ADVANCING EARTH AND SPACE SCIENCE

JAMES Journal of Advances in Modeling Earth Systems®

RESEARCH ARTICLE
10.1029/2021MS002843

Special Section:
Machine learning application to Earth system modeling

Integrating Recurrent Neural Networks With Data Assimilation for Scalable Data-Driven State Estimation

S. G. Penny[1,2], T. A. Smith[1,2], T.-C. Chen[1,2], J. A. Platt[3], H.-Y. Lin[1,2], M. Goodliff[1,2], and H. D. I. Abarbanel[3,4]

[1]Cooperative Institute for Research in Environmental Sciences (CIRES) at the University of Colorado Boulder, Boulder, CO, USA, [2]Physical Sciences Laboratory (PSL), National Oceanic and Atmospheric Administration (NOAA), Boulder, CO, USA, [3]Department of Physics, University of California San Diego (UCSD), La Jolla, CA, USA, [4]Scripps Institution of Oceanography (SIO), La Jolla, CA, USA

Key Points:

- Recurrent neural networks (RNNs) can replace conventional forecast models, producing accurate ensemble forecast statistics and linearized dynamics
- Data assimilation (DA) is compatible with RNNs by applying state estimation in the hidden state space using a modified observation operator
- The integrated RNN-DA methods can be scaled to higher dimensions by applying domain localization techniques

Correspondence to:
S. G. Penny,
Steve.Penny@noaa.gov

Citation:
Penny, S. G., Smith, T. A., Chen, T.-C., Platt, J. A., Lin, H.-Y., Goodliff, M., & Abarbanel, H. D. I. (2022). Integrating recurrent neural networks with data assimilation for scalable data-driven state estimation. *Journal of Advances in Modeling Earth Systems*, *14*, e2021MS002843. https://doi.org/10.1029/2021MS002843

Received 27 SEP 2021
Accepted 14 FEB 2022

Author Contributions:
Conceptualization: S. G. Penny
Formal analysis: H.-Y. Lin
Funding acquisition: S. G. Penny
Investigation: S. G. Penny, T. A. Smith, J. A. Platt
Methodology: S. G. Penny, J. A. Platt
Project Administration: S. G. Penny
Software: S. G. Penny, T. A. Smith, T.-C. Chen, J. A. Platt, H.-Y. Lin, M. Goodliff

Abstract

Data assimilation (DA) is integrated with machine learning in order to perform entirely data-driven online state estimation. To achieve this, recurrent neural networks (RNNs) are implemented as pretrained surrogate models to replace key components of the DA cycle in numerical weather prediction (NWP), including the conventional numerical forecast model, the forecast error covariance matrix, and the tangent linear and adjoint models. It is shown how these RNNs can be initialized using DA methods to directly update the hidden/reservoir state with observations of the target system. The results indicate that these techniques can be applied to estimate the state of a system for the repeated initialization of short-term forecasts, even in the absence of a traditional numerical forecast model. Further, it is demonstrated how these integrated RNN-DA methods can scale to higher dimensions by applying domain localization and parallelization, providing a path for practical applications in NWP.

Plain Language Summary

Weather forecast models derived from fundamental equations of physics continue to increase in detail and complexity. While this evolution leads to consistently improving daily weather forecasts, it also leads to associated increases in computational costs. In order to make a forecast at any given moment, these models must be initialized with our best guess of the current state of the atmosphere, which typically includes information from a limited set of observations as well as forecasts from the recent past. Modern methods for initializing these computer forecasts typically require running many copies of the model, either simultaneously or in sequence, to compare with observations over the recent past and ensure that our best guess estimate of the current state of the atmosphere agrees closely with those observations before making a new forecast. This repeated execution of the computer forecast model is often a time-consuming and costly bottleneck in the initialization process. Here, it is shown that techniques from the fields of artificial intelligence and machine learning (AI/ML) can be used to produce simple surrogate models that provide sufficiently accurate approximations to replace the original costly model in the initialization phase. The resulting process is self-contained, and does not require any further utilization of the original computer model when making daily forecasts.

1. Introduction

Numerical weather prediction (NWP) requires two key components: a computational forecast model and an initialization method, both of which have been demonstrated to contribute approximately equally to the steady improvement in forecast skill over the past 40 years (Buizza et al., 2005; Kalnay, 2002). We seek to replace the computational forecast model with a data-driven surrogate model and integrate these two key components. Weather forecast models typically push the boundaries of computational feasibility, even on the largest super-computers in the world, with a drive toward increased grid resolutions and better resolved physical processes. The most sophisticated initialization methods require executing the forecast model many times, using iterative loops and ensembles of forecasts initialized from perturbed initial conditions. This creates a competing paradigm where computational resources must be balanced between model fidelity and initialization accuracy. As a result, the models serve as a major limiting factor in the development of new initialization methods.

The application of artificial intelligence and machine learning (AI/ML) methods in weather and climate is a rapidly growing activity. Boukabara et al. (2021) described multiple instances in which AI/ML is being developed

Supervision: S. G. Penny
Validation: T. A. Smith
Visualization: S. G. Penny, T. A. Smith, J. A. Platt
Writing – original draft: S. G. Penny, T. A. Smith
Writing – review & editing: S. G. Penny, T. A. Smith, H. D. I. Abarbanel

for target applications in operational NWP. An area of interest noted by Boukabara et al. (2021) is the synergy between AI/ML and data assimilation (DA). Abarbanel et al. (2018) described the deep connections between ML and DA, and in special cases mathematical equivalences (further details will be provided in an upcoming work (Abarbanel, 2022)). Bocquet et al. (2020) established an embedding of ML into DA using a Bayesian perspective and demonstrated numerically that the alternation of ML and DA as reported in Brajard et al. (2020) acts as a coordinate descent minimization for the defined loss function. Other recent approaches that have been applied to combine ML with DA include the application of a neural network design combined with a DA operation to train the network on noisy data (Bocquet et al., 2021; Brajard et al., 2020), the application of artificial neural networks for correcting errors in numerical model forecasts in the DA cycle (Bonavita & Laloyaux, 2020; Farchi et al., 2021), the use of a convolution neural network (CNN) to enforce conservation of mass in a DA procedure (Ruckstuhl et al., 2021), the development of a NN-based tangent linear and adjoint model for use in variational DA methods (Hatfield et al., 2021), an end-to-end application of a combined DA and model bias correction (Arcucci et al., 2021), and the application of DA in the reduced dimension latent space of an autoencoder (Peyron et al., 2021).

Here, we will focus on a simplified form of recurrent neural network (RNN), based on the reservoir computing (RC) paradigm (Jaeger, 2001) that can be used to replace the numerical model in the DA process. RC has been used to emulate dynamical systems ranging from simple chaotic models to global atmospheric models (Arcomano et al., 2020; Lu et al., 2017, 2018; Pathak et al., 2017, 2018). However, it has not been shown whether such models produce an accurate response to perturbations in initial conditions. Integrating a data-driven model with DA techniques requires accurate characterization of dynamical error growth. We will demonstrate that the RNN architecture can produce sufficiently accurate representations of such error growth to the degree that the RNN-based models can replace key components of foundational DA algorithms such as the ensemble Kalman filter (EnKF; Evensen, 1994) and the 4D variational method (4D-Var; Courtier et al., 1994; Courtier & Talagrand, 1987; Talagrand & Courtier, 1987). Such key components include forecast error covariance statistics derived from ensemble forecasts, and the tangent linear model and its adjoint.

We introduce a method to achieve this in a direct manner by applying DA to update the "hidden" or "reservoir" space of the RNN/RC dynamics (Figure 1). Results are shown assimilating both fully observed and sparsely observed dynamics, with a range of observation noise levels, using an RNN-based ensemble transform Kalman filter (ETKF; Bishop et al., 2001; Hunt et al., 2007) and an RNN-based strong constraint incremental 4D-Var (Courtier et al., 1994).

2. Methods

The weather and climate communities regularly produce three primary categories of data: (a) observations of the Earth system, either drawn directly from in situ measurements or from satellite-based sensors, (b) simulations produced on regular gridded fields, generated from numerical models derived from basic principles of physics, and (c) retrospective analyses (reanalyzes), that attempt to optimally combine the previous two categories to produce historical reconstructions of the atmosphere or other components of the Earth system. All of these data sets have inherent weaknesses. Observations are generally sparse and irregular, can be indirect (e.g., integrated quantities like satellite radiance measurements), and can contain unpredictable errors. Numerical models have systematic errors that result from abstractions, approximations, and unresolved processes. Reanalysis products attempt to mitigate these weaknesses, but still inherit them to some degree. However, as these data sets have already been produced and archived, they are valuable resources that can be leveraged to develop data-driven methods.

We assume that for a realistic application either a long numerical model simulation or retrospective historical analysis is available as training data. Thus, we train the RNN to generate accurate predictions using a dedicated training data set that resolves all components of the target dynamics. For this study, we use the output from the model described in Section 2.4 to train our RNN models. We focus our attention on developing the capabilities to integrate the pretrained data-driven RNN model with an online DA process that repeatedly ingests new noisy and sparse observations, updates the state estimate of the system, and makes new short-term forecasts. This has applications ranging from operational NWP to the efficient reconstruction of historical Earth system states.

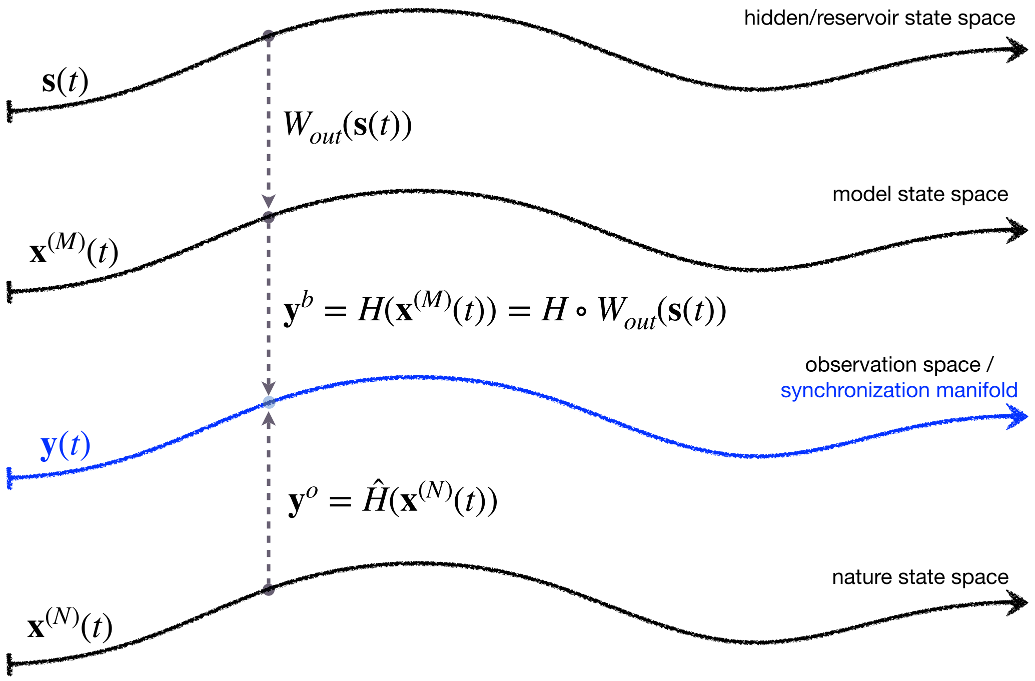

Figure 1. Data assimilation is applied to update the hidden/reservoir state space $\mathbf{s}(t)$. Observations $\mathbf{y}^{\circ}$ are sampled from the nature state space $\mathbf{x}^{(N)}(t)$, while the composition $H \circ W_{out}$ is used as an observation operator to map the hidden/reservoir state space to an equivalent representation that can be used to form the innovations $\mathbf{d}(t) = \mathbf{y}^{\circ}(t) - H \circ W_{out}(\mathbf{s}(t))$ in the observation space.

2.1. Network Design

Reservoir computing is a category of machine learning methods. It originated in the works of Jaeger (2001), who introduced Echo State Networks (ESN), and Maass et al. (2002) who introduced Liquid State Machines (LSM). Both methods assume that an input signal can be mapped to a fixed nonlinear system of higher dimension than the input. Such systems can be trained using a readout layer to map the state of the reservoir to the desired output. The result is a simple model that can reproduce the potentially complex dynamics of the original system. A survey is provided by Lukoševičius and Jaeger (2009), while Konkoli (2017) provides further discussion on the generality of RC. Successful applications of RC have been demonstrated for the prediction of spatiotemporally chaotic dynamics (Chattopadhyay et al., 2020; Pathak et al., 2018; Platt et al., 2021), and in particular of geophysical fluid dynamics (Arcomano et al., 2020; Lin & Penny, 2021).

RNNs have long been used as a preferred ML method for cases in which temporal considerations are necessary. We use a basic RNN (Elman, 1990) with a simplified structure to produce a variant of the RC network. The RNN/RC design is useful for prediction because the "hidden" or "reservoir" state $\mathbf{s}(t_i)$ provides a short-term memory of the target system trajectory $\mathbf{x}(t)$ up to time t_i. Assuming an accurate forecast can be made of the hidden/reservoir state $\mathbf{s}(t_{i+1})$ by a well-trained RNN, then that forecast can be mapped to the target system space to produce a forecast $\mathbf{x}'(t_{i+1})$ of the true system state $\mathbf{x}(t_{i+1})$.

As highlighted by Schrauwen et al. (2007), Steil (2004) showed that the state-of-the-art learning rule for RNNs at the time had the same weight dynamics as the methods proposed by Jaeger (2001) and Maass et al. (2002). The Atiya and Parlos (2000) recurrent learning rule trains the output weights while the internal weights are only globally scaled up or down. Similarly, we classify the RNN model parameters as either "micro-scale" or "macro-scale". By "micro-scale" parameters, we refer to the individual matrix or vector elements. These are typically trained in an artificial neural network using backpropagation. By "macro-scale" parameters, we refer to weighting factors that scale all of the matrix or vector elements at once. We note that what we call "macro-scale" parameters were called "global parameters" by Lukoševičius (2012). The general form of our RNN is given as,

$$\mathbf{s}(t_{i+1}) = F(\mathbf{s}(t_i), \mathbf{x}(t_i)) = l * f(\rho \mathbf{W}_{res}\mathbf{s}(t_i) + \sigma \mathbf{W}_{in}\mathbf{x}(t_i)) + (1 - l) * \mathbf{s}(t_i), \tag{1}$$

$$\mathbf{x}'(t_{i+1}) = G(\mathbf{s}(t_{i+1})) = \mathbf{W}_{out}(\mathbf{s}(t_{i+1})), \tag{2}$$

where $\mathbf{x}(t_i)$ is the system state at time t_i, provided from data, $\mathbf{s}(t_i)$ is the hidden/reservoir state, and $\mathbf{x}'(t_{i+1})$ is the predicted system state at the next time t_{i+1}. The parameter ρ determines the spectral radius of the reservoir adjacency matrix $\mathbf{W}_{res}$, σ scales the input signal mapped to the reservoir space by $\mathbf{W}_{in}$. The parameter l is the "leak' parameter that gates new information into the system, which can also be interpreted as a relaxation parameter. We will use $f(\cdot) \equiv \tanh(\cdot)$.

After training the RNN model parameters, we expect the predicted state $\mathbf{x}'(t_{i+1})$ to be close to the true system state $\mathbf{x}(t_{i+1})$. In a typical RNN, all parameters of the system described by Equations 1 and 2 are trained, usually by a gradient descent type optimization method. For RC, however, all model parameters in Equation 1 are assumed fixed. Equation 1 is then iterated to generate a time series $\mathbf{s}(t)$ of hidden/reservoir states that corresponds to the training data $\mathbf{x}(t)$. We will assume the "readout" operator $\boldsymbol{W}_{out}(\cdot)$ in Equation 2 is linear, in which case we use the notation $\mathbf{W}_{out}$.

The microscale parameters in $\mathbf{W}_{out}$ are trained (while the other parameters of our RNN are held fixed) by solving a regularized least squares equation using a loss function that targets forecasts that are one time step in the future (Jaeger, 2001). The corresponding loss function is,

$$\mathcal{L}_{micro}(\mathbf{W}_{out}) = \|\mathbf{W}_{out}\mathbf{S}_{data} - \mathbf{X}_{data}\|^2 + \beta\|\mathbf{W}_{out}\|^2, \tag{3}$$

where $\mathbf{S}_{data}$ and $\mathbf{X}_{data}$ are matrices that comprise the vector-valued states $\mathbf{s}(t)$ and $\mathbf{x}(t)$ for the entire training data set, ordered columnwise, and β is a Tikhonov regularization parameter.

To move closer toward the approach of the general RNN, in which all parameters are trained, we regard the scalars l, ρ, σ, and β as macroscale parameters subject to training as well. The matrix $\mathbf{W}_{res}$ is initialized with a spectral radius of 1 prior to training, but otherwise the values are assigned randomly using a uniform distribution centered at 0. For computational efficiency, $\mathbf{W}_{res}$ is assumed to be sparse (i.e., only 1% of entries are nonzero). The matrix $\mathbf{W}_{in}$ is initialized using a uniform random distribution with values ranging from -1 to 1.

An extended forecast $\mathbf{x}^f(t)$ is made with the RNN from time t_0 to t_i, for $i > 0$, by recursively replacing the input state with the RNN prediction initialized from the previous time. Defined inductively, commencing with $\mathbf{x}(t_0)$,

$$\mathbf{x}^f(t_1) = \mathbf{W}_{out} \circ F(\mathbf{s}(t_0), \mathbf{x}(t_0)), \tag{4}$$

$$\mathbf{x}^f(t_i) = \mathbf{W}_{out} \circ F\left(\mathbf{s}(t_{i-1}), \mathbf{x}^f(t_{i-1})\right), \tag{5}$$

where $\circ$ is the function composition operator. We note that due to the nature of chaotic dynamical systems (with leading Lyapunov exponent greater than 0), any error in one step of this recursion will accumulate and result in exponential error growth over time.

The macroscale parameters are trained using a nonlinear Bayesian optimization method, which uses a surrogate model representation of the loss function (Ginsbourger et al., 2010; Jones et al., 1998). For this macroscale optimization we use a loss function that targets longer-range prediction,

$$\mathcal{L}_{macro}\left(\mathbf{x}^f(t)\right) = \sum_{i=1}^{M}\sum_{t=t_i}^{t_{i+N}} \|\mathbf{x}^f(t) - \mathbf{x}(t)\|^2 \exp\left(-\frac{t - t_i}{t_{i+N} - t_i}\right), \tag{6}$$

where M represents the number of separate initial times $[t_1, t_2, t_3, \ldots, t_M]$ used to make independent forecasts, selected randomly without replacement from the training data set, and N represents the number of time steps used for each forecast. We apply an exponential scaling term in order to account for the exponential growth of errors that is typical of chaotic dynamics. This term gives the forecast errors in the earlier portion of the forecast more weight, as this period is more relevant for cycled DA applications. Griffith et al. (2019) similarly applied long forecasts at multiple initial times in a reservoir computing application to identify model parameters that resulted in the reconstruction of the full system attractor.

We note that in the context of reservoir computing these macroscale parameters are often named hyperparameters, however, because they are trained here via optimization we do not consider them as such. Recall that by definition a hyperparameter is any design decision that is set before the learning process begins, is generally tunable, and can directly affect how well a model trains. The qualified hyperparameters of this optimization are provided in Table A1 of Appendix A.

By computing the Jacobian of the forecast model defined by Equations 1, 2, 4, and 5 with respect to the hidden/reservoir state, we can determine the linear propagator **M** of the reservoir dynamics from time t_i to t_{i+1} as,

$$\mathbf{W} = \rho \mathbf{W}_{res} + \sigma \mathbf{W}_{in} \mathbf{W}_{out}, \tag{7}$$

$$\mathbf{M}_{[t_{i+1}, t_i]}(\mathbf{s}(t_i)) = \frac{dF}{d\mathbf{s}} = l * \text{diag}\left(1 - \tanh(\mathbf{W}\mathbf{s}(t_i))^2\right)\mathbf{W} + (1 - l) * \mathbf{I}. \tag{8}$$

The linear propagator describes the evolution of small perturbations from a reference trajectory. It can be used in DA methods, in particular 4D-Var, where it is called the tangent linear model (TLM). Further, the Lyapunov exponents of the system are determined by integrating the linear propagator from time $t = 0 \rightarrow \infty$ and computing the eigenvalues of the resulting system. Practical algorithms based on QR decomposition (Golub & Van Loan, 2012) are provided by Geist et al. (1990). For computational efficiency, we implement the TLM and its adjoint as linear operators to avoid matrix multiplications and allow for efficient matrix-vector operations applied within the iterative minimization schemes.

2.2. Data Assimilation

Trevisan et al. (2010), Trevisan and Palatella (2011), Palatella et al. (2013), and Bocquet et al. (2017) showed that the number of observations needed to constrain any DA system is related to the number of non-negative Lyapunov exponents in the system. Platt et al. (2021) indicated that reproducing the Lyapunov spectrum is critical to generating accurate predictions with reservoir computing models-with deviations from the true spectrum leading to significantly degraded forecast skill. Considering these points, we presume that even if the hidden/reservoir state space is large, if the RNN is trained to be sufficiently accurate such that the true Lyapunov spectrum is well approximated, then the number of observations required to constrain the hidden state space dynamics should be the same as is required to constrain the original system dynamics.

Following this presumption, we apply DA in the hidden/reservoir space of the RNN system, and apply the composition of an observation operator with the readout operator in order to compare hidden/reservoir states with observations of the original system. To test this approach, we apply two well-known DA algorithms integrated with the pretrained RNN forecast model-the ensemble transform Kalman filter (Bishop et al., 2001; Hunt et al., 2007) and the strong constraint incremental 4-dimensional variational method (4D-Var; Courtier et al., 1994).

From the perspective of operational forecasting, the RNN provides a simple and low-cost replacement for the production of essential information needed for the online DA cycle, such as forecast error covariance statistics and the tangent linear and adjoint model dynamics. From the machine learning perspective, the DA algorithms allow the RNN hidden/reservoir dynamics to be driven with a noisy and sparsely observed signal. We will show that in cases where the direct insertion of observations quickly corrupts the hidden/reservoir state and leads to inaccurate forecasts, the DA methods can produce valid reconstructions of the system state as well as viable initial conditions for short-term forecasts.

Kalnay et al. (2006) described the ideal initial ensemble perturbations as those that effectively span the space defined by the analysis error covariance. We use ensemble forecast statistics produced by the RNN model to generate dynamically estimated forecast error covariance statistics, and then apply an ETKF to assimilate noisy observations of the true system state and estimate the analysis error covariance. Bocquet and Carrassi (2017) showed that the minimum ensemble size required to constrain a (non-localized) deterministic ensemble filter such as the ETKF is equal to the number of non-negative Lyapunov exponents of the system dynamics. Thus, we expect that with a well-trained RNN that closely approximates the correct Lyapunov spectrum, the minimum number of ensemble members needed to constrain the ETKF will be the same as the number of members needed to constrain the original system.

We define a new modified observation operator by composing the conventional observation operator $H(\cdot)$, which maps from the system space to the observation space, with the readout operator $W_{out}(\cdot)$, which maps from the hidden/reservoir space to the system space (see Figure 1). Our implementation of the ETKF follows the formulation of Hunt et al. (2007). Let $\bar{\mathbf{y}}^b = H\left(W_{out}\left(\bar{\mathbf{s}}^b\right)\right)$, where $\bar{\mathbf{s}}^b$ is the background ensemble mean hidden/reservoir state, and $\mathbf{Y}^b = H(W_{out}(\mathbf{S}^b))$, where the columns of $\mathbf{S}^b$ are ensemble perturbations around the mean, then

$$\tilde{\boldsymbol{P}}^a = \left[\frac{k-1}{\gamma}\mathbf{I} + \left(\mathbf{Y}^b\right)^T \mathbf{R}^{-1}\mathbf{Y}^b\right]^{-1}, \tag{9}$$

$$\mathbf{W}^a = \left[(k-1)\tilde{\boldsymbol{P}}^a\right]^{\frac{1}{2}}, \tag{10}$$

$$\mathbf{S}^a = \mathbf{S}^b\mathbf{W}^a, \tag{11}$$

$$\bar{\boldsymbol{w}}^a = \tilde{\boldsymbol{P}}^a\left(\mathbf{Y}^b\right)^T\mathbf{R}^{-1}\left(\mathbf{y}^o - \bar{\mathbf{y}}^b\right), \tag{12}$$

$$\bar{\mathbf{s}}^a = \mathbf{S}^b\bar{\boldsymbol{w}}^a + \bar{\mathbf{s}}^b, \tag{13}$$

where $\mathbf{R}$ is the observation error covariance matrix, k is the ensemble dimension, γ is a multiplicative inflation factor, $\tilde{\boldsymbol{P}}^a$ is the analysis error covariance matrix represented in the ensemble perturbation subspace, $\mathbf{W}^a$ is applied as a transform operator to map the background ensemble perturbations to analyze ensemble perturbations, and $\bar{\boldsymbol{w}}^a$ determines the weighting coefficients of the column vectors $\mathbf{S}^b$, which are used as a linear basis to form the new ensemble mean analysis state vector $\bar{\mathbf{s}}^a$. For reference, substituting Equation 12 into 13, and further substituting the equations provided above, the resulting Kalman gain matrix for the integrated RNN-ETKF is,

$$\mathbf{K} = \mathbf{S}^b\left[\frac{k-1}{\gamma}\mathbf{I} + \left[H\left(W_{out}\left(\mathbf{S}^b\right)\right)\right]^T\mathbf{R}^{-1}\left[H\left(W_{out}\left(\mathbf{S}^b\right)\right)\right]\right]^{-1}\left[H\left(W_{out}\left(\mathbf{S}^b\right)\right)\right]^T\mathbf{R}^{-1}. \tag{14}$$

The control vector for 4D-Var can similarly be formed in the hidden/reservoir space. We use the strong constraint incremental 4D-Var, implemented using an outer and inner loop. In the outer loop, a nonlinear forecast $\mathbf{s}_t^f = \mathcal{M}_{[t,0]}(\mathbf{s}_0)$ is generated over a short optimization period, called the analysis window. In the inner loop, the linearized dynamics are used to find an improved guess for the initial state $\mathbf{s}_0$ using an iterative linear solver, and then the outer loop is repeated. If we let $\delta\mathbf{s}_0 = \left(\mathbf{s}_0 - \mathbf{s}_0^f\right)$, $\delta\mathbf{s}_0^b = \left(\mathbf{s}_0^b - \mathbf{s}_0^f\right)$, where $\mathbf{s}_0^b$ is the initial background state estimate, and $\mathbf{d}_t = \left(\mathbf{y}_t^o - H_t \circ W_{out}\left(\mathbf{s}_t^f\right)\right)$, then the objective function is,

$$J(\delta\mathbf{s}_0) = J_b(\delta\mathbf{s}_0) + J_o(\delta\mathbf{s}_0), \tag{15}$$

where,

$$J_b(\delta\mathbf{s}_0) = \frac{1}{2}\left(\delta\mathbf{s}_0 - \delta\mathbf{s}_0^b\right)^T\mathbf{B}^{-1}\left(\delta\mathbf{s}_0 - \delta\mathbf{s}_0^b\right), \tag{16}$$

$$J_o(\delta\mathbf{s}_0) = \frac{1}{2}\sum_{t=0}^{N_t}\left(\mathbf{d}_t - H_t\circ W_{out}\left(\mathbf{M}_{[t,0]}\delta\mathbf{s}_0\right)\right)^T\mathbf{R}_t^{-1}\left(\mathbf{d}_t - H_t\circ W_{out}\left(\mathbf{M}_{[t,0]}\delta\mathbf{s}_0\right)\right), \tag{17}$$

$$\mathbf{M}_{[0,0]} = \mathbf{I}, \tag{18}$$

$$\mathbf{M}_{[t+1,t]} = l * \operatorname{diag}\left(1 - \tanh(\mathbf{W}\mathbf{s}(t))^2\right)\mathbf{W} + (1-l) * \mathbf{I}, \tag{19}$$

as in Equation 8, and the initial condition in the original system space can be recovered by,

$$\mathbf{x}_0 = W_{out}(\mathbf{s}_0). \tag{20}$$

In our implementation, $H_t(\cdot)$ and $W_{out}(\cdot)$ are linear, so we now replace them with the matrix notations $\mathbf{H}_t$ and $\mathbf{W}_{out}$. When such operators are not linear, a linear approximation via Taylor series expansion is typically applied. The minimum is found when the gradient with respect to the control vector $\delta\mathbf{s}_0$ equals 0,

$$\nabla_{\delta\mathbf{s}_0}J = \mathbf{B}^{-1}\left(\delta\mathbf{s}_0 - \delta\mathbf{s}_0^b\right)\sum_{t=0}^{N_t}\mathbf{M}_{[t,0]}^T\mathbf{H}_t^T\mathbf{R}_t^{-1}\left(\mathbf{d}_t - \mathbf{H}_t\mathbf{M}_{[t,0]}\mathbf{W}_{out}\delta\mathbf{s}_0\right) = 0. \tag{21}$$

We solve this by separating terms into the form '$\mathbf{Ax} = \mathbf{b}$', where $\delta\mathbf{s}_0$ is the only unknown quantity,

$$\left(\mathbf{I} + \mathbf{B}\sum_{t=0}^{N_t} \mathbf{H}_t \mathbf{M}_{[t,0]}\right)\delta\mathbf{s}_0 = \mathbf{B}\sum_{t=0}^{N_t} \mathbf{M}_{[t,0]}^T \mathbf{H}_t^T \mathbf{R}_t^{-1}\mathbf{d}_i + \delta\mathbf{s}_0^b \tag{22}$$

and then applying the biconjugate gradient stabilized method (Van der Vorst, 1992). Alternative forms of Equation 22 are available, for example, making the '**A**' matrix symmetric so that the conjugate gradient method can be applied. Returning to the outer loop, a new nonlinear forecast $\mathbf{s}^f(t) = \mathcal{M}_{[t,0]}\left(\mathbf{s}_0^f + \delta\mathbf{s}_0\right)$ is generated and the entire process is repeated with the goal of converging to the optimal nonlinear trajectory.

2.3. Localization

For the RNN model itself, scalability is enabled by partitioning the model system domain into smaller local patches, with a separate RNN trained for each patch (Pathak et al., 2018). Each local patch is assigned a small radius of "halo" points that allow information from neighboring patches as inputs to the RNN model, while computing a forecast only for the points within the patch. This follows a similar paradigm to that used for the domain decomposition of general circulation models. Localization of a geophysical forecast model is motivated by the presence of locally low-dimensional chaotic dynamics (Oczkowski et al., 2005). In previous works, Arcomano et al. (2020) demonstrated the use of RC for the prediction of global scale atmospheric dynamics by applying the localization scheme described above, while Lin and Penny (2021) further showed that this spatial localization approach could be improved for geophysical systems by applying transformations into Fourier space.

Localization has also been an important tool for scaling DA methods to enable the application to high-dimensional systems (Greybush et al., 2011). We apply localization of the DA using an approach similar to the Local Ensemble Transform Kalman Filter (LETKF; Hunt et al., 2007). In its original formulation, the LETKF computes a separate ETKF analysis at each model grid point, while only assimilating observations within a prescribed localization radius around that grid point. In our RNN-LETKF, we instead choose a radius around local patches. Observations are selected from an area larger than the patch itself based on the RNN localization in order to promote consistency with the analyses computed for neighboring patches. We maintain correspondence with the local RNN model architecture by using a radius that aligns with the input field of the local RNN, which includes the local patch and its halo points. As with the LETKF, the RNN-LETKF local analyses can be computed in parallel, after observation innovations are computed globally and distributed to each local patch. The results of the local analyses are then communicated to the neighboring patches in order to initialize the next forecast.

2.4. Source Data for Training, Validation, and Testing

Here, we describe the underlying model equations that we use to generate data for training, validating, and testing the RNN models. Lorenz (1996) developed a simple model (L96) that includes advection, dissipation, and external forcing to describe basic wavelike dynamics in the atmosphere around a latitude ring. The L96 model is a frequently used test system for DA studies (Abarbanel et al., 2010; Brajard et al., 2020; Chen & Kalnay, 2019; Goodliff et al., 2017; Penny, 2014, 2017), and multiple varieties of RNNs have been applied successfully for emulation of L96 dynamics (Vlachas et al., 2020). The L96 system is defined by a set of ordinary differential equations on a discrete finite cyclic domain,

$$f_{L96}(\mathbf{x}_i) = \frac{d\mathbf{x}_i}{dt} = \mathbf{x}_{i-1}(\mathbf{x}_{i+1} - \mathbf{x}_{i-2}) - \mathbf{x}_i + F_{L96}. \tag{23}$$

The domain can be of any size. We use two configurations, one that is 6-dimensional (L96-6D) and another that is 40-dimensional (L96-40D). We use forcing $F_{L96} = 8.0$, which ensures fully developed chaotic dynamics, meaning that at least one Lypaunov exponent is greater than 0. This implies initial errors will grow exponentially on average. The model is integrated with a time step of $\delta t = 0.01$ model time units (MTUs) using scipy odeint, which is a wrapper for LSODA from the FORTRAN package odepack (Hindmarsh, 1983), and is based on an automatic selection method that chooses between the Adams method (for the nonstiff case) and the backward differentiation formula (BDF) method (for the stiff case) (Petzold, 1983). Lorenz (1996) scaled the coefficients of the model so that the error growth over 1 MTU is roughly equivalent to 5 days in the atmosphere, relative to the state-of-the-art atmospheric models of the time. In operational prediction centers, analyses are often produced with 6-hr, 12-hr, or 24-hr update cycles, thus we will focus on DA cycles ranging up to 0.2 MTU (≈24 hr).

It is assumed that in practice one or more pre-computed numerical simulations or reanalysis data sets will be available for training the surrogate RNN models. To emulate this scenario, the training data are produced using a model integration of 100,000 time steps (200,000 for the L96-40D). This is produced after a spin-up integration of 20,000 time steps from an initial condition in which the first coordinate is set at $x_1(t_{init}) = 0.01$ and the remaining coordinates are set at $x_j(t_{init}) = 0.0, \forall j > 1$. Following the target application described above, we assume data at all model grid points are available for training. Next, a transient integration is performed for 1,000 time steps. The test data are then produced using an additional integration of 100,000 time steps. Test results evaluating the prediction skill of the trained RNN models are shown in Figures A3, A4 and A5 in Appendix A. The transient integration is reserved as a validation data set for the RNN training. The DA experiments are conducted using the test data set.

2.5. Experiment Design

Three RNN models are first trained and then used as surrogate models in the DA experiments. RNN model 1 and model 2 are trained and tested using the L96-6D data set, while model 3 is trained and tested using the L96-40D data set. Model 1 uses a larger hidden/reservoir state to improve accuracy, while model 2 uses a smaller hidden/reservoir state to sacrifice accuracy for improved computational efficiency. Further details of the model design and training are provided in Appendix A.

We compare results using a variety of DA configurations. All forecast model integrations are computed with a time step of $\delta t = 0.01$. Unless otherwise noted, we use the following parameter settings: each DA experiment is integrated for 100 MTU (or 10,000 time steps); we use observation noise of $\sigma_{noise} = 0.5$ and a corresponding estimated observation error of $\sigma_{obs} = 0.5$ to form $\mathbf{R}$; observations are sampled every $\tau_{obs} = 0.02$ MTU (≈2.4 hr); and we use an analysis cycle window of $\tau_{da} = 0.2$ MTU (≈24 hr).

We use a 10-member ensemble for the RNN-ETKF applied to the L96-6D data set, a 30-member ensemble for the RNN-LETKF applied to the L96-40D data set, and for the purposes of this discussion a 1-member “ensemble” for the RNN-4DVar applied to the L96-6D data set. All DA experiments are initialized by first preparing a set of perturbed spin-up data sets, one for each ensemble member, applying Gaussian random noise with standard deviation $\sigma_{init} = 0.5$ to the true state over a 1,000 time step window (2,000 for the L96-40D system). Each RNN ensemble member is synchronized with its corresponding perturbed data set in order to produce an initial ensemble of hidden/reservoir states that reflect the uncertainty present in the noisy input data.

As noted by Lorenc (2003), additional covariance inflation is needed in the presence of model error if that error is not addressed explicitly. Covariance inflation is also typically needed due to the use of a finite ensemble size. We found an inflation parameter of 1%–5% (i.e., $\gamma = 1.01$–1.05) to be effective for the ETKF when applied with the “perfect” numerical model. To account for model error in the RNN, we increase the multiplicative inflation parameter to 20% ($\gamma = 1.2$) for the RNN-ETKF and 30% ($\gamma = 1.3$) for the RNN-LETKF. Lorenc and Jardak (2018) note the importance of the background error covariance matrix used by 4D-Var. The canonical 4D-Var uses a static climatological background error covariance matrix $\mathbf{B}_x$. Here, $\mathbf{B}_x$ is derived using a time average of the dynamic ETKF estimates, rescaled so that the mean on the diagonal has variance equal to $\sigma_b^2 = \sigma_{obs}^2 = 0.25$. It uses an analysis time at the start of the window, and uses 2 outer loops. For the RNN-4DVar, we apply the following transformation applied to the background error covariance $\mathbf{B}_x$ to map to the hidden/reservoir space,

$$\mathbf{B}_s = \mathbf{W}_{in}\mathbf{B}_x\mathbf{W}_{in}^T. \tag{24}$$

To mimic a realistic scenario of geophysical prediction, we focus mainly on cases where the variables are sparsely observed. If not otherwise stated, the L96-6D model is observed only at the first, second, and fourth nodes. For the L96-40D model, we limit the observing network to only 15 nodes.

3. Results

3.1. Assessment of Error Growth Rates

An essential consideration of DA is the behavior of long- and short-term error growth, which can be characterized by the Lyapunov exponent (LE) spectrum and finite-time (also known as “local”) Lyapunov exponents (FTLEs; Abarbanel, 1996; Abarbanel et al., 1992). If one considers DA as the synchronization of a model with the natural

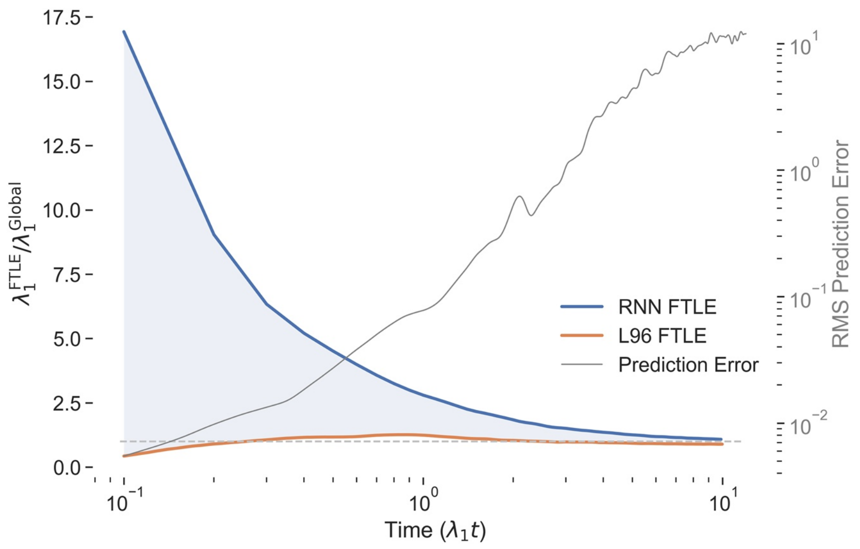

Figure 2. Convergence of the leading finite-time Lyapunov exponents (λ_1) for a trained RNN (model 1 in Table A2) averaged over 100 initial conditions of the L96-6D system. As the recurrent neutral networks (RNN) is integrated for longer periods of time, the error growth rates generated by the RNN model become more accurate. However, over the same period there is an exponential growth of errors in initial conditions.

process from which measurements are drawn, then the conditional LE spectrum of this coupled model-nature system must be driven negative to ensure the model synchronizes with the observed system (Penny, 2017). Previous studies (Griffith et al., 2019; Pathak et al., 2018; Platt et al., 2021) have already shown that reservoir computing can be used to reproduce the Lyapunov spectrum of the source system. Brajard et al. (2020) and Bocquet et al. (2020) also showed this capability with alternative ML methods. The Lyapunov spectrum characterizes the long time average exponential growth rates of small errors in the system trajectory. However, at very short timescales, we find that the error growth rates of our trained RNNs are not well representative of the error growth rates produced by the source system dynamics. We find instead that the FTLEs of the RNN dynamics converge toward the Lyapunov exponents of the source system dynamics over a transient period of a few Lyapunov timescales (Figure 2).

To examine error growth rates, we also examine the forecast error covariance matrices approximated by an ensemble forecast, and compare it to the climatological (i.e., time-averaged) forecast error covariance. Assuming the forecast error covariance matrix is estimated as,

$$\mathbf{P}_t = \frac{1}{k-1}\mathbf{X}_t^f\left(\mathbf{X}_t^f\right)^T \tag{25}$$

where the columns of $\mathbf{X}_t^f$ are perturbations around the forecast ensemble mean at time t, then the forecast error correlation matrix $\mathbf{C}_t$ rescales $\mathbf{P}_t$ between [0,1] as,

$$\mathbf{C}_t = \text{diag}(\boldsymbol{\sigma}_t)^{-1}\mathbf{P}_t\text{diag}(\boldsymbol{\sigma}_t)^{-1}, \tag{26}$$

where $\boldsymbol{\sigma}_t = \sqrt{\text{diag}\left(\mathbf{P}_t\right)}$ is a vector of standard deviations that correspond to the diagonal elements of $\mathbf{P}_t$. While the growth rates of errors in the RNN models take some time to converge to the true growth rates, the forecast error correlations appear to be estimated relatively accurately at short lead times (an example is shown in Figure 3). This indicates that while it may be desirable to improve the convergence rates of the FTLEs produced by the RNN, the effect can be compensated for by using a scalar multiplicative inflation applied to the forecast error covariance matrix. This multiplicative inflation will be used below in the context of ensemble-based DA.

3.2. Control RNN Case

To demonstrate the need for DA, we commence our cycled forecast experiments with a control case that sets a baseline for the performance of the RNN without using DA. Here, observations are inserted directly into the

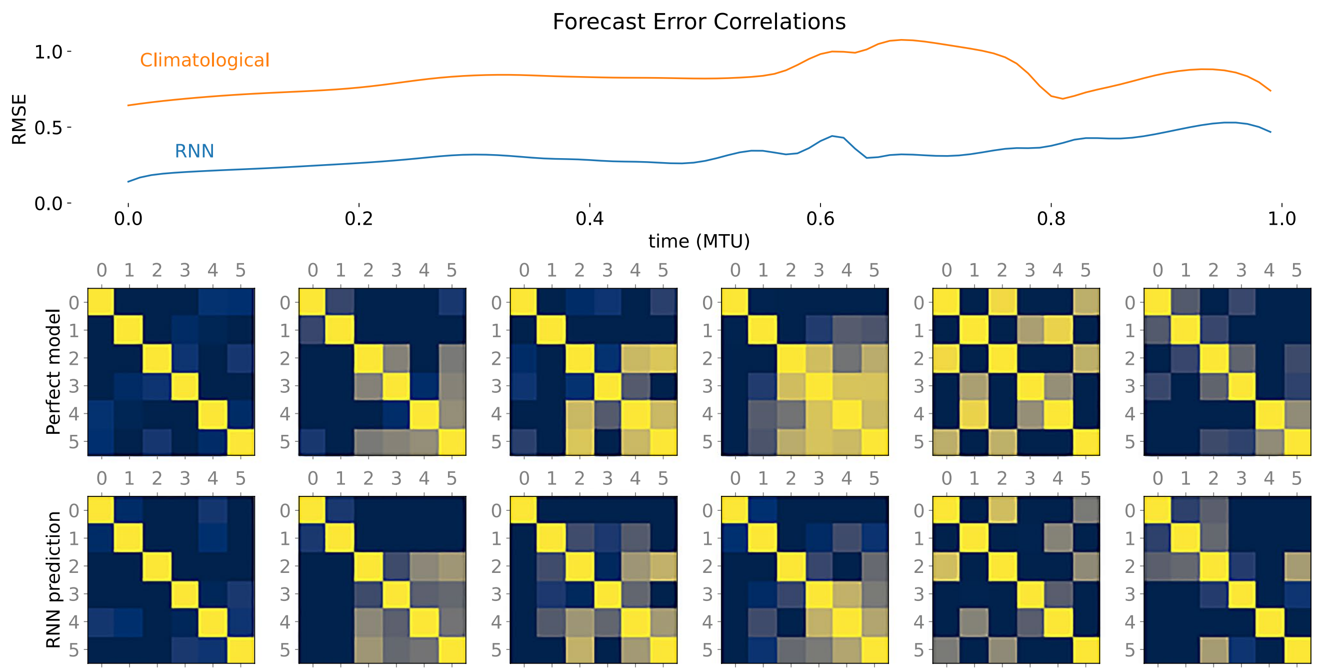

Figure 3. (Top) RMSE of the forecast error correlation matrix over time for the L96-6D system, comparing an example recurrent neutral networks (RNN) ensemble forecast (blue) to the climatological error correlation matrix (orange), both evaluated versus a 100-member ensemble forecast using the perfect reference model. The initial conditions are sampled from the test data set, and initial ensemble perturbations are generated using a Gaussian distribution with standard deviation 0.1. (Bottom) Forecast error correlation matrix for the reference perfect numerical model and the RNN model over the same forecast period, valid at times 0.0, 0.2, 0.4, 0.6, 0.8, and 1.0. The color scale ranges from 0 to 1.

RNN as components of the vector $\mathbf{x}(t_i)$ in Equation 1. For the sparse observing case, only the components that are observed are replaced, while the remaining components are predicted by the RNN.

Lu et al. (2017) examined a "sparse in space" case that limited the forcing of an RC model to only a subset of inputs. Their results showed that synchronization of the full state can be achieved even when observing only a subset of the variables of the system. However, we find that this type of direct insertion method for synchronization fails as the observations become more sparse in time (see Figure 4 and the middle and bottom rows of Figure 5).

Before further evaluating the RNN, we first consider for reference the case of direct insertion of observations into the original L96-6D numerical model, using the update equation,

$$\mathbf{x}^a = \mathbf{x}^b + \mathbf{H}^T\left(\mathbf{y}^o - \mathbf{H}\mathbf{x}^b\right). \quad (27)$$

As should be expected, providing perfect observations of all variables at every model time step ($\delta t = 0.01$) produces exact synchronization between the driver signal and the numerical model trajectory. Increasing the time step between observations as high as 0.2 MTU does not significantly degrade the state estimates, with errors peaking at 2.5e−7. Increasing the observational noise generally increases the error in the state estimates by a similar magnitude. When the number of observed variables is reduced (e.g., to 3 or 2 out of 6), the system experiences transient synchronization with occasional bursting. While still using perfect observations, combining reduced observations (e.g., 50%) and using longer time steps (e.g., $\delta t = 0.1$) actually improves stability compared to using a time step of $\delta t = 0.01$ and leads to synchronization. However, when observation noise is added to this combination of sparseness in space and time, the bursting phenomena return, particularly in the unobserved variables (e.g., see top row of Figure 5).

The situation is quite different with the RNN model. Increasing the time step between the (noise-free) observations significantly degrades the RNN estimates, first adding high frequency oscillations (e.g., with $\delta t = 0.02$ to $\delta t = 0.1$), and then leading to trajectories with little discernible connection to the L96 dynamics (e.g., at $\delta t = 0.2$). Recall that for these experiments the RNN is trained on data that have a temporal resolution matching

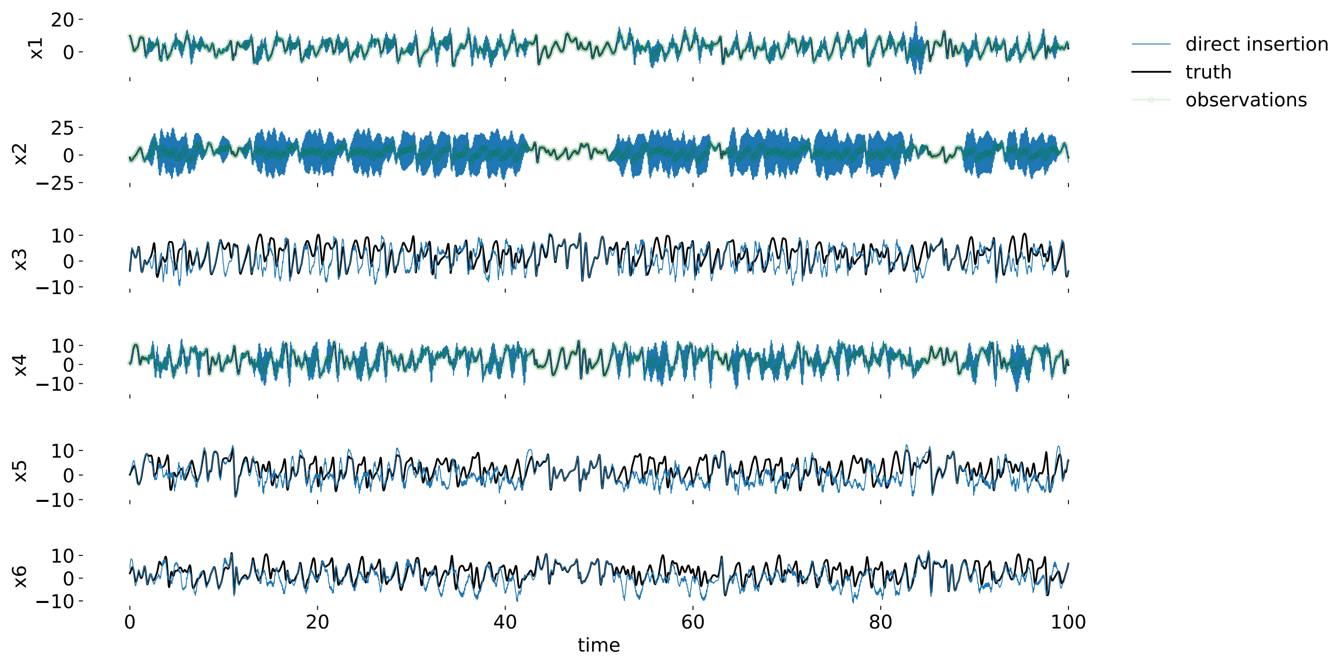

Figure 4. Direct insertion using the RNN with perfect ($\sigma_{noise} = 0$) observations of the system at points (1, 2, and 4), sampled with $\tau_{obs} = 0.05$ (≈6 hr), which is every 5 model time steps. The RNN alone cannot successfully recover the true trajectory when observations are sparse and noisy. This figure provides a corresponding entry in Figure 5.

the underlying model time step $\delta t = 0.01$. Reducing the frequency of the input driving signal allows the hidden/reservoir state to drift. Observing frequently ($\delta t = 0.01$) but removing the observation of one variable degrades the estimates of that variable without noticeably affecting the rest, while removing the observation of more than one variable can cause occasional degradation of the remaining observed variables. Due to the presence of the hidden/reservoir state, which maintains a memory of the past trajectory, the RNN itself is relatively insensitive to the introduction of noise to the observations. When observing the full system state, noisy observations supplied as driving data to the RNN simply increase high frequency noise in the analyzed state estimate, without resulting in divergence between the RNN and the true signal. Combining any of these constraints on the RNN appears to have additive effects. A comparison of the errors in the RNN using direct insertion with a range of observation noise and observing frequency (of which Figure 4 is one instance) is shown in Figure 5.

We note that even for this simple L96 model, the total set of direct insertion experiments using RNN model 1 was about 10% faster, and RNN model 2 about 30%–40% faster, than the total set of direct insertion experiments using the conventional numerical integration of the L96 differential Equation 23. We do not claim that these results can be easily extrapolated to other applications. However, we do emphasize that the projection of the system dynamics to the higher dimensional hidden/reservoir state does not necessarily imply that the computations become more costly.

To summarize-simply providing the sparse and noisy observation data directly to the RNN is not adequate for initializing forecasts, which provides motivation for the use of a more sophisticated DA strategy.

3.3. RNN-DA With Sparsely Observed Dynamics

DA methods provide most of their value when observations are sparse and noisy. Here, we restrict the observing network to only the first, second, and fourth nodes of the 6-dimensional cyclic L96 system (L96-6D). This leaves two patches that are unobserved for the duration of each experiment - the third node and the combined fifth and sixth nodes. We noticed no qualitative differences in other configurations of the observing system layout at the same 50% coverage level.

We first implement the RNN-ETKF and compare it to the conventional ETKF using the "perfect" numerical model. We find that with the exception of the case in which analyses are updated frequently ($\tau_{DA} = \tau_{obs} = \delta t = 0.01$), the

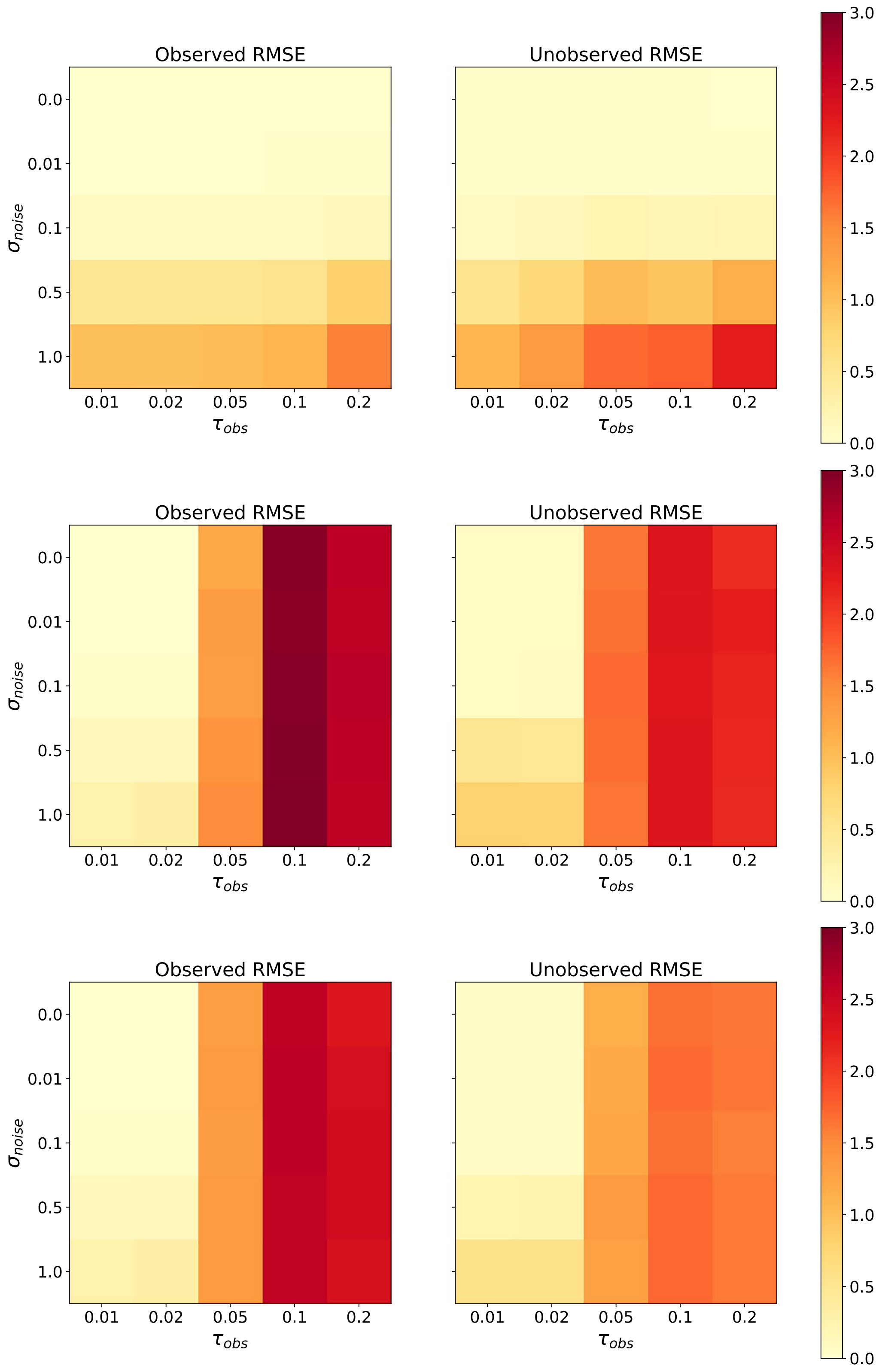

Figure 5. Normalized RMSE using direct insertion using (Top) the 'perfect' numerical model (Middle) RNN model 2, and (Bottom) RNN model 1. Each is applied to the L96-6D system integrated over 100 MTU, varying the observation noise σ_{noise} and the observation time step τ_{obs}. (Left) RMSE of observed points (indexes 1, 2, and 4). (Right) RMSE of points that are not observed (indexes 3, 5, and 6). A normalized RMSE of 1.0 equals the L96 system's climatological standard deviation. Note that the conventional model is more sensitive to increased noise, while the RNN model is more sensitive to the observing frequency.

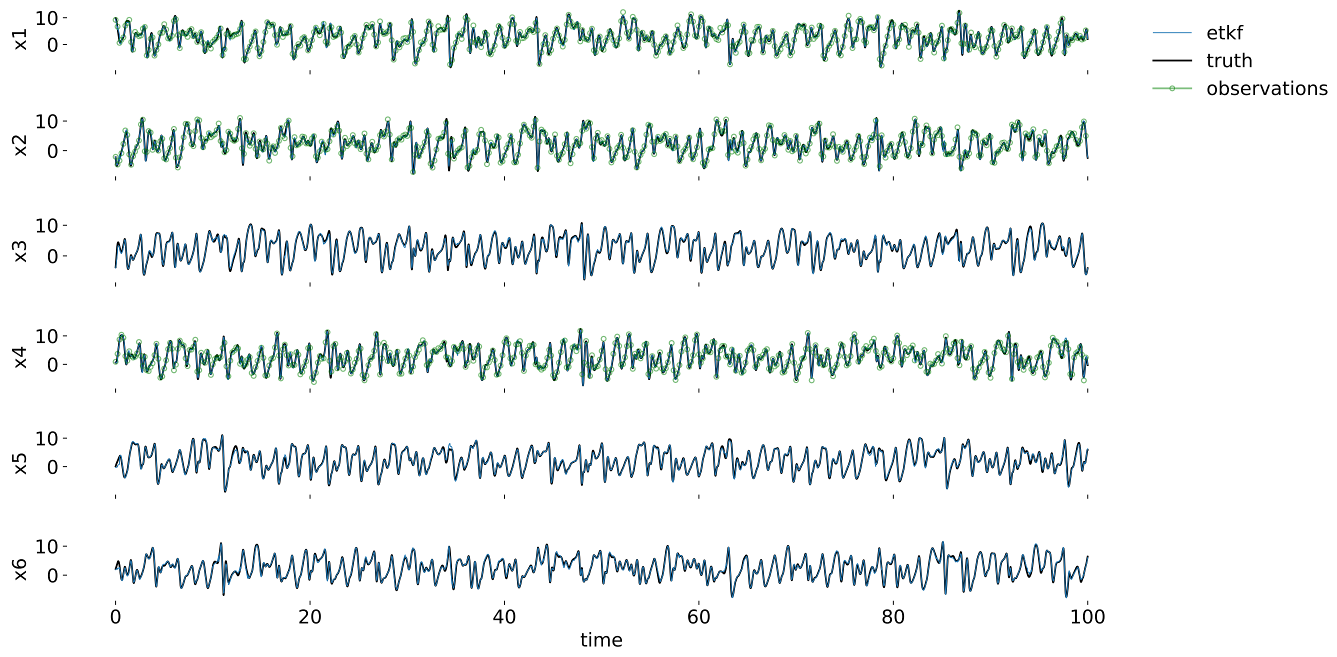

Figure 6. The RNN-ETKF, assimilating observations at only three points (1, 2, and 4) at increments of $\tau_{obs} = 0.2$ (i.e., every 20 model time steps), converges to the true system trajectory within a few time steps. Note the true and estimated trajectories are nearly indistinguishable. This figure provides a corresponding entry in Figure 7.

RNN-ETKF using both RNN models 1 and 2 performs quite well in comparison (Figures 6 and 7). We note that the total set of ETKF experiments using RNN model 1 had about equal run time, while RNN model 2 was about 25% faster, compared to the run time of the total set of ETKF experiments using the conventional numerical integration of the L96 differential Equation 23, noting again the RNN form does not necessarily imply higher computational cost.

The RNN-4DVar method performs well when the observation noise is small, but is sensitive to increasingly sparse and noisy observing sets (Figures 8 and 9). As the underlying RNN model is improved (from model 2 to model 1 in Table A2), this appears to improve the performance of the 4D-Var correspondingly. The sensitivity of the RNN-4DVar to observation noise may be exacerbated by errors in the RNN model equations from which the TLM and adjoint operators are derived, and also the approximated background error covariance matrix. A further drawback is that the experiments using the RNN-4DVar required 1–2 orders of magnitude more computational time than the conventional 4D-Var applied using the numerical integration of the L96 differential Equation 23 and its TLM and adjoint equations.

The difference between FTLEs estimated by the RNN and the numerical model at short lead times indicates that the linearized dynamics (i.e., the TLM and adjoint) are not well represented at these timescales. The RNN models used here generally underrepresent the magnitude of error growth at short timescales. This affects the ETKF as well but is alleviated by the application of multiplicative inflation, and gives some explanation for why the ETKF is more stable than 4D-Var. The strong-constraint 4D-Var used here assumes a "perfect' model. We expect that transitioning from the strong-constraint 4D-Var formulation to the weak-constraint 4D-Var approach should further improve the RNN-4DVar performance, as the latter explicitly accounts for errors in the model.

3.4. Scaling RNN-DA to Higher Dimensions

Next we demonstrate that the components of the RNN-DA system can scale as the system size increases. Given the results of the previous section, we will focus on the RNN-ETKF. We increase the dimension of the L96 system from 6 to 40 in order to demonstrate the scalability of the system via the aforementioned localization scheme. The system is partitioned into 20 local subgroups of 2 points each, and a separate RNN is trained for each local subgroup. The input signal to each RNN is made up of the 2 points in its associated local subgroup, plus 4 neighboring points on either side, giving each RNN an input dimension of 10 and an output dimension of 2.

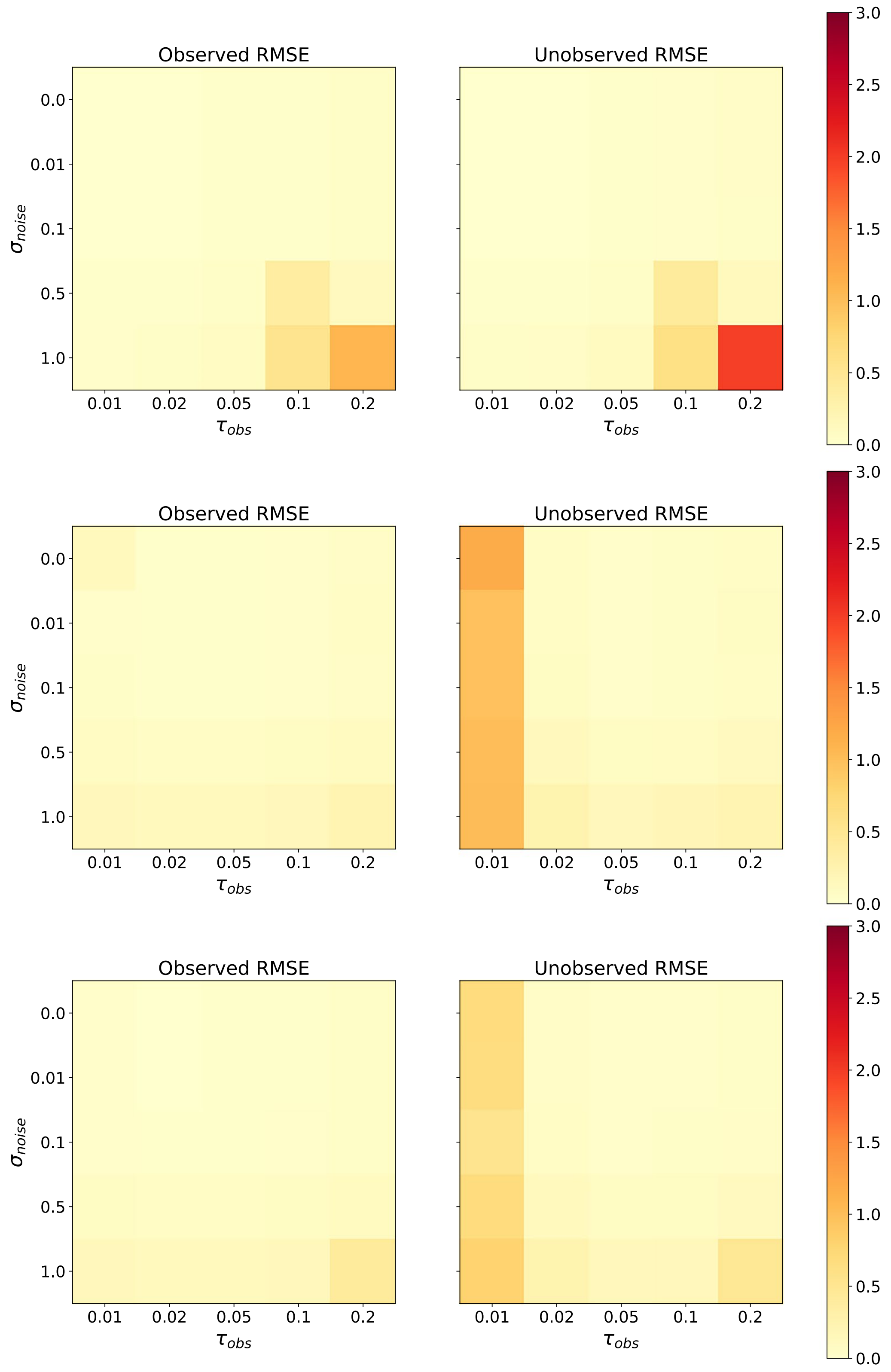
Observed RMSE
Unobserved RMSE
σ_{noise}
τ_{obs}
0.0
0.01
0.1
0.5
1.0
0.01
0.02
0.05
0.1
0.2
3.0
2.5
2.0
1.5
1.0
0.5
0.0

Figure 7.

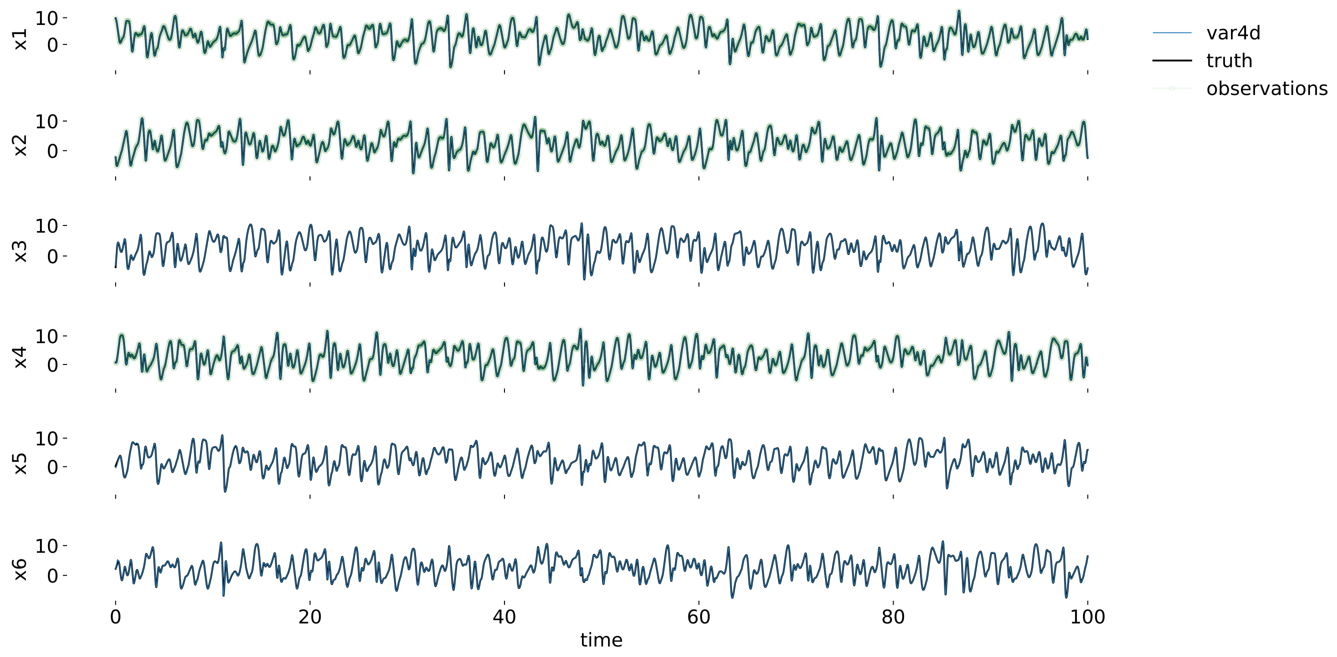

Figure 8. The RNN-4DVar, assimilating observations at only three points (1, 2, and 4) at increments of $\tau_{obs} = 0.02$ (i.e., every 2 model time steps), with observation noise set to $\sigma_{noise} = 0$. The analysis cycle is $\tau_{obs} = 0.2$ (i.e., every 20 model time steps). This figure provides a corresponding entry in Figure 9.

For each local subgroup, observations are assimilated if they are located within the range of the local RNN input domain. In this example we observe 15 nodes of the system, leaving 25 nodes unobserved.

The Normalized Root Square Error (NRSE) of the trajectory estimated by the RNN-LETKF is shown in Figure 10 using observation noise $\sigma_{noise} = 0.5$, and in Figure 11 with reduced observation noise $\sigma_{noise} = 0.1$. An additional perturbation is added at the first time step to ensure that the background estimate of the system state is far from truth. After roughly 20 MTU, the cycled DA system converges and provides an accurate estimation of the system trajectory. The lower panels in Figures 10 and 11 show the Normalized Root Mean Square Error (NRMSE) of the observed and unobserved nodes of the L96-40D system separately. The RNN-LETKF system appears to provide an accurate transfer of information from the observed to the unobserved variables.

Compared to the L96-6D system, the L96-40D system required the use of a larger hidden/reservoir dimension in order to generate accurate predictions. Specifically, here we used a 6,000 (2,000) dimensional reservoir during prediction (training), compared to 1,200 and 800 for Models 1 and 2, respectively. However, we note that the larger reservoir is chosen to compensate for the larger input signal dimension (which here is 10D for each of the 20 reservoirs) not the overall increase in the global state space dimension. Applying the RNN-LETKF algorithm to higher dimensional systems can therefore be achieved by increasing the number of reservoirs, which require minimal communication between neighboring groups.

Finally, we note that predictions for the L96-40D system required a larger training data set of 200,000 time steps, and a longer spin-up time of 2,000 time steps, see details in Table A2 (Appendix A). With the localization scheme employed, training for the microscale parameters is completely independent for each subgroup, and is easily parallelized. Additionally, with much longer training data sets, the training of the microscale parameters can easily be applied in batches with equivalent results, with the macroscale parameters estimated from a sample of batches.

Figure 7. Normalized RMSE of (Top) conventional ensemble transform Kalman filter (ETKF) using the "perfect" numerical model, (Middle) the RNN-ETKF using recurremt meural networks (RNN) model 2 (hidden/reservoir dimension 800), and (Bottom) the RNN-ETKF using RNN model 1 (hidden/reservoir dimension 1,600). All are applied to the L96-6D system integrated over 100 MTU, varying the observation noise σ_{noise} and the observation time step τ_{obs}. The analysis cycle is adjusted for each case so that $\tau_{DA} = \tau_{obs}$. (Left) RMSE of observed points (indexes 1, 2, and 4). (Right) RMSE of points that are not observed (indexes 3, 5, and 6). Notably, the RNN-ETKF outperforms the conventional ETKF (which uses the perfect model) when both the observation noise and observing time step are large.

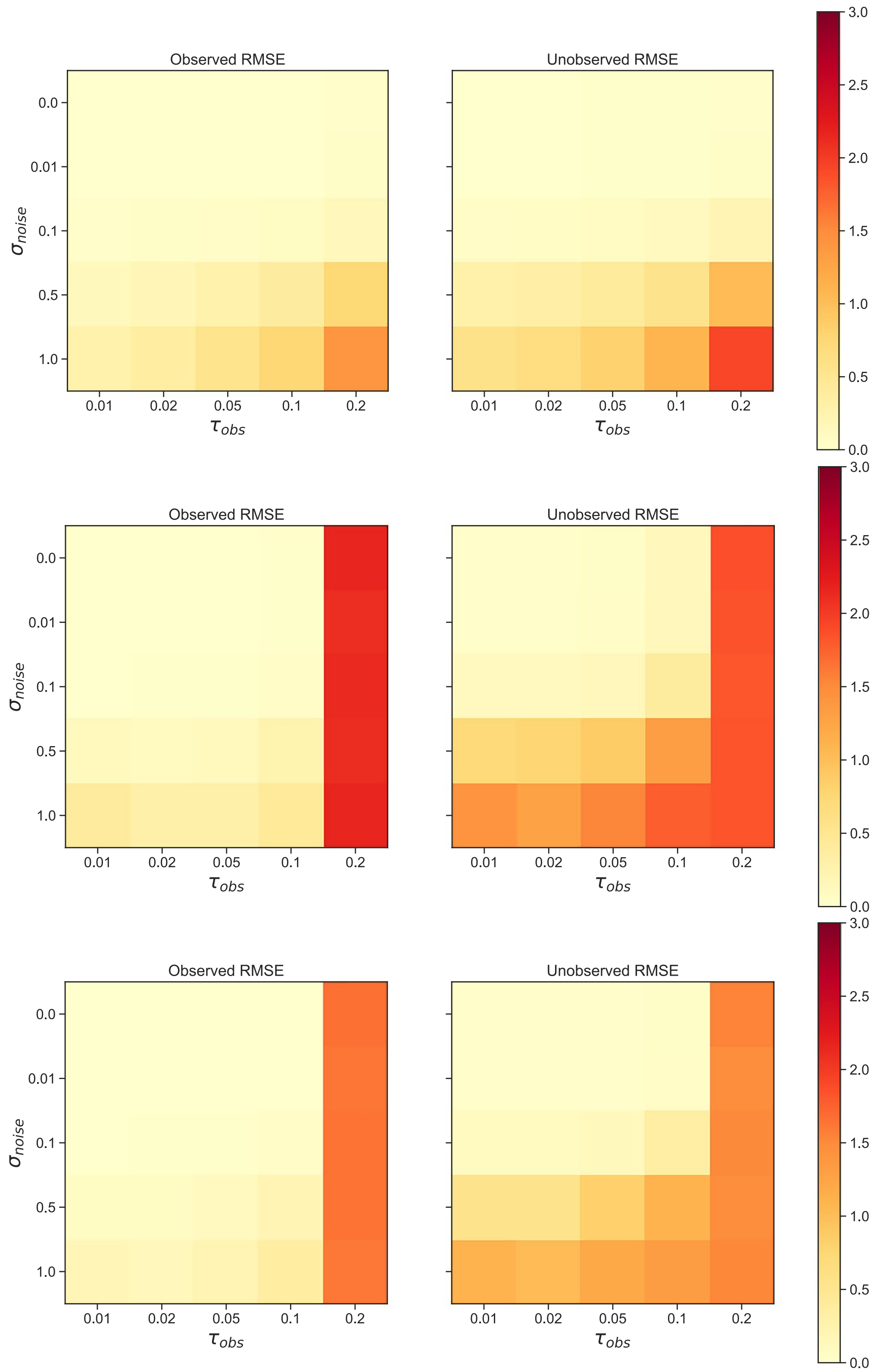

Figure 9. Normalized RMSE of (Top) conventional 4D-Var using the "perfect" numerical model, (Middle) the RNN-4DVar using RNN model 2 (hidden/reservoir dimension 800), and (Bottom) the RNN-4DVar using RNN model 1 (hidden/reservoir dimension 1,600). All are applied using an analysis cycle of $\tau_{DA} = 0.2$ (i.e., every 20 model time steps). The RNN-4DVar performs best with frequent observations and relatively low observation noise, but otherwise has degraded performance.

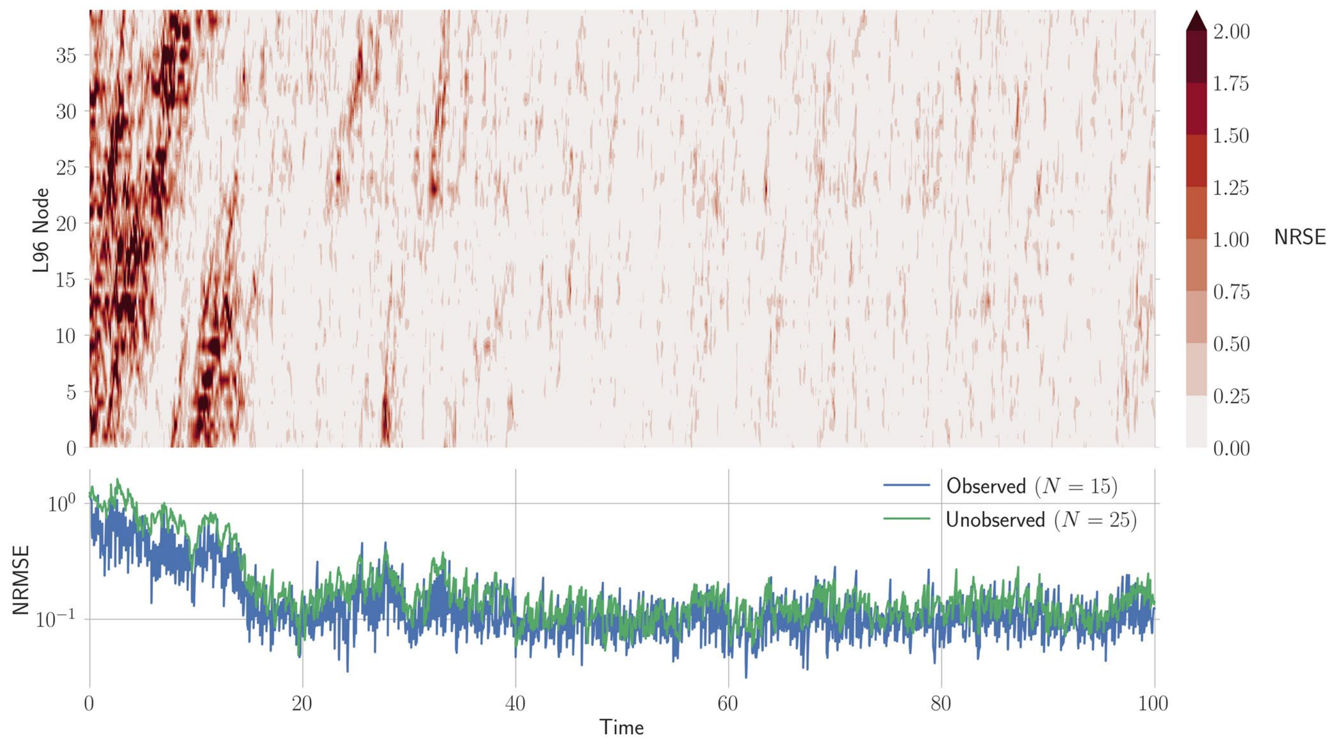

Figure 10. Normalized error of the RNN-LETKF based state estimation for the L96-40D system using RNN Model 3. (Top) Normalized Root Square Error (NRSE) shown for each node of the L96 system (*y*-axis). (Bottom) NRMSE computed separately for the observed and unobserved nodes of the Lorenz system, with 15 nodes of the system observed. Note the *y*-axis is logarithmic in the lower plot. The error in both plots are normalized by the temporal standard deviation of the true trajectory. The RNN-LETKF uses a 30 member ensemble, with $\sigma_{obs} = \sigma_{noise} = 0.5$, and macroscale parameters indicated by RNN model 3 in Table A2. The observed nodes of the system are [0, 3, 5, 8, 10, 14, 16, 19, 20, 25, 27, 30, 34, 36, 39].

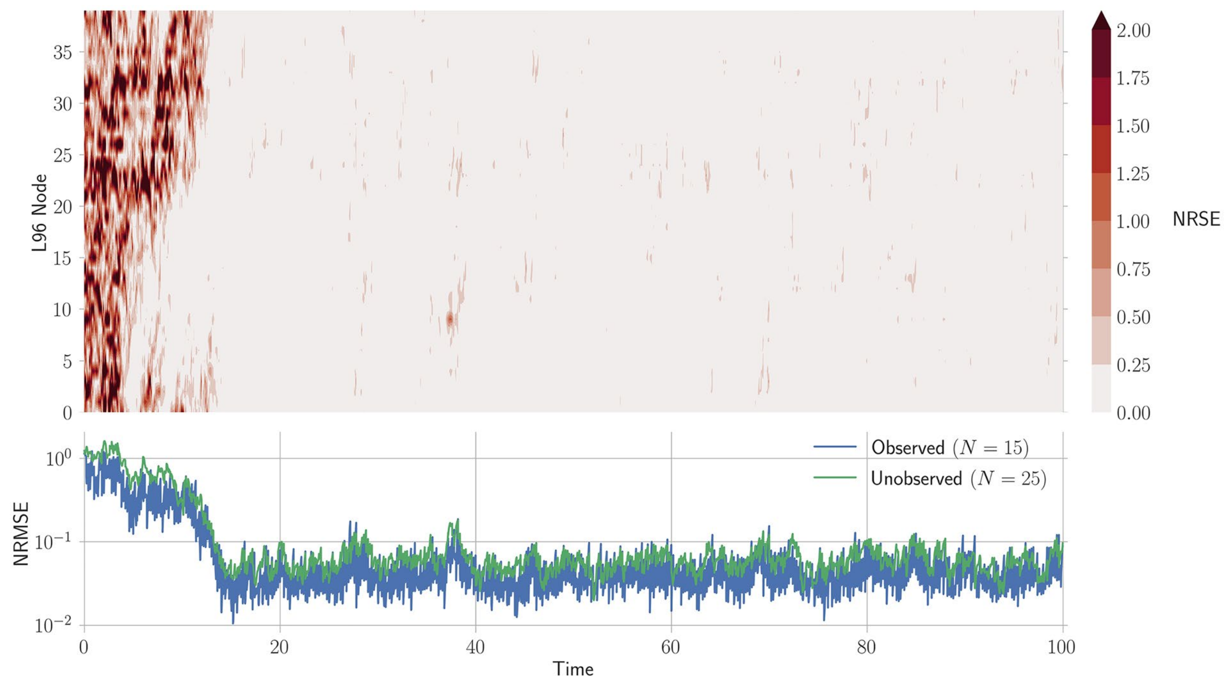

Figure 11. As in Figure 10, but with $\sigma_{noise} = 0.1$.

4. Conclusions

One of the most common procedures in the field of data assimilation (DA) is to combine a computational model with observations to estimate the state of a partially observed system. This procedure is used for applications such as initializing models to make real-time forecasts or creating historical reconstructions based on a limited archive of observation data. An essential element of modern DA algorithms is the expectation that the forecast model responds accurately to small perturbations in the initial conditions. By integrating recurrent neural networks (RNN) with the ensemble Kalman filter and 4D-Variational DA methods, we have demonstrated that RNNs can produce reasonable representations of the system response to uncertainty in initial conditions. Critical to this demonstration was the assimilation of sparse observation data, which requires sufficiently accurate ensemble forecast error covariance statistics and tangent linear model dynamics to propagate information from observed to unobserved variables.

Comparing the application of a DA method using RNN-based forecast models to the same DA method using a "perfect" model provides a useful analogue to the real-world scenario in which an imperfect numerical model is used to estimate the true state of a natural system. Beyond the practical applications that a ML model may have, there is also much to be learned from determining the necessary elements for ML models to be used in applications like reanalysis and operational numerical weather prediction. One example provided was the reproduction of the finite-time Lyapunov exponents (FTLEs).

We note that while the RNN-DA methods have been applied to training data generated from a known model, as long as adequate training data exists the methods apply equally to systems for which no known theoretical or computational model is available. Further, the RNN-4DVar can be produced for models in which the tangent linear and adjoint models are either not available or are too difficult to calculate. The RNN-4DVar could easily be implemented in hybrid forms in which a conventional numerical model is used for the outer loop and RNN-based tangent linear model is used for the inner loop, or alternatively the RNN could be used as the nonlinear model in the outer loop to reduce computational costs while still using a numerical tangent linear model and adjoint in the inner loop. We also note that while the RNN models were trained once on historical data and held fixed during the RNN-DA cycling, online retraining of the RNN model can be performed during the DA cycle, e.g. as suggested by Brajard et al. (2020) and later explored by Bocquet et al. (2021) and Farchi et al. (2021). For the RNN used here, it is straightforward to perform continued online training using low rank matrix updates to update the linear readout operator $\mathbf{W}_{out}$.

Combining the localization schemes for the RC architecture (Pathak et al., 2018) and ETKF (Hunt et al., 2007) produces a data-driven forecast system that can scale to high-dimensional problems. With this approach, communication is required between neighboring patches only during the prediction phase and during the DA cycle. The main design decision when scaling to high dimensional problems is determining the input dimension for each local group's reservoir, as the reservoir size must increase accordingly. The RC architecture used by Arcomano et al. (2020) to make predictions of the global atmospheric state use a reservoir size of 9,000 for each local 132-dimensional state vector, suggesting that this increase is tractable for realistic applications. However, we consider the development of an optimized RNN architecture that can minimize the reservoir size an important activity for future work.

Further methods are needed to optimize the design and training of the ML model used in this study to replace the numerical forecast model. More development is needed in ML modeling of chaotic dynamics to ensure a rapid convergence of the finite time Lyapunov exponents toward the true values. While a simplified RNN resembling the reservoir computing approach was applied here, we expect that our approach could be applied with more sophisticated types of RNN that also use hidden state representations, such as Long Short Term Memory or Gated Recurrent Unit architectures.

Appendix A: Details of the RNN Training

Bayesian optimization is an optimization technique that "minimizes the expected deviation from the extremum" of a target loss function Močkus (1975). The technique is useful mainly for global optimization of expensive nonlinear functions for which no gradient is computed. The specific algorithm used here is the efficient global optimization algorithm described by Jones et al. (1998) and implemented by Bouhlel et al. (2019), using a Kriging

Table A1
Hyperparameters Used for the Bayesian Optimization. For Model 3, We Perform the Optimization Using a Reduced Reservoir Size (2,000) due to the Computational Cost of the Algorithm. Predictions are Made With a Larger Reservoir (6,000), See Text for Details

Hyperparameter	RNN model 1	RNN model 2	RNN model 3
hidden/reservoir dimension	1,600	800	2,000 (6,000)
size of training set	100,000	100,000	200,000
number of long forecasts (M)	100	100	100
length of long forecasts (N) in MTU	10.0	10.0	10.0
sparsity of reservoir matrix	99%	99%	99%
input weighting (σ) limits	[0.001, 1.0]	[0.001, 1.0]	[0.001,1.0]
leak rate (l) limits	[0.001, 1.0]	[0.001, 1.0]	[0.001, 1.0]
spectral radius (ρ) limits	[0.1, 1.5]	[0.1, 1.5]	[0.1, 1.5]
Tikhonov parameter ($\log \beta$) limits	[log(1e−8), log(1.0)]	[log(1e−8), log(1.0)]	[log(1e−8), log(1.0)]

surrogate model. In short, the algorithm starts by sampling a number of initial points over the search space. It then fits a Gaussian process regression across those points, enabling interpolation and extrapolation. After the fit, the algorithm computes the "expected improvement" of searching a new region of the space and then chooses points based on maximizing this criterion.

The loss function chosen is based on computing the scaled mean squared error (MSE) of long-range predictions over the test data set, as described by Equation 6. Hyperparameters of this optimization are shown in Table A1, and include the length of the training data for the RC, the number of forecasts in the validation set, and the length of those forecasts over which the MSE is computed. The following options for the Bayesian optimization algorithm are kept fixed: the number of iterations of the algorithm (15), the number of parallel samples computed (4), and the number of optimization start points (100).

All RNNs here use a sparsity of 99% (density of 1%) for the reservoir adjacency matrix $\mathbf{W}_{res}$. For L96-6D experiments, we use the following hyperparameters for the Bayesian optimization: $M = 100$ initial points for validation forecasts, chosen randomly without replacement; a forecast length of 10.0 MTU, or $N = 1{,}000$, equal to ≈10 Lyapunov timescales. The macroscale parameters learned from the Bayesian optimization process are provided in Table (A2).

The Bayesian optimization algorithm is computationally intensive for Model 3, as each iteration of the algorithm requires training the microscale parameters for 20 localized RC models and evaluating their forecast skill. Lukoševičius (2012) suggested that using a reduced reservoir size while searching for optimal hyperparameters is an effective means to reduce computational costs. Vlachas et al. (2020) also showed that increasing the reservoir size for a localized RC model trained on the L96 system with fixed macroscale parameters simply increases the valid prediction time. Thus, to make the training of Model 3 more tractable, we use a reduced reservoir size within the Bayesian optimization algorithm to identify the best-performing macroscale parameters (Table A2). We use a hidden/reservoir dimension of 2,000 during the optimization, while for all Model 3 forecasts used in DA experiments and statistical tests we use the larger hidden/reservoir dimension of 6,000 (Figures 10 and 11, and Figure A5).

Table A2
Trained Values of the Macroscale Parameters for the RNNs Used as Forecast Models by the Data Assimilation Algorithms

Trained Parameter	RNN model 1	RNN model 2	RNN model 3
spectral radius (ρ)	0.10036271	0.10000000	0.34378377
input weighting (σ)	0.06627321	0.05343709	0.05219330
leak parameter (l)	0.70270733	0.69460913	0.40813549
Tikhonov parameter (β)	exp(−18.41726026)	exp(−14.33030495)	exp(−12.53138825)

A visualization of the macro loss function landscape in the (σ, ρ) plane is shown in Figure A1 based on RNN forecasts of the 6-dimensional Lorenz 96 model (described in Section 2.4). When $M = 1$, the macro loss function exhibits many local minima. This indicates that the resulting trained RNN may produce accurate forecasts for a particular set of initial conditions, but does not generalize well to other points on the system attractor. Evaluating forecasts from a greater number of initial conditions results in a smoother loss function landscape. However, increasing M also carries a corresponding increase in cost for the evaluation of the macro loss function. We select $M = 100$ to balance the computational cost and the diminishing returns observed beyond this point.

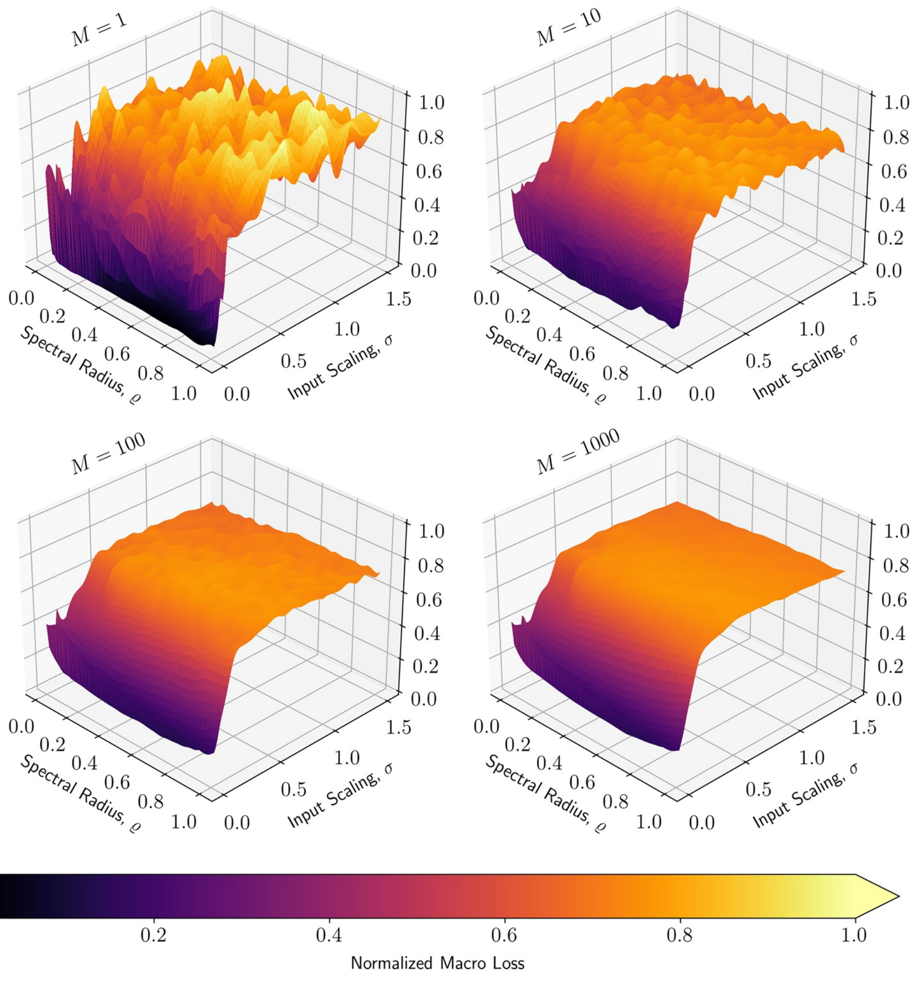

Figure A1. Normalized macro loss function, $\mathcal{L}_{macro}$ (see Equation 6) for various M visualized in the (σ, ρ) plane. Increasing M regularizes the loss function, revealing an approximate global minimum in parameter space. The loss function shown here is based on an 800 dimensional hidden/reservoir state RNN model prediction of the L96-6D system.

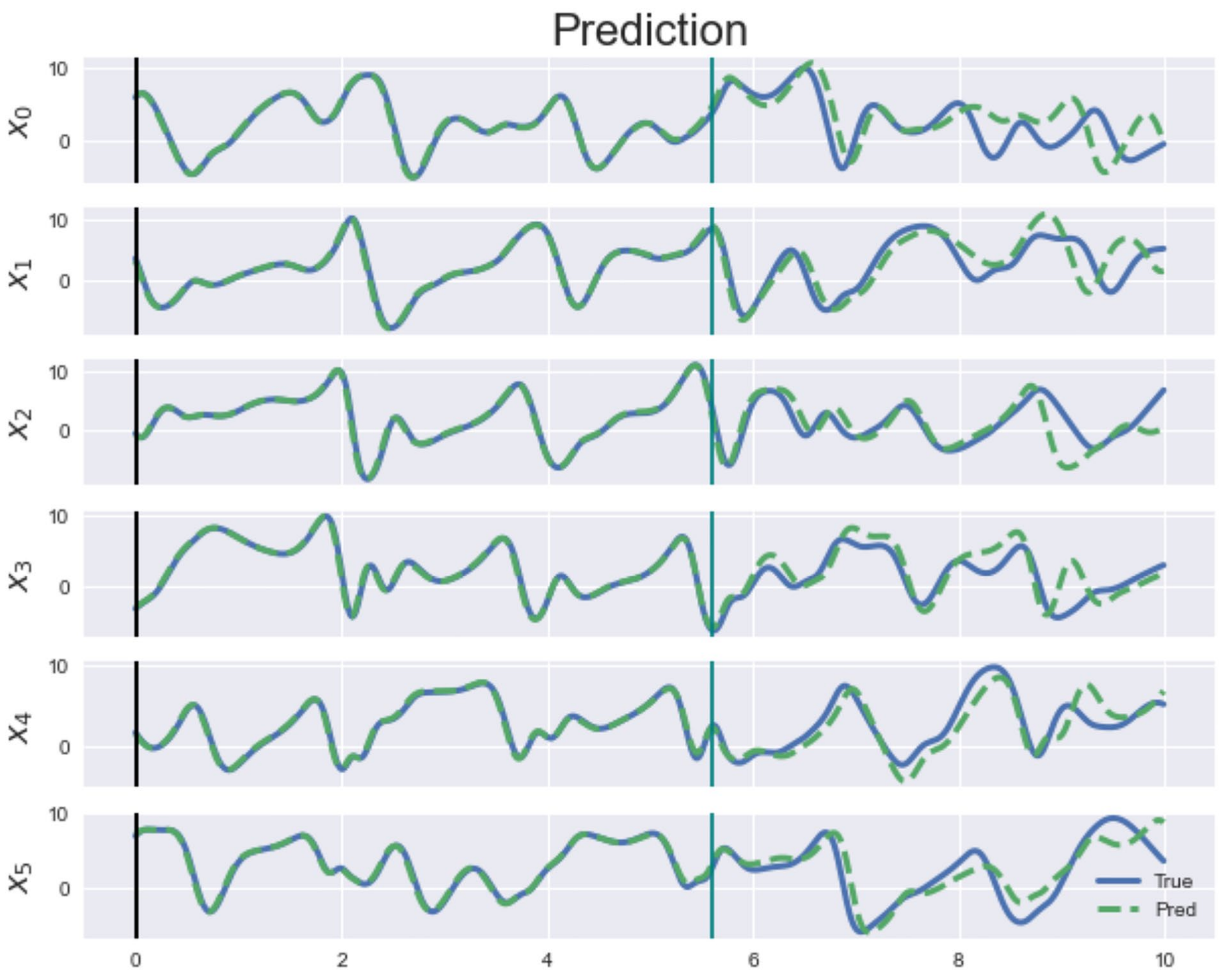

Figure A2. A free forecast of the RNN model 1 initialized from a random point in the test data set at time 0, compared to the true trajectory. The valid prediction time (VPT) using $\epsilon = 0.2$ is marked as a vertical line at about 5.6 MTU.

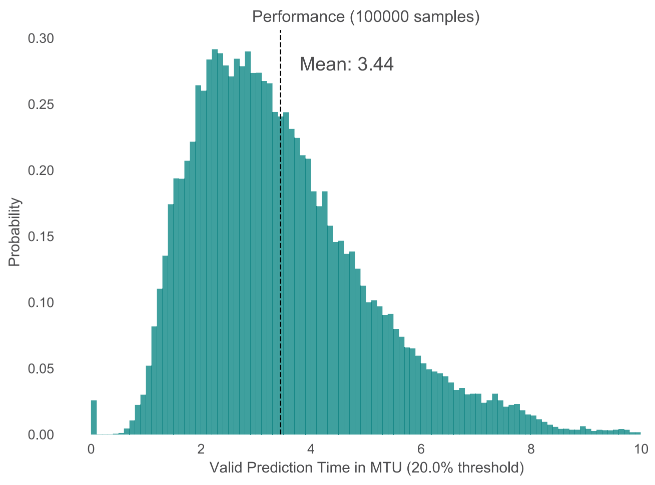

Figure A3. Histogram of valid prediction time (VPT) for RNN model 1 forecasts drawn from 100,000 initial conditions in the test data set.

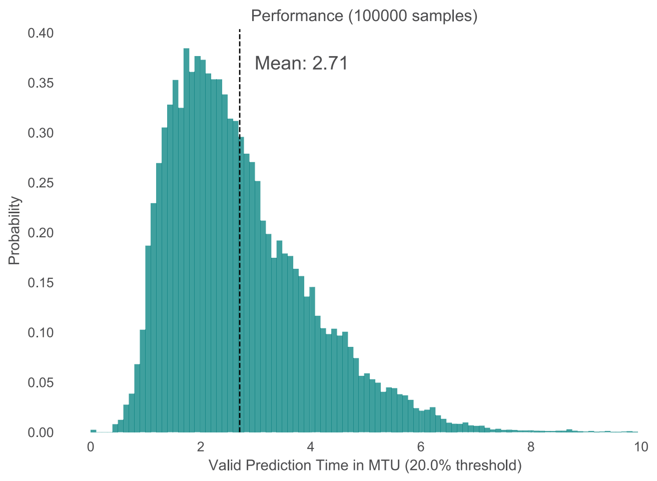

Figure A4. As in Figure A3, but using RNN model 2. Note the model is skillful, but the mean VPT is reduced compared to model 1.

We establish a "Valid Prediction Time" (VPT) as the length of time that the RMSE of any forecast starting at time t_0 remains below a threshold ϵ. The VPT is defined precisely as,

$$\sigma_i^{clim} = \sqrt{\sum_{t=t_0^{train}}^{t_N^{train}} x_i(t)}, \tag{A1}$$

$$x_{RMSE}^f(t,t_0) = \sqrt{\sum_{i=1}^{D}\left(\frac{x_i^f(t,t_0) - x_i(t)}{\sigma_i^{clim}}\right)^2}, \tag{A2}$$

$$VPT(t_0) = \max\left\{t : x_{RMSE}^f(t,t_0) < \epsilon, \forall t > t_0\right\}. \tag{A3}$$

where σ_i^{clim} is the standard deviation in time of the *ith* model variable, $x_i^f(t,t_0)$ is the forecast from t_0 to t, x_i is the true state, D is the number of model variables, and $x_{RMSE}^f(t,t_0)$ is the corresponding RMSE of the forecast

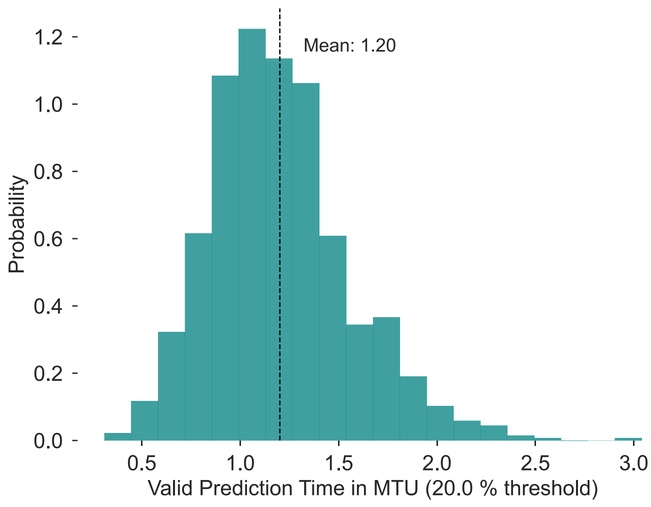

Figure A5. As in Figures A3 and A4, but using RNN Model 3. Due to the computational cost of Model 3, only 1,000 sample forecasts are used. The model is still skillful, but the mean VPT is reduced when compared to Models 1 and 2.

error at time t. An example RNN forecast using RNN model 1 is shown in Figure A2, with the VPT shown using $\epsilon = 0.2$. We reiterate that the Bayesian optimization is used to identify parameters that produce the best average forecast skill across the training data set. Data assimilation experiments are applied using a separately generated test data set. Figures A3 and A4 demonstrate the distribution of prediction skill for Models 1 and 2 initialized from 100,000 initial conditions from the test data set. The RMSE is normalized by climatological variability, defined as the standard deviation in time of each model variable calculated over the training data set.

Data Availability Statement

No new data were collected for this work. Simulated datasets can be reproduced using the steps and methods provided in the manuscript.

Acknowledgments
S.G. Penny, J.A. Platt, H.-Y. Lin, M. Goodliff, and H.D.I. Abarbanel acknowledge support from the Office of Naval Research (ONR) grants N00014-19-1-2522 and N00014-20-1-2580. S.G. Penny and T.A. Smith are supported by National Oceanographic and Atmospheric Administration (NOAA) grants NA20OAR4600277 and NA19NES4320002. T.-C. Chen is supported by the NOAA Cooperative Agreement with CIRES, NA17OAR4320101. S.G. Penny, H.-Y. Lin, and M. Goodliff are supported by NOAA grant NA18NWS4680048. S.G. Penny developed the concepts for the integration of recurrent neural networks with DA, and scaling of the RNN-based DA methods via localization. T.A. Smith implemented the localized RNN-based DA and conducted corresponding experiments. T.-C. Chen implemented routines for the DA algorithms. J.A. Platt proposed and implemented the Bayesian optimization for training the macroscale parameters of the RNN, and produced the estimates of the FTLEs. Coauthors Penny, Smith, Chen, Platt, Lin, and Goodliff all contributed to the software used to produce the results reported. We thank Brian R. Hunt and Daniel J. Gauthier for productive discussions on reservoir computing.

References

Abarbanel, H. D. I. (1996). *The analysis of observed chaotic data*. Springer-Verlag.

Abarbanel, H. D. I. (2022). *The statistical physics of data assimilation and machine learning*. Cambridge University Press.

Abarbanel, H. D. I., Brown, R., & Kennel, M. B. (1992). Local lyapunov exponents computed from observed data. *Journal of Nonlinear Science*, *2*, 343–365. https://doi.org/10.1007/bf01208929

Abarbanel, H. D. I., Kostuk, M., & Whartenby, W. (2010). Data assimilation with regularized nonlinear instabilities. *Quarterly Journal of the Royal Meteorological Society*, *136*(648), 769. https://doi.org/10.1002/qj.600

Abarbanel, H. D. I., Rozdeba, P. J., & Shirman, S. (2018). *Machine learning; deepest learning as statistical data assimilation problems*. Neural Computation.

Arcomano, T., Szunyogh, I., Pathak, J., Wikner, A., Hunt, B. R., & Ott, E. (2020). A machine learning-based global atmospheric forecast model. *Geophysical Research Letters*, *47*(9), e2020GL087776. https://doi.org/10.1029/2020gl087776

Arcucci, R., Zhu, J., Hu, S., & Guo, Y.-K. (2021). Deep data assimilation: Integrating deep learning with data assimilation. *Applied Sciences*, *11*(3), 1114. https://doi.org/10.3390/app11031114

Atiya, A., & Parlos, A. (2000). New results on recurrent network training: Unifying the algorithms and accelerating convergence. *IEEE Transactions on Neural Networks*, *11*(3), 697–709. https://doi.org/10.1109/72.846741

Bishop, C. H., Etherton, B. J., & Majumdar, S. J. (2001). Adaptive sampling with the ensemble transform kalman filter. part i: Theoretical aspects. *Monthly Weather Review*, *129*(3), 420–436. https://doi.org/10.1175/1520-0493(2001)129<0420:aswtet>2.0.co;2

Bocquet, M., Brajard, J., Carrassi, A., Bertino, L., & Bertino, L. (2020). Bayesian inference of chaotic dynamics by merging data assimilation, machine learning and expectation-maximization. *Foundations of Data Science*, *2*(1), 55–80. https://doi.org/10.3934/fods.2020004

Bocquet, M., & Carrassi, A. (2017). Four-dimensional ensemble variational data assimilation and the unstable subspace. *Tellus A: Dynamic Meteorology and Oceanography*, *69*(1), 1304504. https://doi.org/10.1080/16000870.2017.1304504

Bocquet, M., Farchi, A., & Malartic, Q. (2021). Online learning of both state and dynamics using ensemble kalman filters. *Foundations of Data Science*, *3*(3), 305–330. https://doi.org/10.3934/fods.2020015

Bocquet, M., Gurumoorthy, K. S., Apte, A., Carrassi, A., Grudzien, C., & Jones, C. K. R. T. (2017). Degenerate kalman filter error covariances and their convergence onto the unstable subspace. *SIAM/ASA Journal on Uncertainty Quantification*, *5*(1), 304–333. https://doi.org/10.1137/16M1068712

Bonavita, M., & Laloyaux, P. (2020). Machine learning for model error inference and correction. *Journal of Advances in Modeling Earth Systems*, *12*(12), e2020MS002232. https://doi.org/10.1029/2020ms002232

Bouhlel, M. A., Hwang, J. T., Bartoli, N., Lafage, R., Morlier, J., & Martins, J. R. R. A. (2019). A python surrogate modeling framework with derivatives. *Advances in Engineering Software*, *135*, 102662. https://doi.org/10.1016/j.advengsoft.2019.03.005

Boukabara, S., Krasnopolsky, V., Penny, S. G., Stewart, J. Q., McGovern, A., Hall, D., et al. (2021). Outlook for exploiting artificial intelligence in the earth and environmental sciences. *Bulletin of the American Meteorological Society*, *102*(5), E1016–E1032. https://doi.org/10.1175/bams-d-20-0031.1

Brajard, J., Carrassi, A., Bocquet, M., & Bertino, L. (2020). Combining data assimilation and machine learning to emulate a dynamical model from sparse and noisy observations: A case study with the lorenz 96 model. *Journal of Computational Science*, *44*, 101171. https://doi.org/10.1016/j.jocs.2020.101171

Buizza, R., Houtekamer, P. L., Pellerin, G., Toth, Z., Zhu, Y., & Wei, M. (2005). A comparison of the ECMWF, MSC, and NCEP global ensemble prediction systems. *Monthly Weather Review*, *133*(5), 1076–1097. https://doi.org/10.1175/MWR2905.1

Chattopadhyay, A., Hassanzadeh, P., & Subramanian, D. (2020). Data-driven predictions of a multiscale lorenz 96 chaotic system using machine-learning methods: Reservoir computing, artificial neural network, and long short-term memory network. *Nonlinear Processes in Geophysics*, *27*(3), 373–389. https://doi.org/10.5194/npg-27-373-2020

Chen, T.-C., & Kalnay, E. (2019). *Proactive quality control: Observing system simulation experiments with the lorenz'96 model* (Vol. 147, pp. 53–67). American Meteorological Society Section: Monthly Weather Review. https://doi.org/10.1175/MWR-D-18-0138.1

Courtier, P., & Talagrand, O. (1987). Variational assimilation of meteorological observations with the adjoint vorticity equation. ii: Numerical results. *Quarterly Journal of the Royal Meteorological Society*, *113*(478), 1329–1347. https://doi.org/10.1002/qj.49711347813

Courtier, P., Thépaut, J. N., & Hollingsworth, A. (1994). A strategy for operational implementation of 4D-Var, using an incremental approach. *Quarterly Journal of the Royal Meteorological Society*, *120*(519), 1367–1387. https://doi.org/10.1002/qj.49712051912

Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, *14*(2), 179–211. https://doi.org/10.1207/s15516709cog1402_1

Evensen, G. (1994). Sequential data assimilation with a nonlinear quasi-geostrophic model using Monte Carlo methods to forecast error statistics. *Journal of Geophysical Research: Oceans*, *99*(C5), 10143–10162. https://doi.org/10.1029/94JC00572

Farchi, A., Bocquet, M., Laloyaux, P., Bonavita, M., & Malartic, Q. (2021). A comparison of combined data assimilation and machine learning methods for offline and online model error correction. *Journal of Computational Science*, *55*, 101468. https://doi.org/10.1016/j.jocs.2021.101468

Geist, K., Parlitz, U., & Lauterborn, W. (1990). Comparison of different methods for computing lyapunov exponents. *Progress of Theoretical Physics*, *83*(5), 875–893. https://doi.org/10.1143/ptp.83.875

Ginsbourger, D., Le Riche, R., & Carraro, L. (2010). Kriging is well-suited to parallelize optimization. In *Computational intelligence in expensive optimization problems* (pp. 131–162). Springer. https://doi.org/10.1007/978-3-642-10701-6_6

Golub, G. H., & Van Loan, C. F. (2012). *Matrix Computations* (Vol. 3). JHU Press.

Goodliff, M., Amezcua, J., & Leeuwen, P. J. V. (2017). A weak-constraint 4densemblevar. Part ii: Experiments with larger models. *Tellus A: Dynamic Meteorology and Oceanography*, *69*(1), 1271565. https://doi.org/10.1080/16000870.2016.1271565

Greybush, S. J., Kalnay, E., Miyoshi, T., Ide, K., & Hunt, B. R. (2011). Balance and ensemble kalman filter localization techniques. *Monthly Weather Review*, *139*(2), 511–522. https://doi.org/10.1175/2010mwr3328.1

Griffith, A., Pomerance, A., & Gauthier, D. J. (2019). Forecasting chaotic systems with very low connectivity reservoir computers. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, *29*(12), 123108. https://doi.org/10.1063/1.5120710

Hatfield, S. E., Chantry, M., Dueben, P. D., Lopez, P., Geer, A. J., & Palmer, T. N., (2021). Building tangent-linear and adjoint models for data assimilation with neural networks. *Earth and Space Science Open Archive ESSOAr*. http://dx.doi.org.ezproxy.lib.utexas.edu/10.1002/essoar.10506310.1

Hindmarsh, A. C. (1983). Odepack, a systematized collection of ode solvers. *IMACS Transactions on Scientific Computation*, *1*, 55–64.

Hunt, B. R., Kostelich, E. J., & Szunyogh, I. (2007). Efficient data assimilation for spatiotemporal chaos: A local ensemble transform kalman filter. *Physica D: Nonlinear Phenomena*, *230*(1–2), 112–126. https://doi.org/10.1016/j.physd.2006.11.008

Jaeger, H. (2001). The "echo state" approach to analysing and training recurrent neural networks-with an erratum note. *German National Research Center for Information Technology GMD Technical Report*. (Vol. *148*, p. 13).

Jones, D. R., Schonlau, M., & Welch, W. J. (1998). Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, *13*(4), 455–492. https://doi.org/10.1023/a:1008306431147

Kalnay, E. (2002). *Atmospheric modeling, data assimilation and predictability*. Cambridge University Press. https://doi.org/10.1017/CBO9780511802270

Kalnay, E., Hunt, B., Ott, E., & Szunyogh, I. (2006). Ensemble forecasting and data assimilation: Two problems with the same solution. *Predictability of weather and climate*, *157*, 180.

Konkoli, Z. (2017). Reservoir computing. In R. A. Meyers (Ed.), *Encyclopedia of complexity and systems science* (pp. 1–12). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-27737-5_683-1

Lin, H.-Y., & Penny, S. G. (2021). Fourier reservoir computing for data-driven prediction of multi-scale coupled quasi-geostrophic dynamics. *Earth and Space Science Open Archive*, *19*. https://doi.org/10.1002/essoar.10509867.1

Lorenc, A. C. (2003). The potential of the ensemble Kalman filter for NWP – a comparison with 4D-Var. *Quarterly Journal of the Royal Meteorological Society*, *129*, 3183–3203.

Lorenc, A. C., & Jardak, M. (2018). A comparison of hybrid variational data assimilation methods for global nwp. *Quarterly Journal of the Royal Meteorological Society*, *144*(717), 2748–2760. https://doi.org/10.1002/qj.3401

Lorenz, E. N. (1996). Predictability—A problem partly solved. *Proceedings of a Seminar Held at ECMWF on Predictability, ECMWF Seminar Proceedings*, *1*, 1–18.

Lu, Z., Hunt, B. R., & Ott, E. (2018). Attractor reconstruction by machine learning. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, *28*(6), 061104. https://doi.org/10.1063/1.5039508

Lu, Z., Pathak, J., Hunt, B., Girvan, M., Brockett, R., & Ott, E. (2017). Reservoir observers: Model-free inference of unmeasured variables in chaotic systems. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, *27*(4), 041102. https://doi.org/10.1063/1.4979665

Lukoševičius, M. (2012). A practical guide to applying echo state networks. In G. Montavon, G. B. Orr, & K.-R. Müller (Eds.), *Neural networks: Tricks of the trade* (2nd ed., pp. 659–686). Springer. https://doi.org/10.1007/978-3-642-35289-8_36

Lukoševičius, M., & Jaeger, H. (2009). Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, *3*(3), 127–149. https://doi.org/10.1016/j.cosrev.2009.03.005

Maass, W., Natschläger, T., & Markram, H. (2002). 11Real-Time computing without stable states: A new framework for neural computation based on perturbations. *Neural Computation*, *14*(11), 2531–2560. https://doi.org/10.1162/089976602760407955

Močkus, J. (1975). On bayesian methods for seeking the extremum. In G. I. Marchuk (Ed.), *Optimization techniques ifip technical conference novosibirsk july 1-7, 1974*. Springer Berlin Heidelberg, (pp. 400–404).

Oczkowski, M., Szunyogh, I., & Patil, D. (2005). Mechanisms for the development of locally low-dimensional atmospheric dynamics. *Journal of the Atmospheric Sciences*, *62*(4), 1135–1156. https://doi.org/10.1175/jas3403.1

Palatella, L., Carrassi, A., & Trevisan, A. (2013). Lyapunov vectors and assimilation in the unstable subspace: Theory and applications. *Journal of Physics A: Mathematical and Theoretical*, *46*(25), 254020. https://doi.org/10.1088/1751-8113/46/25/254020

Pathak, J., Hunt, B., Girvan, M., Lu, Z., & Ott, E. (2018). Model-free prediction of large spatiotemporally chaotic systems from data: A reservoir computing approach. *Physical Review Letters*, *120*(2), 024102. https://doi.org/10.1103/physrevlett.120.024102

Pathak, J., Lu, Z., Hunt, B. R., Girvan, M., & Ott, E. (2017). Using machine learning to replicate chaotic attractors and calculate lyapunov exponents from data. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, *27*(12), 121102. https://doi.org/10.1063/1.5010300

Penny, S. G. (2014). The hybrid local ensemble transform kalman filter. *Monthly Weather Review*, *142*(6), 2139–2149. https://doi.org/10.1175/mwr-d-13-00131.1

Penny, S. G. (2017). Mathematical foundations of hybrid data assimilation from a synchronization perspective. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, *27*(12), 126801. https://doi.org/10.1063/1.5001819

Petzold, L. (1983). Automatic selection of methods for solving stiff and nonstiff systems of ordinary differential equations. *SIAM Journal on Scientific and Statistical Computing*, *4*(1), 136–148. https://doi.org/10.1137/0904010

Peyron, M., Fillion, A., Gürol, S., Marchais, V., Gratton, S., Boudier, P., & Goret, G. (2021). Latent space data assimilation by using deep learning. *Quarterly Journal of the Royal Meteorological Society*, *147*(740), 3759–3777. https://doi.org/10.1002/qj.4153

Platt, J. A., Wong, A., Clark, R., Penny, S. G., & Abarbanel, H. D. (2021). *Forecasting using reservoir computing: The role of generalized synchronization*. arXiv preprint arXiv:2102.08930.

Ruckstuhl, Y., Janjić, T., & Rasp, S. (2021). Training a convolutional neural network to conserve mass in data assimilation. *Nonlinear Processes in Geophysics*, *28*(1), 111–119. https://doi.org/10.5194/npg-28-111-2021

Schrauwen, B., Verstraeten, D., Campenhout, J. V., & Van Campenhout, J. (2007). An overview of reservoir computing: Theory, applications and implementations. In *Proceedings of the 15th european symposium on artificial neural networks* (pp. 471–479). https://doi.org/10.1007/978-3-540-74690-4_48

Steil, J. (2004). Backpropagation-decorrelation: Online recurrent learning with o(n) complexity. In *2004 ieee international joint conference on neural networks*. (ieee cat. no.04ch37541), 22, 843–848. https://doi.org/10.1109/IJCNN.2004.1380039

Talagrand, O., & Courtier, P. (1987). Variational assimilation of meteorological observations with the adjoint vorticity equation. i: Theory. *Quarterly Journal of the Royal Meteorological Society*, *113*(478), 1311–1328. https://doi.org/10.1002/qj.49711347812

Trevisan, A., D'Isidoro, M., & Talagrand, O. (2010). Four-dimensional variational assimilation in the unstable subspace and the optimal subspace dimension. *Quarterly Journal of the Royal Meteorological Society: A journal of the atmospheric sciences, applied meteorology and physical oceanography*, *136*(647), 487–496. https://doi.org/10.1002/qj.571

Trevisan, A., & Palatella, L. (2011). On the kalman filter error covariance collapse into the unstable subspace. *Nonlinear Processes in Geophysics*, *18*(2), 243–250. https://doi.org/10.5194/npg-18-243-2011

Van der Vorst, H. A. (1992). Bi-cgstab: A fast and smoothly converging variant of bi-cg for the solution of nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, *13*(2), 631–644. https://doi.org/10.1137/0913035

Vlachas, P. R., Pathak, J., Hunt, B. R., Sapsis, T. P., Girvan, M., Ott, E., & Koumoutsakos, P. (2020). Backpropagation algorithms and reservoir computing in recurrent neural networks for the forecasting of complex spatiotemporal dynamics. *Neural Networks*, *126*, 191–217. https://doi.org/10.1016/j.neunet.2020.02.016