Journal of Machine Learning for Modeling and Computing, 3(3):97–119 (2022)

DEEP LEARNING OF CHAOTIC SYSTEMS FROM PARTIALLY-OBSERVED DATA

Victor Churchill & Dongbin Xiu*

Department of Mathematics, The Ohio State University, Columbus, 43210 Ohio, USA

*Address all correspondence to: Victor Churchill, Department of Mathematics, The Ohio State University, Columbus, OH 43210, USA, E-mail: churchill.77@osu.edu

Original Manuscript Submitted: 8/26/2022; Final Draft Received: 10/11/2022

Recently, a general data-driven numerical framework was developed for learning and modeling of unknown dynamical systems using fully- or partially-observed data. The method utilizes deep neural networks (DNNs) to construct a model for the flow map of the unknown system. Once an accurate DNN approximation of the flow map is constructed, it can be recursively executed to serve as an effective predictive model of the unknown system. In this paper, we apply this framework to chaotic systems, in particular the well-known Lorenz 63 and 96 systems, and critically examine the predictive performance of the approach. A distinct feature of chaotic systems is that even the smallest perturbations will lead to large (albeit bounded) deviations in the solution trajectories. This makes long-term predictions of the method, or any data-driven methods, questionable, as the local model accuracy will eventually degrade and lead to large pointwise errors. Here we employ several other qualitative and quantitative measures to determine whether the chaotic dynamics has been learned. These include phase plots, histograms, autocorrelation, correlation dimension, approximate entropy, and Lyapunov exponent. Using these measures, we demonstrate that the flow map based DNN learning method is capable of accurately modeling chaotic systems, even when only a subset of the state variables is available to the DNNs. For example, for the Lorenz 96 system with 40 state variables, when the data of only three variables are available, the method is able to learn an effective DNN model for the three variables and produce accurately the chaotic behavior of the system.

KEY WORDS: *deep neural networks, chaotic behavior, flow map*

1. INTRODUCTION

Due to recent advances in machine learning software and computing hardware combined with the availability of vast amounts of data, data-driven learning of unknown dynamical systems has been a very active research area in the past few years. One way to approach this problem is governing equation discovery, where a map is constructed from state variables to their time derivatives. Among other techniques, this can be achieved via sparse approximation, where under certain circumstances exact equation recovery is possible. See, for example, Brunton et al. (2016) and its many extensions in recovering both ODEs in Brunton et al. (2016), Kang et al. (2019), Schaeffer and McCalla (2017), Schaeffer et al. (2018), and Tran and Ward (2017) and PDEs in

2689–3967/22/$35.00

Rudy et al. (2017) and Schaeffer (2017). Deep neural networks (DNNs) have also been used to construct this mapping. See, for example, ODE modeling in Lu et al. (2021b), Qin et al. (2019), Raissi et al. (2018), and Rudy et al. (2019) and PDE modeling in Long et al. (2018a,b), Lu et al. (2021a), Raissi et al. (2017a,b), Raissi (2018), and Sun et al. (2019).

Another approach for learning unknown systems, our focus here, is flow map or evolution discovery, where a map is constructed between two system states separated by a short time to approximate the flow map of the system (Qin et al., 2019). Unlike governing equation discovery, approximating the flow map does not directly yield the specific terms in the underlying equations. Rather, if an accurate flow map is discovered, then an accurate predictive model can be defined for the evolution of the unknown system such that a new initial condition can be marched forward in time. This approach relies on a DNN, in particular a residual network (ResNet) (He et al., 2016), to approximate the flow map. Since its introduction in Qin et al. (2019) to model autonomous systems, a general framework has been developed for the flow map approximation of unknown systems from their trajectory data that extends to nonautonomous systems (Qin et al., 2021a), parametric dynamical systems (Qin et al., 2021b), partially observed dynamical systems (Fu et al., 2020), as well as partial differential equations (Chen et al., 2022; Wu and Xiu, 2020). Of particular interest in this paper is Fu et al. (2020), where a finite memory of the state variable time history is used to learn reduced systems where only some of the state variables are observed per the Mori–Zwanzig formulation.

The focus of this paper is to extend this flow map deep learning framework to *chaotic systems* and examine its performance for both fully and partially observed chaotic systems. Chaotic systems exhibit ultrasensitivity to perturbations of the system parameters and initial conditions. Hence, they represent a highly challenging case for any learning and modeling methods, particularly for long-term system behavior. Although well known chaotic systems, e.g., the Lorenz 63 system, have been adopted in the literature, they were mostly used as demonstrative examples of the proposed methods in the papers by using visual examination of phase plots. The nearly impossible task of matching the long-term evolution of the true chaotic system is rarely addressed in the existing literature. With an exclusive focus on chaotic systems, the purpose of this paper is to systematically examine the long-term predictive accuracy using a set of measures beyond phase plots. These include bounded pointwise error, histograms, autocorrelation functions, correlation dimension, approximate entropy, and Lyapunov exponent. Using these measures, we further establish that the flow map based deep learning method is capable of learning and modeling chaotic systems and producing accurate long-term system predictions, for both fully- and partially-observed systems.

1.1 Literature Review

The problem of learning fully- and partially-observed chaotic systems, especially the famous Lorenz 63 system, has been a popular topic particularly in the age of machine learning. Hence, we limit the scope of this review to papers particularly relevant to the subject of learning chaotic dynamics.

The recent paper by Bhat and Munch (2022) is most similar to this work in that the authors seek to learn partially-observed chaotic dynamics using memory. This paper considers observing just one variable, while we consider many combinations of partially-observed variables, and focuses on optimizing the network parameters (e.g., number of neurons and memory length) of a particular recursive structure based on root mean squared error (RMSE), while we consider additional measures of chaotic dynamics and a general ResNet. There are also several

other relevant papers including Trischler and D'Eleuterio (2016), which uses a recurrent neural network to predict chaotic systems including Lorenz 63. In Wulkow et al. (2021), the authors discuss memory length and Takens' theorem (Takens, 1981) in the context of learning Lorenz 96. In addition, Scher and Messori (2019) consider approximating chaotic dynamical systems using limited data and external forcing. Also, Vlachas et al. (2018), Pawar et al. (2021), Chattopadhyay et al. (2020), and Dubois et al. (2020) use long short-term memory (LSTM) recurrent networks to study fully- and partially-observed Lorenz systems.

There are also several earlier papers, including Zimmermann and Neuneier (2000) which proposes modeling dynamical systems via recurrent neural networks, Kim et al. (1999) which deals with estimating time delays (memory length) in Lorenz and other systems, Bakker et al. (2000) that learns chaotic dynamics of reduced systems via neural networks, Miyoshi et al. (1995) which learns chaotic dynamics with neural networks, and Han et al. (2004a,b), which predict the chaotic Rossler system using a simple RNN. These papers are certainly relevant, but lack the extensive examples and systematic verification experiments that are now more easily achieved with today's computing systems.

It is also worth mentioning a large body of work that uses a different approach (governing equation discovery) to learning chaotic systems. In Brunton et al. (2016), the authors approximate the particular terms in chaotic systems (the right-hand sides) through sparse optimization or network learning. Other work in this area includes Brunton et al. (2017) and Lusch et al. (2018) which focus on learning delayed embeddings, Raissi et al. (2018) which uses physics-informed neural networks, Pan and Duraisamy (2018), which explicitly learns the closure of the partially observed Lorenz 63 system via a NN, Champion et al. (2019) which combines learning a coordinate transform from the delayed embedding coordinates with learning the dynamic coordinates of chaotic systems, Rudy et al. (2019) which learns the Lorenz system from noisy data, and Bakarji et al. (2022) which learns the governing equations for Lorenz (or a Lorenz-like surrogate) from just one observation variable.

1.2 Contributions

The chief contribution of this paper is a systematic and rigorous examination of learning the flow maps of fully- and partially-observed chaotic dynamical systems using an approachable and mathematically grounded DNN framework. In most papers dealing with this topic, pointwise error or a visual comparison of phase plots is used to assess the accuracy of network prediction. However, it is well known that chaotic systems are extremely sensitive to perturbation, and therefore in the long run the model predictions will dramatically stray from the truth. This makes assessing the efficacy of the learned system, particularly for long-term, a challenging problem in and of itself. Hence, part of our contribution is the proposal of a more robust approach to the assessment of learning chaotic behavior not explored in the existing literature that includes standard techniques of error analysis such as pointwise error and phase plots as well as comparison of a variety of other measures that demonstrate accurate behavior, including matching of histograms and autocorrelation, and statistics that quantify chaos such as correlation dimension, approximate entropy, and Lyapunov exponent. Finally, we contribute several ambitious numerical examples not explored in the literature. As a benchmark, we consider learning the well-known Lorenz systems with different combinations of observed variables. Of particular note is a 40-dimensional Lorenz 96 system with only three variables observed, where we demonstrate that the DNN method can learn the chaotic behavior in general despite training data being collected from a single long trajectory consisting of only three variables (out of 40).

2. FLOW MAP MODELING OF UNKNOWN CHAOTIC DYNAMICAL SYSTEMS

We are interested in constructing effective models for the evolution laws behind chaotic dynamical data. We follow the framework in Qin et al. (2019) and Fu et al. (2020) for learning fully- and partially-observed dynamical systems via residual DNNs. The following review closely follows that of Churchill et al. (2022). Throughout this paper our discussion will be on dynamical systems observed over discrete time instances with a constant time step Δt,

$$t_0 < t_1 < \cdots, \qquad t_{n+1} - t_n = \Delta t, \quad \forall n. \tag{1}$$

Generality is not lost with the constant time step assumption, as the variable time step can be treated as a separate entry to the DNN structure (Qin et al., 2021a). We will use a subscript to denote the time variable of a function, e.g., $\mathbf{x}_n = \mathbf{x}(t_n)$.

2.1 ResNet Modeling of Fully-Observed Systems

Consider an unknown autonomous system,

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}), \qquad \mathbf{x} \in \mathbb{R}^d, \tag{2}$$

where $\mathbf{f} : \mathbb{R}^d \to \mathbb{R}^d$ is not known. Because it is autonomous, its flow map depends only on the time difference as opposed to the actual time, i.e., $\mathbf{x}_n = \mathbf{\Phi}_{t_n - t_s}(\mathbf{x}_s)$. Thus, the solution having been marched forward one time step satisfies

$$\mathbf{x}_{n+1} = \mathbf{\Phi}_{\Delta t}(\mathbf{x}_n) = \mathbf{x}_n + \mathbf{\Psi}_{\Delta t}(\mathbf{x}_n), \tag{3}$$

where $\mathbf{\Psi}_{\Delta t} = \mathbf{\Phi}_{\Delta t} - \mathbf{I}$, with $\mathbf{I}$ as the identity operator.

When data for all of the state variables $\mathbf{x}$ over the time stencil (1) are available, they can be grouped into sequences,

$$\{\mathbf{x}^{(m)}(0), \mathbf{x}^{(m)}(\Delta t), \ldots, \mathbf{x}^{(m)}(K\Delta t)\}, \quad m = 1, \ldots, M,$$

where M is the total number of such data sequences and $K + 1$ is the length of each sequence (which is assumed to be a constant for notational convenience). This serves as the training dataset. Inspired by basic numerical schemes for solving ODEs, one can model the unknown evolution operator using a residual network (ResNet) (He et al., 2016) in the form of

$$\mathbf{y}^{out} = [\mathbf{I} + \mathbf{N}](\mathbf{y}^{in}), \tag{4}$$

where $\mathbf{N} : \mathbb{R}^d \to \mathbb{R}^d$ stands for the mapping operator of a standard feedforward fully connected neural network. The network is then trained by using the training dataset and minimizing the recurrent mean squared loss function

$$\frac{1}{M} \sum_{m=1}^{M} \sum_{k=1}^{K} \left\| \mathbf{x}^{(m)}(k\Delta t) - [\mathbf{I} + \mathbf{N}]^k(\mathbf{x}^{(m)}(0)) \right\|^2, \tag{5}$$

where $[\mathbf{I} + \mathbf{N}]^k$ indicates composition of the network function k times. Recurrent loss is used to increase the stability of the network approximation over long-term prediction. The trained network thus accomplishes

$$\mathbf{x}^{(m)}(k\Delta t) \approx [\mathbf{I} + \mathbf{N}]^k(\mathbf{x}^{(m)}(0)), \qquad \forall m = 1, \ldots, M, \ k = 1, \ldots, K.$$

After the network is trained to a satisfactory accuracy level, it can then be used as a predictive model,

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \mathbf{N}(\mathbf{x}_n), \qquad n = 0, 1, \ldots, \tag{6}$$

starting from any initial condition $\mathbf{x}(t_0)$. This framework was proposed in Qin et al. (2019) and was extended to parametric systems and time-dependent (nonautonomous) systems (Qin et al., 2021a,b).

2.2 Memory-Based ResNet Modeling of Partially-Observed Systems

A notable extension of the flow-map modeling in Qin et al. (2019) is to partially-observed systems where some state variables are not observed at all. Let $\mathbf{x} = (\mathbf{z}; \mathbf{w})$, where $\mathbf{z} \in \mathbb{R}^m$ and $\mathbf{w} \in \mathbb{R}^{d-m}$. Let $\mathbf{z}$ be the observables and $\mathbf{w}$ be the missing variables. That is, no information or data of $\mathbf{w}$ are available. When data are available only for $\mathbf{z}$, it is possible to derive a system of equations for $\mathbf{z}$ only via the Mori–Zwanzig formulation (Mori, 1965; Zwanzig, 1973). However, the Mori–Zwanzig formulation asserts that the reduced system for $\mathbf{z}$ requires a memory integral, whose kernel function, along with other terms in the formula, is unknown. By making a mild assumption that the memory is of finite (problem-dependent) length, memory-based DNN structures were investigated in Wang et al. (2020) and Fu et al. (2020). While Wang et al. (2020) utilized LSTM networks, Fu et al. (2020) proposed a relatively simple DNN structure, in direct correspondence to the Mori–Zwanzig formulation, that takes the following mathematical form,

$$\mathbf{z}_{n+1} = \mathbf{z}_n + \mathbf{N}(\mathbf{z}_n, \mathbf{z}_{n-1}, \ldots, \mathbf{z}_{n-n_M}), \qquad n \geq n_M, \tag{7}$$

where $n_M \geq 0$ is the number of memory terms in the model. In this case, the DNN operator is $\mathbf{N} : \mathbb{R}^{m\times(n_M+1)} \to \mathbb{R}^m$, which corresponds to a ResNet with additional time history inputs. The special case of $n_M = 0$ corresponds to the standard ResNet model (6) for modeling fully-observed systems without missing variables (thus no need for memory).

In the case of $m = 1$, Takens' theorem (Takens, 1981) proves (nonconstructively) the existence of a map from $F : \mathbb{R}^{n_M+1} \to \mathbb{R}$ such that $z_{n+1} = F(z_n, \ldots, z_{n-n_M})$ provided $n_M \geq 2d$. This serves as inspiration that, given enough data, a universal approximator can find this map. However, in this paper we consider many values for m corresponding to different combinations of observed variables.

3. COMPUTATIONAL FRAMEWORK

In the main task of this paper, we apply the flow map learning methods described in the previous section to chaotic systems. First, we review the setting including the DNN structure used as well as the specifics of the data generation and model training. Next, we discuss the qualitative and quantitative metrics to evaluate the flow map learning. We then present extensive numerical results for learning fully- and partially-observed Lorenz systems.

3.1 DNN Structure

The structure of the DNNs used to achieve chaotic flow map learning is modeled by Eq. (7) (Fu et al., 2020). The network function $\mathbf{N} : \mathbb{R}^{m\times(n_M+1)} \to \mathbb{R}^m$ maps the input time history of the observed variables to the output future time through a series of fully-connected (also known as dense) layers with ReLU activation. An illustration of this structure with memory

length $n_M = 2$ is shown in Fig. 1. Note that this structure is general in that while it is built for partially-observed systems that require memory based on the Mori–Zwanzig formulation, setting the memory length $n_M = 0$ returns a standard ResNet that is appropriate for learning fully-observed systems. Our extensive numerical experimentation and previous work with this framework indicate that particularly wide or deep networks are not typically necessary for learning with this structure, and hence most examples in this paper use three hidden layers with 20 neurons in each layer. The choice of memory length n_M is in general problem-dependent and hinges on a number of factors including the time step and the relationship between the observed and missing variables, and is explored in detail in Fu et al. (2020). If the memory length is too short, accuracy suffers because there is still missing information. If the memory length is too long, the number of trainable parameters in the network increases as does the difficulty of the optimization to usefully utilize all of the inputs, which can also result in a loss of accuracy. Therefore, in practice it requires thorough testing and model comparison to find an appropriate memory length. In the examples in this paper, there are typically around $n_M = 10$ time steps (equivalent to 0.1 s in time as discussed below).

3.2 Data Generation and Model Training

For benchmarking purposes, in all examples the true chaotic systems we seek to approximate are known. However, these true models serve only two purposes: (1) to generate synthetic data with which to train the DNN flow map approximations, and (2) to generate reference solutions for comparison with DNN predictions in testing. Therefore, the knowledge of the true system does not in any way facilitate the DNN model approximation.

Data generation for both of these tasks is achieved by solving the true systems using a high-order numerical solver, and observing this reference solution at discrete time steps with $\Delta t = 0.01$ s. To generate the training data, a single initial condition $\mathbf{x}(0)$ generates a long trajectory from $t = 0$ to $t = 10{,}000$ s (1,000,000 time steps). From this long trajectory, M sequences of length $n_M + K + 1$ are collected uniformly at random to form the training dataset

$$\{\mathbf{x}^{(m)}_{n-n_M}, \ldots, \mathbf{x}^{(m)}_{n-1}, \mathbf{x}^{(m)}_{n}, \mathbf{x}^{(m)}_{n+1}, \ldots, \mathbf{x}^{(m)}_{n+K}\}, \quad m = 1, \ldots, M, \tag{8}$$

where n_M is the memory length and K is the recurrent loss parameter. In our examples, typically $M = 10{,}000$ and $K = 10$ time steps (equivalent to 0.1 s). For systems requiring memory, data are

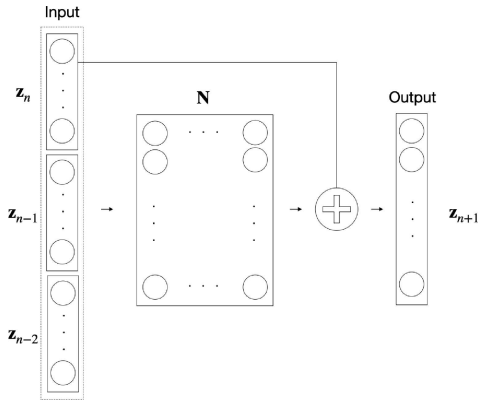

FIG. 1: Memory-based DNN structure

generated from the full true system for all state variables $\mathbf{x}$, with only the data for the observed state variables $\mathbf{z}$ being kept and data for the missing variables $\mathbf{w}$ being discarded.

Once the training dataset has been generated, the learning task is achieved by training the DNN model (7) with the data set (8). In particular, the network hyperparameters (weights and biases) are trained by minimizing the recurrent loss function,

$$\frac{1}{M}\sum_{m=1}^{M}\sum_{k=1}^{K}\left\|\mathbf{x}_{n+k}^{(m)} - [\mathbf{I}_n + \mathbf{N}]^k(\mathbf{x}_n^{(m)},\ldots,\mathbf{x}_{n-n_M}^{(m)})\right\|^2, \tag{9}$$

where $\mathbf{I}_n(\mathbf{x}_n,\ldots,\mathbf{x}_{n-n_M}) = \mathbf{x}_n$, using the stochastic optimization method Adam (Kingma and Ba, 2014). In the examples below we typically train for 10,000 epochs with batch size 50 and a constant learning rate of 10^{-3} in Tensorflow (Abadi et al., 2015).

3.3 DNN Prediction and Validation

After satisfactory network training, we obtain a predictive model (7) for the unknown system which can be marched forward in time from any new initial condition. To validate the network prediction, testing data are generated in the same manner as training data above. In particular, a *new* initial condition generates a reference solution from $t = 0$ to $t = 100$ s (10,000 time steps) using the true governing equations. For prediction, the DNN model is marched forward starting with the first $n_M + 1$ time steps of the test trajectory until $t = 100$ and is compared against the reference. Note that we march forward significantly longer than $K\Delta t$ (the length of each training sequence which is typically $t = 0.1$) to examine the long-term system behavior.

For chaotic systems, it is practically impossible to approximate the flow map with a model that achieves low pointwise error in the long term due to the fact that even machine epsilon changes in the initial condition or other system parameters will drastically change the evolution of the system. However, these changes will not alter the physics of the system, i.e. the nature of the behavior of the system. Hence, in this study, we recognize the challenge of low long-term pointwise error and focus on learning the physics of the system to match the chaotic behavior. In particular, we look at a myriad of qualitative and quantitative metrics in order to demonstrate a strong match of physics including pointwise error, phase plot, histogram, autocorrelation function, correlation dimension, approximate entropy, and Lyapunov dimension. These tools have been used to classify chaotic behavior in Lorenz and other chaotic systems, e.g., in Bakker et al. (2000), Chattopadhyay et al. (2020), Kim et al. (1999), and Rudy et al. (2019). We briefly review each of the evaluation tools below. All metrics are computed using MATLAB (MATLAB, 2022).

- Pointwise error: In the following examples, we look at the reference test trajectories versus the trajectories predicted by the network model. However, the predictions will quickly deviate from the reference trajectories since the dynamics we learn are in fact an approximation and at best some small perturbation away from the true dynamics.[†] For a chaotic system, this small perturbation causes drastic changes, which stay bounded, in the system later in time. Hence, we can also look at the log absolute error between the trajectories, and check that this quantity remains bounded. Bounded pointwise error demonstrates stability of the predictive model.

[†]For that matter, even the reference trajectories themselves are just an approximation of the unknown true trajectories as they are generated with a high-order numerical solver rather than analytically solving the system.

- Phase plot: We also qualitatively compare the phase plots of reference and predicted test trajectories when multiple state variables are observed. This can serve as a qualitative measure of the behavior of the system, e.g., if both systems exhibit two attractors centered at the same particular points.

- Histogram: We compare the approximate densities of reference and predicted values for the trajectories as well, where a match in the distribution indicates similar behavior in the long term. The histograms are computed over all time steps starting with the initial condition.

- The **autocorrelation function** (Box et al., 2015) measures the correlation between the time series x_t and its lagged counterpart x_{t+k}, where $k = 0, \ldots, K$ and x_t is a stochastic process. The autocorrelation for lag k is defined as $r_k = c_k/c_0$ where

$$c_k = \frac{1}{T}\sum_{t=1}^{T-k}(x_t - \bar{x})(x_{t+k} - \bar{x}), \tag{10}$$

  and c_0 is the sample variance of the time series, with $\bar{x}$ the mean of the time series and T the length of the time series. In our implementation, the sample autocorrelation is computed by MATLAB's Econometrics Toolbox (The MathWorks, 2022a) using the default settings. See this link[‡] for further computational details.

- The **correlation dimension** (Theiler, 1987) is a measure of chaotic signal complexity in multidimensional phase space. Specifically, it is a measure of the dimensionality of the space occupied by a set of random points, whereby a higher correlation dimension represents a higher level of chaotic complexity in the system. It is computed by first generating a delayed reconstruction $Y_{1:N}$ with embedding dimension m (in our implementation m is set to be the dimension of the full system regardless of whether the system is partially- or fully-observed) and lag τ of reference or predicted trajectories assembled in a matrix X. The number of within range points, at point i, is calculated by

$$N_i(R) = \sum_{i=1, i\neq k}^{N} 1(\|Y_i - Y_k\|_\infty < R), \tag{11}$$

  where 1 is the indicator function, R is the radius of similarity, and N is the number of points used to compute R. The correlation dimension is the slope of $C(R)$ vs. R, where the correlation integral is defined as

$$C(R) = \frac{2}{N(N-1)}\sum_{i=1}^{N} N_i(R). \tag{12}$$

  In our implementation, the correlation dimension is computed by MATLAB's Predictive Maintenance Toolbox (The MathWorks, 2022b) using default settings (e.g., for N and R). See this link[§] for further computational details.

[‡]https://www.mathworks.com/help/econ/autocorr.html
[§]https://www.mathworks.com/help/predmaint/ref/correlationdimension.html

- The **approximate entropy** (Pincus, 1991) is a measure used to quantify the amount of regularity and unpredictability of fluctuations over a nonlinear time series. It is computed as $\Phi_m - \Phi_{m+1}$, where

$$\Phi_m = (N - m + 1)^{-1} \sum_{i=1}^{N-m+1} \log(N_i(R)), \tag{13}$$

  where m, N, R, and $N_i(R)$ are defined as above when discussing correlation dimension using the same delayed reconstruction. In our implementation, the approximate entropy is computed by MATLAB's Predictive Maintenance Toolbox (The MathWorks, 2022b) using default settings. See this link[¶], which contains a working example of the Lorenz 63 system, for further computational details.

- The **Lyapunov exponent** (Rosenstein et al., 1993) characterizes the rate of separation of infinitesimally close trajectories in phase space to distinguish different attractors, which can be useful in quantifying the level of chaos in a system. A positive Lyapunov exponent indicates divergence and chaos, with the magnitude indicating the rate of divergence. For some point i, the Lyapunov exponent is computed using the same delayed reconstruction $Y_{1:N}$ as the correlation dimension and approximate entropy as

$$\lambda(i) = \frac{1}{(K_{\max} - K_{\min} + 1)dt} \sum_{K=K_{\min}}^{K_{\max}} \frac{1}{K} \ln \frac{\|Y_{i+K} - Y_{i^*+K}\|_2}{\|Y_i - Y_{i^*}\|_2}, \tag{14}$$

  where $K_{\min}$ and $K_{\max}$ represent the expansion range, dt is the sample time (equal to $\Delta t = 0.01$ in our case), and i^* is the nearest neighbor point to i satisfying a minimum separation. A single scalar for the Lyapunov exponent is then computed as the slope of a linear fit to the $\lambda(i)$ values. In our implementation, the Lyapunov exponent is computed by MATLAB's Predictive Maintenance Toolbox (The MathWorks, 2022b) using sampling frequency of 100 (corresponding to $\Delta t = 0.01$) and otherwise default settings. See this link[||], which also contains a working example of the Lorenz 63 system, for computational details.

We note that while the computational specifics of each of these evaluation tools are in fact tunable and indeed changing them would yield different values, the comparisons that follow are not confined to accuracy for these particular default implementation choices and we present them simply as an example of one configuration.

4. COMPUTATIONAL RESULTS

In this section we provide numerical examples of DNN modeling of chaotic systems. We focus on two well-known systems: the three-dimensional Lorenz 63 system and the 40-dimensional Lorenz 96 system. In each system, we start with the learning of fully-observed variables to demonstrate the effectiveness of ResNet learning. We then focus on partially-observed cases. For the three-dimensional Lorenz 63 system, we examine the DNN learning using training data of (different combinations of) only two variables, as well as of only one variable. For the 40-dimensional Lorenz 96 system, we examine the DNN learning when only three variables are observed in the training data.

[¶] https://www.mathworks.com/help/predmaint/ref/approximateentropy.html

[||] https://www.mathworks.com/help/predmaint/ref/lyapunovexponent.html

4.1 Low-Dimensional System: Lorenz 63

The Lorenz 63 system is a nonlinear deterministic chaotic three-dimensional system,

$$\begin{aligned} \frac{dx}{dt} &= \sigma(y - x), \\ \frac{dy}{dt} &= x(\rho - z) - y, \\ \frac{dz}{dt} &= xy - \beta z, \end{aligned} \tag{15}$$

where σ, ρ, and β are parameters. It was proposed in Lorenz (1963) as a simplified model for atmospheric convection. When $\sigma = 10$, $\rho = 28$, and $\beta = 8/3$, the system exhibits chaotic behavior and is a widely studied case.

4.1.1 Example 1: Full Three-Dimensional System

In this example all three state variables x, y, and z are observed and stored in the training data set. In particular, a trajectory is generated by a high-order numerical solver and observed at every $\Delta t = 0.01$ starting from the initial condition $(x_0, y_0, z_0) = (1, 1, 1)$ until $T = 10{,}000$ s. From this long trajectory, 10,000 data sequences of length 0.1 s (11 time steps) are taken as training data, which allows 10 steps to be used for recurrent loss. A standard ResNet with three hidden layers with 20 neurons each is used, and the mean squared recurrent loss function is minimized using Adam with a constant learning rate of 10^{-3} for 10,000 epochs.

Prediction is carried out to $T = 100$ s (10,000 time steps) from a new initial condition $(10, 10, 20)$. Figure 2 shows the trajectory prediction as well as the log absolute error. We see that despite how the prediction trajectories quickly deviate from the reference trajectories (as should be expected for a chaotic system), the pointwise error in all three variables remains bounded over the long term. In addition, Figs. 3–5 show qualitatively similar phase plots, histograms, and autocorrelation functions, with at worst the quantitative chaos statistics of 9% in relative error. See Table 1 for details.

4.1.2 Example 2: Reduced One- or Two-Dimensional Systems

In this example we consider the six possible combinations of partially-observed systems arising from Lorenz 63. Specifically, we consider observing two variables of only x and y, only x and z, and only y and z, as well as one variable of only x, only y, and only z. When variables are missing, the Mori–Zwanzig formulation informs us that memory is required to learn the reduced system dynamics. Hence, in all of the following experiments a trajectory of the full Lorenz system is generated by a high-order numerical solver and the observed variables are observed at every $\Delta t = 0.01$ starting from the initial condition $(1, 1, 1)$ until $T = 10{,}000$ s. (The unobserved variables are discarded from the true system solutions.) From this long trajectory, 10,000 data sequences of length 0.2 s (21 time steps) are taken from only the observed variable(s) as training data. This allows 0.1 s (10 steps) for memory and 0.1 s (10 steps) for recurrent loss. A standard ResNet with three hidden layers and 20 neurons each is used, and the mean squared recurrent loss function is minimized using Adam with a constant learning rate of 10^{-3} for 10,000 epochs.

Prediction for the observed variables is carried out to $T = 100$ s (10,000 time steps) from a new initial condition $(10, 10, 20)$, with the corresponding observables in different cases. The

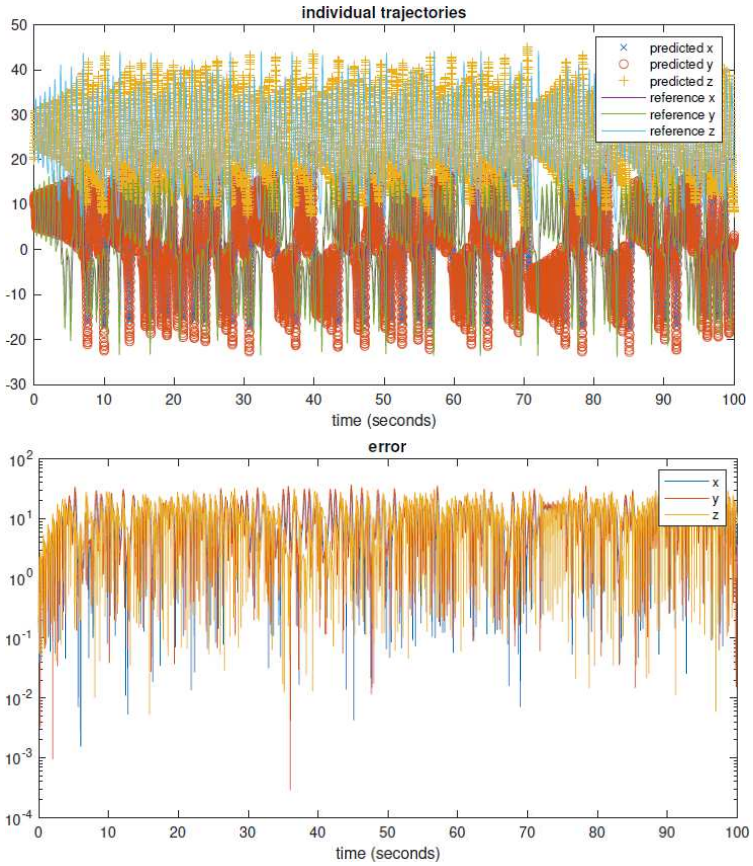

FIG. 2: Ex. 1: Lorenz 63 full system—individual trajectory comparison and pointwise error

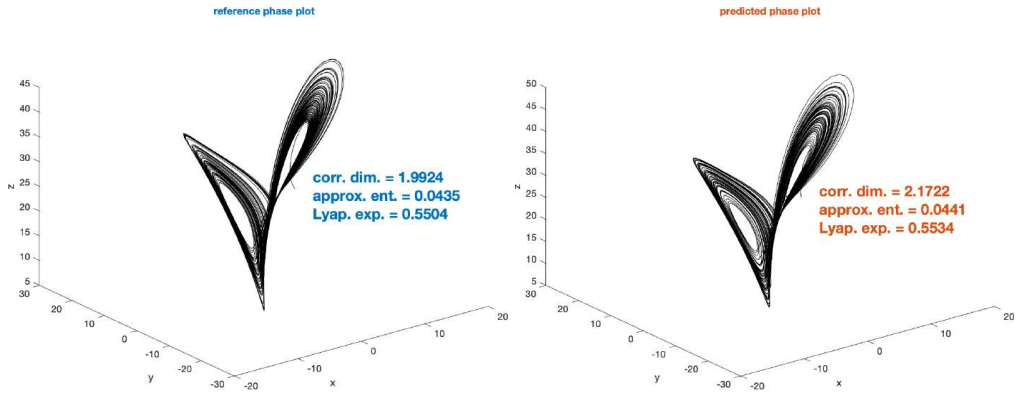

FIG. 3: Ex. 1: Lorenz 63 full system—phase plots (left: reference; right: DNN prediction)

two-variable x and y system is shown in Figs. 6–9. We see bounded pointwise error indicating stability, qualitatively similar run in phase plots indicating similar behavior, a similar histogram indicating the appropriate density, and chaos statistics very close to the reference.

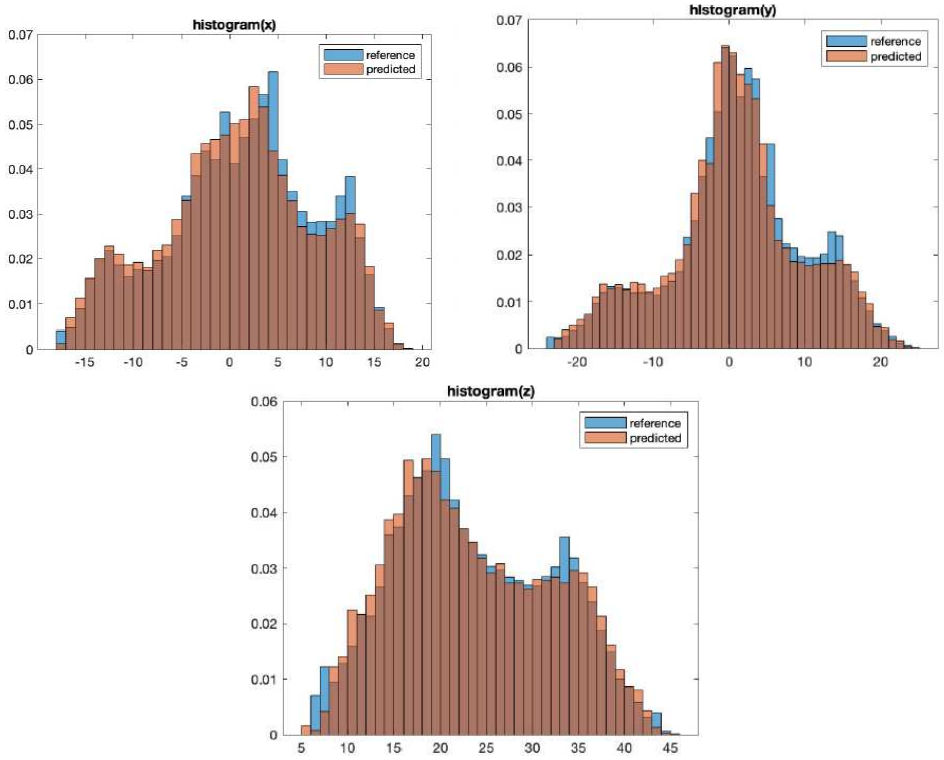

FIG. 4: Ex. 1: Lorenz 63 full system—histogram comparison

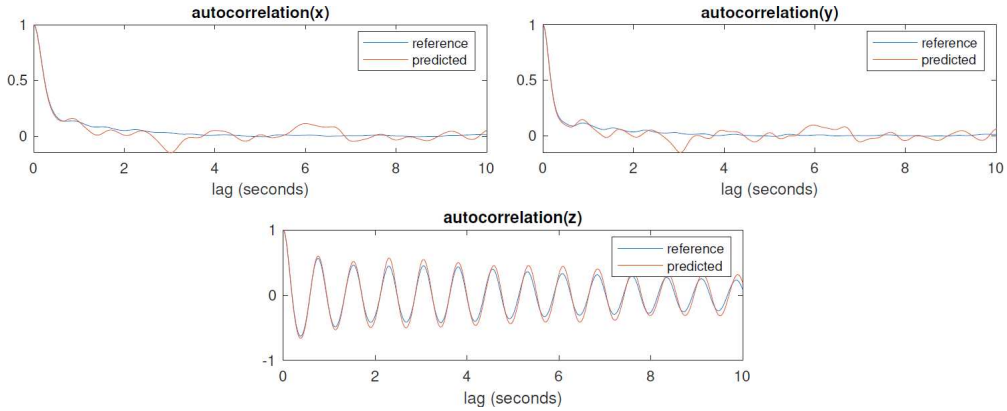

FIG. 5: Ex. 1: Lorenz 63 full system—autocorrelation function comparison

The one-variable z system is shown in Figs. 10 and 11. Once again we see bounded pointwise error, similar histograms and autocorrelation functions, and very accurate chaos statistics. The correlation dimension, approximate entropy, and Lyapunov exponent values for all of the combinations of observed variables are reported in Table 2. The predicted statistics are at worst 17.5% off from the reference statistics in relative error, but are typically significantly lower, and we note that in the few cases where relative error exceeds 10% it is only in one of the three

TABLE 1: Ex. 1: Lorenz 63 full system—metrics for chaotic behavior comparison

Metrics	Reference Solution	DNN Prediction	Relative Error
Correlation dimension	1.9924	2.1722	9.0%
Approximate entropy	0.0435	0.0441	1.4%
Lyapunov exponent	0.5504	0.5534	0.5%

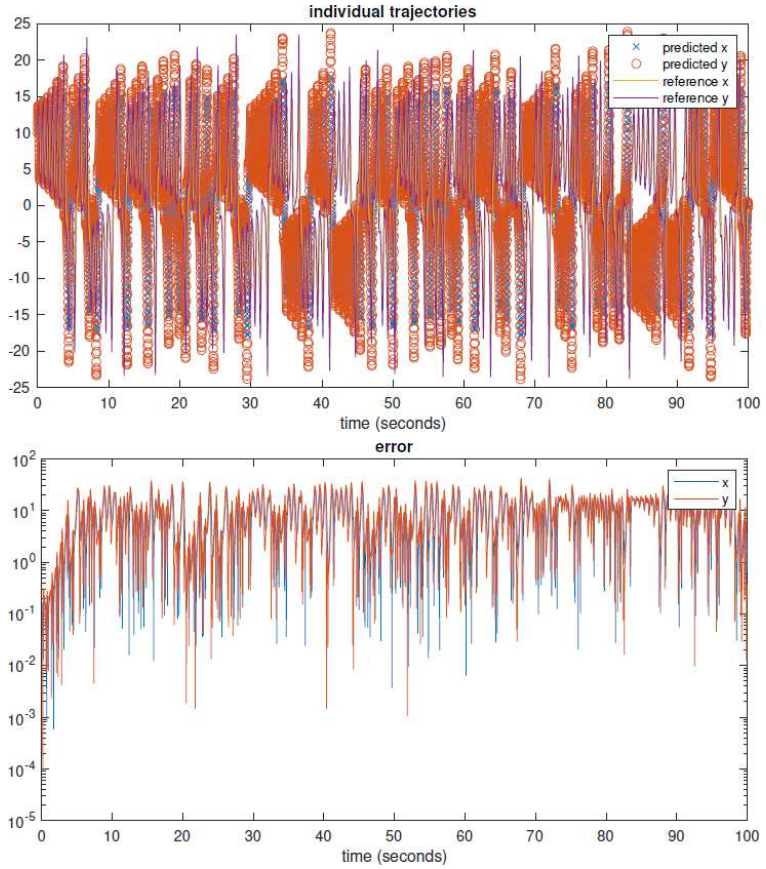

FIG. 6: Ex. 2: Lorenz 63 reduced system of x and y—individual trajectory comparison and pointwise error

statistics with the other two being much more accurate. For example, we see that the reduced system of x and z has a relative error in Lyapunov exponent of 17.5%, but matches correlation dimension and approximate entropy at relative errors of 0.1% and 3.0%.

4.2 High-Dimensional System: Lorenz 96

The Lorenz 96 system (Lorenz, 1996) is an N-dimensional dynamical system given by

$$\frac{dx_i}{dt} = (x_{i+1} - x_{i-2})x_{i-1} + F, \tag{16}$$

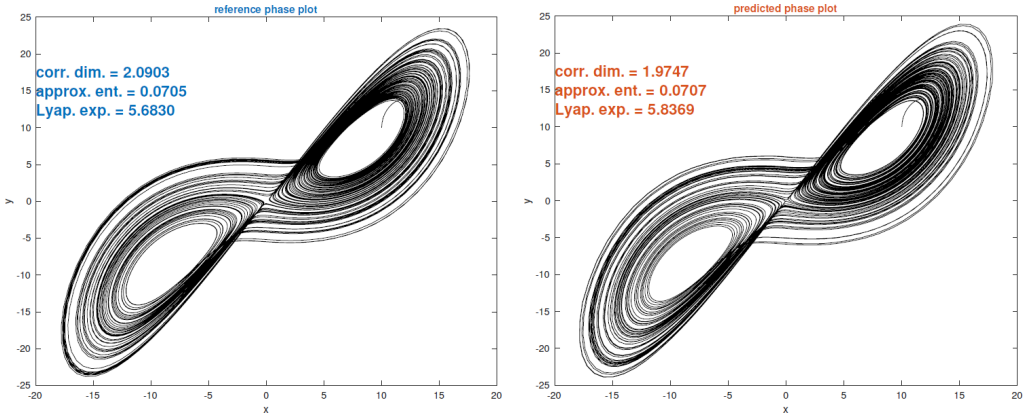

FIG. 7: Ex. 2: Lorenz 63 reduced system of x and y—phase plots (left: reference; right: DNN prediction)

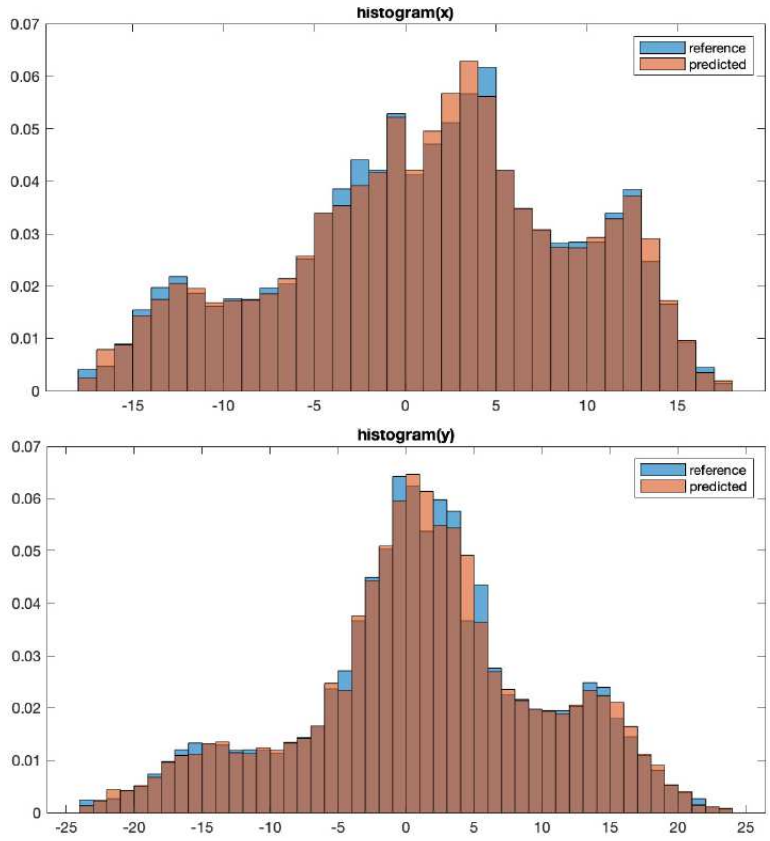

FIG. 8: Ex. 2: Lorenz 63 reduced system of x and y—histogram comparison

for $i = 1, \ldots, N$, for $N \geq 4$ where $x_{-1} = x_{N-1}$, $x_0 = x_N$, $x_{N+1} = x_1$, and F is a forcing parameter. If $F = 8$, the system exhibits chaotic behavior. In the examples below, we choose $N = 40$ as in Lorenz and Emanuel (1998).

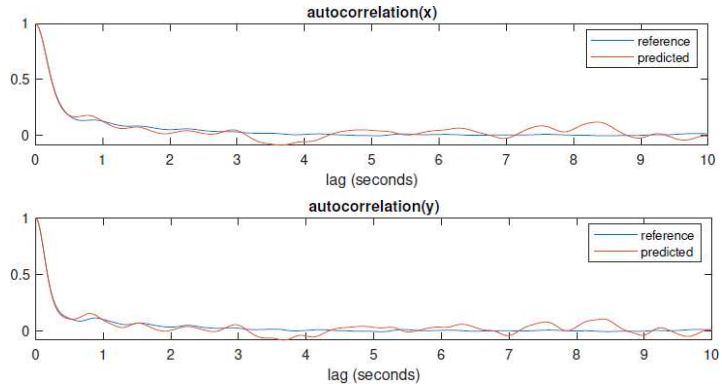

FIG. 9: Ex. 2: Lorenz 63 reduced system of x and y—autocorrelation function comparison

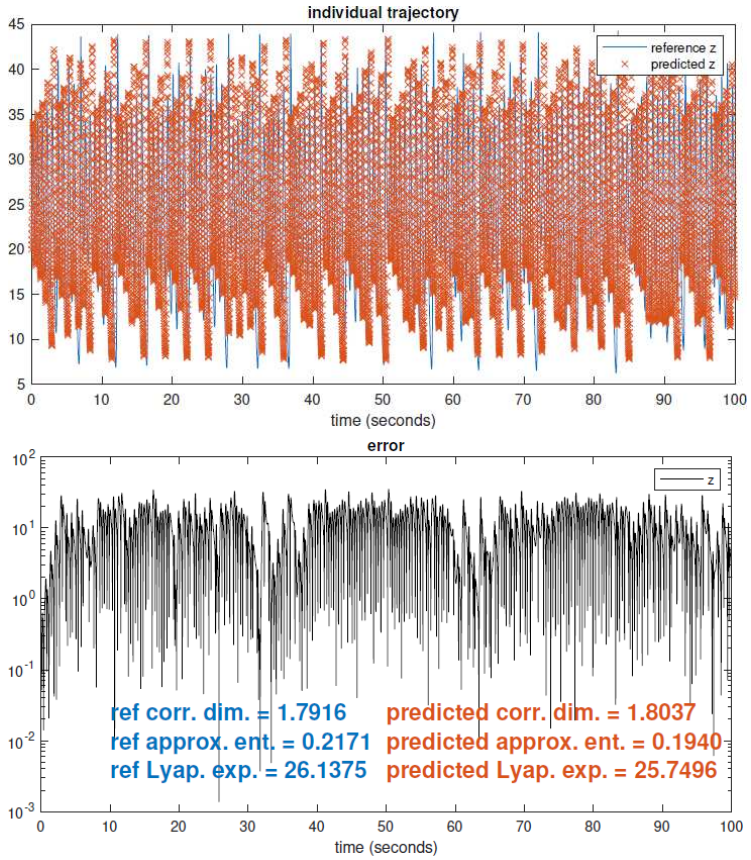

FIG. 10: Ex. 2: Lorenz 63 reduced system of z—individual trajectory and chaos statistics comparison, and pointwise error

4.2.1 Example 3: Full 40-Dimensional System

In this example all 40 state variables are observed. In particular, a trajectory is generated by a high-order numerical solver with $\Delta t = 0.01$ from the initial condition $(8.0081, 8, 8, \ldots, 8)$ until

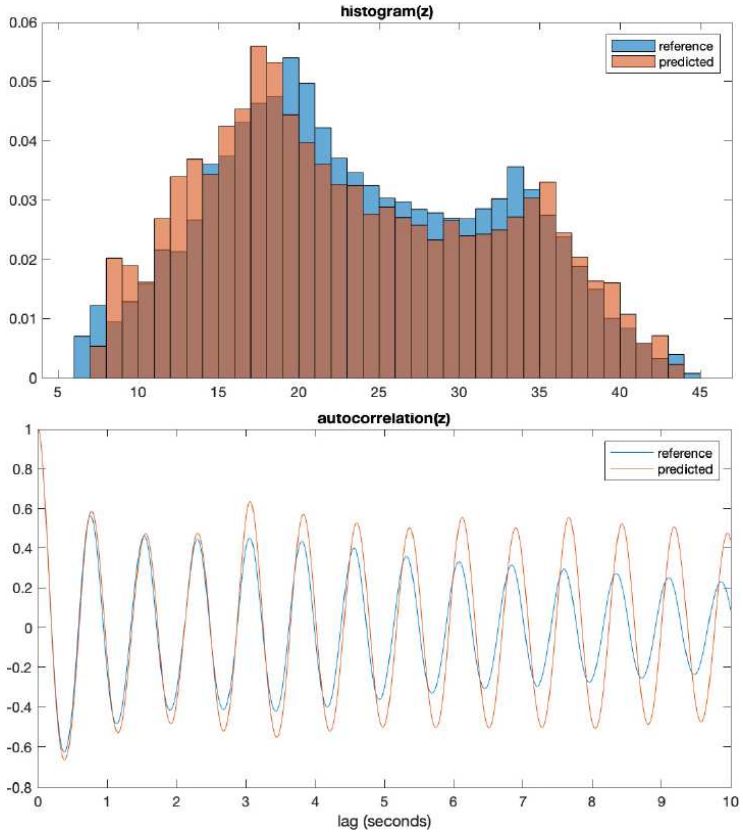

FIG. 11: Ex. 2: Lorenz 63 reduced system of z—histogram and autocorrelation function comparison

$T = 10{,}000$ s (1,000,000 time steps). From this long trajectory, 100,000 chunks of length 0.1 s (11 time steps) are taken as training data. This allows 10 steps to be used for recurrent loss. A standard ResNet with three hidden layers with 200 neurons each is used, and the mean squared recurrent loss function is minimized using Adam with a constant learning rate of 10^{-3} for 2000 epochs.

Prediction is carried out to $T = 500$ (50,000 time steps) from the new initial condition $(8.01, 8, 8, \ldots, 8)$. Figure 12 shows the prediction results for $t = 450$ through $t = 500$ with the ith row corresponding to the trajectory of the variable x_i. We see the same type of qualitative behavior in the reference and prediction, with wavelike forms flowing through the 40 variables. Table 3 shows the correlation dimension, approximate entropy, and Lyapunov exponent for this experiment. As in Ex. 2, we note that while the relative error for approximate entropy is very high at 46.3%, we consider that it may be an anomaly as the other two statistics are significantly more accurate at 3.1% and 3.5%.

4.2.2 Example 4: Reduced Three-Dimensional System

In this example we consider observing only x_1, x_2, and x_3 from the 40-dimensional Lorenz 96 system. Again, Mori–Zwanzig informs us that memory is required to learn the reduced system

TABLE 2: Ex. 2: Lorenz 63 reduced systems—metrics for chaotic behavior comparison

Metrics	Reference Solution	DNN Prediction	Relative Error
	(x, y)		
Correlation dimension	2.0903	1.9747	5.5%
Approximate entropy	0.0705	0.0707	0.3%
Lyapunov exponent	5.6830	5.8369	2.7%
	(x, z)		
Correlation dimension	2.1484	2.1472	0.1%
Approximate entropy	0.0796	0.0820	3.0%
Lyapunov exponent	4.6859	5.5066	17.5%
	(y, z)		
Correlation dimension	2.2618	2.1986	2.8%
Approximate entropy	0.0564	0.0571	1.2%
Lyapunov exponent	3.1263	3.5128	12.4%
	(x)		
Correlation dimension	2.0160	1.8821	6.6%
Approximate entropy	0.1956	0.1986	1.5%
Lyapunov exponent	27.0396	27.3691	1.2%
	(y)		
Correlation dimension	1.7573	1.7979	2.3%
Approximate entropy	0.2053	0.2027	1.3%
Lyapunov exponent	29.1866	29.5374	1.2%
	(z)		
Correlation dimension	1.7916	1.8037	0.7%
Approximate entropy	0.2171	0.1940	10.6%
Lyapunov exponent	26.1375	25.7496	1.5%

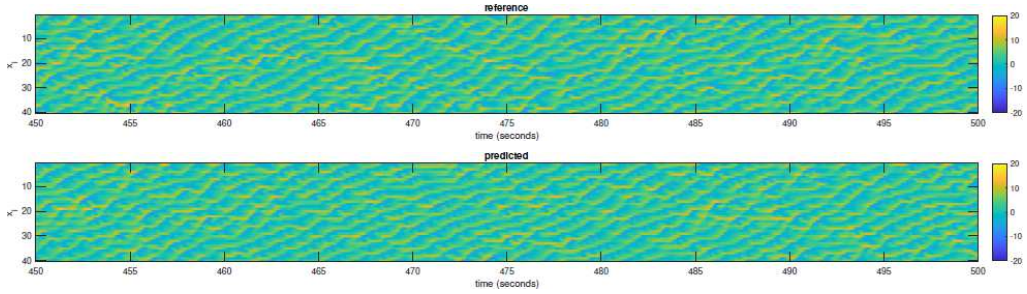

FIG. 12: Ex. 3: Lorenz 96 full system—individual trajectory comparison

dynamics. Hence, a trajectory of the full system is generated by a high-order numerical solver with $\Delta t = 0.01$ from the initial condition $(8.0081, 8, 8, \ldots, 8)$ until $T = 10{,}000$ s. From this long trajectory, 10,000 chunks of length 1.1 s (111 time steps) are taken from only the observed variables as training data. This allows 1 s (100 time steps) for memory and 0.1 s (10 time steps) for recurrent loss. A standard ResNet with ten hidden layers and 20 neurons each is used, and

TABLE 3: Ex. 3: Lorenz 96 full system—metrics for chaotic behavior comparison

Metrics	Reference Solution	DNN Prediction	Relative Error
Correlation dimension	1.4025	1.3585	3.1%
Approximate entropy	0.0067	0.0036	46.3%
Lyapunov exponent	0.2419	0.2506	3.6%

the mean squared recurrent loss function is minimized using Adam with a constant learning rate of 10^{-3} for 10,000 epochs.

Prediction is carried out to $T = 100$ (10,000 time steps) from the new initial condition $(8.01, 8, 8, \ldots, 8)$. Figure 13 shows the individual trajectories between $t = 80$ and $t = 100$ s along with bounded pointwise error, which again indicates the stability of the prediction. Figure 14 shows the phase plots and quantitative measures, while Figs. 15 and 16 show the histogram and autocorrelation function comparisons. Due to the significantly more chaotic and complex nature of the Lorenz 96 system, it is now more difficult to pick out identifying characteristics in the trajectories and phase plots to compare. Nevertheless, the chaos statistics reported in Table 4 indicate a strong match with only one of three statistics exceeding 10% relative error.

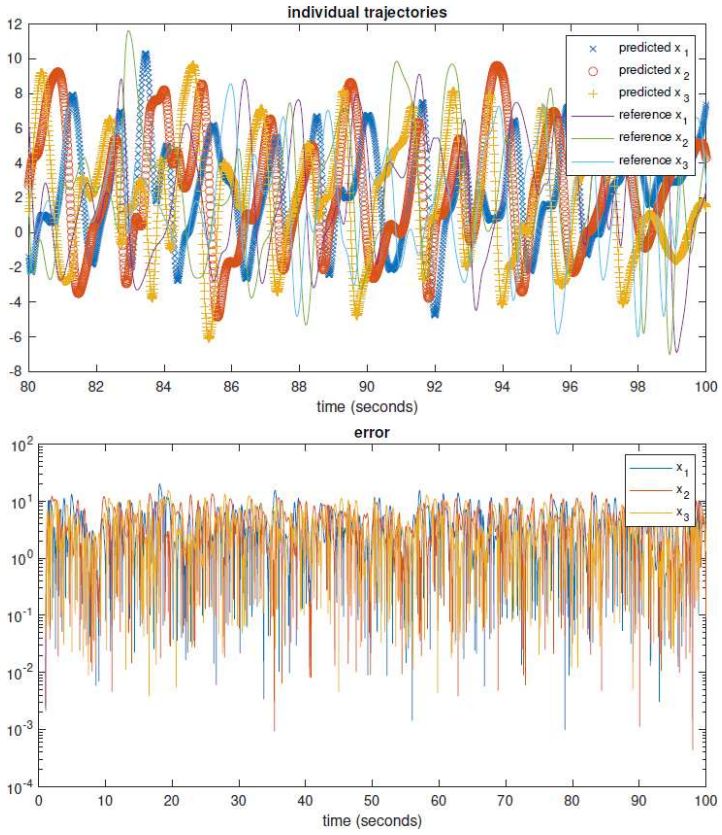

FIG. 13: Ex. 4: Lorenz 96 reduced system of x_1, x_2, x_3—individual trajectory comparison and pointwise error

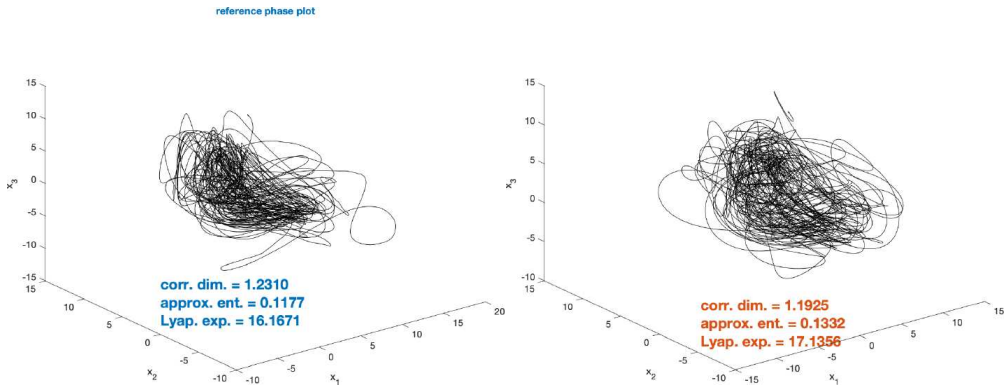

FIG. 14: Ex. 4: Lorenz 96 reduced system of x_1, x_2, x_3—phase plots (left: reference; right: DNN prediction)

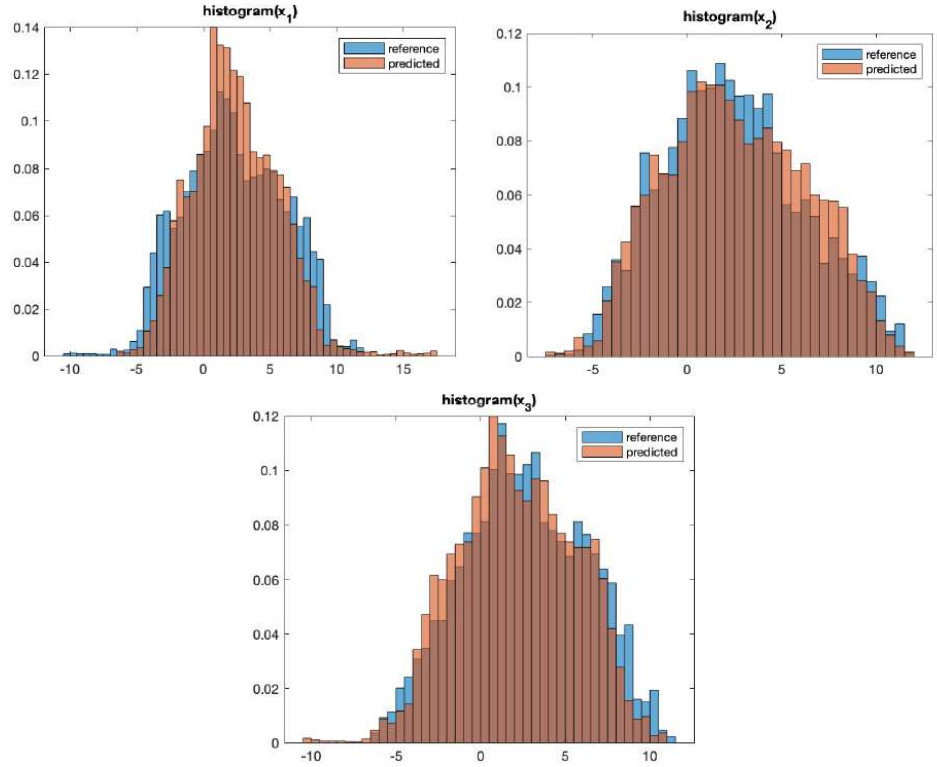

FIG. 15: Ex. 4: Lorenz 96 reduced system of x_1, x_2, x_3—histogram comparison

5. CONCLUSION

We have presented a systematic and rigorous examination of learning the flow maps of fully- and partially-observed chaotic systems using an approachable DNN framework. While in most papers that examine this topic, pointwise error or a visual comparison of phase plots are used to assess the accuracy of network prediction, here we use these tools as well as a variety of

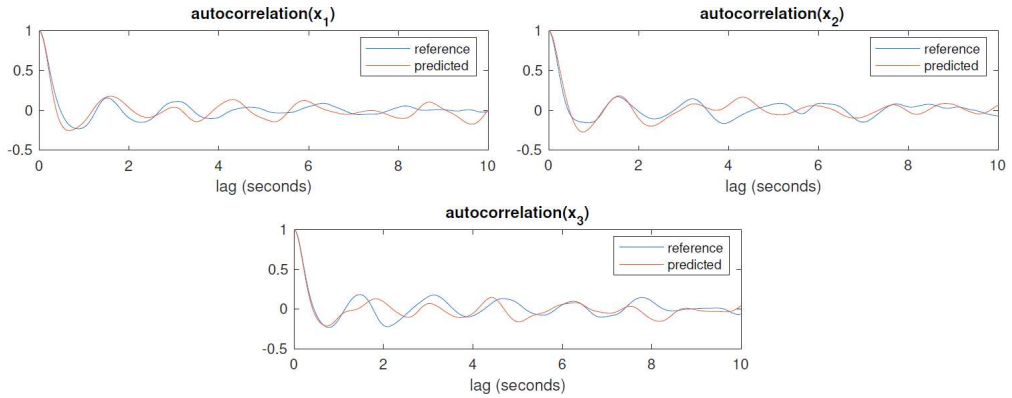

FIG. 16: Ex. 4: Lorenz 96 reduced system of x_1, x_2, x_3—autocorrelation function comparison

TABLE 4: Ex. 4: Lorenz 96 reduced system—metrics for chaotic behavior comparison

Metrics	Reference solution	DNN prediction	Relative error
Correlation dimension	1.2310	1.1925	3.1%
Approximate entropy	0.1177	0.1332	13.1%
Lyapunov exponent	16.1671	17.1356	6.0%

other measures to demonstrate accurate long-term chaotic behavior prediction. Our numerical examples show that this DNN framework is able to learn long-term chaotic behavior even when systems are severely underobserved and training data are collected from a single initial condition.

DATA

The datasets used in this paper are available from the corresponding author upon request.

ACKNOWLEDGMENT

This work was partially supported by AFOSR FA9550-22-1-0011.

REFERENCES

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X., TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, accessed from tensorflow.org, 2015.

Bakarji, J., Champion, K., Kutz, J.N., and Brunton, S.L., Discovering Governing Equations from Partial Measurements with Deep Delay Autoencoders, arXiv: 2201.05136, 2022.

Bakker, R., Schouten, J.C., Giles, C.L., Takens, F., and Van den Bleek, C.M., Learning Chaotic Attractors by Neural Networks, *Neural Comput.*, vol. **12**, no. 10, pp. 2355–2383, 2000.

Bhat, U. and Munch, S.B., Recurrent Neural Networks for Partially Observed Dynamical Systems, *Phys. Rev. E*, vol. **105**, no. 4, p. 044205, 2022.

Box, G.E., Jenkins, G.M., Reinsel, G.C., and Ljung, G.M., *Time Series Analysis: Forecasting and Control*, Hoboken, NJ: John Wiley & Sons, 2015.

Brunton, S.L., Brunton, B.W., Proctor, J.L., Kaiser, E., and Kutz, J.N., Chaos as an Intermittently Forced Linear System, *Nat. Commun.*, vol. **8**, p. 19, 2017.

Brunton, S.L., Proctor, J.L., and Kutz, J.N., Discovering Governing Equations from Data by Sparse Identification of Nonlinear Dynamical Systems, *Proc. Natl. Acad. Sci. USA*, vol. **113**, no. 15, pp. 3932–3937, 2016.

Champion, K., Lusch, B., Kutz, J.N., and Brunton, S.L., Data-Driven Discovery of Coordinates and Governing Equations, *Proc. Natl. Acad. Sci. USA*, vol. **116**, no. 45, pp. 22445–22451, 2019.

Chattopadhyay, A., Hassanzadeh, P., and Subramanian, D., 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 Process. Geophys.*, vol. **27**, no. 3, pp. 373–389, 2020.

Chen, Z., Churchill, V., Wu, K., and Xiu, D., Deep Neural Network Modeling of Unknown Partial Differential Equations in Nodal Space, *J. Comput. Phys.*, vol. **449**, p. 110782, 2022.

Churchill, V., Manns, S., Chen, Z., and Xiu, D., Robust Modeling of Unknown Dynamical Systems via Ensemble Averaged Learning, arXiv: 2203.03458, 2022.

Dubois, P., Gomez, T., Planckaert, L., and Perret, L., Data-Driven Predictions of the Lorenz System, *Phys. D: Nonlinear Phenom.*, vol. **408**, p. 132495, 2020.

Fu, X., Chang, L.-B., and Xiu, D., Learning Reduced Systems via Deep Neural Networks with Memory, *J. Mach. Learn. Model. Comput.*, vol. **1**, no. 2, pp. 97–118, 2020.

Han, M., Shi, Z., and Wang, W., Modeling Dynamic System by Recurrent Neural Network with State Variables, *Int. Symposium Neural Networks*, Dalian, China, pp. 200–205, 2004a.

Han, M., Xi, J., Xu, S., and Yin, F.-L., Prediction of Chaotic Time Series Based on the Recurrent Predictor Neural Network, *IEEE Trans. Signal Process.*, vol. **52**, no. 12, pp. 3409–3416, 2004b.

He, K., Zhang, X., Ren, S., and Sun, J., Deep Residual Learning for Image Recognition, in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, Las Vegas, NV, pp. 770–778, 2016.

Kang, S.H., Liao, W., and Liu, Y., IDENT: Identifying Differential Equations with Numerical Time Evolution, arXiv: 1904.03538, 2019.

Kim, H., Eykholt, R., and Salas, J., Nonlinear Dynamics, Delay Times, and Embedding Windows, *Phys. D: Nonlinear Phenom.*, vol. **127**, nos. 1-2, pp. 48–60, 1999.

Kingma, D.P. and Ba, J., Adam: A Method for Stochastic Optimization, arXiv: 1412.6980, 2014.

Long, Z., Lu, Y., and Dong, B., PDE-Net 2.0: Learning PDEs from Data with a Numeric-Symbolic Hybrid Deep Network, arXiv: 1812.044267, 2018a.

Long, Z., Lu, Y., Ma, X., and Dong, B., PDE-Net: Learning PDEs from Data, in *Proc. of the 35th Int. Conf. on Machine Learning*, Stockholm, Sweden, pp. 3208–3216, 2018b.

Lorenz, E.N., Deterministic Nonperiodic Flow, *J. Atmos. Sci.*, vol. **20**, no. 2, pp. 130–141, 1963.

Lorenz, E.N., Predictability: A Problem Partly Solved, in *Proc. of Seminar on Predictability*, Shinfield Park, Reading, UK, 1996.

Lorenz, E.N. and Emanuel, K.A., Optimal Sites for Supplementary Weather Observations: Simulation with a Small Model, *J. Atmos. Sci.*, vol. **55**, no. 3, pp. 399–414, 1998.

Lu, L., Jin, P., Pang, G., Zhang, Z., and Karniadakis, G.E., Learning Nonlinear Operators via DeepONet Based on the Universal Approximation Theorem of Operators, *Nat. Mach. Intell.*, vol. **3**, no. 3, pp. 218–229, 2021a.

Lu, L., Meng, X., Mao, Z., and Karniadakis, G.E., DeepXDE: A Deep Learning Library for Solving Differential Equations, *SIAM Rev.*, vol. **63**, no. 1, pp. 208–228, 2021b.

Lusch, B., Kutz, J.N., and Brunton, S.L., Deep Learning for Universal Linear Embeddings of Nonlinear Dynamics, *Nat. Commun.*, vol. **9**, no. 1, pp. 1–10, 2018.

MATLAB, R2022a, The MathWorks Inc., Natick, MA, 2022.

Miyoshi, T., Ichihashi, H., Okamoto, S., and Hayakawa, T., Learning Chaotic Dynamics in Recurrent RBF Network, in *Proc. of ICNN'95-Int. Conf. on Neural Networks*, Perth, Australia, pp. 588–593, 1995.

Mori, H., Transport, Collective Motion, and Brownian Motion, *Prog. Theor. Phys.*, vol. **33**, no. 3, pp. 423–455, 1965.

Pan, S. and Duraisamy, K., Data-Driven Discovery of Closure Models, *SIAM J. Appl. Dyn. Syst.*, vol. **17**, no. 4, pp. 2381–2413, 2018.

Pawar, S., San, O., Rasheed, A., and Navon, I.M., A Nonintrusive Hybrid Neural-Physics Modeling of Incomplete Dynamical Systems: Lorenz Equations, *GEM – Int. J. Geomath.*, vol. **12**, no. 1, pp. 1–31, 2021.

Pincus, S.M., Approximate Entropy as a Measure of System Complexity. *Proc. Natl. Acad. Sci.*, vol. **88**, no. 6, pp. 2297–2301, 1991.

Qin, T., Wu, K., and Xiu, D., Data Driven Governing Equations Approximation Using Deep Neural Networks, *J. Comput. Phys.*, vol. **395**, pp. 620–635, 2019.

Qin, T., Chen, Z., Jakeman, J., and Xiu, D., Data-Driven Learning of Non-Autonomous Systems, *SIAM J. Sci. Comput.*, vol. **43**, no. 3, pp. A1607–A1624, 2021a.

Qin, T., Chen, Z., Jakeman, J., and Xiu, D., Deep Learning of Parameterized Equations with Applications to Uncertainty Quantification, *Int. J. Uncertainty Quantif.*, vol. **11**, no. 2, pp. 63–82, 2021b.

Raissi, M., Deep Hidden Physics Models: Deep Learning of Nonlinear Partial Differential Equations, *J. Mach. Learn. Res.*, vol. **19**, no. 25, pp. 1–24, 2018.

Raissi, M., Perdikaris, P., and Karniadakis, G.E., Physics Informed Deep Learning (Part I): Data-Driven Solutions of Nonlinear Partial Differential Equations, arXiv: 1711.10561, 2017a.

Raissi, M., Perdikaris, P., and Karniadakis, G.E., Physics Informed Deep Learning (Part II): Data-Driven Discovery of Nonlinear Partial Differential Equations, arXiv: 1711.10566, 2017b.

Raissi, M., Perdikaris, P., and Karniadakis, G.E., Multistep Neural Networks for Data-Driven Discovery of Nonlinear Dynamical Systems, arXiv: 1801.01236, 2018.

Rosenstein, M.T., Collins, J.J., and De Luca, C.J., A Practical Method for Calculating Largest Lyapunov Exponents from Small Data Sets, *Phys. D: Nonlinear Phenom.*, vol. **65**, nos. 1-2, pp. 117–134, 1993.

Rudy, S.H., Brunton, S.L., Proctor, J.L., and Kutz, J.N., Data-Driven Discovery of Partial Differential Equations, *Sci. Adv.*, vol. **3**, no. 4, p. e1602614, 2017.

Rudy, S.H., Kutz, J.N., and Brunton, S.L., Deep Learning of Dynamics and Signal-Noise Decomposition with Time-Stepping Constraints, *J. Comput. Phys.*, vol. **396**, pp. 483–506, 2019.

Schaeffer, H., Learning Partial Differential Equations via Data Discovery and Sparse Optimization, *Proc. R. Soc. London A: Math. Phys. Eng. Sci.*, vol. **473**, p. 2197, 2017.

Schaeffer, H. and McCalla, S.G., Sparse Model Selection via Integral Terms, *Phys. Rev. E*, vol. **96**, no. 2, p. 023302, 2017.

Schaeffer, H., Tran, G., and Ward, R., Extracting Sparse High-Dimensional Dynamics from Limited Data, *SIAM J. Appl. Math.*, vol. **78**, no. 6, pp. 3279–3295, 2018.

Scher, S. and Messori, G., Generalization Properties of Feed-Forward Neural Networks Trained on Lorenz Systems, *Nonlinear Process. Geophys.*, vol. **26**, no. 4, pp. 381–399, 2019.

Sun, Y., Zhang, L., and Schaeffer, H., NeuPDE: Neural Network Based Ordinary and Partial Differential Equations for Modeling Time-Dependent Data, arXiv: 1908.03190, 2019.

Takens, F., Detecting Strange Attractors in Turbulence, in *Dynamical Systems and Turbulence, Warwick 1980*, pp. 366–381, Berlin: Springer, 1981.

The MathWorks, Econometrics Toolbox, Natick, MA, 2022a.

The MathWorks, Predictive Maintenance Toolbox, Natick, MA, 2022b.

Theiler, J., Efficient Algorithm for Estimating the Correlation Dimension from a Set of Discrete Points, *Phys. Rev. A*, vol. **36**, no. 9, p. 4456, 1987.

Tran, G. and Ward, R., Exact Recovery of Chaotic Systems from Highly Corrupted Data, *Multiscale Model. Simul.*, vol. **15**, no. 3, pp. 1108–1129, 2017.

Trischler, A.P. and D'Eleuterio, G.M., Synthesis of Recurrent Neural Networks for Dynamical System Simulation, *Neural Networks*, vol. **80**, pp. 67–78, 2016.

Vlachas, P.R., Byeon, W., Wan, Z.Y., Sapsis, T.P., and Koumoutsakos, P., Data-Driven Forecasting of High-Dimensional Chaotic Systems with Long Short-Term Memory Networks, *Proc. R. Soc. A: Math. Phys. Eng. Sci.*, vol. **474**, no. 2213, p. 20170844, 2018.

Wang, Q., Ripamonti, N., and Hesthaven, J., Recurrent Neural Network Closure of Parametric POD-Galerkin Reduced-Order Models Based on the Mori–Zwanzig Formalism, *J. Comput. Phys.*, vol. **410**, p. 109402, 2020.

Wu, K. and Xiu, D., Data-Driven Deep Learning of Partial Differential Equations in Modal Space, *J. Comput. Phys.*, vol. **408**, p. 109307, 2020.

Wulkow, N., Koltai, P., Sunkara, V., and Schütte, C., Data-Driven Modelling of Nonlinear Dynamics by Barycentric Coordinates and Memory, arXiv: 2112.06742, 2021.

Zimmermann, H.G. and Neuneier, R., Modeling Dynamical Systems by Recurrent Neural Networks, *WIT Trans. Inf. Commun. Technol.*, vol. **25**, 2000.

Zwanzig, R., Nonlinear Generalized Langevin Equations, *J. Stat. Phys.*, vol. **9**, no. 3, pp. 215–220, 1973.