Reservoir Computing for Forecasting Large Spatiotemporal Dynamical Systems

Jaideep Pathak and Edward Ott

Abstract Forecasting of spatiotemporal chaotic dynamical systems is an important problem in several scientific fields. Crucial scientific applications such as weather forecasting and climate modeling depend on the ability to effectively model spatiotemporal chaotic geophysical systems such as the atmosphere and oceans. Recent advances in the field of machine learning have the potential to be an important tool for modeling such systems. In this chapter, we review several key ideas and discuss some reservoir-computing-based architectures for purely data-driven as well as hybrid data-assisted forecasting of chaotic systems with an emphasis on scalability to large, high-dimensional systems.

1 Motivation

This chapter is motivated by problems in the forecasting of large, complex, spatiotemporally chaotic systems and by the possibility that machine learning might be a useful tool for the significant improvement of such forecasts. Examples of the type of potential tasks we have in mind are forecasting ocean conditions; forecasting conditions in the solar wind, Earth's magnetosphere, and ionosphere (so-called 'space weather', important for Earth-orbiting spacecraft, GPS accuracy, power-grid disruptions, etc.); forecasting the spatial distribution of plant growth in response to environmental changes; forecasting the development of forest fires and their response to fire fighting strategies; and weather forecasting.

Focusing on weather forecasting as perhaps the most important such example, we note the following two relevant points: (i) weather forecasts impact the lives of many millions of people, e.g., by providing warnings of destructive events, like hurricanes or snowstorms; (ii) currently used weather forecasting employs physics-based models (the equations of fluid dynamics, radiative heat transfer, etc.), plus

J. Pathak (✉) · E. Ott
University of Maryland, College Park, MD, USA
e-mail: jpathak@umd.edu

E. Ott
e-mail: edott@umd.edu

K. Nakajima and I. Fischer (eds.), *Reservoir Computing*, Natural Computing Series,
https://doi.org/10.1007/978-981-13-1687-6_6

geographical knowledge of mountains, oceans, etc. The models in (ii), however, have substantial errors, which, for example, may arise due to imperfect modeling of crucial subgrid-scale dynamics (like clouds, turbulent atmospheric motions, and interactions with small-scale geographic features).

Can machine learning from data potentially correct such knowledge deficits and thus contribute to significant improvement of forecasts? For other recent work which addresses the issue of using machine learning for analyzing and forecasting spatiotemporal dynamical systems, see Brunton et al. (2016), Lusch et al. (2018), Raissi et al. (2019), Vlachas et al. (2018), and Wan et al. (2018). In this paper, focusing on reservoir computing, we discuss and summarize some recent research that may be relevant to this question.

2 Background: Prediction of 'Small' Chaotic Systems

Machine Learning (ML) prediction of the ergodic chaotic evolution of a dynamical system was considered by Jaeger and Haas (2004) in the context of reservoir computing (Jaeger 2001). The basic idea of their scheme is illustrated (in its discrete time version) in Fig. 1. Given time-series training data $\mathbf{u}(n)$, obtained from sampling measurements of some unknown chaotic dynamical system on an ergodic attractor (with a sampling time interval Δ), for $n = -T, -T + 1, -T + 2, \ldots, -1$, the ML system is trained to output $\mathbf{u}(n + 1)$, when $\mathbf{u}(n)$ is the input (Fig. 1). If the vector state of the unknown dynamical system is denoted by $\mathbf{x}$, we can represent the measurements by a 'measurement function' $\mathbf{H}$ such that the measurement vector $\mathbf{u}$ is given by

$$\mathbf{u}(n) = \mathbf{H}(\mathbf{x}(t)), \qquad t = n\Delta. \tag{1}$$

Since the dimension of $\mathbf{u}$ may be less than twice the dimension of the attractor of the dynamical system in $\mathbf{x}$-space, Eq. 1 may not be an embedding (Sauer et al. 1991), and, to compensate for this, it is important that the ML system has memory. That is, the current state of the ML device depends on its current input and on the history of the inputs. (Memory can also be realized or supplemented by incorporating time-delayed measurements as additional components of $\mathbf{u}(t)$; however, for simplicity we will not further consider this possibility here.)

As an example, assume that the goal is to predict the future value of the measurements. Following Jaeger and Haas (2004), when, after the training phase (Fig. 1a), the time to predict comes, the input from the measured state is no longer available and, as shown in Fig. 1b, is replaced by the ML system output; i.e., the output is fed back into the input. Thus, at the beginning step of the forecasting phase, $\mathbf{u}(0)$ is made the input, which, if all goes well, then produces a new (forecasted) output $\mathbf{u}(1)$, which, when fed back, outputs a forecast for $\mathbf{u}(2)$, which when fed back outputs a forecast for $\mathbf{u}(3)$, and so on. Of course, there is always some small error in the output (e.g., if the input is $\mathbf{u}(0)$, the output is only approximately $\mathbf{u}(1)$), and due to the

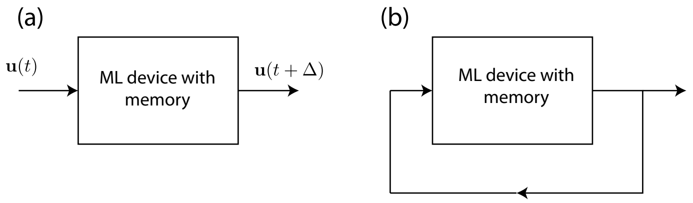

Fig. 1 Schematic illustration of the **a** training phase, and **b** prediction phase for a simple ML forecasting system

assumed chaos of the dynamical system generating the measurements, these errors build up as the feedback loop is successively traversed. Thus, as is typical for chaotic processes, the prediction accuracy eventually breaks down. So good prediction can only be expected for several Lyapunov times.

Note that the closed-loop system shown in Fig. 1a may itself be regarded as an autonomous dynamical system. Thus in Lu et al. (2018), Lu et al. have employed dynamical systems theory concepts (especially the concepts of 'generalized synchronization' Afraimovich et al. 1986; Kocarev and Parlitz 1996; Pecora et al. 1999; Rulkov et al. 1995 and Lyapunov exponents Abarbanel 2012; Kantz and Schreiber 2004; Ott 2002; Ott et al. 1994) to analyze conditions on the ML system that make for good reproduction of the dynamics of the unknown system that produces the data.

With regard to the time step Δ, one might have the following question. Assuming that one is interested in forecasting a chaotic process forward by an amount of time T, why not simply set $\Delta = T$ and do one prediction step (thus eliminating the need for the closed-loop configuration in Fig. 1b)? The answer is that for typical cases, one is often interested in prediction times T that may be as large as several Lyapunov times (a Lyapunov time is a typical time it takes a small orbit perturbation to grow by a factor of e). In such cases, with $\Delta = T$, small changes in $\mathbf{u}(n)$ can lead to relatively large changes in $\mathbf{u}(n+1)$. Thus, the functional relationship that the ML system is trained to learn is relatively complex ('wiggly') making its task relatively hard. Accordingly, it has been found that using smaller Δ with the feedback loop as shown in Fig. 1b is advantageous.

One aspect of the forecasting scheme illustrated in Fig. 1 is that the various versions of ML can in principle be employed. Since memory is typically required, and is, in any case, expected to be advantageous for prediction tasks, the two most natural candidates for consideration are reservoir computing (as in the paper of Jaeger and Haas 2004) and Long Short-Term Memory (Hochreiter and Schmidhuber 1997) (as in the paper of Vlachas et al. 2018). In this chapter, we consider reservoir computing due to its appreciably shorter training times and its potential for advantageous physical implementations discussed in other chapters of this book.

3 Machine Learning and the Forecasting of Large, Complex, Spatiotemporally Chaotic Systems

The scheme (Jaeger and Haas 2004) described in Fig. 1 and Sect. 2 works well for small to moderate size systems. However, we find that straightforward scaling of, e.g., a reservoir computing implementation of Fig. 1 to very large size results in requirements that appear to be unfeasible or, at least, very demanding, in practical terms (e.g., with respect to the reservoir size required, amount of training data, and computations for training). Thus we seek ways of mitigating this problem. Specifically, we wish to apply prior physical knowledge of the system to be forecasted and integrate this prior knowledge with a machine learning approach via a suitable prediction system architecture. In particular, we consider two types of prior knowledge as described below.

First, we note that information in spatially extended physical systems generally propagates at a finite speed. Thus, a perturbation applied at some point in space will not immediately affect the state at some distant point. We refer to this as 'the locality of short-term causal interactions' (LSTCI). Assuming, for illustrative purposes, that space is one dimensional, if we want to predict the state at time $t + \Delta$ in the region $x_0 - l_0 < x < x_0 + l_0$ from the state at time t, we only need to consider the state at time t within the region, $x_0 - (l_0 + d) < x < x_0 + (l_0 + d)$, where d is large enough such that information affecting the prediction of the state in the region, $x_0 - l_0 < x < x_0 + l_0$, does not propagate fast enough to move a distance d over the one-step prediction time Δ. Thus, a parallel approach (Sect. 4) can be employed in which multiple ML systems predict $\mathbf{u}$ in corresponding limited overlapping spatial regions where the lengths of the overlap between regions are at least d. This will be discussed in Sect. 4.3 (note that this consideration provides an added motivation for considering Δ to be small).

The second type of physical knowledge comes from knowledge-based modeling, typically in the form of inaccurate partial differential equations like those discussed in Sect. 2 in the context of weather forecasting. In Sect. 5, we discuss a hybrid technique that utilizes both an imperfect knowledge-based model and a relatively small Reservoir Computing ML forecaster (see Wan et al. 2018 for a similar implementation using Long Short-Term Memory ML). In the training of the hybrid system, the state variables of *both* the ML system and the knowledge-based system are combined via a set of adjustable 'weights' in such a way as to very closely fit the desired prediction system outputs as determined by the training data. Thus, we view the training as being designed to take the best aspects of the predictions of the ML component and the knowledge-based component and to combine these good aspects in a semi-optimal fashion. Indeed, as we later show (Sect. 5), even in a case where the knowledge-based system error and the relatively small size of the ML component were such that each acting alone gave relatively worthless forecasts, that, when incorporated into our hybrid scheme, excellent forecasts can result. Furthermore, as we will document elsewhere, the machine learning component typically requires less training data for use in the hybrid scheme than would be the case for a much larger, pure machine learning system.

In Sect. 6, we discuss and illustrate a prediction system architecture for combining the parallel and hybrid schemes so as to create a methodology that is potentially scalable to very large complex systems. In this combined scheme, we envision the knowledge-based component of this combined system to be global and not based on the LSTCI assumption (e.g., like models currently used for weather prediction).

4 Distributed Parallel Prediction

In this section, we describe how to efficiently train a Reservoir Computer to predict time-series from high-dimensional spatiotemporal chaotic systems. This scheme was introduced in Pathak et al. (2018a). As mentioned in Sect. 3, we will exploit the locality of short-term causal interactions (LSTCI), present in many spatiotemporal systems of interest, to divide the computational task over many independent computing units or 'cores'. The key idea behind this division is our assumption that the near-term future of the state of a particular spatial region of the spatiotemporal system is only affected by dynamics occurring nearby (in a spatial sense), and the dynamics occurring far away from it has no effect. This assumption presupposes the absence of short-term long-range interactions in the spatiotemporal system.

For simplicity, in most of what follows, we consider a spatiotemporal dynamical system defined by some set of equations evolving a scalar state variable $y(x, t)$ forward in time (t) in a one-dimensional spatial domain (x) with periodic boundary conditions. Thus, $x \in [0, L)$ and $y(x + L, t) = y(x, t)$. We assume that the measurement vector $\mathbf{u}(n)$, is $K-$dimensional with each scalar element of $\mathbf{u}(n)$ being the state variable $y(x, t)$, measured at regular intervals of time Δ and over a uniformly spaced spatial grid with K grid points: $x = (L/K), (2L/K), (3L/K), \ldots, L$.

4.1 *Partitioning the Spatial Grid*

We thus have a set of K time series $u_k(n)$, $1 \leq k \leq K$ on K grid points where $t = n\Delta$ and n is an integer. The measurement vector $\mathbf{u}(n)$ is thus a $K-$vector whose kth scalar element is $u_k(n)$. As shown in Fig. 2, we then use the $K-$dimensional measurement vector $\mathbf{u}(n)$ to form a set of M vectors $\{\mathbf{v}_i(n)\}_{i=1,2,\ldots,M}$ where each such vector i has dimension $(K/M) + 2l$, and its entries consist of the y values at time n in *overlapping* regions where each region i has K/M central nodes supplemented by overlap buffer regions to its left and right of length l nodes each (e.g., in the schematic illustration shown in Fig. 2, $l = 2$ and $(K/M) = 4$). The choice of l is made using our LSTCI assumption and requiring that the time Δ prediction of the K/M central nodes of vector $\mathbf{v}_i(n)$ is (to a very good approximation) not influenced by nodal states not included in $\mathbf{v}_i(n)$.

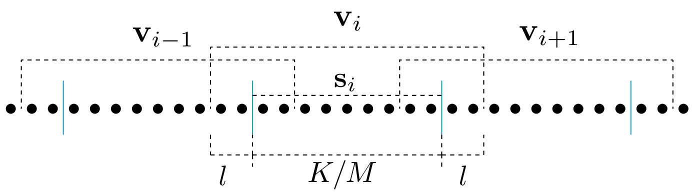

Fig. 2 Partitioning scheme

The K grid points are also partitioned into M *non-overlapping* groups with K/M-dimensional state vectors, $\mathbf{s}_i$ for $i = 1, 2, \ldots, K/M$, as illustrated in Fig. 2 (from Fig. 2, we see that $\mathbf{v}_i$ becomes $\mathbf{s}_i$ when l is set to zero). The task of obtaining the time Δ prediction of each such group is assigned to M separate ML systems which will be trained to learn the local group dynamics and predict the future state of the time series at those grid points. Making use of our LSTCI assumption, we suppose that Δ is small enough that $\mathbf{s}_i(n+1)$ depends on $\mathbf{s}_i(n+1)$, but is independent of $\mathbf{v}_{i\pm k}(n)$ for $k \geq 1$.

Note that l and M are 'hyperparameters' of our model and can be tuned while optimizing with respect to factors such as computational cost and prediction results. We use the term hyperparameters to denote a small set of parameters that are not adjusted via the training procedure and that characterize gross overall features of the ML device (e.g., the hyperparameters we deal with are the amount of nonlinearity and memory, the reservoir size, input coupling strength, and training regularization). Hyperparameters are often set by the user on an empirical, trial-and-error basis so as to achieve 'good' results on a test data set (typically, the test data set is separate from the training data set so as to ensure generalization of training). Hyperparameters can also be set more systematically via optimization techniques.

The above description (e.g., Fig. 2) is for the case of a spatially one-dimensional system. The generalization to higher dimensions is straightforward and is indicated for the two-dimensional case in Fig. 3, in which the solid black lines divided space into square patches labeled by the two subscripts (i, j), the vector $\mathbf{s}_{i,j}$ (analogous to $\mathbf{s}_i$ in the one-dimensional case) specifies the system state within patch (i, j), and the vector $\mathbf{v}_{i,j}$ (analogous to $\mathbf{v}_i$ in the one-dimensional case) specifies the system state in the expanded overlapping regions indicated by the dashed square in Fig. 3. (A similar scheme was also used in Zimmermann and Parlitz 2018 to infer unmeasured state variables.)

4.2 Training

Now specializing in the case of Reservoir Computing, using a simple procedure outlined in Jaeger and Haas (2004), we generate M reservoir systems each based on a $D \gg K/M$ node recurrent artificial neural network characterized by a weighted

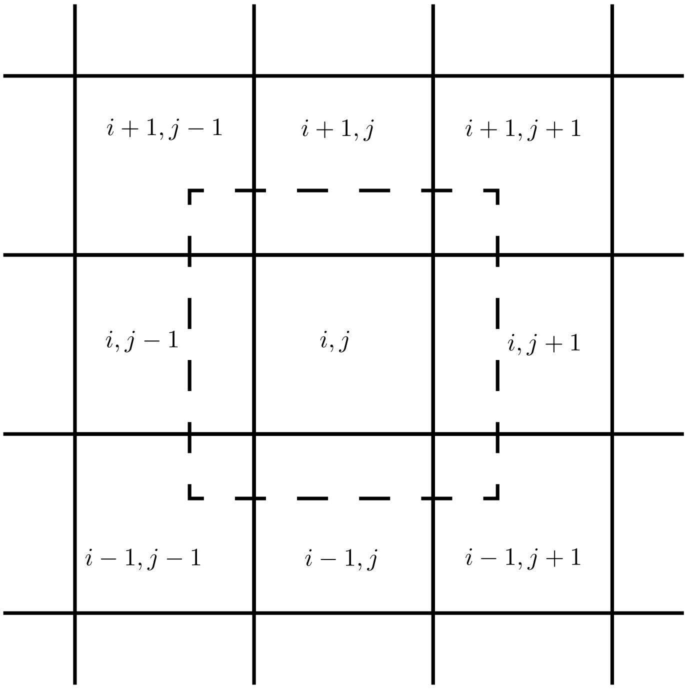

Fig. 3 Spatial regions $\mathbf{s}_{i,j}$ and $\mathbf{v}_{i,j}$ for two-dimensional ML parallel prediction. See also Zimmermann and Parlitz (2018)

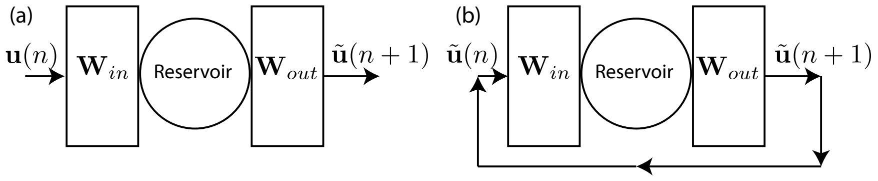

Fig. 4 **a** Open-loop 'training' configuration; **b** Closed-loop 'prediction' configuration for the reservoir computing prediction scheme

adjacency matrix $\mathbf{A}_i$. Assuming that initial start-up transient activity is omitted, during the training phase, which we take to run from $n = -T$ to $n = -1$, each of the M reservoir networks (Fig. 5) evolves according to the following equation:

$$\mathbf{r}_i(n+1) = \tanh\left[\mathbf{A}_i\mathbf{r}_i(n) + \mathbf{W}_{\text{in},i}\mathbf{v}_i(n)\right], \tag{2}$$

where $\mathbf{r}_i$ is the $D-$vector whose elements are the states (which are here taken to be scalars) of each of the network nodes, and the $D \times [(K/M) + 2l]$ matrix $\mathbf{W}_{\text{in},i}$ couples the ith input training vector $\mathbf{v}_i$ to nodes of the recurrent reservoir network. The reservoir states $\mathbf{r}_i(n)$, $-T \leq n \leq -1$ are stored in a matrix $\mathbf{R}_i$, such that the T

columns of $\mathbf{R}_i$ are the vectors $\mathbf{r}_i(n)$. The state vector $\mathbf{r}_i$ is then used to produce an output vector whose dimension is K/M. In the simplest case (not necessarily the only choice), this is done via a $(K/M) \times D$ output coupling matrix $\mathbf{W}_{\text{out},i}$, such that the output is $\mathbf{W}_{\text{out},i}\mathbf{r}_i$. We regard the parameters formed by the elements of $\mathbf{W}_{\text{in},i}$ and $\mathbf{A}_i$ as fixed and use only the parameters of the output coupling function, i.e., the matrix elements of $\mathbf{W}_{\text{out},i}$ for training. That is, we adjust $\mathbf{W}_{\text{out},i}$ so that a desired training output results. In accord with the left panel of Fig. 1, we desire the output to approximate $\mathbf{s}_i(n+1)$ when the training input is $\mathbf{v}_i(n)$,

$$\mathbf{W}_{\text{out},i}\mathbf{r}_i(n+1) \simeq \mathbf{s}_i(n+1), \tag{3}$$

for $-T \leq n \leq -1$. Calculating a matrix $\mathbf{W}_{\text{out},i}$ that satisfies Eq. (3) is referred to as 'training' the neural network. This is illustrated in Fig. 5. In Eq. 3, the training problem is incompletely defined as we have not specified what exactly we mean by the '$\simeq$' sign. The simplest possible choice is to require that the right-hand side and left-hand side of Eq. (3) be approximately equal in the sense of their ℓ_2 norms. Thus, one might choose the matrix $\mathbf{W}_{\text{out},i}$ that minimizes the sum of squared deviations of the output from its desired target value,

$$\epsilon = \sum_{n=-T}^{-1} \|\mathbf{W}_{\text{out},i}\mathbf{r}_i(n+1) - \mathbf{s}_i(n+1)\|^2. \tag{4}$$

However, this can often be problematic, and to avoid overfitting to the training data and better promote the generalizability of the training to cases beyond the training data, a regularization procedure is typically employed. To this end, $\mathbf{W}_{\text{out},i}$ is often

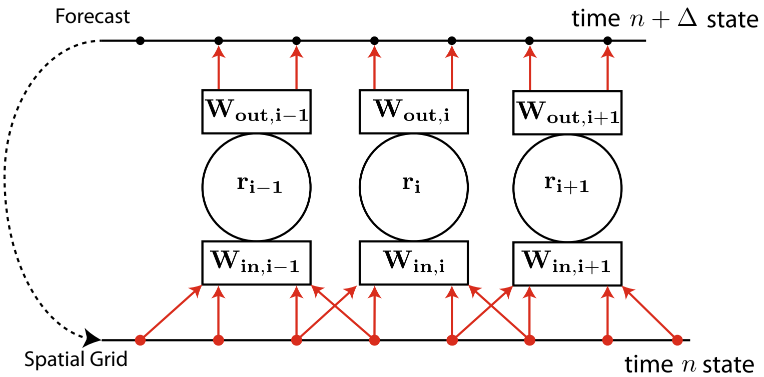
Forecast
time $n+\Delta$ state
$\mathbf{W}_{\text{out},i-1}$
$\mathbf{W}_{\text{out},i}$
$\mathbf{W}_{\text{out},i+1}$
$\mathbf{r}_{i-1}$
$\mathbf{r}_i$
$\mathbf{r}_{i+1}$
$\mathbf{W}_{\text{in},i-1}$
$\mathbf{W}_{\text{in},i}$
$\mathbf{W}_{\text{in},i+1}$
Spatial Grid
time n state

Fig. 5 Parallelized prediction scheme. $(K/M) = 2$, $l = 1$. The open-loop configuration corresponds to this figure with the dashed line ignored. The closed-loop configuration is represented by the dashed line which indicates that the required inputs to the reservoirs are taken from the corresponding outputs

required to minimize $\epsilon + \epsilon_r$, where ϵ_r is a term that penalizes excessively large values of the training parameters,

$$\epsilon_r = \beta \sum_j \|\mathbf{W}_{\text{out},i}\|_{:j}^2. \tag{5}$$

In Eq. (5), $\|\cdot\|_{:j}$ denotes the ℓ_2 norm of the jth column of a matrix and β is a hyperparameter called the regularization constant that determines the strength of the regularization term. Note that this minimization is a standard linear regression problem with a well-known matrix-based solution (e.g., see Eq. (6) below). Alternatively, if the matrix inverse is computationally onerous (as might be the case for very large D), one can minimize $(\epsilon + \epsilon_r)$ by the steepest descent. Another device, used in what follows to increase the expressive power of the reservoir computing network without sacrificing the training simplicity afforded by linear regression is to construct a vector $\mathbf{r}^*$ from $\mathbf{r}$ such that the elements r_j of $\mathbf{r}$ and r_j^* of $\mathbf{r}^*$ are related by the following rule: $r_j^* = r_j^2$ if j is even and $r_j^* = r_j$ if j is odd. Putting all of this together, we obtain the expression for $\mathbf{W}_{\text{out},i}$,

$$\mathbf{W}_{\text{out},i} = \left(\mathbf{R}_i^* \mathbf{R}_i^{*T} + \beta \mathbf{I}\right)^{-1} \mathbf{R}_i^* \mathbf{S}_i^T \tag{6}$$

where $\mathbf{R}_i^*$ is the $D \times T$ matrix with columns given by the vectors $\mathbf{r}_i^*(n)$, and $\mathbf{S}_i$ is the $(N/M) \times T$ matrix whose columns are the training data time-series vectors $\mathbf{s}_i(n)$. Note that each of the individual reservoir systems i is trained independently, and thus training is parallelized. Furthermore, the input and output dimensions can be much smaller than the size of the global measurement state. Thus, the individual parallel reservoirs can be much smaller than would be the case without making use of LSTCI. Having determined $\mathbf{W}_{\text{out},i}$, we rewrite Eq. (3) as

$$\mathbf{W}_{\text{out},i} \mathbf{r}_i^*(n+1) = \tilde{\mathbf{s}}_i(n+1) \tag{7}$$

where $\tilde{\mathbf{s}}_i(n)$ denotes the machine learning approximation to the true group i state vector $\mathbf{s}_i(n)$. Similarly, for later reference, we will also use $\tilde{\mathbf{v}}_i$ to denote the corresponding approximation to $\mathbf{v}_i$.

For later reference, in all of the numerical experiments reported in what follows, adjacency matrices are random Erdős–Renyi matrices of fixed average degree that are scaled by multiplication by a constant so as to fix the matrix spectral radius (magnitude of its largest eigenvalue) at a pre-selected value denoted by ρ, and the elements of the input coupling matrices are each randomly selected numbers with uniform probability in $[-\sigma, \sigma]$. Both ρ and σ are hyperparameters. For the tasks we study, we find that the reservoir computer performance is largely insensitive to the reservoir network topology. For specificity in the example given in this paper, we use random Erdős–Renyi networks with an average degree equal to 3.

4.3 Prediction

At the end of the minimization procedure described above, we obtain the set of matrices $\mathbf{W}_{\text{out},i}$ that maps the internal state of the reservoir $\mathbf{r}_i(n)$ at a given instant of time to a good approximation of the state of the vector $\mathbf{u}(n)$. Since $\mathbf{r}_i(n)$ is dependent on past inputs $(\mathbf{v}_i(n-k), k \geq 1)$, we hope to have trained the reservoir to perform single step forecasts of step size Δ. We know this to be true on the training data. Whether the training generalizes to 'out-of-sample' data, i.e., the time series $\mathbf{u}(n)$ outside the interval $-T \leq n \leq -1$ is a separate question. This question will be addressed empirically by numerical experiments in sections to follow. In the prediction stage, we re-configure our parallel ML system to generate forecasts for $n > 0$ (see Fig. 5 as follows):

$$\begin{aligned} &\text{step 1:} \qquad \tilde{\mathbf{s}}_i(n) = \mathbf{W}_{\text{out},i}\mathbf{r}_i^*(n) \\ &\text{step 2:} \quad \text{Construct } \tilde{\mathbf{v}}_i(n) \text{ from } \tilde{\mathbf{s}}_i(n), \tilde{\mathbf{s}}_{i\pm1}(n) \\ &\text{step 3: } \mathbf{r}_i(n+1) = \tanh\left[\mathbf{A}_i\mathbf{r}_i(n) + \mathbf{W}_{\text{in},i}\tilde{\mathbf{v}}_i(n)\right] \end{aligned} \tag{8}$$

That is, the one-step-ahead output predictions of $\mathbf{s}_j(n+\Delta)$ for $j = i-1, i, i+1$, are used to construct a prediction for $\mathbf{v}_i(n+\Delta)$, which is then fed back to the input of the reservoir system i, producing a new output prediction of $\mathbf{s}_i(n+2\Delta)$, and this process is cyclically repeated. To summarize, the prediction algorithm in Eq. 8 has three key steps. In the first step, we compute the output of each reservoir network $\tilde{\mathbf{s}}_i(n)$ to get a Δ-step prediction. Next, we construct the overlapping partitions $\tilde{\mathbf{v}}_i(n)$ from $\tilde{\mathbf{s}}_i(n)$, $\tilde{\mathbf{s}}_{i\pm1}(n)$. The vector $\tilde{\mathbf{v}}_i$ is the feedback from the output of the reservoir network to the input. Equation 8 forms an autonomous dynamical system that predicts the future states of the dynamical system that it was trained on.

4.4 Re-synchronization

The prediction scheme described in Sect. 4.3 will be expected to generate accurate predictions for a finite amount of time determined by the fundamental properties of the chaotic dynamical system being predicted, especially the average error e-folding time (the inverse of the largest Lyapunov exponent of the chaotic process of interest Ott 2002). Because of the chaos, the predictions will necessarily diverge from the ground truth after some amount of time. After the predicted trajectory of the dynamical system has diverged far enough from the true trajectory, the predictions made by the reservoir system are no longer accurate enough to be useful. If, after such a divergence, one wishes to restart prediction from some later time, the reservoir system does not have to be re-trained in order to generate accurate new predictions. Rather, we find that it is sufficient to re-synchronize the reservoir network states $\mathbf{r}_i(n)$ with the ground truth by simply running the reservoir networks without feedback according to Eq. (2) for ξ time steps prior to the desired beginning of prediction.

Importantly, we find that the necessary re-synchronization time is very much smaller than the necessary training time $\xi \ll T$. Thus, we emphasize that the output weights $\mathbf{W}_{\mathrm{out},i}$ do not need to be re-computed for subsequent predictions at later times.

4.5 An Example: A Lorenz 96 Model

We consider one of the classes of 'toy' models of atmospheric dynamics proposed in a 1996 paper of Lorenz (1996) and use it as a testbed for our parallelized prediction setup. The particular model we use is defined as a set of interacting scalar variables $X_j(t)$, $1 \le j \le N$ on a spatially periodic grid ($X_{j+N}(t) = X_j(t)$) with the dynamics given by the coupled ordinary differential equations,

$$\frac{dX_j}{dt} = -X_j + X_{j-1}X_{j+1} - X_{j-1}X_{j-2} + F. \tag{9}$$

We numerically integrate Eq. (9) using a standard fourth-order Runge–Kutta scheme and generate simulated time-series data. We sample the simulated data at the time step intervals $\Delta = 0.01$ to generate the training data vectors $\mathbf{u}(n)$ in the interval $-T \le n \le -1$, where $T = 80{,}000$. We also generate a test data set $\mathbf{y}(n)$ in the interval $0 \le n \le 20{,}000$. The training data set is used to train the reservoir computing system according to the scheme outlined in Sect. 4 while the test data set is used to validate the accuracy of the predictions. The validation scheme is as follows:

- Step 1: We generate a random set of 50 'initial time points' n_k, $1 \le k \le 50$, such that $1 \le n_k \le 20{,}000 - \tau - \xi$.
- Step 2: The test data $\mathbf{y}(n)$ is used to synchronize the reservoirs to the true trajectory for ξ time steps between n_k and $n_k + \xi$ according to the synchronization procedure outlined in Sect. 4.4.
- Step 3: The reservoir network is run in prediction mode according to Eq. (8) for τ time steps. We evaluate the prediction error in this interval by comparing the predicted data with the ground truth. The spatially averaged RMS prediction error at time n is evaluated according to

$$\mathbf{e}(n) = \frac{\|\tilde{\mathbf{u}}(n) - \mathbf{u}(n)\|}{\sqrt{\langle \|\mathbf{u}(n)\|^2 \rangle_n}}. \tag{10}$$

- Steps 2 and 3 are repeated for the next point in the set, n_{k+1}.

The reservoir and model hyperparameters are listed in Table 1. Figure 6 shows the results of forecasting a single interval. Panel (a) of Fig. 6 shows a direct numerical solution of Eq. (9) for the value of X_j (represented on a color scale) as a function of spatial position j (vertical axis) and of time Λt plotted along the horizontal axis where Λ denotes the maximum Lyapunov exponent of the chaotic process. (Note the wave-like behavior visible in this pattern. This wave-like behavior is purposely

Table 1 Lorenz 96 prediction hyperparameters

Hyperparameter	Value	Hyperparameter	Value
ρ	0.6	D	5000
σ	0.1	T	80,000
ξ	32	β	10^{-4}
l	2	τ	1000

induced by Lorenz's design of the model so as to mimic the presence of atmospheric Rossby waves.) The largest Lyapunov exponent for this system with parameters $N = 40$, $F = 8$ is $\Lambda = 1.4$. See Karimi and Paul (2010). Panel (b) shows the error in the ML prediction starting at time zero, where the error is the ML predicted value of X_j (plotted in panel (b)) minus the 'true' value of X_j (plotted in panel (a)). We see that, for this case, a useful forecast (error near zero over a significant spatial region) is obtained out to about four Lyapunov times. Panel (c) shows the spatially averaged RMS error $e(t)$ corresponding to panel (b) versus time, showing how that prediction quality degrades with time. In order to illustrate the typical forecasting quality for this system and the variance in the forecast quality on different parts of the attractor, Fig. 7 shows the RMS error for 50 trajectories of length τ (cyan curves) plotted along with the mean (black curve). Additionally, Fig. 8 shows the effect of changing the number (M) of parallel reservoirs (and thus, changing the computational power) on the quality of prediction. We see that longer forecasting times result as M increases. We emphasize that our results are for the case of perfectly accurate measurements, and that prediction quality degrades as the measurements are corrupted by noise. For example, for large enough noise the improvement of prediction with an increase of M from 5 to 20 seen in Fig. 8 might be absent when the noise becomes the prediction limiting factor.

4.6 Another Example: The Kuramoto–Sivashinsky Equation

We now report on tests of our parallel reservoir prediction scheme using the Kuramoto–Sivashinsky system defined by the partial differential equation,

$$\frac{\partial y}{\partial t} + y\frac{\partial y}{\partial x} + \frac{\partial^2 y}{\partial x^2} + \frac{\partial^4 y}{\partial x^4} = 0. \tag{11}$$

Here, $y(x, t)$ is a scalar field defined on the spatial domain $x \in [0, L)$ with periodic boundary conditions so that $y(x + L, t) = y(x, t)$. We numerically integrate Eq. (11) using a pseudo-spectral scheme described in Kassam and Trefethen (2005). The time-series data is sampled at $\Delta = 0.25$ and used to create a training data set with $T =$ 80,000 time steps and a test data set of length 20,000 time steps. We follow the same

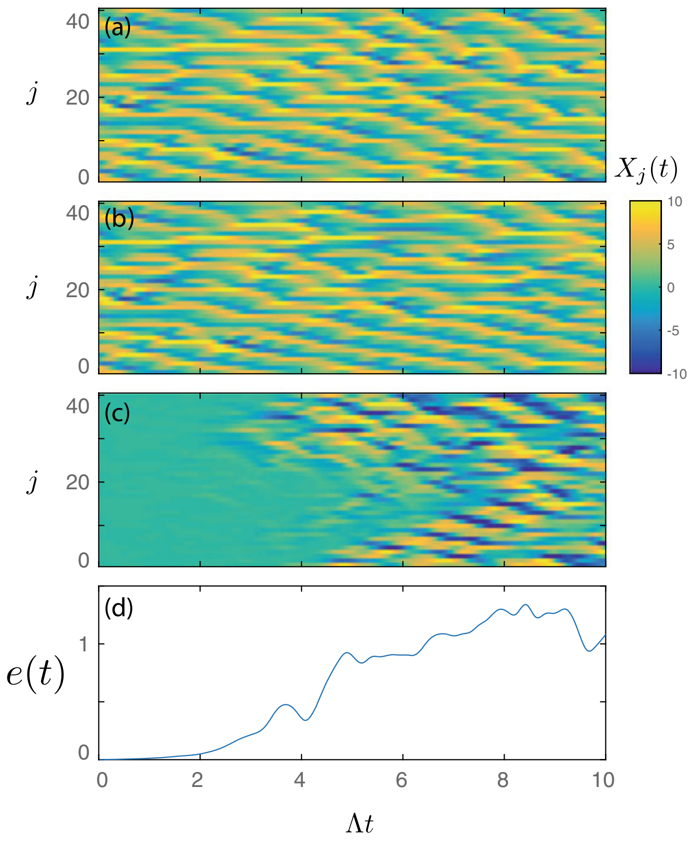

Fig. 6 Lorenz 96 prediction results for parameter value $F = 8$, $N = 40$. Panel **a** shows the true trajectory to be predicted by the reservoir. Panel **b** shows the reservoir prediction with $M = 20$ reservoirs and a locality parameter $l = 2$. Panel **c** shows the difference between the reservoir prediction and the true trajectory. Panel **d** shows the normalized RMS error $e(t)$ in the reservoir prediction as a function of time

validation procedure as outlined in Sect. 4.5 and test the accuracy of our forecasting results. Figure 9 shows the results for a single prediction interval. Figure 10 shows the variability in the prediction accuracy (as measured by the RMS error) on different intervals of the attractor. The effect of changing the number of reservoirs (M) used in the prediction setup is qualitatively similar to those resulting from our tests on the Lorenz 96 model, Eq. (9) shown in Fig. 8.

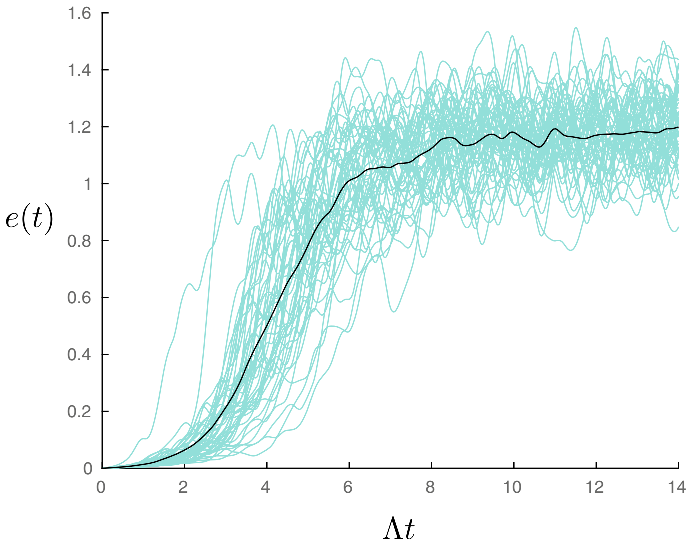

Fig. 7 RMS error in reservoir predictions over multiple intervals starting from different points on the attractor. The parameters of the Lorenz model are $F = 8$ and $N = 40$. The reservoir system is composed of $M = 20$ reservoirs with hyperparameters given in Table 1

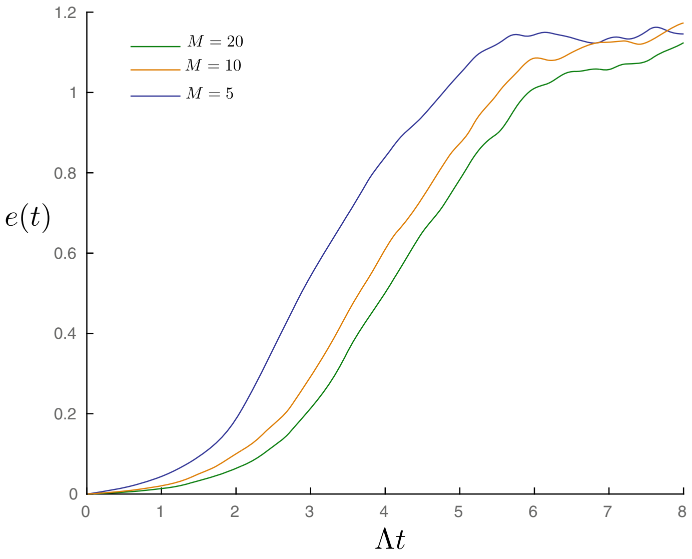

Fig. 8 RMS error $e(n)$ averaged over 50 prediction intervals in the Lorenz 96 prediction for parameter value $F = 8$, $N = 40$. The number of reservoirs used (M) in the parallel prediction is as indicated in the legend

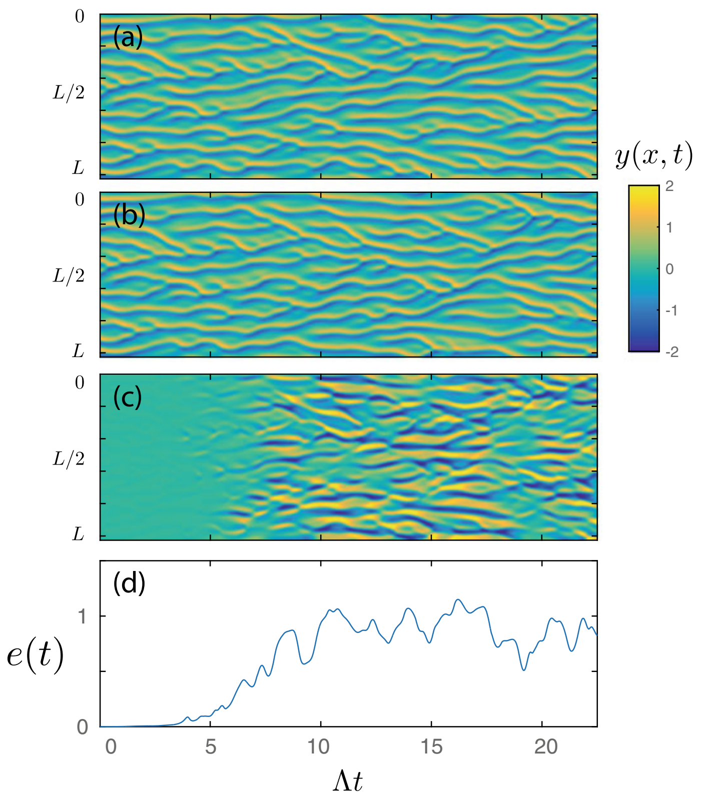

Fig. 9 Kuramoto–Sivashinsky prediction results. Panel **a** shows the true trajectory to be predicted by the reservoir. The periodicity length is $L = 100$. The $K = 256$ time series is predicted using $M = 32$ reservoir and a locality parameter $l = 6$. Panel **b** shows the reservoir prediction. Panel **c** shows the difference between the reservoir prediction and the true trajectory, i.e., **b** minus **a**. Panel **d** shows the normalized RMS error $e(t)$ in the reservoir prediction as a function of time

5 Hybrid Forecasting

We next consider the important and frequently encountered situation where a physical, knowledge-based model of a dynamical system is available but is imperfect in the sense that its dynamics deviates from that of the system that it is meant to model. This kind of model error can significantly degrade the predictions made by such a knowledge-based model. In this section, we show that machine learning can be a very useful tool for mitigating deficiencies of typical knowledge-based prediction systems. The hybrid forecasting configuration used in this section was introduced in Pathak et al. (2018b).

Figure 11 illustrates our scheme for implementing a hybrid machine learning setup that combines an imperfect knowledge-based model of a dynamical system with a

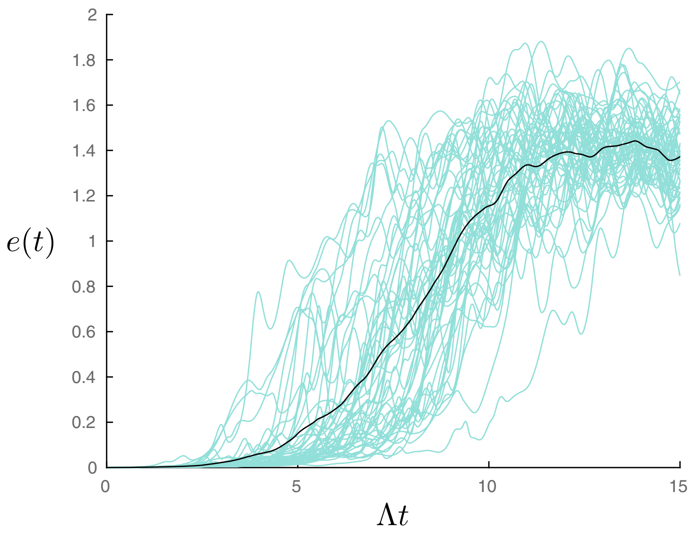

Fig. 10 RMS error in reservoir predictions over multiple intervals starting from different points on the attractor. The periodicity length of the KS system is $L = 100$. The reservoir system is composed of $M = 32$ reservoirs with hyperparameters given in Table 2

Table 2 Kuramoto–Sivashinsky system prediction hyperparameters

Hyperparameter	Value	Hyperparameter	Value
ρ	0.6	D	5000
σ	1	T	80,000
ξ	32	β	10^{-4}
l	6	τ	1000

reservoir-computing-based machine learning setup. In the next section, we describe the training and prediction scheme illustrated in Fig. 11.

5.1 Training

We assume that we have a training data set $\mathbf{u}(n)$, $-T \leq n \leq -1$, of measurements from the dynamical system of interest that have been sampled on a time interval Δ. Further, we assume that we have an imperfect model of the dynamical system denoted by $\mathcal{M}$, so that

$$\tilde{\mathbf{u}}_{\mathcal{M}}(n+1) = \mathcal{M}[\mathbf{u}(n)] \tag{12}$$

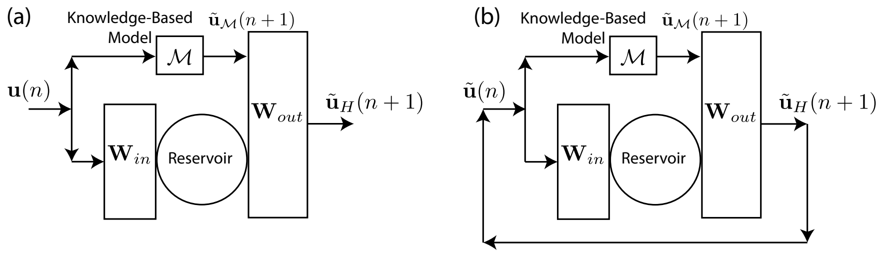

Fig. 11 Hybrid Forecasting Scheme for combining the reservoir prediction with an imperfect knowledge-based model. During the training phase **a**, the system is in an open-loop configuration while in the prediction phase **b**, the system is in a closed-loop configuration

gives us an approximate Δ-step prediction of $\mathbf{u}(n)$. We construct a reservoir with D nodes connected according to the adjacency matrix $\mathbf{A}$ and denote the state of the reservoir by $\mathbf{r}$. In the interval $-T \leq n \leq -1$, we evolve the reservoir according to the equation,

$$\mathbf{r}(n+1) = \tanh(\mathbf{A}\mathbf{r}(n) + \mathbf{W}_{\text{in}}\mathbf{u}(n)), \tag{13}$$

and collect the states $\mathbf{r}(n)$ for $-T \leq n \leq -1$. We also collect the imperfect model forecasts $\tilde{\mathbf{u}}_{\mathcal{M}}(n+1) = \mathcal{M}[\mathbf{u}(n)]$ for $-T \leq n \leq -1$. Let $\mathbf{h}(n)$ be the vector formed by concatenation of the reservoir state and the imperfect model forecast as $\mathbf{h}(n) = [\mathbf{r}(n); \tilde{\mathbf{u}}_M(n+1)]$. The trained output weights of the hybrid reservoir forecasting system are calculated similar to Eq. (6) so that

$$\mathbf{W}_{\text{out}}\mathbf{h}(n+1) = \tilde{\mathbf{u}}_H(n+1). \tag{14}$$

Note that, by the minimization carried out by the training procedure, it is reasonable to think of this output as a semi-optimal combination of the ML component and the imperfect knowledge-based component.

5.2 Prediction

After the training weights have been computed, we can start prediction at any time $n_0 \geq 0$. For example, if $n_0 > \xi$ we do the following steps:

- Step 1: Synchronize the hybrid system to the ground truth for ξ steps by running the open-loop system, Eqs. 12, 13, from $n = n_0 - \xi$ to $n = n_0$.
- Step 2: Predict for the next τ ($\gg \xi$) steps using the closed-loop system illustrated in Fig. 11b and described by the equations,

$$\tilde{\mathbf{u}}_H(n) = \mathbf{W}_{\text{out}}[\mathbf{r}(n); \tilde{\mathbf{u}}_{\mathcal{M}}(n+1)], \tag{15}$$

$$\mathbf{r}(n+1) = \tanh(\mathbf{A}\mathbf{r} + \mathbf{W}_{\text{in}}\tilde{\mathbf{u}}_H(n)). \tag{16}$$

5.3 An Example: Kuramoto–Sivashinsky Equations

We demonstrate the hybrid prediction setup using the Kuramoto–Sivashinsky equation. In this section, we consider training data generated by numerically simulating Eq. (11). Let the imperfect model be given by the following equation:

$$\frac{\partial y}{\partial t} + y\frac{\partial y}{\partial x} + (1+\epsilon)\frac{\partial^2 y}{\partial x^2} + \frac{\partial^4 y}{\partial x^4} = 0 \tag{17}$$

with $x \in [0, L)$ where L is the periodicity length. In Eq. (17), the parameter ϵ describes how closely the equation models the true dynamical system given by Eq. (11). A larger value of ϵ indicates a larger model error, and thus, a less accurate model. We consider a KS system with $L = 35$. Figure 12 illustrates the advantage of the hybrid forecasting scheme over either a pure machine learning approach or the imperfect model by itself. The top panel shows the result of the direct numerical solution of the KS equation (i.e., Eq. 11). The next three panels [(a), (b), (c)] show the error of the reservoir prediction [panel (a)], of the imperfect model [panel (b)] and of the hybrid [panel (c)] for a case with a moderate size reservoir ($D = 5000$ nodes) and a relatively small amount of model error ($\epsilon = 0.01$). We see from panels (a)–(c) that the predictions from the reservoir and from the imperfect knowledge-based model both yield prediction results that are of reasonable value (duration of useful predictions lasting about 1.5 and 2.3 Lyapunov times, respectively), but that the hybrid yields a substantially longer duration of useful prediction (about 6.2 Lyapunov times) than either of its two components. Panels (d), (e), and (f) are for a situation in which the reservoir is substantially smaller ($D = 500$) and the error in the knowledge-based model is substantially greater ($\epsilon = 0.1$). Panels (d) and (e) show that for this case, the predictions of both the reservoir and the knowledge-based model are fairly worthless. Nevertheless, when these two components are combined in a hybrid, they yield substantial prediction power as indicated in panel (f) (i.e., prediction time of about 4.5 Lyapunov times).

6 Parallel/Hybrid Forecasting

For our goal of using machine learning to improve forecasting of large complex spatiotemporally chaotic systems, we believe that the most useful strategy will be to combine the parallel approach of Sect. 4 with the hybrid approach of Sect. 5. This combination (1) will, via the parallelization, effectively exploit the LSTCI properties

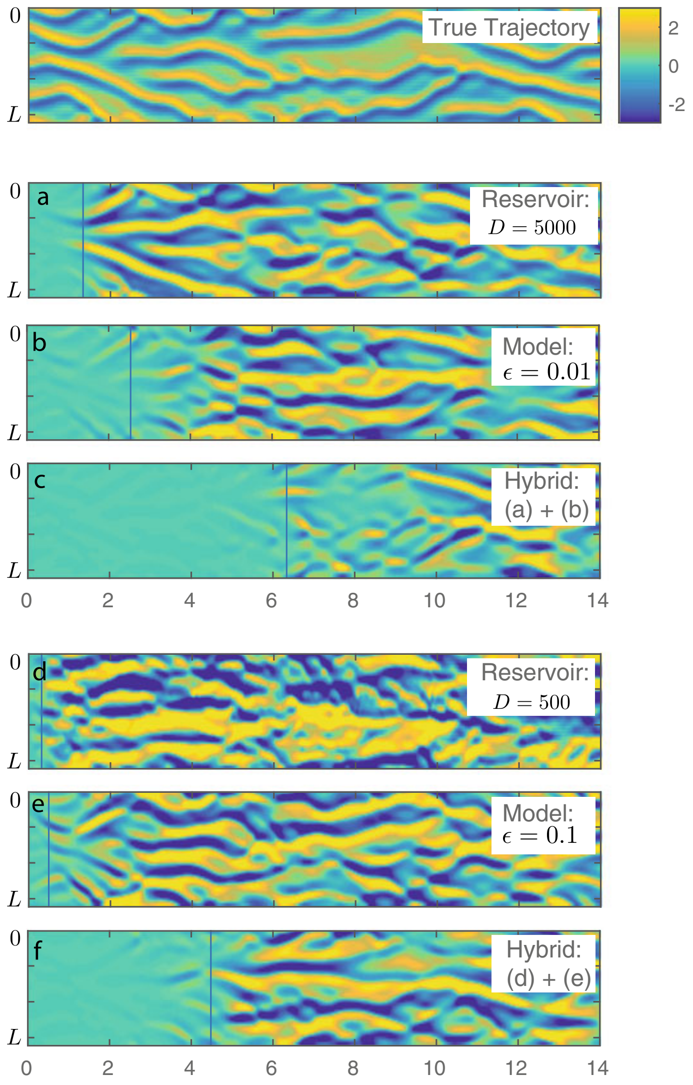

Fig. 12 Top Panel: True trajectory of the KS equation being predicted (Eq. (11) with $L = 35$). Panels **a**–**f** are forecast errors using the indicated schemes. Panels **a** and **d** are prediction errors using a reservoir-only scheme with the indicated reservoir size. Panels **b** and **e** are prediction errors using only the knowledge-based model with the indicated model error (ϵ). Panels **c** and **f** are the prediction errors upon using the hybrid scheme that combines the reservoir and the knowledge-based schemes. Panel **c** combines a reservoir of size $D = 5000$ with the knowledge-based model with error $\epsilon = 0.01$. Panel **f** combines a reservoir of size $D = 500$ with the knowledge-based model with error $\epsilon = 0.1$

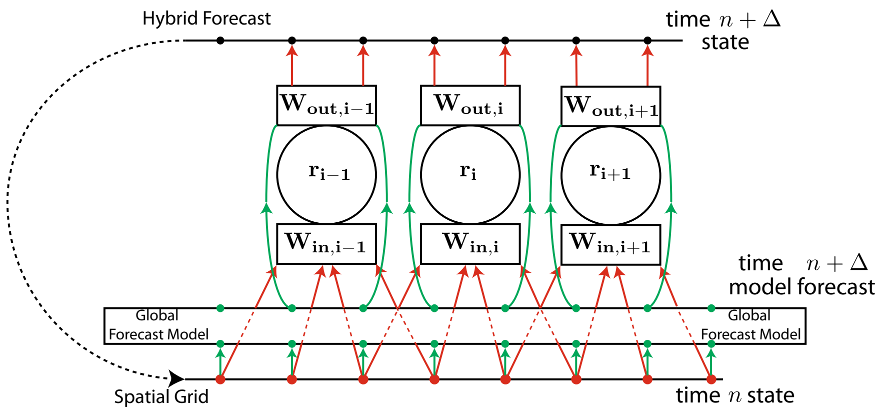
Hybrid Forecast
time $n+\Delta$ state
$\mathbf{W}_{\mathbf{out},\mathbf{i-1}}$
$\mathbf{W}_{\mathbf{out},\mathbf{i}}$
$\mathbf{W}_{\mathbf{out},\mathbf{i+1}}$
$\mathbf{r}_{\mathbf{i-1}}$
$\mathbf{r}_{\mathbf{i}}$
$\mathbf{r}_{\mathbf{i+1}}$
$\mathbf{W}_{\mathbf{in},\mathbf{i-1}}$
$\mathbf{W}_{\mathbf{in},\mathbf{i}}$
$\mathbf{W}_{\mathbf{in},\mathbf{i+1}}$
time $n+\Delta$ model forecast
Global Forecast Model
Global Forecast Model
Spatial Grid
time n state

Fig. 13 A suggested architecture for a parallelized hybrid forecasting scheme. The machine learning component is implemented in a manner similar to Sect. 4, with an additional global forecast model to assist in the forecast

of the dynamical system and allow for computational efficiency in the machine learning component with respect to reservoir size, training, and amount of training data, and (2) will, via our hybrid scheme, provide a very effective platform for simultaneously utilizing data and first-principles system knowledge embodied by an imperfect global model.

Figure 13 shows a possible implementation of such a parallel/hybrid forecasting scheme. The spatial grid is partitioned similar to Sect. 4 (Figs. 2, 5). Additionally, a global knowledge-based forecast model (shown as the horizontally oriented long, thin rectangle) makes predictions which are distributed to the reservoirs and combined with the machine learning forecasts similar to Sect. 5 (Fig. 11). Since the knowledge-based predictions are global, if there is any aspect of the dynamics for which our LSTCI assumption (used in the ML parallelization) is deficient, we expect that the training process will allow potentially reasonable modeling of such an aspect via the knowledge-based model. Note that the grid for the ML component need not be the same as that for the global knowledge-based component, and that the ML grid density can be inhomogeneous. This freedom can be utilized by making the ML grid denser than that of the global knowledge-based component to provide extra resolution, or by restricting a denser ML component to a limited area for regional forecasting. The parallel/hybrid forecasting approach outlined here has been examined in further detail in Wikner et al. (2020).

7 Conclusion

In this chapter, we have considered the situation in which one desires to forecast the evolution of a large complex spatiotemporally chaotic system for which there is some, possibly incomplete and/or inaccurate, descriptive knowledge that can be used to formulate an imperfect computational model. The limitations of such a computational model may stem from deficiencies in our knowledge of basic processes determining the system evolution, or of computational feasibility of modeling such processes (e.g., in situations where there is a very wide range of relevant scales), or a combination of these. While a model of this type may have deficiencies, we wish to utilize whatever capabilities it has to further our end of forecasting.

On the other hand, machine learning purely from past time-series data of an evolving dynamical system has had success in forecasting for certain situations. However, when systems are large and complex and the system state description to be forecasted is correspondingly large and complex, it appears that a purely machine learning approach might often not be feasible.

Thus, it makes sense to try and combine these two very different approaches. The combination of the two approaches may potentially be capable of outperforming either one of them acting alone. Even so, implementation of an ML system combined with a knowledge-based computational model still faces a substantial challenge due to the size and complexity of the states that we desire to forecast. Thus, in this chapter we have addressed what we believe are two key issues in this approach to forecasting large complex spatiotemporally chaotic systems: (i) how to wed (hybridize) machine learning with an imperfect knowledge-based computational model, and (ii) how to parallelize the machine learning component in combination with global knowledge-based code, in a manner feasible for the machine learning component. Our review of these two key issues leads us to conclude that preliminary results provide a possible path that may be effective in enabling improved forecasting of large complex spatiotemporally chaotic systems.

However, we emphasize that many issues remain. The task of weather forecasting provides a basis for assessing difficulties and directions for future work aimed at ultimately using this hybrid/parallel approach for forecasting large, complex, spatiotemporal systems. Primary among these is the issue of cyclic prediction and data assimilation. Typically, a new set of weather forecasts is made every 6 h. At the beginning of each 6 h cycle, new measurements of the atmospheric state are used to correct an estimate of the probability distribution of the atmospheric state provided by the 6 h forecast from the previous cycle, and the new probability distribution estimate is used as an initial condition for an atmospheric model, which is then integrated forward to make a new set of probabilistic forecasts. The process by which the previous forecast is combined with the new data is called 'data assimilation'. Moreover, the nature of the data for weather forecasting is itself complex, resulting from measurements with stochastic errors from an array of different types of diagnostic sources (e.g., balloons, satellites, ground stations, ships, aircraft, and radar), and these data sources can vary greatly in space and time (e.g., balloon measurements are typically dense over technologically developed regions, typically less dense over technologically less developed or more sparsely populated regions, and typically very much less dense over oceans).

Thus, to proceed past the preliminary promising results of this chapter, many issues such as incorporation of cyclic data assimilation and data source heterogeneity await. We hope, however, that this chapter will provide a basis for moving forward in this area.

Acknowledgements We thank our colleagues, particularly Brian Hunt, Michelle Girvan, and Istvan Szunyogh, for their contributions. We also acknowledge support from DARPA contract HR00111890044.

References

H. Abarbanel, *Analysis of Observed Chaotic Data* (Springer Science & Business Media, 2012)
V. Afraimovich, N. Verichev, M.I. Rabinovich, Radiophys. Quantum Electron. **29**, 795 (1986)
S.L. Brunton, J.L. Proctor, J.N. Kutz, Proc. Natl. Acad. Sci. **201517384** (2016)
S. Hochreiter, J. Schmidhuber, Neural Comput. **9**, 1735 (1997)
H. Jaeger, GMD Technical report **148**, 13. German National Research Center for Information Technology, Bonn, Germany (2001)
H. Jaeger, H. Haas, Science **304**, 78 (2004)
H. Kantz, T. Schreiber, *Nonlinear Time Series Analysis*, vol. 7 (Cambridge University Press, Cambridge, 2004)
A. Karimi, M.R. Paul, Chaos: Interdiscipl. J. Nonlinear Sci. **20**, 043105 (2010)
A.-K. Kassam, L.N. Trefethen, SIAM J. Sci. Comput. **26**, 1214 (2005)
L. Kocarev, U. Parlitz, Phys. Rev. Lett. **76**, 1816 (1996)
E.N. Lorenz, in *Proceedings of Seminar on Predictability*, vol. 1 (1996)
Z. Lu, B.R. Hunt, E. Ott, Chaos: Interdiscip. J. Nonlinear Sci. **28**, 061104 (2018)
B. Lusch, J.N. Kutz, S.L. Brunton, Nat. Ccommun. **9**, 4950 (2018)
E. Ott, *Chaos in Dynamical Systems* (Cambridge University Press, Cambridge, 2002)
E. Ott, T. Sauer, J.A. Yorke, *Wiley Series in Nonlinear Science* (Wiley, New York, 1994)
J. Pathak, B. Hunt, M. Girvan, Z. Lu, E. Ott, Phys. Rev. Lett. **120**, 024102 (2018a)
J. Pathak, A. Wikner, R. Fussell, S. Chandra, B.R. Hunt, M. Girvan, E. Ott, Chaos: Interdiscip. J. Nonlinear Sci. **28**, 041101 (2018b)
L. Pecora T. Carroll, J. Heagy, *Handbook of Chaos Control*, vol. 227 (1999)
M. Raissi, P. Perdikaris, G. Karniadakis, J. Comput. Phys. **378**, 686 (2019)
N.F. Rulkov, M.M. Sushchik, L.S. Tsimring, H.D. Abarbanel, Phys. Rev. E **51**, 980 (1995)
T. Sauer, J.A. Yorke, M. Casdagli, J. Stat. Phys. **65**, 579 (1991)
P.R. Vlachas, W. Byeon, Z.Y. Wan, T.P. Sapsis, P. Koumoutsakos, Proc. R. Soc. A **474**, 20170844 (2018)
Z.Y. Wan, P. Vlachas, P. Koumoutsakos, T. Sapsis, PloS One **13**, e0197704 (2018)
A. Wikner J. Pathak, B. Hunt, M. Girvan, T. Arcomano, I. Szunyogh, A. Pomerance, E. Ott, Combining machine learning with knowledge-based modeling for scalable forecasting and subgrid-scale closure of large, complex, spatiotemporal systems (2020), arXiv:2002.05514 [cs.LG]
R.S. Zimmermann, U. Parlitz, Chaos: Interdiscip. J. Nonlinear Sci. **28**, 043118 (2018)