Combining machine learning with knowledge-based modeling for scalable forecasting and subgrid-scale closure of large, complex, spatiotemporal systems

Cite as: Chaos **30**, 053111 (2020); doi: 10.1063/5.0005541
Submitted: 21 February 2020 · Accepted: 17 April 2020 ·
Published Online: 7 May 2020

Alexander Wikner,[1,a)] Jaideep Pathak,[1] Brian Hunt,[2,3] Michelle Girvan,[1,2] Troy Arcomano,[4] Istvan Szunyogh,[4] Andrew Pomerance,[5] and Edward Ott[1,6]

AFFILIATIONS

[1]Department of Physics and Institute for Research in Electronics and Applied Physics, University of Maryland, College Park, Maryland 20740, USA
[2]Institute for Physical Science and Technology, University of Maryland, College Park, Maryland 20740, USA
[3]Department of Mathematics, University of Maryland, College Park, Maryland 20740, USA
[4]Department of Atmospheric Sciences, Texas A&M University, College Station, Texas 77843, USA
[5]Potomac Research LLC, Alexandria, Virginia 22311, USA
[6]Department of Electrical and Computer Engineering, University of Maryland, College Park, Maryland 20740, USA

Note: This paper is part of the Focus Issue, "When Machine Learning Meets Complex Systems: Networks, Chaos and Nonlinear Dynamics."
[a)]**Author to whom correspondence should be addressed:** awikner1@umd.edu

ABSTRACT

We consider the commonly encountered situation (e.g., in weather forecast) where the goal is to predict the time evolution of a large, spatiotemporally chaotic dynamical system when we have access to both time series data of previous system states and an imperfect model of the full system dynamics. Specifically, we attempt to utilize machine learning as the essential tool for integrating the use of past data into predictions. In order to facilitate scalability to the common scenario of interest where the spatiotemporally chaotic system is very large and complex, we propose combining two approaches: (i) a parallel machine learning prediction scheme and (ii) a hybrid technique for a composite prediction system composed of a knowledge-based component and a machine learning-based component. We demonstrate that not only can this method combining (i) and (ii) be scaled to give excellent performance for very large systems but also that the length of time series data needed to train our multiple, parallel machine learning components is dramatically less than that necessary without parallelization. Furthermore, considering cases where computational realization of the knowledge-based component does not resolve subgrid-scale processes, our scheme is able to use training data to incorporate the effect of the unresolved short-scale dynamics upon the resolved longer-scale dynamics (subgrid-scale closure).

We consider the general problem of predicting the future time evolution of a large, complex spatial system when we have access to information coming from two sources: (i) measurements of what has happened in the past over some span of time and (ii) more general scientific knowledge, typically in the form of a numerical implementation of a mathematical system description, which is imperfect because of lack of knowledge regarding some aspects of the system processes or because of practical numerical restrictions like limited spatial resolution. Among many important examples of the general type of problem, what we have in mind is that of weather prediction, where the system involves details of complex geographical features (like mountains, oceans, etc.), as well as complex atmospheric structures (like clouds, convective vortices, etc.). In this paper, we ask if it is possible to

use machine learning to combine knowledge from both sources (i) and (ii) to produce better predictions in a way that is feasible for very large systems. We describe a machine learning technique, Combined Hybrid-Parallel Prediction (CHyPP), that can potentially be applied to this situation and demonstrate its efficacy on test systems. Our proposed machine learning technique makes use of a version of "reservoir computing," a method for training recurrent artificial neural networks that is effective at predicting complex evolving dynamics, which can be trained and applied using parallel computing systems. We combine this technique with a predictor formulated using imperfect knowledge to produce more accurate forecasts. In the future, we hope that our technique will prove effective for a large class of important problems.

I. INTRODUCTION

In recent years, machine learning techniques have been used to solve a number of complex modeling problems ranging from effective translation between hundreds of different human languages[1] to predicting the bioactivity of small molecules for drug discovery.[2] Typically, the most impressive results have been obtained using artificial neural networks with many hidden neural states.[3] These hidden layers form a "black box" model, where internal parameters are trained given a set of measured training data, but after which only the final model output is observed. This formulation using measured training data contrasts with how models used for forecasting physical spatiotemporally chaotic processes are formulated, which is typically done using the available scientific knowledge of the underlying mechanisms that govern the system's evolution. For example, in the case of forecasting weather, this knowledge includes the Navier–Stokes equations, the first law of thermodynamics, the ideal gas law, and simplified representations of physics at the unresolved spatial scales (see Sec. IV).[4]

In this paper, focusing on the key issues of scalability and unresolved subgrid physics, we consider the general problem of forecasting a very large and complex spatiotemporally chaotic system, where we have access to both past time series of measurements of the system state evolution and to an imperfect knowledge-based prediction model. We present a method for combining machine learning prediction with imperfect knowledge-based forecasting that is scalable to large systems with the aim that the resulting combined prediction system can be significantly more accurate and efficient than either a pure knowledge-based prediction or a pure machine learning-based prediction. A main source of difficulty for scalability of the machine learning is that the dimension of the state of the systems we are interested in can be extremely large. For example, in state-of-the-art global numerical weather models, the state dimension (number of variables at all grid points) can be on the order of 10^9. Thus, both the machine learning input (the current atmospheric state) and output (the predicted atmospheric state) have this dimensionality. (In contrast to the description of some machine learning techniques as "deep," one might refer to the situations we address as "wide.") The prediction method that we propose for such large complex systems builds on the previous work on parallelizable machine learning prediction[5] and hybridization of knowledge-based modeling with machine learning.[6] We call our technique Combined Hybrid/Parallel Prediction (CHyPP, pronounced "chip"). Although the general method we propose is applicable to different kinds of machine learning, the numerical examples presented in this paper use a machine learning method known as reservoir computing.[7,8] Jaeger and Haas[9] described the effectiveness of reservoir computing for predicting low-dimensional chaotic systems. Research surrounding this technique has since expanded,[10,11] and it has recently been shown that reservoir computing using recurrent neural networks can produce similar quality predictions for chaotic systems to those of other recurrent architectures, such as Long Short-Term Memory (LSTM)[12] and Gated Recurrent Units (GRU),[13] while often requiring much less computational time to train.[14] Reservoir computing techniques can additionally be extended to physical implementations using, e.g., photonics[15] and Field Programable Gate Arrays (FPGAs).[16,17]

The rest of the paper is organized as follows. In Sec. II, we first review a simple version of reservoir computing[7,8] and discuss its shortcomings for forecasting high-dimensional spatiotemporal chaos. We next describe the hybrid reservoir prediction technique (Refs. 6 and 18), as well as previous work on how machine learning can be parallelized for prediction of spatiotemporal systems.[5] We then present our proposed CHyPP architecture combining the two. In Sec. III, we demonstrate how the CHyPP methodology improves on each of the component prediction methods. For these demonstrations, we use the paradigmatic example of the Kuramoto–Sivashinky model as our test model of the spatiotemporally chaotic system that we aim to predict. We highlight the scalability of the proposed method to very large systems as well as its efficient use of training data, which we view as the crucial issues for the general class of applications in which we are interested. In Sec. IV, we consider a situation with multiple time and space scales and show by numerical simulation tests that CHyPP can, through its use of data, effectively account for unknown subgrid-scale processes. The main conclusion of this paper is that our CHyPP methodology provides an extremely promising framework, potentially facilitating significant advances in the forecasting of large, complex, spatiotemporally chaotic systems. We believe that, in addition to weather, the method that we propose may potentially be applicable to a host of important areas, enabling currently unattainable capabilities. Some speculative examples of potential applications are forecasting of ocean conditions, forecasting conditions in the solar wind, magnetosphere, and ionosphere (also known as "space weather," important for its impact on the Earth-orbiting satellites, Global Positioning System (GPS) accuracy, and high frequency communications), forecasting the evolution of forest fires and their response to mitigating interventions, forecasting the responses of ecological systems to climate change, analysis of neuronal activity, etc.

II. RESERVOIR COMPUTING ARCHITECTURE FOR CHyPP

A. A simple machine learning predictor

To begin, we initially consider the goal of a generic machine learning system for time series prediction of an unknown dynamical system evolving on an attractor of that system. Later, we will consider that the machine learning system is a reservoir computer

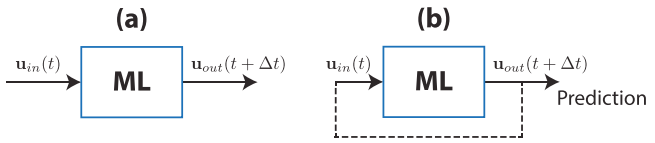

FIG. 1. Machine learning prediction device (a) open-loop training phase and (b) closed-loop prediction phase.

and that the unknown chaotic system is not completely unknown, and we will try to make use of that knowledge. Given a finite duration time series of the unknown system state's evolution up to a certain time t_0, where the state at each time is represented by the K-dimensional vector $\mathbf{u}(t) = [u_1(t), u_2(t), \ldots, u_K(t)]^T$, our goal is to predict the subsequent evolution of the state. As illustrated in Fig. 1(a), in the initial training phase, at each time $t = n\Delta t \le t_0$, $\mathbf{u}(t)$ is input to the machine learning system ($\mathbf{u}_{in}(t) = \mathbf{u}(t)$), which is trained to output a time Δt prediction of the dynamical system state $\mathbf{u}(t+\Delta t)$ ($\mathbf{u}_{out}(t+\Delta t) \simeq \mathbf{u}(t+\Delta t)$). We refer to the just-described input–output configuration as the "open-loop" system [Fig. 1(a)]. To ensure an accurate representation of the true dynamics with a reservoir of limited size, Δt is typically short compared to natural time scales (such as the correlation time or the "Lyapunov time") present in the unknown dynamical system whose state is to be predicted. Once trained, the machine learning system can be run in a "closed-loop" feedback configuration [Fig. 1(b)] to autonomously generate predictions over a finite duration of time. That is, with t_0 representing the time at the end of the training data, we replace the former input from the training data by the output by inserting an output-to-input feedback loop, shown by the dashed line in Fig. 1(b). Then, when $\mathbf{u}_{in}(t_0) = \mathbf{u}(t_0)$ is the input, the reservoir computer produces an output prediction for $\mathbf{u}(t_0 + \Delta t)$, which we refer to as $\tilde{\mathbf{u}}(t_0 + \Delta t) = \mathbf{u}_{out}(t_0 + \Delta t)$. When this predicted state is then used as the input ($\mathbf{u}_{in}(t_0 + \Delta t) = \tilde{\mathbf{u}}(t_0 + \Delta t)$), the reservoir computer produces an output prediction for $\mathbf{u}(t_0 + 2\Delta t)$, denoted as $\tilde{\mathbf{u}}(t_0 + 2\Delta t)$ ($\mathbf{u}_{out}(t_0 + 2\Delta t) = \tilde{\mathbf{u}}(t_0 + 2\Delta t)$). This process is then iterated to produce predictions of the system state at $t = t_0 + m\Delta t$ for $m = 1, 2, 3, \ldots$ [Fig. 1(b)].

In the rest of this section, we first present background from previous work (Secs. II B and II D), then introduce our CHyPP method for combining a knowledge-based model with reservoir-based machine learning to form a scalable system concept suitable for state prediction of very large, spatiotemporally chaotic systems. Specifically, in Sec. II B, we review a basic reservoir computing setup based on the methods of Refs. 7 and 8 along with the proposal for its use as a predictor carried out in Ref. 9. In Sec. II C, we build upon the simple setup of Sec. II B and describe the methodology from Ref. 5 for hybrid forecasting of the dynamical system using a single reservoir computing network and an imperfect model. In Sec. II D, the reservoir computing forecasting technique of Sec. II B is extended via parallelization of the machine learning with multiple parallel reservoir computers, in order to predict high-dimensional spatiotemporally chaotic systems, as was first described in Ref. 5 (but without the incorporation of a knowledge-based model). Finally, in Sec. II E, we present our proposed CHyPP architecture and technique for combining the parallel reservoir method of Sec. II D with the hybridization of a knowledge-based predictor and a parallel reservoir-based machine learning prediction of Sec. II C. It is our belief that it is only by means of such a combination that the most effective application of machine learning-enabled prediction can be realized for large, complex, spatiotemporally chaotic dynamical systems.

B. Basic reservoir computing

We now consider that the ML device shown in Fig. 1 is a reservoir computer which, as shown in Fig. 2 (and further discussed subsequently), consists of a linear input coupler ($\mathbf{W}_{in}$) that couples the state $\mathbf{u}_{in}(t)$ into the reservoir (the circle in Fig. 2). The state of the reservoir is given by a high-dimensional vector $\mathbf{r}(t)$, which is then linearly coupled by $\mathbf{W}_{out}$ to produce an output vector $\mathbf{u}_{out}(t+\Delta t)$ which, through the training, is a very good approximation to $\mathbf{u}(t+\Delta t)$. In this paper, our implementation of the reservoir is an artificial recurrent neural network with a large number of nodes. The artificial neural network that forms our basis of the reservoir computing implementation is illustrated in Fig. 2. The reservoir network adjacency matrix is chosen to be a randomly generated, sparse matrix $\mathbf{A}$ that represents a directed graph with weighted edges. The adjacency matrix $\mathbf{A}$ has dimensions $D_r \times D_r$, where D_r is the number of nodes in the network. Elements of $\mathbf{A}$ are randomly generated such that the average number of incident connections per node (average number of nonzero elements of the matrix in each row) is set to a chosen value $\langle d \rangle$, the "average in-degree," while the nonzero elements of $\mathbf{A}$ are chosen from a uniform distribution over the interval $[-1, 1]$. Once generated, $\mathbf{A}$ is re-scaled (i.e., multiplied by a scalar) so that the its largest absolute eigenvalue is a prescribed value ρ, called the spectral radius. Each node i in the network has an associated scalar state $r_i(t)$. The state of the network is represented by the D_r dimensional vector $\mathbf{r}(t)$, whose elements are $r_i(t)$, where $i = 1, 2, 3, \ldots, D_r$.

The reservoir network state $\mathbf{r}(t)$ evolves dynamically while receiving input through a $K \times D_r$ input coupling matrix, $\mathbf{W}_{in}$. We choose the matrix $\mathbf{W}_{in}$ to contain an equal number of nonzero elements in each column, which corresponds to coupling each

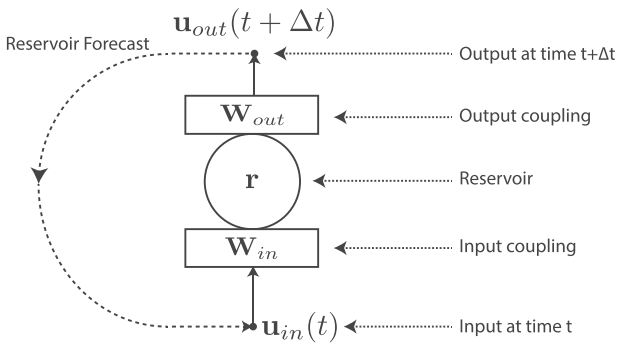

FIG. 2. Diagram of the reservoir computer setup. In the "open-loop" training phase [analogous to Fig. 1(a)], the dashed line representing coupling from the output back to the input is absent. In the "closed-loop" prediction phase [analogous to Fig. 1(b)], the coupling from the output back to the input (dashed line) is activated.

element of the reservoir input to an equal number of reservoir node states. Nonzero elements of this matrix are selected randomly and uniformly from the interval $[-\sigma, \sigma]$, where σ is referred to as the input weight. Given the current state of the reservoir $\mathbf{r}(t)$, the reservoir state is advanced at each time using a hyperbolic tangent activation function,

$$\mathbf{r}(t+\Delta t) = \tanh[\mathbf{A}\mathbf{r}(t) + \mathbf{W}_{in}\mathbf{u}_{in}(t)]. \quad (1)$$

Before prediction begins, the reservoir computer is trained in the "open-loop" configuration. During this training phase, $\mathbf{u}_{in}(t) = \mathbf{u}(t) + s\boldsymbol{\eta}(t)$. Here, $\mathbf{u}(t)$ is the measurement of dynamical system state at time t in the form of a K-dimensional vector. As in Ref. 7, we add a small, normally distributed K-dimensional vector $s\boldsymbol{\eta}(t)$ of mean 0 and standard deviation s to the input dynamical system state during training. The elements of the vector $\boldsymbol{\eta}(t)$ are chosen randomly and independently at each time t. The function of this added "stabilization noise" is to allow the reservoir computer to learn how to return to the true trajectory when the input trajectory has been perturbed away from it. We find that, in many cases, this additional small noise input beneficially promotes stability of the closed-loop prediction configuration once training has been completed.

The adjustable constants characterizing the overall prediction system (e.g., D_r, $\langle d\rangle$, σ, s, and Δt) are referred to as "hyperparameters," and it is important that they be chosen carefully in order for the reservoir computer to predict accurately. For example, as explained in the previous literature, the hyperparameters can be chosen so that the reservoir system has the so-called "echo-state property" (see, e.g., Ref. 7), whereby when in the open-loop training phase, the reservoir state $\mathbf{r}(t)$, aside from an initial transient and with the random sequence $\boldsymbol{\eta}(t)$ fixed, the reservoir state $\mathbf{r}(t)$ becomes uniquely determined by the reservoir input sequence $\mathbf{u}(t)$ (and hence independent of the initial values of $\mathbf{r}$). Accordingly, prior to initiation of the training, we ignore and discard the reservoir and input states for the first few time steps. The state of the reservoir at the end of this transient nullification period is labeled $\mathbf{r}(0)$. Starting with $\mathbf{r}(0)$, the training system states $\mathbf{u}(j\Delta t)$ (j an integer, $j\Delta t \le t_0$) and the resulting reservoir states, $\mathbf{r}((j+1)\Delta t)$, are recorded and saved. We then desire to use these saved states to produce an output, $\tilde{\mathbf{u}}(t+\Delta t)$, when $\mathbf{u}(t)$ is the input, which we desire to be very close to $\mathbf{u}(t+\Delta t)$. To do this, we find it useful to perform an *ad hoc* operation on the reservoir state vectors that squares the value of half of the node states. Specifically, we define $\tilde{\mathbf{r}}(j\Delta t)$ such that

$$\tilde{r}_i = r_i \quad \text{for } i \text{ odd}, \quad (2)$$

$$\tilde{r}_i = r_i^2 \quad \text{for } i \text{ even}. \quad (3)$$

As surmised in footnote 16 of Ref. 5, this operation improves prediction by breaking a particular odd symmetry of the reservoir dynamics that is not generally present in the dynamics to be predicted. We next couple the transformed reservoir state $\tilde{\mathbf{r}}(t+\Delta t)$ via a $K \times D_r$ output coupling matrix $\mathbf{W}_{out}$ to produce an output $\mathbf{u}_{out}(t+\Delta t)$,

$$\mathbf{u}_{out}(t+\Delta t) = \mathbf{W}_{out}\tilde{\mathbf{r}}(t+\Delta t), \quad (4)$$

and we endeavor to choose (train) the matrix elements of $\mathbf{W}_{out}$ so that $\mathbf{u}_{out}(t+\Delta t)$ is close to $\mathbf{u}(t+\Delta t)$. In general, this will require that $D_r \gg K$. To accomplish this, we try to minimize the L^2 difference between $\mathbf{u}(t+\Delta t)$ and $\mathbf{u}_{out}(t+\Delta t)$. To prevent overfitting, we insert a Tikhonov regularization term to penalize very large values of the matrix elements of $\mathbf{W}_{out}$,[19] that is, we find

$$\min_{\mathbf{W}_{out}} \left\{ \sum_{0\le t<t_0} \left[\|\mathbf{W}_{out}\tilde{\mathbf{r}}(t) - \mathbf{u}(t)\|^2 \right] + \beta \mathrm{Trace}(\mathbf{W}_{out}\mathbf{W}_{out}^T) \right\}, \quad (5)$$

over the KD_r scalar values of the matrix $\mathbf{W}_{out}$. Here, β is a small regularization parameter and $\|\cdots\|$ denotes the L^2 norm. This technique is commonly known as ridge regression. In our subsequent numerical experiments (Secs. III and IV), we use the "matrix solution" for the minimization problem to determine the trained $\mathbf{W}_{out}$. In particular, we proceed as follows. We first form a matrix $\tilde{\mathbf{R}}$ where the jth column is the jth transformed reservoir state $\tilde{\mathbf{r}}(j\Delta t)$. We define a target matrix $\mathbf{U}$ consisting of the time series of training data such that the jth column of $\mathbf{U}$ is $\mathbf{u}(j\Delta t)$. We then determine a matrix $\mathbf{W}_{out}$ that satisfies the following linear system:

$$\mathbf{W}_{out}(\tilde{\mathbf{R}}\tilde{\mathbf{R}}^T + \beta I) = \mathbf{U}\tilde{\mathbf{R}}^T. \quad (6)$$

This can be done by explicitly calculating the matrix inverse, $\mathbf{W}_{out} = \mathbf{U}\tilde{\mathbf{R}}^T(\tilde{\mathbf{R}}\tilde{\mathbf{R}}^T + \beta I)^{-1}$, or, more efficiently, by using a matrix division algorithm such as **mrdivide** in MATLAB[20] as we do in our numerical experiments. We additionally note that methods of solving Eq. (5) for $\mathbf{W}_{out}$ other than direct matrix solution are also available and may sometimes be advantageous (e.g., Generalized Minimal Residual Method (GMRES),[21] stochastic gradient descent,[22] etc.). By means of this minimization, it is hoped that $\mathbf{u}_{out}(t) \cong \mathbf{u}(t)$ is achieved. This completes the training process, following which we can switch to the closed-loop configurations [Fig. 1(b) and the dashed line in Fig. 2] and attempt to predict the subsequent evolution of $\mathbf{u}(t)$. Prediction can then proceed via Eqs. (1), (2), and (6) where the prediction of the dynamical system state $\tilde{\mathbf{u}}(t) = \mathbf{u}_{out}(t)$ and the reservoir input is received from the feedback loop ($\mathbf{u}_{in}(t) = \mathbf{u}_{out}(t)$).

The closed-loop configuration system can be regarded as a surrogate dynamical system that mimics the original unknown dynamical system. As such, if the original unknown dynamical system is chaotic, the closed-loop predictor system will also be chaotic. Due to the exponential growth of small errors implied by chaos, we cannot expect prediction to be good for more than several Lyapunov times (the Lyapunov time is the typical e-folding time for error growth in a chaotic system). Thus, we will regard our predictions to be successful when they are good for a few Lyapunov times.

Now consider that we have made a prediction for $\mathbf{u}(t)$, and, at some later time, we wish to perform another prediction of the same spatiotemporally evolving system with unknown dynamics. It is not necessary to retrain our predictor; we can, instead, re-use the previously obtained $\mathbf{W}_{out}$.[5] To do so, we re-initialize the reservoir state to zero, switch the reservoir computer into its open-loop configuration, and allow it to evolve given input states of the unknown dynamical system measured at times $t_p - T_S \le t \le t_p$ [i.e., $\mathbf{u}_{in}(t) = \mathbf{u}(t)$]. T_S is some synchronization time that is sufficiently longer than the characteristic memory of the reservoir computer but, importantly, is much shorter than the necessary training time needed to determine $\mathbf{W}_{out}$. t_p is the time at which we want to begin our prediction. After

this synchronization period, the reservoir computer is switched to its standard closed-loop prediction configuration and is used to make predictions at later times.

If the original system is very high dimensional (i.e., the dimension K of the measure vector $\mathbf{u}(t)$ is very large), then $D_r \gg K$ must be exceedingly large. This can make the training to determine $\mathbf{W}_{out}$ infeasible. For example, if we solve Eq. (5) by the direct matrix method, Eq. (6) shows that we must solve a $D_r \times D_r$ linear system of equations. For our computational resources, we find that this becomes impossible as D_r approaches 2×10^4. Due to this and other similar considerations, we deem the method discussed in this section to be untenable for the prediction of large, spatiotemporally chaotic systems of the type we are interested in.

C. Hybrid reservoir computing

In this section, we briefly review a hybrid scheme proposed in Ref. 6 for combining reservoir computing with an imperfect knowledge-based model of the dynamical system of interest. We again assume access to time series data of measurements of the state of the dynamical system. We further assume that an imperfect knowledge-based model of the system producing the measurements is available and that this imperfect model is capable of forecasting the state of the dynamical system with some degree of accuracy, which we wish to improve upon. In the hybrid setup of Ref. 6 (Fig. 3) described below, it has been shown that the machine learning method and the knowledge-based model augment each other and, in conjunction, can provide a significantly better forecast than either the knowledge-based model or the pure machine learning model acting alone.

As in Sec. II, we assume that the data used for training is given by K measurements of the state of the dynamical system at equally spaced increments in time, Δt, forming a vector time series $\mathbf{u}(t)$. The

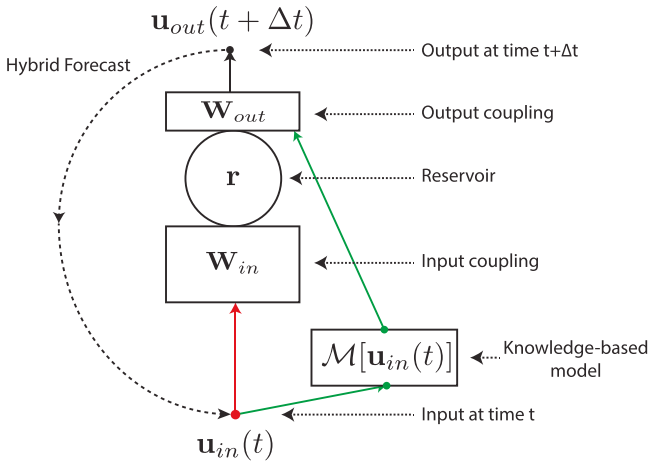

FIG. 3. Diagram of the hybrid reservoir computer setup. In the "open-loop" training phase [analogous to Fig. 1(a)], the dashed line representing coupling from the output back to the input is absent. In the "closed-loop" prediction phase [analogous to Fig. 1(b)], the coupling from the output back to the input (dashed line) is activated.

imperfect knowledge-based model $\mathcal{M}$ is an operator that maps the state $\mathbf{u}(t)$ to a forecast of the state at time $(t + \Delta t)$.

We advance the reservoir state in time using the same activation function as described in Sec. II B,

$$\mathbf{r}(t + \Delta t) = \tanh[\mathbf{A}\mathbf{r}(t) + \mathbf{W}_{in}\mathbf{u}_{in}(t)]. \tag{7}$$

Once again, during the training phase, $\mathbf{u}_{in}(t) = \mathbf{u}(t) + s\boldsymbol{\eta}(t)$. The training process is similar to the one employed in Sec. II for the basic reservoir computer but with the addition of the knowledge-based prediction (as illustrated in Fig. 3). Using ridge regression, we find a linear mapping $\mathbf{W}_{out}$ from $\tilde{\mathbf{r}}(t)$ and $\mathcal{M}[\mathbf{u}(t) + s\boldsymbol{\eta}(t)]$ to produce an approximate prediction of $\mathbf{u}(t + \Delta t)$,

$$\mathbf{W}_{out}\begin{bmatrix}\tilde{\mathbf{r}}(t+\Delta t)\\ \mathcal{M}[\mathbf{u}_{in}(t)]\end{bmatrix} \simeq \mathbf{u}(t+\Delta t). \tag{8}$$

Here, $\mathbf{u}_{in}(t)$ is the same as that input to the reservoir, $\mathbf{u}_{in}(t) = \mathbf{u}(t) + s\boldsymbol{\eta}(t)$. We again include the small $s\boldsymbol{\eta}(t)$ vector in the knowledge-based model input during training to improve the stability of the method. Additionally, recall that $\tilde{\mathbf{r}}$ is related to $\mathbf{r}$ by Eq. (2). In the prediction phase, we run the hybrid system in a closed loop feedback configuration (Fig. 3 with the dashed line feedback connection present) using Eqs. (2) and (7), and the following equation:

$$\mathbf{u}_{out}(t + \Delta t) = \mathbf{W}_{out}\begin{bmatrix}\tilde{\mathbf{r}}(t+\Delta t)\\ \mathcal{M}[\mathbf{u}_{out}(t)]\end{bmatrix}. \tag{9}$$

During the prediction phase, the hybrid forecast $\tilde{\mathbf{u}}_{\mathcal{H}}(t) = \mathbf{u}_{out}(t)$ and the hybrid input is received from the feedback loop ($\mathbf{u}_{in}(t) = \mathbf{u}_{out}(t)$). Note that, in this scheme, the output is a linear combination of the reservoir state and the knowledge-based model output that optimizes the agreement of the combined system output with the training data. Thus, we can regard the result as being an optimum combination of the reservoir and knowledge-based components. Hence, we expect that if one component is superior for some aspect of the prediction, then it will be weighted more highly for that aspect of the prediction. This suggests that predictions by this method may be greatly improved over those available from either the knowledge-based component or the reservoir component acting alone (e.g., see Fig. 7 of Ref. 6).

In addition to the hybrid configuration shown in Fig. 3, we have also tested a modified configuration in which there is an additional input to the reservoir component from the output of the knowledge-based model $\mathcal{M}$. We have empirically found that this modification sometimes yields a small positive improvement in prediction; however, for simplicity, we henceforth only consider the configuration in the figure.

D. Parallelization

To obtain a good prediction of a chaotic dynamical system state using reservoir computing, the reservoir dimensionality must be much greater than that of the dynamical system (i.e., $D_r \gg K$) so that there are enough free parameters available in $\mathbf{W}_{out}$ for fitting the reservoir output state to the measured dynamical system state at time $(t + \Delta t)$. This can cause the computational cost of determining an optimum output matrix to become unfeasibly high for large dynamical systems, e.g., because the implementation of this step by the method of Eq. (6) involves solving a $D_r \times D_r$ linear system. As a

point of reference, we note that the dimension of the state vector of a current typical operational global weather forecasting model is on the order of 10^9. A method to make consideration of such problems feasible for machine learning approaches was proposed in Ref. 5. The idea is to exploit the short range of causal interactions over a small time period in many spatiotemporally chaotic dynamical systems. This was shown to allow the use of multiple relatively small reservoir computers that each make predictions of a part of the full dynamical system state as in a local region, illustrated in Fig. 4 and explained below. This method has the advantage that, in the training phase, all of the relatively small output matrices $\mathbf{W}_{out,p}$ of each reservoir computer can be trained independently, in parallel, thus greatly reducing the difficulty of training.

For illustrative purposes, consider a spatiotemporal dynamical system in one spatial dimension with periodic boundary conditions. Let the dynamical system state be represented by a K-dimensional vector time series $\mathbf{u}(t) = [u_1(t), u_2(t), \ldots, u_K(t)]^T$, where each scalar component $u_j(t)$ represents the time series at a single spatial grid point. We divide the system state into P equally sized, contiguous regions containing Q system variables, where $PQ = K$. We denote the system variables in these regions as $\mathbf{u}_p(t) = [u_{Q(p-1)+1}(t), \ldots, u_{Qp}(t)]^t$, where $1 \le p \le P$. Each local region in space is predicted by a reservoir R_p, each of which has internal reservoir states $\mathbf{r}_p(t)$ and adjacency matrix $\mathbf{A}_p$ generated via the process described in Sec. II B. Each reservoir is coupled to its input, $\mathbf{u}_{in,p}(t)$, via a matrix of input weights, $\mathbf{W}_{in,p}$. This input corresponds to a local region of the system that contains the region to be predicted by that reservoir as well as a size ℓ overlap region on either side, $\mathbf{u}_{in,p}(t) = [u_{in,Q(p-1)+1-\ell}(t), u_{in,Q(p-1)+2-\ell}(t), \ldots, u_{in,Qp+\ell}(t)]^T$. We denote the dynamical system state in this input region by the size $(2\ell + Q)$ dimensional vector $\mathbf{v}_p(t) = [u_{Q(p-1)+1-\ell}(t), u_{Q(p-1)+2-\ell}(t), \ldots, u_{Qp+\ell}(t)]^T$. This overlap accounts for the short range causal interactions across the boundaries of the local regions. The assumption here is that, over the incremental prediction time Δt, state information does not propagate fast enough for nodes outside the input regions of reservoir p to influence the time Δt change in the dynamical system states predicted by reservoir p.

Each reservoir state is advanced using the following equation:

$$\mathbf{r}_p(t + \Delta t) = \tanh[\mathbf{A}_p \mathbf{r}_p(t) + \mathbf{W}_{in,p}\mathbf{u}_{in,p}(t)]. \tag{10}$$

During the training phase (Fig. 4 with the dashed output-to-input connection absent), $\mathbf{u}_{in,p}(t) = \mathbf{v}_p(t) + s\boldsymbol{\eta}(t)_p$. Here, the $(2\ell + Q)$ dimensional vector $\boldsymbol{\eta}(t)_p$ is the pth local region of a global vector of normally distributed random variables, $\boldsymbol{\eta}(t)$, chosen independently at each time t,

$$\boldsymbol{\eta}(t)_p = \begin{bmatrix} \eta_{Q(p-1)+1-\ell} \\ \eta_{Q(p-1)+2-\ell} \\ \vdots \\ \eta_{Qp+\ell} \end{bmatrix}. \tag{11}$$

After a suitably long transient nullification period, we determine the output matrices $\mathbf{W}_{out,p}$ for each reservoir that solve the least squared optimization problem using ridge regression,

$$\min_{\mathbf{W}_{out,p}} \left\{ \sum_{0 \le t < t_0} \left[\|\mathbf{W}_{out,p}\tilde{\mathbf{r}}_p(t + \Delta t) - \mathbf{u}_p(t + \Delta t)\|^2 \right] + \beta\, \mathrm{Trace}(\mathbf{W}_{out,p}\mathbf{W}_{out,p}^T) \right\}. \tag{12}$$

Note that, for each p, matrix $\mathbf{W}_{out,p}$ can be relatively small as the number of outputs is the number of state variables in region p (not the entire global state). Furthermore, the determinations of the relatively small $\mathbf{W}_{out,p}$ matrices are independent for each region p, and thus can be computed in parallel. The "direct matrix method" solution for determining each of the $\mathbf{W}_{out,p}$ matrices proceeds as follows. First, we rewrite Eq. (12) as

$$\min_{\mathbf{W}_{out,p}} \|\mathbf{U}_p - \mathbf{W}_{out,p}\tilde{\mathbf{R}}_p\|^2 + \beta \mathrm{Trace}(\mathbf{W}_{out,p}\mathbf{W}_{out,p}^T), \tag{13}$$

where in Eq. (13), $\mathbf{U}_p$ and $\tilde{\mathbf{R}}_p$ are analogous to $\mathbf{U}$ and $\tilde{\mathbf{R}}$ in the single reservoir prediction [see Eq. (6)]. $\mathbf{U}_p$ is the target matrix such that the jth column is $\mathbf{u}_p(j\Delta t)$, while $\tilde{\mathbf{R}}_p$ is obtained from $\mathbf{R}_p$ analogous to the single reservoir case as described in Eq. (2). Each $\mathbf{W}_{out,p}$ is calculated by solving the following equation:

$$\mathbf{W}_{out,p}(\tilde{\mathbf{R}}_p\tilde{\mathbf{R}}_p^T + \beta I) = \mathbf{U}_p\tilde{\mathbf{R}}_p^T. \tag{14}$$

Note that, as previously claimed, the minimization problem for each p, Eq. (13), is completely independent and can be solved for the different p in parallel. As in Sec. II B, in the prediction phase and, after a period of synchronization, we produce a full state prediction $\tilde{\mathbf{u}}(t)$ by running the system in a closed loop feedback configuration (i.e., Fig. 4 with the dashed output-to-input feedback connection present). This is done by concatenating the local predictions from each reservoir $\tilde{\mathbf{u}}_p(t) = \mathbf{u}_{out,p}(t)$ [where $\mathbf{u}_{out,p}(t)$ is the output state from each reservoir]. Reservoir p then receives inputs from its own outputs in addition to the left and right overlap zone inputs from the ℓ left grid points and the ℓ right grid points. The entire system thus evolves as follows:

$$\mathbf{u}_{out,p}(t) = \mathbf{W}_{out,p}[\tilde{\mathbf{r}}_p(t)], \tag{15}$$

$$\tilde{\mathbf{u}}(t) = \begin{bmatrix} \tilde{\mathbf{u}}_1(t) \\ \tilde{\mathbf{u}}_2(t) \\ \vdots \\ \tilde{\mathbf{u}}_P(t) \end{bmatrix} = \begin{bmatrix} \mathbf{u}_{out,1}(t) \\ \mathbf{u}_{out,2}(t) \\ \vdots \\ \mathbf{u}_{out,P}(t) \end{bmatrix}, \quad \mathbf{u}_{in,p}(t) = \tilde{\mathbf{v}}_p(t) = \begin{bmatrix} \tilde{u}_{Q(p-1)+1-\ell}(t) \\ \tilde{u}_{Q(p-1)+2-\ell}(t) \\ \vdots \\ \tilde{u}_{Qp+\ell}(t) \end{bmatrix}, \tag{16}$$

$$\mathbf{r}_p(t + \Delta t) = \tanh[\mathbf{A}_p\mathbf{r}_p(t) + \mathbf{W}_{in,p}\mathbf{u}_{in,p}(t)], \tag{17}$$

where $\tilde{\mathbf{r}}_p(t)$ is obtained from $\mathbf{r}_p(t)$ using Eq.(2).

Finally, we compare the parallel reservoir method with Convolutional Neural Networks (CNNs), a commonly used form of the artificial neural network architecture that is constructed to learn spatial features.[23] The main differences between the parallel reservoir

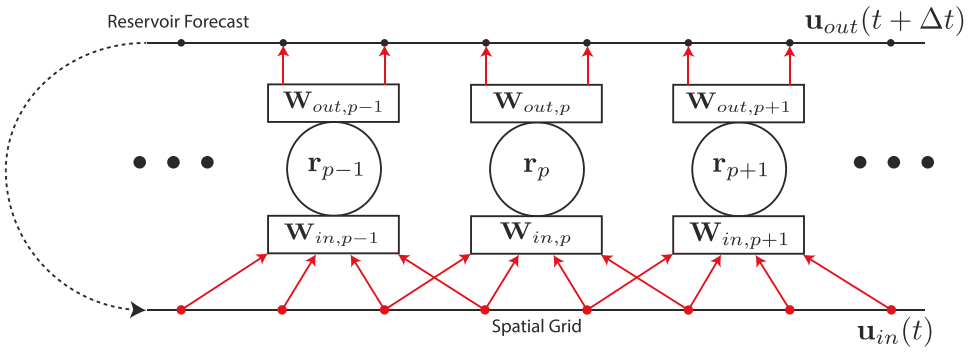

FIG. 4. Diagram of the parallel reservoir computer setup.

method and CNN-based methods are in the type of models they produce and the method for training each of them. CNNs capture spatial features via a global model of the local dynamics that parameterizes many different local fields. Our chosen approach, on the other hand, determines many local models for each spatial region which, in the case of a spatially inhomogeneous system such as terrestrial weather, we believe will be more accurate than a global model. In addition, we expect our chosen method is easier to train due to the simplicity of obtaining the least squares optimal solution for each relatively small local region.

E. Combined hybrid/parallel prediction (CHyPP)

In our previous work,[6] we constructed a hybrid prediction method using a single reservoir. In this section, we consider a combination of the parallel approach of Sec. II D (which enables computationally efficient scaling to high-dimensional spatiotemporal chaos) and the hybrid approach of Sec. II C (which allows us to utilize partial prior knowledge of the dynamical system), where we assume that the knowledge-based system provides *global* predictions over the entire spatial domain. A diagram of this method can be found in Fig. 5. While the approach is easily generalized to 2- and 3-dimensional spatial processes, in order to most simply demonstrate our proposed methodology, we again consider a one-dimensional, spatiotemporally chaotic dynamical system with periodic boundary conditions, with a state represented by a K-dimensional vector time series $\mathbf{u}(t) = [u_1(t), u_2(t), \ldots, u_K(t)]^T$. Our approximate knowledge-based prediction operator $\mathcal{M}$ gives a global prediction of the full system state for a time Δt: $\mathcal{M}[\mathbf{u}(t)] = \hat{\mathbf{u}}(t+\Delta t)$. As in our parallel reservoir computer prediction described in Sec. II D, we partition the system state into P equally sized, continuous regions containing Q variables, where $PQ = K$ and each such region is predicted by a reservoir R_p, $p = 1, 2, \ldots, P$.

Each reservoir R_p input is coupled to a local region of the system states as in Sec. II D, and the reservoir state $\mathbf{r}_p(t)$ is advanced using the following equation:

$$\mathbf{r}_p(t+\Delta t) = \tanh[\mathbf{A}_p\mathbf{r}_p(t) + \mathbf{W}_{in,p}\mathbf{u}_{in,p}(t)]. \tag{18}$$

During the initial training phase, $\mathbf{u}_{in,p}(t) = \mathbf{v}_p(t) + s\boldsymbol{\eta}(t)_p$. In Eq. (18), $\mathbf{W}_{in,p}$ is the input coupling matrix for the local system states, analogous to that described in Sec. II B. As in Sec. II D, $\mathbf{v}_p(t)$ is the state measurements at grid points within the local region to be predicted along with ℓ grid points to either side and $\boldsymbol{\eta}(t)_p$ is the pth local region of the global vector of normally distributed random numbers $\boldsymbol{\eta}(t)$. Each reservoir is trained independently, in parallel, using a set of training data consisting of an equally spaced time series of measured states of the large scale dynamics beginning at $t = 0$ after some initial transient nullification period. Again, we solve the least squares optimization problem with ridge regression to determine an output mapping for each reservoir [analogous to Eq. (13)],

$$\min_{\mathbf{W}_{out,p}} \left\{ \left\| \mathbf{U}_p - \mathbf{W}_{out,p} \begin{bmatrix} \tilde{\mathbf{R}}_p \\ \hat{\mathbf{U}}_p \end{bmatrix} \right\|^2 + \beta \mathrm{Trace}(\mathbf{W}_{out,p}\mathbf{W}_{out,p}^T) \right\}. \tag{19}$$

In Eq. (19), $\hat{\mathbf{U}}_p$ is a matrix whose jth column is $\hat{\mathbf{u}}_p(j\Delta t)$, where $\hat{\mathbf{u}}_p(j\Delta t)$ is the knowledge-based prediction of the pth local region

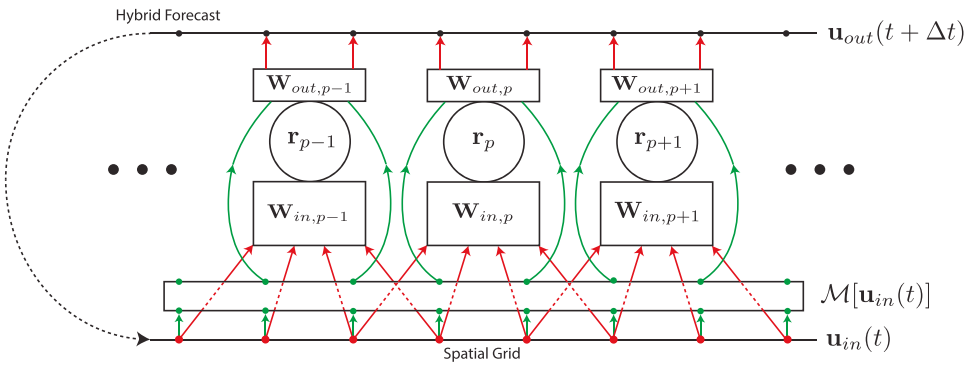

FIG. 5. Diagram of the Combined Hybrid/Parallel Prediction (CHyPP) architecture using reservoir computing.

TABLE I. Hyperparameters used in the results in Sec. III.

$\langle d \rangle$ (average in degree)	3	ℓ (local overlap length)	6
ρ (spectral radius)	0.6	Δt (prediction time step)	0.25
σ (input coupling strength)	0.1	T_S (synchronization time)	25
β (regularization)	10^{-6}	s (CHyPP tests)	0
s [reservoir(s)-only tests]	0.001		

of the system,

$$\hat{\mathbf{u}}_p(t+\Delta t) = \begin{bmatrix} \mathcal{M}[\mathbf{u}(t)+s\boldsymbol{\eta}(t)]_{Q(p-1)+1} \\ \mathcal{M}[\mathbf{u}(t)+s\boldsymbol{\eta}(t)]_{Q(p-1)+2} \\ \vdots \\ \mathcal{M}[\mathbf{u}(t)+s\boldsymbol{\eta}(t)]_{Qp} \end{bmatrix}. \tag{20}$$

The solution to Eq. (19) by the direct matrix method is

$$\mathbf{W}_{out,p}\left(\begin{bmatrix}\tilde{\mathbf{R}}_p \\ \hat{\mathbf{U}}_p\end{bmatrix}[\tilde{\mathbf{R}}_p^T, \hat{\mathbf{U}}_p^T] + \beta I\right) = \mathbf{U}_p[\tilde{\mathbf{R}}_p^T, \hat{\mathbf{U}}_p^T]. \tag{21}$$

In the prediction phase, we run the system in a closed-loop configuration according to Eqs. (22)–(25) to predict each local region of the system given each reservoir state and knowledge-based model prediction, obtaining $\tilde{\mathbf{u}}_p(j\Delta t)$. The local predictions are appropriately concatenated to form a full state prediction, which is then used as input for the next prediction step, as follows:

$$\tilde{\mathbf{u}}_p(t) = \mathbf{u}_{out,p}(t) = \mathbf{W}_{out,p}\begin{bmatrix}\tilde{\mathbf{r}}_p(t) \\ \hat{\mathbf{u}}_p(t)\end{bmatrix}, \tag{22}$$

$$\tilde{\mathbf{u}}(t) = \begin{bmatrix}\tilde{\mathbf{u}}_1(t) \\ \tilde{\mathbf{u}}_2(t) \\ \vdots \\ \tilde{\mathbf{u}}_P(t)\end{bmatrix}, \quad \mathbf{u}_{in,p}(t) = \tilde{\mathbf{v}}_p(t) = \begin{bmatrix}\tilde{u}_{Q(p-1)+1-\ell}(t) \\ \tilde{u}_{Q(p-1)+2-\ell}(t) \\ \vdots \\ \tilde{u}_{Qp+\ell}(t)]\end{bmatrix}, \tag{23}$$

$$\hat{\mathbf{u}}_p(t+\Delta t) = \begin{bmatrix} \mathcal{M}[\mathbf{u}_{in}(t)]_{Q(p-1)+1} \\ \mathcal{M}[\mathbf{u}_{in}(t)]_{Q(p-1)+2} \\ \vdots \\ \mathcal{M}[\mathbf{u}_{in}(t)]_{Qp} \end{bmatrix}, \tag{24}$$

$$\mathbf{r}_p(t+\Delta t) = \tanh[\mathbf{A}_p\mathbf{r}_p(t) + \mathbf{W}_{in,p}\mathbf{u}_{in,p}(t)]. \tag{25}$$

In the above equations, we once again calculate $\tilde{\mathbf{r}}_p(t)$ from $\mathbf{r}_p(t)$ using Eq. (2).

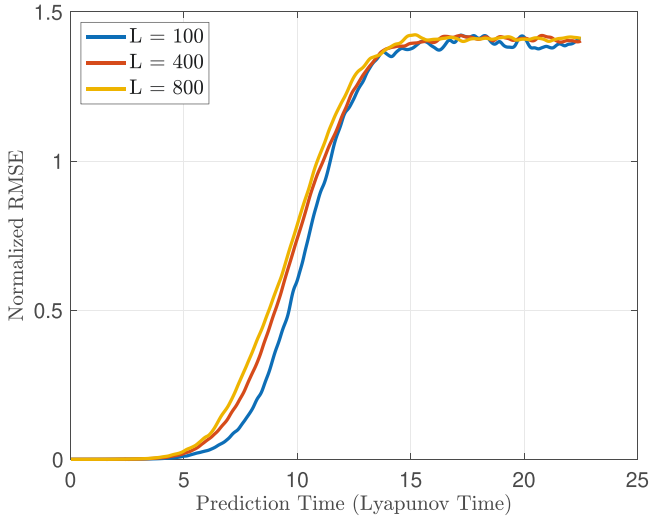

FIG. 6. Average normalized root mean square error (NRMSE) in the CHyPP prediction, where we have fixed the number of system variables predicted in each region to $Q = 8$ and the reservoir spatial density at $P/L = 16/100$. The total number of spatial grid points K in each prediction are $K = 128$ for $L = 100$, $K = 512$ for $L = 400$, and $K = 1024$ for $L = 800$. Predictions are made using an imperfect model with second derivative error $\epsilon = 0.1$. The results shown are the average of 100 predictions, where each prediction is made using the same set of reservoirs and training dataset but where the CHyPP method is synchronized to different initial conditions for $\mathbf{u}(t)$. we see that the NRMSE curves are relatively invariant as L increases, indicating that our method can be scaled to large systems.

III. TEST RESULTS ON THE KURAMOTO–SIVASHINSKY EQUATION

We test the effectiveness of our proposed CHyPP method (Sec. II E) by forecasting the state evolution of the Kuramoto–Sivashinsky equation with periodic boundary conditions,[24,25]

$$\frac{\partial y}{\partial t} = -y\frac{\partial y}{\partial x} - \frac{\partial^2 y}{\partial x^2} - \frac{\partial^4 y}{\partial x^4}, \tag{26}$$

where $y = y(x,t)$ and $y(x+L,t) = y(x,t)$. This spatial one-dimensional model generally produces spatiotemporally chaotic dynamics for periodicity length $L \gtrsim 50$. For the purpose of comparing various methods, we regard Eq. (26) as generating the state measurements of a putative system that we are interested in, while the imperfect prediction model we use is a modified version of this same equation, where an error term ϵ is introduced in the coefficient of the second derivative term,

$$\frac{\partial y}{\partial t} = -y\frac{\partial y}{\partial x} - (1+\epsilon)\frac{\partial^2 y}{\partial x^2} - \frac{\partial^4 y}{\partial x^4}. \tag{27}$$

We have also investigated the case where the error is introduced by multiplying the $y\partial y/\partial x$ term in Eq. (26) by $(1+\epsilon)$. For the latter case, the results of our method are qualitatively similar to those for Eq. (27). We form our simulated measured time series $\mathbf{u}(t)$ by taking the ith element of $\mathbf{u}(t)$ to be $y(i\Delta x, t)$, where $\Delta x = L/K$ is the grid spacing used for our numerical solutions of Eq. (26). We numerically solve the Kuramoto–Sivashinky equation on the same discretized spatial grid using fourth-order Runge–Kutta exponential time differencing.[26] As a metric for how long a prediction is valid, we calculate the normalized root mean square error (NRMSE) between the true and predicted system states. We define the length of valid prediction, or "valid time," to be the time at which NRMSE exceeds 0.2. Since the NRMSE saturates at sqrt(2), we consider this to be the point when error in the prediction reaches about 15% of its saturation value. In Sec. III A, we demonstrate that our CHyPP methodology can scale to predict very large systems. In Sec. III B,

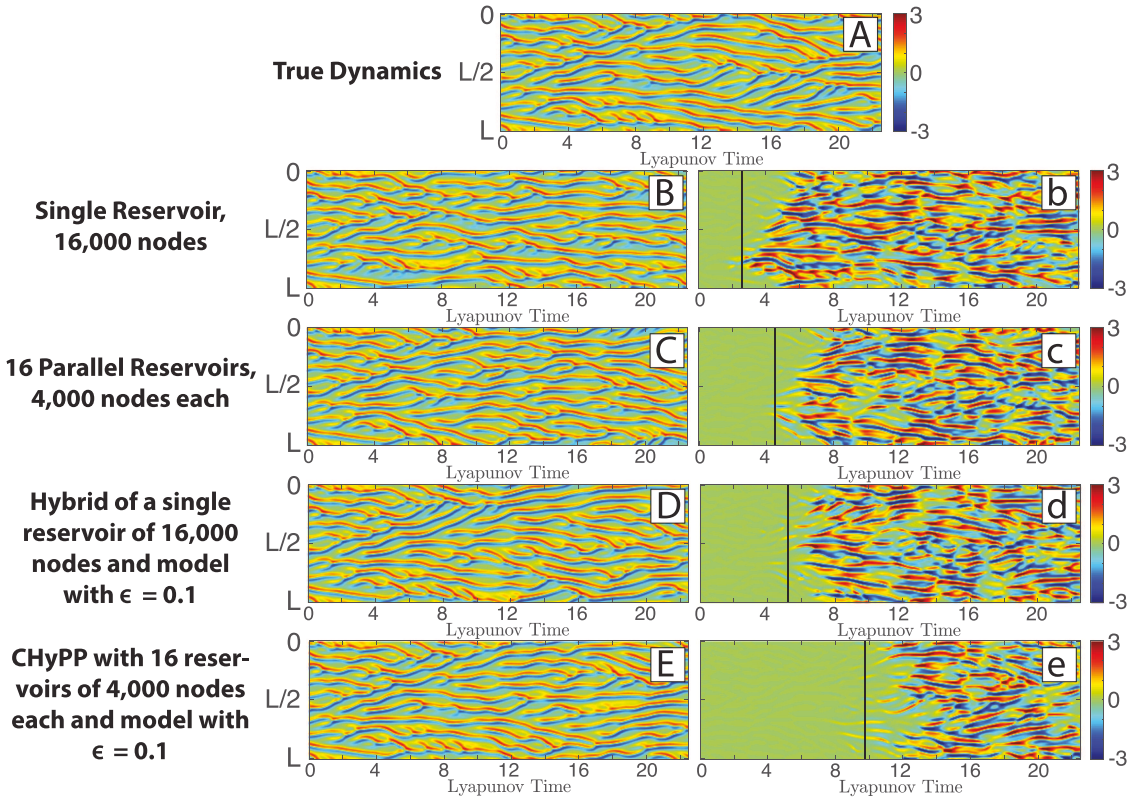

FIG. 7. (a) The true dynamics of the Kuramoto–Sivashinsky equation obtained by numerical evolution of Eq. (26) normalized to have mean 0 and variance 1. We plot the spatial grid point along the vertical axis and the Lyapunov time on the horizontal axis. The Lyapunov time is defined by $\Lambda_{max}t$, where Λ_{max} is the largest Lyapunov exponent computed from Eq. (26). Predictions of this evolution by four different methods are shown in panels (b)–(e). The prediction in (b) is made by a single reservoir computer with 16 000 nodes, the prediction in (c) is made by 16 reservoir computers, in parallel, each with 4000 nodes, the prediction in (d) is made by a hybrid with a single reservoir computer with 16 000 nodes(D), and the prediction in (e) is made by CHyPP with 16 reservoir computers, in parallel, each with 4000 nodes and an imperfect model with second derivative $\epsilon = 0.1$. The plots on the right (b)–(e) display the difference between the true dynamics (a) and the prediction using each of the corresponding techniques. The vertical black line marks the valid prediction time. Each technique was trained using a 3375 Lyapunov time sequence of training data.

we show that the CHyPP method requires significantly less training data than the parallel scheme of Sec. II D (using reservoirs without a knowledge-based component). We then discuss, in Sec. III C, the sensitivity of the CHyPP and parallel machine learning (Sec. II D) methods to the local overlap length ℓ.

For all of our numerical experiments in this paper, in addition to our solution of the imperfect model, we use a digital computer to implement the machine learning. However, if, in the future, CHyPP is applied to very large systems (e.g., weather forecasting) requiring a much larger number of parallel machine learning units, then we envision that it may prove useful to perform the parallel machine learning using a special purpose physically implemented reservoir computing array, e.g., based on FPGAs or photonic devices.[15–17] Such implementations, called "AI hardware accelerators," show great promise with respect to low cost, speed, and compactness.

A. Prediction scalability

We first test the ability of our CHyPP method to scale to large system sizes. We consider the Kuramoto–Sivashinsky equation, where we fix the number of system variables (grid points) each reservoir is trained to predict to $Q = 8$, as well as the reservoir spatial density to $P/L = 16/100$ (P is the number of reservoirs), while varying the periodicity length L. For all tests in this section, we use the hyperparameters in Table I unless otherwise specified. We additionally fix the error in the incorrect model to $\epsilon = 0.1$. Figure 6 shows the resulting NRMSE between the true state and our hybrid parallel prediction averaged over 100 prediction periods vs time. For each value of L plotted in Fig. 6, the density of the parallel reservoir computers is kept constant at $P/L = 0.16$. Additionally, the time plotted horizontally is in units of the Lyapunov time (the average chaos-induced e-folding time of small errors in the predicted state orbit). The NRMSE is relatively unchanged as the value of L is increased, indicating that the CHyPP prediction, like the parallel reservoir-only prediction in Ref. 5, can be scaled to very large systems by the addition of more reservoirs.

Finally, we note that our test system, the Kuramoto–Sivashinky equation, is homogeneous in space, while the real systems we are interested in are generally spatially inhomogeneous. For example,

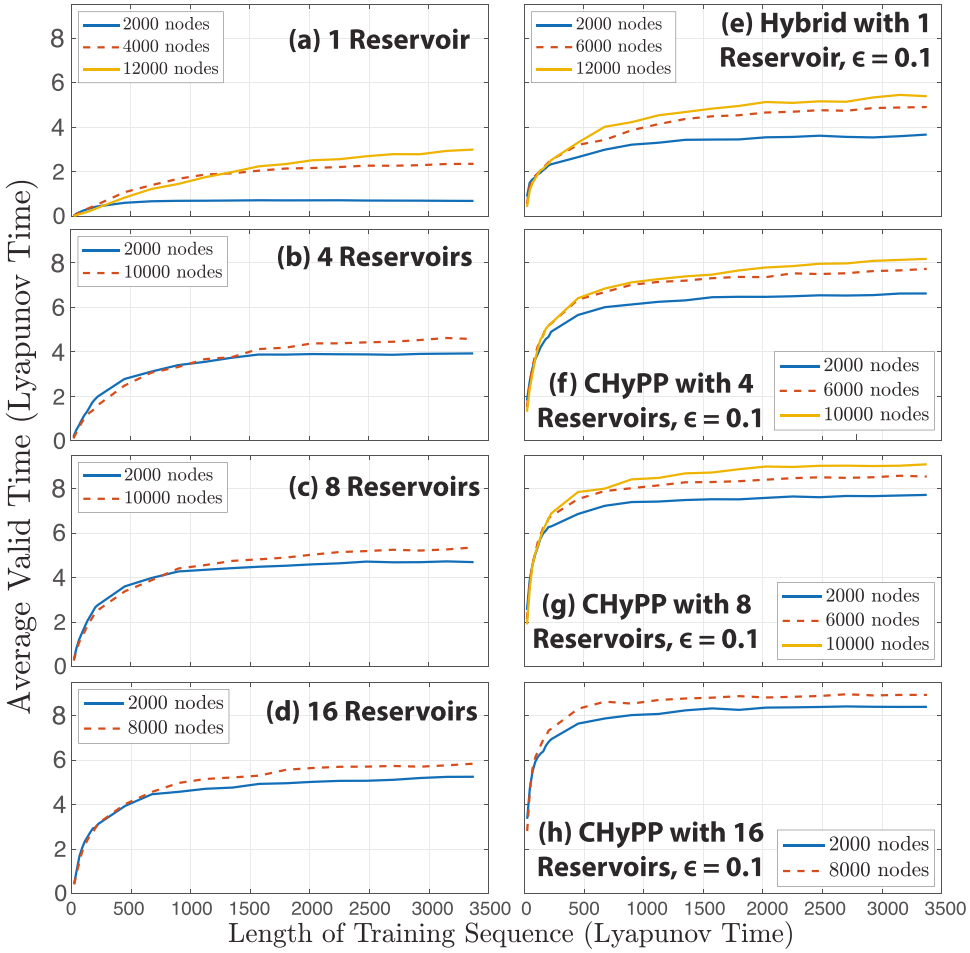

FIG. 8. We plot the average valid time as a function of the duration of the training data, each in Lyapunov time, for the parallel reservoir(s)-only method using (a) one reservoir, (b) four reservoirs, (c) eight reservoirs, and (d) 16 reservoirs. We plot the same quantity obtained from the parallel hybrid method using (e) one reservoir, (f) four reservoirs, (g) eight reservoirs, and (h) 16 reservoirs. In each case, the imperfect knowledge-based predictor in the hybrid prediction had an error in the second derivative term of $\epsilon = 0.1$, resulting in an average valid time for the model-only prediction of 0.48 of a Lyapunov time. A knowledge-based predictor with no error begun using the most accurate initial condition from the parallel hybrid was able to predict for an average of 11 Lyapunov times. The legend in each plot indicates the number of nodes used in each reservoir.

weather forecasting accounts for geographic features (continents, mountains, etc.), as well as for latitudinal variation of solar input, among other spatially inhomogeneous factors. In order to ensure that our numerical tests with a homogeneous model are also relevant for a typical inhomogeneous situation, we have not made any use of the homogeneity of Eq. (26): the adjacency matrices $\mathbf{A}_p$ corresponding to each reservoir p are independently randomly generated and each $\mathbf{W}_{out,p}$ is determined separately (rather than taking all $\mathbf{A}_p$ and $\mathbf{W}_{out,p}$ to be the same, as would be possible for a homogeneous system).

B. CHyPP promotes the possibility of good performance using a relatively small duration of training data

The prediction results shown in Figs. 7 and 8 use the parameters contained in Table I. In addition, the number of reservoirs and the length of training data are specified in the figure captions. Our true Kuramoto–Sivashinky equation dynamics and our imperfect model use a periodicity length of $L = 100$ and are realized over $K = 128$ spatial grid points. The imperfect model has an error of $\epsilon = 0.1$ and a small valid prediction time of only 0.48 Lyapunov times. Each plotted valid time is averaged over 100 predictions that are generated using the same set of reservoirs with the same training data sequence but that are synchronized to different initial conditions. Figure 7 displays a set of example predictions of one of these test time series. Figures 7 and 8 both demonstrate that CHyPP yields significantly longer valid predictions than the parallel reservoir-only method, which (for our choice of ϵ and reservoir size) produces longer valid predictions than the imperfect model alone. Both the parallel hybrid and parallel reservoir-only predictions outperform the imperfect model-only approach. From Fig. 8, we also observe that the CHyPP saturates or reaches a valid prediction time that increases only negligibly with the addition of more training data, at a much shorter length of training data than the parallel reservoir-only. As a result of this, the length of training data before which we would not benefit from increasing the size of the reservoir is also much shorter in the CHyPP prediction. For example, consider the plots in (b) and (f) in Fig. 8, where each prediction uses four reservoirs. If we have a 500 Lyapunov time length of training data, we would not benefit from increasing the number of nodes per reservoir above 2000 in the reservoirs-only prediction

TABLE II. Hyperparameters used in the results in Sec. IV.

$\langle d \rangle$ (average in-degree)	3	ρ (spectral radius)	0.6
ℓ (local overlap length)	N/12	σ (input coupling strength)	1
Δt (Prediction time step)	0.005	β (regularization)	10^{-4}
T_S (synchronization time)	0.5		

but could obtain a significant performance improvement if we did so using CHyPP. CHyPP exhibits its most impressive performance for very short lengths of training data. With only 22.5 Lyapunov times worth of training data, the parallel reservoir-only method is able to predict for only 0.44 Lyapunov times, whereas CHyPP with 16 reservoirs of 2000 nodes each predicts for 3.35 Lyapunov times on average, matching the saturated single reservoir-only prediction that is trained using 150 times more training data.

C. Prediction quality dependence on local overlap length

In this section, we investigate the dependence of CHyPP performance on the local overlap length ℓ. Figure 9 shows that when the local overlap length ℓ is zero or close to zero, the parallel reservoir-only prediction valid time (solid red line) is very poor, predicting for only around 0.5 Lyapunov times.

CHyPP, however, is still able to obtain good predictions with a very little local overlap length. For CHyPP, this indicates that it is able to utilize the inaccurate model prediction to infer the propagation of dynamical influences between the adjacent prediction regions. We note that neither CHyPP nor the 16 parallel reservoirs-only prediction is able to improve upon a corresponding single, large reservoir hybrid prediction or corresponding single large reservoir-only prediction when the local overlap length is near 0. Regarding this comparison, however, it is important to recognize that in systems larger than the one we have used to test our method here, use of a single reservoir prediction is not possible because the reservoir size necessary for prediction becomes infeasible, as discussed in Sec. II. This result is, nevertheless, very important for large-scale implementations of the proposed method. In realistic systems with three spatial dimensions, a small increase in the local overlap length ℓ can lead to increasingly large memory requirements and increasingly large amounts of information that must be communicated between reservoirs at each prediction iteration. Being able to obtain a good prediction with less local overlap length when the propagating dynamics is adequately explained by the imperfect model is thus another advantage of CHyPP.

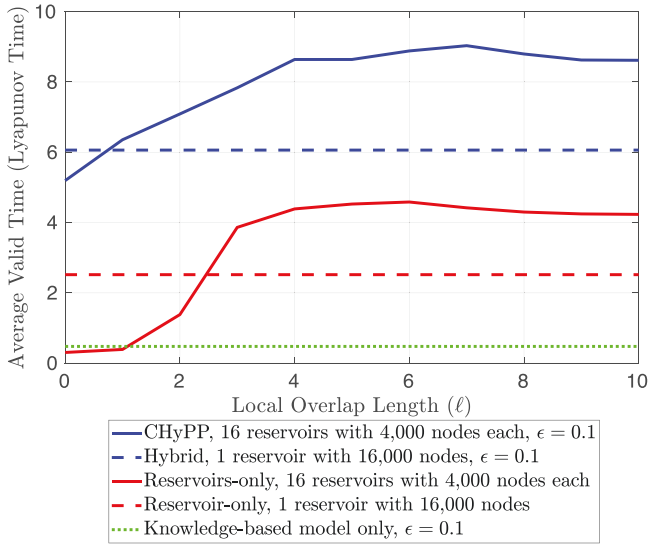

FIG. 9. We plot the valid time for the parallel and non-parallel hybrid and reservoir(s)-only prediction as a function of the local overlap length ℓ.

IV. TEST RESULTS ON A MULTISCALE SYSTEM PREDICTION AND THE PROBLEM OF SUBGRID-SCALE CLOSURE

A common difficulty in numerical modeling arises because many physical processes involve dynamics on multiple scales. As a result, fundamentally crucial subgrid-scale dynamics is often only crudely captured in an *ad hoc* manner. The formulation of subgrid-scale models by analytical techniques has been extensively studied (e.g., see Ref. 27) and is sometimes referred to as "subgrid-scale closure." In this section, we show that our CHyPP methodology provides a very effective data-based (as opposed to analysis-based) approach to subgrid-scale closure.

In particular, we test our CHyPP prediction method on a dynamical system with multiple spatial and temporal scales. The particular system we choose to predict is the multiscale "toy" atmospheric model formulated by Lorenz in his 2005 paper.[28] We refer to this model as Lorenz Model III. This model is a smoothed extension of Lorenz's original "toy" atmospheric model described in his 1996 paper[29] (hereafter referred to as Lorenz Model I), with the addition of small-scale dynamical activity. Lorenz Model III describes the evolution of a single atmospheric variable, Z, on a one-dimensional grid with N grid points and periodic boundary conditions, representing a single latitude. The value of Z at each grid point, Z_n, evolves according to the following equation:

$$dZ_n/dt = [X,X]_{K,n} + b^2[Y,Y]_{1,n} + c[Y,X]_{1,n} - X_n - bY_n + F. \tag{28}$$

In Eq. (28), X is a smoothed version of Z, and Y is the difference between Z and X,

$$X_n = \sum_{i=-I}^{I} (\alpha - \beta|i|) Z_{n+i}, \quad Y_n = Z_n - X_n, \tag{29}$$

$$\alpha = \left(3I^2+3\right)/\left(2I^3+4I\right), \quad \beta = \left(2I^2+1\right)/\left(I^4+2I^2\right). \tag{30}$$

Here, I denotes the smoothing distance. Thus, X describes the large-spatial scale, long-time scale wave component of Z, while Y describes the small-spatial scale, short-time scale wave component. $[V,W]_{K,n}$ indicates a coupling between the variables V and W [i.e., interaction within scales ($V = W = X$ or Y) or between scales for ($V = X$,

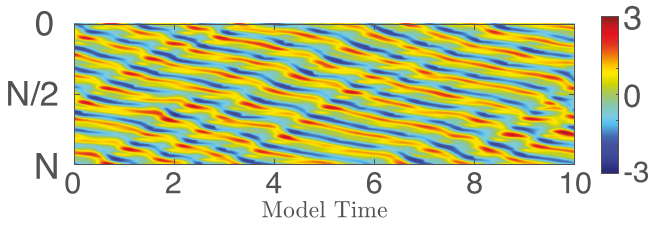

FIG. 10. Plot of the true dynamics from Lorenz Model III using the parameters $N = 960$, $K = 32$, $I = 12$, $F = 15$, $b = 10$, and $c = 2.5$. The dynamics have been normalized to set the mean value to 0 and variance to 1. We plot the value of Z color-coded (color bar on the right), the spatial grid point along the vertical axis, and the model time on the horizontal axis.

$W = Y)$]. For K even, this coupling takes the form

$$[V, W]_{K,n} = \sum_{j=-J}^{J}{}' \sum_{i=-J}^{J}{}' (-V_{n-2K-i} W_{n-K-j} + V_{n-K+j-i} W_{n+K+j})/K^2. \tag{31}$$

Here, $J = K/2$ and $\sum'$ denotes a modified summation where the first and last summands are divided by 2. For K odd, $J = (K-1)/2$ and each $\sum'$ becomes a standard summation $\sum$. In Eq. (28), the parameters K, b, c, and F describe the coupling distance of the system's large-scale dynamics, the increase in small-scale oscillation rapidity and the decrease in amplitude (relative to the large-scale dynamics), the degree of interaction between the large- and small-scale dynamics, and the overall forcing in the system, respectively. We note that when $b = c = 0$ and $K = 1$, this model reduces to the Lorenz Model I.

For the purposes of testing our CHyPP method, we also introduce another model formulated by Lorenz, which we will refer to as Lorenz Model II.[28] This model is equivalent to Lorenz Model III with no distinct small-scale wave component or smoothing present in the equation

$$dZ_n/dt = [Z, Z]_{K,n} - Z_n + F. \tag{32}$$

Importantly for our tests, Lorenz notes that, for constant F, the dominant wavenumber in Lorenz Model II depends only on the ratio N/K.

We test our CHyPP method by using it to predict the dynamics generated from Lorenz Model III using Lorenz Model II as our imperfect knowledge-based predictor. As we discussed in Sec. I, knowledge-based models used to predict physical multiscale systems (i.e., the weather) often use simplified representations of subgrid-scale dynamics. To replicate this type of imperfection in the knowledge-based model in CHyPP, we realize the Lorenz Model II dynamics over an equal or fewer number of grid points $N_{\text{Model II}}$ while using the same N/K ratio as used to generate the true Lorenz Model III dynamics (i.e., $N_{\text{Model III}}/K_{\text{Model III}} = N_{\text{Model II}}/K_{\text{Model II}}$). Our imperfect model thus incorrectly represents the effect of small-scale dynamics that, when $N_{\text{Model III}} > N_{\text{Model II}}$, are also subgrid. In our tests, each of these models is solved numerically using a fourth-order Runge–Kutta scheme.

For the following results in this section, we use the parameters in Table II. In addition, the value of s used in each of the reservoir computing-based prediction methods has been selected for each method to maximize the valid prediction time. Figure 10 shows a solution of Lorenz Model III where the value of Z is color-coded. The spatial variable is plotted vertically, time is plotted horizontally, and the model parameters are given in the caption. As seen in Fig. 10, there is a wave-like motion with a dominant wavenumber of ≈ 7 (seven oscillations along the vertical periodicity length). This corresponds to Lorenz's design of the model to mimic atmospheric dynamics, which has a predominant wavenumber for Rossby waves

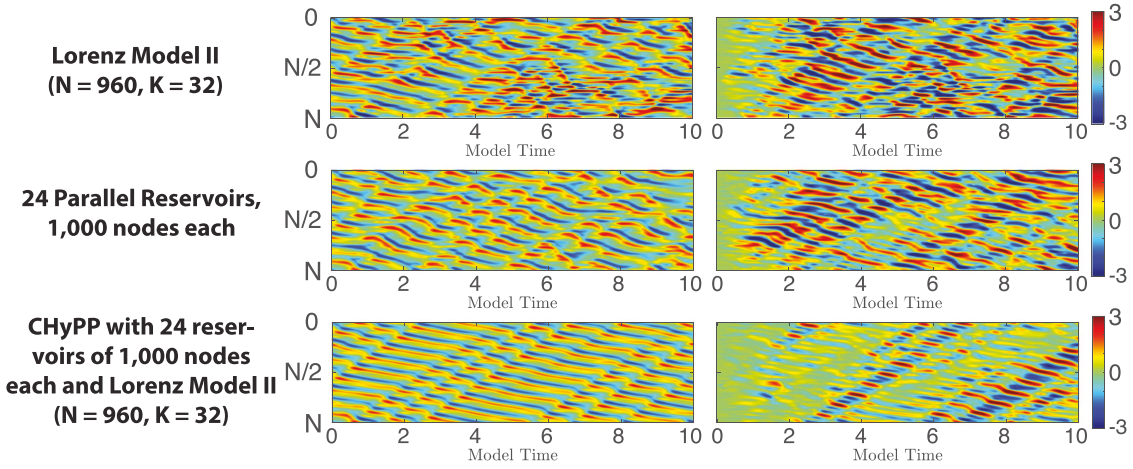

FIG. 11. We plot the predictions of the normalized true dynamics from Lorenz Model III (displayed in Fig. 10), where we have measured every spatial grid point of Model III during training. We plot the spatial grid point along the vertical axis and the model time on the horizontal axis. The plots on the left show the predictions made using each of the specified methods, while the plots on the right show the difference between the true dynamics and the prediction (i.e., the prediction error). The parallel reservoir-only prediction used its optimized value of $s = 0.1$, while CHyPP used its optimized value of $s = 0.085$.

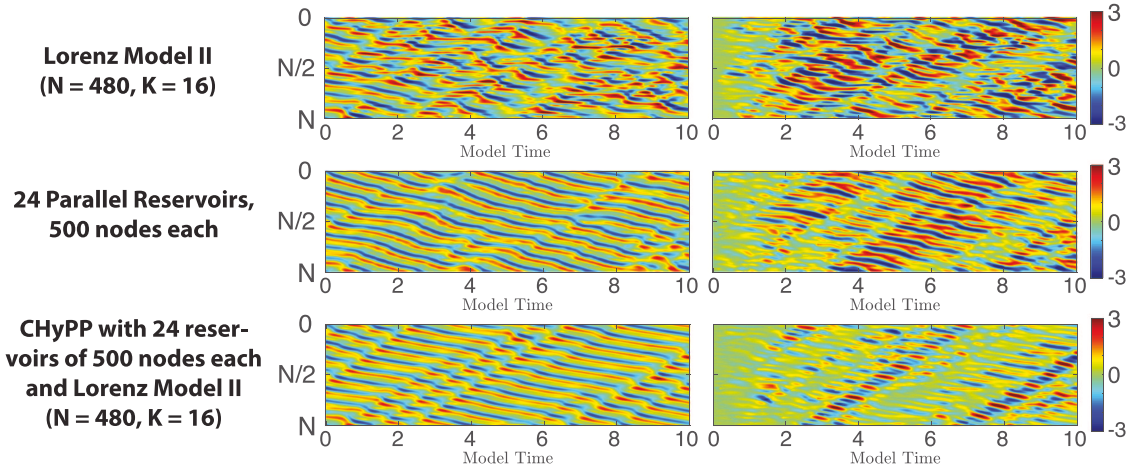

FIG. 12. We plot the predictions of the normalized true dynamics from Lorenz Model III (displayed in Fig. 10), where corresponding to the Model II grid, we have measured *only every second* spatial grid point of Model III during training. We plot the spatial grid point along the vertical axis and the model time on the horizontal axis. The plots on the left show the predictions made using each of the specified methods, while the plots on the right show the difference between the true dynamics and the prediction (i.e., the prediction error). The parallel reservoir-only prediction used its optimized value of $s = 0.25$, while CHyPP used its optimized value of $s = 0.08$.

of this order as one goes around a mid-latitude circle. Figures 11–14 show predictions of the true Lorenz Model III dynamics (in Fig. 10) for grid resolutions of varying coarseness. In particular, while the truth (Fig. 10) is obtained from Model III with $N = 960$ grid points, the number of Model II grid points is $N = 960$, 480, 240, and 120 for Figs. 11, 12, 13, and 14, respectively. Also note that the measurements are taken from the Model III result only at the Model II grid points. For both the parallel reservoir-only predictions and CHyPP, we fix the ratio of the number of nodes per reservoir to the number of grid points each reservoir predicts to be $D_r/Q = 25$. We choose the Lorenz Model III time units rather than the Lyapunov time as our time scale for these plots since the largest Lyapunov exponent of Model III corresponds to the small-spatial scale, short-time scale dynamics and thus will not be relevant to our intended goal of forecasting the large-spatial scale, long-time scale dynamics.

We find that CHyPP significantly outperforms each of its component methods and that this is true for all of the grid resolutions we tested. We note that, unlike in the Model II and parallel reservoir-only predictions, the prediction error in CHyPP seems to appear locally in a small region and manifests as slanted streaks in the

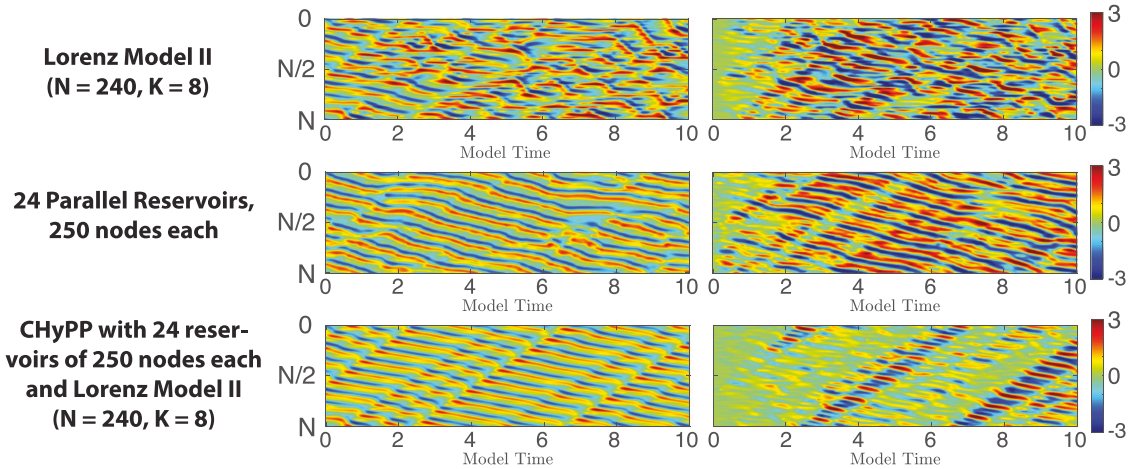

FIG. 13. We plot the predictions of the normalized true dynamics from Lorenz Model III (displayed in Fig. 10), where corresponding to the Model II grid, we have measured *only every fourth* spatial grid point of Model III during training. We plot the spatial grid point along the vertical axis and the model time on the horizontal axis. The plots on the left show the predictions made using each of the specified methods, while the plots on the right show the difference between the true dynamics and the prediction (i.e., the prediction error). The parallel reservoir-only prediction used its optimized value of $s = 0.25$, while CHyPP used its optimized value of $s = 0.017$.

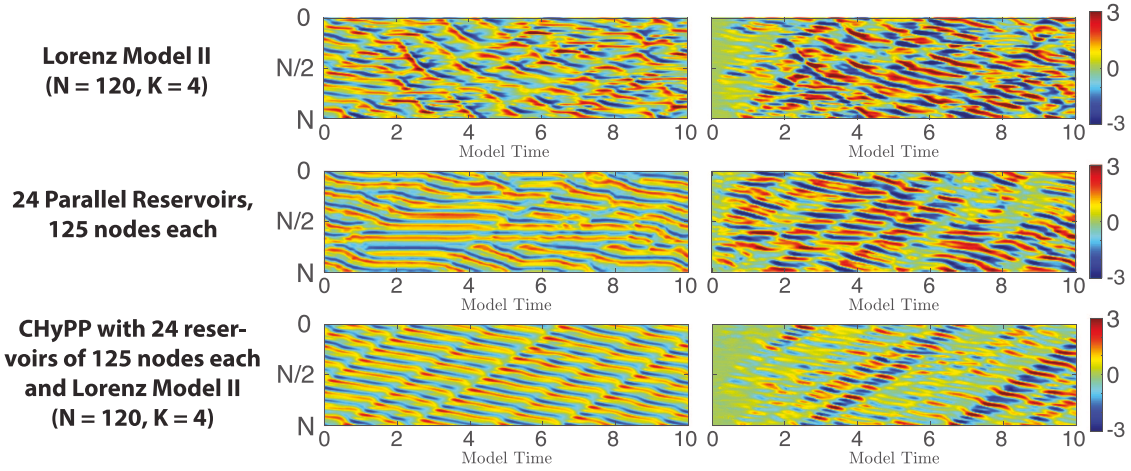

FIG. 14. We plot the predictions of the normalized true dynamics from Lorenz Model III (displayed in Fig. 10), where corresponding to the Model II grid, we have measured *only every eighth* spatial grid point of Model III during training. We plot the spatial grid point along the vertical axis and the model time on the horizontal axis. The plots on the left show the predictions made using each of the specified methods, while the plots on the right show the difference between the true dynamics and the prediction (i.e., the prediction error). The parallel reservoir-only prediction used its optimized value of $s = 0.25$, while CHyPP used its optimized value of $s = 0.0155$.

error plots in Figs. 11–14. We have verified that the slant of these streaks corresponds to the group velocity of the dominant wave motion (wavenumber ≈7). For all predictions made, we again calculate a valid time of prediction; however, since in this case early error growth using CHyPP is local in space and affects only a small part of the prediction domain, we choose to use a higher valid time error threshold of 0.85 (∼ 60% of the error saturation value) so that

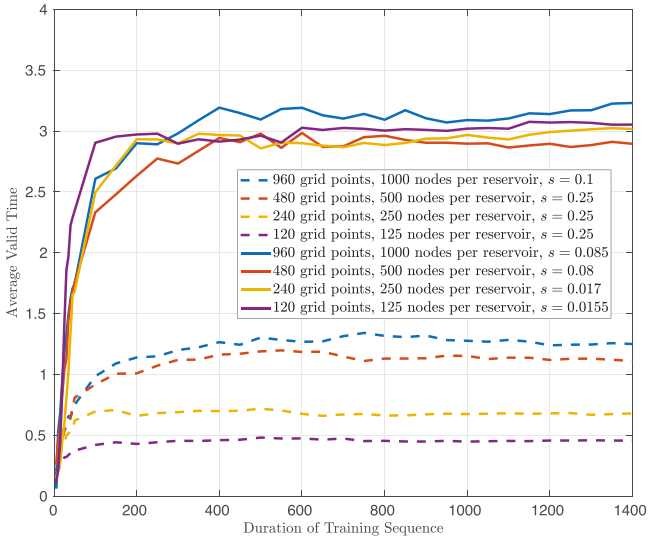

FIG. 15. We plot the average valid time for each prediction as a function of the length of training data used. Dashed lines display the results of parallel reservoir-only predictions, whereas solid lines display results from CHyPP where the knowledge-based model is Lorenz Model II with $N =$ number of grid points, $K = N/30$, and $F = 15$. Each prediction method uses 24 reservoirs.

the valid time metric reflects when error is present in the entire prediction domain. Figure 15 shows the valid time averaged over 100 predictions from different initial conditions vs training time for different grid resolutions and reservoir sizes. The dashed lines are parallel reservoir-only predictions, while the solid lines are CHyPP predictions. In this figure, the reservoir sizes are scaled in proportion to the number of Model II grid points used. We find that, while the quality of the scaled parallel reservoir-only prediction degrades significantly as the grid resolution decreases, the quality of the scaled CHyPP prediction degrades much more slightly from an average valid time of 3.23 at full resolution to 3.05 at 1/8 resolution. The Model II valid time is essentially constant at ∼ 0.95, corresponding to the fact that all of the grid spacings tested are well below the characteristic Model II spatial scale. We see from Fig. 15 that the parallel reservoir-only prediction and CHyPP prediction appear to reach valid time saturation at about the same training data length for each grid resolution.

V. CONCLUSION AND DISCUSSION

In this paper, we address the general goal of utilizing machine learning to enable expanded capability in the forecasting of large, complex, spatiotemporally chaotic systems for which an imperfect knowledge-based model exists. Some typical common sources of imperfection in such a knowledge-based model are unresolved subgrid-scale processes and lack of first principles knowledge or computational ability for modeling some necessary aspect or aspects of the physics. The hope is that these "imperfections" can be compensated for by use of measured time series data and machine learning. The two main foreseeable difficulties in realizing this hope are how to effectively combine the machine learning component with the knowledge-based component in such a way that they mutually enhance each other, and how to promote feasible scaling of the machine learning requirements with respect to its

computational cost and necessary amount of training data. Note that addressing the first of these issues necessarily lessens the difficulty of dealing with the second issue, since good use of any valid scientific knowledge of the system being forecast can potentially reduce the amount of learning required from the machine learning component.

To address these two issues, we propose a methodology (CHyPP) that combines two previously proposed techniques: (i) a hybrid utilization of an imperfect knowledge-based model with a single machine learning component[6,18] and (ii) a parallel machine learning scheme using many spatially distributed machine learning devices.[5] We note that (ii) applies to large spatial systems with the common attribute of what we have called "local short-time causal interactions." Numerical tests of our proposed combination of (i) and (ii) are presented in Secs. III and IV and demonstrate good quality prediction that is scalable with size (Figs. 6 and 7), reduced required length of training data (Fig. 8), and ability to compensate for unresolved subgrid-scale processes (Figs. 10–15).

In this paper, our proof-of-principle problems have been one-dimensional in space with relatively simple mathematical formulations. We note, however, that the CHyPP methodology is readily applicable to higher dimensions and more complicated situations. For example, we are currently in the process of applying CHyPP to global atmospheric weather forecasting for which the atmospheric state is spatially three-dimensional and the system is strongly inhomogeneous, e.g., due to the presence of complex geographic features (including continents, mountains, and oceans), as well as the latitudinal variation of solar heating.

Finally, we note that in many situations, such as operational weather forecasting, predictions are produced cyclically once every cycle time interval. To do this, an accurate estimation of the initial condition is required at the beginning of each cycle. The process of formulating accurate initial conditions is called "data assimilation" and is vital to account for sparse and uncertain measurements. Once every cycle time interval, a set of forecasts are produced for a set of forecast times, where one of these times is equal to the cycle time interval. Then, at the beginning of the next cycle, the previous forecast for the state at the cycle time interval is combined with state measurements taken during the previous cycle and their corresponding estimated uncertainty to obtain an estimate of the current state of the system to be predicted. This state estimate is then used as an initial condition for a forecast model making the next set of predictions. The problem of formulating data assimilation for our improved prediction model (CHyPP) awaits further study, which we are currently pursuing.

In conclusion, we have shown that data-assisted forecasting via parallel reservoir computing and an imperfect knowledge-based model can significantly improve prediction of a large spatiotemporally chaotic system over existing methods. This method is scalable to even larger systems, requires significantly less training data than previous methods to obtain high quality predictions, is able to effectively utilize a knowledge-based forecast of information propagation between local regions to improve prediction quality, and effectively provides a means of using data to compensate for unresolved subgrid-scale processes (playing a role akin to traditional analysis-based closure schemes).

ACKNOWLEDGMENTS

We thank Sarthak Chandra for his helpful discussion and comments. This work was supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. DARPA-PA-18-01(HR111890044).

DATA AVAILABILITY

The data that support the findings of this study are available from the corresponding author upon reasonable request. Our software implementation of CHyPP in MATLAB is publicly available at https://github.com/awikner/CHyPP.

REFERENCES

[1]Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, L. Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, and J. Dean, "Google's neural machine translation system: Bridging the gap between human and machine translation," arXiv:1609.08144 [cs] (2016).
[2]I. Wallach, M. Dzamba, and A. Heifets, "AtomNet: A deep convolutional neural network for bioactivity prediction in structure-based drug discovery," arXiv:1510.02855 [cs, q-bio, stat] (2015).
[3]Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature **521**, 436–444 (2015).
[4]P. Bauer, A. Thorpe, and G. Brunet, "The quiet revolution of numerical weather prediction," Nature **525**, 47–55 (2015).
[5]J. Pathak, B. Hunt, M. Girvan, Z. Lu, and E. Ott, "Model-free prediction of large spatiotemporally chaotic systems from data: A reservoir computing approach," Phys. Rev. Lett. **120**, 024102 (2018).
[6]J. Pathak, A. Wikner, R. Fussell, S. Chandra, B. R. Hunt, M. Girvan, and E. Ott, "Hybrid forecasting of chaotic processes: Using machine learning in conjunction with a knowledge-based model," Chaos **28**, 041101 (2018).
[7]H. Jaeger, "The echo state approach to analysing and training recurrent neural networks-with an erratum note," in *GMD Technical Report 148* (German National Research Center for Information Technology, Bonn, 2001), p. 34.
[8]W. Maass, T. Natschläger, and H. Markram, "Real-time computing without stable states: A new framework for neural computation based on perturbations," Neural Comput. **14**, 2531–2560 (2002).
[9]H. Jaeger and H. Haas, "Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication," Science **304**, 78–80 (2004).
[10]Z. Lu, J. Pathak, B. Hunt, M. Girvan, R. Brockett, and E. Ott, "Reservoir observers: Model-free inference of unmeasured variables in chaotic systems," Chaos **27**, 041102 (2017).
[11]M. Inubushi and K. Yoshimura, "Reservoir computing beyond memory-nonlinearity trade-off," Sci. Rep. **7**, 10199 (2017).
[12]S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural Comput. **9**, 1735–80 (1997).
[13]K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (Association for Computational Linguistics, Doha, 2014), pp. 1724–1734.
[14]P. R. Vlachas, J. Pathak, B. R. Hunt, T. P. Sapsis, M. Girvan, E. Ott, and P. Koumoutsakos, "Forecasting of spatio-temporal chaotic dynamics with recurrent neural networks: A comparative study of reservoir computing and backpropagation algorithms," arXiv:1910.05266 [physics] (2019).
[15]Q. Vinckier, F. Duport, A. Smerieri, K. Vandoorne, P. Bienstman, M. Haelterman, and S. Massar, "High-performance photonic reservoir computer based on a coherently driven passive cavity," Optica **2**, 438 (2015).
[16]B. Penkovsky, L. Larger, and D. Brunner, "Efficient design of hardware-enabled reservoir computing in FPGAs," J. Appl. Phys. **124**, 162101 (2018).

[17]G. Tanaka, T. Yamane, J. B. Héroux, R. Nakane, N. Kanazawa, S. Takeda, H. Numata, D. Nakano, and A. Hirose, "Recent advances in physical reservoir computing: A review," Neural Netw. **115**, 100–123 (2019).
[18]Z. Y. Wan, P. Vlachas, P. Koumoutsakos, and T. Sapsis, "Data-assisted reduced-order modeling of extreme events in complex dynamical systems," PLoS ONE **13**, e0197704 (2018).
[19]A. Tikhonov and V. Arsenin, *Solutions of Ill-Posed Problems*, Scripta Series in Mathematics (Winston, 1977).
[20]MATLAB version9.5.0.944444 (R2018b) (The MathWorks Inc., Natick, MA, 2018).
[21]Y. Saad and M. H. Schultz, "GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems," SIAM J. Sci. Stat. Comput. **7**, 856–869 (1986).
[22]T. Zhang, "Solving large scale linear prediction problems using stochastic gradient descent algorithms," in *Proceedings of the Twenty-First International Conference on Machine Learning*, ICML '04 (Association for Computing Machinery, Banff, 2004), p. 116.
[23]I. Goodfellow, Y. Bengio, and A. Courville, "Convolutional networks," in *Deep Learning* (MIT Press, 2017), p. 321–361.
[24]Y. Kuramoto, "Diffusion-induced chaos in reaction systems," Prog. Theor. Phys. Suppl. **64**, 346–367 (1978).
[25]G. I. Sivashinsky, "Nonlinear analysis of hydrodynamic instability in laminar flames—I. Derivation of basic equations," Acta Astronaut. **4**, 1177–1206 (1977).
[26]S. M. Cox and P. C. Matthews, "Exponential time differencing for stiff systems," J. Comput. Phys. **176**, 430–455 (2002).
[27]C. Meneveau and J. Katz, "Scale-invariance and turbulence models for large-eddy simulation," Annu. Rev. Fluid Mech. **32**, 1–32 (2000).
[28]E. N. Lorenz, "Designing chaotic models," J. Atmos. Sci. **62**, 1574–1587 (2005).
[29]E. Lorenz, "Predictability: A problem partly solved," in *Seminar on Predictability, Shinfield Park, Reading, 4–8 September 1995* (ECMWF, 1995), Vol. 1, pp. 1–18.