Recurrent neural networks for partially observed dynamical systems

Uttam Bhat[*]
Institute of Marine Sciences, University of California, Santa Cruz, California 95064, USA

Stephan B. Munch[†]
Applied Mathematics, University of California, Santa Cruz, California 95064, USA

(Received 27 September 2021; revised 25 February 2022; accepted 22 March 2022; published 13 April 2022)

Complex nonlinear dynamics are ubiquitous in many fields. Moreover, we rarely have access to all of the relevant state variables governing the dynamics. Delay embedding allows us, in principle, to account for unobserved state variables. Here we provide an algebraic approach to delay embedding that permits explicit approximation of error. We also provide the asymptotic dependence of the first-order approximation error on the system size. More importantly, this formulation of delay embedding can be directly implemented using a recurrent neural network (RNN). This observation expands the interpretability of both delay embedding and the RNN and facilitates principled incorporation of structure and other constraints into these approaches.

DOI: 10.1103/PhysRevE.105.044205

I. INTRODUCTION

Forecasting dynamical systems is important in many disciplines. Weather and climate [1], ecology [2,3], biology [4,5], fluid dynamics [6], etc., are generally modeled with nonlinear, discrete time equations or continuous time differential equations. In many cases, these nonlinear systems are chaotic and subject to stochastic drivers. However, the empirical data available are often incomplete; It is common to observe only a subset of the state variables or measure some coarse-grained statistic of the underlying state. In such a situation, all hope is not lost; Takens embedding theorem [7] shows that time-delayed versions of a single observable can be used in place of the unobserved dimensions to reconstruct the attractor manifold permitting accurate short- and midterm forecasts [8].

Takens theorem shows that any universal function approximator (given enough data) would be able to infer the function mapping the delay state vector to its future value. However, the proof of Takens' theorem is topological and nonconstructive. Therefore, one approach to reconstruct dynamics using partial state variables is accomplished by using off-the-shelf function approximation methods to infer the function mapping the delay vector to its future values from time-series data.

Due to the recent developments in machine learning, there are abundant choices for tools to perform nonlinear regression. The common candidates are local linear regressions, neural networks, and Gaussian processes [9–13]. Recurrent neural networks (RNN) and its variants are some of the most widely used tools for time-series prediction. Although the RNNs have been successfully applied to forecasting in a wide range of problems, literature on the mathematical reasons why they work so well is largely lacking [14,15]. Early justifications for using a recurrent architecture include: (1) being able to store temporal information [16], (2) neural networks with feedback capture time dependencies better, (3) are natural candidates for nonlinear autoregressive models [17], and (4) leads to reduced number of parameters due to weight sharing [18]. The RNNs were also used for time-series prediction due to their ability to be continually trained [19].

Neural network architectures for nonlinear dynamics are generally benchmarked on large data applications. Although there are asymptotic results proving the efficacy of some of these architectures, these results are not useful in many real world use cases where data can be limiting. Neural networks are also known to require a high degree of application-specific hyperparameter tuning [18]. This makes it hard to use neural networks where there is not sufficient data for a dedicated validation dataset. Progress can be made by assuming smoothness of the underlying functions to obtain less stringent requirements on the size of data needed to embed high-dimensional dynamics [20,21].

Here we present an approach to delay embedding through simple algebraic manipulation of the dynamical equations. We derive an approximation to the delay dynamics in terms of the original dynamics. We hypothesize that this approximation allows us to infer the delay map more efficiently with less data due to two reasons: (1) the original dynamics will be smoother than the delay function due to the distortions introduced by folding of the attractor manifold [22], and (2) the original dynamics are generally of smaller dimensionality than the delay function. We also discuss how our approximation is amenable to mechanistic interpretation unlike traditional delay embedding and nonlinear autoregressive models.

We use this approximation to encode a RNN to accomplish forecasting chaotic dynamics. Connections between the dynamical system and the RNN have been performed in the past [23–25]. However, these connections do not take advantage of the recurrent nature of partially observed dynamics.

[*]ubhat@ucsc.edu
[†]smunch@ucsc.edu

In general data-driven function approximation methods work well when the time-series data has a wide coverage across the domains of the functions. This is truly achieved when the dynamics are ergodic. In a practic setting, chaos or stochasticity too can be sufficient to achieve this.

In the next section, we calculate the first-order error due to partial observation of a system. We then develop a recursive approximation of the dynamics using only the observed states, and calculate the first-order expansion of the covariance of the recursion error. In Sec. III, we develop a recurrent neural network architecture that uses the recursive structure of the dynamics of the observed states. In Sec. IV, we illustrate the effects of partial observation on the delay dynamics using both an analytically solvable example and more complex dynamics commonly used in biophysical systems. We use short simulated time series as the effectiveness of the RNN structure over feedforward neural network (FNN) is most evident when the number of training points is less than 100. Finally, we discuss the potential for using other function approximators to take advantage of the general structure of dynamics to achieve more efficient representations of data.

II. RECURSIVE EXPANSION OF DYNAMICS

Assume the dynamics are completely represented with a system of M state variables, say $\mathbf{z_t} = \{z_{1,t}, z_{2,t}, \ldots, z_{M,t}\}^T$, and the dynamics are given by

$$\begin{aligned} \frac{dz_1}{dt} &= f_1(\mathbf{z}_t) \\ &\vdots \\ \frac{dz_M}{dt} &= f_M(\mathbf{z}_t). \end{aligned} \tag{1}$$

However, the subsequent arguments are more transparent in discrete time, so we work with the corresponding flow map integrated on a unit time step $z_{1,t} = F_1(\mathbf{z}_{t-1})$,...$z_{M,t} = F_M(\mathbf{z}_{t-1})$ which we write compactly as $\mathbf{z}_t = \mathbf{F}(\mathbf{z}_{t-1})$.

Since our focus is on partially observed systems, we split the state variables $\mathbf{z}_t$ into two subsets: $\mathbf{x}_t = \{z_{1,t}, z_{2,t}, \ldots, z_{n,t}\}^T$ representing the observed state variables and $\mathbf{y}_t = \{z_{n+1,t}, \ldots, z_{M,t}\}^T$ containing the remaining, unobserved state variables. We rewrite the dynamics as

$$\begin{aligned} \mathbf{x}_t &= \mathbf{F}(\mathbf{x}_{t-1}, \mathbf{y}_{t-1}) \\ \mathbf{y}_t &= \mathbf{G}(\mathbf{x}_{t-1}, \mathbf{y}_{t-1}), \end{aligned} \tag{2}$$

where $\mathbf{F}$ represents the maps for the n observed states and $\mathbf{G}$ represents the maps for the $M - n$ unobserved states.

There are several ways to proceed, including: (i) implicitly accounting for the unobserved states using time lags (Refs. [26,27]), or (ii) modeling the complete dynamics and imputing the unobserved states using a hidden Markov approach (e.g., Ref. [28]). However, (ii) requires that we have a reasonable model for the complete state dynamics and significant problems arise when the model is inaccurate. Since we assume the complete dynamics are unknown, we focus on (i).

As a first step to doing this, we shift the map for the unobserved states back by one time step and substitute this into the dynamics for the observed states,

$$\begin{aligned} \mathbf{x}_t &= \mathbf{F}(\mathbf{x}_{t-1}, \mathbf{y}_{t-1}) \\ &= \mathbf{F}(\mathbf{x}_{t-1}, \mathbf{G}[\mathbf{x}_{t-2}, \mathbf{y}_{t-2}]) \\ &= E_{\mathbf{y}_{t-2}}[\mathbf{F}(\mathbf{x}_{t-1}, \mathbf{G}[\mathbf{x}_{t-2}, \mathbf{y}_{t-2}])|\mathbf{x}_{t-1}, \mathbf{x}_{t-2}] + \varepsilon_t \\ &\approx \mathbf{F}(\mathbf{x}_{t-1}, \mathbf{G}[\mathbf{x}_{t-2}, \bar{\mathbf{y}}^{t-2}]) + \varepsilon_t, \end{aligned} \tag{3}$$

where $\bar{\mathbf{y}}^{t-2} = E[\mathbf{y}|\mathbf{x}_{t-1}, \mathbf{x}_{t-2}]$ is the conditional expectation for $\mathbf{y}$ given the current and previous observation for $\mathbf{x}$. The apparent process noise ε_t is given by $\varepsilon_t = \mathbf{F}[\mathbf{x}_{t-1}, \mathbf{G}(\mathbf{x}_{t-2}, \mathbf{y}_{t-2})] - \mathbf{F}[\mathbf{x}_{t-1}, \mathbf{G}(\mathbf{x}_{t-2}, \bar{\mathbf{y}}^{t-2})]$. The approximation in line 4 of Eq. (3) assumes $\mathbf{F}$ and $\mathbf{G}$ are almost linear for simplicity.

We can continue along this path an arbitrary number of times, each iteration adding another lag of $\mathbf{x}$ and pushing back the dependence on $\mathbf{y}$. Doing so d times we get

$$\mathbf{x}_t = \mathbf{F}(\mathbf{x}_{t-1}, \mathbf{G}[\mathbf{x}_{t-2}, \ldots \mathbf{G}\{\mathbf{x}_{t-d}, \bar{\mathbf{y}}^{t-d}\} \cdots]) + \varepsilon_t \tag{4}$$

$$= \tilde{\mathbf{F}}_d(\mathbf{x}_{t-1}, \ldots, \mathbf{x}_{t-d}) + \varepsilon_t, \tag{5}$$

where in keeping with the previous notation $\bar{\mathbf{y}}^{t-d} = E[\mathbf{y}_{t-d}|\mathbf{x}_{t-1}, \ldots, \mathbf{x}_{t-d}]$ and $\varepsilon_t = \mathbf{F}(\mathbf{x}_{t-1}, \mathbf{G}[\mathbf{x}_{t-2}, \ldots \mathbf{G}\{\mathbf{x}_{t-d}, \mathbf{y}_{t-d}\} \cdots]) - \mathbf{F}(\mathbf{x}_{t-1}, \mathbf{G}[\mathbf{x}_{t-2}, \ldots \mathbf{G}\{\mathbf{x}_{t-d}, \bar{\mathbf{y}}^{t-d}\} \cdots])$. As we show in the illustrative example in Sec. IV, we expect the dependence of function $\hat{\mathbf{F}}$ on $\mathbf{F}$ and $\mathbf{G}$ to be complicated. Therefore, it is hard to connect the function $\hat{\mathbf{F}}$ with the parameters of the generators of the dynamics, $\mathbf{F}$ and $\mathbf{G}$. With our recursive approximation (4), we can retain the identity of the ground-truth dynamics $\mathbf{F}$ and $\mathbf{G}$ using our approximation (4).

To provide a benchmark for our approximation, we estimate ε in the limit of large data. We simulate the exact dynamics for 30 000 time steps with sampling intervals matching the data generated in the next section. We discard the first 10 000 points to remove transients. We use the next 10 000 points to fit $\bar{\mathbf{y}}^{t-d} = E[\mathbf{y}_{t-n}|\mathbf{x}_{t-1}, \ldots, \mathbf{x}_{t-d}]$. We calculate the recursion error,

$$\varepsilon_t = \mathbf{x}_t - \mathbf{F}(\mathbf{x}_{t-1}, \mathbf{G}[\mathbf{x}_{t-2}, \ldots \mathbf{G}\{\mathbf{x}_{t-d}, \bar{\mathbf{y}}^{t-d}\} \cdots]). \tag{6}$$

We can also use a first-order approximation to estimate the covariance, $\mathbf{\Sigma}_t$ of the apparent process noise ε_t,

$$\mathbf{\Sigma}_t \approx \mathbf{P}_{t-1}\mathbf{Q}_{t-2} \cdots \mathbf{Q}_{t-d}\mathbf{C}_{t-d}\mathbf{Q}_{t-d}^T \cdots \mathbf{Q}_{t-2}^T\mathbf{P}_{t-1}^T, \tag{7}$$

where $\mathbf{P}_{t-1}$ is the matrix of partial derivatives of $\mathbf{F}$ with respect to $\mathbf{y}$ evaluated at $\mathbf{x}_{t-1}$ and $\bar{\mathbf{y}}^{t-1}$, $\mathbf{Q}_{t-n}$ is the matrix of partial derivatives of $\mathbf{G}$ with respect to $\mathbf{y}$ evaluated at $\mathbf{x}_{t-n}$ and $E[\mathbf{y}_{t-n}|\mathbf{x}_{t-1}, \ldots, \mathbf{x}_{t-d}]$, and $\mathbf{C}_{t-d}$ is the covariance matrix for the $\mathbf{y}_{t-d}$ conditional on $\mathbf{x}_{t-1}, \ldots, \mathbf{x}_{t-d}$, i.e., $\mathbf{C}_{t-d} = E[(\mathbf{y} - \bar{\mathbf{y}}^{t-d})(\mathbf{y} - \bar{\mathbf{y}}^{t-d})^T|\mathbf{x}_{t-1}, \ldots, \mathbf{x}_{t-d}]$.

Similar to the numerical estimation of the recursion error above, we can evaluate the first-order approximation numerically from time-series data by fitting $\mathbf{C}_{t-d} = E[(\mathbf{y} - \bar{\mathbf{y}}^{t-d})(\mathbf{y} - \bar{\mathbf{y}}^{t-d})^T|\mathbf{x}_{t-1}, \ldots, \mathbf{x}_{t-d}]$ from time-series data. The first-order approximation is accurate for maps that are almost linear. For continuous time nonlinear dynamics, this would correspond to short sampling intervals.

Note that neither of these should be treated as strict bounds on practical accuracy. However, these can provide a baseline

for the expected performance independent of the specifics of the forecast model.

III. RECURRENT NEURAL NETWORK

In a practical setting, the dynamics given by Eq. (2) can be learned from time-series data using delay vectors by fitting the function,

$$\mathbf{x}_t = \hat{\mathbf{F}}(\mathbf{x}_{t-1}, \ldots, \mathbf{x}_{t-d}). \tag{8}$$

This can be implemented directly using standard machine learning methods [16,21]. We implement a FNN to approximate $\hat{\mathbf{F}}$ as a benchmark. The recursive form of Eq. (4) suggests that the function approximator should be restricted among the space of functions that can be written as a recursive composition of lower-dimensional functions $\mathbf{F}$ and $\mathbf{G}$. This can be achieved by constructing a RNN that imitates the recursive form in Eq. (4),

$$\begin{aligned}
\hat{\mathbf{x}}_t &= \mathbf{W}_\mathbf{x}\mathbf{f}_t + \mathbf{b}_\mathbf{x},\\
\mathbf{f}_t &= \mathbf{a}_\mathbf{f}[\mathbf{W}_\mathbf{f}(\mathbf{x}_{t-1} \oplus \hat{\mathbf{y}}_{t-1}) + \mathbf{b}_\mathbf{f}],\\
\hat{\mathbf{y}}_{t-1} &= \mathbf{W}_\mathbf{y}\mathbf{g}_{t-1} + \mathbf{b}_\mathbf{y},\\
\mathbf{g}_{t-1} &= \mathbf{a}_\mathbf{g}[\mathbf{W}_\mathbf{g}(\mathbf{x}_{t-2} \oplus \hat{\mathbf{y}}_{t-2}) + \mathbf{b}_\mathbf{g}],\\
&\vdots\\
\hat{\mathbf{y}}_{t-d+1} &= \mathbf{W}_\mathbf{y}\mathbf{g}_{t-d+1} + \mathbf{b}_\mathbf{y},\\
\mathbf{g}_{t-d+1} &= \mathbf{a}_\mathbf{g}[\mathbf{W}_\mathbf{g}(\mathbf{x}_{t-d} \oplus \hat{\mathbf{y}}_{t-d}) + \mathbf{b}_\mathbf{g}],
\end{aligned} \tag{9}$$

where the functions $\mathbf{F}$ and $\mathbf{G}$ are approximated as single layer neural networks with hidden-layers $\mathbf{f}$ and $\mathbf{g}$. $\mathbf{a}_\mathbf{f}$ and $\mathbf{a}_\mathbf{g}$ are the nonlinear activation functions. In this paper, $\mathbf{a}_\mathbf{f} = \mathbf{a}_\mathbf{g} = \tanh$. As $\hat{\mathbf{y}}_t$ is just a linear function of $\mathbf{g}_t$, it can be absorbed into the parameters $\mathbf{W}_\mathbf{f}$, $\mathbf{b}_\mathbf{f}$, $\mathbf{W}_\mathbf{g}$, and $\mathbf{b}_\mathbf{g}$ to obtain a simpler neural network,

$$\begin{aligned}
\hat{\mathbf{x}}_t &= \mathbf{W}_\mathbf{x}\mathbf{f}_t + \mathbf{b}_\mathbf{x},\\
\mathbf{f}_t &= \mathbf{a}_\mathbf{f}[\mathbf{W}_\mathbf{f}(\mathbf{x}_{t-1} \oplus \mathbf{g}_{t-1}) + \mathbf{b}_\mathbf{f}],\\
\mathbf{g}_{t-1} &= \mathbf{a}_\mathbf{g}[\mathbf{W}_\mathbf{g}(\mathbf{x}_{t-2} \oplus \mathbf{g}_{t-2}) + \mathbf{b}_\mathbf{g}],\\
&\vdots\\
\mathbf{g}_{t-d+1} &= \mathbf{a}_\mathbf{g}[\mathbf{W}_\mathbf{g}(\mathbf{x}_{t-d} \oplus \mathbf{g}_{t-d}) + \mathbf{b}_\mathbf{g}].
\end{aligned} \tag{10}$$

The model parameters $\mathbf{W}_\alpha$ and $\mathbf{b}_\alpha$ are chosen to minimize the loss function,

$$L = \sum_t \|\hat{\mathbf{x}}_t - \mathbf{x}_\mathbf{t}^{(\text{data})}\|^2. \tag{11}$$

Note, since we do not observe $\mathbf{y}$, we cannot compute $\mathbf{g}_{t-d}$. In this paper, we choose $\mathbf{g}_{t-d}$ randomly for simplicity of setting up the backpropagation step. Alternatively, g_{t-d} can be included in the training parameters. We train the parameters using backpropagation with the RMSprop optimizer [29] and use early stopping [30] to avoid overfitting the training data. Note, since this is a proof-of-concept demonstration, we did not regularize using a penalty term in the loss function as this would make it difficult to explicitly compare the FNN and RNN in terms of the NN complexity. In the following sections, we use simulated time series from popular nonlinear dynamics models, namely, (A) discrete Lotka-Volterra model (two dimensional), (B) Lorenz 63 model [31] (three dimensional), (C) the Duffing oscillator [32] (four dimensional), and (D) the Lorenz 96 model [33] (5D). In each of these cases, we use just the first variable to train the RNN (i.e., we only observe one variable). We train the RNN using training time series of lengths 30, 50, and 100 data points. We specifically focus on small training datasets as the advantage of the RNN over a FNN is larger in the data-poor regime. We expect this to be the case as the data rich cases will be equivalently fit with any function approximator, and systematic differences in performance would be hard to detect due to stochastic differences in training performance. We divide the training data further into train and validation sets that contain 75% and 25% of the data, respectively, for the datasets of sizes 50 and 100. The early stopping parameter is chosen to minimize validation loss. The errors reported are measured on out-of-sample "test" data of the same size as the training datasets. The errors were averaged across 100 different realizations of the model in each case.

IV. NUMERICAL EXAMPLES

A. Discrete Lotka-Volterra model

To illustrate the effectiveness of the recursive approximation Eq. (4), we first examine a simple two-species system, where the state variables x_t (observed) and y_t (unobserved) are quadratic functions of x_{t-1} and y_{t-1}. This is also known as a discrete Lotka-Volterra model in ecology literature [34],

$$x_t = r_x x_{t-1}(1 - x_{t-1}) + A_{xy} x_{t-1} y_{t-1}, \tag{12a}$$

$$y_t = r_y y_{t-1}(1 - y_{t-1}) + A_{yx} x_{t-1} y_{t-1}. \tag{12b}$$

Solving for y_{t-1} using (12a) and substituting in (12b), we get the evolution of y_t as a function of x alone,

$$y_t = r_y \mathcal{Y}(x)[1 - \mathcal{Y}(x)] + A_{yx} x_{t-1} \mathcal{Y}(x), \tag{13}$$

where

$$\mathcal{Y}(x) = \left(\frac{x_t - r_x x_{t-1}(1 - x_{t-1})}{A_{xy} x_{t-1}}\right).$$

Substituting (13) back in (12a), we obtain the dynamical equation only in terms of x,

$$\begin{aligned}
x_t = r_x x_{t-1}(1 - x_{t-1}) + \Bigg[& r_y \left(\frac{x_{t-1}^2}{x_{t-2}} - r_x x_{t-1}(1 - x_{t-2})\right)\\
& \times \left(1 - \frac{x_{t-1} - r_x x_{t-2}(1 - x_{t-2})}{A_{xy} x_{t-2}}\right)\\
& + A_{yx} x_{t-1}[x_{t-1} - r_x x_{t-2}(1 - x_{t-2})]\Bigg],
\end{aligned} \tag{14}$$

which is highly nonlinear. Note the resulting single-variable dynamics is no longer quadratic and has arbitrarily higher-order nonzero derivatives. However, the recursive approximation (4) has a simpler functional form and ensures that higher derivatives are bounded. Specifically,

$$\begin{aligned}
x_t \approx\ & r_x x_{t-1}(1 - x_{t-1}) + A_{xy} x_{t-1}\\
& \times [r_y y_{t-2}(1 - y^*_{t-2}) + A_{yx} x_{t-2} y^*_{t-2}]
\end{aligned} \tag{15}$$

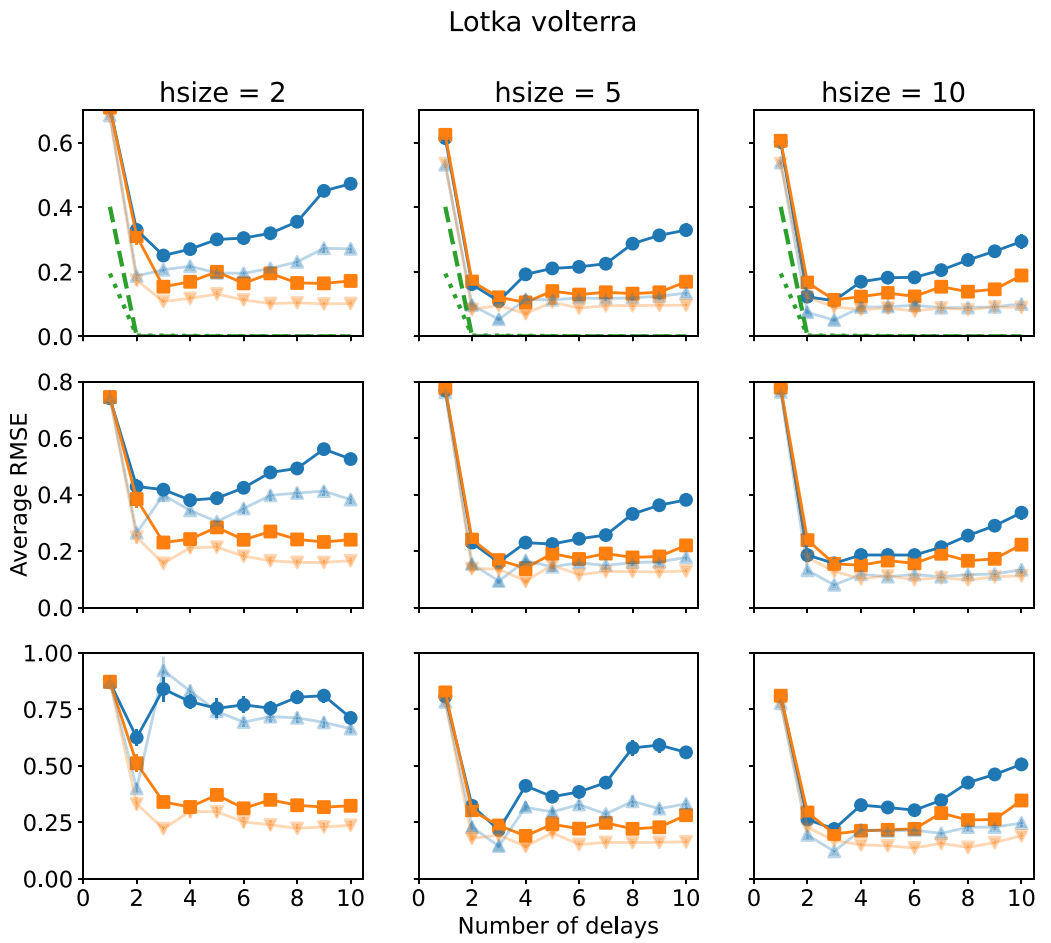

FIG. 1. Normalized RMSE as a function of number of delays from a FNN (blue) vs a RNN (orange) for the discrete Lotka-Volterra model (12). The top, middle, and bottom panels correspond to one-, two-, and three-steps ahead forecast error from a model trained on one-step ahead data. The left, middle, and right panels correspond to neural networks with hidden layers with two, five, and ten neurons each. The green dashed (dotted) lines in the top panel are numerical evaluations of the one-step ahead recursion error (4) [and its first-order approximation (7)]. Dark (light) colors represent results for training size of 50 (100) data points.

has nonzero derivatives only up to second order in x (and, in general, up to order d) considering y_ts are constants, thus, requiring less data to reconstruct the approximate dynamics. We calculate the theoretical recursion error ϵ given by Eq. (6) and its first-order approximation, Eq. (7) in the limit of large data. We see that the errors go to zero when the number of delays is two or greater consistent with Eq. (14). We generate the time series of length 50 and 100 data points and compare the performance of the RNN architecture (10) vs the FNN across different hidden-layer sizes. The hidden-layer size limits the expressivity of a neural network. For example, a neural network with one-dimensional input and two hidden neurons can only fit a function with a single peak. We see a significant improvement in the performance of the RNN over the FNN for small hidden-layer sizes as expected due to the simpler functions $\mathbf{F}$ and $\mathbf{G}$ required to be fit by the RNN (see Fig. 1) as against the more complex delay function $\tilde{\mathbf{F}}$. The small number of hidden neurons forces the neural networks to fit a function with less features, thereby making it difficult to fit the highly nonlinear function in (14). We also see that the difference in performance between the RNN and the FNN widens with increasing number of delays due to the increasing complexity $\tilde{\mathbf{F}}$. We also compare multistep ahead predictions using the single-step neural networks and iteratively applying the function on the delay vector to produce the next state. We hypothesize that the RNN functions $\mathbf{f}$ and $\mathbf{g}$ should be better in producing the multistep forecast. This is because the iteration of the more complex $\hat{\mathbf{F}}$ can lead to a larger variance at the locations in state space not seen by the one-step training data, compared to iterating the lower-dimensional function $\mathbf{f}$. We see that the RNN indeed performs better than the FNN in two- and three-step ahead prediction. The FNN errors increase significantly more than the RNN as we increase the number of steps hinting at the robustness of the recurrent structure of the RNN for the dynamical systems prediction.

We next look at some popular continuous chaotic dynamics in higher dimensions.

B. Lorenz 63 model

Lorenz 63 is one of the most popular chaotic models. It is a first-order differential equation modeling a simplified version

TABLE I. Descriptions of the datasets. (a) Autocorrelation at the time step used for predictions of the observed variable b. Normalized root mean square error (RMSE) values from predicting using the previous value. Normalized such that using mean of the time series leads to an RMSE = 1.

Model	Parameters	LE	Autocorrelation at dt[a]	RMSE "previous-value" Predictor[b]
Lotka-Volterra	$r = [0.933, 1.293], A = [0.758, 1.420]$	0.15	0.632	0.858
Lorenz63	$\rho = 28, \sigma = 10, \beta = 8/3$	0.91	0.869	0.512
Duffing oscillator	$[1.0, -1.0, 0.3, 0.5, 1.2]$	0.17	0.667	0.816
Lorenz 96	$N = 5, F = 8$	0.47	0.866	0.518

of atmospheric convection [31],

$$\begin{aligned} \dot{x} &= \sigma(y - x), \\ \dot{y} &= x(\rho - z) - y, \\ \dot{z} &= xy - \beta z. \end{aligned} \tag{16}$$

We chose a sampling rate of 10 Hz so that the prediction problem was sufficiently nontrivial but not impossible (see Table I for details). The observed variable is x. We compute training and validation loss for the neural networks with 2–20 hidden neurons (same number of hidden neurons for both **f** and **g** in the case of the RNN) and choose the one with the minimum validation loss. We also use the validation loss for early stopping the training. The recursion error (6) and its approximation (7) tend to zero with three or more delays.

The optimal number of delays for both the FNN and the RNN is three (see Fig. 2). The optimal RMSE is statistically indistinguishable between the FNN and the RNN. However, the RNN has a robust performance across all numbers of delays (especially for the smaller dataset), which may be desirable for automated applications.

We also plot the hidden-layer size-specific results (see Fig. 3). We see a similar trend as the discrete Lotka-Volterra model for the smallest hidden-layer size ($h = 2$), but there is no systematic advantage to the RNN with larger hidden layers.

C. Duffing oscillator

The Duffing oscillator is a second-order differential equation with periodic forcing,

$$\ddot{x} + \delta\dot{x} + \beta x + \alpha x^3 = \gamma \cos(\omega t). \tag{17}$$

This can be rewritten as a first-order autonomous system by introducing the variables $y = \dot{x}$, $v = \cos(\omega t)$, and $z = \sin(\omega t)$,

$$\begin{aligned} \dot{x} &= y, \\ \dot{y} &= \gamma v - \delta y - \beta x - \alpha x^3, \\ \dot{v} &= -\omega z, \\ \dot{z} &= \omega v. \end{aligned} \tag{18}$$

We chose a sampling rate of 1 Hz as this model has a Lyapunov horizon that is roughly an order of magnitude larger

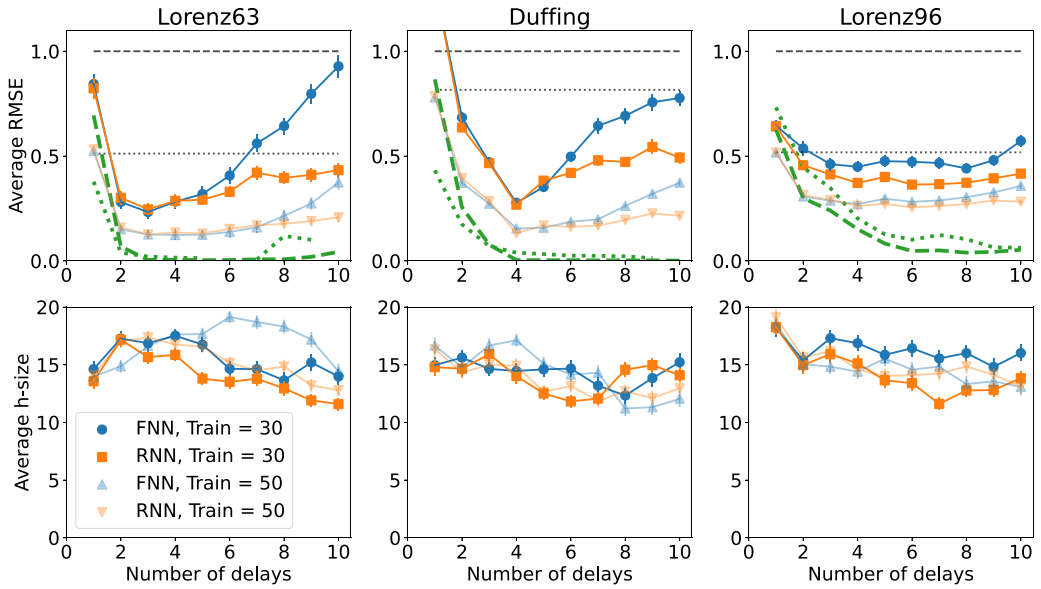

FIG. 2. (Top) The normalized RMSE as a function of number of delays from a FNN (blue) vs a RNN (orange) for the Lorenz 63 model (left), Duffing (middle), and the Lorenz 96 model in 5D (right) with parameters in Table I. The black dashed (dotted) lines are benchmark predictions using the time-series mean (previous value). The green dashed (dotted) lines are numerical evaluations of the recursion error (4) [and its first-order approximation (7)]. (Bottom) The average number of optimal hidden neurons. Dark (light) colors represent results for training size of 50 (100) data points.

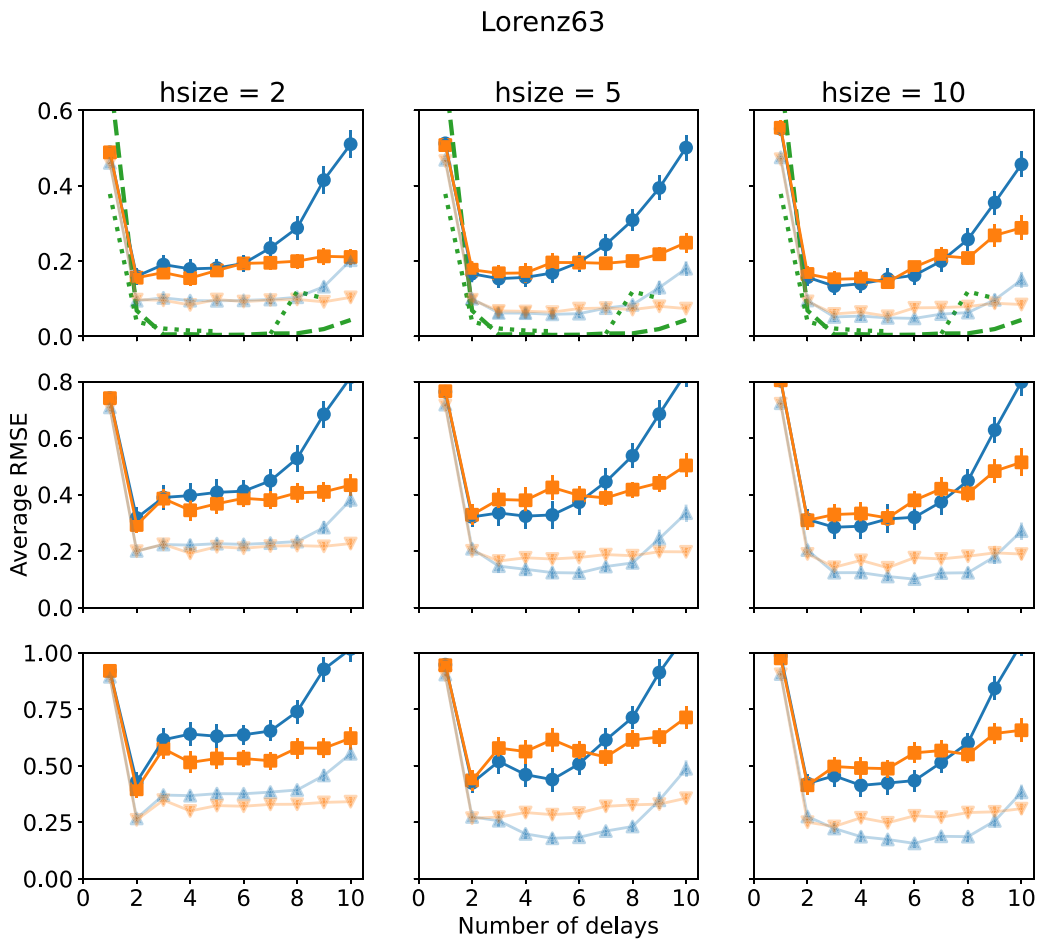

FIG. 3. Normalized RMSE as a function of number of delays from a FNN (blue) vs RNN (orange) for the Lorenz 63 model (16). The top, middle, and bottom panels correspond to one-, two-, and three-steps ahead forecast errors from a model trained on one-step ahead data. The left, middle, and right panels correspond to neural networks with hidden layers with two, five, and ten neurons each. The green dashed (dotted) lines in the top panel are numerical evaluations of the one-step ahead recursion error (4) [and its first-order approximation (7)]. Dark (light) colors represent results for training size of 50 (100) data points.

than the Lorenz 63 model (see Table I). The observed variable is x. The recursion error (6) and approximation (7) go to zero with four or more delays. The optimal number of delays in this case is four (see Fig. 2). The results for this model are qualitatively similar to the Lorenz 63 model, that is, at optimal number of delays, the performance is indistinguishable for the two neural networks, but the RNN is more robust across the number of delays. This trend is similar even when we restrict the neural networks to have small hidden layers (see Fig. 4).

D. Lorenz 96 model

Lorenz 96 is a popular model to test tools for chaotic time-series prediction in a high-dimensional setting [35,36]. We generated time-series data using the Lorenz 96 model [33],

$$\frac{dx_i}{dt} = (x_{i+1} - x_{i-2})x_{i-1} - x_i + F, \qquad 1 \leqslant i \leqslant N, \tag{19}$$

where it is assumed that $x_{-1} = x_{N-1}$, $x_0 = x_N$, and $x_{N+1} = x_1$. We use the parameters $N = 5$, $F = 8$. The dynamics are chaotic with Lyapunov exponent $= 0.472 \pm 0.002$. The sampling rate is 10 Hz. The observed variable is x_1. The recursion error (6) and approximation (7) tend to zero with roughly six or more delays. The optimal number of delays is six and eight for the RNN and the FNN, respectively (see Fig. 2).

The RNN shows significantly better performance as measured by the average RMSE in the case of the smaller dataset. There is also a significant difference between the optimal number of hidden neurons between the two NNs with the RNN opting for a lower number of hidden neurons indicating that the function to be fit is of a lower complexity. However, there is no significant difference in performance when the neural networks are restricted to small sizes (see Fig. 5).

V. DISCUSSION

In this paper, we showed that choosing the neural network architecture that is derived from the structure of generating dynamics can lead to more efficient recovery of the dynamics from data. This is demonstrated by a significantly better prediction performance by the recurrent neural network as compared to the feedforward neural network in the small-data regime. We also see that the RNN increasingly outperforms the FNN in multistep prediction tasks when the dynamics are trained on single-step data. The systematic advantage the RNN has over the FNN when trained with small hidden layers

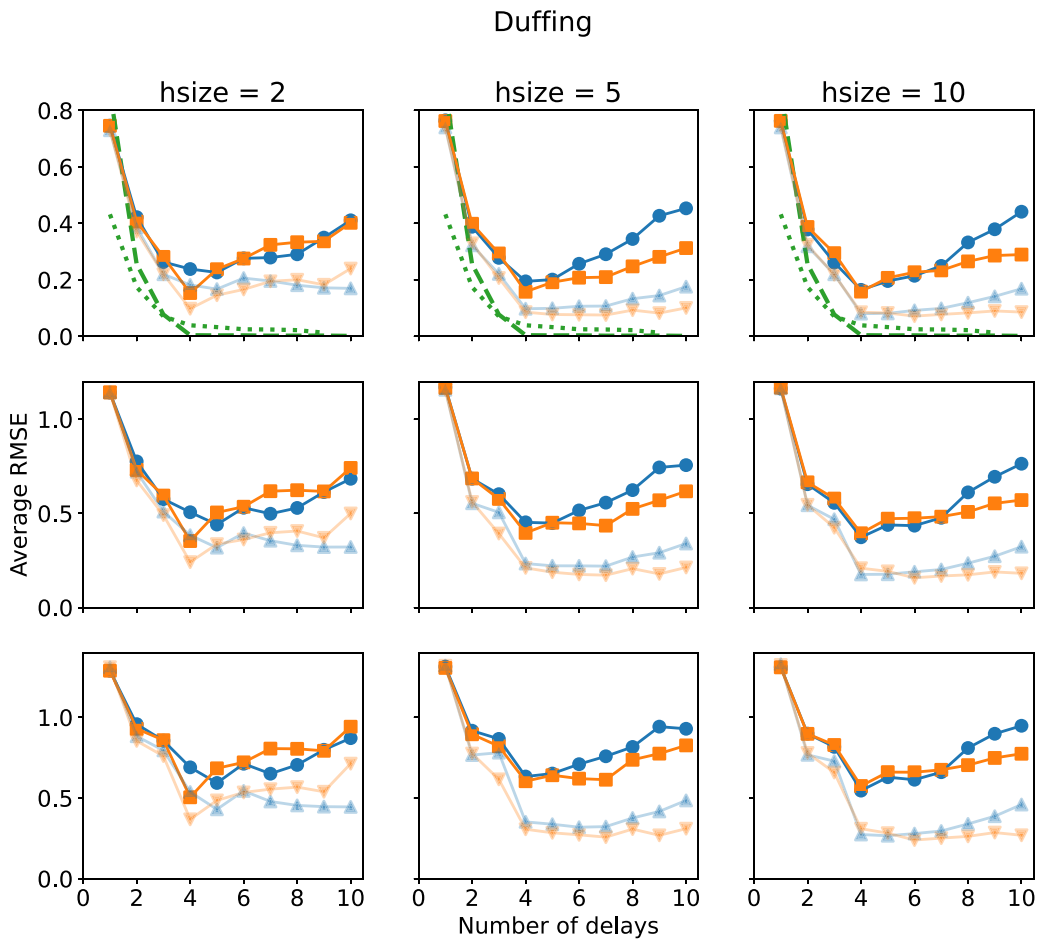

FIG. 4. Normalized RMSE as a function of number of delays from a FNN (blue) vs a RNN (orange) for the Lorenz 63 model (18). The top, middle, and bottom panels correspond to one-, two-, and three-steps ahead forecast errors from a model trained on one-step ahead data. The left, middle, and right panels correspond to neural networks with hidden layers with two, five, and ten neurons each. The green dashed (dotted) lines in the top panel are numerical evaluations of the one-step ahead recursion error (4) [and its first-order approximation (7)]. Dark (light) colors represent results for training size of 50 (100) data points.

suggest that the smoothness of **F** and **G** functions compared to the delay vector $\tilde{\mathbf{F}}$ can be leveraged by manipulating the structure of neural networks.

In this paper, we see that the structural advantage of the RNN comes into play when the attractor dimension is small, and the manifold is smooth. The advantage of the RNN seems to systematically diminish with increasing dimensionality of dynamics. This is consistent with the interpretation that the RNN is exposed to smoother functional forms than the FNN which is directly fit to the more nonlinear delay map. More work needs to be performed to fully characterize the regime where incorporating the dynamical structure in the neural network will yield better predictions.

Having dynamically meaningful units within the neural networks is useful in applications where it is important to learn the mechanistic relationships between variables. It also makes incorporating auxiliary information straightforward. For example, information about interactions between states can be implemented by conditioning the model to constrain the partial derivatives $\partial H_i/\partial z_j = 0$ (where $H \in \{F, G\}$ and $z \in \{x, y\}$) that correspond to noninteracting states. This can be achieved through regularization or constraint optimization of neural networks. Our methods can be extended to take advantage of the structure of spatial dynamics as well. Since we expect the spatial interaction structure to be sparse, we expect **F** and **G** to have a much lower dimensionality compared to fitting the full delay-embedding function.

In the field of statistical mechanics, the problem of unobserved states has been addressed by the Mori-Zwanzig formulation where the Zwanzig operator is used to project the dynamics onto the linear subspace of the observed dynamics where the ignored degrees of freedom appear as a memory term and a noise term. Calculating the memory term for nonlinear dynamics is nontrivial and requires the expansion of the basis to lift the dynamics to a linear space. This can lead to an unbounded expansion of the state space in chaotic systems. In contrast, our approximation provides a straightforward way to incorporate the induced memory from partial observations.

To summarize, we address the gap in the theoretical literature on the efficacy of recurrent neural networks. We show how partially observed dynamics can be restructured to reveal a recurrent structure, which can be learned by fitting recurrent neural networks on time-series data. We also provide a

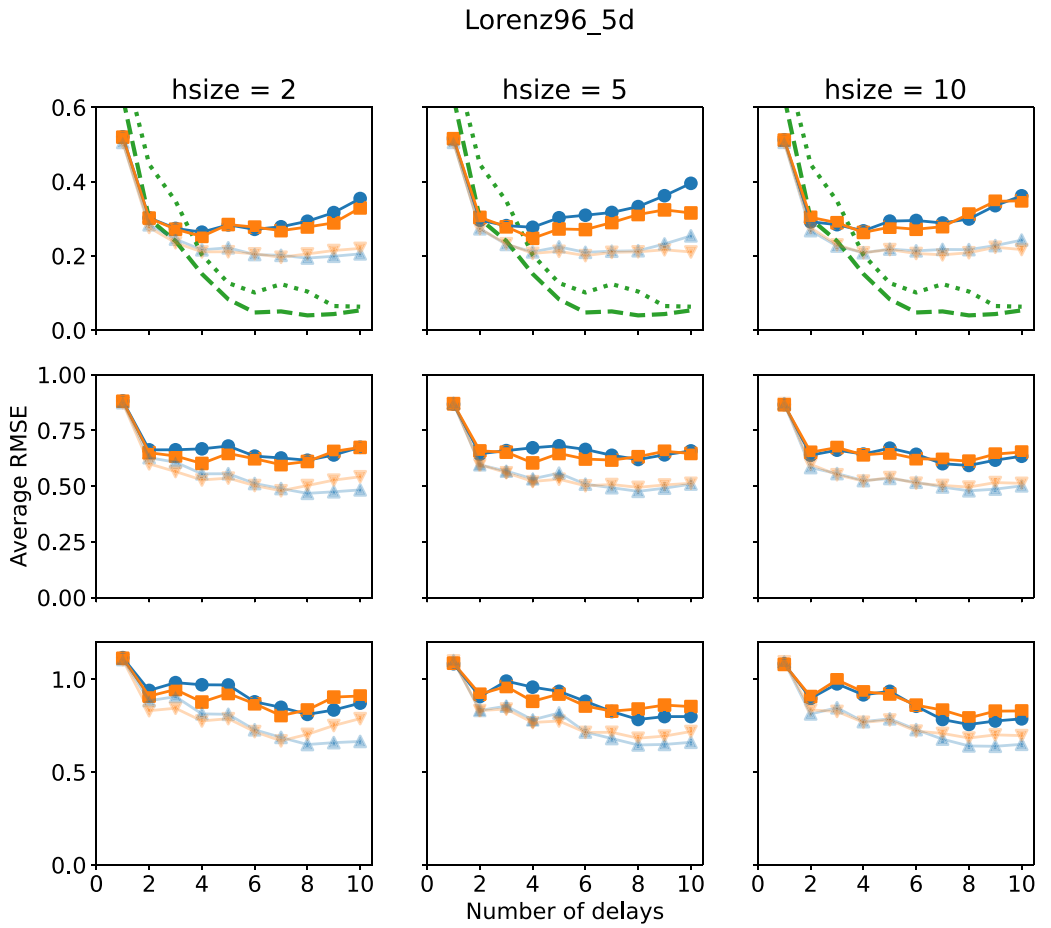

FIG. 5. Normalized RMSE as a function of number of delays from a FNN (blue) vs a RNN (orange) for the Lorenz 63 model (19). The top, middle, and bottom panels correspond to one-, two-, and three-steps ahead forecast errors from a model trained on one-step ahead data. The left, middle, and right panels correspond to neural networks with hidden layers with two, five, and ten neurons each. The green dashed (dotted) lines in the top panel are numerical evaluations of the one-step ahead recursion error (4) [and its first-order approximation (7)]. Dark (light) colors represent results for training size of 50 (100) data points.

connection to time-delay embedding and discuss the potential applications of this methodology.

ACKNOWLEDGMENTS

This work was supported by NOAA's HPCC incubator.

[1] T. N. Palmer, A nonlinear dynamical perspective on model error: A proposal for non-local stochastic-dynamic parametrization in weather and climate prediction models, Quarterly J. R. Meteorological Soc. **127**, 279 (2001).

[2] S. Lek, M. Delacoste, P. Baran, I. Dimopoulos, J. Lauga, and S. Aulagnier, Application of neural networks to modelling nonlinear relationships in ecology, Ecol. Modell. **90**, 39 (1996).

[3] Y. Luo, K. Ogle, C. Tucker, S. Fei, C. Gao, S. LaDeau, J. S. Clark, and D. S. Schimel, Ecological forecasting and data assimilation in a data-rich era, Ecological Appl. **21**, 1429 (2011).

[4] M. Lachowicz, Individually-based markov processes modeling nonlinear systems in mathematical biology, Nonlinear Anal.: Real World Appl. **12**, 2396 (2011).

[5] S. Imoto, S. Kim, T. Goto, S. Aburatani, K. Tashiro, S. Kuhara, and S. Miyano, Bayesian network and nonparametric heteroscedastic regression for nonlinear modeling of genetic network, J. Bioinf. Comput. Biol.y **1**, 231 (2003).

[6] J. D. Farmer and J. J. Sidorowich, Predicting Chaotic Time Series, Phys. Rev. Lett. **59**, 845 (1987).

[7] F. Takens, Detecting strange attractors in turbulence, in *Dynamical Systems and Turbulence, Warwick 1980* (Springer-Verlag, Berlin, Heidelberg, 1981), pp. 366–381.

[8] G. Sugihara and R. M. May, Nonlinear forecasting as a way of distinguishing chaos from measurement error in time series, Nature (London) **344**, 734 (1990).

[9] M. Ragwitz and H. Kantz, Markov models from data by simple nonlinear time series predictors in delay embedding spaces, Phys. Rev. E **65**, 056201 (2002).

[10] S. P. Garcia and J. S. Almeida, Multivariate phase space reconstruction by nearest neighbor embedding with different time delays, Phys. Rev. E **72**, 027205 (2005).

[11] T. Qin, K. Wu, and D. Xiu, Data driven governing equations approximation using deep neural networks, J. Comput. Phys. **395**, 620 (2019).

[12] M. Raissi, P. Perdikaris, and G. E. Karniadakis, Multistep neural networks for data-driven discovery of nonlinear dynamical systems, arXiv:1801.01236.

[13] B. Lusch, J. N. Kutz, and S. L. Brunton, Deep learning for universal linear embeddings of nonlinear dynamics, Nat. Commun. **9**, 4950 (2018).

[14] E. Weinan, C. Ma, S. Wojtowytsch, and L. Wu, Towards a mathematical understanding of neural network-based machine learning: What we know and what we don't, arXiv:2009.10713.

[15] A. L. Caterini and D. E. Chang, *Deep Neural Networks in a Mathematical Framework* (Springer, Gewerbestrasse, Cham, Switzerland, 2018).

[16] D. Mandic and J. Chambers, *Recurrent Neural Networks for Prediction: Learning Algorithms, architectures and Stability* (Wiley, Chichester, West Sussex, England, 2001).

[17] J. Connor, R. Martin, and L. Atlas, Recurrent neural networks and robust time series prediction, IEEE Trans. Neural Networks **5**, 240 (1994).

[18] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning* (MIT Press, Cambridge, MA, 2016).

[19] R. J. Williams and D. Zipser, A learning algorithm for continually running fully recurrent neural networks, Neural Comput. **1**, 270 (1989).

[20] B. Cheng and H. Tong, Orthogonal projection, embedding dimension and sample size in chaotic time series from a statistical perspective, Philos. Trans. R. Soc. London. Series A: Phys. Eng. Sci. **348**, 325 (1994).

[21] S. B. Munch, V. Poynor, and J. L. Arriaza, Circumventing structural uncertainty: A bayesian perspective on nonlinear forecasting for ecology, Ecol. Complexity **32**, 134 (2017), uncertainty in Ecology.

[22] H. Kantz and E. Olbrich, Scalar observations from a class of high-dimensional chaotic systems: Limitations of the time delay embedding, Chaos **7**, 423 (1997).

[23] J. L. Elman, Finding structure in time, Cognitive science **14**, 179 (1990).

[24] M. Han, Z. Shi, and W. Wang, Modeling dynamic system by recurrent neural network with state variables, *International Symposium on Neural Networks* (Springer, Berlin, Heidelberg, 2004), pp. 200–205.

[25] H. Zimmermann and R. Neuneier, Modeling dynamical systems by recurrent neural networks, in *WIT Transactions on Information and Communication Technologies*, edited by C. A. Brebbia and N. F. F. Ebecken, Vol. 25 (WIT Press, 2000).

[26] E. R. Deyle, M. Fogarty, C.-h. Hsieh, L. Kaufman, A. D. MacCall, S. B. Munch, C. T. Perretti, H. Ye, and G. Sugihara, Predicting climate effects on Pacific sardine, Proc. Natl. Acad. Sci. USA **110**, 6430 (2013).

[27] S. B. Munch, A. Giron-Nava, and G. Sugihara, Nonlinear dynamics and noise in fisheries recruitment: A global meta-analysis, Fish Fisheries **19**, 964 (2018).

[28] J. M. Morales, D. T. Haydon, J. Frair, K. E. Holsinger, and J. M. Fryxell, Extracting more out of relocation data: Building movement models as mixtures of random walks, Ecology **85**, 2436 (2004).

[29] G. Hinton, N. Srivastava, and K. Swersky, Neural networks for machine learning lecture 6a overview of mini-batch gradient descent (unpublished).

[30] L. Prechelt, Automatic early stopping using cross validation: Quantifying the criteria, Neural Networks **11**, 761 (1998).

[31] E. N. Lorenz, Deterministic nonperiodic flow, J. Atmospheric Sci. **20**, 130 (1963).

[32] G. Duffing, *Erzwungene Schwingungen bei veränderlicher Eigenfrequenz und ihre Technische Bedeutung* (Vieweg, Braunschweig, Germany, 1918), pp. 41 and 42.

[33] E. N. Lorenz, Predictability: A problem partly solved, in *Proceedings of the Seminar on Predictability* (ECMWF, Reading, UK, 1996), Vol. 1.

[34] A. J. Lotka, Contribution to the theory of periodic reactions, J. Phys. Chem. **14**, 271 (1910).

[35] A. Chattopadhyay, P. Hassanzadeh, and D. Subramanian, Data-driven predictions of a multiscale lorenz 96 chaotic system using machine-learning methods: Reservoir computing, artificial neural network, and long short-term memory network, Nonlinear Processes Geophys. **27**, 373 (2020).

[36] P. D. Dueben and P. Bauer, Challenges and design choices for global weather and climate models based on machine learning, Geosci. Model Dev. **11**, 3999 (2018).