DATA ASSIMILATION EMPOWERED NEURAL NETWORK PARAMETERIZATIONS FOR SUBGRID PROCESSES IN GEOPHYSICAL FLOWS

Suraj Pawar  
School of Mechanical & Aerospace Engineering, Oklahoma State University, Stillwater, Oklahoma - 74078, USA. supawar@okstate.edu

Omer San  
School of Mechanical & Aerospace Engineering, Oklahoma State University, Stillwater, Oklahoma - 74078, USA. osan@okstate.edu

ABSTRACT

In the past couple of years, there is a proliferation in the use of machine learning approaches to represent subgrid scale processes in geophysical flows with an aim to improve the forecasting capability and to accelerate numerical simulations of these flows. Despite its success for different types of flow, the online deployment of a data-driven closure model can cause instabilities and biases in modeling the overall effect of subgrid scale processes, which in turn leads to inaccurate prediction. To tackle this issue, we exploit the data assimilation technique to correct the physics-based model coupled with the neural network as a surrogate for unresolved flow dynamics in multiscale systems. In particular, we use a set of neural network architectures to learn the correlation between resolved flow variables and the parameterizations of unresolved flow dynamics and formulate a data assimilation approach to correct the hybrid model during their online deployment. We illustrate our framework in an application of the multiscale Lorenz 96 system for which the parameterization model for unresolved scales is exactly known. Our analysis, therefore, comprises a predictive dynamical core empowered by (i) a data-driven closure model for subgrid scale processes, (ii) a data assimilation approach for forecast error correction, and (iii) both data-driven closure and data assimilation procedures. We show significant improvement in the long-term prediction of the underlying chaotic dynamics with our framework compared to using only neural network parameterizations for future prediction. Moreover, we demonstrate that these data-driven parameterization models can handle the non-Gaussian statistics of subgrid scale processes, and effectively improve the accuracy of outer data assimilation workflow loops in a modular non-intrusive way.

Keywords Neural network, subgrid scale processes, data assimilation, ensemble Kalman filter, chaotic system, multiscale Lorenz 96 model

1 Introduction

Geophysical flows are characterized by the multiscale nature of flows where there is a massive difference between the largest and smallest scales, and these scales interact with each other to exchange heat, momentum, and moisture. This makes the numerical simulations of geophysical flows in which every flow feature is resolved computationally unmanageable, even though the physical laws governing these processes are well known. Therefore, the atmosphere and ocean models compute the approximate numerical solution on the computational grid that consists of $O(10^7)$ to $O(10^8)$ grids with a spacing of $O(10 \text{ km})$ to $O(100 \text{ km})$. The effect of unresolved scales is taken into account by using several parameterization schemes, which represent the dynamics of subgrid scale processes as a function of resolved dynamics [1][3]. However, the weather projection is marred by large uncertainties in the parameters of these parameterization schemes, and also due to incorrect structure of these parameterizations equations itself [4][6].

Typically, the parameters of these parameterization schemes are estimated by the model tuning process based on the observations from experimental and field measurements or the data generated from high-resolution numerical simulations [7][8]. The nonlinear and multiscale nature of geophysical flows makes this tuning procedure cumbersome and can impede accurate climate prediction [9]. A recent development in machine learning, particularly deep learning...
Along with the Earth system science, there is a surge in the application of machine learning for fluid mechanics. Readers are directed to an excellent review by Brunton et al. [18] on how ML algorithms are being used for augmenting the domain knowledge, automating tasks such as flow-control and optimization by the fluid mechanics’ community. In a recent perspective, Brenner at al. [19] discuss the strength and limitations of ML based algorithms to advance fluid mechanics. The closure problem in turbulence modeling is similar to the parameterization in climate modeling and is encountered in Reynolds-Averaged Navier-Stokes (RANS) and large eddy simulation (LES) which are widely adopted for engineering flow simulations. There have been several studies that use ML algorithms to address the turbulence closure problem [20-23]. [24] proposed a novel neural network architecture with embedded Galilean invariance for the prediction of Reynolds stress anisotropy tensor. Wang et al. [25] employed random forest as an ML algorithm to reconstruct the discrepancy RANS-modeled Reynolds stresses and evaluated its performance for fully developed turbulent flows and separated flows. Deep learning has also been utilized for LES of turbulent flows, for example, [26] demonstrated the use of deep learning for subgrid scale closure modeling of Kraichnan turbulence, [27] decaying homogeneous isotropic turbulence, [28] forced isotropic turbulence, [29] compressible isotropic turbulence, and wall-bounded turbulence [30]. The feasibility of deep learning has been investigated to produce a predictive model for turbulent fluxes, such as heat fluxes [31] and anomalous fluxes in drift-wave turbulence [32]. In a recent work, Novati at al. [33] introduced a multi-agent reinforcement learning framework as an automated discovery tool for turbulence models and applied it to forced homogeneous isotropic turbulence. Besides turbulence closure modeling, deep learning has been proved to be successful for challenging problems such as super-resolution of turbulent flows [34-36], data-driven modeling of chaotic systems [37-39], reduced order modeling of high-dimensional multiphysics systems [40-43], and developing forecast models for complex physical systems [44-47].

Despite the development of deep learning algorithms as a powerful tool to extract spatio-temporal patterns from the data, these methods are criticized for their black-box nature and are prone to produce physically inconsistent results due to their lack of generalizability [48,49]. Moreover, the increase in spatial and temporal dimensionalities raises a computational challenge in terms of the training. Hence, it is essential to integrate machine learning with physics-based modeling to address the challenge of interpretability, physical consistency, and computational burden [50]. One way to combine machine learning with physics-based modeling is by incorporating physical conservation laws into training through a regularization term added to the loss function of a neural network [51-54]. Another way is to change the structure of neural network architecture to enforce physical conservation laws as hard constraints [55,56]. The hybrid modeling in which a sub-model within the physics-based model is replaced by machine learning methods is another approach to address the limitation of pure data-driven methods. [15,50,57]. One of the issues with hybrid models is that the trained neural network often suffers from instability once they are deployed in the forward model. For example, a small change in the training dataset or the input and output vector of the neural network led to unpredictable blow-ups in the global circulation model that employs a neural network to emulate cloud resolving model [16,58]. Similarly, Brenowitz et al. [59] found that the nonphysical correlations learned by neural networks were the cause of instabilities in their online deployment within the global circulation model [60] and developed an approach to ensure stability. Wu et al. [61] highlighted the gap between a priori and a posteriori performance of data-driven Reynolds stress closure models as the RANS equations with such model can be ill-conditioned. Therefore, even though data-driven turbulence closure models predicted better closure terms, their online deployment does not lead to significant improvement in the mean velocity field prediction [22,25]. Wu et al. [61] proposed a metric to evaluate the conditioning of RANS equations in the a priori settings and showed that the implicit treatment of Reynolds stresses leads to reduced error in mean velocity prediction.

Data assimilation (DA) is a well-established discipline where observations are blended with the model to take uncertainties into account for improving the numerical prediction of the system [62,67] and can be applied to achieve accurate prediction in hybrid models that employ data-driven model as a submodel for some processes (for example subgrid scale processes). DA tools are being extensively utilized in geoscience and numerical weather forecast centers to correct background predictions based on a combination of heterogeneous measurement data coming from ground observations.
and satellite remote-sensing. These techniques have been also investigated recently for integrating experimental data into large-eddy simulations of engineering flows [68]. In a DA workflow, we merge forward model predictions with observational data. However, it has been often remarked that no-model is correct but some of them are useful. In typical DA studies and twin experiments, therefore, the subgrid scale processes have been modeled as Gaussian noise due to the lack of structural information on their mechanisms. If we would know their dynamics either structurally or functionally, for sure it would be wise to include them in the model before a DA analysis is executed. However, the subgrid scale processes in turbulent flows often cannot be accurately modeled by Gaussian noise, and ML methodologies can be adopted to get a grip on subgrid scale processes. Hence, we put forth a neural network based statistical learning approach to improve model uncertainty and incorporate this information as a data-driven closure term to the forward model. We examine how the forecast error reduces due by including ML based closure term to the underlying forward model. Indeed, the integration of DA with ML methodologies holds immense potential in various fields of physical science [69–73] and we demonstrate this through our study.

In this work, we propose a neural network closure framework in developing hybrid physics-ML models through DA for multiscale systems. In particular, we advocate the use of sequential DA techniques to tackle the closure modeling problem by incorporating real-time observations into a model equipped with neural network parameterization schemes for unresolved physics. To this end, we use real-time observations to regularize ML empowered predictive tools through ensemble Kalman filter based approach. We focus on a two-level Lorenz 96 model [74] for our numerical experiments since it generates a controllable test case for advancing turbulence parameterization theories, especially in the age of data-driven models. The Lorenz 96 is an idealized model of atmospheric circulation and is used widely to test research ideas [75–77]. Even though the dynamics of both large and small scales are known exactly for a two-level Lorenz 96 model, it is very difficult to predict it because of the strong interplay between fast and slow subsystems. Therefore, we select this multiscale model for the assessments of data-driven closures for capturing the physics of subgrid scales. Since we use an “explicit” evolution equation for the closure parameterizations, we can easily assess the data-driven models in a posteriori simulations. This often comprises a challenging task in LES computations since the low-pass filtering operation is “implicitly” applied to the governing equations. Our approach is multifaceted in at least two ways. We first show that the infusion of the DA approaches improves the forecasting quality of predictive models equipped with data-driven parameterizations. Second, we also demonstrate that the data-driven parameterizations help significantly to reduce forecast errors in DA workflows. Therefore, our modular framework can be considered as a way to incorporate real-time observations that are prevalent in today’s weather forecast station into hybrid models constituted from a physics-based model as the dynamical core of the system, and a data-driven model to describe unresolved physics.

The paper is structured as follows. In Section 2, we discuss the problem of parameterizations using a two-level Lorenz 96 model as a prototypical example. Section 3 details two types of neural network utilized in this study for learning the mapping between resolved variables and parameterizations of unresolved scales. We explain the methodology of data assimilation and the deterministic ensemble Kalman filter algorithm in Section 4. In Section 5, we discuss the findings of our numerical experiments with a two-level Lorenz 96 model. Finally, we conclude with the summary and direction for future work in Section 6.

2 Parameterizations in the Lorenz 96 model

In this section, we describe the two-level variant of the Lorenz 96 model proposed by Lorenz [74]. This model has been extensively investigated to study stochastic parameterization schemes [78–80], scale-adaptive parameterizations [81], and neural network parameterizations [58]. The two-level Lorenz 96 model can be written as

\[
\frac{dX_i}{dt} = -X_{i-1}(X_{i-2} - X_{i+1}) - X_i - \frac{hc}{b} \sum_{j=1}^{J} Y_{j,i} + F, \quad (1)
\]

\[
\frac{dY_{j,i}}{dt} = -c b Y_{j+1,i}(Y_{j+2,i} - Y_{j-1,i}) - c Y_{j,i} + \frac{hc}{b} X_i, \quad (2)
\]

where Equation 1 represents the evolution of slow, high-amplitude variables \(X_i\) \((i = 1, \ldots, n)\), and Equation 2 provides the evolution of a coupled fast, low-amplitude variable \(Y_{j,i}\) \((j = 1, \ldots, J)\). We use \(n = 36\) and \(J = 10\) in our computational experiments. We utilize \(c = 10\) and \(b = 10\), which implies that the small scales fluctuate 10 times faster than the larger scales. Also, the coupling coefficient \(h\) between two scales is equal to 1 and the forcing is set at \(F = 10\) to make both variables exhibit the chaotic behavior.

In parameterization research, small scale variables are not resolved and their effect is typically parameterized as a function of resolved large scale variables. A forecast model for the resolved variables given in Equation 1 can be
An artificial neural network is made up of several layers consisting of the predefined number of neurons. Each neuron consists of certain coefficients called weights and some bias. The weight determines how significant certain input feature is to the output. The input from the previous layer is multiplied by a weight matrix as shown below

$$S^l = W^l \chi^{l-1},$$

(5)

where $\chi^{l-1}$ is the output of the $(l-1)^{th}$ layer, $W^l$ is the matrix of weights for the $l^{th}$ layer. The summation of the above input-weight product and the bias is then passed through a node’s activation function which is usually some nonlinear function. The introduction of nonlinearity through activation function allows the neural network to learn highly complex relations between the input and output. The output of the $l^{th}$ layer can be written as

$$\chi^l = \zeta(S^l + B^l),$$

(6)

where $B^l$ is the vector of biasing parameters for the $l^{th}$ layer and $\zeta$ is the activation function. If there are L layers between the input and the output in a neural network, then the output of the neural network can be represented mathematically as follow

$$\hat{Y} = \zeta_L(W^L, B^L, \ldots, \zeta_2(W^2, B^2, \zeta_1(W^1, B^1, \chi))),$$

(7)

where $\chi$ and $\hat{Y}$ are the input and output of the ANN, respectively. There are several activation functions that provides different nonlinearity. Some of the widely used activation functions are sigmoid $\zeta(\phi) = 1/(1 + e^{-\phi})$, hyperbolic tangent (tanh) $\zeta(\phi) = (e^{\phi} - e^{-\phi})/(e^{\phi} + e^{-\phi})$, and rectified linear unit (ReLU) $\zeta(\phi) = \max[0, \phi]$.

The matrix $W$ and $B$ are determined through the minimization of the loss function (for example mean squared error between true and predicted labels). The gradient of the objective function with respect to weights and biases are calculated with the backpropagation algorithm. The optimization algorithms like the stochastic gradient descent method provide a rapid way to learn optimal weights. The training procedure for ANN can be summarized as follow:

- The input and output of the neural network are specified along with some initial weights initialization for neurons.
- The training data is run through the network to produce output $\hat{Y}$ whose true label is $Y$.
- The derivative of the objective function with each of the training weight is computed using the chain rule.
• The weights are then updated based on the learning rate and the optimization algorithm.

We continue to iterate through this procedure until convergence or the maximum number of iterations is reached. There are different ways in which the relationship between resolved and unresolved variables in multiscale systems can be learned with the ANN. The most common method is to employ point-to-point mapping, where the input features at a single grid point are utilized to learn the output labels at that point [22, 29, 83]. Another method is to include the information at neighboring grid points to determine the output label at a single point [26, 84]. We train our ANN by including information at different number of neighboring grid points and assess how does this additional information affects in learning the correlation between resolved and unresolved variables. We investigate three types of ANN models and they can be written as

\[
\text{ANN-3 : } \{X_{i-2}, X_i, X_{i+2}\} \in \mathbb{R}^3 \rightarrow \{G_i\} \in \mathbb{R}^1, \quad (8)
\]

\[
\text{ANN-5 : } \{X_{i-2}, \ldots, X_{i+2}\} \in \mathbb{R}^5 \rightarrow \{G_i\} \in \mathbb{R}^1, \quad (9)
\]

\[
\text{ANN-7 : } \{X_{i-3}, \ldots, X_{i+3}\} \in \mathbb{R}^7 \rightarrow \{G_i\} \in \mathbb{R}^1, \quad (10)
\]

where \(G_i\) is the parameterization at \(i\)th grid point and \(X_i\) is the resolved variable. For the training, we assume that the resolved variables and the parameterizations are known exactly and are computed by solving Equation [85] and Equation [26] in a coupled manner. For all ANN architectures used in this study, we apply two hidden layers with 40 neurons and ReLU activation function. The ANN is trained using an Adam optimizer for 300 iterations.

### 3.2 Convolutional neural network

The convolutional neural network (CNN) is particularly attractive when the data is in the form of two-dimensional images [85]. Here, we present the CNN architecture assuming that the input and output of the neural network have the structure of two-dimensional images. This formulation can be easily applied to one-dimensional images when the dimension in one direction is collapsed to one. The Conv layers are the fundamental building blocks of the CNN. Each Conv layer has a predefined number of filters (also called kernels) whose weights have to be learned using the backpropagation algorithm. The shape of the filter is usually smaller than the actual image and it extends through the full depth of the input volume from the previous layer.

For example, if the input to the CNN has \(256 \times 256 \times 1\) dimension where 1 is the number of input features, the kernels of the first Conv layer can have \(3 \times 3 \times 1\) shape. During the forward propagation, the filter is convolved across the width and height of the input volume to produce the two-dimensional map. The two-dimensional map is constructed by computing the dot product between the weights of the filter and the input volume at any position and then sliding it over the whole volume. Mathematically the convolution operation corresponding to one filter can be written as

\[
S_{ij}^l = \sum_{p=-\Delta_i/2}^{\Delta_i/2} \sum_{q=-\Delta_j/2}^{\Delta_j/2} \sum_{r=-\Delta_k/2}^{\Delta_k/2} W_{pqr}^l \chi_{i+p,j+q,k+r}^{l-1} + B_{pqr}, \quad (11)
\]

where \(\Delta_i, \Delta_j, \Delta_k\) are the sizes of filter in each direction, \(W_{pqr}^l\) are the entries of the filter for \(l\)th Conv layer, \(B_{pqr}\) is the biasing parameter, and \(\chi_{i+j+k}^{l-1}\) is the input from \((l-1)\)th layer. Each Conv layer will have a set of predefined filters and the two-dimensional map produced by each filter is then stacked in the depth dimension to produce a three-dimensional output volume. This output volume is passed through an activation function to produce a nonlinear map between inputs and outputs. The output of the \(l\)th layer is given by

\[
\chi_{ijk}^l = \zeta(S_{ijk}^l), \quad (12)
\]

where \(\zeta\) is the activation function. It should be noted that as we convolve the filter across the input volume, the size of the input volume shrinks in height and width dimension. Therefore, it is common practice to pad the input volume with zeros called zero-padding. The zero-padding permits us to control the shape of the output volume and is used in our neural network parameterization framework to preserve the shape so that input and output width and height are the same. The main advantage of CNN is its weight sharing property because the filter of the smaller size is shared across the whole image which is larger in size. This allows CNN to handle large data without the significant computational overhead. The CNN mapping in our work can be mathematically presented as

\[
\text{CNN : } \{X_1, \ldots, X_n\} \in \mathbb{R}^n \rightarrow \{G_1, \ldots, G_n\} \in \mathbb{R}^n, \quad (13)
\]

where \(X_i\) is the resolved variable and \(G_i\) is the parameterization. Therefore, the solution at a single time step corresponds to one training example for training the CNN. In our CNN architecture, we use only one hidden layer between the input and output. This hidden layer has 128 filters with \(7 \times 1\) shape. We apply ReLU activation function and use zero-padding to keep the input and output shape the same. The CNN is trained with an Adam optimizer for 400 iterations.
4 Data assimilation

As highlighted in many studies, neural network parameterizations suffer from instabilities and biases once the trained model is deployed in a forward solver \[16,16,59,61\]. From our numerical experiments, we observe that the forward model with only neural network parameterizations delivers accurate prediction only up to some time and after that the model starts deviating from the true trajectory. In order to address this issue and improve the long-term forecast with hybrid models, we utilize the data assimilation (DA) to incorporate noisy measurements into future state prediction. The main theme of DA is to extract the information from observational data to correct dynamical models and improve their prediction. There is a rich literature on DA \[2,6,41,66,87\] and here we discuss only sequential data assimilation problem and then outline the algorithm procedure for the deterministic ensemble Kalman filter (DEnKF).

We consider the dynamical system whose evolution can be represented as

\[ x_{k+1} = M(x_k) + w_{k+1}, \tag{14} \]

where \( x_k \in \mathbb{R}^n \) is the state of the dynamical system at discrete time \( t_k \), \( M : \mathbb{R}^n \rightarrow \mathbb{R}^n \) is the nonlinear model operator that defines the temporal evolution of the system. In this work, the dynamical system is the two-level Lorenz 96 model with neural network parameterizations whose evolution is governed by Equation \[3\]. The term \( w_{k+1} \) denotes the model noise that takes into account any type of uncertainty in the model that can be attributed to boundary conditions, imperfect models, etc. Let \( z_k \in \mathbb{R}^m \) be observations of the state vector obtained through noisy measurements procedure as given below

\[ z_k = h(x_k) + v_k, \tag{15} \]

where \( h(\cdot) \) is a nonlinear function that maps \( \mathbb{R}^n \rightarrow \mathbb{R}^m \), and \( v_k \in \mathbb{R}^m \) is the measurement noise. We assume that the measurement noise is a white Gaussian noise with zero mean and the covariance matrix \( R_k \), i.e., \( v_k \sim \mathcal{N}(0,R_k) \). Additionally, the noise vectors \( w_k \) and \( v_k \) are assumed to be uncorrelated at two different time steps. The sequential data assimilation can be considered as a problem of estimating the state \( x_k \) of the system given the observations up to time \( t_k \), i.e., \( z_1, \ldots, z_k \). When we utilize observations to estimate the state of the system, we say that the data are assimilated into the model. We will use the notation \( \tilde{x}_k \) to denote an analyzed state of the system at time \( t_k \) when all of the observations up to and including time \( t_k \) are used in determining the state of the system. When all the observations before (but not including) time \( t_k \) are utilized for estimating the state of the system, then we call it the forecast estimate and denote it as \( x_{k-f} \).

We use the DEnKF algorithm proposed by Sakov et al. \[88\] for the data assimilation and its procedure is summarized in Algorithm \[1\]. We start the DEnKF algorithm by initializing the state estimate for all ensemble members using Equation \[16\]. The anomalies between the forecast estimate of all ensembles and its sample mean is computed utilizing Equation \[19\]. Once the observations are available at time \( t_k+1 \), the forecast state estimate is assimilated as given in Equation \[20\] where the Kalman gain \( K \) is computed using its square root version. The anomalies for all ensemble members are updated separately with half the Kalman gain as shown in Equation \[23\]. The analyzed state estimate for all ensemble members are obtained by offsetting the analyzed anomalies with the analyzed state estimate and is calculated with Equation \[24\]. We adopt the twin experiment setting \[59\] to test the DA algorithm for a two-level Lorenz 96 model with neural network parameterizations. Also, we validate our implementation of the DEnKF algorithm using the one-level Lorenz 96 model and is discussed in detail in Appendix A.

5 Numerical Experiments

In this section, we discuss the results of numerical experiments with a two-level variant of the Lorenz 96 system embedded with neural network parameterizations for the unresolved variables. We utilize the fourth-order Runge-Kutta numerical scheme with a time step \( \Delta t = 0.001 \) for temporal integration of the Lorenz 96 model. We apply the periodic boundary condition for the slow variables, i.e., \( X_{i-1} = X_{i+n} = X_i \). The fast variables are extended by letting \( Y_{j,i-1} = Y_{j,i+n} = Y_{j,i} \), and \( Y_{j+i-1} = Y_{j+i+1} \). The physical initial condition is computed by starting with an equilibrium condition at time \( t = -5 \) for slow variables. The equilibrium condition for slow variables is \( X_i = F \) for \( i = 1, 2, \ldots, n \). We perturb the equilibrium solution for the 18th state variable as \( X_{18} = F + 0.01 \). At the time \( t = -5 \), the fast variables are assigned with random numbers between \(-F/10 \) to \( F/10 \). We integrate a two-level Lorenz 96 model by solving both Equation \[4\] and Equation \[2\] in a coupled manner up to time \( t = 0 \). With this initial condition (i.e., \( t = 0 \)), we generate the training data for neural networks by integrating the two-level Lorenz 96 model from \( t = 0 \) to \( t = 10 \). Therefore, we gather 10,000 temporal snapshots to generate the training data. For all our numerical experiments, we use 80% of the data to train the neural network and 20% data to validate the training. We assess the performance of a trained neural network by deploying it in a forecast model for temporal integration between time \( t = 10 \) and \( t = 20 \). Therefore, there is no overlap between the data used for training and testing. Since the neural
Algorithm 1 Deterministic ensemble Kalman filter

1: Initialize the state of the system for different ensemble members.

\[ \hat{X}_0(i) = m_0 + y_0(i), \]  
\[ \text{where } y_0(i) \sim N(0, P_0). \]

2: For \( k = 0, 1, \ldots \) proceed with the forecast and data assimilation step as follow

- **Forecast step:**
  - Integrate the state estimate all ensemble members from time \( t_k \) to \( t_{k+1} \) as follow
  \[ X^f_{k+1}(i) = M(\hat{X}_k(i)) \]  
  - Compute the sample mean, ensemble anomalies, and error covariance as follow
  \[ x^f_{k+1} = \frac{1}{N} \sum_{i=1}^{N} X^f_{k+1}(i), \]
  \[ A^f_{k+1}(i) = X^f_{k+1}(i) - x^f_{k+1}, \]

- **Data assimilation step:**
  - Once the observations are available at time \( t_{k+1} \), forecast state estimate is assimilated with the observation as follow
  \[ \hat{x}_{k+1} = x^f_{k+1} + K[z_{k+1} - h(x^f_{k+1})]. \]
  Here, the Kalman gain is given as
  \[ K = \frac{A^f(HA^f)^T}{N-1} \left[ \frac{(HA^f)(HA^f)^T}{N-1} + R \right]^{-1}, \]
  \[ \text{where } H \in \mathbb{R}^{m \times n} \text{ is the Jacobian of the observation operator (i.e., } H_{kl} = \frac{\partial h_k}{\partial x_l}, \text{ and a size of } \mathbb{R}^{n \times N} \text{ matrix is concatenated as follows} \]
  \[ A^f = [A^f_{k+1}(1), A^f_{k+1}(2), \ldots, A^f_{k+1}(N)]. \]
  - Compute the analyzed anomalies as below
  \[ \hat{A}_{k+1}(i) = A^f_{k+1}(i) - \frac{1}{2}KHA^f_{k+1}(i). \]
  - Calculate the analyzed ensemble using the analyzed state estimate and analyzed anomalies as follow
  \[ \hat{X}_{k+1}(i) = \hat{A}_{k+1}(i) + \hat{x}_{k+1}. \]
network has not seen the testing data during the training, the performance of neural network parameterizations in this temporal region will give us an insight on its generalizability to unseen data.

First, we present results for ANN based parameterizations trained using neighboring stencil mapping as discussed in Section 3.1. Figure 1 displays the full state trajectory of the Lorenz 96 model from time $t = 10$ to $t = 20$ computed by solving both the evolution of slow and fast variables (i.e., True) and with ANN based parameterizations for fast variables (i.e., ANN-3, ANN-5, ANN-7). The difference between the true solution field and the predicted solution field is also depicted in Figure 1. It can be observed that the predicted solution field starts deviating from the true solution field at around $t \approx 12$ for all ANN-based parameterizations.

Next, we illustrate how the prediction of a two-level Lorenz 96 model with neural network parameterizations can be improved using data assimilation by incorporating noisy observations in the future state prediction. For our twin experiment, we obtain observations by adding noise drawn from the Gaussian distribution with zero mean and the covariance matrix $R_k$, i.e., $v_k \sim \mathcal{N}(0, R_k)$. We use $R_k = \sigma^2 I$, where $\sigma$ is the standard deviation of measurement noise and is set at $\sigma = 1$. We assume that observations are sparse in space and are collected at every 10th time step. We present two levels of observation density in space for the DA. For the first case, we employ observations at $[X_4, X_8, \ldots, X_{36}] \in \mathbb{R}^9$ for the assimilation. The second set of observations consists of 50% of the full state of the system, i.e., $[X_2, X_4, \ldots, X_{36}] \in \mathbb{R}^{18}$. In Figure 2 we provide the full state trajectory prediction for the ANN-5 parameterization without any DA and with DA for two sets of observations. We can observe that there is a substantial improvement in the long-term prediction even with only 25% of the observations incorporated through the DEnKF algorithm. The results in Figure 2 provide the evidence for the good performance of the present framework in achieving accurate long-term prediction for hybrid models embedded with data-driven parameterizations. Therefore, the present framework can lead to accurate forecasting by exploiting online measurements coming from various types of sensor.

---

Figure 1: Full state trajectory of the multiscale Lorenz 96 model with the closure term computed using the different neighboring stencil mapping feedforward ANN architecture.

---
networks and can find applications in different fields like climate modeling, turbulence closure modeling where the subgrid scale parameterizations are unavoidable.

Figure 2: Full state trajectory of the multiscale Lorenz 96 model with the closure term computed using the five-point neighboring stencil mapping feedforward ANN architecture and the DEnKF used for data assimilation.

Figure 2 illustrates the time evolution of the full state trajectory of a two-level Lorenz 96 model with CNN based parameterizations for unresolved scales. CNN is fed with the entire state of the slow variables as an input and it calculates the parameterizations of fast variables at all grid points. From Figure 2, we can deduce that the predicted state trajectory starts deviating from the true state at around $t \approx 12$ when only CNN based parameterizations are employed in the forward model of slow variables. When we incorporate observations through DA, we observe considerable improvement in the state prediction over a longer period.

Based on results presented in Figure 2 and Figure 3, we can notice that the error is slightly higher between time $t = 18$ to $t = 20$ for the CNN based parameterizations empowered with DA. One potential reason for this discrepancy can be the stochastic nature of the parameterization model. The true parameterization model in itself is stochastic and might not follow a Gaussian distribution. Another reason for the inaccurate forecast can be attributed to the uncertainty in the prediction of parameterizations by CNN. To isolate the source of error, we integrate the forecast model for a two-level Lorenz 96 model without any parameterizations. The results for this numerical experiment are discussed in Appendix B. In this numerical experiment, the observations include the effect of unresolved scales and can be considered as an added noise. The sequential DA methods based on Kalman filters deliver a considerably accurate solution when the model and observations noise is drawn from a Gaussian distribution and enough observations are provided. If the parameterization of unresolved scales follows a Gaussian distribution, we should be able to recover the accurate state of the system as the density of observations is increased. However, as reported in Figure 5, there is a high level of inaccuracy even when 100% of the state is observable. Therefore, we can conclude that there is a considerable benefit of including neural network parameterizations compared to using no parameterization in the forecast model. The results provided in Figure 2 and Figure 3 also shows that the neural network parameterizations can capture the non-Gaussian statistics of...
subgrid scale processes and this leads to accurate forecasting over a longer period. There are other DA approaches that deal with non-Gaussian distributions for noise vectors \([90–95]\). We restrict ourselves to the DEnKF algorithm for DA in this study and plan to explore other DA algorithms in our future work.

We assess the quantitative performance of different numerical experiments performed in this study using the root mean squared error (RMSE) between the true and predicted state of slow variables in a two-level Lorenz 96 model. The RMSE is computed as shown below

$$\text{RMSE} = \sqrt{\frac{1}{n_t} \sum_{i=1}^{n_t} \left( X^T_{i}(t_k) - X^P_{i}(t_k) \right)^2}, \quad (25)$$

where \(X^T\) is the true state of the system and \(X^P\) is the predicted state of the system. Table\([\text{I}]\) reports the RMSE for a two-level Lorenz 96 model for all cases investigated in this work. We can see that the RMSE is very high when we do not use any parameterizations for unresolved scales even when measurements for an entire state of the system are incorporated through DA. The data assimilation alone can not account for the effect of unresolved scales, even though their effect is present in the observations data. Therefore, it is imperative to include parameterizations of fast variables in the forecast model of slow variables. We observe that the ANN architecture provides slightly more accurate results than the CNN based parameterizations for fast variables. Also, the RMSE is minimum for the ANN-3 parameterizations and we observe a slight increase in RMSE by including more neighboring information. One potential reason for this observation can be the use of the same hyperparameters for all ANN architectures. However, this change is very small and the RMSE is the same order of magnitude for all types of neural network parameterizations. The RMSE is almost the same when 25% or 50% of the full state of the system is observed in data assimilation framework.
Table 1: Quantitative assessment of different neural network parameterizations for subgrid scale processes using the total root mean square error given by Equation (25).

| Framework | RMSE  |
|-----------|-------|
| Only neural network parameterizations |       |
| ANN-3     | 3.38  |
| ANN-5     | 3.73  |
| ANN-7     | 3.77  |
| CNN       | 3.79  |
| Only data assimilation |       |
| No parameterizations ($m = 9$) | 5.11  |
| No parameterizations ($m = 18$) | 4.30  |
| No parameterizations ($m = 36$) | 3.92  |
| Neural network parameterizations with data assimilation |       |
| ANN-5 ($m = 9$) | 0.52  |
| ANN-5 ($m = 18$) | 0.53  |
| CNN ($m = 9$) | 2.13  |
| CNN ($m = 18$) | 2.20  |

6 Concluding Remarks

In the present study, we introduce a framework to apply data assimilation methods to the physics-based model embedded with data-driven parameterizations to achieve accurate long-term forecast in multiscale systems. We demonstrate that the forecasting capability of hybrid models can be significantly improved by exploiting online measurements from various types of sensor networks. Specifically, we use neural networks to learn the relation between resolved scales and the effect of unresolved scales (i.e., parameterizations). The deployment of the trained neural network in the forward simulation provides accurate prediction up to a short period and then there is a large discrepancy between true and predicted state of the system. To address this issue, we exploit the sparse observations data through data assimilation to improve the accuracy of the forecasting over a longer period. We illustrate this framework for a two-scale variant of the Lorenz 96 model which consists of fast and slow variables whose dynamics are exactly known. We obtain a considerable improvement in the prediction by combining neural network parameterizations and data assimilation compared to employing only neural network parameterizations. We also found that including an ML based closure term seems to capture non-Gaussian statistics and significantly improve the forecast error. Based on our numerical experiments with data assimilation empowered neural network parameterizations, we can conclude that improving machine learning-based model prediction with data assimilation methods offers a promising research direction.

Our future work aims at leveraging the underlying physical conservation laws into neural network training to produce physically consistent parameterizations. As the deep learning field is evolving rapidly, we can integrate modern neural network architectures and training methodology into our framework to attain higher accuracy. In the present framework, we employ the deterministic ensemble Kalman filter (DEnKF) algorithm for data assimilation in the present study. This algorithm gives accurate prediction when the uncertainty in model and observations follows a Gaussian distribution. We plan to investigate other data assimilation approaches like maximum likelihood ensemble filter methods that can handle the non-Gaussian nature of uncertainty in the mathematical model to get further improvement in the accuracy prediction. We will also test the present framework for more complex turbulent flows as a part of our future effort. Finally, we conclude by reemphasizing that the integration of data assimilation with hybrid physics-ML models can be effectively used for modeling of multiscale systems.

Acknowledgement

This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research under Award Number [de-sc0019290]. O.S. gratefully acknowledges their support.

Disclaimer. This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any
A  Validation of the Deterministic Ensemble-Kalman Filter

In this Appendix, we provide results of data assimilation with the DEnKF algorithm for one level Lorenz 96 model. The one level Lorenz 96 model is given as

\[ \frac{dX_i}{dt} = -X_{i-1}(X_{i-2} - X_{i+1}) - X_i + F, \]  

(26)

for \( i \in \{1, 2, \ldots, 36\} \) and \( F = 10 \). The above model is completely deterministic as there is no parameterization of the unresolved scales. We use the similar settings as the two-level variant of the Lorenz 96 model for temporal integration using the fourth-order Runge-Kutta numerical scheme. The true initial condition is generated by integrating the solution starting from an equilibrium condition from \( t = -5 \) to \( t = 0 \). For all ensemble members, we start with an initial condition obtained by perturbing the true initial condition with a noise drawn from the Gaussian distribution with zero mean and the variance of \( 1 \times 10^{-2} \). The observations are generated for data assimilation by adding a measurement noise from the Gaussian distribution with zero mean and the variance of \( \sigma^2 = 1 \) (i.e., \( R_k = I \)) to the true state of the system. The observations are assumed to be available at every 10th time step, similar to the two-level variant of the Lorenz 96 model.

As depicted in Figure 4, we can conclude that the DEnKF can correct the erroneous trajectory even when only 9 observations are employed for data assimilation. As the amount of observations is increased to 18, we observe a reduction in the error. We reiterate here that, we have complete control over the model (since it is deterministic) in the numerical experiments with a one-level Lorenz 96 model. As we introduce fast scale variables, the evolution of slow variables in a two-level Lorenz 96 model is no longer deterministic and simple Kalman filter based algorithms might not be enough to give accurate prediction over a longer period.

B  Data-assimilation with no parameterization

In this Appendix, we report the performance of the DEnKF algorithm for the data assimilation of a two-level variant of the Lorenz 96 model with no parameterizations employed for unresolved scales. The two-level Lorenz 96 model with no parameterizations reduces to one-level Lorenz 96 model as presented in Equation (26). We note here that the observations used for data assimilation are the same as the numerical experiments with a two-level Lorenz 96 model. Therefore, the effect of unresolved scales is embedded in observations. The parameterization of fast variables (i.e., \( \frac{h_c b}{\tau} \sum_{j=1}^{J} Y_{j,i} \) term in Equation 1) can be considered as an added noise to the true state of the system for a one-level Lorenz 96 model presented in Equation (26).

In Figure 5, we report the true state of a two-level Lorenz 96 model and also the predicted state trajectory using the DA framework with no parameterization. We provide the results for three sets of observations utilized in DA. The observations are incorporated at every 10th time step of the model through assimilation stage. We can observe that, even when 100% of the full state is observable, we do not recover the true state trajectory of a two-level Lorenz 96 model. With this observation, we can conclude that it is essential to incorporate parameterization of unresolved scales into a forward model of the DA procedure to recover the accurate state trajectory. The root mean squared error between the assimilated states and true states for three sets of observations is provided in Table 1.

References

[1] David J Stensrud. Parameterization schemes: keys to understanding numerical weather prediction models. Cambridge University Press, 2009.

[2] Jinqiao Duan and Balasubramanya Nadiga. Stochastic parameterization for large eddy simulation of geophysical flows. Proceedings of the American Mathematical Society, 135(4):1187–1196, 2007.

[3] David A Randall. Cloud parameterization for climate modeling: Status and prospects. Atmospheric research, 23(3-4):345–361, 1989.
Figure 4: Full state trajectory of the Lorenz 96 model with the DEnKF algorithm.

[4] Tapio Schneider, Shiwei Lan, Andrew Stuart, and Joao Teixeira. Earth system modeling 2.0: A blueprint for models that learn from observations and targeted high-resolution simulations. *Geophysical Research Letters*, 44(24):12–396, 2017.

[5] David Draper. Assessment and propagation of model uncertainty. *Journal of the Royal Statistical Society: Series B (Methodological)*, 57(1):45–70, 1995.

[6] Christopher E Holloway and J David Neelin. Moisture vertical structure, column water vapor, and tropical deep convection. *Journal of the atmospheric sciences*, 66(6):1665–1683, 2009.

[7] Christian Jakob. An improved strategy for the evaluation of cloud parameterizations in GCMs. *Bulletin of the American Meteorological Society*, 84(10):1387–1402, 2003.

[8] Christian Jakob. Accelerating progress in global atmospheric model development through improved parameterizations: Challenges, opportunities, and strategies. *Bulletin of the American Meteorological Society*, 91(7):869–876, 2010.

[9] Ming Zhao, J-C Golaz, Isaac M Held, Venkatachalam Ramaswamy, S-J Lin, Y Ming, P Ginoux, B Wyman, LJ Donner, D Paynter, et al. Uncertainty in model climate sensitivity traced to representations of cumulus precipitation microphysics. *Journal of Climate*, 29(2):543–560, 2016.

[10] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[11] Philipp Neumann, Peter Düben, Panagiotis Adamidis, Peter Bauer, Matthias Brück, Luis Kornblueh, Daniel Klocke, Bjorn Stevens, Nils Wedi, and Joachim Biercamp. Assessing the scales in numerical weather and climate predictions: will exascale be the rescue? *Philosophical Transactions of the Royal Society A*, 377(2142):20180148, 2019.
Figure 5: Full state trajectory of the multiscale Lorenz 96 model with no closure for subgrid processes. The observation data for the DEnKF algorithm is obtained by adding measurement noise to the exact solution of the multiscale Lorenz 96 system.

[12] Claudia Kuenzer, Marco Ottinger, Martin Wegmann, Huadong Guo, Changlin Wang, Jianzhong Zhang, Stefan Dech, and Martin Wikelski. Earth observation satellite sensors for biodiversity monitoring: potentials and bottlenecks. *International Journal of Remote Sensing*, 35(18):6599–6647, 2014.

[13] Yunjie Liu, Evan Racah, Joaquin Correa, Amir Khosrowshahi, David Lavers, Kenneth Kunkel, Michael Wehner, William Collins, et al. Application of deep convolutional neural networks for detecting extreme weather in climate datasets. *arXiv preprint arXiv:1605.01156*, 2016.

[14] Xingjian Shi, Zhihan Gao, Leonard Lausen, Hao Wang, Dit-Yan Yeung, Wai-kin Wong, and Wang-chun Woo. Deep learning for precipitation nowcasting: A benchmark and a new model. In *Advances in neural information processing systems*, pages 5617–5627, 2017.

[15] Emmanuel de Bezenac, Arthur Pajot, and Patrick Gallinari. Deep learning for physical processes: Incorporating prior scientific knowledge. *Journal of Statistical Mechanics: Theory and Experiment*, 2019(12):124009, 2019.

[16] Stephan Rasp, Michael S Pritchard, and Pierre Gentine. Deep learning to represent subgrid processes in climate models. *Proceedings of the National Academy of Sciences*, 115(39):9684–9689, 2018.

[17] Pierre Gentine, Mike Pritchard, Stephan Rasp, Gael Reinaudi, and Galen Yacalis. Could machine learning break the convection parameterization deadlock? *Geophysical Research Letters*, 45(11):5742–5751, 2018.

[18] Steven L Brunton, Bernd R Noack, and Petros Koumoutsakos. Machine learning for fluid mechanics. *Annual Review of Fluid Mechanics*, 52, 2019.

[19] MP Brenner, JD Eldredge, and JB Freund. Perspective on machine learning for advancing fluid mechanics. *Physical Review Fluids*, 4(10):100501, 2019.
[20] Karthik Duraisamy, Gianluca Iaccarino, and Heng Xiao. Turbulence modeling in the age of data. *Annual Review of Fluid Mechanics*, 51:357–377, 2019.
[21] F Sarghini, G De Felice, and S Santini. Neural networks based subgrid scale modeling in large eddy simulations. *Computers & Fluids*, 32(1):97–108, 2003.
[22] Masataka Gamahara and Yuji Hattori. Searching for turbulence models by artificial neural network. *Physical Review Fluids*, 2(5):054604, 2017.
[23] Romit Maulik, Himanshu Sharma, Saumil Patel, Bethany Lusch, and Elise Jennings. Accelerating RANS turbulence modeling using potential flow and machine learning. *arXiv preprint arXiv:1910.10878*, 2019.
[24] Julia Ling, Andrew Kurzawski, and Jeremy Templeton. Reynolds averaged turbulence modelling using deep neural networks with embedded invariance. *Journal of Fluid Mechanics*, 807:155–166, 2016.
[25] Jian-Xun Wang, Jin-Long Wu, and Heng Xiao. Physics-informed machine learning approach for reconstructing Reynolds stress modeling discrepancies based on dns data. *Physical Review Fluids*, 2(3):034603, 2017.
[26] Romit Maulik, Omer San, Adil Rasheed, and Prakash Vedula. Subgrid modelling for two-dimensional turbulence using neural networks. *Journal of Fluid Mechanics*, 858:122–144, 2019.
[27] Andrea Beck, David Flad, and Claus-Dieter Munz. Deep neural networks for data-driven LES closure models. *Journal of Computational Physics*, page 108910, 2019.
[28] Chenyue Xie, Jianchun Wang, and E Weinan. Modeling subgrid-scale forces by spatial artificial neural networks in large eddy simulation of turbulence. *Physical Review Fluids*, 5(5):054606, 2020.
[29] Chenyue Xie, Jianchun Wang, Ke Li, and Chao Ma. Artificial neural network approach to large-eddy simulation of compressible isotropic turbulence. *Physical Review E*, 99(5):053113, 2019.
[30] PA Srinivasan, L Guastoni, Hossein Azizpour, PHILIPP Schlatter, and Ricardo Vinuesa. Predictions of turbulent shear flows using deep neural networks. *Physical Review Fluids*, 4(5), 2019.
[31] Junhyuk Kim and Changhoon Lee. Prediction of turbulent heat transfer using convolutional neural networks. *Journal of Fluid Mechanics*, 882:A18, 2020.
[32] RA Heinonen and PH Diamond. Turbulence model reduction by deep learning. *Physical Review E*, 101(6):061201, 2020.
[33] Guido Novati, Hugues Lascombes de Laroussilhe, and Petros Koumoutsakos. Automating turbulence modeling by multi-agent reinforcement learning. *arXiv preprint arXiv:2005.09023*, 2020.
[34] Kai Fukami, Koji Fukagata, and Kunihiko Taira. Super-resolution reconstruction of turbulent flows with machine learning. *Journal of Fluid Mechanics*, 870:106–120, 2019.
[35] Chiyu Max Jiang, Soheil Esmaeilzadeh, Kamyar Azizzadenesheli, Karthik Kashinath, Mustafa Mustafa, Hamdi A Tchelepi, Philip Marcus, Anima Anandkumar, et al. Meshfreeflownet: A physics-constrained deep continuous space-time super-resolution framework. *arXiv preprint arXiv:2005.01463*, 2020.
[36] Akshay Subramaniam, Man Long Wong, Raunak D Borker, Sravya Nimmagadda, and Sanjiva K Lele. Turbulence enrichment using physics-informed generative adversarial networks. *arXiv*, pages arXiv–2003, 2020.
[37] Jaideep Pathak, Brian Hunt, Michelle Girvan, Zhixin Lu, and Edward Ott. Model-free prediction of large spatiotemporally chaotic systems from data: A reservoir computing approach. *Physical Review Letters*, 120(2):024102, 2018.
[38] PR Vlachas, J Pathak, BR Hunt, TP Sapsis, M Girvan, E Ott, and P Koumoutsakos. Backpropagation algorithms and reservoir computing in recurrent neural networks for the forecasting of complex spatiotemporal dynamics. *Neural Networks*, 2020.
[39] Zhong Yi Wan, Pantelis Vlachas, Petros Koumoutsakos, and Themistoklis Sapsis. Data-assisted reduced-order modeling of extreme events in complex dynamical systems. *PloS one*, 13(5), 2018.
[40] Kookjin Lee and Kevin Carlberg. Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders. *Journal of Computational Physics*, 404:108973, 2020.
[41] Arvind Mohan, Don Daniel, Michael Chertkov, and Daniel Livescu. Compressed convolutional lstm: An efficient deep learning framework to model high fidelity 3d turbulence. *arXiv preprint arXiv:1903.00033*, 2019.
[42] Elizabeth Qian, Boris Kramer, Benjamin Pechnerstorfer, and Karen Willcox. Lift & learn: Physics-informed machine learning for large-scale nonlinear dynamical systems. *Physica D: Nonlinear Phenomena*, 406:132401, 2020.
[43] Sk Mashfiquar Rahman, Suraj Pawar, Omer San, Adil Rasheed, and Traian Iliescu. Nonintrusive reduced order modeling framework for quasigeostrophic turbulence. *Physical Review E*, 100:053306, 2019.
[44] Romit Maulik, Romain Egele, Bethany Lusch, and Prasanna Balaprakash. Recurrent neural network architecture search for geophysical emulation. arXiv preprint arXiv:2004.10928, 2020.
[45] Rui Wang, Karthik Kashinath, Mustafa Mustafa, Adrian Albert, and Rose Yu. Towards physics-informed deep learning for turbulent flow prediction. arXiv preprint arXiv:1911.08655, 2019.
[46] Istvan Szunyogh, Troy Arcomano, Jaideep Pathak, Alexander Wikner, Brian Hunt, and Edward Ott. A machine-learning-based global atmospheric forecast model. 2020.
[47] M Cheng, F Fang, C C Pain, and I M Navon. Data-driven modelling of nonlinear spatio-temporal fluid flows using a deep convolutional generative adversarial network. Computer Methods in Applied Mechanics and Engineering, 365:113000, 2020.
[48] James H Faghmous, Arindam Banerjee, Shashi Shekhar, Michael Steinbach, Vinod Kumar, Auroop R Ganguly, and Nagiza Samatova. Theory-guided data science for climate change. Computer, 47(11):74–78, 2014.
[49] Nicholas Wagner and James M Rondinelli. Theory-guided machine learning in materials science. Frontiers in Materials, 3:28, 2016.
[50] Markus Reichstein, Gustau Camps-Valls, Bjorn Stevens, Martin Jung, Joachim Denzler, Nuno Carvalhais, et al. Deep learning and process understanding for data-driven Earth system science. Nature, 566(7743):195–204, 2019.
[51] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. Journal of Computational Physics, 378:686–707, 2019.
[52] Maziar Raissi and George Em Karniadakis. Hidden physics models: Machine learning of nonlinear partial differential equations. Journal of Computational Physics, 357:125–141, 2018.
[53] Jin-Long Wu, Karthik Kashinath, Adrian Albert, Dragos Chirila, Heng Xiao, et al. Enforcing statistical constraints in generative adversarial networks for modeling chaotic dynamical systems. Journal of Computational Physics, 406:109209, 2020.
[54] N Benjamin Erichson, Lionel Mathelin, Zhewei Yao, Steven L Brunton, Michael W Mahoney, and J Nathan Kutz. Shallow neural networks for fluid flow modeling with limited sensors. Proceedings of the Royal Society A, 476(2238):20200097, 2020.
[55] Arvind T Mohan, Nicholas Lubbers, Daniel Livescu, and Michael Chertkov. Embedding hard physical constraints in neural network coarse-graining of 3D turbulence. arXiv preprint arXiv:2002.00021, 2020.
[56] Pablo Márquez-Neilia, Mathieu Salzmann, and Pascal Fua. Imposing hard constraints on deep networks: Promises and limitations. arXiv preprint arXiv:1706.02025, 2017.
[57] Anuj Karpatne, Gowtham Atluri, James H Faghmous, Michael Steinbach, Arindam Banerjee, Auroop Ganguly, Shashi Shekhar, Nagiza Samatova, and Vinod Kumar. Theory-guided data science: A new paradigm for scientific discovery from data. IEEE Transactions on Knowledge and Data Engineering, 29(10):2318–2331, 2017.
[58] S. Rasp. Coupled online learning as a way to tackle instabilities and biases in neural network parameterizations: general algorithms and Lorenz 96 case study (v1.0). Geoscientific Model Development, 13(5):2185–2196, 2020.
[59] Noah D Brenowitz and Christopher S Bretherton. Spatially extended tests of a neural network unified physics parameterization. Geophysical Research Letters, 45(12):6289–6298, 2018.
[60] Jinlong Wu, Heng Xiao, Rui Sun, and Qiqi Wang. Reynolds-averaged Navier–Stokes equations with explicit data-driven Reynolds stress closure can be ill-conditioned. Journal of Fluid Mechanics, 869:553–586, 2019.
[61] John M Lewis, Sivaramakrishnan Lakshmivarahan, and Sudarshan Dhall. Dynamic data assimilation: a least squares approach, volume 104. Cambridge University Press, Cambridge, 2006.
[62] Dan Simon. Optimal state estimation: Kalman, H infinity, and nonlinear approaches. John Wiley & Sons, 2006.
[63] Geir Evensen. Data assimilation: the ensemble Kalman filter. Springer Science & Business Media, 2009.
[64] D Xiao, J Du, F Fang, CC Pain, and J Li. Parameterised non-intrusive reduced order methods for ensemble Kalman filter data assimilation. Computers & Fluids, 177:69–77, 2018.
[65] Camille Zerfas, Leo G Rebholz, Michael Schneier, and Traian Iliescu. Continuous data assimilation reduced order models of fluid flow. Computer Methods in Applied Mechanics and Engineering, 357:112596, 2019.
[66] Rossella Arcucci, Laetitia Mottet, Christopher Pain, and Yi-Ke Guo. Optimal reduced space for variational data assimilation. Journal of Computational Physics, 379:51–69, 2019.
[68] Jeffrey W Labahn, Hao Wu, Shaun R Harris, Bruno Coriton, Jonathan H Frank, and Matthias Ihme. Ensemble Kalman filter for assimilating experimental data into large-eddy simulations of turbulent flows. *Flow, Turbulence and Combustion*, 104:861—-893, 2020.

[69] Xin Li, Feng Liu, and Miao Fang. Harmonizing models and observations: data assimilation for earth system science. *Sci China Earth Sci.*, 2020.

[70] Redouane Lguensat, Pierre Tandeo, Pierre Ailliot, Manuel Pulido, and Ronan Fablet. The analog data assimilation. *Monthly Weather Review*, 145(10):4093–4107, 2017.

[71] Meng Tang, Yimin Liu, and Louis J Durlofsky. A deep-learning-based surrogate model for data assimilation in dynamic subsurface flow problems. *arXiv preprint arXiv:1908.05823*, 2019.

[72] Marc Bocquet, Julien Brajard, Alberto Carrassi, and Laurent Bertino. Bayesian inference of chaotic dynamics by merging data assimilation, machine learning and expectation-maximization. *Foundations of Data Science*, 2(1):55, 2020.

[73] Julien Brajard, Alberto Carassi, Marc Bocquet, and Laurent Bertino. Combining data assimilation and machine learning to emulate a dynamical model from sparse and noisy observations: a case study with the Lorenz 96 model. *arXiv preprint arXiv:2001.01520*, 2020.

[74] Edward N Lorenz. Predictability: A problem partly solved. In *Proc. Seminar on Predictability*, volume 1, 1996.

[75] KJH Law, D Sanz-Alonso, Abhishek Shukla, and AM Stuart. Filter accuracy for the Lorenz 96 model: Fixed versus adaptive observation operators. *Physica D: Nonlinear Phenomena*, 325:1–13, 2016.

[76] Alireza Karimi and Mark R Paul. Extensive chaos in the Lorenz-96 model. *Chaos: An interdisciplinary journal of nonlinear science*, 20(4):043105, 2010.

[77] S Herrera, Diego Paz ó, J Ferná Ndez, and Miguel A Rodríguez. The role of large-scale spatial patterns in the chaotic amplification of perturbations in a Lorenz’96 model. *Tellus A: Dynamic Meteorology and Oceanography*, 63(5):978–990, 2011.

[78] Tim N Palmer. A nonlinear dynamical perspective on model error: A proposal for non-local stochastic-dynamic parametrization in weather and climate prediction models. *Quarterly Journal of the Royal Meteorological Society*, 127(572):279–304, 2001.

[79] Daniel S Wilks. Effects of stochastic parametrizations in the Lorenz’96 system. *Quarterly Journal of the Royal Meteorological Society: A journal of the atmospheric sciences, applied meteorology and physical oceanography*, 131(606):389–407, 2005.

[80] Daan Crommelin and Eric Vanden-Eijnden. Subgrid-scale parameterization with conditional markov chains. *Journal of the Atmospheric Sciences*, 65(8):2661–2675, 2008.

[81] Gabriele Vissio and Valerio Lucarini. A proof of concept for scale-adaptive parametrizations: the case of the lorenz’96 model. *Quarterly Journal of the Royal Meteorological Society*, 144(710):63–75, 2018.

[82] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[83] XIA Yang, S Zafar, J-X Wang, and H Xiao. Predictive large-eddy-simulation wall modeling via physics-informed neural networks. *Physical Review Fluids*, 4(3):034602, 2019.

[84] San O Rasheed A Pawar, S and P Vedula. A priori analysis on deep learning of subgrid-scale parameterizations for kraichnan turbulence. *Theoretical and Computational Fluid Dynamics*, pages 387–401, 2020.

[85] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[86] Arthur Gelb. *Applied optimal estimation*. MIT press, 1974.

[87] Greg Welch and Gary Bishop. An introduction to the Kalman filter. 1995.

[88] Pavel Sakov and Peter R Oke. A deterministic formulation of the ensemble Kalman filter: an alternative to ensemble square root filters. *Tellus A: Dynamic Meteorology and Oceanography*, 60(2):361–371, 2008.

[89] Henry Abarbanel. *Predicting the future: completing models of observed complex systems*. Springer, 2013.

[90] Weixuan Li, W Steven Rosenthal, and Guang Lin. Trimmed ensemble Kalman filter for nonlinear and non-gaussian data assimilation problems. *arXiv preprint arXiv:1808.05465*, 2018.

[91] Jeffrey L Anderson. A non-Gaussian ensemble filter update for data assimilation. *Monthly Weather Review*, 138(11):4186–4198, 2010.
[92] Amit Apte, Martin Hairer, A M Stuart, and Jochen Voss. Sampling the posterior: An approach to non-Gaussian data assimilation. *Physica D: Nonlinear Phenomena*, 230(1-2):50–64, 2007.

[93] Milija Zupanski. Maximum likelihood ensemble filter: Theoretical aspects. *Monthly Weather Review*, 133(6):1710–1726, 2005.

[94] Alberto Carrassi, Stephane Vannitsem, Dusanka Zupanski, and Milija Zupanski. The maximum likelihood ensemble filter performances in chaotic systems. *Tellus A: Dynamic Meteorology and Oceanography*, 61(5):587–600, 2008.

[95] Elias David Nino-Ruiz, Alfonso Mancilla-Herrera, Santiago Lopez-Restrepo, and Olga Quintero-Montoya. A maximum likelihood ensemble filter via a modified Cholesky decomposition for non-Gaussian data assimilation. *Sensors*, 20(3):877, 2020.