing Nonlinearity: Predicting Chaotic Systems and Saving Energy in Wireless Communication”. In: *Science* 304 (2004), pp. 78–80. DOI: [10.1126/science.1091277](https://doi.org/10.1126/science.1091277).
- [22] Wolfgang Maass, Thomas Natschläger, and Henry Markram. “Real-Time Computing Without Stable States: A New Framework for Neural Computation Based on Perturbations”. In: *Neural Computation* 14 (2002), pp. 2531–2560.
- [23] Benjamin Schrauwen, David Verstraeten, and Jan Van Campenhout. “An overview of reservoir computing: Theory, applications and implementations”. In: *ESANN’2007 proceedings - European Symposium on Artificial Neural Networks Bruges (Belgium), 25-27 April 2007*. 2007, pp. 471–482.
- [24] J.J. Steil. “Backpropagation-decorrelation: online recurrent learning with O(N) complexity”. In: *2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No.04CH37541)*. Vol. 2. 2004, 843–848 vol.2. DOI: [10.1109/IJCNN.2004.1380039](https://doi.org/10.1109/IJCNN.2004.1380039).
- [25] A.F. Atiya and A.G. Parlos. “New results on recurrent network training: unifying the algorithms and accelerating convergence”. In: *IEEE Transactions on Neural Networks* 11.3 (2000), pp. 697–709. DOI: [10.1109/72.846741](https://doi.org/10.1109/72.846741).
- [26] Jaideep Pathak et al. “Model-Free Prediction of Large Spatiotemporally Chaotic Systems from Data: A Reservoir Computing Approach”. In: *Physical Review Letters* 120 (2018), p. 024102.
- [27] Zhixin Lu, Brian R. Hunt, and Edward Ott. “Attractor reconstruction by machine learning”. In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 28.6 (2018), p. 061104. DOI: [10.1063/1.5039508](https://doi.org/10.1063/1.5039508).
- [28] Aaron Griffith, Andrew Pomerance, and Daniel J. Gauthier. “Forecasting chaotic systems with very low connectivity reservoir computers”. In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 29.12 (2019), p. 123108. DOI: [10.1063/1.5120710](https://doi.org/10.1063/1.5120710).
- [29] P.R. Vlachas et al. “Backpropagation algorithms and Reservoir Computing in Recurrent Neural Networks for the forecasting of complex spatiotemporal dynamics”. In: *Neural Networks* 126 (2020), pp. 191–217. ISSN: 0893-6080. DOI: <https://doi.org/10.1016/j.neunet.2020.02.016>.
- [30] Troy Arcomano et al. “A Machine Learning-Based Global Atmospheric Forecast Model”. In: *Geophysical Research Letters* 47.9 (2020), e2020GL087776. DOI: <https://doi.org/10.1029/2020GL087776>.
- [31] S. G. Penny et al. “Integrating Recurrent Neural Networks With Data Assimilation for Scalable Data-Driven State Estimation”. In: *Journal of Advances in Modeling Earth Systems* 14.3 (2022), e2021MS002843. DOI: <https://doi.org/10.1029/2021MS002843>.
- [32] Daniel Canaday, Aaron Griffith, and Daniel J. Gauthier. “Rapid time series prediction with a hardware-based reservoir computer”. In: *Chaos* 28 (2018), p. 123119.
- [33] Marc Bocquet et al. “Bayesian inference of chaotic dynamics by merging data assimilation, machine learning and expectation-maximization”. In: *Foundations of Data Science* 2.1 (2020), pp. 55–80.
- [34] Marc Bocquet, Alban Farchi, and Quentin Malartic. “Online learning of both state and dynamics using ensemble Kalman filters”. In: *Foundations of Data Science* 3.3 (2021), pp. 305–330.
- [35] Casper Kaae Sønderby et al. *MetNet: A Neural Weather Model for Precipitation Forecasting*. 2020. arXiv: [2003.12140 \[cs.LG\]](https://arxiv.org/abs/2003.12140).
- [36] Erik Bollt. “On explaining the surprising success of reservoir computing forecaster of chaos? The universal machine learning dynamical system with contrast to VAR and DMD”. In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 31.1 (2021), p. 013108. DOI: [10.1063/5.0024890](https://doi.org/10.1063/5.0024890).
- [37] Daniel J. Gauthier et al. “Next generation reservoir computing”. en. In: *Nature Communications* 12.1 (Sept. 2021). DOI: [10.1038/s41467-021-25801-2](https://doi.org/10.1038/s41467-021-25801-2). (Visited on 11/22/2021).
- [38] Randall E Clark et al. “Reduced Dimension, Biophysical Neuron Models Constructed From Observed Data”. In: *bioRxiv* (2021). DOI: [10.1101/2021.12.03.471194](https://doi.org/10.1101/2021.12.03.471194).

- [39] H. D. I. Abarbanel, P. J. Rozdeba, and Sasha Shirman. “Machine Learning; Deepest Learning as Statistical Data Assimilation Problems”. In: *Neural Computation* (2018), pp. 2025–2055.
- [40] Mantas Lukoševičius. “A Practical Guide to Applying Echo State Networks”. In: (Jan. 2012), pp. 659–686. DOI: [10.1007/978-3-642-35289-8_36](https://doi.org/10.1007/978-3-642-35289-8_36).
- [41] Gouhei Tanaka et al. “Recent advances in physical reservoir computing: A review”. In: *Neural Networks* 115 (2019), pp. 100–123. ISSN: 0893-6080. DOI: <https://doi.org/10.1016/j.neunet.2019.03.005>.
- [42] Jeffrey L. Elman. “Finding structure in time”. In: *Cognitive Science* 14.2 (1990), pp. 179–211. ISSN: 0364-0213. DOI: [https://doi.org/10.1016/0364-0213\(90\)90002-E](https://doi.org/10.1016/0364-0213(90)90002-E).
- [43] F. Takens. “Detecting strange attractors in turbulence”. In: *Lecture Notes in Math.* 898 (1981), pp. 366–381.
- [44] H. D. I. Abarbanel. *The Analysis of Observed Chaotic Data*. Springer-Verlag, New York, 1996.
- [45] H. Kantz and T. Schreiber. *Nonlinear Time Series Analysis, 2nd ed.* Cambridge University Press, Cambridge, UK, 2004.
- [46] Allen Hart, James Hook, and Jonathan Dawes. “Embedding and approximation theorems for echo state networks”. In: *Neural Networks* 128 (2020), pp. 234–247. ISSN: 0893-6080. DOI: <https://doi.org/10.1016/j.neunet.2020.05.013>.
- [47] A. N. Tikhonov and V. Y. Arsenin. *Solutions of Ill-Posed Problems*. Wiley:New York, 1977.
- [48] Endong Wang et al. “Intel math kernel library”. In: *High-Performance Computing on the Intel® Xeon Phi™*. Springer, 2014, pp. 167–188.
- [49] A. M. LYAPUNOV. “The general problem of the stability of motion”. In: *International Journal of Control* 55.3 (1992), pp. 531–534. DOI: [10.1080/00207179208934253](https://doi.org/10.1080/00207179208934253).
- [50] J. -P. Eckmann and D. Ruelle. “Ergodic theory of chaos and strange attractors”. In: *Rev. Mod. Phys.* 57 (3 July 1985), pp. 617–656. DOI: [10.1103/RevModPhys.57.617](https://doi.org/10.1103/RevModPhys.57.617).
- [51] Jason A. Platt et al. “Robust forecasting using predictive generalized synchronization in reservoir computing”. In: *Chaos* 31 (2021), p. 123118. URL: <https://doi.org/10.1063/5.0066013>.
- [52] Karlheinz Geist, Ulrich Parlitz, and Werner Lauterborn. “Comparison of Different Methods for Computing Lyapunov Exponents”. In: *Progress of Theoretical Physics* 83.5 (May 1990), pp. 875–893. ISSN: 0033-068X. DOI: [10.1143/PTP.83.875](https://doi.org/10.1143/PTP.83.875).
- [53] M. M. Sushchik et al. “Generalized Synchronization of Chaos in Directionally Coupled Chaotic Systems”. In: *Physical Review E* 51 (1995), pp. 980–994.
- [54] S. Boyd and L.O. Chua. “Fading memory and the problem of approximating nonlinear operators with Volterra series”. In: *IEEE Trans. Circuits Syst., CAS-32* 11 (1985), pp. 1150–1161.
- [55] Brian R. Hunt, Edward Ott, and James A. Yorke. “Differentiable generalized synchronization of chaos”. In: *Phys. Rev. E* 55 (4 Apr. 1997), pp. 4029–4034. DOI: [10.1103/PhysRevE.55.4029](https://doi.org/10.1103/PhysRevE.55.4029). URL: <https://link.aps.org/doi/10.1103/PhysRevE.55.4029>.
- [56] Lyudmila Grigoryeva, Allen Hart, and Juan-Pablo Ortega. “Chaos on compact manifolds: Differentiable synchronizations beyond the Takens theorem”. In: *Phys. Rev. E* 103 (6 June 2021), p. 062204. DOI: [10.1103/PhysRevE.103.062204](https://doi.org/10.1103/PhysRevE.103.062204).
- [57] Lyudmila Grigoryeva, Allen Hart, and Juan-Pablo Ortega. *Learning strange attractors with reservoir systems*. 2021. arXiv: [2108.05024 \[math.DS\]](https://arxiv.org/abs/2108.05024).
- [58] Louis M. Pecora et al. “Fundamentals of synchronization in chaotic systems, concepts, and applications”. In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 7.4 (1997), pp. 520–543. DOI: [10.1063/1.166278](https://doi.org/10.1063/1.166278).
- [59] Louis M. Pecora and Thomas L. Carroll. “Synchronization in chaotic systems”. In: *Phys. Rev. Lett.* 64 (8 Feb. 1990), pp. 821–824. DOI: [10.1103/PhysRevLett.64.821](https://doi.org/10.1103/PhysRevLett.64.821).
- [60] Lou Pecora et al. “Synchronization Stability in Coupled Oscillator Arrays: Solution for Arbitrary Configurations”. In: *International Journal of Bifurcation and Chaos* 10.02 (2000), pp. 273–290. DOI: [10.1142/S0218127400000189](https://doi.org/10.1142/S0218127400000189).
- [61] Ljupco Kocarev, Ulrich Parlitz, and Reggie Brown. “Robust synchronization of chaotic systems”. In: *Phys. Rev. E* 61 (4 Apr. 2000), pp. 3716–3720. DOI: [10.1103/PhysRevE.61.3716](https://doi.org/10.1103/PhysRevE.61.3716). URL: <https://link.aps.org/doi/10.1103/PhysRevE.61.3716>.
- [62] H. D. I. Abarbanel, N. F. Rul’kov, and M. M. Sushchik. “The Auxiliary Systems Approach to Generalized Synchronization of Chaos”. In: *Physical Review E* 53 (1996), pp. 4528–4535.
- [63] David Verstraeten and Benjamin Schrauwen. “On the Quantification of Dynamics in Reservoir Computing”. In: *Artificial Neural Networks – ICANN 2009*. Ed. by Cesare Alippi et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 985–994. ISBN: 978-3-642-04274-4.

- [64] Joschka Boedecker et al. “Information processing in echo state networks at the edge of chaos”. In: *Theory in biosciences = Theorie in den Biowissenschaften* 131.3 (Sept. 2012), pp. 205–213. ISSN: 1431-7613. DOI: [10.1007/s12064-011-0146-8](https://doi.org/10.1007/s12064-011-0146-8).
- [65] T. L. Carroll. “Do reservoir computers work best at the edge of chaos?” In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 30.12 (2020), p. 121109. DOI: [10.1063/5.0038163](https://doi.org/10.1063/5.0038163).
- [66] H. Jaeger. “Foreword”. In: *Reservoir Computing: Theory, Physical Implementations, and Applications*. Ed. by K. Nakajima and I. Fischer. Natural Computing Series. Springer Nature Singapore, 2021, pp. v–x.
- [67] Li Deng. “The mnist database of handwritten digit images for machine learning research”. In: *IEEE Signal Processing Magazine* 29.6 (2012), pp. 141–142.
- [68] *List of datasets for machine-learning research*. 2021. URL: https://en.wikipedia.org/wiki/List_of_datasets_for_machine-learning_research.
- [69] H.D.I. Abarbanel, R. Brown, and M.B. Kennel. “Local Lyapunov exponents computed from observed data”. In: *Journal of Nonlinear Science* 2 (1992), pp. 343–365. DOI: <https://doi.org/10.1007/BF01208929>.
- [70] M. Peña and E. Kalnay. “Separating fast and slow modes in coupled chaotic systems”. In: *Nonlinear Processes in Geophysics* 11.3 (2004), pp. 319–327. DOI: [10.5194/npg-11-319-2004](https://doi.org/10.5194/npg-11-319-2004). URL: <https://npg.copernicus.org/articles/11/319/2004/>.
- [71] G. Manjunath and H. Jaeger. “Echo State Property Linked to an Input: Exploring a Fundamental Characteristic of Recurrent Neural Networks”. In: *Neural Computation* 25.3 (2013), pp. 671–696.
- [72] I. B. Yildiz, H. Jaeger, and S. J. Kiebel. “Re-Visiting the Echo State Property”. In: *Neural Networks* 35 (2012), pp. 1–20.
- [73] G. Manjunath. “Stability and memory-loss go hand-in-hand: three results in dynamics and computation”. In: *Proceedings of the Royal Society A* 476.2242 (2020), p. 20200563.
- [74] J. Močkus. “On bayesian methods for seeking the extremum”. In: *Optimization Techniques IFIP Technical Conference Novosibirsk, July 1–7, 1974*. Ed. by G. I. Marchuk. Berlin, Heidelberg: Springer Berlin Heidelberg, 1975, pp. 400–404. ISBN: 978-3-540-37497-8.
- [75] Donald Jones, Matthias Schonlau, and William Welch. “Efficient Global Optimization of Expensive Black-Box Functions”. In: *Journal of Global Optimization* 13 (Dec. 1998), pp. 455–492. DOI: [10.1023/A:1008306431147](https://doi.org/10.1023/A:1008306431147).
- [76] Mohamed Amine Bouhrel et al. “A Python surrogate modeling framework with derivatives”. In: *Advances in Engineering Software* 135 (2019), p. 102662. ISSN: 0965-9978. DOI: <https://doi.org/10.1016/j.advengsoft.2019.03.005>.
- [77] Joschka Herteux and Christoph R ath. “Breaking symmetries of the reservoir equations in echo state networks”. In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 30.12 (2020), p. 123142. DOI: [10.1063/5.0028993](https://doi.org/10.1063/5.0028993).
- [78] Wendson A. S. Barbosa et al. “Symmetry-aware reservoir computing”. In: *Phys. Rev. E* 104 (4 Oct. 2021), p. 045307. DOI: [10.1103/PhysRevE.104.045307](https://doi.org/10.1103/PhysRevE.104.045307).
- [79] James D. Bjorken and Sidney David Drell. *Relativistic Quantum Mechanics*. McGraw-Hill, 1964.
- [80] V. I. Oseledec. “A multiplicative ergodic theorem. Lyapunov characteristic numbers for dynamical systems”. In: *Trudy Mosk. Mat. Obsc.* 19 (1968), pp. 197–.
- [81] Steven H. Strogatz. *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry and Engineering*. Westview Press, 2000.
- [82] Leslie E. Matson. “The Malkus–Lorenz water wheel revisited”. In: *American Journal of Physics* 75.12 (2007), pp. 1114–1122. DOI: [10.1119/1.2785209](https://doi.org/10.1119/1.2785209). URL: <https://doi.org/10.1119/1.2785209>.
- [83] Lucas Illing et al. “Experiments with a Malkus–Lorenz water wheel: Chaos and Synchronization”. In: *American Journal of Physics* 80.3 (2012), pp. 192–202. DOI: [10.1119/1.3680533](https://doi.org/10.1119/1.3680533). URL: <https://doi.org/10.1119/1.3680533>.
- [84] O.E. R ossler. “An equation for continuous chaos”. In: *Physics Letters A* 57.5 (1976), pp. 397–398. ISSN: 0375-9601. DOI: [https://doi.org/10.1016/0375-9601\(76\)90101-8](https://doi.org/10.1016/0375-9601(76)90101-8).
- [85] Edward N Lorenz. “Predictability: A problem partly solved”. In: *Predictability of weather and climate*. Ed. by Tim Palmer and Renate Hagedorn. Cambridge, 2006.
- [86] Mark Kostuk. “Synchronization and Statistical Methods for the Data Assimilation of HVC Neuron Models”. PhD thesis. University of California San Diego, 2012. URL: <http://escholarship.org/uc/item/2fh4d086>.

A Lyapunov Exponents

Given an N -dimensional dynamical system ⁵

$$\dot{\mathbf{x}}(t) = f_x(\mathbf{x}(t), t) \quad (11)$$

with \mathbf{x} the state, f the dynamics and \cdot denoting $\frac{d}{dt}$, the spectrum of LEs $[\lambda_1, \lambda_2, \dots, \lambda_N]$ can be computed as follows. First we find the linearization of the system. We derive the variational equations, which describe the evolution of the tangent vectors, by taking $\frac{\partial}{\partial x_j(t')}$ on both sides of eq (11). Applying the chain rule we find

$$\frac{d}{dt} \frac{\partial x_i(t)}{\partial x_j(t')} = \frac{\partial f_i(x(t), t)}{\partial x_k(t)} \frac{\partial x_k(t)}{\partial x_j(t')}$$

which we rewrite as

$$\dot{\phi}(t', t) = \mathbf{J}(x(t)) \cdot \phi(t', t) \quad (12)$$

with $\phi(t', t) = \frac{\partial x_i(t)}{\partial x_j(t')}$ being the $N \times N$ variational matrix and $\mathbf{J}(x(t)) = \mathbf{J}(t) = \frac{\partial f_i(x(t), t)}{\partial x_k(t)}$ being the Jacobian. The general solution to Eq.(12) involves the matrix exponential

$$\phi(t', t) = \mathcal{T}_+ \exp \left\{ \int_t^{t'} \mathbf{J}(x(t)) dt \right\} \phi(t, t)$$

with $\phi(t, t) = \mathbb{I}$ and \mathcal{T}_+ being the *time ordering operator* [79] for two operators (here matrices) $\mathbf{A}(t)$ and $\mathbf{B}(t')$ that are not commutative,

$$\mathcal{T}_+ \mathbf{A}(t) \mathbf{B}(t') = \begin{cases} \mathbf{A}(t) \mathbf{B}(t') & \text{if } t > t' \\ \mathbf{B}(t') \mathbf{A}(t) & \text{if } t < t' \end{cases}.$$

\mathcal{T}_+ is important because the Jacobian matrices are not commutative *i.e.*, $[\mathbf{J}(t), \mathbf{J}(t')] \neq 0$.

The variational equation is much easier to work with in discrete time $t = t_0 + n\Delta t$; $t' = t_0 + n'\Delta t$ where it appears as

$$\phi(n', n) = \mathbf{DF}(n) \phi(n' - 1, n),$$

and $\phi(n, n) = \mathbb{I}$.

Then the solution for $\phi(n + L, n)$ is the product of Jacobians

$$\begin{aligned} \phi(n + L, n) &= \mathbf{DF}(n + L) \cdot \phi(n + L - 1, n) \\ &= \mathbf{DF}(n + L) \cdot \mathbf{DF}(n + L - 1) \cdot \phi(n + L - 2) \\ &= \mathbf{DF}(n + L) \cdot \mathbf{DF}(n + L - 1) \cdot \dots \cdot \mathbf{DF}(n). \end{aligned} \quad (13)$$

The matrix $\phi(t', t)$ describes how small perturbations to a state $x(t)$ propagate to the state at $x(t')$. Given equations (11) and (12) one can solve them concurrently to find Oseledec's matrix $\Phi = \phi(t', t) \phi(t', t)^T$.

The eigenvalues of the log of $\Phi(t', t)$ for large times

$$\lim_{t' \rightarrow \infty} \frac{1}{2t'} \log \phi(t', t) \phi(t', t)^T \quad (14)$$

are the **global** Lyapunov Exponents; the N eigenvalues are by definition real and the eigenvectors orthogonal since Φ is a symmetric $N \times N$ matrix. We order the LEs $\lambda_1 > \lambda_2 > \dots > \lambda_N$. The matrix Φ is ill-conditioned, so accurately evaluating all of its eigenvalues requires a stable algorithm, for example as provided by [44, 50].

Oseledec's multiplicative ergodic theorem [80] states that all N LEs of a dynamical system (a) exist, (b) are independent of the initial starting point $x(t)$, and (c) are invariant under smooth coordinate transformations ⁶. The finite time Lyapunov exponents (FTLE) [69], Eq.(14) evaluated for finite t' , are not independent of $x(t)$ nor are they invariant under a smooth coordinate transformation. Furthermore, for a continuous time dynamical system one of the LEs must

⁵ $f_x(\mathbf{x}(t), t)$ defines a dynamical system with external forcing. Formally, the LEs calculated for this system are called conditional Lyapunov exponents (CLE). If $f_x(\mathbf{x}(t))$ does not depend explicitly on time then it is an autonomous system and we are just calculating LEs.

⁶ The statement of invariance under coordinate transformation is important when attempting to recover the LEs from a time delay embedding.

be 0, and $\sum_{i=1}^N \lambda_i \leq 0$. In Hamiltonian systems the spectrum of LEs is symmetric around 0 and thus $\sum_{i=1}^N \lambda_i = 0$, a natural statement of phase space volume conservation guaranteed by Liouville’s theorem.

These definitions carry over to the forced dynamics with vector field $f_r(\mathbf{r}(t), \mathbf{u}(t))$, which we find in driven RNNs such as the driven RC Eq.(4) considered here. They are then called conditional Lyapunov exponents (CLEs) as they are conditional on the driving forces $\mathbf{u}(t)$ [59].

B Malkus Water Wheel

For physical applications, determining the correct normalization scheme can be more difficult when the state variables have different units *e.g.*, temperature, pressure, velocity, . . . All of the source models that we used in this study were nondimensionalized. In practice we may instead have data measured in quantities with different units. There are two solutions that we propose:

1. Nondimensionalize the state variables using physical parameters of the system. This does not require knowledge of the full equations of motion, only basic knowledge of the physics involved.
2. Allow the RC to learn the correct scaling laws by setting a separate σ and σ_b for each unit type. For example, one might have a $\sigma_{\text{Temperature}}$ and a σ_{Pressure} . Note that this replaces the scalar σ with a vector σ that has the structure $\sigma = [\sigma_1, \dots, \sigma_1, \dots, \sigma_p]$ for p different units for the state variables.

As an illustrative example, we consider the Malkus water wheel [81–83]. The chaotic water wheel is a physical model of the L63 equations and thus provides a mixed-units example with which to apply the proposed normalization schemes. Following the notation of [82], the three state variables are ω , y and z where ω is the angular velocity of the water wheel $\frac{d\theta}{dt}$ for the angle θ in the plane of the wheel, and y/z gives the position of the center of mass (COM) also in the plane of the wheel. The equations of motion are

$$\begin{aligned}\dot{\omega} &= ay - f\omega \\ \dot{y} &= \omega z - \lambda y \\ \dot{z} &= -\omega y + \lambda(R - z)\end{aligned}$$

where a is the angular acceleration due to gravity per horizontal displacement of the COM, f^{-1} is the time constant for the axle friction and input water drag, λ is the leakage rate of the water and R is the radius of the wheel. We see that λ^{-1} also gives the relaxation time constant for the COM. The maximal LE when $R = 1$ m, $a = 1$ (m s) $^{-1}$, $f = 0.4$, $\lambda = 0.1$ s $^{-1}$ is 0.053 giving a time constant of ~ 19 model time units.

To nondimensionalize the system we see that there are two different units for the state variables. ω has units 1/time $\equiv 1/T$ while y and z have units length $\equiv \ell$. Therefore, if we choose a time scale T and length scale ℓ , we can nondimensionalize by taking $\omega_* = \omega/T$, $y_* = \ell y$ and $z_* = \ell z$. Using knowledge of the physics but without knowing the equations of motion we can write a list of parameters on which the equations may depend—see table 3. Again, no a priori knowledge of the equations of motion is needed.

α	axle friction	$kg\ m/s^2$
I	moment of inertia	$kg\ m^2$
λ	leak rate	$1/s$
R	radius	m
$g \sin \phi$	gravity at angle ϕ	m/s^2
M	mass	kg

Table 3: Deduced parameters of the equations of motion for the Malkus water wheel.

We can construct T and ℓ using this set of candidate parameters. An obvious choice for ℓ is R , since both have the same units. For T we could use $T = 1/\lambda$ or $T = \sqrt{\frac{RMg \sin \phi}{I}}$. There is no wrong answer as long as all the parameters with the same units are scaled in the same way and one can measure the parameters needed.

As a (somewhat extreme) example, let us say the distance measurements from our sensor are in μm , and ω is measured in rad/s. In this case the RC is not able to predict the system at all Fig.(15). This is not really surprising considering the wildly different scales between ω and the x and y variables. We correct this by scaling ω by λ and x and y by R . This leads to reasonably accurate forecasts. Letting the optimization determine the correct scaling between the variables also leads to accurate forecasts.

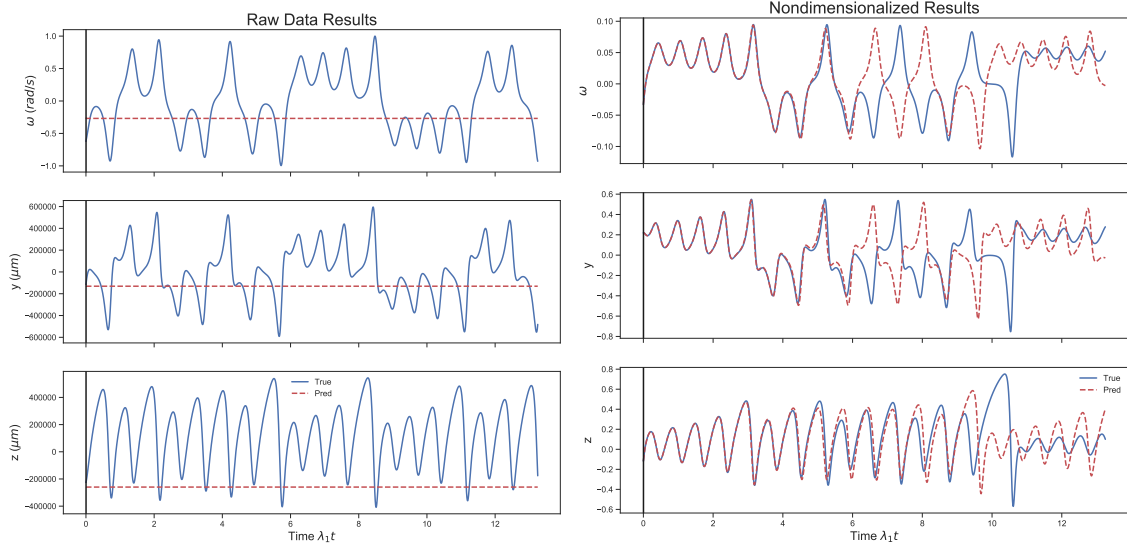


Figure 15: **(left)** Our hypothetical sensor measures the COM in μm , leading to an RC that utterly fails. **(right)** By normalizing the measurements correctly we find that the RC can now predict the Malkus water wheel with reasonable accuracy.

C Numerical Solvers

We evaluate different solvers for the linear regression problem for \mathbf{W}_{out} . The main takeaway is that "pinv" solvers have a mean/median VPT of $\sim 1 - 2$ less than the other solvers.

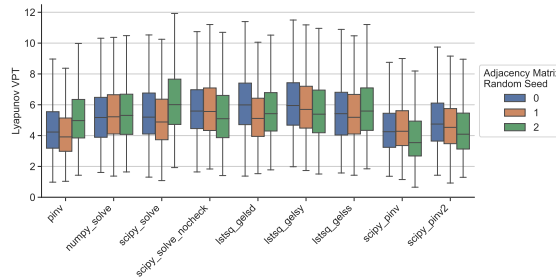


Figure 16: VPT from running the full optimization and VPT from 1000 sample points with Lorenz96-10D. Each histogram is created by fixing the reservoir size=2000, the readout method is linear, and the adjacency matrix random seed and training method as indicated. The definitions can be found in the numpy and scipy libraries.

D Details of Datasets

D.0.1 Rossler

The Rossler system [84] was proposed as a simplified form of the L63 system with a single second order nonlinearity and only one lobe of the strange attractor. The equations are

$$\begin{aligned} dx &= -(y + z) \\ dy &= x + 0.2y \\ dz &= 0.2 + z(x - 5.7). \end{aligned}$$

The LEs are $[0.06, 0, -4.9]$.

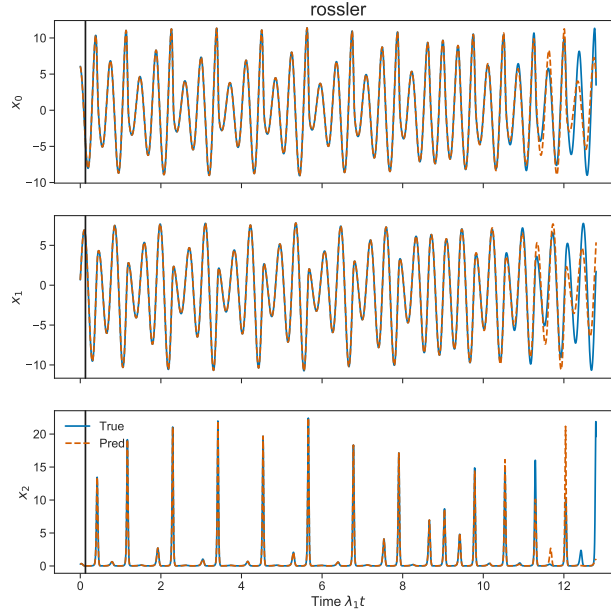


Figure 17: Rossler Attractor

D.0.2 Colpitts

The Colpitts Oscillator [1] is a three dimensional nonlinear dynamical system describing chaos in a nonlinear circuit. The equations of the system are given

$$\begin{aligned}\frac{dx(t)}{dt} &= \alpha y(t) \\ \frac{dy(t)}{dt} &= -\gamma(x(t) + z(t)) - qy(t) \\ \frac{dz(t)}{dt} &= \eta(y(t) + 1 - \exp(-x(t)))\end{aligned}$$

with $\alpha = 5$, $\gamma = 0.0797$, $q = 0.6898$ and $\eta = 6.2723$. For $\alpha < 5$ this circuit has limit cycle oscillations.

The Lyapunov exponents are $\{\lambda_1, \lambda_2, \lambda_3\} = [0.09, 0, -0.8]$ calculated via the QR decomposition algorithm given by Eckmann and Ruelle [50].

D.0.3 L63

The Lorenz-63 [3] equations form a deterministic nonlinear dynamical system that exhibits chaos for certain ranges of parameters. It was originally found as a three dimensional, reduced, approximation to the partial differential equations for the heating of the lower atmosphere of the Earth by solar radiation. The dynamical equations of motion are

$$\begin{aligned}\frac{dx(t)}{dt} &= \sigma[y(t) - x(t)] \\ \frac{dy(t)}{dt} &= x(t)[\rho - z(t)] - y(t) \\ \frac{dz(t)}{dt} &= x(t)y(t) - \beta z(t)\end{aligned}$$

with time independent parameters $\sigma = 10$, $\rho = 28$, $\beta = 8/3$.

The Lyapunov exponents are $\{\lambda_1, \lambda_2, \lambda_3\} = [0.9, 0, -14.7]$ calculated using the QR decomposition approach given by Eckmann and Ruelle [50].

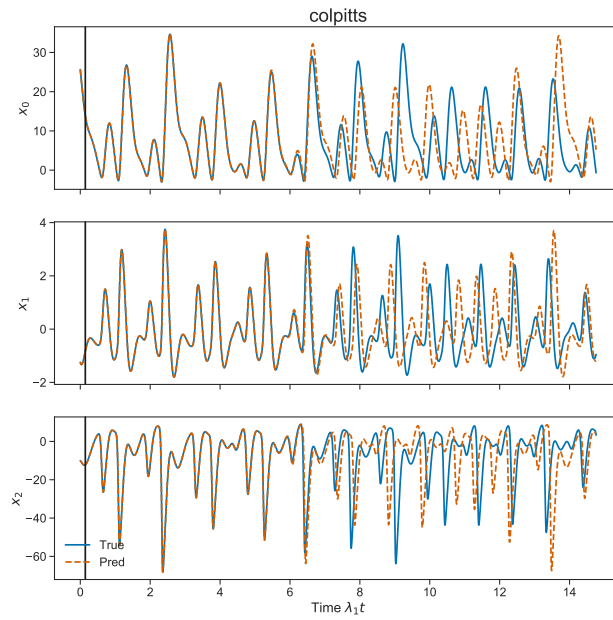


Figure 18: Colpitt's Oscillator

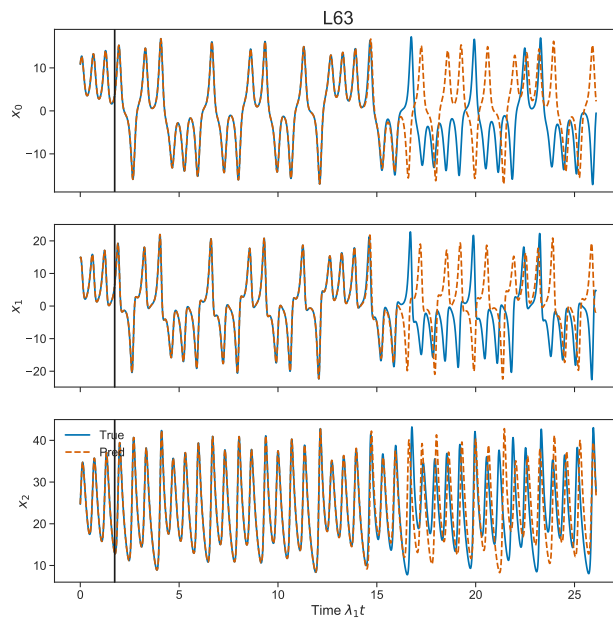


Figure 19: Lorenz 1963

D.0.4 L96

These dynamical equations were introduced by [85]:

$$\frac{dx_a(t)}{dt} = x_{a-1}(t)(x_{a+1}(t) - x_{a-2}(t)) - x_a(t) + f$$

and $a = 1, 2, \dots, D$; $x_{-1}(t) = x_{D-1}(t)$; $x_0(t) = x_D(t)$; $x_{D+1}(t) = x_1(t)$. f is a fixed parameter that we take to be 8.0 where the solutions to these dynamical equations are chaotic [86]. The equations for the states $x_a(t)$; $a = 1, 2, \dots, D$ are meant to describe points on a periodic spatial lattice. We use $D = 5$, $D = 10$ and $D = 40$.

The Lyapunov exponents are $\{\lambda_1, \dots, \lambda_5\} = [0.4, 0, -0.5, -1.3, -3.5]$ and

$$\{\lambda_1, \dots, \lambda_{10}\} = [1.1, 0.7, 0.1, 0, -0.4, -0.8, -1.3, -1.9, -2.7, -4.5]$$

calculated using the QR decomposition approach given by Eckmann and Ruelle [50].

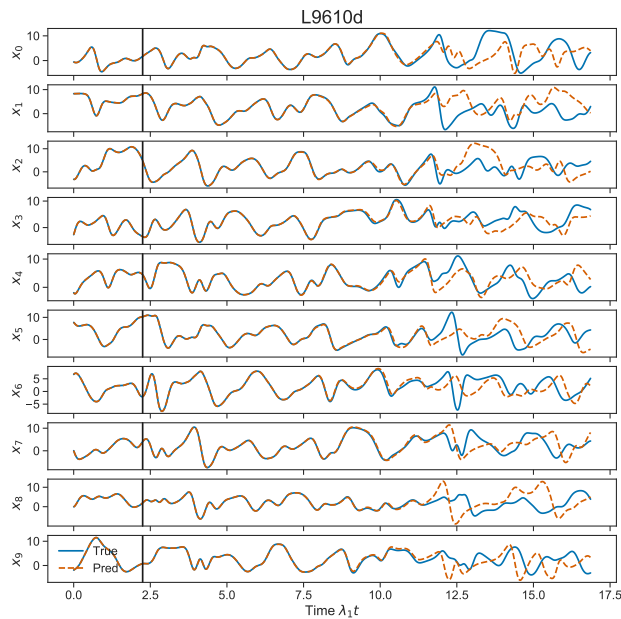


Figure 20: Lorenz 1996 10 dimensions

D.0.5 CL63

The climate Lorenz 63 system [70] consist of three L63 systems coupled together with multiple time scales. The three layers are an extratropical atmosphere, a tropical atmosphere, and a tropical ocean system. The presence of multiple

time scales often creates a challenge for RC models.

Extratropical atmosphere system

$$\frac{dx_e}{dt} = \sigma(y_e - x_e) - \kappa_e(Sx_t + k_1)$$

$$\frac{dy_e}{dt} = \rho x_e - y_e - x_e z_e + \kappa_e(Sy_t + k_1)$$

$$\frac{dz_e}{dt} = x_e y_e - \beta z_e$$

Tropical atmosphere system

$$\frac{dx_t}{dt} = \sigma(y_t - x_t) - \kappa(Sx_o + k_2) - \kappa_e(Sx_e + k_1)$$

$$\frac{dy_t}{dt} = \rho x_t - y_t - x_t z_t + \kappa(Sy_o + k_2) + \kappa_e(Sy_e + k_1)$$

$$\frac{dz_t}{dt} = x_t y_t - \beta z_t + \kappa_z z_o$$

Tropical ocean system

$$\frac{dx_o}{dt} = \tau \sigma(y_o - x_o) - \kappa(x_t + k_2)$$

$$\frac{dy_o}{dt} = \tau \rho x_o - \tau y_o - \tau S x_o z_o + \kappa(y_t + k_2)$$

$$\frac{dz_o}{dt} = \tau S x_o y_o - \tau \beta z_o - \kappa_z z_t$$

$\sigma = 10, \rho = 28, \beta = 8/3., S = 1, k_1 = 10, k_2 = -11, \tau = 0.1, \kappa = 1, \kappa_e = 0.08, \kappa_z = 1$ $LEs = [0.9, 0.4, 0, -0.1, -0.6, -0.8, -1.6, -11.7, -14]$

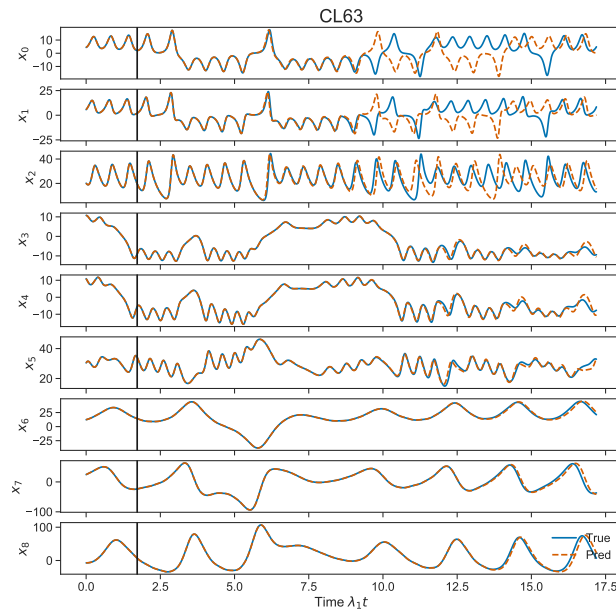


Figure 21: Climate Lorenz 63 model

E Parameters

dataset	σ	α	ρ_{SR}	β	N	σ_b	readout	ρ_A	dt	Train Samples
CL63	0.007	1.00	0.01	2.201375e-08	2000	0.66	linear	0.98	0.01	50,000
L63	0.084	0.60	0.80	8.493901e-08	2000	1.60	linear	0.98	0.01	50,000
L9610d	0.005	0.72	0.21	7.640822e-09	2000	1.47	linear	0.98	0.01	50,000
L96-5D	0.060	0.70	0.58	6.332524e-09	2000	1.59	linear	0.98	0.01	50,000
colpitts	0.100	1.00	1.20	1.000000e-08	2000	2.00	linear	0.98	0.01	50,000
rossler	0.066	0.47	0.50	2.101845e-09	2000	1.23	linear	0.98	0.01	50,000

Table 4: Fig 3

dataset	σ	α	ρ_{SR}	β	N	σ_b	readout	ρ_A	dt	Train Samples
CL63	0.002	0.84	0.53	1.033811e-07	250	1.31	linear	0.98	0.01	50,000
CL63	0.007	1.00	0.01	2.201375e-08	2000	0.66	biased	0.98	0.01	50,000
L63	0.024	0.96	0.64	1.339452e-08	250	1.63	biased	0.98	0.01	50,000
L63	0.084	0.60	0.80	8.493901e-08	2000	1.60	quadratic	0.98	0.01	50,000
L9610d	0.005	0.51	0.68	2.844557e-10	250	1.37	biased	0.98	0.01	50,000
L9610d	0.005	0.72	0.21	7.640822e-09	2000	1.47	biased	0.98	0.01	50,000
L96-5D	0.022	0.85	0.34	5.756201e-08	250	0.41	biased	0.98	0.01	50,000
L96-5D	0.060	0.70	0.58	6.332524e-09	2000	1.59	biased	0.98	0.01	50,000
colpitts	0.100	1.00	0.01	3.548470e-09	250	0.00	biased	0.98	0.01	50,000
colpitts	0.100	1.00	1.20	1.000000e-08	2000	2.00	linear	0.98	0.01	50,000
rossler	0.100	1.00	1.20	1.000000e-08	250	2.00	quadratic	0.98	0.01	50,000
rossler	0.066	0.47	0.50	2.101845e-09	2000	1.23	quadratic	0.98	0.01	50,000

Table 5: Fig 4

dataset	σ	α	ρ_{SR}	β	N	σ_b	readout	ρ_A	dt	Train Samples
CL63	0.004	0.98	0.66	5.216114e-10	1200	1.57	linear	0.98	0.01	50,000
CL63	0.014	0.88	0.01	1.572121e-05	1200	0.00	linear	0.98	0.01	50,000
L63	0.011	0.49	0.59	1.115693e-10	1200	0.28	linear	0.98	0.01	50,000
L63	0.014	0.89	1.02	2.390891e-08	1200	0.00	linear	0.98	0.01	50,000
L9610d	0.017	0.95	0.10	1.443850e-09	1200	1.59	linear	0.98	0.01	50,000
L9610d	0.001	0.41	0.92	4.553807e-03	1200	0.00	linear	0.98	0.01	50,000
L96-5D	0.100	1.00	0.05	1.012426e-10	1200	0.55	linear	0.98	0.01	50,000
L96-5D	0.096	0.80	0.40	4.087443e-08	1200	0.00	linear	0.98	0.01	50,000
colpitts	0.100	0.55	1.87	2.836222e-09	1200	2.85	linear	0.98	0.01	50,000
colpitts	0.022	0.91	1.11	3.177257e-07	1200	0.00	linear	0.98	0.01	50,000
rossler	0.042	0.89	0.79	1.006571e-08	1200	0.71	linear	0.98	0.01	50,000
rossler	0.028	0.48	0.81	1.000000e-08	1200	0.00	linear	0.98	0.01	50,000

Table 6: Fig 6

dataset	σ	α	ρ_{SR}	β	N	σ_b	readout	ρ_A	dt	Train Samples
L9610d	0.001	0.68	1.01	2.202428e-02	1000	1.40	linear	0.98	0.01	100
L9610d	0.003	0.85	0.43	1.043649e-09	1000	1.04	linear	0.98	0.01	1000
L9610d	0.006	1.00	0.01	1.769733e-10	1000	1.15	linear	0.98	0.01	10000
L9610d	0.008	0.22	0.02	9.382122e-08	1000	1.33	linear	0.98	0.01	100000
L96-5D	0.012	0.96	1.26	1.236301e-09	1000	3.00	linear	0.98	0.01	100
L96-5D	0.023	0.83	0.31	2.517159e-09	1000	1.23	linear	0.98	0.01	1000
L96-5D	0.039	0.69	0.85	1.349428e-08	1000	1.08	linear	0.98	0.01	10000
L96-5D	0.066	0.60	0.66	1.319216e-09	1000	1.04	linear	0.98	0.01	100000

Table 7: Fig. 8

dataset	σ	α	ρ_{SR}	β	N	σ_b	readout	ρ_A	normalization	dt	Train Samples
CL63	0.007	0.28	0.40	6.630623e-03	1512.0	1.44	quadratic	0.98	Scheme 1	0.01	50,000
CL63	1.566	1.00	0.01	2.201375e-08	2000.0	0.69	biased	0.98	Scheme 2	0.01	50,000
L63	0.002	0.17	1.16	3.530047e-03	1833.0	2.36	biased	0.98	Scheme 1	0.01	50,000
L63	5.985	0.60	0.80	8.493901e-08	2000.0	1.71	quadratic	0.98	Scheme 2	0.01	50,000
L9610d	0.005	0.40	0.75	2.292198e-01	1550.0	0.44	biased	0.98	Scheme 1	0.01	50,000
L9610d	0.116	0.72	0.21	7.640822e-09	2000.0	1.57	biased	0.98	Scheme 2	0.01	50,000
L96-5D	0.004	0.87	0.14	1.972360e-08	1739.0	3.12	biased	0.98	Scheme 1	0.01	50,000
L96-5D	1.017	0.70	0.58	6.332524e-09	2000.0	1.73	biased	0.98	Scheme 2	0.01	50,000
colpitts	0.527	0.19	0.44	1.780703e-03	1897.0	1.19	biased	0.98	Scheme 1	0.01	50,000
colpitts	10.184	1.00	1.20	1.000000e-08	2000.0	2.00	linear	0.98	Scheme 2	0.01	50,000
rossler	0.712	0.67	0.68	1.533086e-02	1595.0	4.30	quadratic	0.98	Scheme 1	0.01	50,000
rossler	2.204	0.47	0.50	2.101845e-09	2000.0	1.23	quadratic	0.98	Scheme 2	0.01	50,000

Table 8: Fig. 9

dataset	σ	α	ρ_{SR}	β	N	σ_b	readout	ρ_A	dt	Train Samples
CL63	0.002	0.64	0.43	3.963368e-09	1500	1.63	linear	0.98	0.005	100,000
CL63	0.004	0.96	0.28	8.664170e-07	1500	0.96	linear	0.98	0.010	50,000
CL63	0.008	0.94	0.03	7.266341e-08	1500	1.85	linear	0.98	0.050	10,000
CL63	0.004	0.61	1.55	1.322695e-02	1500	3.06	linear	0.98	0.100	5,000
L63	0.020	1.00	1.10	3.671288e-08	1500	1.34	linear	0.98	0.005	100,000
L63	0.034	0.92	0.69	1.294075e-08	1500	1.28	linear	0.98	0.010	50,000
L63	0.056	0.79	0.18	5.401025e-10	1500	1.02	linear	0.98	0.050	10,000
L63	0.040	0.79	0.05	1.436348e-07	1500	0.19	linear	0.98	0.100	5,000
L9610d	0.012	0.54	0.91	7.059347e-07	1500	1.82	linear	0.98	0.005	100,000
L9610d	0.020	0.51	0.87	4.387346e-08	1500	1.74	linear	0.98	0.010	50,000
L9610d	0.025	0.89	0.10	1.180616e-08	1500	2.37	linear	0.98	0.050	10,000
L9610d	0.003	0.61	0.75	1.237630e-02	1500	0.38	linear	0.98	0.100	5,000
L96-5D	0.014	0.95	1.64	1.604735e-07	1500	1.26	linear	0.98	0.005	100,000
L96-5D	0.066	0.90	1.31	1.672450e-09	1500	2.23	linear	0.98	0.010	50,000
L96-5D	0.025	0.67	0.07	1.654891e-10	1500	1.28	linear	0.98	0.050	10,000
L96-5D	0.100	0.69	0.03	1.903574e-08	1500	0.12	linear	0.98	0.100	5,000
colpitts	0.059	0.41	1.84	2.170770e-04	1500	1.69	linear	0.98	0.005	100,000
colpitts	0.100	0.83	2.00	5.228608e-05	1500	4.00	linear	0.98	0.010	50,000
colpitts	0.043	0.62	0.30	6.942212e-09	1500	0.15	linear	0.98	0.050	10,000
colpitts	0.100	0.98	1.95	1.000000e+00	1500	1.88	linear	0.98	0.100	5,000
rossler	0.050	0.66	0.23	8.252811e-09	1500	0.58	linear	0.98	0.005	100,000
rossler	0.030	1.00	1.52	2.060958e-08	1500	0.95	linear	0.98	0.010	50,000
rossler	0.035	0.43	2.00	2.999644e-10	1500	4.00	linear	0.98	0.050	10,000
rossler	0.081	0.60	1.48	1.025288e-10	1500	2.64	linear	0.98	0.100	5,000

Table 9: Fig. 11

σ	α	ρ_{SR}	β	N	σ_b	readout	ρ_A	dt	Train Samples
0.001	0.71	0.93	9.988696e-01	1200	0.01	linear	0.98	0.01	46167
0.002	0.32	0.78	5.411962e-02	2400	1.50	linear	0.98	0.01	25043
0.002	0.21	0.80	9.887132e-01	3600	1.29	linear	0.98	0.01	25043
0.001	0.90	1.50	1.000000e+00	4800	2.00	linear	0.98	0.01	31257
0.001	1.00	0.07	1.758120e-07	6000	0.63	linear	0.98	0.01	48647

N	Opt Walltime (min)
1200	86
2400	143
3600	211
4800	333
6000	396

Table 10: Fig. 12. Optimization walltime includes an initial sampling period and 35 optimization iterations. All simulations were run on the Google Cloud Platform using Cascade Lake processors with 60 virtual CPUs (i.e. c2-standard-60), where each experiment ran separately on a single compute node.

N_{output}	N_{halo}	σ	α	ρ_{SR}	β	N	σ_b	readout	ρ_A	dt	Train Samples	Opt time (min)
2	0	0.001	1.00	1.50	3.655278e-01	2000	1.11	linear	0.98	0.01	43995	798
2	2	0.027	0.67	1.18	1.221827e-08	2000	1.54	linear	0.98	0.01	38077	1747
2	4	0.017	1.00	0.16	3.432201e-07	2000	0.75	linear	0.98	0.01	49128	1748
2	6	0.005	0.94	0.46	6.443480e-07	2000	0.58	linear	0.98	0.01	49128	1678
2	8	0.006	0.74	0.71	1.801521e-07	2000	1.23	linear	0.98	0.01	49128	1733
4	0	0.004	0.45	0.39	2.475150e-01	2000	0.29	linear	0.98	0.01	36332	864
4	2	0.021	0.77	1.03	1.372422e-08	2000	1.27	linear	0.98	0.01	38077	882
4	4	0.003	0.99	0.64	4.054118e-07	2000	1.17	linear	0.98	0.01	32821	909
4	6	0.003	0.99	0.64	4.054118e-07	2000	1.17	linear	0.98	0.01	32821	833
4	8	0.003	0.99	0.64	4.054118e-07	2000	1.17	linear	0.98	0.01	32821	914
8	0	0.001	0.44	0.90	1.000000e+00	2000	1.74	linear	0.98	0.01	25286	462
8	2	0.003	0.99	0.64	4.054118e-07	2000	1.17	linear	0.98	0.01	32821	477
8	4	0.001	0.74	0.17	1.000000e-08	2000	0.72	linear	0.98	0.01	49128	456
8	6	0.003	0.99	0.64	4.054118e-07	2000	1.17	linear	0.98	0.01	32821	460
8	8	0.003	0.99	0.64	4.054118e-07	2000	1.17	linear	0.98	0.01	32821	485

N_{output}	N_{halo}	Opt Walltime (min)
2	0	798
2	2	1747
2	4	1748
2	6	1678
2	8	1733
4	0	864
4	2	882
4	4	909
4	6	833
4	8	914
8	0	462
8	2	477
8	4	456
8	6	460
8	8	485

Table 11: Fig. 13(a)

N_{output}	N_{halo}	σ	α	ρ_{SR}	β	N	σ_b	readout	ρ_A	dt	Train Samples
4	0	0.004	0.45	0.39	2.475150e-01	2000	0.29	linear	0.98	0.01	36332
4	2	0.021	0.77	1.03	1.372422e-08	2000	1.27	linear	0.98	0.01	38077
4	4	0.003	0.99	0.64	4.054118e-07	2000	1.17	linear	0.98	0.01	32821
4	6	0.003	0.99	0.64	4.054118e-07	2000	1.17	linear	0.98	0.01	32821
4	8	0.003	0.99	0.64	4.054118e-07	2000	1.17	linear	0.98	0.01	32821
4	0	0.004	0.45	0.39	2.475150e-01	1200	0.29	linear	0.98	0.01	36332
4	2	0.005	1.00	1.50	1.000000e-08	2400	2.00	linear	0.98	0.01	38077
4	4	0.005	0.36	0.48	1.000000e-08	3600	0.93	linear	0.98	0.01	31563
4	6	0.003	0.99	0.64	4.054118e-07	4800	1.17	linear	0.98	0.01	32821
4	8	0.003	0.99	0.64	4.054118e-07	6000	1.17	linear	0.98	0.01	32821

N_{output}	N_{halo}	N	Opt	Walltime (min)
4	0	2000	864	
4	2	2000	882	
4	4	2000	909	
4	6	2000	833	
4	8	2000	914	
4	0	1200	275	
4	2	2400	1016	
4	4	3600	1496	
4	6	4800	2544	
4	8	6000	3190	

Table 12: Fig. 13(b)

Name	N_{output}	N_{halo}	σ	α	ρ_{SR}	β	N	σ_b	readout	ρ_A	dt	Train Samples
Local RC	2	2	0.011	0.49	0.85	1.155972e-08	720	1.11	linear	0.98	0.01	38077
Single RC	2	2	0.001	1.00	0.07	1.758120e-07	6000	0.63	linear	0.98	0.01	48647
Model 3												
Penny et al, (2022)	2	4	0.052	0.41	0.34	3.611497e-06	6000	0.00	linear	0.99	0.01	200000

Name	Opt Walltime (min)
Local RC	521
Single RC	396
Model 3	
Penny et al, (2022)	2256

Table 13: Fig. 14