Referee #1

This is a very interesting paper that is certainly worthy of publication. I have attached a copy of the paper with my comments that were added by Adobe Acrobat. While I have a large number of comments, most of them are editorial or typographical. The authors should be able to address these easily. A few of my comments are more substantive and I provide a complete list of those below. Addressing any of the comments that fall into this category should not require the authors to carry out new numerical simulations or to do a major revision of the paper. I recommend that the paper be accepted subject to a minor revision

We thank the referee for his/her thoughtful comments and for the positive assessment of our paper. We have addressed the referee's comments and suggestion that were provided on the pdf and all the changes have been tracked in the revised manuscript in blue (attached to the end of this pdf).

Below, we provide point-by-point responses to the comments that the referee included in the summary.

Referee: Title and at many places elsewhere: Reservoir computing is not deep learning. I suggest using the term 'machine learning' rather than 'deep learning' in most places in the paper, including the title.

Authors: We have replaced deep learning with "machine learning" in the title and in other places in the paper.

Referee: Equation 5: This equation and the related discussion, including Appendix A, should be fixed.

Authors: The equation has been changed in the main text. In Appendix A, although the formulation has been kept the same, we identify $\boldsymbol{r}$ and $\tilde{\boldsymbol{r}}$ as matrices so that the notation is easier to read.

Referee: Equation 6 and 7: Introducing the dependent variable v is unnecessary. The two equations should be combined into a single one.

Authors: Fixed.

Referee: Line 170: You cannot use the same notation A for both the connectivity matrix and its normalized form.

Authors: Thanks for the comment. The notation has been changed. The normalized version of $\boldsymbol{A}$ has been designated the symbol $\widetilde{\boldsymbol{A}}$.

Referee: Equation 8 and text above. You cannot use the notation X(t+\Delta t) for two different quantities.

Authors: Thanks for bringing this issue to our attention. The notation has been changed. During training we put a super-script of "train" where we have evaluated $\Delta X(t)^{train} = X(t+\Delta t)^{train} - X(t)^{train}$ and during testing we use evaluated $\Delta X(t)^{test}$ to march forward in time with Adams-Bashforth method to calculate $X(t+\Delta t)^{test}$ from $X(t)^{test}$.

Apart from these comments, the referee suggested that we add legends to the figures if possible. We found it difficult to add legible legends while avoiding overlaps with the contents of the figure. Thus, in the figures, we have made sure that the captions clearly describe what different lines/symbols refer to.

Again, we thank the reviewer for the careful read of our paper and for insightful and constructive comments.

Referee #2

The manuscript presents a comparison of Recurrent neural networks with Long ShortTerm Memory (LSTM), Reservoir Computers termed Echo State Networks (ESNs) and deep multi-layered Artificial Neural Networks (ANNs) on forecasting the evolution of the slow dynamics (X) of a Lorenz-96 multi-scale problem, while two other states with faster dynamics cannot be observed (i.e. assumed to be unknown) both during training and testing. According to the findings of the study, the ESN shows a remarkable performance surpassing both ANN and LSTM by a large margin.

We thank the referee for his/her thoughtful comments. We have addressed the referee's comments by revising the text and all the changes have been tracked in the revised manuscript in blue (attached to the end of this pdf).

Below, we provide point-by-point responses to the referee's comments

Referee: The authors are augmenting the dynamics of the reservoir of the ESN based on the dynamics of the equation used to generate the data. As reported in the appendix, the equation agnostic ESN diverges. In order to cope with this divergence, the authors augmented the ESN reservoir with the grid dynamics of the equations (refer to Algorithm T2 and T3 in the Appendix and compare to equation (1) of the main text). This augmentation seems to be very heuristic and is driven by the form of the equations used to generate the data in this application and is by no means standard in Reservoir Computing (ESNs).

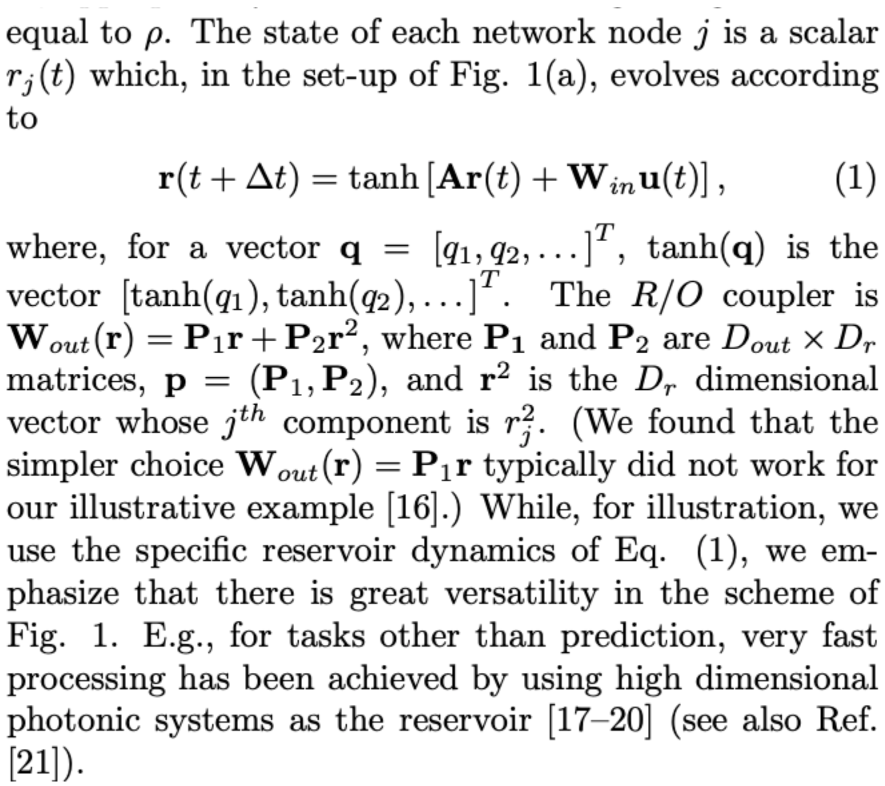
As a consequence, the overall approach utilized to generate the ESN predictions cannot be considered equation-agnostic and as such the comparison with ANNs and LSTMs is not fair. The same augmentation can be easily extended to LSTM and ANN. The authors are not emphasizing this augmentation in the main document, even though this seems to be the main reason for the success of the ESN approach.

Authors: We thank the reviewer for raising two points here: 1) That the state augmentation is not standard in ESN, 2) That the same augmentation could be used for ANNs and LSTMs and thus our comparison is not fair. Below we discuss that in fact such state augmentation is commonly used in recent studies that apply ESN to complex chaotic dynamical systems, and that such augmentation cannot be used for ANNs (because they do not have state) and is challenging to use for LSTMs (because they have more than one state). Still, we have done further tests in which we performed basis function expansion on the inputs for both ANN and LSTM but did not find improved predictions (see below).

The state augmentation we have used here is following what has been used in other papers. The state of the reservoir in ESN is augmented by introducing non-linearity, based on either algorithms T1, T2, or T3. Despite this augmentation being non-standard to traditional ESN literature (e.g. Jaeger and Haas 2004 Science), it has been used in previous recent studies that employed ESN to predict complex chaotic dynamical systems: e.g., Pathak et al. (Physical Review Letters, 2018) used ESN with its states augmented in a similar fashion as T1 following the non-linearity of the

system (i.e., quadratic non-linearity in Kuramoto–Sivashinsky equation). The state augmentation is clearly mentioned in the aforementioned paper's main text (in Page 2, column 1, Line 50). A screen grab from the paper has been shown below Figure 1 for your convenience. Moreover, Eq. S1 in the supplementary material of this reference further illustrates the state augmentation algorithm
https://journals.aps.org/prl/supplemental/10.1103/PhysRevLett.120.024102/aps_supplement.pdf.

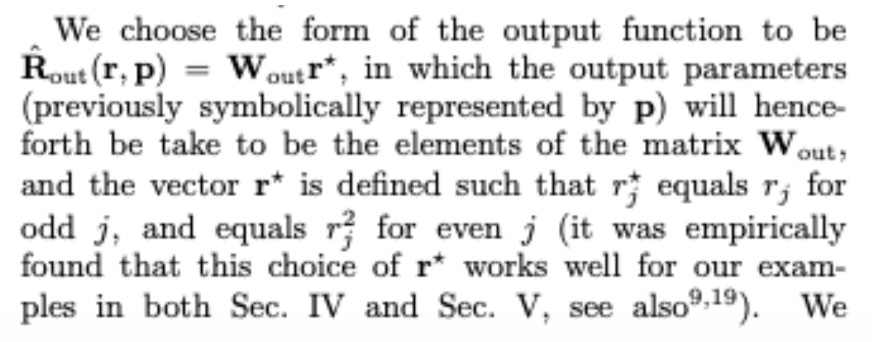
Another paper by Pathak et al. (Chaos, 2017) used a non-linear transformation as well to incorporate quadratic non-linearity in the objective function that is minimized (Eq.3 of the main text on Page 3). Similarly, another recent paper by Pathak et al., (Chaos, 2018) uses the non-linear transformation of the state following algorithm T1. The information is provided on Page 4, column 1, Line 50 of their paper. A screen grab from their paper is also provided in Figure 2 below for your convenience. The authors further mention (also present in the screen grab) that this transformation is empirical and works well for their application.


equal to ρ. The state of each network node j is a scalar $r_j(t)$ which, in the set-up of Fig. 1(a), evolves according to

$$\mathbf{r}(t+\Delta t) = \tanh\left[\mathbf{A}\mathbf{r}(t) + \mathbf{W}_{in}\mathbf{u}(t)\right], \qquad (1)$$

where, for a vector $\mathbf{q} = [q_1, q_2, \dots]^T$, $\tanh(\mathbf{q})$ is the vector $[\tanh(q_1), \tanh(q_2), \dots]^T$. The R/O coupler is $\mathbf{W}_{out}(\mathbf{r}) = \mathbf{P}_1\mathbf{r} + \mathbf{P}_2\mathbf{r}^2$, where $\mathbf{P_1}$ and $\mathbf{P}_2$ are $D_{out} \times D_r$ matrices, $\mathbf{p} = (\mathbf{P}_1, \mathbf{P}_2)$, and $\mathbf{r}^2$ is the D_r dimensional vector whose j^{th} component is r_j^2. (We found that the simpler choice $\mathbf{W}_{out}(\mathbf{r}) = \mathbf{P}_1\mathbf{r}$ typically did not work for our illustrative example [16].) While, for illustration, we use the specific reservoir dynamics of Eq. (1), we emphasize that there is great versatility in the scheme of Fig. 1. E.g., for tasks other than prediction, very fast processing has been achieved by using high dimensional photonic systems as the reservoir [17–20] (see also Ref. [21]).

Figure 1. A screen grab from Pathak et al. (Physical Review Letters, 2018) where the state augmentation introducing quadratic non-linearity is mentioned in the main text.



We choose the form of the output function to be $\hat{\mathbf{R}}_{\text{out}}(\mathbf{r},\mathbf{p}) = \mathbf{W}_{\text{out}}\mathbf{r}^\star$, in which the output parameters (previously symbolically represented by $\mathbf{p}$) will henceforth be take to be the elements of the matrix $\mathbf{W}_{\text{out}}$, and the vector $\mathbf{r}^\star$ is defined such that $r_j^\star$ equals r_j for odd j, and equals r_j^2 for even j (it was empirically found that this choice of $\mathbf{r}^\star$ works well for our examples in both Sec. IV and Sec. V, see also[9,19]). We

Figure 2. A screen grab from Pathak et al. (Chaos, 2018) where the empirical state augmentation introducing non-linearity is mentioned in the main text.

Furthermore, while a non-linear transformation of the state is essential for accurate prediction, the exact form of the non-linear transformation does not substantially affect the performance. In our model, the form of non-linearity introduced in the state augmentation algorithm affects the performance of the ESN only by a small margin, e.g., the prediction horizon of the ESN using T1 (where knowledge of the exact structure of the system's equations is not needed) is only behind T2 and T3 by $5\Delta t$.

We have explicitly mentioned in the main text of the paper that we have augmented the states (Lines 150-155 in section 2.2.1 under RC-ESN architecture) and have further provided details in Appendix A. We have clearly emphasized that this procedure is *essential* for having skillful predictions for ESN (Line 155).

For ANN, such an intuitive state transformation is not possible, since the method is stateless. However, basis function expansion can be performed. We have performed basis function expansion using both $X(t)$ and $X^2(t)$ as input to predict $\Delta X(t)$ and then calculated $X(t+\Delta t)$ using Adams-Bashforth method, as explained in the paper. However, we found no improvement or less accurate predictions. Below, in Table 1, we present the results comparing the ANN results over different training sample sizes with and without basis function expansion. Other preliminary basis function expansions (e.g. using other non-linear functions of $X(t)$) have been performed but none of them yielded results that were better than the ones reported in the paper.

Table 1. Prediction horizon in terms of MTU (defined in the paper) for two types of ANN, one with and one without basis function expansion for different training sample sizes (N).

Type of ANN	N=10K samples	N=100K samples	N=500K samples	N=1M samples	N=2M samples
With basis function expansion	0.18 MTU	0.20 MTU	0.28 MTU	0.30 MTU	0.30 MTU
Without basis function expansion	0.20 MTU	0.24 MTU	0.28 MTU	0.30 MTU	0.30 MTU

For an RNN-LSTM, there are two states, $h(t)$ and $C(t)$ as described in equations C1-C8 in Appendix C of the paper. An intuitive mapping between these states and the input space is not possible (it is straightforward in an ESN which has only one state), since there are multiple layers in an RNN-LSTM, and hence state augmentation is not a common practice for RNN-LSTMs even in applications to complex chaotic dynamical systems (Vlachas et al., Proceedings of Royal Society A, 2018). RNN-LSTM is a sophisticated algorithm where each layers' weights are trained and have been used in multiple different studies that deal with chaotic dynamical systems (Vlachas et al., Proceedings of Royal Society A**,** 2018; Wang et al., Nonlinear Dynamics, 2019; Yu et al., arXiv:1711.00073, 2017). However, we conducted basis function expansion in the input space (as in the case of ANN) and the results did not improve from the results reported in the paper and had thus been not shown for brevity.

Our aim, as clearly stated in Lines 55-60 in the Introduction section of the paper, was to compare the performance of most commonly used versions of ESN, ANN, and LSTM for predicting complex chaotic system. We have been careful in stating the implications and potential caveats of our findings (Lines 330-335 in the Discussion section). We have further discussed (Lines 360-365 in the Discussion section) variants of these architectures that have been used in recent studies and the key steps that made the reported architectures in our paper improve their performance (Line 357 in the Discussion section). Also, please note that we have provided all architectural configurations of our methods in the text and in the codes, which are openly hosted on GitHub. We hope that our paper would serve as a reference or a baseline for newer algorithms to be developed in the future for performing data-driven prediction in spatio-temporal chaotic systems.

Referee: No information is provided about the value of ρ in Fig 5. This is an important parameter and needs to be discussed

Authors: We thank the referee for pointing out this issue. We have used the value of $\rho = 0.1$ in Figure 5. We found this value of ρ to yield the best results (the value is now added to the captions in Figure 5 and Figure 6). In general, one needs $\rho \leq 1$ for the ESN to be stable (Yildiz et al., Neural Networks, 2012; Venayagamoorthy et al., Neural Networks, 2009). Beyond the requirement of $\rho \leq 1$, there is no theoretical understanding of the optimal ρ for a given system/data. Previous studies on using ESN for chaotic dynamical system have used a range of ρ: $\rho = 0.6$ (Pathak et al., Physical Review Letters, 2018), $\rho = 0.9$ (Lu et al., Chaos), and $\rho = 0.4$ (Pathak et al., Chaos, 2018). Each of these values of ρ were obtained by trial and error and the sensitivity of the prediction horizon on ρ is highly dependent on the underlying chaotic dynamics.

We have conducted an extensive experimental analysis on the dependence of the prediction horizon on the value of ρ, and found $\rho = 0.1$ to produce the best prediction horizon for any sample size N and reservoir size D for <u>this system</u>. We did not report the full analysis for brevity. Below, in Table 2, we show part of those results for $N = 1M$ and $N = 500K$ and $D = 5000$ over 100 initial conditions. While $\rho = 0.1$ yields the best results, we see that the prediction horizon is barely sensitive to the choice of ρ within a wide range and similar overall prediction horizons are obtained for $0.05 \leq \rho \leq 0.7$.

Table 2: Prediction horizon (MTU) over 100 initial conditions for an ESN with samples size of $N = 1M$ and $N = 500K$ and $D = 5000$ for different ranges of ρ.

ρ	0.01-0.04	0.05-0.09	0.1	0.2	0.3	0.4	0.40-0.70	0.70-0.90
Prediction horizon (MTU) at $N = 1M$	1.01	1.18	1.30	1.28	1.26	1.20	1.19	0.83
Prediction horizon (MTU) at $N = 500K$	0.84	0.92	1.10	1.02	1.02	1.02	1.02	0.77

Data-driven prediction of a multi-scale Lorenz 96 chaotic system using machine learning methods: Reservoir computing, artificial neural network, and long short-term memory network

Ashesh Chattopadhyay[1], Pedram Hassanzadeh[1,2], and Devika Subramanian[3,4]

[1]Department of Mechanical Engineering, Rice University
[2]Department of Earth Environmental and Planetary Sciences, Rice University
[3]Department of Electrical and Computer Engineering, Rice University
[4]Department of Computer Science, Rice University

Correspondence: Pedram Hassanzadeh (pedram@rice.edu)

Abstract. In this paper, the performance of three machine learning methods for predicting short-term evolution and for reproducing the long-term statistics of a multi-scale spatio-temporal Lorenz 96 system is examined. The methods are: echo state network (a type of reservoir computing, RC-ESN), deep feed-forward artificial neural network (ANN), and recurrent neural network with long short-term memory (RNN-LSTM). This Lorenz 96 system has three tiers of nonlinearly interacting variables representing slow/large-scale (X), intermediate (Y), and fast/small-scale (Z) processes. For training or testing, only X is available; Y and Z are never known or used. We show that RC-ESN substantially outperforms ANN and RNN-LSTM for short-term prediction, e.g., accurately forecasting the chaotic trajectories for hundreds of numerical solver's time steps, equivalent to several Lyapunov timescales. The RNN-LSTM and ANN show some prediction skills as well; with RNN-LSTM outperforming ANN. Furthermore, even after losing the trajectory, data predicted by RC-ESN and RNN-LSTM have probability density functions (PDFs) that closely match the true PDF, even at the tails. The PDF of the data predicted using ANN, however, deviates from the true PDF. Implications, caveats, and applications to data-driven and data-assisted surrogate modeling of complex nonlinear dynamical systems such as weather/climate are discussed.

1 Introduction

Various components of the Earth system involve multi-scale, multi-physics processes and high-dimensional chaotic dynamics. These processes are often modeled using sets of strongly-coupled nonlinear partial differential equations (PDEs), which are solved numerically on supercomputers. As we aim to simulate such systems with increasing levels of fidelity, we need to increase the numerical resolutions and/or incorporate more physical processes from a wide range of spatio-temporal scales into the models. For example, in atmospheric modeling for predicting the weather and climate systems, we need to account for the nonlinear interactions across the scales of cloud microphysics processes, gravity waves, convection, baroclinic waves, synoptic eddies, and large-scale circulation, just to name a few, not to mention the fast/slow processes involved in feedbacks from the ocean, land, and cryosphere (Collins et al., 2006, 2011; Flato, 2011; Bauer et al., 2015; Jeevanjee et al., 2017).

Solving the coupled systems of PDEs representing such multi-scale processes is computationally highly challenging for many practical applications. As a result, to make the simulations feasible, a few strategies have been developed, which mainly involve only solving for slow/large-scale variables and accounting for the fast/small-scale processes through surrogate models. In weather/climate models, for example, semi-empirical physics-based parameterizations are often used to represent the effects of processes such as gravity waves and moist convection in the atmosphere or sub-mesoscale eddies in the ocean (Stevens and Bony, 2013; Hourdin et al., 2017; Garcia et al., 2017; Jeevanjee et al., 2017; Schneider et al., 2017b; Chattopadhyay et al., 2020b, a). A more advanced approach is "super-parameterization", which for example, involves solving the PDEs of moist convection on a high-resolution grid inside each grid point of large-scale atmospheric circulation, whose governing PDEs (the Navier-Stokes equations) are solved on a coarse grid (Khairoutdinov and Randall, 2001). While computationally more expensive, super-parameterized climate models have been shown to outperform parameterized models in simulating some aspects of climate variability and extremes (Benedict and Randall, 2009; Andersen and Kuang, 2012; Kooperman et al., 2018).

More recently, "inexact computing" has received attention from the weather/climate community. This approach involves reducing the computational cost of each simulation by decreasing the *precision* of some of the less-critical calculations (Palem, 2014; Palmer, 2014), thus allowing the saved resources to be *reinvested*, for example, in more simulations (e.g., for probabilistic predictions) and/or higher resolutions for critical processes (Düben et al., 2014; Düben and Palmer, 2014; Thornes et al., 2017; Hatfield et al., 2018). One type of inexact computing involves solving the PDEs of some of the processes with single- or half-precision arithmetic, which requires less computing power and memory compared to the conventional double-precision calculations (Palem, 2014; Düben et al., 2015; Leyffer et al., 2016).

In the past few years, the rapid algorithmic advances in machine learning (ML), and in particular data-driven modeling, have been explored for improving the simulation and prediction of nonlinear dynamical systems (Schneider et al., 2017a; Kutz, 2017; Gentine et al., 2018; Moosavi et al., 2018; Wu et al., 2019; Toms et al., 2019; Brunton and Kutz, 2019; Duraisamy et al., 2019; Reichstein et al., 2019; Lim et al., 2019; Scher and Messori, 2019; Chattopadhyay et al., 2020c). One appeal of the data-driven approach is that fast/accurate data-driven surrogate models that are trained on data of high-fidelity, can be used to accelerate and/or improve the prediction and simulation of complex dynamical systems. Furthermore, for some poorly understood processes for which observational data are available (e.g., clouds), data-driven surrogate models built using such data might potentially outperform physics-based surrogate models (Schneider et al., 2017a; Reichstein et al., 2019). Recent studies have shown promising results in using ML to build data-driven parameterizations for modeling of some atmospheric and oceanic processes (Rasp et al., 2018; Brenowitz and Bretherton, 2018; Gagne et al., 2020; O'Gorman and Dwyer, 2018; Bolton and Zanna, 2019; Düeben and Bauer, 2018; Watson, 2019; Salehipour and Peltier, 2019). In the turbulence and dynamical systems communities, similarly encouraging outcomes have been reported (Ling et al., 2016; McDermott and Wikle, 2017; Pathak et al., 2018a; Rudy et al., 2018; Vlachas et al., 2018; Mohan et al., 2019; Wu et al., 2019; Raissi et al., 2019; Zhu et al., 2019; McDermott and Wikle, 2019) .

The objective of the current study is to make progress toward addressing the following question: Which AI-based data-driven technique(s) can best predict the spatio-temporal evolution of a multi-scale chaotic system, when only the slow/large-scale variables are known (during training) and are of interest (to predict during testing)? We emphasize that unlike many other studies, our focus is not on reproducing long-term statistics of the underlying dynamical system (although that will be examined too), but on predicting the short-term trajectory from a given initial condition. Furthermore, we emphasize that, following the problem formulation introduced by Düeben and Bauer (2018), the system's state-vector is only partially known and the fast/small-scale variables are unknown even during training.

Our objective is more clearly demonstrated using the canonical chaotic system that we will use as a testbed for the data-driven methods: A multi-scale Lorenz 96 system:

$$\frac{dX_k}{dt} = X_{k-1}\left(X_{k+1} - X_{k-2}\right) + F - \frac{hc}{b}\Sigma_j Y_{j,k}, \tag{1}$$

$$\frac{dY_{j,k}}{dt} = -cbY_{j+1,k}\left(Y_{j+2,k} - Y_{j-1,k}\right) - cY_{j,k} + \frac{hc}{b}X_k - \frac{he}{d}\Sigma_i Z_{i,j,k}, \tag{2}$$

$$\frac{dZ_{i,j,k}}{dt} = edZ_{i-1,j,k}\left(Z_{i+1,j,k} - Z_{i-2,j,k}\right) - geZ_{i,j,k} + \frac{he}{d}Y_{j,k}. \tag{3}$$

This set of coupled nonlinear ODEs is a 3-tier extension of Lorenz's original model Lorenz (1996) and has been proposed by Thornes et al. (2017) as a fitting prototype for multi-scale chaotic variability of the weather/climate system and a useful testbed for novel methods. In these equations, $F = 20$ is a large-scale forcing that makes the system highly chaotic; $b = c = e = d = g = 10$ and $h = 1$ are tuned to produce appropriate spatio-temporal variability of the three variables (see below). The indices are $i, j, k = 1, 2, \ldots 8$, thus X has 8 elements while Y and Z have 64 and 512 elements, respectively. Figure 1 shows examples of the chaotic temporal evolution of X, Y, and Z obtained from directly solving Eqs. (1)-(3). The examples demonstrate that X has large amplitudes and slow variability; Y has relatively small amplitudes, high-frequency variability, and intermittency; and Z has small amplitudes and high-frequency variability. In the context of atmospheric circulation, the slow variable X can represent the low-frequency variability of the large-scale circulation while the intermediate variable Y and fast variable Z can represent synoptic/baroclinic eddies and small-scale convection, respectively. Similar analogies in the context of ocean circulation and various other natural or engineering systems can be found, making this multi-scale Lorenz 96 system a useful prototype to focus on.

Our objective is to predict the spatio-temporal evolution of $X(t)$ using a data-driven model that is trained on past observations of $X(t)$ (Figure 2). In the conventional approach of solving Eqs. (1)-(3) numerically, the governing equations have to be known, initial conditions for $Y(t)$ and $Z(t)$ have to be available, and the numerical resolution is dictated by the fast/small-scale variable Z, leading to high computational costs. In the fully data-driven approach that is our objective here, the governing equations do not have to be known, $Y(t)$ and $Z(t)$ do not have to be observed at any time, and evolution of $X(t)$ is predicted just from knowledge of the past observations of X, leading to low computational costs. To successfully achieve this objective, a data-driven method should be able to

1. Accurately predict the evolution of a chaotic system along a trajectory for some time,

2. Account for the effects of $Y(t)$ and $Z(t)$ on the evolution of $X(t)$.

Inspired by several recent studies (which will be discussed below), we have focused on evaluating the performance of three ML techniques in accomplishing (1) and (2). These data-driven methods are

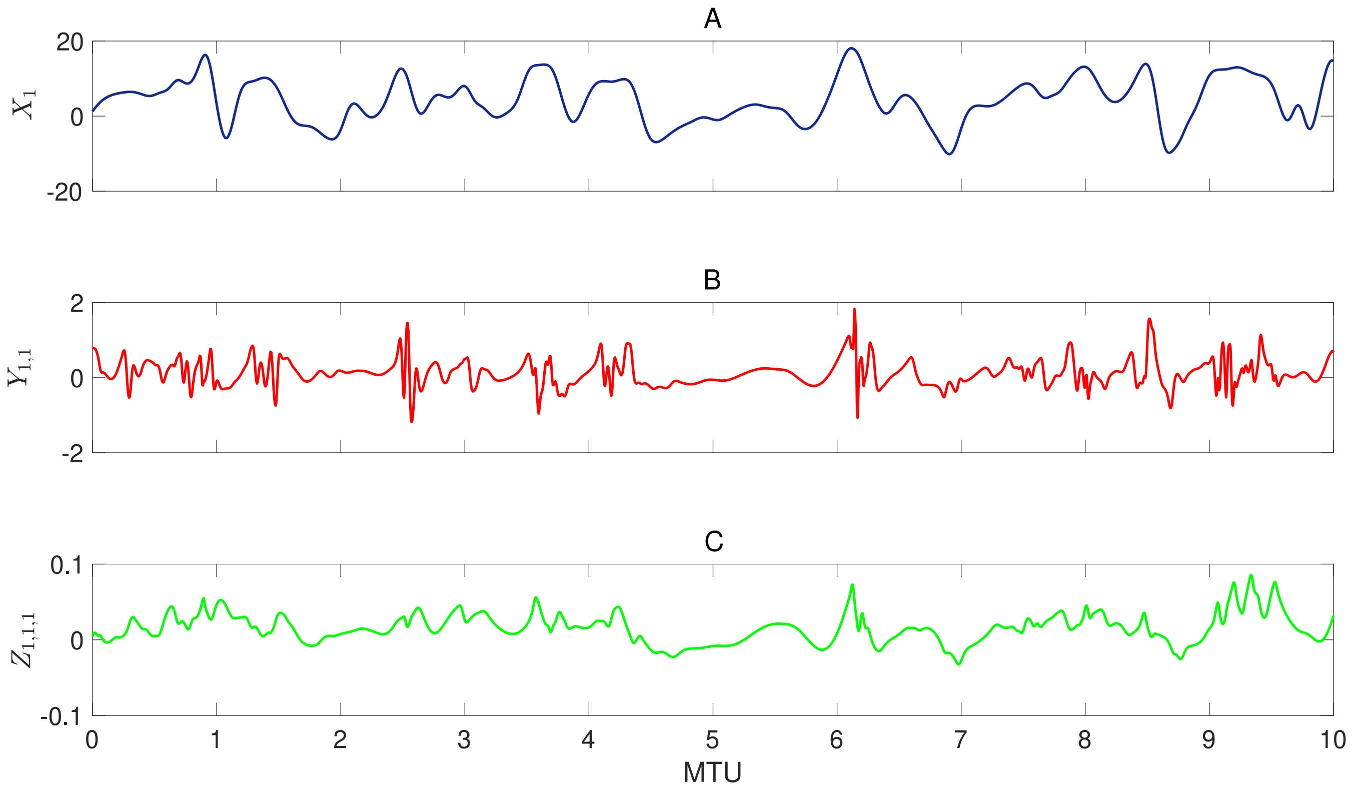


Figure 1. Time evolution of (A) X_k at grid point $k = 1$, (B) $Y_{j,1}$ at grid point $j = 1$, and (C) $Z_{i,1,1}$ at grid point $i = 1$. The time series show chaotic behavior in X, which has large amplitude and slow variability; Y, which has relatively small amplitudes, high frequency variability, and intermittency; Z, which has small amplitudes and high frequency variability. The x-axis is in model time unit (MTU) which is related to the time step of the numerical solver (Δt) and largest positive Lyapunov exponent (λ_{max}) as 1 MTU $= 200\Delta t \approx 4.5/\lambda_{max}$ (see section 2.1). Note that here we are presenting raw data, which have not been standardized for prediction/testing yet.

- RC-ESN: Echo state network (ESN), a specialized type of recurrent neural network (RNN), which belongs to the family of reservoir computing (RC),
- ANN: A deep feed-forward artificial neural network,
- RNN-LSTM: An RNN with long short-term memory (LSTM).

We have focused on these three methods because they have either shown promising performance in past studies (RC-ESN and ANN), or they are considered state of the art in learning from sequential data (RNN-LSTM). There is a growing number of studies focused on using deep learning techniques for data-driven modeling of chaotic and turbulent systems, for example to improve weather/climate modeling and prediction. Some of these studies have been referenced above. Below, we briefly describe three sets of studies with closest relevance to our objective and approach. Pathak and co-workers (Pathak et al., 2018a, 2017; Lu et al., 2017; Fan et al., 2020) have recently shown promising results with RC-ESN for predicting short-term spatio-

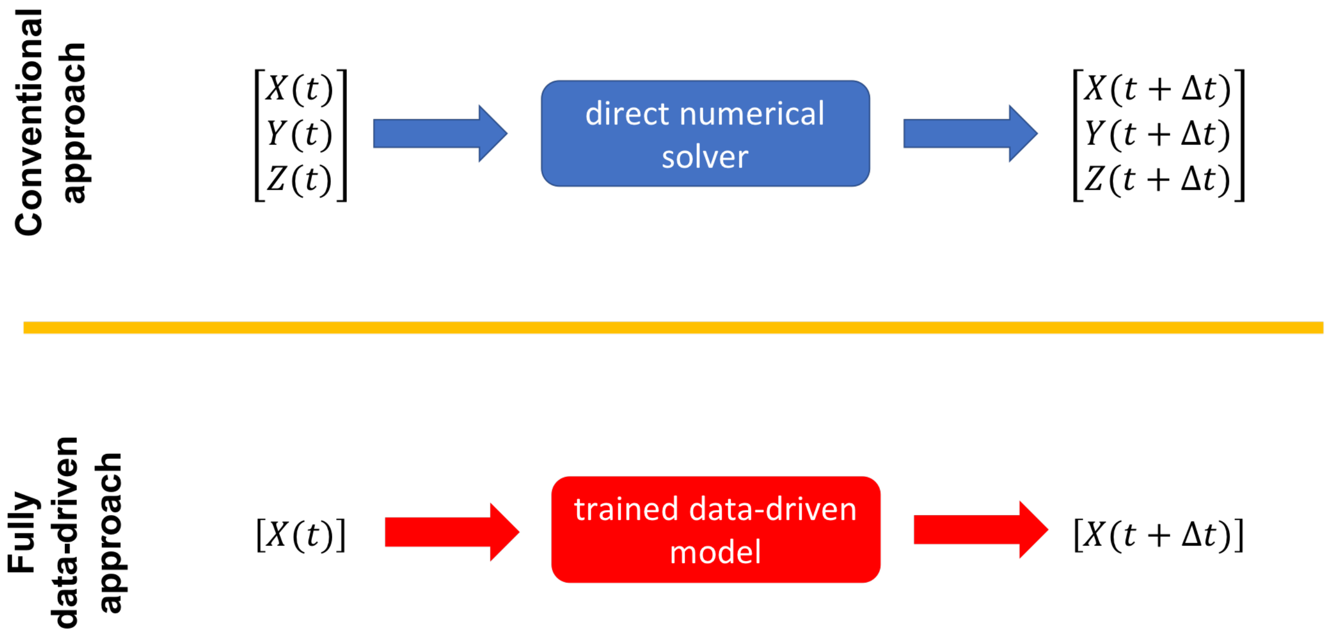

Conventional approach
$\begin{bmatrix} X(t) \\ Y(t) \\ Z(t) \end{bmatrix}$
direct numerical solver
$\begin{bmatrix} X(t+\Delta t) \\ Y(t+\Delta t) \\ Z(t+\Delta t) \end{bmatrix}$
Fully data-driven approach
$[X(t)]$
trained data-driven model
$[X(t+\Delta t)]$

Figure 2. The conventional approach of solving the full governing equations numerically versus our data-driven approach for predicting the spatio-temporal evolution of the slow/large-scale variable X. For the direct numerical solution, the governing equations have to be known and the numerical resolution is dictated by the fast and small-scale variable Z, resulting in high computational costs. In the data-driven approach, the governing equations do not have to be known, Y and Z do not have to be known, and evolution of X is predicted just from knowing the past observations of X, leading to low computational costs.

temporal evolution (item 1) and in replicating attractors for the Lorenz 63 and Kuramoto-Sivashinsky equations. The objective of our paper (items 1-2) and the employed multi-scale Lorenz 96 system are identical to that of Düeben and Bauer (2018), who reported their ANN to have some skills in data-driven prediction of the spatio-temporal evolution of X. Finally, Vlachas et al. (2018) found an RNN-LSTM to have skills (though limited) for predicting the short-term spatio-temporal evolution (item 1) of a number of chaotic toy models such as the original Lorenz 96 system. Here we aim to build on these pioneering studies and examine, *side by side*, the performance of RC-ESN, ANN, and RNN-LSTM in achieving (1) and (2) for the chaotic multi-scale Lorenz 96 system. We emphasize the need for such a side-by-side comparison that uses the exact same system as the testbed and metrics for assessing performance.

The structure of the paper is as follows: in section 2, the multi-scale Lorenz 96 system and the three ML methods are discussed; results on how these methods predict the short-term spatio-temporal evolution of X and reproduce the long-term statistics of X are presented in section 3; key findings and future work are discussed in section 4.

2 Materials and Methods

2.1 The multi-scale Lorenz 96 system

2.1.1 Numerical solution

We have used a 4th-order Runge-Kutta solver with time step $\Delta t = 0.005$ to solve the system of Eqs. (1)-(3). The system has been integrated for 100 million time steps to generate a large dataset for training, testing, and examination of the robustness of the results. In the Results section, we often report time in terms of model time units (MTUs), where 1 MTU $= 200\Delta t$. In terms of Lyapunov timescale, 1 MTU in this system is $\approx 4.5/\lambda_{max}$ (Thornes et al., 2017), where λ_{max} is the largest positive Lyapunov exponent. In terms of the e-folding decorrelation timescale (τ) of the leading principal component (PC1) of X (Khodkar et al., 2019), we estimate 1 MTU $\approx 6.9\tau_{\mathrm{PC1}}$. Finally, as discussed in Thornes et al. (2017), 1 MTU in this system corresponds to ≈ 3.75 Earth days in the real atmosphere.

2.1.2 Training and testing datasets

To build the training and testing datasets from the numerical solution, we have sampled $X \in \Re^8$ at every Δt and then standardized the data by subtracting the mean and dividing by the standard deviation. We have constructed a training set consisting of N sequential samples and a testing set consisting of the next 2000 sequential samples from point $N+1$ to $N+2000$. We have randomly chosen 100 such training/testing sets, each of length $(N+2000)\Delta t$ starting from a random point and separated from the next training/testing set by at least $2000\Delta t$. There is never any overlap between the training and the testing sequences in any of the 100 training/testing sets.

2.2 Reservoir computing-echo state network (RC-ESN)

2.2.1 Architecture

The RC-ESN (Jaeger and Haas, 2004; Jaeger, 2007) is an RNN that has a reservoir with D neurons, which are sparsely connected in an Erdős-Rényi graph configuration (see Figure 3). The connectivity of the reservoir neurons is represented by the adjacency matrix $\mathbf{A}$ of size $D \times D$ whose values are drawn from a uniform random distribution on the interval $[-1, 1]$. The adjacency matrix $\mathbf{A}$ is then normalized by its maximum eigenvalue (Λ_{max}) and then further multiplied with a scalar ($\rho \leq 1$) to build the $\tilde{\mathbf{A}}$ matrix. The state of the reservoir, representing the activations of its constituent neurons, is a vector $r(t) \in \Re^D$. The typical reservoir size used in this study is $D = 5000$ (but we have experimented with D as large as 20000, as discussed later). The other two components of the RC-ESN are an input-to-reservoir layer with weight matrix $\mathbf{W_{in}}$, and a reservoir-to-output layer with weight matrix $\mathbf{W_{out}}$. The inputs for training are N sequential samples of $X(t) \in \Re^8$. At the beginning of the training phase, $\mathbf{A}$ and $\mathbf{W_{in}}$ are chosen randomly and are fixed; i.e., they do not change during training or testing and $\tilde{\mathbf{A}}$ is calculated. During training, only the weights of the output-to-reservoir layer ($\mathbf{W_{out}}$) are updated.

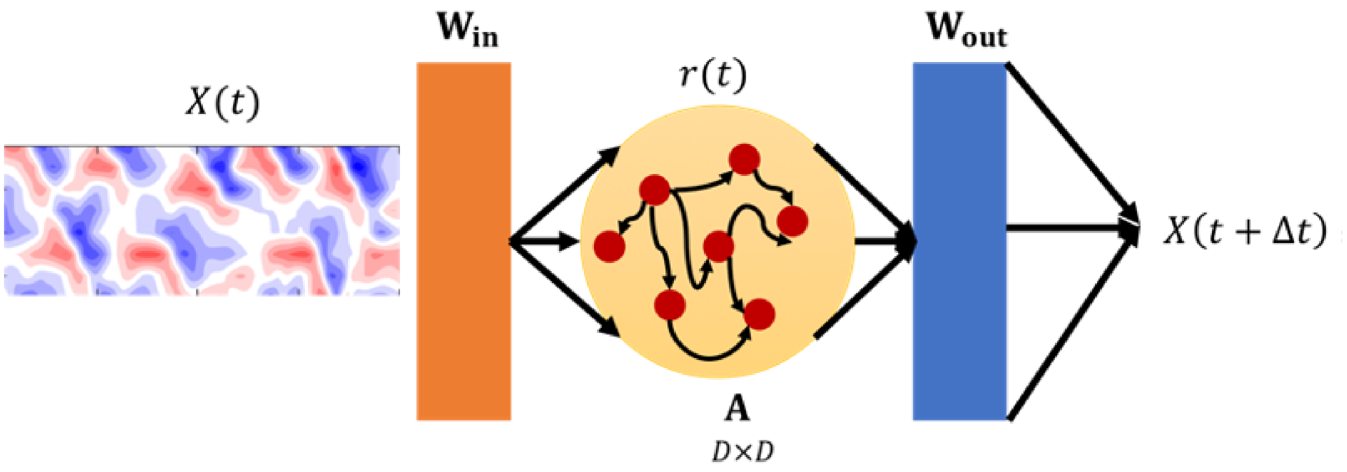


Figure 3. A schematic of RC-ESN. $D \times D$ is the size of the adjacency matrix of the reservoir $\mathbf{A}$. $X(t)$ ($-T \leq t \leq 0$) is the input during training. $\mathbf{W_{in}}$ and $\mathbf{A}$ are random matrices that are chosen at the beginning of training and fixed during training and testing. Only $\mathbf{W_{out}}$ is computed during training. During testing ($t > 0$), $X(t+\Delta t)$ is predicted from a given $X(t)$ that is either known from the initial condition or has been previously predicted.

$\mathbf{W_{out}}$ is the only trainable matrix in this network. This architecture yields a simple training process that has two key advantages:

– It does not suffer from the vanishing and the exploding gradient problem, which has been a major difficulty in training RNNs, especially before the advent of LSTMs (Pascanu et al., 2013),

– $\mathbf{W_{out}}$ can be computed in one shot (see below), thus this algorithm is orders of magnitude faster than the backpropagation through time (BPTT) algorithm (Goodfellow et al., 2016), which is used for training general RNNs and RNN-LSTMs (see section 2.4).

The equations governing the RC-ESN training process are as follows:

$$r(t+\Delta t) = \tanh\left(\tilde{\mathbf{A}} r(t) + \mathbf{W_{in}} X(t)\right), \tag{4}$$

$$\mathbf{W_{out}} = \arg\min_{\mathbf{W_{out}}} ||\mathbf{W_{out}} \tilde{r}(t) - X(t)|| + \alpha ||\mathbf{W_{out}}||. \tag{5}$$

Here $||\cdot||$ is the L_2-norm of a vector and α is the L_2 regularization (ridge regression) constant. Equation (4) maps the observable $X(t) \in \Re^8$ to the higher dimensional reservoir's state $r(t+\Delta t) \in \Re^D$ (reminder: D is $O(1000)$ for this problem). Note that Eq. (5) contains $\tilde{r}(t)$ rather than $r(t)$. As observed by Pathak et al. (2018a) for the Kuramoto-Sivashinsky equation, and observed by us in this work, the columns of the matrix $\tilde{\mathbf{r}}$ should be chosen as nonlinear combinations of the columns of the reservoir state matrix $\mathbf{r}$ (each row of the matrix $\mathbf{r}$ is a snapshot $r(t)$ of size D). For example, following Pathak et al. (2018a), we compute $\tilde{\mathbf{r}}$ as having the same even columns as $\mathbf{r}$ while its odd columns are the square of the odd columns of $\mathbf{r}$ (algorithm T_1

hereafter). As shown in Appendix A, we have found that a nonlinear transformation (between $\mathbf{r}$ and $\tilde{\mathbf{r}}$) is essential for skillful predictions, while several other transformation algorithms yield similar results as T_1. The choices of these transformations (T_2 and T_3, see Appendix A), although not based on a rigorous mathematical analysis, are inspired from the nature of the quadratic nonlinearity that is present in Eqs. (1)-(3).

The prediction process is governed by:

$$X(t+\Delta t) = \mathbf{W_{out}}\tilde{r}(t+\Delta t), \tag{6}$$

$\tilde{r}(t+\Delta t)$ in Eq. (6) is computed by applying one of the T_1, T_2 or T_3 algorithms on $r(t+\Delta t)$, which itself is calculated via Eq. (4) from $X(t)$ that is either known from initial condition or has been previously predicted.

See (Jaeger and Haas, 2004; Jaeger, 2007; Lukoševičius and Jaeger, 2009; Gauthier, 2018) and references therein for further discussions on RC-ESNs, and (Lu et al., 2017; Pathak et al., 2017; McDermott and Wikle, 2017; Pathak et al., 2018a, b; Zimmermann and Parlitz, 2018; Lu et al., 2018; McDermott and Wikle, 2019; Lim et al., 2019) for examples of recent applications to dynamical systems.

2.2.2 Training and Prediction

During training ($-T \leq t \leq 0$), $\mathbf{W_{in}}$ and $\mathbf{A}$ are initialized with random numbers, which stay fixed during the training (and testing) process, The state matrix $\mathbf{r}$ is initialized to $\mathbf{0}$. For all the experiments conducted in this study, we have empirically found that the value of $\rho = 0.1$ to yield the best results in our experiments. We have further found that between $0.05 \leq \rho \leq 0.7$, the prediction horizon of RC-ESN is not sensitive. Then the state matrix $\mathbf{r}$ is computed using Eq. (4) for the training set, and $\mathbf{W_{out}}$ is computed using Eq. (5). During prediction (i.e., testing) corresponding to $t > 0$, the computed $\mathbf{W_{out}}$ is used to march forward in time (Eqs. (6)) while as mentioned earlier, $r(t)$ keeps getting updated using the predicted $X(t)$ (Eq. (4)). A non-linear transformation is used to compute $\tilde{r}(t)$ from $r(t)$ before using Eq. (6).

Our RC-ESN architecture and training/prediction procedure are similar to the ones used in (Pathak et al., 2018a). There is no overfitting in the training phase because the final training and testing accuracies are the same. Our code is developed in Python and is made publicly available (see Code and data availability).

2.3 Feed-forward artificial neural network (ANN)

2.3.1 Architecture

We have developed a deep ANN that has the same architecture as the one used in Düeben and Bauer (2018) (which they employed to conduct prediction on the same multi-scale Lorenz 96 studied here). The ANN has 4 hidden layers with 100 neurons each, and 8 neurons in the input and output layers (Figure 4). It should be noted that unlike RC-ESN (and RNN-LSTM), this ANN is stateless, i.e., there is no hidden variable such as $r(t)$ that tracks temporal evolution. Furthermore, unlike RC-ESN, for which the input and output are $X(t)$ and $X(t+\Delta t)$, following Düeben and Bauer (2018), the ANN's inputs/outputs are chosen to be pairs of $X(t)^{train}$ and $\Delta X(t)^{train} = X(t+\Delta t)^{train} - X(t)^{train}$ (Figure 4). During prediction, using $X(t)^{test}$

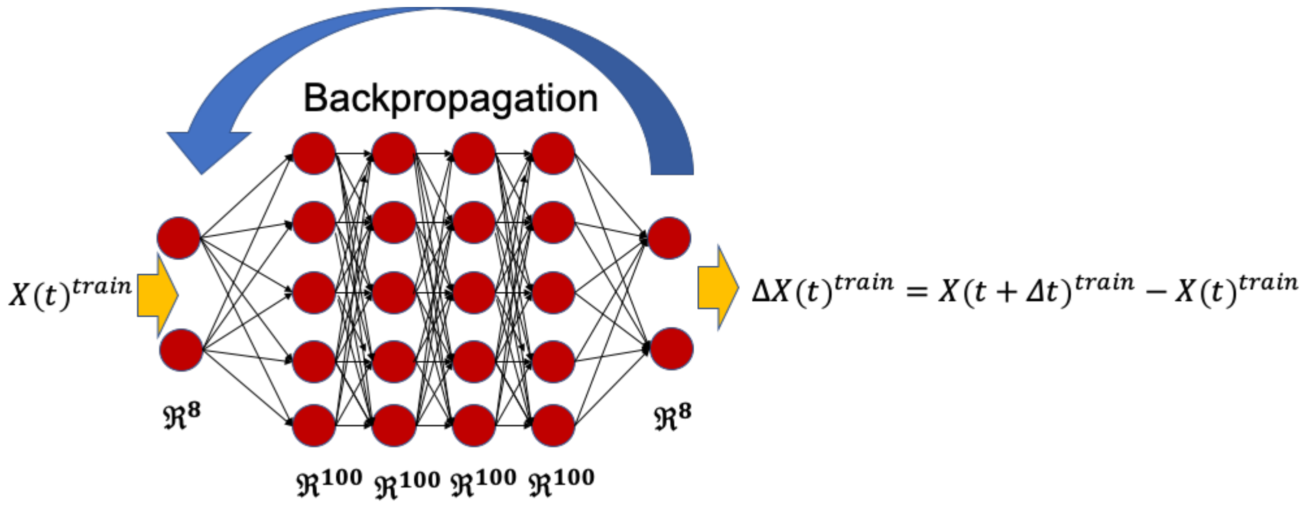

Backpropagation
$X(t)^{train}$
$\Re^8$
$\Re^{100}$ $\Re^{100}$ $\Re^{100}$ $\Re^{100}$
$\Re^8$
$\Delta X(t)^{train} = X(t+\Delta t)^{train} - X(t)^{train}$

Figure 4. A schematic of ANN, which has 4 hidden layers each with 100 neurons, and an input and an output layer each with 8 neurons (for better illustration only few neurons from each layer are shown). Training is performed on pairs of $X(t)^{train}$ and $\Delta X(t)^{train}$, and the weights in each layer are learned through the backpropagation algorithm. During testing (i.e., prediction) $\Delta X(t)^{test}$ is predicted for a given $X(t)^{test}$ that is already predicted or is known from initial condition. Knowing $X(t)^{test}$, $\Delta X(t)^{test}$, and $\Delta X(t-\Delta t)^{test}$, $X(t+\Delta t)^{test}$ is then computed using Adams-Bashforth integration scheme.

that is known from initial condition or previously calculated, $\Delta X(t)^{test}$ is predicted. $X(t+\Delta t)^{test}$ is then computed via Adams-Bashforth integration scheme

$$X(t+\Delta t)^{test} = X(t)^{test} + \frac{1}{2}\left[3\Delta X(t)^{test} - \Delta X(t-\Delta t)^{test}\right]. \tag{7}$$

More details on the ANN architecture has been reported in Appendix B.

2.3.2 Training and Prediction

Training is performed on N pairs of sequential $(X(t)^{train}, \Delta X(t)^{train})$ from the training set. During training, the weights of the network are computed using backpropagation optimized by the stochastic gradient descent algorithm. During prediction, as mentioned above, $\Delta X(t)^{test}$ is predicted from $X(t)^{test}$ that is known from initial condition or has been previously predicted, and $X(t+\Delta t)^{test}$ is then calculated using Eq. (7).

Our ANN architecture and training/prediction procedure are similar to the ones used in Düeben and Bauer (2018) (we have optimized, by trial and error, the hyperparameters for this particular network). There is no overfitting in the training phase because the final training and testing accuracies are the same. Our code is developed in Keras and is made publicly available (see Code and data availability).

2.4 Recurrent neural network with long short-term memory (RNN-LSTM)

2.4.1 Architecture

The RNN-LSTM (Hochreiter and Schmidhuber, 1997) is a deep learning algorithm most suited for prediction of sequential data such as time series, and has received a lot of attention in recent years (Goodfellow et al., 2016). Variants of RNN-LSTMs are the best performing models for time series modeling in areas such as stock pricing (Chen et al., 2015), supply chain (Carbonneau et al., 2008), natural language processing (Cho et al., 2014), and speech recognition (Graves et al., 2013). A major improvement over regular RNNs, which have issues with the vanishing and the exploding gradients (Pascanu et al., 2013), LSTMs have become the state-of-the-art approach for training RNNs in the deep learning community. Unlike regular RNNs, the RNN-LSTMs have gates that control the information flow into the neural network from previous time steps of the time series. The RNN-LSTM used here, like the RC-ESN but unlike the ANN, is stateful (i.e., actively maintains state). Our RNN-LSTM has 50 hidden layers in each RNN-LSTM cell. More details on our RNN-LSTM are presented in C. RNN-LSTMs have many tunable parameters and are trained with the expensive BPTT algorithm (Goodfellow et al., 2016).

2.4.2 Training and Prediction

The input to the RNN-LSTM is a time-delay-embedded matrix of $X(t)$ with embedding dimension q (also known as lookback) (Kim et al., 1999). An extensive hyperparameter optimization (by trial and error) is performed to find the optimal value of q for which the network has the largest prediction horizon (exploroing $q = 1 - 22$, we found $q = 3$ to yield the best performance). The RNN-LSTM predicts $X(t+\Delta t)$ from the previous q time steps of $X(t)$. This is in contrast with RC-ESN, which only uses $X(t)$ and reservoir state $r(t)$ to predict $X(t+\Delta t)$, and ANN, which only uses $X(t)$ and no state to predict $X(t+\Delta t)$ (via predicting $\Delta X(t)$). The weights of the LSTM layers are determined during the training process (see C). During testing, $X(t+\Delta t)$ is predicted using the past q observables $[X(t-(q-1)\Delta t)\dots X(t-\Delta t), X(t)]$ that are either known from initial condition or have been previously predicted. We have found the best results with a stateless LSTM (which means that the state of the LSTM gets refreshed during the beginning of each batch during training, see C) that outperforms a stateful LSTM (where the states gets carried over to the next batch during training).

The architecture of our RNN-LSTM is similar to the one used in Vlachas et al. (2018). There is no overfitting in the training phase because the final training and testing accuracies are the same. Our code is developed in Keras and is made publicly available (see Code and data availability).

3 Results

3.1 Short-term prediction: Comparison of the RC-ESN, ANN, and RNN-LSTM performances

The short-term prediction skills of the three ML methods for the same training/testing sets are compared in Figure 5. Given the chaotic nature of the system, the performance of the methods depends on the initial condition from which the prediction

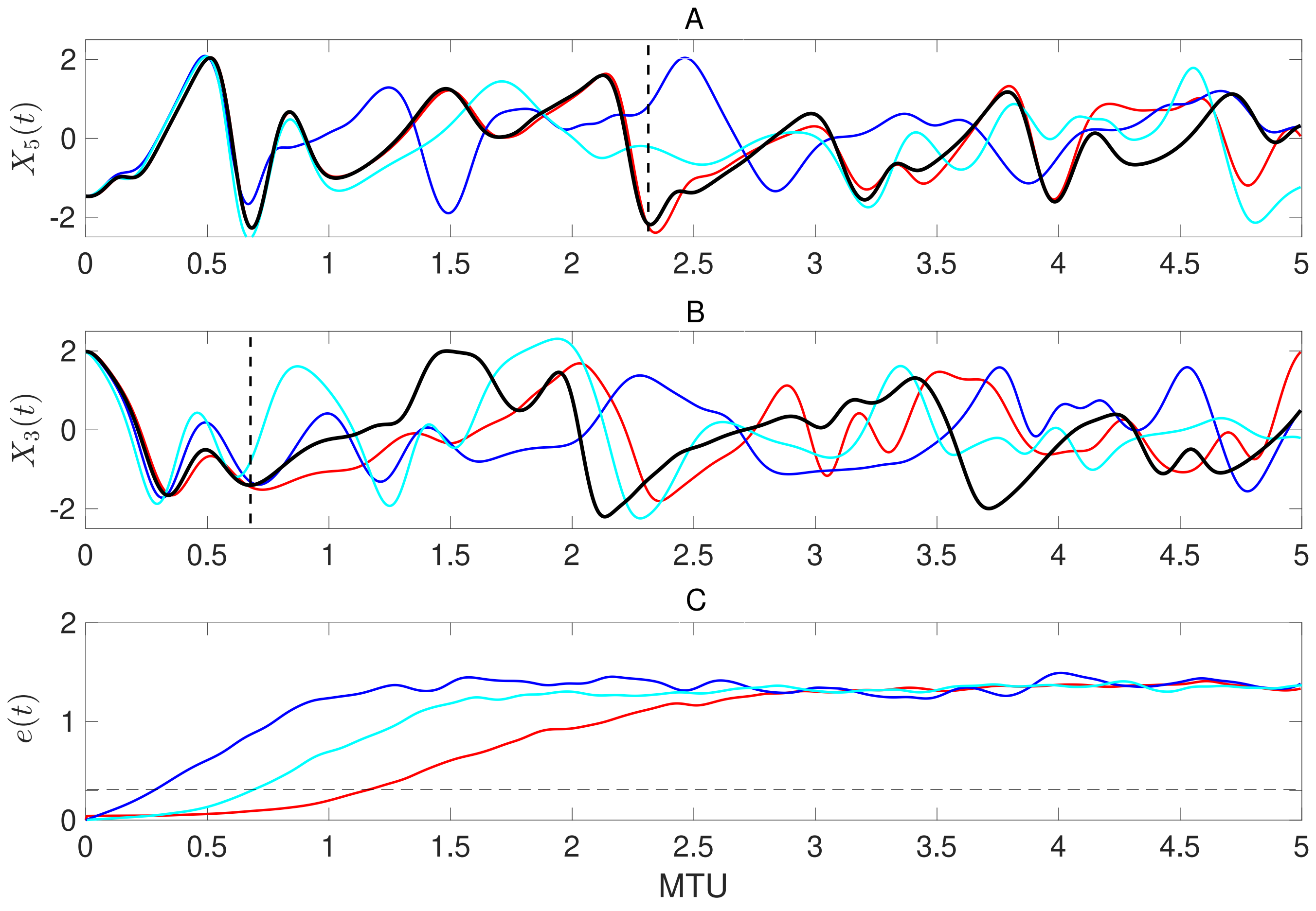


Figure 5. Comparison of the short-term forecast skills among the three ML methods. The lines show truth (black), RC-ESN (red), ANN (blue), and RNN-LSTM (cyan). (A) and (B) show examples (differing in initial condition) for which RC-ESN yields the longest and shortest prediction horizons, respectively. Vertical dashed lines show the approximate prediction horizon of RC-ESN. (C) Relative L_2 error averaged over 100 randomly chosen initial conditions $e(t)$ (see Eq. (8)). The dashed horizontal line marks $e = 0.3$. Time is in MTU, where 1 MTU $= 200\Delta t \approx 4.5/\lambda_{max}$. Training size is $N = 500000$. $\rho = 0.1$ is used here.

is conducted. To give the readers a comprehensive view of the performance of these methods, Figures 5(A) and (B) show examples of the predicted trajectories (for one element of $X(t)$) versus the true trajectory for two specific initial conditions (from the 100 initial conditions we used): the one for which RC-ESN shows the best performance (Figure 5(A)), and the one for which RC-ESN shows the worst performance (Figure 5(B)).

For the initial condition in Figure 5(A), RC-ESN accurately predicts the time series for over 2.3 MTU, which is equivalent to $460\Delta t$ and over 10.35 Lyapunov timescales. Closer examination shows that the RC-RSN prediction follows the true trajectory well even up to ≈ 4 MTU. The RNN-LSTM has the next best prediction performance (up to around 0.9 MTU or $180\Delta t$). The prediction from ANN is for around 0.6 MTU or $120\Delta t$. For the example in Figure 5(B), all methods have shorter prediction horizons, but RC-ESN still has the best performance (accurate prediction up to ≈ 0.7 MTU), followed by ANN and RNN-LSTM with similar prediction accuracies (≈ 0.3 MTU). Searching through all 100 initial conditions, the best prediction with RNN-LSTM is up to ≈ 1.7 MTU (equivalent to $340\Delta t$), and the best prediction with ANN is up to ≈ 1.2 MTU (equivalent to $240\Delta t$),

To compare the results over all 100 randomly chosen initial conditions, we have defined an averaged relative L_2 error between the true and predicted trajectories

$$e(t) = \left[\frac{||X_{true}(t) - X_{pred}(t)||}{\langle ||X_{true}(t)|| \rangle}\right]. \tag{8}$$

Here $[\cdot]$ and $\langle\cdot\rangle$ indicate, respectively, averaging over 100 initial conditions and over $2000\Delta t$. To be clear, $X_{true}(t)$ refers to the data at time t obtained from the numerical solution while $X_{pred}(t)$ refers to the predicted value at t using one of the deep learning methods. Figure 5(C) compares $e(t)$ for the three methods. It is clear that RC-ESN significantly outperforms ANN and RNN-LSTM for short-term prediction in this multi-scale chaotic testbed. Figure 6 shows an example of the spatio-temporal evolution of X_{pred} (from RC-ESN), X_{true}, and their difference, which further demonstrates the capabilities of RC-ESN for short-term spatio-temporal prediction.

3.2 Short-term prediction: Scaling of RC-ESN and ANN performance with training size N

How the performance of ML techniques scales with the size of the training set is of significant practical importance, as the amount of available data is often limited in many problems. Given that currently there is no theoretical understanding of such scaling for these ML techniques, we have empirically examined how the quality of short-term predictions scales with N. We have conducted the scaling analysis for $N = 10^4$ to $N = 2\times10^6$ for the three methods. Two metrics for the quality of prediction are used: the prediction horizon, defined as when the averaged L_2 error $e(t)$ reaches 0.3, and the prediction error E, defined as the average of $e(t)$ between $0 - 0.5$ MTU:

$$E = \frac{1}{100\Delta t}\sum_{i=0}^{i=100} e(i\Delta t). \tag{9}$$

Figure 7(A) shows that for all methods, the prediction horizon increases monotonically, but nonlinearly, as we increase N. The prediction horizons of RC-ESN and ANN appear to saturate after $N = 10^6$, although the RC-ESN has a more complex step-like scaling curve that needs further examination in future studies. The prediction horizon of RC-ESN exceeds that of ANN by factors ranging from 3 (for high N) to 9 (for low N). In the case of RNN-LSTM, the factor ranges from 1.2 (for high N) to 2 (for low N). Figure 7(B) shows that for all methods, the average error E decreases as N increases (as expected), most notably for ANN when N is small.

Overall, compared to both RNN-LSTM and ANN, the prediction horizon and accuracy of RC-ESN have a weaker dependence on the size of the training set, which is a significant advantage for RC-ESN when the dataset available for training is short, which is common in many practical problems.

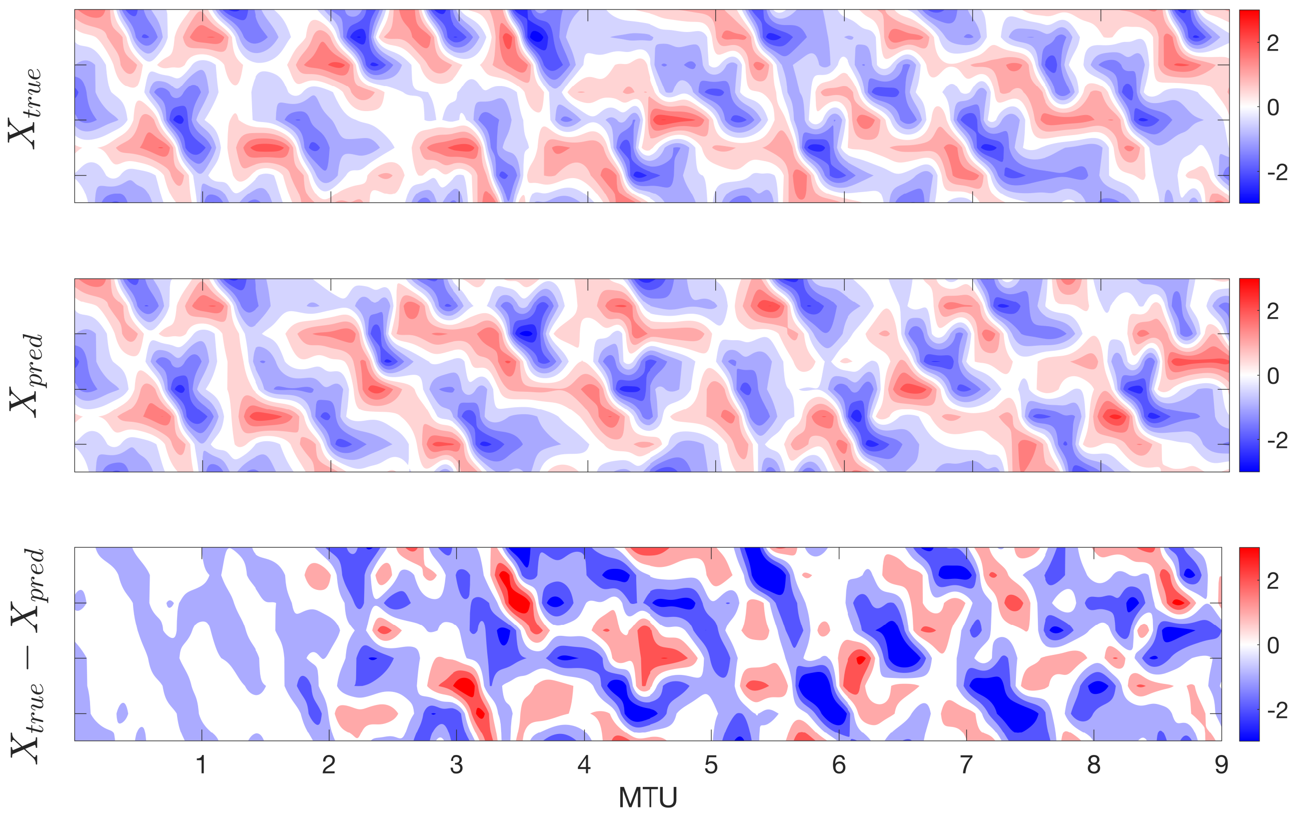


Figure 6. Performance of RC-ESN for short-term spatio-temporal prediction. We remind the readers that only the slow/large-scale variable X has been used during training/testing, and the fast/small-scale variables Y and Z have not been used at any point during training or testing. RC-ESN has substantial forecast skills, providing accurate predictions up to ≈ 2 MTU, which is around $400\Delta t$ or 9 Lyapunov timescales. $N = 500000$, algorithm T_2, $D = 5000$, and $\rho = 0.1$ are used.

3.3 Short-term prediction: Scaling of RC-ESN performance with reservoir size D

Given the superior performance of RC-ESN for short-term prediction, here we focus on one concern with this method: the need for large reservoirs, which can be computationally demanding. This issue has been suggested as a potential disadvantage of ESNs versus LSTMs for training RNNs (Jaeger, 2007). Aside from the observations here that RC-ESN significantly outperforms RNN-LSTM for short-term predictions, the problem of reservoir size can be tackled in at least two ways. First, Pathak et al. (2018a) have proposed, and shown the effectiveness of, using a set of parallelized reservoirs, which allowed for easily dealing with high-dimensional chaotic toy models.

Second, Figure 8 shows that E rapidly declines by a factor of around 3 as D is increased from 500 to 5000, decreases slightly as D is further doubled to 10000, and then barely changes as D is doubled again to 20000. Training the RC-ESN with $D = 20000$ versus $D = 5000$ comes with a higher computational cost (16 GB memory and ≈ 6 CPU hours for $D = 5000$ and

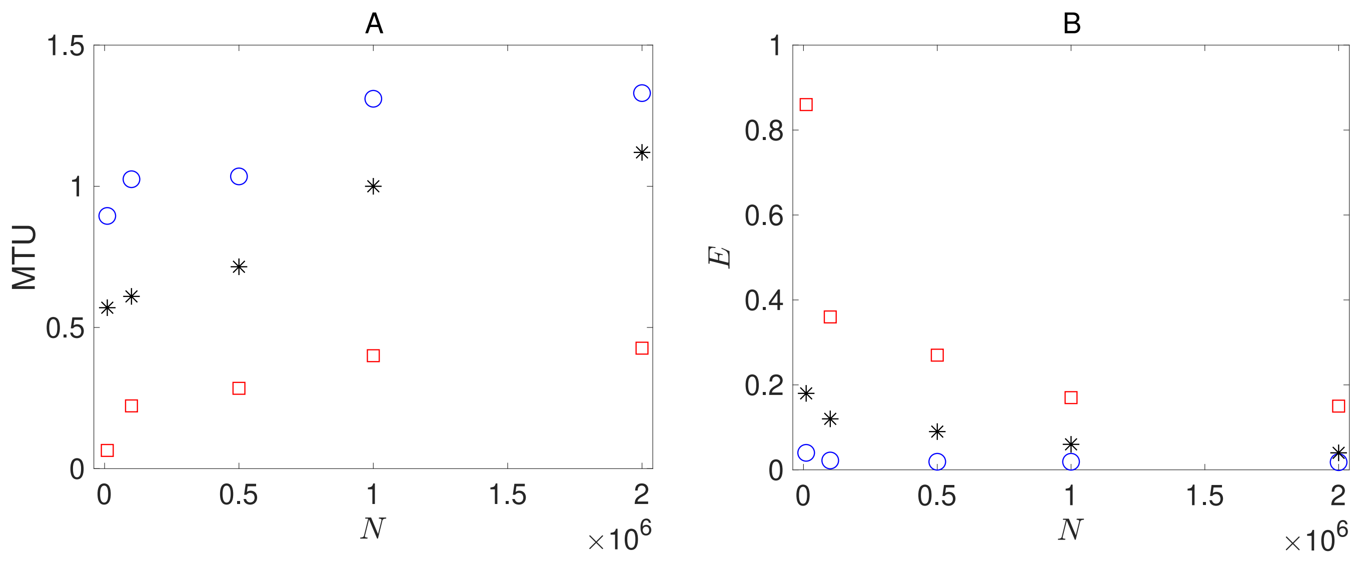


Figure 7. Comparison of the short-term prediction quality for RC-ESN (blue circles), RNN-LSTM (black stars), and ANN (red squared) as the size of the training set N is changed from $N = 10^4$ to $N = 2 \times 10^6$. (A) Prediction horizon (when $e(t)$ reaches 0.3). (B) Average error E (see Eq. (9)). The hyperparameters in each method are optimized for each N.

64 GB memory and ≈ 18 CPU hours for $D = 20000$), while little improvement in accuracy is gained. Thus, concepts from inexact computing can be used to choose D such that precision is traded for large savings in computational resources, which can be then reinvested into more simulations, higher resolutions for critical processes etc. (Palem, 2014; Palmer, 2014; Düben and Palmer, 2014; Leyffer et al., 2016; Thornes et al., 2017).

3.4 Long-term statistics: Comparison of RC-ESN, ANN, and RNN-LSTM performance

All the data-driven predictions discussed earlier eventually diverge from the true trajectory (as would even predictions using the numerical solver). Still, it is interesting to examine whether the freely predicted spatio-temporal data have the same long-term statistical properties as the actual dynamical system (i.e., Eqs. (1)-(3)). Reproducing the actual dynamical system's long-term statistics (sometimes refer to as the system's *climate*) using a data-driven method can be significantly useful. In some problems, a surrogate model does not need to predict the evolution of a specific trajectory, but only the long-term statistics of a system. Furthermore, synthetic long datasets produced using an inexpensive data-driven method (trained on a shorter real dataset) can be used to examine the system's probability density functions (PDFs), including its tails, which are important for studying the statistics of rare/extreme events.

By examining return maps, Pathak et al. (2017) have already shown that RC-ESNs can reproduce the long-term statistics of the Lorenz 63 and Kuramoto-Sivashinsky equations (Jaeger and Haas (2004) and Pathak et al. (2017) have also shown that RC-ESNs can be used to accurately estimate a chaotic system's Lyapunov spectrum). Here, we focus on comparing the performance of RC-ESN, ANN, and RNN-LSTM in reproducing the system's long-term statistics by

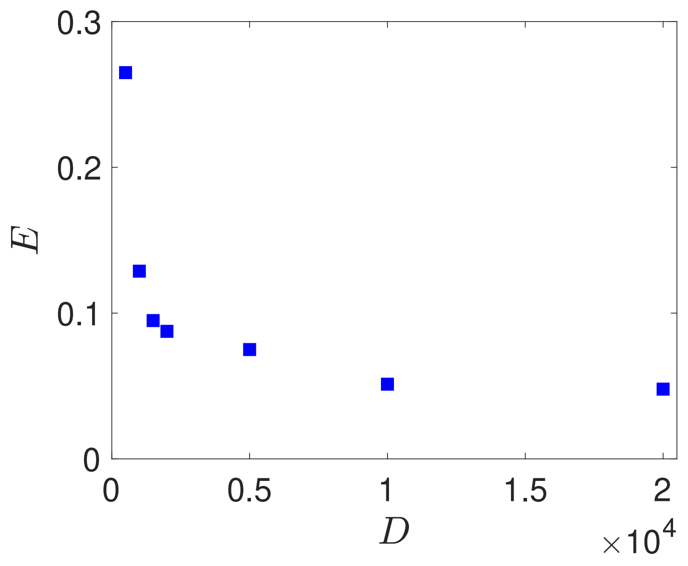


Figure 8. For RC-ESN: scaling of average prediction error between $0 - 0.5$ MTU (E, Eq. (9)) with reservoir size D. $N = 500000$ is used.

- Examining the estimated PDFs and in particular their tails,
- Investigating whether the quality of the estimated PDFs degrades with time, which can negate the usefulness of long synthetic datasets.

Figure 9 compares the estimated PDFs obtained using the three ML methods. The data predicted using RC-ESN and RNN-LSTM are found to have PDFs closely matching the true PDF, even at the tails. Deviations at the tails of the PDF predicted by these methods from the true PDF are comparable to the deviations of the PDFs obtained from true data using the same number of samples. The ANN-predicted data have reasonable PDFs between ± 2 standard deviations, but the tails have substantial deviations from those of the true PDFs. All predicted PDFs are robust and do not differ much (except near the end of the tails) among the quartiles.

The results show that RC-ESN and RNN-LSTM can accurately reproduce the system's long-term statistics, and can robustly produce long synthetic datasets with PDFs that are close to the PDF of the true data even near the tails. The ability of ANN to accomplish these tasks is limited.

4 Discussion

By examining the true and predicted trajectories (Figures 5-6) and the prediction errors and horizons (Figure 7), we have shown that RC-ESN substantially outperforms ANN and RNN-LSTM in predicting the short-term evolution of a multi-scale Lorenz 96 chaotic system (Eqs. (1)-(3)). Additionally, RC-ESN and RNN-LSTM both work well in reproducing the long-term statistics of this system (Figure 9). We emphasize that following the problem formulation of Düeben and Bauer (2018), and

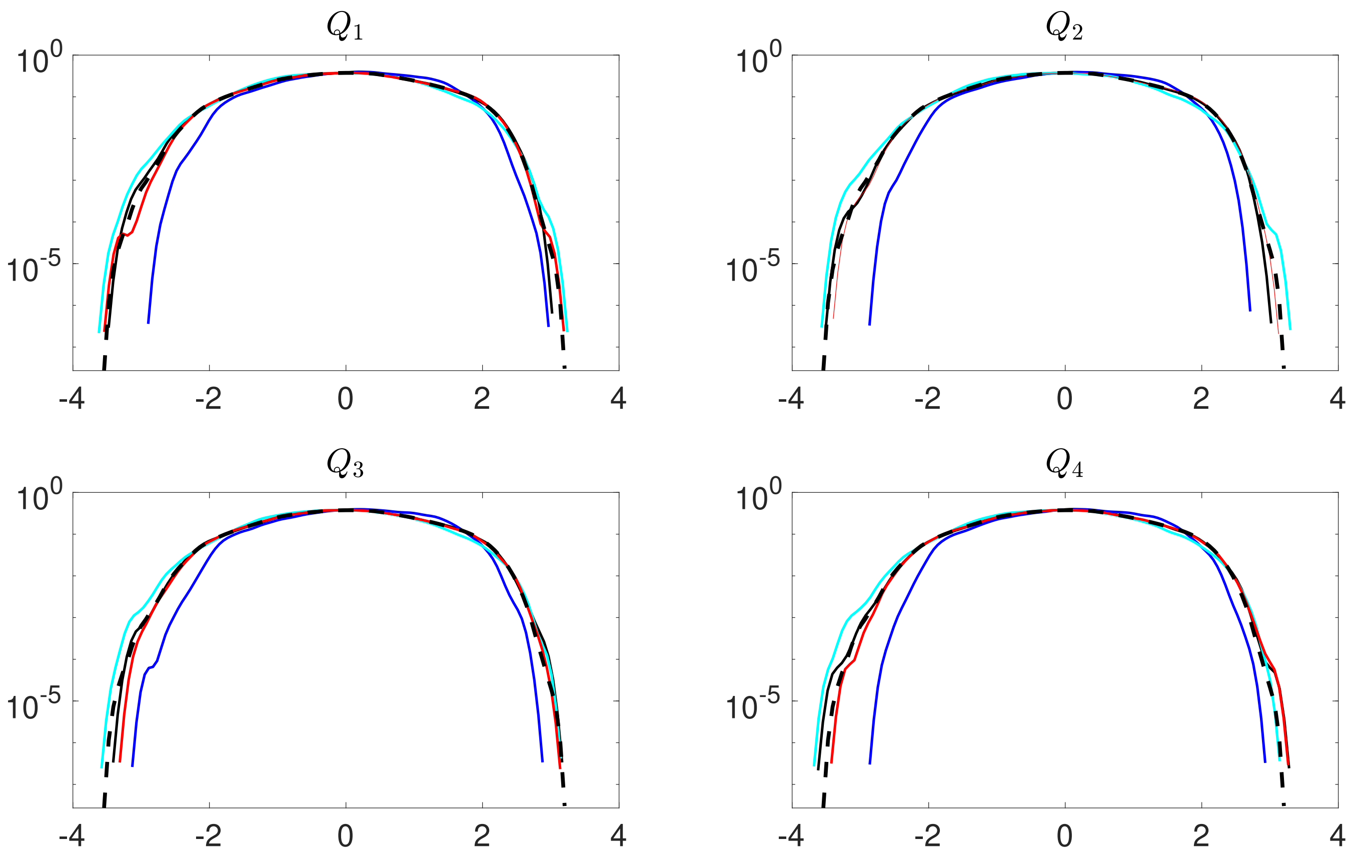


Figure 9. Estimated probability density functions (PDFs) of long datasets freely predicted by using RC-ESN (red), ANN (blue), and RNN-LSTM (cyan) compared to the true PDF (dashed black). RC-ESN, ANN, and RNN-LSTM are first trained with $N = 500000$, and then predict, from an initial condition, for $4 \times 10^6 \Delta t$. The panels $Q_1 - Q_4$ correspond to the equally divided quartiles of 10^6 predicted samples. PDFs are approximated using kernel density estimation (Epanechnikov, 1969). The green lines show the estimated PDFs from different sets of 10^6 samples that are obtained from numerical solution of Eqs. (1)-(3). Small differences between the green lines show the uncertainties in estimating the tails from 10^6 samples. The dashed black lines show the true PDF estimated from 10^7 samples from numerical solution. Note that the presented PDFs are for standardized data.

unlike most other studies, only part of the multi-scale state-vector (the slow/large-scale variable X) has been available for training the data-driven model and has been of interest for testing. This problem design is more relevant to many practical problems but is more challenging as it requires the data-driven model to not only predict the evolution of X based on its past observations, but also to account for the effects of the intermediate and fast/small-scale variables Y and Z on X.

315 We have also found that the prediction horizon, and in particular, the prediction accuracy, of RC-ESN to have a weak dependence on the size of the training set (Figure 7). This is an important property, as in many practical problems the data available for training are limited. Furthermore, the prediction error of RC-ESN is shown to have an asymptotic behavior for large reservoir sizes, which suggests that reasonable accuracy can be achieved with moderate reservoir sizes. Note that the

skillful predictions with RC-ESN in Figures 5-6 have been obtained with a moderate-sized training set ($N = 500000$) and reservoir ($D = 5000$). Figures 7 and 8 suggest that slightly better results could have been obtained using larger N and D, although such improvements come at a higher computational cost during training.

The order we have found for the performance of the three deep learning methods (RC-ESN $>$ RNN-LSTM $>$ ANN) is different from the order of complexity of these methods in terms of the number of trainable parameters (RC-ESN $<$ ANN $<$ RNN-LSTM) but aligned with the order in terms of the learnable function space. ANN $<$ RC-ESN and RNN-LSTM, as the latter methods are both RNNs and thus Turing complete, while ANN is a state-less feed-forward network and thus not Turing complete (Siegelmann and Sontag, 1992). Turing completeness of RNNs, in theory, enables them to compute any function or implement any algorithm. Furthermore, state augmentation of ANN is not possible because it has no state. Whether the superior predictive performance of RC-ESN (especially over RNN-LSTM) is due to its explicit state representation, which is updated at every time step both during training and testing, or lower likelihood of overfitting due to the order-of-magnitude smaller number of trainable parameters, remains to be seen. Also whether we have not fully harnessed the power of RNN-LSTMs (see below) is unclear at this point, particularly because a complete theoretical understanding of how/why these methods work (or do not work) is currently lacking. That said, there have been some progress in understanding the RC-ESNs, in particular by modeling the network itself as a dynamical system (Yildiz et al., 2012; Gauthier, 2018). Such efforts, for example those aimed at understanding the echo states that are learned in the RC-ESN's reservoir, might benefit from recent advances in dynamical systems theory (Mezić, 2005; Tu et al., 2014; Williams et al., 2015; Arbabi and Mezic, 2017; Giannakis et al., 2017; Khodkar and Hassanzadeh, 2018).

An important next step in our work is to determine how generalizable our findings are. This investigation is important for the following two reasons. First, here we have only studied one system, a specially designed version of the Lorenz 96 system. The performance of these methods should be examined in a hierarchy of chaotic dynamical systems and high-dimensional turbulent flows. That said, our findings are overall consistent with the recently reported performance of these methods applied to chaotic toy models. (Pathak et al., 2018a, 2017; Lu et al., 2017) demonstrated that RC-ESN can predict, for several Lyapunov timescales, the spatio-temporal evolution, and can reproduce the climate of the Lorenz 63 and Kuramoto-Sivashinsky chaotic systems. Here, we have shown that RC-ESN performs similarly well even when only the slow/large-scale component of the multi-scale state-vector is known. Our results with ANN are consistent with those of Düeben and Bauer (2018), who showed examples of trajectories predicted accurately up to 1 MTU with a large training set ($N = 2 \times 10^6$) using ANN for the same Lorenz 96 system (see their Fig. 1). While RNN-LSTM is considered state-of-the-art for sequential data modeling, and has worked well for a number of applications involving time series, to the best of our knowledge, simple RNN-LSTMs, such as the one used here, have not been very successful when applied to chaotic dynamical systems.Vlachas et al. (2018) found some prediction skills using RNN-LSTM applied to the original Lorenz 96 system, (which does not have the multi-scale coupling); (see their Fig. 5).

Second, we have only used a simple RNN-LSTM. There are other variants of this architecture as well as more advanced deep learning RNNs that might potentially yield better results. For our simple RNN-LSTM, we have extensively explored optimization of hyperparameters and tried variant formulations of the problem: predict $\Delta X(t)$ with or without updating the state of the

LSTM. We have found that such variants do not lead to better results, consistent with the findings of Vlachas et al. (2018). Our preliminary explorations with more advanced variants of RNN-LSTM (seq2seq and encoder-decoder LSTM (Sutskever et al., 2014)) have not resulted in any improvement either. However, just as the skillful predictions of RC-ESN and ANN hinge on one key step (nonlinear transformation for the former and predicting ΔX for the latter), it is possible that changing/adding one step leads to major improvements in RNN-LSTM (it is worth mentioning that similar nonlinear transformation of the state in LSTMs is not straightforward due to their more complex architecture; see Appendix C). We have shared our codes publicly to help others explore such changes to our RNN-LSTM. Furthermore, there are other more sophisticated RNNs that we have not yet explored. For example (Yu et al., 2017) have introduced a tensor-train RNN that outperformed a simple RNN-LSTM in predicting the temporal evolution of the Lorenz 63 chaotic system. Mohan et al. (2019) showed that a compressed convolutional LSTM performs well in capturing the long-term statistics of a turbulent flow. The performance of more advanced deep learning RNNs should be examined and compared, side by side, with the performance of RC-ESNs and ANNs in future studies. The multi-scale Lorenz 96 system provides a suitable starting point for such comparisons.

The results of our work show the promise of ML methods such as RC-ESNs (and to some extent RNN-LSTM) for data-driven modeling of chaotic dynamical systems, which has broad applications in geosciences, e.g., in weather/climate modeling. Practical and fundamental issues such as interpretability, scalability to higher dimensional systems (Pathak et al., 2018a), presence of measurement noise in the training data and initial conditions (Rudy et al., 2018), non-stationarity of the time series, and dealing with data that have two or three spatial dimensions (e.g., through integration with convolutional neural networks, CNN-LSTM (Xingjian et al., 2015) and CNN-ESN (Ma et al., 2017) should be studied in future work.

Here we have focused on a fully data-driven approach, as opposed to the conventional approach of direct numerical solutions (Figure 2). In practice, for example for large-scale, multi-physics, multi-scale dynamical systems such as weather and climate models, it is likely that a hybrid framework yields the best performance: depending on the application and the spatio-temporal scales of the physical processes involved (Thornes et al., 2017; Chantry et al., 2019), some of the equations could be solved numerically with double precision, some could be solved numerically with lower precisions, and some could be approximated with a surrogate model learned via a data-driven approach, such as the ones studied in this paper.

Appendix A: More details on RC-ESN

Here we show a comparison of RC-ESN forecast skills with nonlinear transformation algorithms T_1 (Pathak et al., 2018a), T_2 and T_3, and without any transformation between $\mathbf{r}$ and $\tilde{\mathbf{r}}$. The three algorithms are (for $i = 1, 2, 3 \ldots N$ and $j = 1, 2, 3 \ldots D$)

Algorithm T_1

$\tilde{\mathbf{r}}_{i,j} = \mathbf{r}_{i,j} \times \mathbf{r}_{i,j}$ if j is odd

$\tilde{\mathbf{r}}_{i,j} = \mathbf{r}_{i,j}$ if j is even

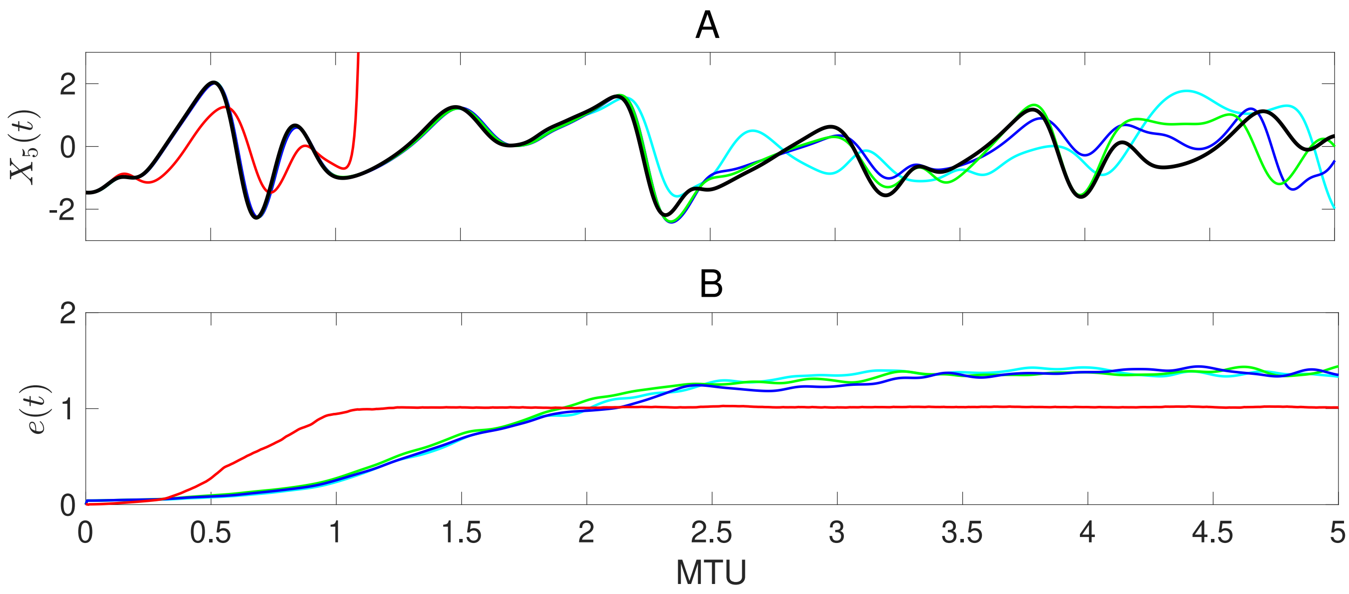


Figure A1. Comparison of RC-ESN forecast skills with nonlinear transformation algorithms T_1, T_2, and T_3 (cyan, green, and blue lines, respectively) and without any transformation (red line). Black line shows the truth. (A) Prediction from one initial condition, (B) Relative L_2-norm error averaged over 100 initial condition $e(t)$ (Eq. (8)). Time is in MTU where 1 MTU $= 200\Delta t \approx 4.5/\lambda_{max}$.

Algorithm T_2

$\tilde{\mathbf{r}}_{i,j} = \mathbf{r}_{i,j-1} \times \mathbf{r}_{i,j-2}$ if $j > 1$ is odd

$\tilde{\mathbf{r}}_{i,j} = \mathbf{r}_{i,j}$ if j is 1 or even

390 **Algorithm** T_3

$\tilde{\mathbf{r}}_{i,j} = \mathbf{r}_{i,j-1} \times \mathbf{r}_{i,j+1}$ if $j > 1$ is odd

$\tilde{\mathbf{r}}_{i,j} = \mathbf{r}_{i,j}$ if j is 1 or even

Figure A1(A) shows an example of short-term predictions from an initial condition using T_1, T_2, T_3, and no transformation, everything else kept the same. It is clear that the nonlinear transformation is essential for skillful predictions, as the prediction obtained without transformation diverges from the truth before 0.25 MTU. The three nonlinear transformation algorithms yield similar results, with accurate predictions for more than 2 MTU. Figure A1(B), which shows the relative prediction error averaged over 100 initial conditions, further confirms this point.

Why the nonlinear transformation is needed, and the best/optimal choice for the transformation (if it exists) should be studied in future work. We highlight that the nonlinear transformation resembles basis function expansion, which is commonly used to capture nonlinearity in linear regression models (Bishop, 2006).

Appendix B: More details on ANN

The ANN used in this study (and the one that yields the best performance) is similar to that of Düeben and Bauer (2018). The ANN has four hidden layers each of which has 100 neurons with a $\tanh$ activation function. The input to the ANN has 8 neurons which takes $(X(t))^{test}$ as the input and outputs $(\Delta X(t))^{test}$ which is also a 8-neuron layer. The obtained $(\Delta X(t))^{test}$ output during test is then used in an Adam-Bashforth integration scheme to obtain . We found that a simpler Euler scheme yields similar results. However, we found that training/testing on pairs of $X(t)$ and $X(t+\Delta t)$ (that was used for RC-ESN) leads to no prediction skill with ANN, and that following the procedure used in Düeben and Bauer (2018) is essential for skillful predictions. We speculate that this might be due to the stateless nature of the ANN. By relating $X(t)$ to the change in $X(t)$ at the next time step, rather than the raw value $X(t+\Delta t)$, this ANN training architecture implicitly contains a reference to the previous time step. While the ANN itself is stateless, this particular training approach essentially encodes a first-order temporal dependence between successive states.

Note that our approach here is the same as the global ANN of Düeben and Bauer (2018). We also tried their local ANN approach, but consistent with their findings for the Lorenz system, found better performance with the global approach (results not reported for brevity).

The ANN is trained with a stochastic gradient descent algorithm with a learning rate of 0.001 with a batch size of 100 and mean absolute error as loss function (mean squared error also gives similar performance).

Appendix C: More details on RNN-LSTM

The governing equations for RNN-LSTM are:

$$g^f(t) = \sigma_f\left(\mathbf{W_f}\left[h(t-1), I(t)\right] + b_f\right), \tag{C1}$$

$$g^i(t) = \sigma_f\left(\mathbf{W_i}\left[h(t-1), I(t)\right] + b_i\right), \tag{C2}$$

$$\tilde{C}(t) = \tanh\left(\mathbf{W_c}\left[h(t-1), I(t)\right] + b_h\right), \tag{C3}$$

$$C(t) = g^f(t)C(t-1) + g^i(t)\tilde{C}(t), \tag{C4}$$

$$g^o(t) = \sigma_h\left(\mathbf{W_h}\left[h(t-1), I(t)\right] + b_h\right), \tag{C5}$$

$$h(t) = g^o(t)\tanh\left(C(t)\right), \tag{C6}$$

$$\Phi(t) = \mathbf{W_{oh}}h(t), \tag{C7}$$

$$X(t+\Delta t) \approx \Phi(t). \tag{C8}$$

σ_f is the softmax activation function; $g^f(t)$, $g^i(t)$, and $g^o(t) \in \Re^{d_h \times (d_h+d_i)}$ are the forget gate, input gate, and output gate respectively. d_h is the dimension of the hidden layers (chosen as 50 in our study) and d_i is the dimension of the input ($8 \times q$). b_f, b_i, and b_h are the biases in the forget gate, input gate, and the hidden layers. $I(t) \in \Re^{d_i}$ is the input, which is a column of a time-delay-embedded matrix of $X(t)$. This matrix has the dimension of $(8 \times q) \times N$. $h(t) \in \Re^{d_h}$ is the hidden state and $C(t) \in \Re^{d_h}$ is the cell state (the states track the temporal evolution). The weights $\mathbf{W_o}$, $\mathbf{W_i}$, $\mathbf{W_f}$, $\mathbf{W_c}$, and $\mathbf{W_{oh}}$ are learned through the BPTT algorithm (Goodfellow et al., 2016) using an ADAM optimizer (Kingma and Ba, 2014). $\Phi(t)$ is the output from the RNN-LSTM.

The LSTM used in this study is a stateless LSTM, wherein the two hidden states (C and h) are refreshed at the beginning of each batch during training (this is not the same as the stateless ANN, where there is no hidden state at all). Here, the training sequences in each batch are shuffled randomly, leading to an unbiased gradient estimator in the stochastic gradient descent algorithm (Meng et al., 2019).

Code and data availability. All data and codes can be accessed at https://github.com/ashesh6810/RCESN_spatio_temporal.

Author contributions. All authors developed the idea. P.H. designed and supervised the study. A.C. with help from D.S. developed the deep learning codes. All authors discussed the results and revised the manuscript.

Competing interests. We declare we have no competing interests.

Acknowledgements. This work was supported by NASA grant 80NSSC17K0266 and an Early-Career Research Fellowship from the Gulf Research Program of the National Academies of Science, Engineering, and Medicine (to P.H.). A.C. thanks the Rice University Ken Kennedy Institute for a BP HPC Graduate Fellowship. Computational resources on Bridge Large, Bridge GPU, and Stampede2 clusters were provided by the NSF XSEDE allocation ATM170020. We are grateful to Krishna Palem for many stimulating discussions and helpful comments. We thank Michelle Girvan, Reza Malek-Madani, Rohan Mukherjee, Guido Novati, and Pantelis Vlachas for insightful discussions, and Jaideep Pathak for sharing his Matlab ESN code for the Kuramoto-Sivashinsky system. We are grateful to Mingchao Jiang and Adam Subel for help with running some of the simulations.

References

Andersen, J. and Kuang, Z.: Moist static energy budget of MJO-like disturbances in the atmosphere of a zonally symmetric aquaplanet, Journal of Climate, 25, 2782–2804, 2012.

Arbabi, H. and Mezic, I.: Ergodic theory, dynamic mode decomposition, and computation of spectral properties of the Koopman operator, SIAM Journal on Applied Dynamical Systems, 16, 2096–2126, 2017.

Bauer, P., Thorpe, A., and Brunet, G.: The quiet revolution of numerical weather prediction, Nature, 525, 47, 2015.

Benedict, J. and Randall, D.: Structure of the Madden–Julian oscillation in the superparameterized CAM, Journal of the Atmospheric Sciences, 66, 3277–3296, 2009.

Bishop, C.: Pattern recognition and machine learning, springer, 2006.

Bolton, T. and Zanna, L.: Applications of deep learning to ocean data inference and subgrid parameterization, Journal of Advances in Modeling Earth Systems, 11, 376–399, 2019.

Brenowitz, N. and Bretherton, C.: Prognostic validation of a neural network unified physics parameterization, Geophysical Research Letters, 45, 6289–6298, 2018.

Brunton, S. and Kutz, J.: Data-driven Science and Engineering: Machine Learning, Dynamical Systems, and Control, Cambridge University Press, 2019.

Carbonneau, R., Laframboise, K., and Vahidov, R.: Application of machine learning techniques for supply chain demand forecasting, European Journal of Operational Research, 184, 1140–1154, 2008.

Chantry, M., Thornes, T., Palmer, T., and Düben, P.: Scale-selective precision for weather and climate forecasting, Monthly Weather Review, 147, 645–655, 2019.

Chattopadhyay, A., Hassanzadeh, P., and Pasha, S.: Predicting clustered weather patterns: A test case for applications of convolutional neural networks to spatio-temporal climate data, Scientific Reports, 10, 1–13, 2020a.

Chattopadhyay, A., Nabizadeh, E., and Hassanzadeh, P.: Analog forecasting of extreme-causing weather patterns using deep learning, Journal of Advances in Modeling Earth Systems, 12, 2020b.

Chattopadhyay, A., Subel, A., and Hassanzadeh, P.: Data-driven super-parameterization using deep learning: Experimentation with multi-scale Lorenz 96 systems and transfer-learning, arXiv preprint arXiv:2002.11167, 2020c.

Chen, K., Zhou, Y., and Dai, F.: A LSTM-based method for stock returns prediction: A case study of China stock market, in: 2015 IEEE International Conference on Big Data, pp. 2823–2824, IEEE, 2015.

Cho, K., V.M, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y.: Learning phrase representations using RNN encoder-decoder for statistical machine translation, arXiv preprint arXiv:1406.1078, 2014.

Collins, W., Rasch, P., Boville, B., Hack, J., McCaa, J., Williamson, D., Briegleb, B., Bitz, C., Lin, S., and Zhang, M.: The formulation and atmospheric simulation of the Community Atmosphere Model version 3 (CAM3), Journal of Climate, 19, 2144–2161, 2006.

Collins, W. et al.: Development and evaluation of an Earth-System model–HadGEM2, Geoscientific Model Development, 4, 1051–1075, 2011.

Düben, P. and Palmer, T.: Benchmark tests for numerical weather forecasts on inexact hardware, Monthly Weather Review, 142, 3809–3829, 2014.

Düben, P., Joven, J., Lingamneni, A., McNamara, H., De Micheli, G., Palem, K., and Palmer, T.: On the use of inexact, pruned hardware in atmospheric modelling, Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 372, 20130 276, 2014.

Düben, P., Yenugula, S., Augustine, J., Palem, K., Schlachter, J., Enz, C., and Palmer, T.: Opportunities for energy efficient computing: A study of inexact general purpose processors for high-performance and big-data applications, in: 2015 Design, Automation & Test in Europe Conference & Exhibition (DATE), pp. 764–769, IEEE, 2015.

Düeben, P. and Bauer, P.: Challenges and design choices for global weather and climate models based on machine learning, Geoscientific Model Development, 11, 3999–4009, 2018.

Duraisamy, K., Iaccarino, G., and Xiao, H.: Turbulence modeling in the age of data, Annual Review of Fluid Mechanics, 51, 357–377, 2019.

Epanechnikov, V.: Non-parametric estimation of a multivariate probability density, Theory of Probability & Its Applications, 14, 153–158, 1969.

Fan, H., Jiang, J., Zhang, C., Wang, X., and Lai, Y.-C.: Long-term prediction of chaotic systems with machine learning, Physical Review Research, 2, 012 080, 2020.

Flato, G.: Earth system models: an overview, Wiley Interdisciplinary Reviews: Climate Change, 2, 783–800, 2011.

Gagne, I., John, D., Christensen, H. M., Subramanian, A. C., and Monahan, A. H.: Machine Learning for Stochastic Parameterization: Generative Adversarial Networks in the Lorenz'96 Model, Journal of Advances in Modeling Earth Systems (in press), https://doi.org/10.1029/2019MS001896, 2020.

Garcia, R., Smith, A., Kinnison, D., Cámara, Á. l., and Murphy, D.: Modification of the gravity wave parameterization in the Whole Atmosphere Community Climate Model: Motivation and results, Journal of the Atmospheric Sciences, 74, 275–291, 2017.

Gauthier, D.: Reservoir computing: Harnessing a universal dynamical system, SIAM News, 51, 12, 2018.

Gentine, P., Pritchard, M., Rasp, S., Reinaudi, G., and Yacalis, G.: Could machine learning break the convection parameterization deadlock?, Geophysical Research Letters, 45, 5742–5751, 2018.

Giannakis, D., Ourmazd, A., Slawinska, J., and Zhao, Z.: Spatiotemporal pattern extraction by spectral analysis of vector-valued observables, arXiv preprint arXiv:1711.02798, 2017.

Goodfellow, I., Bengio, Y., and Courville, A.: Deep learning, MIT press, 2016.

Graves, A., Mohamed, A., and Hinton, G.: Speech recognition with deep recurrent neural networks, in: 2013 IEEE international Conference on Acoustics, Speech and Signal Processing, pp. 6645–6649, IEEE, 2013.

Hatfield, S., Subramanian, A., Palmer, T., and Düben, P.: Improving weather forecast skill through reduced-precision data assimilation, Monthly Weather Review, 146, 49–62, 2018.

Hochreiter, S. and Schmidhuber, J.: Long short-term memory, Neural Computation, 9, 1735–1780, 1997.

Hourdin, F., M., T., G., A., Golaz, J., Balaji, V., Duan, Q., Folini, D., Ji, D., Klocke, D., Qian, Y., et al.: The art and science of climate model tuning, Bulletin of the American Meteorological Society, 98, 589–602, 2017.

Jaeger, H.: Echo state network, Scholarpedia, 2, 2330, https://doi.org/10.4249/scholarpedia.2330, revision #188245, 2007.

Jaeger, H. and Haas, H.: Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication, Science, 304, 78–80, 2004.

Jeevanjee, N., Hassanzadeh, P., Hill, S., and Sheshadri, A.: A perspective on climate model hierarchies, Journal of Advances in Modeling Earth Systems, 9, 1760–1771, 2017.

Khairoutdinov, M. and Randall, D.: A cloud resolving model as a cloud parameterization in the NCAR Community Climate System Model: Preliminary results, Geophysical Research Letters, 28, 3617–3620, 2001.

Khodkar, M. and Hassanzadeh, P.: Data-driven reduced modelling of turbulent Rayleigh–Bénard convection using DMD-enhanced fluctuation–dissipation theorem, Journal of Fluid Mechanics, 852, 2018.

Khodkar, M., Hassanzadeh, P., Nabi, S., and Grover, P.: Reduced-order modeling of fully turbulent buoyancy-driven flows using the Green's function method, Physical Review Fluids, 4, 013 801, 2019.

Kim, H., Eykholt, R., and Salas, J.: Nonlinear dynamics, delay times, and embedding windows, Physica D: Nonlinear Phenomena, 127, 48–60, 1999.

Kingma, D. and Ba, J.: Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980, 2014.

Kooperman, G., Pritchard, M., O'Brien, T., and Timmermans, B.: Rainfall From Resolved Rather Than Parameterized Processes Better Represents the Present-Day and Climate Change Response of Moderate Rates in the Community Atmosphere Model, Journal of Advances in Modeling Earth Systems, 10, 971–988, 2018.

Kutz, J.: Deep learning in fluid dynamics, Journal of Fluid Mechanics, 814, 1–4, 2017.

Leyffer, S., Wild, S., Fagan, M., Snir, M., Palem, K., Yoshii, K., and Finkel, H.: Doing Moore with Less–Leapfrogging Moore's Law with Inexactness for Supercomputing, arXiv preprint arXiv:1610.02606, 2016.

Lim, S. H., Giorgini, L. T., Moon, W., and Wettlaufer, J.: Predicting Rare Events in Multiscale Dynamical Systems using Machine Learning, arXiv preprint arXiv:1908.03771, 2019.

Ling, J., Kurzawski, A., and Templeton, J.: Reynolds averaged turbulence modelling using deep neural networks with embedded invariance, Journal of Fluid Mechanics, 807, 155–166, 2016.

Lorenz, E.: Predictability: A problem partly solved, in: Predcitibility of Weather and Climate, vol. 1, pp. 40–58, 1996.

Lu, Z., Pathak, J., Hunt, B., Girvan, M., Brockett, R., and Ott, E.: Reservoir observers: Model-free inference of unmeasured variables in chaotic systems, Chaos, 27, 041 102, 2017.

Lu, Z., Hunt, B., and Ott, E.: Attractor reconstruction by machine learning, Chaos, 28, 061 104, 2018.

Lukoševičius, M. and Jaeger, H.: Reservoir computing approaches to recurrent neural network training, Computer Science Review, 3, 127–149, 2009.

Ma, Q., Shen, L., Chen, E., Tian, S., Wang, J., and Cottrell, G.: WALKING WALKing walking: Action Recognition from Action Echoes., in: Internationa Joint Conference on Artificial Intelligence, pp. 2457–2463, 2017.

McDermott, P. and Wikle, C.: An ensemble quadratic echo state network for non-linear spatio-temporal forecasting, Stat, 6, 315–330, 2017.

McDermott, P. and Wikle, C.: Deep echo state networks with uncertainty quantification for spatio-temporal forecasting, Environmetrics, 30, e2553, 2019.

Meng, Q., Chen, W., Wang, Y., Ma, Z.-M., and Liu, T.-Y.: Convergence analysis of distributed stochastic gradient descent with shuffling, Neurocomputing, 337, 46–57, 2019.

Mezić, I.: Spectral properties of dynamical systems, model reduction and decompositions, Nonlinear Dynamics, 41, 309–325, 2005.

Mohan, A., Daniel, D., Chertkov, M., and Livescu, D.: Compressed Convolutional LSTM: An Efficient Deep Learning framework to Model High Fidelity 3D Turbulence, arXiv preprint arXiv:1903.00033, 2019.

Moosavi, A., Attia, A., and Sandu, A.: A machine learning approach to adaptive covariance localization, arXiv preprint arXiv:1801.00548, 2018.

O'Gorman, P. and Dwyer, J.: Using Machine Learning to Parameterize Moist Convection: Potential for Modeling of Climate, Climate Change, and Extreme Events, Journal of Advances in Modeling Earth Systems, 10, 2548–2563, 2018.

Palem, K.: Inexactness and a future of computing, Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 372, 20130 281, 2014.

Palmer, T.: Climate forecasting: Build high-resolution global climate models, Nature News, 515, 338, 2014.

Pascanu, R., Mikolov, T., and Bengio, Y.: On the difficulty of training recurrent neural networks, in: International Conference on Machine Learning, pp. 1310–1318, 2013.

Pathak, J., Lu, Z., Hunt, B., Girvan, M., and Ott, E.: Using machine learning to replicate chaotic attractors and calculate Lyapunov exponents from data, Chaos, 27, 121 102, 2017.

Pathak, J., Hunt, B., Girvan, M., Lu, Z., and Ott, E.: Model-free prediction of large spatiotemporally chaotic systems from data: A reservoir computing approach, Physical Review Letters, 120, 024 102, 2018a.

Pathak, J., Wikner, A., Fussell, R., Chandra, S., Hunt, B., Girvan, M., and Ott, E.: Hybrid forecasting of chaotic processes: Using machine learning in conjunction with a knowledge-based model, Chaos, 28, 041 101, 2018b.

Raissi, M., Perdikaris, P., and Karniadakis, G.: Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, Journal of Computational Physics, 378, 686–707, 2019.

Rasp, S., Pritchard, M., and Gentine, P.: Deep learning to represent subgrid processes in climate models, Proceedings of the National Academy of Sciences, 115, 9684–9689, 2018.

Reichstein, M. et al.: Deep learning and process understanding for data-driven Earth system science, Nature, 566, 195, 2019.

Rudy, S., Kutz, J., and Brunton, S.: Deep learning of dynamics and signal-noise decomposition with time-stepping constraints, arXiv preprint arXiv:1808.02578, 2018.

Salehipour, H. and Peltier, W.: Deep learning of mixing by two atoms of stratified turbulence, Journal of Fluid Mechanics, 861, 2019.

Scher, S. and Messori, G.: Generalization properties of feed-forward neural networks trained on Lorenz systems, Nonlinear Processes in Geophysics, 26, 381–399, 2019.

Schneider, T., Lan, S., Stuart, A., and Teixeira, J.: Earth system modeling 2.0: A blueprint for models that learn from observations and targeted high-resolution simulations, Geophysical Research Letters, 44, 2017a.

Schneider, T., Teixeira, J., Bretherton, C., Brient, F., Pressel, K., Schär, C., and Siebesma, A.: Climate goals and computing the future of clouds, Nature Climate Change, 7, 3, 2017b.

Siegelmann, H. and Sontag, E.: On the computational power of neural nets, in: Proceedings of the fifth annual workshop on Computational learning theory, pp. 440–449, ACM, 1992.

Stevens, B. and Bony, S.: What are climate models missing?, Science, 340, 1053–1054, 2013.

Sutskever, I., Vinyals, O., and Le, Q.: Sequence to sequence learning with neural networks, in: Advances in Neural Information Processing Systems, pp. 3104–3112, 2014.

Thornes, T., Düben, P., and Palmer, T.: On the use of scale-dependent precision in Earth System modelling, Quarterly Journal of the Royal Meteorological Society, 143, 897–908, 2017.

Toms, B. A., Kashinath, K., Yang, D., et al.: Deep Learning for Scientific Inference from Geophysical Data: The Madden–Julian oscillation as a Test Case, arXiv preprint arXiv:1902.04621, 2019.

Tu, J., Rowley, C., Luchtenburg, D., Brunton, S., and Kutz, J.: On dynamic mode decomposition: Theory and applications, Journal of Computational Dynamics, 1, 391–421, 2014.

Vlachas, P., Byeon, W., Wan, Z., Sapsis, T., and Koumoutsakos, P.: Data-driven forecasting of high-dimensional chaotic systems with long short-term memory networks, Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences, 474, 20170 844, 2018.

Watson, P.: Applying machine learning to improve simulations of a chaotic dynamical system using empirical error correction, Journal of Advances in Modeling Earth Systems, 2019.

Williams, M., Kevrekidis, I., and Rowley, C.: A data–driven approximation of the Koopman operator: Extending dynamic mode decomposition, Journal of Nonlinear Science, 25, 1307–1346, 2015.

Wu, J. et al.: Enforcing Statistical Constraints in Generative Adversarial Networks for Modeling Chaotic Dynamical Systems, arXiv preprint arXiv:1905.06841, 2019.

Xingjian, S., Chen, Z., Wang, H., Yeung, D., Wong, W., and Woo, W.: Convolutional LSTM network: A machine learning approach for precipitation nowcasting, in: Advances in Neural Information Processing Systems, pp. 802–810, 2015.

Yildiz, I., Jaeger, H., and Kiebel, S.: Re-visiting the echo state property, Neural Networks, 35, 1–9, 2012.

Yu, R., Zheng, S., Anandkumar, A., and Yue, Y.: Long-term forecasting using tensor-train RNNs, arXiv preprint arXiv:1711.00073, 2017.

Zhu, Y., Zabaras, N., Koutsourelakis, P., and Perdikaris, P.: Physics-Constrained Deep Learning for High-dimensional Surrogate Modeling and Uncertainty Quantification without Labeled Data, arXiv preprint arXiv:1901.06314, 2019.

Zimmermann, R. and Parlitz, U.: Observing spatio-temporal dynamics of excitable media using reservoir computing, Chaos, 28, 043 118, 2018.