PHYSICS-GUIDED MACHINE LEARNING FOR TURBULENCE CLOSURE AND REDUCED-ORDER MODELING

By

SURAJ PAWAR

Bachelor of Technology in Mechanical Engineering
Veermata Jijabai Technological Institute
Mumbai, India
2014

Master of Science in Mechanical Engineering
Virginia Polytechnic Institute & State University
Blacksburg, VA, USA
2018

Submitted to the Faculty of the
Graduate College of
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
Doctor of Philosophy
July, 2022

PHYSICS-GUIDED MACHINE LEARNING FOR TURBULENCE CLOSURE AND REDUCED-ORDER MODELING

Dissertation Approved:

Omer San, PhD

Dissertation Advisor

Arvind Santhanakrishnan, PhD

Kursat Kara, PhD

JaEun Ku, PhD

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my advisor Dr. San for the continuous support during my Ph.D. studies and research, for his motivation, patience, enthusiasm, and immense knowledge. Special thanks to him for letting me work on different research projects and for always taking the time to explain complex and difficult concepts in the simplest form. I could not have imagined having a better advisor and mentor for my Ph.D. studies. I would like to thank the rest of my committee members: Dr. Santhanakrishnan, Dr. Kara, and Dr. Ku, for their insightful comments, encouragement, and interesting questions. Further, I would like to express great appreciation to all my lecturers for imparting a great deal of knowledge without which my academic journey would not have been successful.

This work received support from the U.S. Department of Energy, Office of Science, Office of Advanced Scientifc Computing Research under Award Number DE-SC0019290 The author gratefully acknowledges the financial support received from them. I am also grateful for the assistance given by our collaborators: Dr. Rasheed, Dr. Vedula, Dr. Kvamsdal, and Dr. Nair. I would like to offer my special thanks to Romit Maulik and Ashesh Sharma for giving me an opportunity to do summer internships with them.

Special thanks to my friends and colleagues in the lab: Shady, Harsha, Mashfiq, Mansoor, Abdullah, Saadbin, Saeed, Mehrdad, Pedram, Furkan, Rohit, and Mobashera for the stimulating discussions, and for all the fun we have had in the last few years.

Acknowledgments reflect the views of the author and are not endorsed by committee members or Oklahoma State University.

In particular, I would like to thank Shady for sharing his insights on many topics in computational science and it has been a great learning experience to work with you.

I am grateful to my dear parents: Arun Pawar and Shobha Pawar for their unconditional love and support. Without their sacrifice, this would have never been possible. Especially, my father who passed away around one year back, made me an independent and determined person. Finally, I would like to thank my brother, Rajan Pawar for his constant encouragement throughout my graduate studies. I owe every bit of my achievement to my family.

Acknowledgments reflect the views of the author and are not endorsed by committee members or Oklahoma State University.

Dedicated to my grandfather, my father, and my mother

Acknowledgments reflect the views of the author and are not endorsed by committee members or Oklahoma State University.

Name: SURAJ PAWAR

Title of Study: PHYSICS-GUIDED MACHINE LEARNING FOR TURBULENCE CLOSURE AND REDUCED-ORDER MODELING

Major Field: MECHANICAL & AEROSPACE ENGINEERING

Abstract: A recent advance in scientific machine learning has started to show promising results in fluid mechanics. Despite their early success, the application of data-driven methods to turbulent flow simulation is non-trivial due to underlying highly nonlinear multiscale interactions. Here we present novel physics-guided machine learning (PGML) approaches for turbulence closure model discovery and model order reduction of complex multiscale systems. Our turbulence closure model discovery approach is based on exploiting big data without relying on underlying turbulence physics and learning from physical constraints. Specifically, we propose a frame invariant neural network model that can incorporate physical symmetries as inductive biases and illustrates its stable performance in the coarse-grid simulation without any kind of post-processing of the predicted subgrid-scale closure model. The frame invariant SGS model guarantees desired physical constraints without the need for any regularization terms and ultimately generalizes to different initial conditions and Reynolds numbers. To achieve data-efficient training and improved generalization, we propose a concatenated neural network with an uncertainty quantification mechanism that leverages information from hierarchies of models. The concatenated neural network is based on embedding information from cheap to evaluate low-fidelity approximations into the certain hidden layer of the neural network both during training and deployment. This framework is demonstrated for a range of problems, including turbulent boundary layer reconstruction, and reduced-order modeling of the vortex merging process. Furthermore, we investigate the seamless integration of sparse and noisy observations into non-intrusive reduced-order models, and hybrid models where the dynamical core of the system is modeled using the known governing equations, and the subgrid-scale processes are modeled using a deep learning model. To summarize, this work builds a bridge between extensive physics-based theories and data-driven modeling paradigms and paves the way for using hybrid physics-informed learning algorithms to generate predictive technologies for turbulent fluid flows.

TABLE OF CONTENTS

Chapter Page

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

ABBREVIATIONS

The following abbreviations are used in this manuscript:

CFD	Computational Fluid Dynamce
DNS	Direct Numerical Simulation
LES	Large eddy simulation
RANS	Reynolds-averaged Navier-Stokes equations
SGS	Sub Grid Scale
DSM	Dynamic Smagorinsky model
UNS	Unresolved Numerical Simulation
FDNS	Filtered Direct Numerical Simulation
ML	Machine Learning
DL	Deep Learning
PGML	Physics-Guided Machine Learning
ANN	Artificial Neural Networks
DNN	Deep Neural Networks
CNN	Convolutional Neural Networks
LSTM	Long Short-Term Memory
FI-CNN	Frame Invariant Convolutional Neural Networks
DA	Data Assimilation
EnKF	Ensemble Kalman filter
DEnKF	Deterministic Ensemble Kalman filter
ROM	Reduced Order Model

NIROM	Non-Intrusive Reduced Order Model
POD	Proper Orthogonal Decomposition
GP	Galerkin Projection
RMSE	Root Mean Squared Error
PDF	Probability Density Function
SD	Standard Deviation
SST	Sea Surface Temperature

CHAPTER I

Introduction

1.1 Motivation and Background

Machine learning (ML) methods have entered the renaissance due to groundbreaking findings in natural language and image processing. This can be attributed to several factors, including vast and increasing volumes of data; advances in computational hardware; reduced costs for computation, data storage, and transfer; sophisticated data-driven algorithms; an abundance of open source libraries and benchmark datasets; and significant investment by industry on data-driven problem-solving. In recent years, data-driven modeling has started to play important role in advancing many scientific disciplines traditionally dominated by physics-based models [166, 141]. The fluid mechanics field which has traditionally dealt with massive amounts of data gathered from experiments, field observations, and high-resolution simulation is also seeing a surge in the application of ML algorithms to solve a variety of problems [50, 46].

The application of ML models is particularly promising for problems, where the complete mechanistic description of certain processes is not well understood, or where it is computationally intractable to run physics-based models at desired resolution in space and time. These situations are very common in fluid mechanics. For example, closure models deployed in many applications encountered in combustion, climate, and wind energy are based on isotropic homogeneous turbulence assumption which is not quite applicable for nonstandard stratified, inhomogeneous, and massively separated flows. The second challenge is related to online tasks like uncertainty quantification (UQ) which is very important for model calibration, designing effective risk mitigation strategies, and configuring optimal sensor networks that require running a forward model multiple times. While ML models are a potential choice for these problems, there are several challenges in applying pure data-driven methods to scientific discovery problems due to various reasons, such as limited availability of data, poor generalization under distribution shift, interpretability of black-box methods, and violation of known physics, such as symmetries, constraints, and conserved quantities.

The physics-based approaches, on the other hand, are interpretable, trustworthy, and highly generalizable implying that they can be applied to a much wider array of problems (in the context of fluid mechanics, atmospheric flows and flow around heat exchangers) provided that they are governed by similar physics. However, they can be computationally very demanding. Therefore, there is a need to build hybrid models that can take advantage of data-driven methods without violating the domain knowledge built over decades [318, 164, 167]. This dissertation will focus on developing different physics-guided machine learning approaches for closure model discovery and model order reduction problems encountered in turbulent fluid flows.

1.2 Turbulence Closure Model Discovery

Turbulent flows are prevalent in nature and are characterized by a wide range of length and time scales. Most of the transient flows involve the development of the boundary layer, flow separation due to adverse pressure gradient, and turbulence transition. The multiscale nature of the flow is even more pronounced in geophysical flows where there is a massive difference between the largest and smallest eddies, and these eddies interact with each other to exchange heat, momentum, and water. While direct numerical simulation (DNS) of the spatio-temporal evolution of turbulent flows can resolve the full spectra of turbulence down to the Kolmogorov scale, it is computationally intractable even with today's supercomputers. One often has to resort to lower-order models such as large eddy simulation (LES) or Raynolds-Averaged Navier-Stokes (RANS) simulation to model turbulent flows. The nonlinearity of Navier-Stokes equations introduces unclosed terms (also called subgrid-scale (SGS) closure terms in LES) to account for interactions between resolved and unresolved scales. In LES, the large scales are explicitly resolved and small scales are parameterized. If the interactions between large and small scales are not accurately modeled, then an increase in grid resolution will not necessarily improve the accuracy of the large scales [177]. Many closure models have been proposed with varying levels of accuracy and computational requirements [244, 330, 84, 361].

Some of the most commonly used parameterization models are based on phenomenological arguments and involve tuning model parameters to match the experimental findings. The physical processes in geophysical and engineering applications are tightly coupled where the accuracy and computational performance of closure models play a critical role. It is also well known that the parameterizations are a major source

of uncertainties in turbulent flow simulation and have been a topic of research for many decades [101, 425]. Therefore, despite the plethora of turbulence closure models, the development of a universal closure model to account for nonlinear and nonlocal interactions of small scales on larger scales remains a grand challenge. Furthermore, the development of LES models specific to geophysical flows is relatively scarce compared to many LES models developed for engineering flows. This motivates the need for research on developing novel turbulence closure models that do not assume any heuristics of turbulence physics but learn from big data in combination with physical constraints.

1.3 Model Order Reduction

There has been tremendous progress in simulating multiphysics problems using partial differential equation (PDE) based discretization methods and these methods have been scaled for large problems (for example turbulent flow simulation, atmospheric flows, wind energy) with high-performance computing infrastructure. Despite this progress, their use in online tasks like optimization, optimal control, uncertainty quantification, and data assimilation for high-dimensional parameterized systems is computationally intractable. Many modeling paradigms attempts to address this challenge by constructing a reduced-order representation of the system that captures the essential features at significantly reduced computational cost [326, 369, 302]. One of the main requirements of physics-based reduced-order models (ROMs) is that they require complete information about the underlying governing equations. However, for many problems in engineering and geophysical applications, the exact structure of governing equations is not known or is insufficient for the desired purpose and this limits their capability in practical applications.

Recently, ML has emerged as a promising tool in building non-intrusive surrogate models that do not require any information about the underlying dynamics and solely learn from the data. The adjoints for these surrogates models can be efficiently computed using automatic differentiation and this makes them attractive for outer loop applications, such as design optimization and variational data assimilation. However, the black-box nature of ML is currently preventing its full potential from being utilized in engineering and geophysical applications. Hence, there is a prevailing trend towards *hybrid analysis and modeling* (HAM) which explores the continuum between physics-based and ML methods to design models which are generalizable, trustworthy,

computationally efficient, and dynamically evolving in time. Furthermore, data-driven surrogate models should be capable of capturing the uncertainty associated with the prediction to safeguard against the dangers of extrapolation, and therefore, new algorithms and frameworks are warranted to address these challenges.

1.4 Organization

The dissertation is organized as follows. In **Chapter II**, we investigate different data-driven frameworks for learning SGS stresses in the a priori settings and determine a stabilization rule for a posteriori deployment of data-driven closures into a coarse-grid simulation. Specifically, we study how the choice of a learning map affects the accuracy of the data-driven SGS model and compare the performance of two neural network architectures for LES of Kraichnan turbulence.

Chapter III presents a data assimilation framework to incorporate sparse and noisy observations in the hybrid model built using the governing equations for dynamical core and data-driven model for subgrid-scale processes. We illustrate our framework for the multiscale Lorenz 96 system for which the parameterization model for unresolved scales is exactly known, and the two-dimensional Kraichnan turbulence system for which the parameterization model for unresolved scales is not known a priori. 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.

In **Chapter IV**, we propose a novel frame invariant neural network framework for SGS closure modeling that yields the stable and most accurate solution without using any stabilization rule. We design a tailored neural network architecture that incorporates physical symmetries (for example, translation, Galilean, and rotation) as hard constraints and satisfy them to the precision of discretization error. This leads to a physically consistent neural network-based SGS model that has the most accurate prediction in *a priori* analysis. Furthermore, the frame invariant model is stable in *a posteriori* deployment without any kind of post-processing or addition of artificial dissipation. We show that the frame invariant model is generalizable across different initial conditions and Reynolds numbers and our numerical investigation demonstrates

the best agreement for the frame invariant model with the filtered DNS solution for several statistical metrics.

Chapter V puts forth a concatenated neural network approach to build tailored, effective, and efficient machine learning models that can leverage the information from simplified physics-based models. In particular, we combine the self-similarity solution and power-law velocity profile (low-fidelity models) with the noisy data obtained either from experiments or computational fluid dynamics simulations (high-fidelity models) through a concatenated neural network. We illustrate how the knowledge from these simplified models results in reducing uncertainties associated with deep learning models and improved generalization for boundary layer flow prediction problems.

Chapter VI extends the application of concatenated neural network framework to projection-based reduced-order models. We apply this framework as a novel model fusion approach combining the physics-based Galerkin projection model and long-short term memory (LSTM) network for parametric model order reduction of fluid flows. We show that the framework is capable of enhancing the generalizability of data-driven models, and effectively protecting against or informing about the inaccurate predictions resulting from extrapolation.

Chapter VII describes the integration of equation-free surrogate model within sequential data assimilation framework and its application to forecasting weekly average sea surface temperature. The surrogate model uses proper orthogonal decomposition (POD) to identify the dominant structures of the flow, and a long short-term memory network to model the dynamics of the POD modes. The surrogate model is then integrated within the deterministic ensemble Kalman filter (DEnKF) to incorporate sparse and noisy observations at optimal sensor locations obtained through QR pivoting. It is demonstrated that the prediction accuracy of the NIROM gets improved by almost one order of magnitude by the DEnKF algorithm.

Finally, in **Chapter VIII**, the concluding remarks regarding the use of physics-guided machine learning to turbulence and opportunities to address some of the open questions are provided.

CHAPTER II

Deep Learning Frameworks for Subgrid Scale Closure Models

The contents of this chapter has been published in Theoretical Computational Fluid Dynamics (TCFD)[1].

Abstract: In this chapter, we investigate different data-driven parameterizations for large eddy simulation of two-dimensional turbulence in the *a priori* settings. These models utilize resolved flow field variables on the coarser grid to estimate the subgrid-scale stresses. We use data-driven closure models based on localized learning that employs a multilayer feedforward artificial neural network with point-to-point mapping and neighboring stencil data mapping, and convolutional neural network fed by data snapshots of the whole domain. The performance of these data-driven closure models is measured through a probability density function and is compared with the dynamic Smagorinksy model (DSM). The quantitative performance is evaluated using the cross-correlation coefficient between the true and predicted stresses. We analyze different frameworks in terms of the amount of training data, selection of input and output features, their characteristics in modeling with accuracy, and training and deployment computational time. We also demonstrate computational gain that can be achieved using the intelligent eddy viscosity model that learns eddy viscosity computed by the DSM instead of subgrid-scale stresses. We detail the hyperparameters optimization of these models using the grid search algorithm. After numerical analysis in the *a priori* settings, we deploy data-driven SGS models in coarse-grid simulations and design a post-processing step to achieve stable large eddy simulation.

[1]Pawar, S., San, O., Rasheed, A., & Vedula, P. (2020). A priori analysis on deep learning of subgrid-scale parameterizations for Kraichnan turbulence. Theoretical and Computational Fluid Dynamics, 34(4), 429-455.

2.1 Introduction

Direct numerical simulation (DNS) of complex fluid flows encountered in many engineering and geophysical applications are computationally unmanageable because of the need to resolve a wide range of spatiotemporal scales. Large eddy simulation (LES) and Reynolds Averaged Navier-Stokes (RANS) modeling are two most commonly used mathematical modeling frameworks that give accurate predictions by considering the interaction between the unresolved and grid-resolved scales. The development of these models is termed as the turbulence closure problem and has been a long-standing challenge in the fluid mechanics community [88, 195, 244, 243].

In LES, we filter the Navier-Stokes equations using a low-pass filtering operator that separates the motion into small and large scales, and in turn, produces modified equations, which are computationally faster to solve than actual Navier-Stokes equations [22, 324, 96]. The interaction between grid-resolved and unresolved scales is then taken into account by introducing subgrid-scale stress (SGS) term in the modified equation. The main task of the SGS model is to provide mean dissipation that corresponds to the transfer of energy from resolved scales to unresolved scales (the production of energy at large scales is balanced by the dissipation of energy at small scales based on Kolmogorov's theory of turbulence). The dissipation effect of unresolved scales can be included utilizing an eddy viscosity parameterization obtained through grid-resolved quantities. Providing such dissipation mechanism often results in increasing the numerical stability of the under-resolved discretization. These eddy viscosity approaches are called as functional models, which assume isotropy of small scales to present the average dissipation of the unresolved scales [107]. The most widely used functional model is the Smagorinsky model [350] that uses a global constant called the Smagorinsky coefficient to produce mean dissipation of energy. It is observed in many studies that a single value of the Smagorinsky coefficient cannot be used for a variety of flow phenomenon [78, 241, 228, 295]. The deficiencies of static Smagorinsky model can be overcome by using dynamic Smagorisnky model (DSM) proposed by [118]. [212] introduced the modification in Germano's DSM by which the stress-strain relationship is optimized with a least-squares approach (we discuss Lilly's version of DSM in detail in Section 2.2.1). Several other versions of Germano's DSM have been proposed, such as localized version to overcome mathematical inconsistencies in standard DSM [120], Lagrangian version of DSM [245], and DSM with a corrector step [280]. Even the dynamic procedure is not free from parameter tuning and one

has to specify the test filter and grid-filter width ratio to accurately model the SGS stresses. Hence, there is a constant effort to develop a subgrid-scale model that is free from heuristics and can predict the SGS stresses accurately.

In the past decade, the unprecedented amount of data collected from experiments, high-fidelity simulations has facilitated using machine learning (ML) algorithms in fluid mechanics [50, 46]. ML algorithms are now used for flow control, flow optimization, reduced order modeling, flow reconstruction, super-resolution, and flow cleansing [188, 50]. One of the first applications of deep learning in fluid mechanics was by [246] who implemented neural network methodology to reconstruct near-wall turbulence and showed an improvement in prediction capability of velocity fields. Subsequently, several ML algorithms such as shallow decoder for flow reconstruction [93], a convolutional neural network (CNN) for super-resolution of turbulent flows [108], deep convolutional autoencoder for nonlinear model order reduction [200, 257] have been proposed. Several studies have been conducted to model the dynamics of chaotic fluid flows using ML algorithms [329, 220, 308, 281, 382, 310, 285]. Recently there is a growing interest in using the physical knowledge in combination with the data-driven algorithms [309, 95, 225, 213, 406, 238, 281, 250]. The physics can be incorporated into these learning algorithms by adding a regularization term (based on governing equations) in loss function or modifying the neural network architecture to enforce certain physical constraints.

In addition to reduced order modeling and chaotic dynamical systems, the turbulence closure problem has also benefited from the application of ML algorithms and has led to reducing uncertainties in RANS and LES models [87, 194, 174, 393, 368]. Different machine learning algorithms like kernel regression, single hidden layer neural network, random forest [373, 374, 214] have been proposed for turbulence closure modeling. [340] proposed the hybrid approach in which the neural network is used for learning Bardina's scale similar subgrid-scale model for turbulent channel flow. Their neural network architecture employed 15 input features consisting of velocity gradients and Reynolds tensor components (made up of fluctuating component of velocity), and turbulent viscosity as the learned variable. The motivation behind this approach was to improve computational performance rather than to learn the true turbulent dynamics. [213] presented a novel neural network architecture that utilizes a multiplicative layer with an invariant tensor to embed Galilean invariance for the prediction of Reynolds stress anisotropy tensor. Their tensor basis neural network (TBNN) uses five invariants of strain-rate tensor and rotation-rate tensor at a point

in the input layer. In addition to the input layer, the TBNN has tensor input layers that take a tensor basis [296] (tensor basis includes 10 isotropic basis tensors). They demonstrated the superiority of applying constrained neural network over generic neural network architecture in predicting Reynolds stress anisotropy tensor for various complex flow problems such as duct flow, and wavy channel flow. [238] introduced data-driven turbulence closure framework for subgrid-scale modeling and performed *a priori* and *a posteori* analysis for two-dimensional Kraichnan turbulence. Their neural network architecture employs vorticity, streamfunction, and eddy-viscosity kernel information at nine surrounding grid points to learn the turbulence source term at the central point. They found that the inclusion of eddy viscosity kernels leads to accurate prediction of the turbulence source term. [112] tested an artificial neural network for finding a new subgrid-scale model in LES of channel flow using the pointwise correlation between grid resolved variables and subgrid stresses. They investigated the effect of different input variables to the neural network and observed that including velocity gradients and vertical distance gives the most accurate prediction for SGS stresses. [388] developed a data-driven framework to learn discrepancies in Reynolds stress models as a function of mean flow features using random forest regression algorithm. They evaluated the performance of the proposed framework in terms of different training and testing parameters for flow characteristics and different geometries. [37] built an approximation model using encoder-decoder CNN architecture to determine the aerodynamic flow field around airfoils using the angle of attack, Reynolds number, and airfoil shape as the input variables. [28] developed a data-driven approach based on recurrent convolutional neural network for learning the LES closure term for decaying homogeneous isotropic turbulence problem and presented a methodology to construct stable models that can be used in CFD codes. Their architecture includes snapshots of primitive variables and the coarse grid LES operator as input features and unknown subgrid terms in labels. [354] evaluated the capability of multilayer perceptron and long short-term memory network in predicting the turbulent statistics for shear flow. In the recent work, [276] illustrated the two to eight times computational gain that can be attained with a data-driven model that utilizes deep neural network to learn eddy viscosity obtained from the dynamic Smagorinsky model.

The motivation behind the present work is to address the following questions: which data-driven algorithms are suitable for particular applications, which input features have a significant influence on learning subgrid stresses, which algorithm

has better predictive capability, which algorithm is faster, and how much data to use for different ML algorithms for efficient learning? In addition to addressing these questions, we also study the effect of data locality where the information at neighbouring points is found to give improved prediction than point-to-point mapping. The work presented here is concurrent with many of the ideas presented in above studies [238, 28, 233, 213, 108, 276], and our main objective is to investigate the performance of different approaches for subgrid-scale modeling in LES of turbulence.

To achieve these objectives, we examine the performance of data-driven closure models for two-dimensional Kraichnan turbulence [180]. Even though the two-dimensional turbulence cannot be realized in practice or experiments but only in numerical simulations, it represents many geophysical flows and provides a starting point in modeling these flows. It finds application in modeling many atmospheric and ocean flows [182, 204, 39]. A reduction in dimensionality compared to three-dimensional turbulence leads to inverse energy cascade, i.e., the transfer of energy from small scales to large scales and direct enstrophy (spatial average of the square of the vorticity) cascade from large scales to small scales [181, 24, 204]. Therefore, with the presence of complex flow interactions and simplicity of two-dimensional analysis, Kraichnan turbulence will serve as a good testbed for our data-driven closure model analysis. Our approaches are based on three models that employ velocity field, velocity gradients, and the Laplacian of the velocity. These variables are available in any CFD solver and the SGS stresses can be learned in several ways such as point-to-point mapping, neighboring stencil mapping, and learning from the whole field or snapshot. In this work, we demonstrate these different approaches and analyze them in the context of the predictive performance, amount of training data, and computational overhead for training and testing, as well as their data structures.

In Section 2.2, we introduce the turbulence closure problem and the dynamic Smagorinsky model. Section 2.3 will present different frameworks investigated in this study. In Section 2.4, we detail the data generation using DNS and will evaluate data-driven turbulence closure models in terms of predictive performance, computational overhead, and data requirement for training. We demonstrate an additional modeling approach using intelligent eddy viscosity model in Section 2.5 that is computationally faster than the DSM. Finally, we will present the conclusions and future work in Section 2.7. We also describe the hyperparameters selection procedure in Appendix 2.B to obtain optimal neural network architecture.

2.2 Turbulence Closure

We begin with the introduction of the turbulence closure framework by outlining governing equations in its primitive variables form used to model incompressible fluid flows. The spatial and temporal evolution of the fluid flow are governed by the Navier-Stokes equations that describe the conservation of mass and momentum:

$$\frac{\partial u_i}{\partial x_i} = 0, \tag{2.1}$$

$$\frac{\partial u_i}{\partial t} + \frac{\partial u_i u_j}{\partial x_j} = -\frac{1}{\rho}\frac{\partial p}{\partial x_i} + \nu\frac{\partial}{\partial x_j}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right), \tag{2.2}$$

where u_i is the i^{th} component of velocity, p is the pressure, ρ is the density, and ν is the kinematic viscosity of fluid. The governing equations for LES (also called as the filtered Navier-Stokes equations) are obtained by applying a low-pass filter operation and it results in a grid-filtered system of equations:

$$\frac{\partial \bar{u}_i}{\partial x_i} = 0, \tag{2.3}$$

$$\frac{\partial \bar{u}_i}{\partial t} + \frac{\partial \overline{u_i u_j}}{\partial x_j} = -\frac{1}{\rho}\frac{\partial \bar{p}}{\partial x_i} + \nu\frac{\partial}{\partial x_j}\left(\frac{\partial \bar{u}_i}{\partial x_j} + \frac{\partial \bar{u}_j}{\partial x_i}\right), \tag{2.4}$$

where the overbar quantities represent the filtered variables. The filtered Navier-Stokes equations have the nonlinear term $\overline{u_i u_j}$ which is unknown due to truncation of small eddies by spatial filtering operation. The decomposition of nonlinear term [205] can be given as

$$\overline{u_i u_j} = \tau_{ij} + \bar{u}_i \bar{u}_j, \tag{2.5}$$

where $\tau_{ij} = \overline{u_i u_j} - \bar{u}_i \bar{u}_j$ is the subgrid-scale stress that consists of cross-stress tensor (which represents interaction between large and small scales), Reynolds subgrid tensor (which represents interaction between subgrid scales), and Leonard tensor (which represents the interactions among large scales). Using this decomposition, the filtered Navier-Stokes equations can be written as

$$\frac{\partial \bar{u}_i}{\partial t} + \frac{\partial \bar{u}_i \bar{u}_j}{\partial x_j} = -\frac{1}{\rho}\frac{\partial \bar{p}}{\partial x_i} + \nu\frac{\partial}{\partial x_j}\left(\frac{\partial \bar{u}_i}{\partial x_j} + \frac{\partial \bar{u}_j}{\partial x_i}\right) - \frac{\partial \tau_{ij}}{\partial x_j}. \tag{2.6}$$

The main challenge in subgrid-scale modeling is to approximate this τ_{ij} term and the approximated model should provide sufficient dissipation corresponding to the transfer of energy from large eddies to unresolved eddies. The static Smagorinsky model [350]

which uses an effective eddy viscosity to model SGS stresses is given by

$$\tau_{ij}^{M,d} = -2(C_s\Delta)^2|\bar{S}|\bar{S}_{ij}, \tag{2.7}$$

where the superscript M stands for the model, d means the deviatoric (traceless) part of the tensor, Δ is the grid-filter width, and C_s is the static Smagorinsky coefficient. A derivation of Smagorinsky model for two-dimensional case is provided in Appendix 2.A. The terms $|\bar{S}|$ and $\bar{S}_{ij}$ in the above equation are calculated as

$$\bar{S}_{ij} = \frac{\partial \bar{u}_i}{\partial x_j} + \frac{\partial \bar{u}_j}{\partial x_i}, \quad |\bar{S}| = \sqrt{2\bar{S}_{ij}\bar{S}_{ij}}. \tag{2.8}$$

It should be noted that the static Smagorinsky model might be excessive or under dissipative with suboptimal values of C_s. It was found in many studies that the Smagorinsky coefficient is different for different flows and additional modifications are needed in the near-wall region [78, 241, 228, 295]. To tackle these problems, the dynamic Smagorinksy model [118, 212] was introduced that allowed the C_s to be computed dynamically based on the flow, time, resolution, and spatial location. The dynamic Smagorinsky model is discussed in detail in Section 2.2.1.

2.2.1 Dynamic Smagorinsky Model

[118] introduced the dynamic procedure that calculates the Smagorinsky coefficient based on the local flow structure dynamically instead of assuming a constant value. The dynamic procedure consists of applying a secondary spatial filter called as the test filter to the grid-filtered Navier-Stokes equations. The test filtered equations can be written as

$$\frac{\partial \hat{\bar{u}}_i}{\partial t} + \frac{\partial \hat{\bar{u}}_i \hat{\bar{u}}_j}{\partial x_j} = -\frac{1}{\rho}\frac{\partial \hat{\bar{p}}}{\partial x_i} + \nu\frac{\partial}{\partial x_j}\left(\frac{\partial \hat{\bar{u}}_i}{\partial x_j} + \frac{\partial \hat{\bar{u}}_j}{\partial x_i}\right) - \frac{\partial \mathcal{T}_{ij}}{\partial x_j}, \tag{2.9}$$

where the caret over the overbar represents the test filtered variables. The test filtered subgrid stress $\mathcal{T}_{ij}$ (also called as subtest-scale stress) is given by

$$\mathcal{T}_{ij} = \widehat{\overline{u_i u_j}} - \hat{\bar{u}}_i \hat{\bar{u}}_j. \tag{2.10}$$

Similar to Equation 2.7, the subtest-scale stress can be approximated as

$$\mathcal{T}_{ij}^{M,d} = -2(C_s\hat{\Delta})^2|\hat{\bar{S}}|\hat{\bar{S}}_{ij}, \tag{2.11}$$

where $\hat{\Delta}$ is the test filter scale. The application of the dynamic procedure leads to introduction of grid filtered SGS stresses given by

$$\mathcal{L}_{ij} = \mathcal{T}_{ij} - \hat{\tau}_{ij}, \tag{2.12}$$

$$= \widehat{\bar{u}_i \bar{u}_j} - \hat{\bar{u}}_i \hat{\bar{u}}_j. \tag{2.13}$$

In the dynamic procedure, the value of C_s is chosen in such a way that the error (also called as Germano identity error) given in the below equation is minimized

$$\epsilon_{ij} = \mathcal{T}_{ij}^{M,d} - \hat{\tau}_{ij}^{M,d} - \mathcal{L}_{ij}^{d}, \tag{2.14}$$

$$= -2(C_s\hat{\Delta})^2 |\hat{\bar{S}}| \hat{\bar{S}}_{ij} + 2\widehat{[(C_s\Delta)^2 |\bar{S}| \bar{S}_{ij}]} - \mathcal{L}_{ij}^{d}. \tag{2.15}$$

The computation of C_s in the above equation that minimizes the Germano identity error is not straight-forward as Equation 2.14 is a tensor equation (three equations in case of two-dimensional flows) for only one unknown C_s. Also, the coefficient C_s in the second term of Equation 2.15 is inside the test filter operator. However, it is often approximated as

$$\epsilon_{ij} = -2(C_s\hat{\Delta})^2 |\hat{\bar{S}}| \hat{\bar{S}}_{ij} + 2(C_s\Delta)^2 \widehat{|\bar{S}| \bar{S}_{ij}} - \mathcal{L}_{ij}^{d}, \tag{2.16}$$

which makes the formulation mathematically consistent only when C_s is a constant valued variable.

$$(C_s\Delta)^2 = \frac{\mathcal{M}_{ij}\mathcal{L}_{ij}^{d}}{\mathcal{M}_{ij}\mathcal{M}_{ij}}, \tag{2.17}$$

where

$$\mathcal{M}_{ij} = 2\widehat{|\bar{S}| \bar{S}_{ij}} - 2\left(\frac{\hat{\Delta}}{\Delta}\right) |\hat{\bar{S}}| \hat{\bar{S}}_{ij}. \tag{2.18}$$

From the original dynamic Smagorinsky model [118], it was found that the denominator in Equation 2.17 can become very small leading to excessively large value of C_s. Furthermore, Equation 2.17 becomes mathematically ill-posed since we factor C_s from the convolution filter (i.e., see Equation 2.16). Therefore some type of averaging is necessary in practice as given below

$$(C_s\Delta)^2 = \frac{< \mathcal{M}_{ij}\mathcal{L}_{ij}^{d} >_h^+}{< \mathcal{M}_{ij}\mathcal{M}_{ij} >_h}, \tag{2.19}$$

where $< \cdot >_h$ denotes the spatial averaging, and $< \cdot >_h^+ = 0.5(< \cdot > + | < \cdot > |)$ denotes the positive clipping. The above averaging gives a global value of C_s, which changes over time. Even though the spatial adaptivity of the dynamic model is lost due to this averaging procedure, the eddy viscosity field given by Equation 2.65 provides spatial variability. One of the advantage of the dynamic Smagorinsky model is that the numerator can also take negative values corresponding to backscatter, i.e., transfer of energy from small scales to large scales. If the averaging is not done, the dynamic model leads to a highly variable eddy viscosity field and can cause numerical simulations to become unstable [218, 244]. These findings are also applicable to data-driven turbulence closure modeling as demonstrated in recent studies [238, 28]. From computational point of view, the dynamic Smagorinsky model often stabilizes numerical schemes by providing absolute dissipation to numerical oscillations associated with truncation or aliasing errors at the small scales [332, 234].

2.3 Data-driven Turbulence Closure

In this section, we outline different data-driven turbulence closure frameworks investigated in this work. As discussed in Section 2.2, we try to approximate τ_{ij} using resolved flow variables on coarse grid in subgrid-scale modeling. We can consider this as a regression problem that can be studied using various classes of supervised machine learning algorithms. In the case of supervised algorithms, we try to learn the optimal map between inputs and outputs. We focus on two algorithms: an artificial neural network (ANN) also called as multilayer perceptron and convolutional neural network (CNN) to build data-driven closure models.

An artificial neural network consists of several layers made up of the predefined number of nodes (also called as neurons). A node combines the input from the data with a set of coefficients called weights. These weights either amplify or dampen the input and thereby assign the significance to the input in relation to the output that the ANN is trying to learn. In addition to the weights, these nodes have a bias for each input to the node. The input-weight product and the bias are summed and this sum is passed through a node's activation function. The activation function introduces nonlinearity and this allows the neural network to map complex relations between inputs and outputs. The above process can be described using the matrix operation as given by [132]

$$S^l = \mathbf{W}^l X^{l-1}, \tag{2.20}$$

where X^{l-1} is the output of the $(l-1)$th layer, $\mathbf{W}^l$ is the matrix of weights for the lth layer. The output of the lth layer is given by

$$X^l = \zeta(S^l + B^l), \tag{2.21}$$

where B^l is the vector of biasing parameters for the lth layer and ζ is the activation function. If there are L layers between the input and the output, then the mapping of the input to the output can be derived as follow

$$\tilde{Y} = \zeta_L(\mathbf{W}^L, B^L, \ldots, \zeta_2(\mathbf{W}^2, B^2, \zeta_1(\mathbf{W}^1, B^1, X))), \tag{2.22}$$

where X and $\tilde{Y}$ are the input and output of the ANN, respectively.

The matrix $\mathbf{W}$ and B are optimized through backpropagation and some optimization algorithm. The backpropagation algorithm provides a way to compute the gradient of the objective function efficiently and the optimization algorithm gives a rapid way to learn optimal weights. For the regression problem, usually, the objective is to learn the weights associated with each node in such a way that the root mean square error between the true labels Y and output of the neural network $\tilde{Y}$ is minimized. The backpropagation algorithm proceeds as follows: (i) the input and output of the neural network are specified along with some initial weights, (ii) the training data is run through the network to produce output $\tilde{Y}$ whose true label is Y, (iii) the derivative of the objective function with each of the training weight is computed using the chain rule, (iv) the weights are updated based on the learning rate and then we go to step (ii). We continue to iterate through this procedure until convergence or the maximum number of iterations is reached. There are a number of ways in which the weights can be initialized [121], the optimization algorithm is selected [365, 175, 328], and the loss function be regularized [386, 355] either to speed up the learning process or to prevent overfitting. Furthermore, highly nonlinear relationship between the input and output (as in the case of turbulence) necessitates the need of deep neural network architecture, which are prone to overfitting. Pruning neural network weights can significantly reduce the parameter count leading to better generalization [23].

It has been demonstrated in many studies how an ANN can be used for learning input-output relationship in the context of turbulence closure modeling [213, 238, 233, 388, 28, 354, 427, 413, 417]. We use two types of mapping using ANN as shown in Figure 2.1. The first one is the point-to-point mapping in which only the information

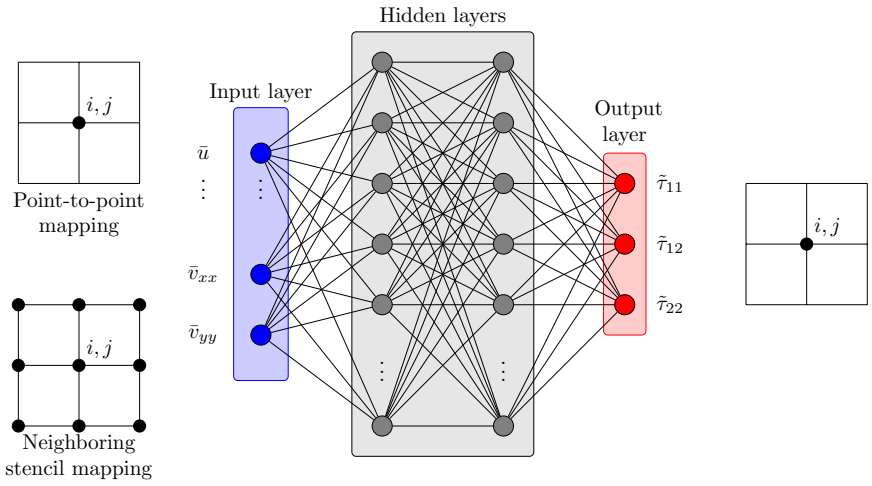

Figure 2.1: Feedforward neural network for point-to-point and neighboring stencil mapping of resolved flow variables to SGS stresses.

at a point is used to learn the SGS stresses at that point. We can include different features at that point and evaluate its predictive capability by means of probability density function based analysis. We use three classes of point-to-point mapping in our data-driven closure models as given below

$$\mathbb{M}1 : \{\bar{u}, \bar{v}\} \in \mathbb{R}^2 \rightarrow \{\tilde{\tau}_{11}, \tilde{\tau}_{12}, \tilde{\tau}_{22}\} \in \mathbb{R}^3, \tag{2.23}$$

$$\mathbb{M}2 : \{\bar{u}, \bar{v}, \bar{u}_x, \bar{u}_y, \bar{v}_x, \bar{v}_y\} \in \mathbb{R}^6 \rightarrow \{\tilde{\tau}_{11}, \tilde{\tau}_{12}, \tilde{\tau}_{22}\} \in \mathbb{R}^3, \tag{2.24}$$

$$\mathbb{M}3 : \{\bar{u}, \bar{v}, \bar{u}_x, \bar{u}_y, \bar{v}_x, \bar{v}_y, \bar{u}_{xx}, \bar{u}_{yy}, \bar{v}_{xx}, \bar{v}_{yy}\} \in \mathbb{R}^{10} \rightarrow \{\tilde{\tau}_{11}, \tilde{\tau}_{12}, \tilde{\tau}_{22}\} \in \mathbb{R}^3, \tag{2.25}$$

where $\bar{u}$, $\bar{v}$ are the the velocities in x and y direction, the subscript x and y denote the first-derivative, the subscript xx and yy are the second-derivative, and $\tilde{\tau}_{11}, \tilde{\tau}_{12}, \tilde{\tau}_{22}$ are the approximated SGS stresses.

The second approach is to use the information at neighbouring points to learn SGS stresses at a point. We can either use information of just north, south, east, and west points or information at all nine neighbouring points. In our neighboring stencil mapping, we use information at nine grid points. As we will see in Section 2.4, one of the advantages of this approach is that the ANN can learn the input-output mapping with less number of input features. Similar to point-to-point mapping, we use three classes of input features for neighboring stencil mapping. Therefore, in case of neighboring stencil mapping, we will have nine times the number of input features as in case of point-to-point mapping.

In addition to ANN, we also investigate CNN for subgrid-scale modeling. CNNs have been found to perform better than ANNs when the data is in the form of snapshots such as images and is widely used for computer vision tasks such as object detection [186, 320], and improving the quality of images [172, 81]. CNNs have also been successfully applied for detecting flow disturbances [146], super-resolution analysis of turbulent flow field [108], and turbulence closure modeling [28, 194, 264, 265]. One of the differences between ANN and CNN is that the training sample to the CNN is not given as one-dimensional vector but as a two-dimensional snapshot image. This will preserve the original multi-dimensional structure and will aid in learning the SGS stresses. Apart from that, the number of parameters to be learned in CNN is significantly less than ANN due to parameter sharing scheme.

The Conv layers are the fundamental building blocks of the CNN. Similar to weights in case of ANN, Conv layers have filters, also called as kernels that has to be learned using the backpropagation algorithm. The filter has a smaller shape but it extends in through the full depth of the input volume of previous layer. For example, if the input to the CNN has $64 \times 64 \times 3$ dimension where 3 is the number of input features, the kernels of first Conv layer can have $3 \times 3 \times 3$ shape. During the forward propagation, we convolve the filter 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 entries 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 given as

$$S^l_{ij} = \sum_{p=-\Delta_i/2}^{\Delta_i/2} \sum_{q=-\Delta_j/2}^{\Delta_j/2} \sum_{r=-\Delta_k/2}^{\Delta_k/2} \mathbf{W}^l_{pqr} X^{l-1}_{i+p\ j+q\ k+r} + B_{pqr}, \tag{2.26}$$

where Δ_i, Δ_j, Δ_k are the sizes of filter in each direction, $\mathbf{W}^l_{pqr}$ are the entries of the filter for lth Conv layer, B_{pqr} is the biasing parameter, and X^{l-1}_{ijk} 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 non-linear map between inputs and outputs. The output of the lth layer is givn by

$$X^l_{ijk} = \zeta(S^l_{ijk}), \tag{2.27}$$

where ζ 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 as zero-padding. The zero-padding allows us to control the shape of the output volume and is used in our data-driven closure framework to preserve the shape so that input and output width and height are the same. The size of the zero-padding is an additional hyperparameter in CNN.

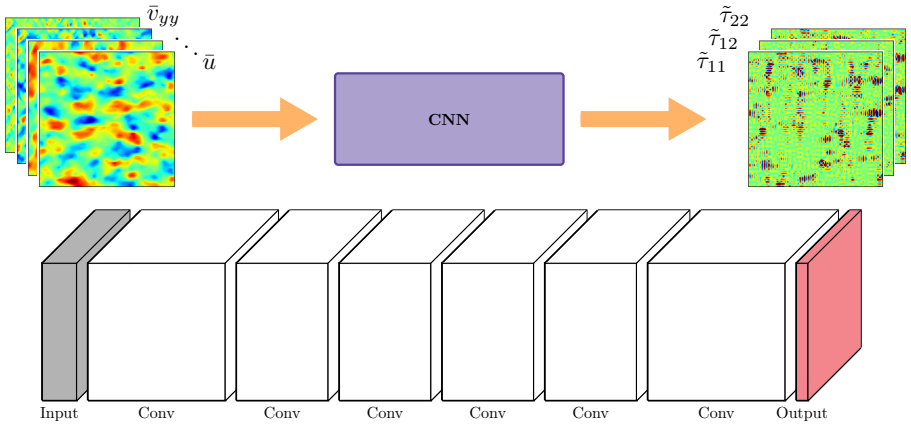

Figure 2.2: Convolution neural network for mapping of resolved variables to SGS stresses. Our CNN architecture is fairly simple and we use zero padding to keep the same shape as we go from input to the output.

Figure 2.2 shows the schematic of the CNN architecture used in our data-driven closure framework. The input to the CNN is obtained by stacking snapshots of resolved variables and their derivatives at the coarse grid. Similar to the ANN, we use three classes of input features as given in Equations 2.23 - 2.25. Therefore, for model $\mathbb{M}1$, each sample of the input volume will have $64 \times 64 \times 2$ shape and the the sample of output volume will have $64 \times 64 \times 3$ shape.

2.4 Intelligent SGS Modeling

The present study is focused on the comparison of data-driven closure approaches discussed in Section 2.3 for SGS modeling. We use two-dimensional Kraichnan turbulence problem as our prototype example to show the comparison of different frameworks. The purpose of this test problem is to see how the abundant population of randomly generated vortices evolve over time [366]. For data-driven frameworks, we use true subgrid-scale stresses (τ_{ij}) generated by solving the two-dimensional Navier-Stokes equation with DNS. The computational domain is square in shape with

the dimension $[0, 2\pi] \times [0, 2\pi]$ in x and y directions. The domain has the periodic boundary condition in x and y directions. We use pseudo-spectral solver for DNS of Kraichnan turbulence problem. The pseudo-spectral solver is accurate in a sense that it does not introduce any discretization error. We use hybrid explicit third-order Runge-Kutta scheme and implicit Crank-Nicolson scheme for the time integration. It should be noted that we solve the Navier-Stokes equations using streamfunction-vorticity formulation and then compute primitive variables using a spectral method for differentiation. The streamfunction-vorticity formulation eliminates the pressure term from the momentum equation and hence, there is no odd-even coupling between the pressure and velocity. This allows us to use collocated grid instead of the staggered grid.

The DNS solution is computed for $\mathrm{Re} = 4000$ with the grid resolution of 1024×1024. We integrate the solution from time $t = 0$ to $t = 4$ with $\Delta t = 1 \times 10^{-3}$. The evolution of the vorticity field and the energy spectrum for two-dimensional Kraichnan turbulence are shown in Figure 2.3. The initial condition for the energy spectrum is assigned in such a way that the maximum value of the energy is designed to occur at the wavenumber $k = 10$. Using this energy spectrum and random phase function, the initial vorticity field is assigned. The random vorticity field assigned is kept identical (using constant seed) in all our numerical experiments for comparison and reproducing the results. Interested readers are referred to related work [272, 336] for the energy spectrum equation and randomization process. We collect 400 snapshots of data from time $t = 0$ to $t = 4$. The Kraichnan-Batchelor-Leith (KBL) theory states that the energy spectrum of two-dimensional turbulence is proportional to k^{-3} in the inertial range and we observe this behavior with our numerical solution at $t = 2.0$ and $t = 4.0$ as shown in Figure 2.3. For LES, we coarsen the solution on 64×64 grid resolution using the spectral cut-off filter. The resolved flow variables at the coarse grid are then used to compute input features for data-driven turbulence closure models.

We analyze the performance of data-driven closure models against the dynamic Smagorinsky model discussed in Section 2.2.1. One of the advantages of DSM is that the Smagorinsky coefficient is computed using the resolved field variables in a dynamic fashion and does not require *a priori* coefficient specification. Due to this advantage, DSM is widely used in LES of engineering and geophysical applications [178, 111, 169, 253]. The only parameter that has to be specified for the DSM is the filter width ratio (i.e., a ratio between the test and grid filters). We use the spectral cut-off filter as a test filter and the test filter scale is $\hat{\Delta} = 2\Delta$. Figure 2.4 shows

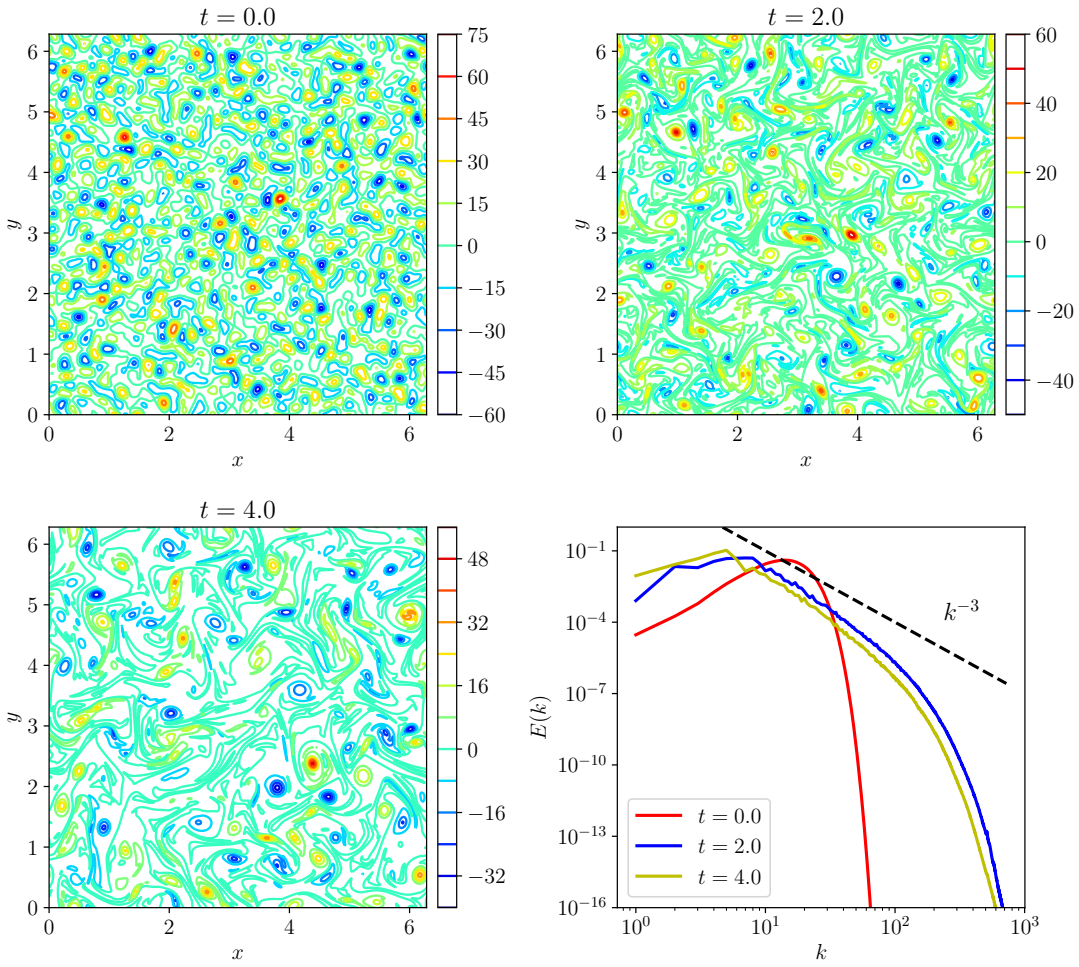

Figure 2.3: Time evolution of the vorticity field and energy spectrum from time $t = 0.0$ to $t = 4.0$ for Re=4000 at grid resolution 1024×1024.

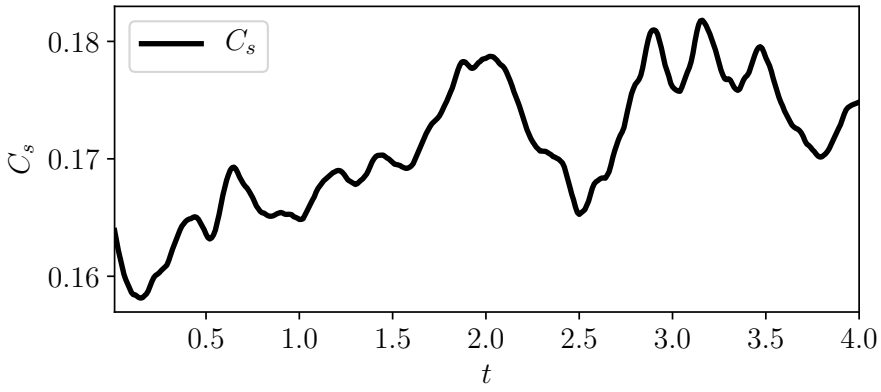

Figure 2.4: Evolution of the Smagorinsky coefficient (C_s) for two-dimensional Kraichnan turbulence problem computed using Lily's version of dynamic model with positive clipping.

the temporal evolution of the Smagorinsky coefficient from time $t = 0.0$ to $t = 4.0$ computed with the DSM. Ths Smagorinsky coefficient changes between 0.16 to 0.18. We have to use this low-pass filtering operation eight times for the DSM and the procedure becomes computationally expensive compared to the static Smagorinsky model. A data-driven turbulence closure model can also be developed to learn dynamic eddy viscosity (computed by DSM) instead of learning true SGS stresses. The similar approach was implemented by [340] for learning Bardina's scale similar subgrid-scale model to improve computational performance. We use the similar framework for learning eddy viscosity computed by the DSM, and is detailed in Section 2.5. The DNS code for the pseudo-spectral solver, and the code for DSM is implemented using vectorization in Python. This will allow us to compare the computational performance of the DSM with data-driven closure models fairly (the most popular libraries for machine learning like Keras, Tensorflow are available in Python).

We use two metrics to determine the performance of data-driven closure models. First one is the cross-correlation between true SGS stresses and the predicted SGS stresses. The cross-correlation (cc) is calculated using the below formula

$$cc = \frac{\text{cov}(Y, \tilde{Y})}{\sigma_Y \sigma_{\tilde{Y}}}, \tag{2.28}$$

where the covariance (cov) is defined as

$$\text{cov}(Y, \tilde{Y}) = E[(Y - E[Y])(\tilde{Y} - E[\tilde{Y}])]. \tag{2.29}$$

In above equations, Y is the true field, $\tilde{Y}$ is the predicted field, σ_Y is the standard deviation of Y, $\sigma_{\tilde{Y}}$ is the standard deviation of $\tilde{Y}$, $E[Y]$ is the expected value of the true field, and $E[\tilde{Y}]$ is the expected value of the predicted field. The expected value and the standard deviation for a sample field Y can be given as

$$E[Y] = \frac{\sum_{i=1}^{n} y_i}{n}, \quad \sigma_Y = \sqrt{\frac{\sum_{i=1}^{n} (y_i - E[Y])^2}{n}}. \tag{2.30}$$

In addition to the cross-correlation, we assess the model's performance using probability density function (PDF) based analysis. We test all data-driven closure models using 350 snapshots of training data from time $t = 0.0$ to $t = 3.5$. We use 20% of the training data for validation. We use the resolved field variables at time $t = 4.0$ to determine SGS stresses as out-of-training data snapshot. This data has not been

seen by the neural network during training and hence the model's performance should be measured against this data. In addition to using 350 data snapshots for training, we test extremely sub-sampled data with 70, and 24 snapshots. This will help us in understanding how much data to use for each of these data-driven closure models for learning true SGS stresses efficiently. We normalize all input features and output labels in the range $[-1, 1]$ using the minimum and maximum values for each features. The normalization of data helps in giving all input features equal importance and also allow optimization algorithm to converge faster. The hyperparameters for all neural network architectures are selected using gridsearch algorithm coupled with five-fold cross-validation and the procedure is discussed in detail in Appendix 2.B. We have to be cautious when measuring the CPU time for deployment of trained model and the sample code for CPU time measurement is given in Appendix 2.C. The sample code for our ANN and CNN architecture is provided in Appendix 2.D.

2.4.1 Point-to-point Mapping

We first discuss the performance of point-to-point mapping ANN in predicting true SGS stresses. Table 2.1 gives the cross-correlation between true and predicted SGS stresses for three different models (i.e., models $\mathbb{M}1$, $\mathbb{M}2$, $\mathbb{M}3$ presented in Section 2.3). The cross-correlation between the DSM and true stresses is low because the DSM model cannot capture the phase correctly (as we will see at the end of this section). It can be seen that the correlation between true and predicted SGS stresses is very poor when we use only coarse-grid resolved velocities at a point to determine the stresses at that point. It is clear from Figure 2.5 that the point-to-point mapping approach is unable to map coarse-grid resolved velocities to SGS stresses and it calculates completely wrong stresses. To investigate further, weather the use of other activation function helps to improve the prediction or not, we test Tanh and Sigmoid function with the same neural network architecture. We find that the predicted stresses are similar even with other activation functions. This confirms that additional input features are needed to learn more accurate mapping between inputs and outputs. The PDF for true and predicted stresses with different activation functions are provided in Appendix B for point-to-point mapping with model $\mathbb{M}1$. For the DSM, the PDF shape is similar to the true PDF despite having low cross-correlation. The DSM captures the bulk eddy viscosity, but the phase is completely distorted with the DSM. From Figure 2.6 and Figure 2.7, we observe an improvement in the prediction of SGS

stresses as we start including more features like coarse-grid velocity gradients (i.e., model $\mathbb{M}2$) and the Laplacian on coarse-grid velocities (i.e., model $\mathbb{M}3$). We test the point-to-point ANN for subsampled data using 70 and 24 data snapshots. The cross-correlation between true and predicted stress is almost similar to the one with 350 data snapshots. Figure 2.8 displays the PDF of true stresses and the predicted stresses computed using point-to-point ANN with a different number of snapshots. It can be observed that the PDF predicted with different number of snapshots is almost the same for ANN point-to-point mapping. Therefore, we can conclude that the ANN can be trained with less number of samples without a significant drop in accuracy. However, neural networks are prone to overfit when we use fewer data and the ability of neural networks to approximate on unseen data reduces. There are different methods to prevent overfitting such as data augmentation, regularization, weight decay, and dropout that should be used when less data is available for learning SGS stresses.

In terms of the computational performance, point-to-point mapping requires less training time for learning SGS stresses from resolved flow variables. This approach is particularly attractive for complex or unstructured mesh and has been applied in many studies [427, 213, 406, 112, 388]. As illustrated in these works, our analysis with simple input features like resolved velocities and their derivatives also shows that the input features are critical for effective learning of SGS stresses for point-to-point mapping approach. From Table 2.1, we can see that the train time does not increase linearly with an increase in the number of input features. The train time for the neural network mainly depends upon its architecture (how deep and wide it is), and the number of training samples. Since we are using the same architecture for all models, we observe that the train time is similar for all cases. In terms of the test time or deployment time, the point-to-point ANN is slightly slower than the DSM (around 1.3 times).

2.4.2 Neighboring Stencil Mapping

In this section, we discuss the numerical assessment of results for ANN with neighboring stencil mapping. Table 2.2 reports the cross-correlation between true and predicted SGS stresses for neighboring stencil mapping ANN. Figure 2.9 shows that this framework can predict SGS stresses with sufficient accuracy close to the dynamic Smagorinsky model with just coarse-grid velocities (i.e., model $\mathbb{M}1$). If we compare

Table 2.1: Cross-correlation between true and predicted SGS stresses, and CPU time for different models with point-to-point mapping for ANN.

Model	N_s	$cc(\tau_{11})$	$cc(\tau_{12})$	$cc(\tau_{22})$	Train time	Test time
DSM	-	0.011	-0.008	0.011	-	0.0095
$\mathbb{M}1$	350	0.043	-0.001	0.044	1577.11	0.0138
$\mathbb{M}2$	350	0.343	0.261	0.343	1608.95	0.0132
$\mathbb{M}3$	350	0.556	0.487	0.556	1642.82	0.0127
$\mathbb{M}3$	70	0.555	0.481	0.555	322.08	0.0125
$\mathbb{M}3$	24	0.549	0.465	0.550	112.04	0.0128

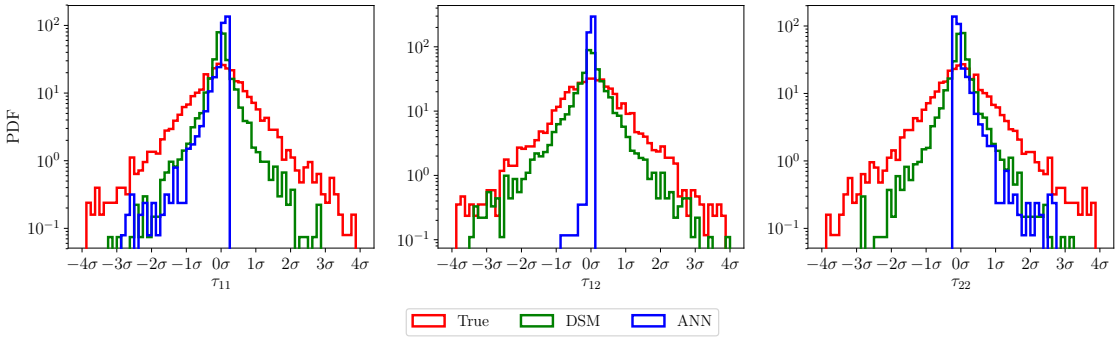

Figure 2.5: Probability density function for SGS stress distribution with point-to-point mapping. The ANN is trained using $\mathbb{M}1$. The training set consists of 350 time snapshots from time $t = 0.0$ to $t = 3.5$ and the model is tested for $400th$ snapshot at $t = 4.0$.

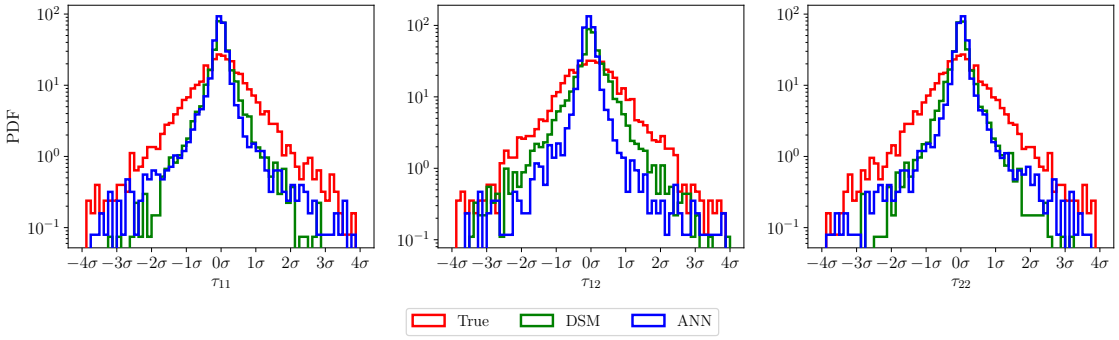

Figure 2.6: Probability density function for SGS stress distribution with point-to-point mapping. The ANN is trained using $\mathbb{M}2$. The training set consists of 350 time snapshots from time $t = 0.0$ to $t = 3.5$ and the model is tested for $400th$ snapshot at $t = 4.0$.

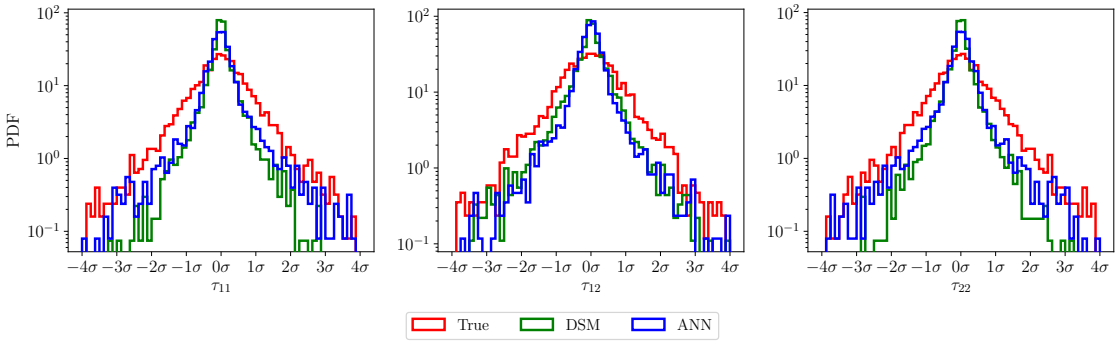

Figure 2.7: Probability density function for SGS stress distribution with point-to-point mapping. The ANN is trained using $\mathbb{M}3$. The training set consists of 350 time snapshots from time $t = 0.0$ to $t = 3.5$ and the model is tested for $400th$ snapshot at $t = 4.0$.

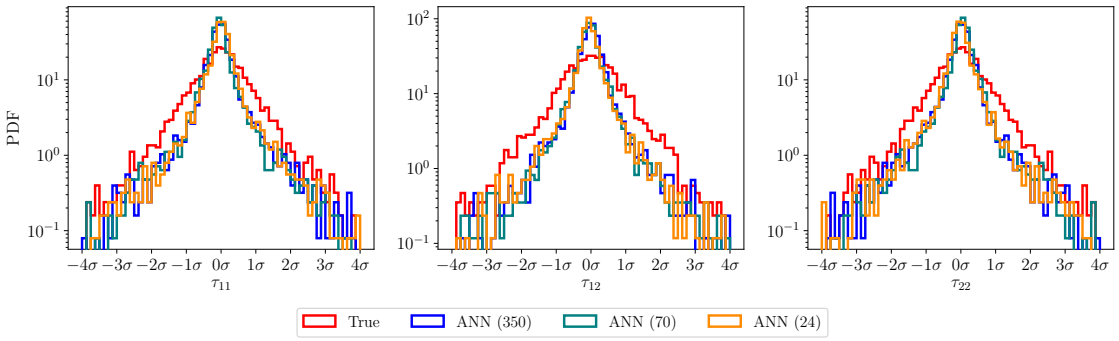

Figure 2.8: Probability density function for SGS stress distribution with point-to-point mapping. The ANN is trained using $\mathbb{M}3$. The model is trained using different number of snapshots between $t = 0.0$ to $t = 3.5$ and the model is tested for $400th$ snapshot at $t = 4.0$.

Table 2.1 and Table 2.2, we see that the neighboring stencil mapping with model $\mathbb{M}1$ provides slightly better correlation than utilizing coarse-grid velocities and their derivatives at a single point. This clearly shows the benefit of incorporating neighboring information to determine SGS stresses. As we begin adding more features (i.e., first and second derivative of coarse-grid resolved velocities), we start getting correlation up to 0.8 between true and predicted SGS stresses. From Figure 2.10 and Figure 2.11, we notice that the SGS stresses predicted by the ANN are very close to true stresses when the first derivative and Laplacian of coarse-grid velocities are also included in the training.

We examine this framework with the different number of data snapshots to check the optimal data needed for ANN to learn SGS stresses with sufficient accuracy. Figure 2.12 shows the PDF of true and predicted stresses calculated with neighboring

Table 2.2: Cross-correlation between true and predicted SGS stresses, and CPU time for different models with neighboring stencil mapping for ANN.

Model	N_s	$cc(\tau_{11})$	$cc(\tau_{12})$	$cc(\tau_{22})$	Train time	Test time
DSM	-	0.011	-0.008	0.011	-	0.0095
$\mathbb{M}1$	350	0.599	0.548	0.599	1675.92	0.0136
$\mathbb{M}2$	350	0.783	0.731	0.783	1845.62	0.0141
$\mathbb{M}3$	350	0.813	0.744	0.813	2065.11	0.0146
$\mathbb{M}3$	70	0.789	0.746	0.789	425.84	0.0152
$\mathbb{M}3$	24	0.786	0.721	0.786	139.49	0.0149

stencil mapping for different number of snapshots. From Table 2.2 we observe that there is a slight drop in cross-correlation as we decrease the amount of data utilized for training. The CPU time required for training drops significantly for less number of training data snapshots. Overall, we can conclude that the accuracy of the prediction will improve with the amount of the training data at the cost of higher computational overhead for training. One more advantage of this approach is that the neighboring stencil mapping can be employed for the complicated and unstructured mesh. This is one of the desirable features of any data-driven frameworks, as the turbulence closure model is deployed for complex fluid flow analysis, which is run on supercomputers. In the neighboring stencil mapping framework, the information at only a few neighboring nodes is required and it can be implemented without much of the communication overhead. For the deployment computational time, we get similar findings as to the point-to-point mapping ANN. The neighboring stencil mapping is around 1.5 times slower than DSM. However, to get the same order of accuracy with DSM, we will need to use a fine mesh for LES and this can be computationally expensive than employing neighboring stencil mapping ANN.

2.4.3 CNN Mapping

In this section, we present the predictive performance of CNN mapping to learn SGS stresses. Table 2.3 lists the cross-correlation between true and predicted SGS stresses computed using CNN mapping. CNN mapping provides the best prediction among three frameworks, and even with just coarse-grid resolved velocities as input features, we obtain cross-correlation around 0.78 between true and predicted SGS stresses. Figure 2.13 shows the PDF of true and predicted stresses calculated using the model $\mathbb{M}1$ with CNN mapping. CNN mapping can predict the spatial distribution

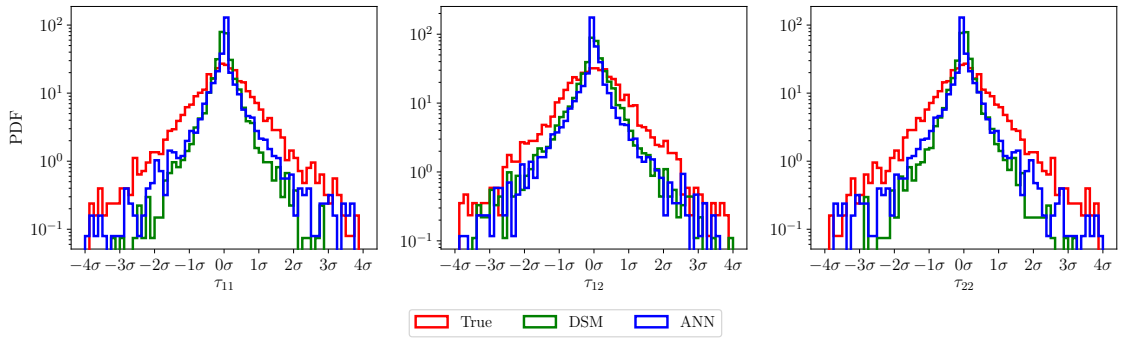

Figure 2.9: Probability density function for SGS stress distribution with neighboring stencil mapping. The ANN is trained using $\mathbb{M}1$. The training set consists of 350 time snapshots from time $t = 0.0$ to $t = 3.5$ and the model is tested for $400th$ snapshot at $t = 4.0$.

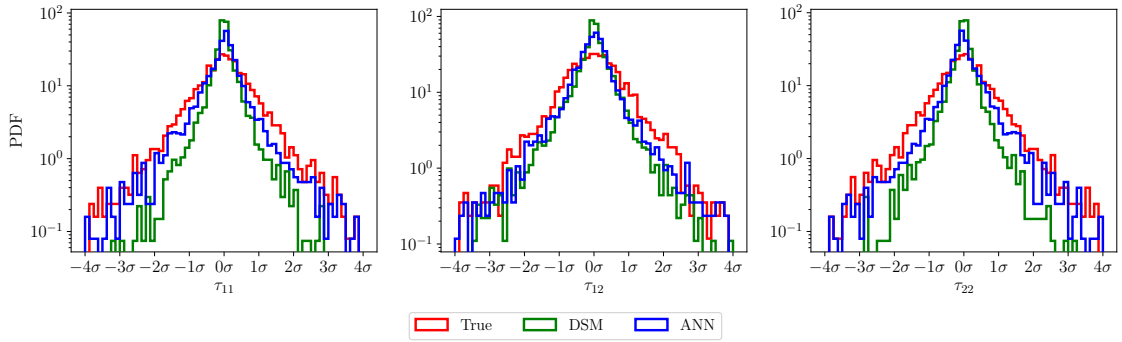

Figure 2.10: Probability density function for SGS stress distribution with neighboring stencil mapping. The ANN is trained using $\mathbb{M}2$. The training set consists of 350 time snapshots from time $t = 0.0$ to $t = 3.5$ and the model is tested for $400th$ snapshot at $t = 4.0$.

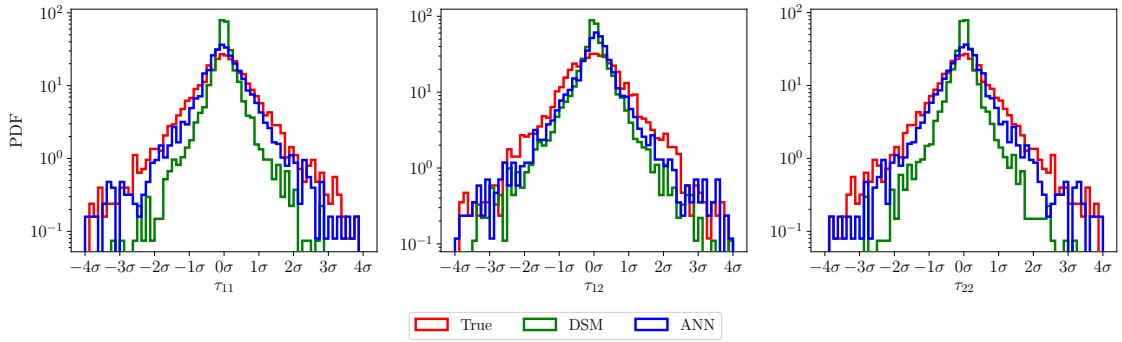

Figure 2.11: Probability density function for SGS stress distribution with neighboring stencil mapping. The ANN is trained using $\mathbb{M}3$. The training set consists of 350 time snapshots from time $t = 0.0$ to $t = 3.5$ and the model is tested for $400th$ snapshot at $t = 4.0$.

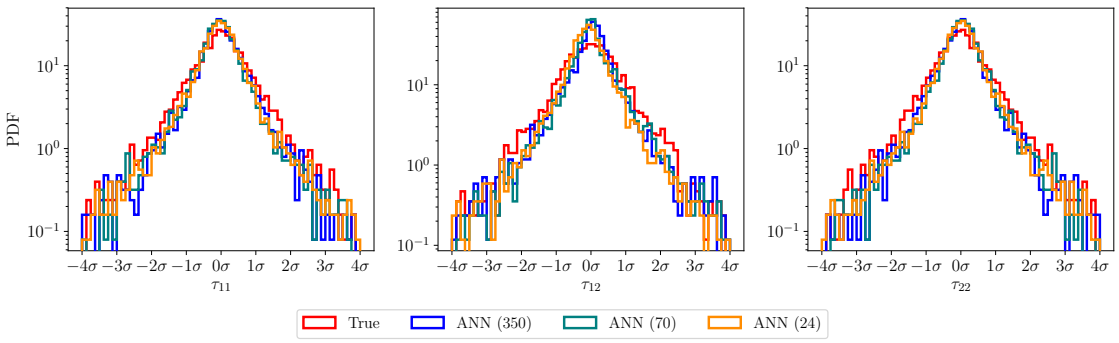

Figure 2.12: Probability density function for SGS stress distribution with neighboring stencil mapping. The ANN is trained using $\mathbb{M}3$. The model is trained using different number of snapshots between $t = 0.0$ to $t = 3.5$ and the model is tested for $400th$ snapshot at $t = 4.0$.

of stresses correctly and we observe that the true and predicted PDF are very close to each other. When we incorporate more input features in the form of first and second derivatives of coarse-grid velocities (i.e., model $\mathbb{M}2$, and $\mathbb{M}3$), we see an improvement in cross-correlation to around 0.84. Figure 2.14 and Figure 2.15 display the PDF of true and predicted stresses for models $\mathbb{M}2$ and $\mathbb{M}3$, respectively. A very good agreement between PDF of true and predicted SGS stresses are observed for CNN mapping with $\mathbb{M}2$ and $\mathbb{M}3$.

Table 2.3: Cross-correlation between true and predicted SGS stresses, and CPU time for different models with CNN mapping.

Model	N_s	$cc(\tau_{11})$	$cc(\tau_{12})$	$cc(\tau_{22})$	Train time	Test time
DSM	-	0.011	-0.008	0.011	-	0.0095
$\mathbb{M}1$	350	0.783	0.728	0.784	374.47	0.0024
$\mathbb{M}2$	350	0.828	0.779	0.827	391.33	0.0021
$\mathbb{M}3$	350	0.835	0.779	0.835	408.65	0.0017
$\mathbb{M}3$	70	0.736	0.674	0.739	77.25	0.0025
$\mathbb{M}3$	24	0.627	0.589	0.621	27.67	0.0025

We also evaluate the performance of CNN mapping with different amount of training snapshots for model $\mathbb{M}3$. Figure 2.16 shows the PDF of true and predicted stresses for the different number of snapshots. We can see that there is a shift in predicted PDF compared to true PDF for τ_{11} and τ_{22} when we use less number of training snapshots. Also the cross-correlation between true and predicted stresses have reduced when we utilize less number of data snapshots for training, and the performance is poorer than neighboring stencil mapping ANN with less number of

snapshots. In terms of the computational performance, CNN mapping surpasses both point-to-point and neighboring stencil mapping ANN. This is due to the weight sharing features of CNN and hence the number of parameters to be learned are less than ANN. The deployment computational time for CNN is around 0.2 times the time required by the DSM. Therefore, CNN can provide a more accurate prediction for LES at a less computational cost. Despite these advantages, the application of CNN for the unstructured grid is an open question. If the computational domain has a simple geometry and the data is available in the form of snapshots as in the case of box turbulence, wall-bounded flows, it is advantageous to use CNN. There have been several studies that introduce novel CNN architectures for point cloud data (as in the case of the unstructured grid) [416, 375, 371, 100]. With these novel CNN architectures, the improved predictive capability of CNN can be exploited for turbulence closure modeling.

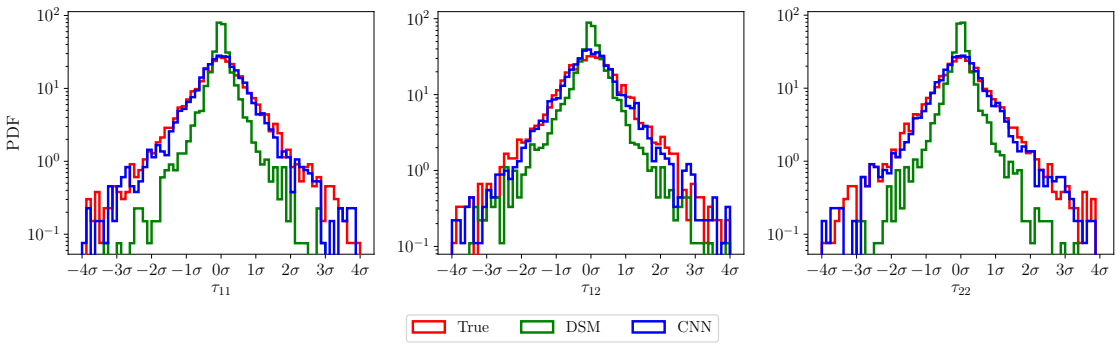

Figure 2.13: Probability density function for SGS stresses distribution with CNN mapping. The CNN is trained using $\mathbb{M}1$. The training set consists of 350 time snapshots from time $t = 0.0$ to $t = 3.5$ and the model is tested for $400th$ snapshot at $t = 4.0$.

Figure 2.17 displays the two-dimensional contour plot of true SGS stress τ_{12} and SGS stress predicted by the DSM, neighboring stencil mapping ANN, and CNN mapping. The DSM model captures the bulk eddy viscosity, but not the actual phase. This is the reason behind low value of cross-correlation between true SGS stresses and SGS stresses predicted by the DSM. Data-driven models on the other hand are able to capture both magnitude and phase correctly in comparison with true SGS stress τ_{12}.

To summarize our analysis, we show the cross-correlation between true and predicted SGS stresses in Figure 2.18 for all three data-driven closure models with a different number of snapshots. We have summarized the results only for model $\mathbb{M}3$, which includes coarse-grid velocities, coarse-grid velocity gradients, and the Laplacian

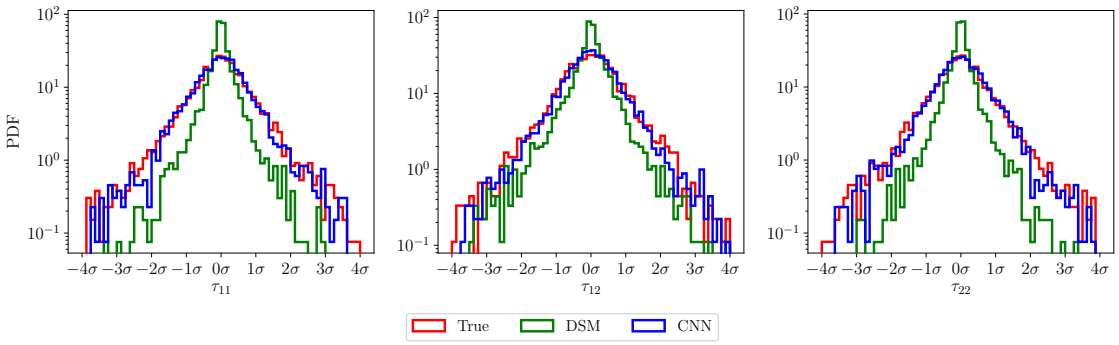

Figure 2.14: Probability density function for SGS stresses distribution with CNN mapping. The CNN is trained using $\mathbb{M}2$. The training set consists of 350 time snapshots from time $t = 0.0$ to $t = 3.5$ and the model is tested for $400th$ snapshot at $t = 4.0$.

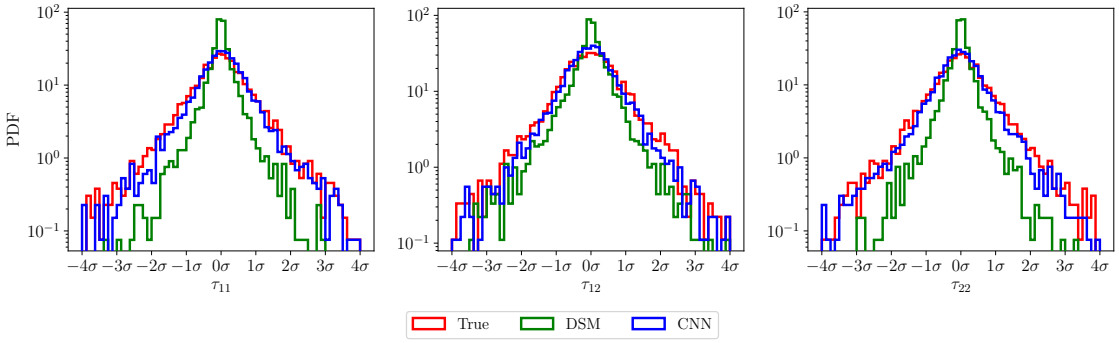

Figure 2.15: Probability density function for SGS stresses distribution with CNN mapping. The CNN is trained using $\mathbb{M}3$. The training set consists of 350 time snapshots from time $t = 0.0$ to $t = 3.5$ and the model is tested for $400th$ snapshot at $t = 4.0$.

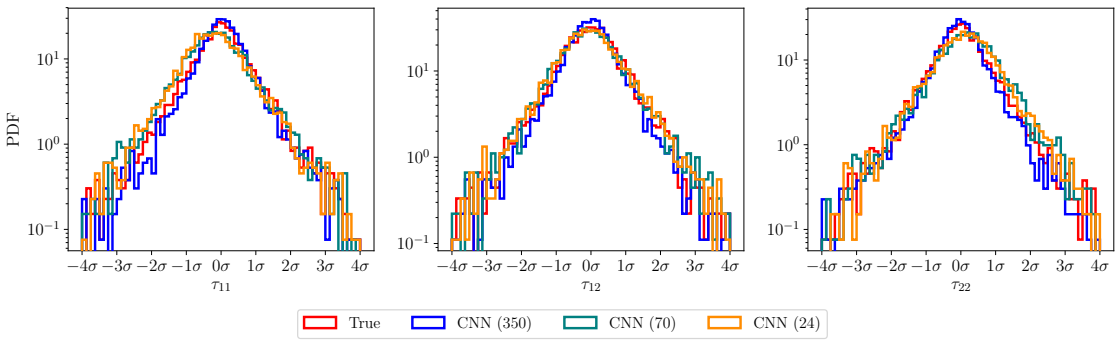

Figure 2.16: Probability density function for SGS stresses distribution with CNN mapping. The CNN is trained using $\mathbb{M}3$. The model is trained using different number of snapshots between $t = 0.0$ to $t = 3.5$ and the model is tested for $400th$ snapshot at $t = 4.0$.

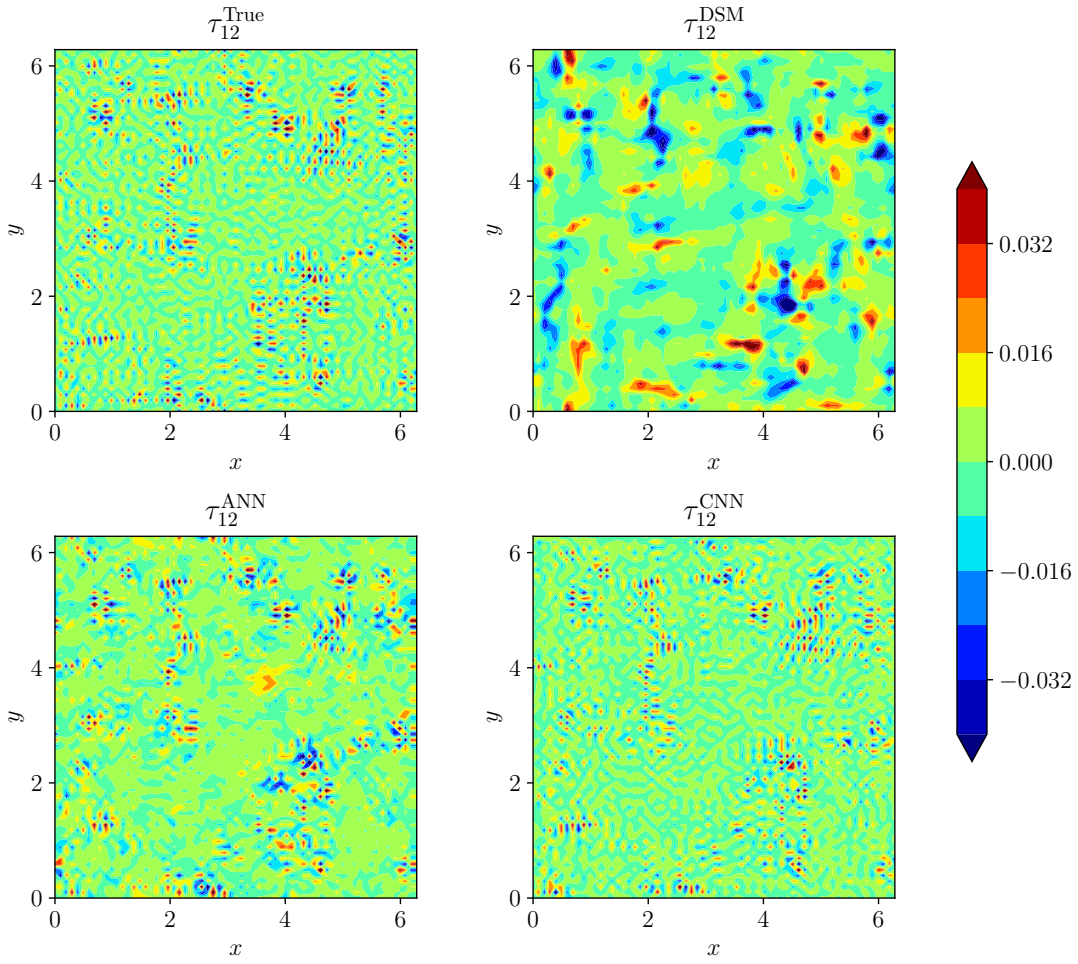

Figure 2.17: Two-dimensional contour plot of the SGS stress τ_{12} at time $t = 4.0$. τ_{12}^{ANN} is the SGS stress computed using neighboring stencil mapping ANN with model $\mathbb{M}3$. τ_{12}^{CNN} is the SGS stress computed using CNN mapping with model $\mathbb{M}3$.

of coarse-grid velocities. The model $\mathbb{M}3$ was found to give a better prediction for all data-driven models without incurring a high computational cost. It can be clearly seen that the CNN are more sensitive to the amount of training data than ANN in terms of its ability to predict SGS stresses. In terms of computational performance, the CNN mapping has the fastest performance (both training and testing/deployment) and has a potential to give accurate prediction with less computational price compared to the dynamic Smagorinsky model.

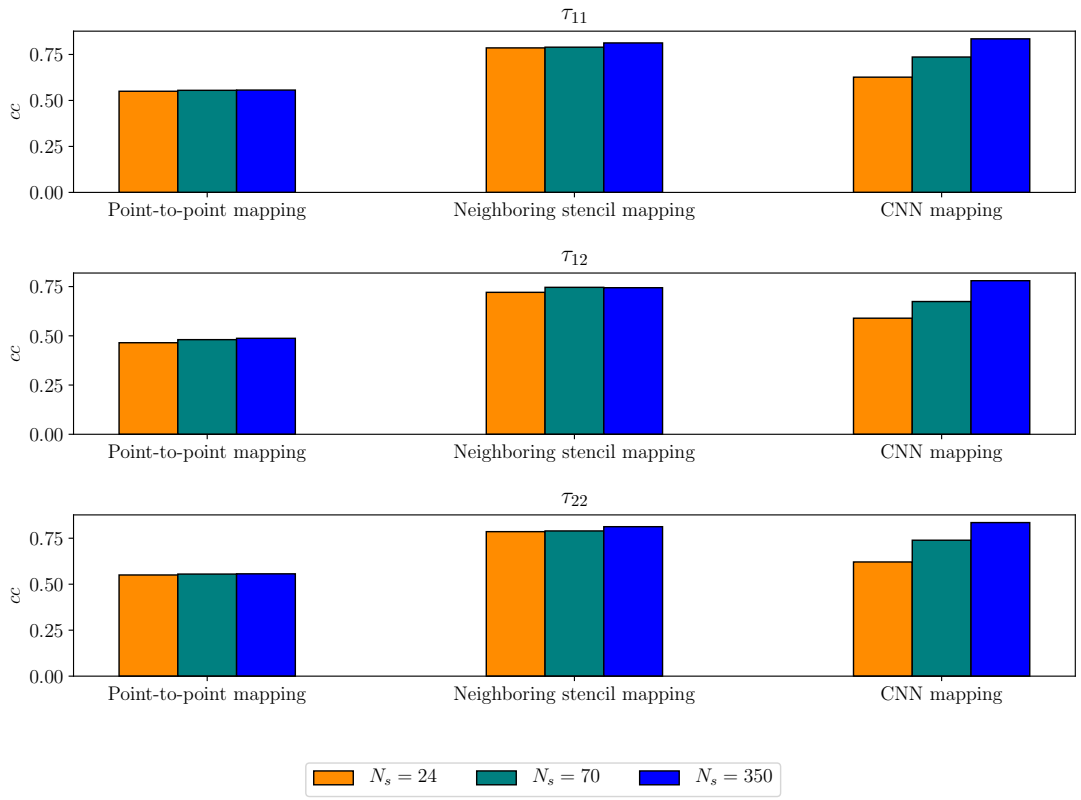

Figure 2.18: Summary of cross-correlation between true and predicted SGS stresses for different data-driven closure models trained using different number of snapshots.

2.5 Intelligent Eddy Viscosity Modeling

The data-driven frameworks presented in Section 2.4 learn SGS stress directly and hence attempt to improve it's prediction by trying to approximate true SGS stresses. Despite the improved prediction, neural networks are black-box models and these models cannot be interpreted or explained. In this section, we demonstrate intelligent eddy viscosity model as an alternative to the dynamic Smagorinksy model. Our aim here is to illustrate that these black-box data-driven tools can be also tailored to

accelerate such phenomenological eddy viscosity models.

In two-dimensional simulations, the dynamic procedure in the computation of Smagorisnky coefficient in DSM involves the application of low-pass filter eight times at each query. Instead of using these filtering operations, the neural networks can be trained to learn the dynamic eddy viscosity and the trained model can be deployed cost-effectively. One more advantage of this approach is that the numerical stability during the *a posteriori* deployment will be enforced. [238] noted that the clipping of vorticity source term is required to attain the numerical stability during the deployment of data-driven SGS model. The similar observation was also found by [28] for the decaying homogeneous isotropic turbulence problem. The data-driven SGS closure models can predict negative source term at some spatial locations and therefore violates the Boussinesq hypothesis for functional SGS modeling. The intelligent model to learn eddy viscosity can be built by enforcing the constraint such that eddy-viscosity predicted by neural network remains non-negative. This eddy viscosity is then used for computing SGS stresses using Equation 2.7. We only use coarse-grid velocity and their gradient in the dynamic procedure to compute the Smagorinsky coefficient. Hence we can include them as input features to learn eddy viscosity. The intelligent eddy viscosity model is given as

$$\mathbb{M}4 : \{\bar{u}, \bar{v}, \bar{u}_x, \bar{u}_y, \bar{v}_x, \bar{v}_y\} \in \mathbb{R}^6 \rightarrow \{\nu_e\} \in \mathbb{R}^1, \tag{2.31}$$

where the eddy viscosity ν_e is given as

$$\nu_e = (C_s\Delta)^2|\bar{S}|, \tag{2.32}$$

where $(C_s\Delta)^2$ is computed from Equation 2.19, and $|\bar{S}|$ is given by Equation 2.8. The similar framework was studied by [276], and they showed that the data-driven model gives two to eight times computational performance gain against the dynamic Smagorinsky model for wall-bounded turbulent flows.

The main advantage of this modeling approach is the numerical stability during *a posteriori* deployment and computational speed up. To avoid repetition, we compare the performance of intelligent eddy viscosity model with CNN mapping only. Table 2.4 lists the performance of intelligent eddy viscosity model trained with a different number of hyperparameters. The task of learning dynamic eddy viscosity is easier as compared to learning true SGS stresses and hence, we get cross-correlation up

to 0.98 with just one hidden layer and 16 kernels. The intelligent eddy viscosity model is around eight times faster than the DSM. If we use deep network similar to the data-driven SGS model, we still get a computational speedup of four times. Figure 2.19 show the comparison of true SGS stresses, SGS stresses predicted by DSM, and SGS stresses computed from different intelligent eddy viscosity models (different CNN architectures). The SGS stresses predicted by the CNN are very close to the stresses computed from the DSM, and hence we can get similar performance similar to the DSM at much less computational cost.

Table 2.4: Cross-correlation between DSM eddy viscosity and intelligent eddy viscosity predicted by data-driven models, and CPU time for different models with CNN mapping.

Model	Hyperparameters	$cc(\nu_e)$	Test time
DSM	-	-	0.0095
CNN-1	[16]	0.988	0.0012
CNN-2	[16,8,16]	0.994	0.0015
CNN-3	[16,8,8,8,8,16]	0.992	0.0024

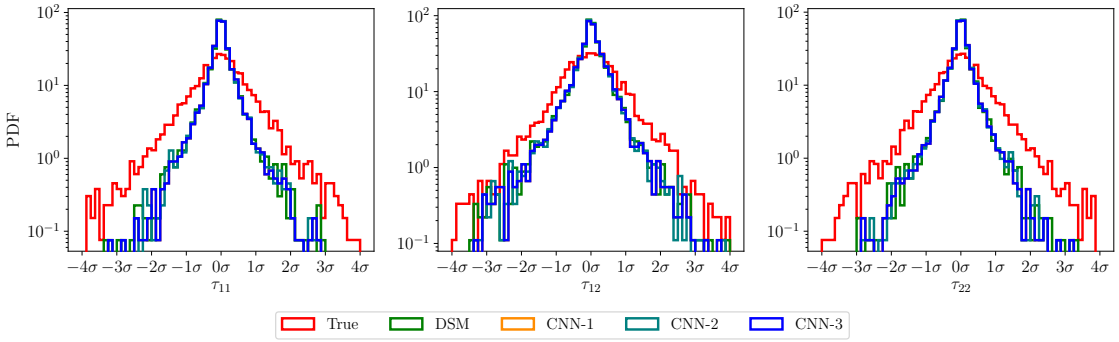

Figure 2.19: Probability density function for true SGS stresses distribution and stresses computed at $t = 4.0$ with DSM and intelligent eddy viscosity model. The CNN is trained using $\mathbb{M}4$.. The model is trained using 350 snapshots between $t = 0.0$ to $t = 3.5$ and the prediction is shown for time $t = 4.0$.

2.6 A Posterior Deployment with CNN-based Closure Model

After a systematic comparison of different deep learning frameworks for SGS closure modeling in the *a priori* settings, we demonstrate the performance of the CNN-based SGS closure model for Kraichnan turbulence in the *a posteriori* setting. The performance of any data-driven framework should be assessed based on the model's

ability to preserve coherent structures and scaling laws in the *a posteriori* setting. The vorticity transport equation can be written as

$$\frac{\partial \omega}{\partial t} + J(\omega, \psi) = \frac{1}{\mathrm{Re}} \nabla^2 \omega;, \tag{2.33}$$

$$J(\omega, \psi) = \frac{\partial \omega}{\partial x}\frac{\partial \psi}{\partial y} - \frac{\partial \omega}{\partial y}\frac{\partial \psi}{\partial x}, \tag{2.34}$$

where ω is the vorticity, ψ is the stremfunction, J is the Jacobian (or the nonlinear term), and Re is the Reynolds number of the flow. The multiscale nature of the flow is governed by the Reynolds number. If high Reynolds number flow is simulated on coarse-grid the finest structures of the flow will not be resolved and this can lead to accumulation of energy at the grid cut-off wavenumber (also referred to as the energy pile up). The vorticity and streamfunction are related by the conservation of mass equation given as

$$\nabla^2 \psi = -\omega. \tag{2.35}$$

The coarse-grained LES equations can be written as

$$\frac{\partial \bar{\omega}}{\partial t} + J(\bar{\omega}, \bar{\psi}) = \frac{1}{\mathrm{Re}} \nabla^2 \bar{\omega} + \Pi; \quad \nabla^2 \bar{\psi} = -\bar{\omega}, \tag{2.36}$$

where Π is the true SGS closure term given by

$$\Pi = J(\bar{\omega}, \bar{\psi}) - \overline{J(\omega, \psi)}. \tag{2.37}$$

We utilize the CNN architecture discussed in section 2.3 to learn the mapping from grid-filtered variables to the SGS closure term given in equation 2.37. The CNN model is trained by giving the whole snapshot of the domain as input and it predicts the SGS closure term for the whole domain. We compare two different CNN models given below

$$\text{CNN-M1} : \{\bar{\omega}, \bar{\psi}\} \in \mathbb{R}^2 \rightarrow \{\tilde{\Pi}\} \in \mathbb{R}^1, \tag{2.38}$$

$$\text{CNN-M2} : \{\bar{\omega}, \bar{\psi}, |\bar{S}|, |\nabla \bar{\omega}|\} \in \mathbb{R}^4 \rightarrow \{\tilde{\Pi}\} \in \mathbb{R}^1, \tag{2.39}$$

where

$$|\bar{S}| = \sqrt{4\left(\frac{\partial^2 \bar{\psi}}{\partial x \partial y}\right)^2 + \left(\frac{\partial^2 \bar{\psi}}{\partial x^2} - \frac{\partial^2 \bar{\psi}}{\partial y^2}\right)^2}, \quad |\nabla \bar{\omega}| = \sqrt{\left(\frac{\partial \bar{\omega}}{\partial x}\right)^2 + \left(\frac{\partial \bar{\omega}}{\partial y}\right)^2} \tag{2.40}$$

are eddy-viscosity kernel information input to the framework and $\tilde{\Pi}$ is the approximation to the true SGS closure term. The first and second derivative of vorticity and streamfunction field is embedded in the eddy-viscosity kernels and can be considered as a data pre-processing step to include prior information in training.

To ensure the numerical stability during *a posterior* deployment, the approximated SGS closure term $\tilde{\Pi}$ is post-processed before directly injecting it into the vorticity transport equation as given bellow

$$\Pi = \begin{cases} \tilde{\Pi}, & \text{if } (\nabla^2\bar{\omega})(\tilde{\Pi}) > 0 \\ 0, & \text{otherwise} \end{cases} \tag{2.41}$$

This post-processing step can be considered as a step similar to positive clipping in the dynamic Smagorinsky model given in Equation 2.19. This constraint ensures that the SGS closure term will not be used at points at which the eddy viscosity embedded in the SGS closure term is negative. The above constraint can also be attributed to the implicit assumption of positive eddy viscosity in the Boussinesq hypothesis for functional sub-grid modeling. It should be noted that Equation 2.41 does not allow the backscatter of enstrophy to ensure numerical stability requirement on a coarse mesh. However, this constraint still allows for both positive and negative values of the SGS closure term. This implies that the proposed framework predicts both vorticity forcing and damping at finer scales with positive and negative values of SGS closure term respectively.

We utilize second-order spatial discretization for the vorticity transport equation with the nonlinear Jacobian term discretized using energy conserving Arakawa numerical scheme [11]. We use third-order Runge-Kutta numerical scheme for time integration with the spectral Poisson solver employed for computing streamfunction from vorticity at every time step. We train the CNN using the DNS data generated for Reynolds number Re = 16000 at the grid resolution of 1024^2 from time $t = 0$ to 3.5 and filtered to the grid resolution of 128^2. The performance of the proposed framework is illustrated for two Reynolds number, Re = 16000 for which the CNN is trained, and Re = 32000 which lies outside the training data. We compare the performance of CNN models against the dynamic Smagorinsky model, which is one of the widely used SGS closure models in the LES community. Additionally, we also present results for coarse-grained forward simulation without any model (referred to as UNS) to show the energy pile at grid cut-off wavenumber.

We assess the performance of proposed frameworks using several statistical quantities such as angle-averaged energy spectrum, the time evolution of turbulent kinetic energy (denoted by TKE) and vorticity variance (denoted by $\sigma^2(\omega)$), and vorticity structure functions (denoted by S_ω). A detailed explanation of the computation of the angle-averaged energy spectrum can be found in [234]. The turbulent kinetic energy is computed as

$$TKE = \mu(u_f^2 + v_f^2), \tag{2.42}$$

where u_f and v_f are the fluctuating components of velocity given by

$$u_f = \bar{u} - \mu(\bar{u}), \tag{2.43}$$

$$v_f = \bar{v} - \mu(\bar{v}), \tag{2.44}$$

where $\mu(a)$ represents the spatial average of the field variable a. The velocity u, and v are computed by spectral differentiation of streamfunction as shown below

$$\bar{u} = \frac{\partial \bar{\psi}}{\partial y}; \quad \bar{v} = -\frac{\partial \bar{\psi}}{\partial x}. \tag{2.45}$$

The vorticity variance at each time step is computed as

$$\sigma^2(\bar{\omega}) = \mu((\bar{\omega} - \mu(\bar{\omega}))^2). \tag{2.46}$$

We compute the vorticity structure function using the formula given by [127] for two-dimensional turbulence and is shown below

$$S_\omega = < |\bar{\omega}(\mathbf{x} + \mathbf{r}) - \bar{\omega}(\mathbf{x})|^2 >, \tag{2.47}$$

where $<>$ indicates ensemble averaging, $\mathbf{x}$ is the position on the grid, and $\mathbf{r}$ is certain distance from this location.

Figure 2.20 displays the vorticity field at time $t = 4.0$ for DNS, filtered DNS, unresolved numerical simulation (UNS), dynamic Smagorinsky model (DSM), and proposed CNN frameworks fro Reynolds number Re = 16000. The vorticity field for UNS is heavily contaminated by high wavenumber noise. It can be observed that the proposed CNN frameworks can significantly reduce the noise and can predict the coherent structures similar to the DSM simulations.

Figure 2.21 shows the angle-averaged and compensated energy spectrum computed

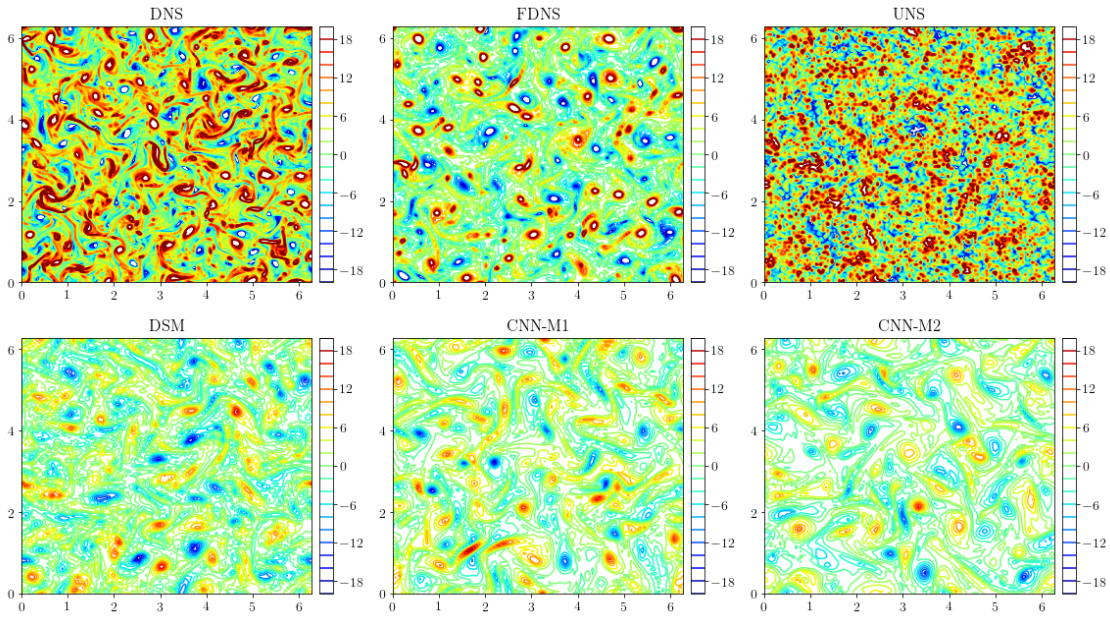

Figure 2.20: *A posteriori* vorticity field at final time $t = 4.0$ for Re $= 16000$ for proposed CNN frameworks. We also provide vorticity field for DNS, filtered DNS (FDNS), and unresolved numerical simulations (UNS) for comparison.

with the deployment of proposed CNN frameworks in coarse-grained simulations at Reynolds number Re $= 16000$. For comparison, we have also included the energy spectrum for DNS, dynamic Smagorinsky model (DSM), and unresolved numerical simulation (UNS). It can be observed that the proposed frameworks follow the k^{-3} scaling law in the inertial subrange which is to be expected for two-dimensional turbulence. We see no pile up of energy near high wavenumber with proposed frameworks like in the case of UNS, and the computed energy spectrum is very close to the DSM. Figure 2.22 displays the time evolution of TKE and vorticity variance from time $t = 0$ to $t = 4.0$ for Re $= 16000$. We notice that the proposed frameworks follow a very similar trend for both TKE and vorticity variance as the DSM. Figure 2.23 details the variation of vorticity structure function as the distance of a point from the grid location increases. It is again observed that the proposed frameworks produce vorticity structure functions similar to the DSM.

To verify weather proposed CNN frameworks have truly learned the mapping from coarse-grid variables to SGS closure term and not just remembered the training data, we deploy the trained CNN models for Reynolds number Re $= 32000$. The assessment of the proposed frameworks for this Reynolds number will also give us an insight on its generalizability. Figure 2.24 shows the qualitative assessment of vorticity field at

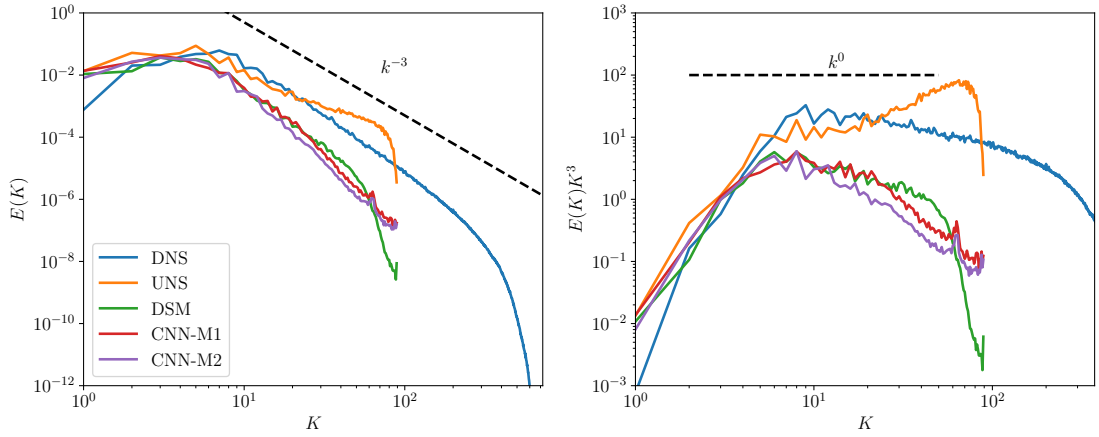

Figure 2.21: *A posteriori* kinetic-energy spectra (left) and compensated kinetic-energy spectra (right) for Re = 16000 at $t = 4$ and at grid resolution $N^2 = 128^2$. This Reynolds number is used for training the CNN.

time $t = 4.0$ for DNS, filtered DNS, unresolved numerical simulation (UNS), dynamic Smagorinsky model (DSM), and proposed CNN frameworks for Reynolds number Re = 32000. The vorticity field for UNS is heavily contaminated with high wavenumber noise. It can be observed that the proposed CNN frameworks can significantly reduce the noise and can retain the coherent structures similar to the DSM simulations.

Figure 2.25 reports the angle-averaged and compensated energy spectrum computed with the deployment of proposed CNN frameworks in coarse-grained simulations at Reynolds number Re = 32000. For comparison, we have also included energy spectrum for DNS, dynamic Smagorinsky model (DSM), and unresolved numerical simulation (UNS). It can be observed that the proposed frameworks follows the k^{-3} scaling law in the inertial subrange which is to be expected for two-dimensional turbulence. We observe no pile up of energy near high wavenumber with proposed frameworks like in the case of UNS, and the computed energy spectrum is very close to the DSM. Figure 2.26 shows the time evolution of TKE and vorticity variance from time $t = 0$ to $t = 4.0$ for Re = 32000. We notice that the proposed frameworks follow a very similar trend for both TKE and vorticity variance as the DSM. Figure 2.27 details the variation of vorticity structure function as the distance of a point from the grid location increases. It is again observed that the proposed frameworks produces vorticity structure functions similar to the DSM. The qualitative and quantitative assessment of the proposed frameworks for Reynolds number Re = 32000 shows a promise of data-driven methods to learn SGS closure term effectively.

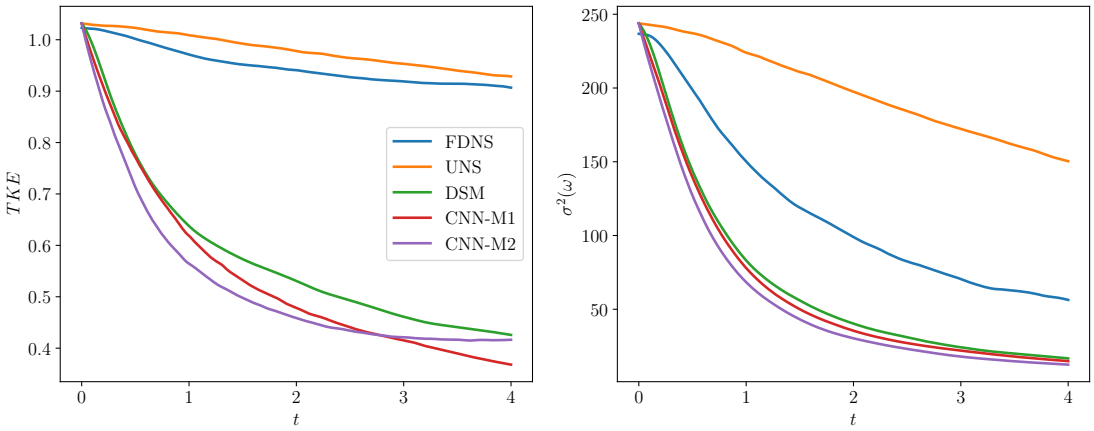

Figure 2.22: Time evolution of turbulent kinetic energy (left) and vorticity variance (right) for Re = 16000 at grid resolution $N^2 = 128^2$. This Reynolds number is used for training the CNN.

The proposed CNN frameworks were able to produce the vorticity field and other statistical quantities similar to the dynamic Smagorinksy model. If we compare the computational time of the DSM, the proposed CNN frameworks are computationally faster. The CPU time for the coarse-grained simulation at 128^2 with the DSM takes around 220 seconds. On the other hand, the coarse-grained simulation at 128^2 with CNN models takes around 150 seconds. Therefore, data-driven frameworks have an advantage in terms of computational overhead compared to traditional closure models. Our proposed framework can also be considered as a hybrid modeling approach, where the bulk nature of the flow evolution is retained though governing laws (vorticity-streamfunction formulation in our test case) and data-driven methods act as a corrector for statistical fidelity.

2.7 Conclusion

In the present study, we investigated different data-driven turbulence closure frameworks to learn SGS stresses using coarse-grained field variables. The blending of data-driven turbulence closure models within a physics-based LES framework presents the hybrid modeling approach that has the potential to give an accurate prediction of fluid flows at a less computational cost. The traditional Smagorinsky model is based on empirical formulas and phenomenological relationships and can either produce insufficient or excessive dissipation. On the other hand, the optimal map between coarse-grid field variables and SGS stresses learned by a data-driven framework provides improved

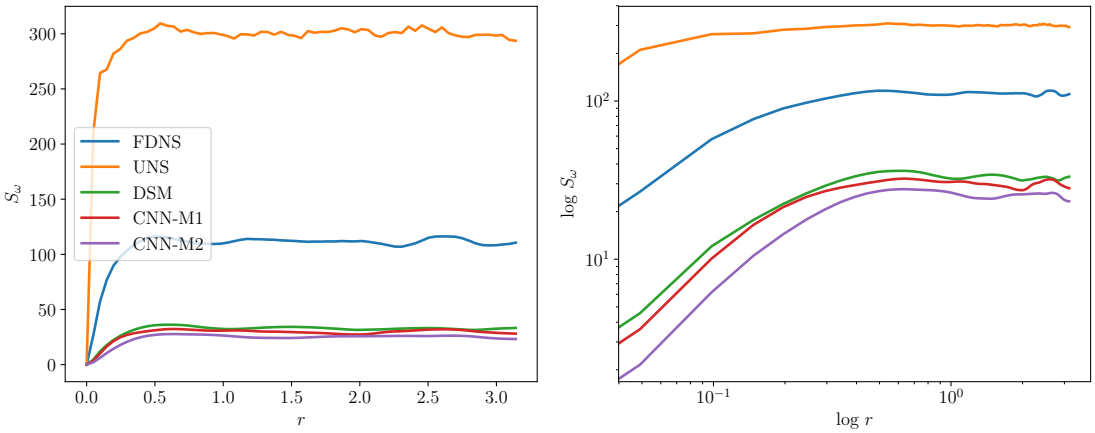

Figure 2.23: *A posteriori* vorticity structure functions plotted against $\mathbf{r}$ (left) and $\log(\mathbf{r})$ (right) for Re = 16000 . This Reynolds number is used for training the CNN.

prediction compared to the dynamic Smagorinsky model. The importance of the selection of input features in the prediction of SGS stresses is illustrated for different data-driven closure models using two-dimensional Kraichnan turbulence as the prototype example. The quantitative analysis using cross-correlation indicates that the prediction of SGS stresses improves when coarse-grid velocities, velocity gradients, and their Laplacian are included in input features. The analysis with localized mapping showed that the improvement in the prediction of SGS stresses is achieved when information from neighboring points is also included without any significant increase in the training and deployment time. The CNN mapping provides the most accurate prediction close to true SGS stresses with less computational overhead for training because of their invariance and weight sharing property.

The analysis of deployment time for different frameworks points out that data-driven closure models can give accurate SGS stresses prediction with the same or less computational overhead as the dynamic Smagorisnky model. The localized point-to-point mapping with ANN is particularly attractive for practical engineering applications due to its ability to handle unstructured mesh. The CNN mapping, on the other hand, seems more suitable for applications where a large amount of training data is available in the form of snapshots. While intelligent SGS modeling frameworks can model true SGS stresses accurately, they are prone to numerically unstable prediction in the *a posteriori* deployment as shown in recent studies [238, 28]. To exploit the potential of these black-box models in safety critical applications, we further investigate their robustness in predicting the eddy viscosity coefficient.

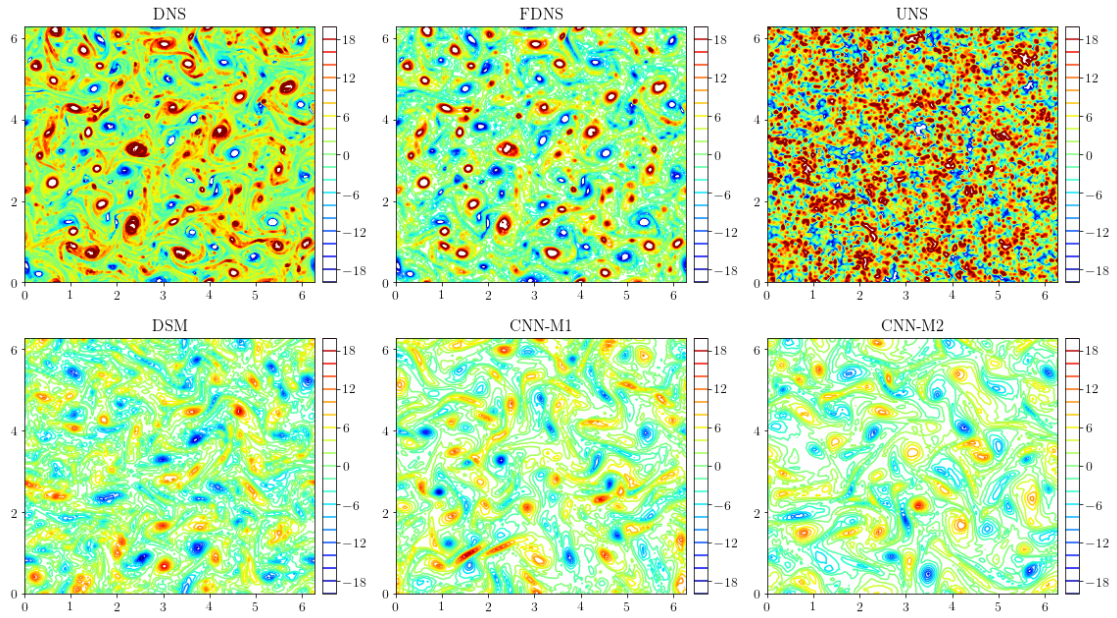

Figure 2.24: *A posteriori* Vorticity field at final time $t = 4.0$ for Re $= 32000$ for proposed CNN frameworks. We also provide vorticity field for DNS, filtered DNS (FDNS), and unresolved numerical simulations (UNS) for comparison.

Although limited in their predictive accuracy by utilizing an eddy viscosity model, we illustrate that the intelligent eddy viscosity approach gives four to eight times computational speedup with the same accuracy as the DSM.

We analyze the performance of CNN based data-driven SGS closure models in the *a posteriori* deployment settings with vorticity-streamfunction formulation for two-dimensional Kraichnan turbulence. To ensure numerical realizability during deployment, we enforce positive viscosity constraint and inject SGS closure terms at only those points where the eddy viscosity is positive. Even with this constraint, proposed CNN models cans predict both positive and negative values of SGS closure term corresponding to vorticity forcing or damping, respectively. Our statistical analysis of different quantities suggests that the proposed CNN frameworks are able to produce a vorticity field similar to the dynamic Smagorinsky model at less computational overhead. Enforcing numerical realizability constraints during the deployment suggests that *a posteriori* analysis should be taken into account when selecting optimal hyperparameters of the neural network. One of the ways to take the numerical stability constraint during deployment into account for optimal neural network architecture selection is through custom loss function during training. Specifically, we can embed numerical realizability constraint as a form of regularization to the original loss function

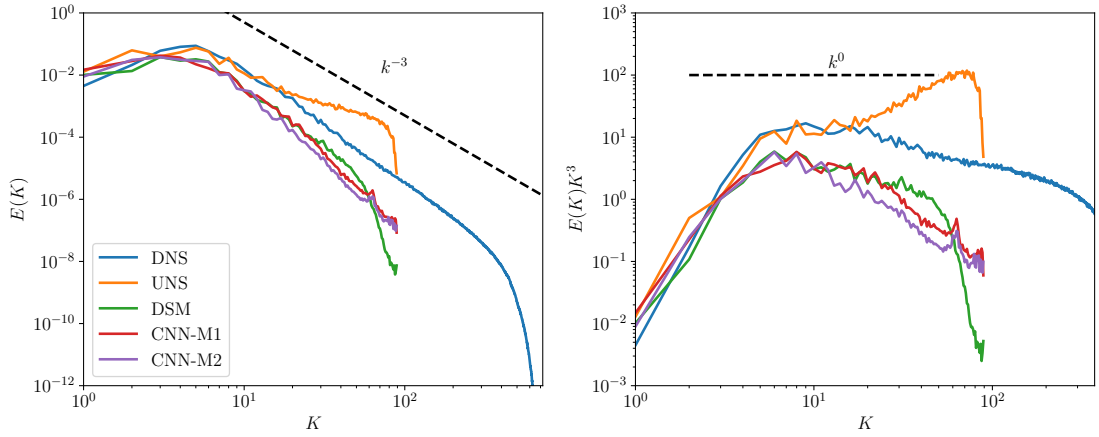

Figure 2.25: *A posteriori* kinetic-energy spectra (left) and compensated kinetic-energy spectra (right) for Re = 32000 at $t = 4$ and at grid resolution $N^2 = 128^2$. This Reynolds number is not utilized for training and it demonstrates the effectiveness of the CNN on test data.

(mean squared error in our case) during *a priori* training of the neural network.

We highlight that the data-driven techniques are in their infancy. Once the trained model is deployed in a CFD code, *a posteriori* analysis of data-driven closure models might give unexpected predictions (i.e., numerically or physically inconsistent). To be able to use the neural network based model in safety-critical applications, we need either of the two: interpret the model and figure out when it can fail, or to use neural networks in a way that we can detect when it fails and produces nonphysical results. Interpreting a deep neural network with millions of parameters is almost impossible. In our future work, we will focus on the second approach with an internal sanity checking mechanism where a black box model helps better modeling of conservation laws, and conservation mechanism puts a sanity check on the black-box model. Furthermore, the neural-network architectures employed in this work are fairly simple plain vanilla versions without any complex structure. The predictive performance of data-driven closure models can be further improved by constructing more sophisticated architecture designs like TBNN [213], and generative adversarial networks [123]. In the future, we would also like to extend these approaches for more complex test cases such as three-dimensional Kolmogorov turbulence, and geophysical flows. Some of the frameworks investigated in this study, especially with ANN can be readily extended to 3D turbulent flows. For the CNN framework, the convolutional filter is very important for an accurate prediction of turbulent flows. The discretization for 3D turbulent flows is usually non-uniform. For example, in the case of the channel flow, the mesh

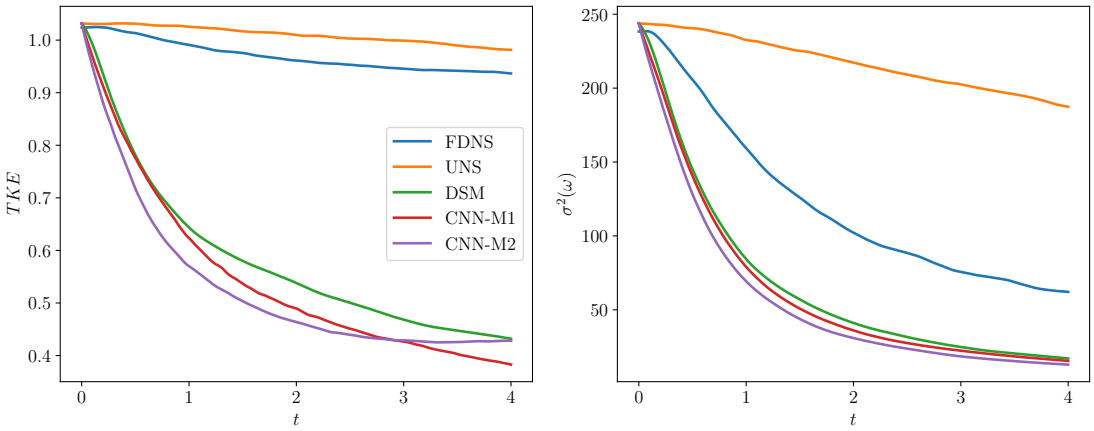

Figure 2.26: Time evolution of turbulent kinetic energy (left) and vorticity variance (right) for Re = 16000 at grid resolution $N^2 = 128^2$. This Reynolds number is not utilized for training and it demonstrates the effectiveness of the CNN on test data.

is clustered near the wall than away from the wall. For such flows, we might design special convolutional filters in different regions of the flow.

2.A Derivation of the Smagorinsky Model in 2D Turbulence

From Equation 2.5, the subgrid-scale stresses in 2D field can be written as

$$\tau_{ij} = \overline{u_i u_j} - \bar{u}_i \bar{u}_j, \tag{2.48}$$

$$= \underbrace{\frac{1}{2}\tau_{kk}\delta_{ij}}_{k_{\mathrm{SGS}}\delta_{ij}} + \Bigg(\underbrace{\tau_{ij} - \frac{1}{2}\tau_{kk}\delta_{ij}}_{\tau^d_{ij}}\Bigg). \tag{2.49}$$

The SGS stresses can be written as

$$\tau = k_{\mathrm{SGS}} I + \tau^d, \tag{2.50}$$

where $k_{\mathrm{SGS}} = \frac{1}{2}\tau_{kk}$ is called subgrid-scale kinetic energy (i.e., using the conventional summation notation with repeating indices, for example, $\tau_{kk} = \tau_{11} + \tau_{22}$, in 2D). In Smagorinsky model, we model the deviotoric (traceless) part of SGS stresses as

$$\tau^d_{ij} = -2\nu_e \bar{S}^d_{ij}, \tag{2.51}$$

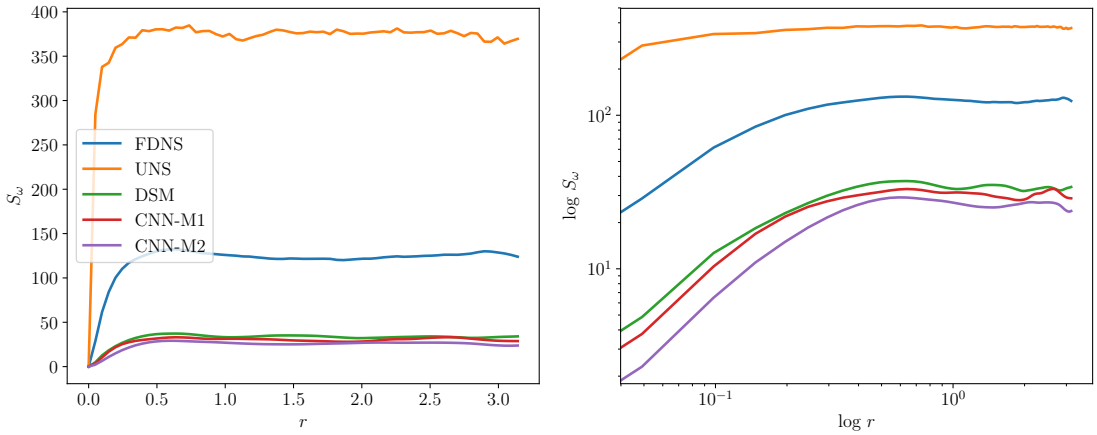

Figure 2.27: *A posteriori* vorticity structure functions plotted against $\mathbf{r}$ (left) and $\log(\mathbf{r})$ (right) Re = 32000 . This Reynolds number is not utilized for training and it demonstrates the effectiveness of the CNN on test data.

where ν_e is the SGS eddy viscosity, and $\bar{S}_{ij}$ is called resolved strain rate tensor given by

$$\bar{S}_{ij} = \frac{1}{2}\left(\frac{\partial \bar{u}_i}{\partial x_j} + \frac{\partial \bar{u}_j}{\partial x_i}\right), \tag{2.52}$$

where we can write explicity as follows

$$\bar{S} = \begin{bmatrix} \frac{\partial \bar{u}}{\partial x} & \frac{1}{2}\left(\frac{\partial \bar{u}}{\partial y} + \frac{\partial \bar{v}}{\partial x}\right) \\ \frac{1}{2}\left(\frac{\partial \bar{v}}{\partial x} + \frac{\partial \bar{u}}{\partial y}\right) & \frac{\partial \bar{v}}{\partial y} \end{bmatrix}. \tag{2.53}$$

The trace of the $\bar{S}$ is zero owing to the continuity equation for incompressible flows. Therefore, $\bar{S}^d_{ij} = \bar{S}_{ij}$ and the Smagorinsky model becomes

$$\tau^d_{ij} = -2\nu_e \bar{S}_{ij}. \tag{2.54}$$

The eddy viscosity approximation computes ν_e using the below relation

$$\nu_e = C_k \Delta \sqrt{k_{\text{SGS}}}, \tag{2.55}$$

where the proportionality constant is often set to $C_k = 0.094$, and Δ is the length scale (usually grid size). The SGS kinetic energy k_{SGS} is computed with the local equilibrium assumption of the balance between subgrid scale energy production and

dissipation

$$\bar{S} : \tau + C_\epsilon \frac{k_{\text{SGS}}^{1.5}}{\Delta} = 0, \tag{2.56}$$

where the first term in above equation is dissipation flux, second term is production flux, and the production constant is often set to $C_\epsilon = 1.048$. The double inner product operation : is given by

$$\bar{S} : \tau = \bar{S}_{ij}\tau_{ij} = \bar{S}_{11}\tau_{11} + \bar{S}_{12}\tau_{12} + \bar{S}_{21}\tau_{21} + \bar{S}_{22}\tau_{22}. \tag{2.57}$$

Substituting Equation 2.50 and 2.54 int Equation 2.56, we get

$$\bar{S} : (k_{\text{SGS}} I - 2C_k \Delta \sqrt{k_{\text{SGS}}} \bar{S}) + C_\epsilon \frac{k_{\text{SGS}}^{1.5}}{\Delta} = 0, \tag{2.58}$$

$$\sqrt{k_{\text{SGS}}} \left(\frac{C_\epsilon}{\Delta} k_{\text{SGS}} + \sqrt{k_{\text{SGS}}} \underbrace{\bar{S} : I}_{\bar{S}_{ij}\delta_{ij}=0} - 2C_k \Delta \bar{S} : \bar{S} \right) = 0, \tag{2.59}$$

$$\frac{C_\epsilon}{\Delta} k_{\text{SGS}} - 2C_k \Delta \bar{S} : \bar{S} = 0, \tag{2.60}$$

From above equations, subgrid-scale kinetic energy can be written as

$$k_{\text{SGS}} = \frac{C_k}{C_\epsilon} \Delta^2 (2\bar{S} : \bar{S}), \tag{2.61}$$

$$k_{\text{SGS}} = \frac{C_k}{C_\epsilon} \Delta^2 |\bar{S}|^2, \tag{2.62}$$

where $|\bar{S}| = \sqrt{2\bar{S}_{ij}\bar{S}_{ij}}$. Furthermore, substituting Equation 2.55 in above Equation, we get

$$\nu_e = C_k \Delta^2 \sqrt{\frac{C_k}{C_\epsilon}} |\bar{S}|. \tag{2.63}$$

We can define a new constant coefficient as

$$C_s^2 = C_k \sqrt{\frac{C_k}{C_\epsilon}}. \tag{2.64}$$

where $C_s = 0.1678$ is called the Smagorinsky coefficient. Finally, we get below expression for SGS eddy viscosity

$$\nu_e = C_s^2 \Delta^2 |\bar{S}|, \tag{2.65}$$

and the Smagorinsky model, given by Equation 2.51, reads as

$$\tau_{ij}^{d} = -2C_s^2\Delta^2|\bar{S}|\bar{S}_{ij}. \tag{2.66}$$

2.B Hyperparameters Optimization

In the appendix, we outline the procedure we followed for selection of hyperparameters for ANN with point-to-point mapping and neighboring stencil mapping. For ANN, there are many hyperparameters such as number of neurons, number of hidden layers, loss function, optimization algorithm, activation function, and batch size etc. If we use regularization, dropout, or weight decay to avoid overfitting, the design space of hyperparameters increases further.

We focus on three main hyperparameters of ANN: number of neurons, number of hidden layers, and learning rate of optimization algorithm. The training data is scaled between [-1,1] using the minimum and maximum value in the training dataset. We use ReLU activation function given by $\zeta(\chi) = \max(0, \chi)$, where ζ is the activation function, and χ is the input to the node. We use Adam optimization algorithm [175] and the batch size is kept constant at 256. Adam optimization algorithm has three hyperparameters: learning rate α, first moment decay rate β_1, and second moment decay rate β_2. We test our ANN for two learning rates $\alpha = 0.001$, and 0.0001. The other two hyperparameters in Adam optimization algorithm are $\beta_1 = 0.9$, and $\beta_2 = 0.999$. We employ mean squared error as the loss functions, since it is a regression problem. We test both ANN with point-to-point mapping and neighboring stencil mapping for four different number of hidden layers $L = 2, 3, 5, 7$. The ANN with point-to-point mapping is tested for four different number of neurons $N = 20, 30, 40, 50$ and the local-stencil-mapping is tested for $N = 40, 60, 80, 100$. The number of neurons is higher in case of local-stencil-mapping because there are more features compared to point-to-point mapping.

The optimal ANN architecture is selected using multi-dimensional gridsearch algorithm coupled with k-fold cross-validation. Cross-validation is a procedure used to determine the performance of the neural network on unseen data. The procedure consists of dividing the training data into k groups, training the ANN by excluding each group and evaluating the model's performance on that group. Therefore, if we use five-fold cross-validation then the model is trained five times and the performance index is computed for five groups. Once the performance for each group is available,

the mean of the performance index is utilized to select optimal hyperparameters. We use 500 epochs for determining the optimal hyperparameters. A good learning is achieved when both training loss and validation loss reduce till the learning rate is minimal. We apply coefficient of determination r^2 as the performance index to decide optimal hyperparameters. The calculation of coefficient of determination is done using below formula

$$r^2 = 1 - \frac{\sum_i (y_i - \tilde{y}_i)^2}{\sum_i (y_i - \bar{y})^2}, \tag{2.67}$$

where y_i is the true label, $\tilde{y}$ is the predicated label, and $\bar{y}$ is the mean of true labels.

Figure 2.28 displays the performance index for ANN with point-to-point mapping and $\mathbb{M}3$ model for all hyperparameters tested using gridsearch algorithm. It can be observed that the performance of the network does not change significantly with hyperparameters and the difference in performance is very small. The optimal hyperparameters obtained for point-to-point mapping ANN are $L = 2$, $N = 40$, and $\alpha = 0.0001$. We use the same hyperparameters for other two models $\mathbb{M}1$ and $\mathbb{M}2$ for point-to-point mapping ANN. We see the similar behaviour in case of neighboring stencil mapping ANN and model $\mathbb{M}3$ as shown in Figure 2.29. The optimal hyperparameters for neighboring stencil mapping ANN are $L = 2$, $N = 40$, and $\alpha = 0.001$.

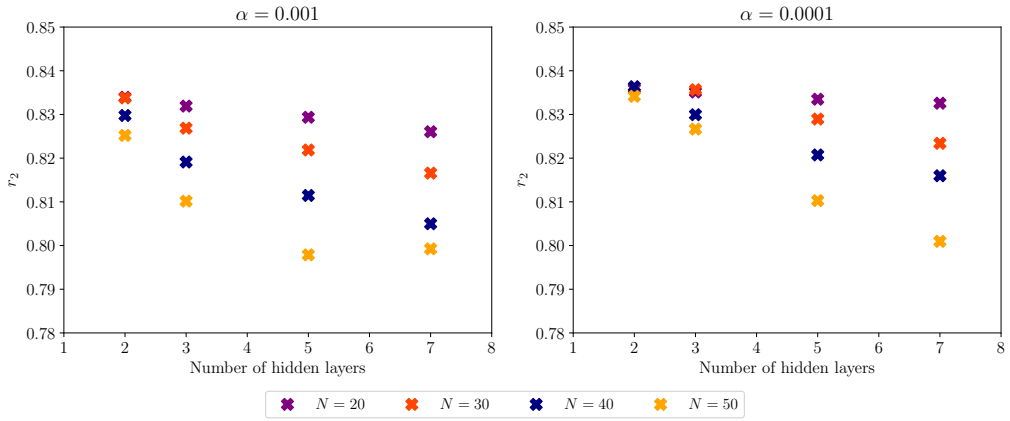

Figure 2.28: Hyperparameters search using the gridsearch algorithm combined with five-fold cross validation for the neural network using point-to-point mapping with $\mathbb{M}3$.

As discussed in Section 2.4.1, we get poor prediction between true and predicted stresses for point-to-point mapping with model $\mathbb{M}1$. Figure 2.30 shows the PDF of true and predicted streeses computed with different activation functions. It can be observed that the predicted stresses are almost same for all activation functions.

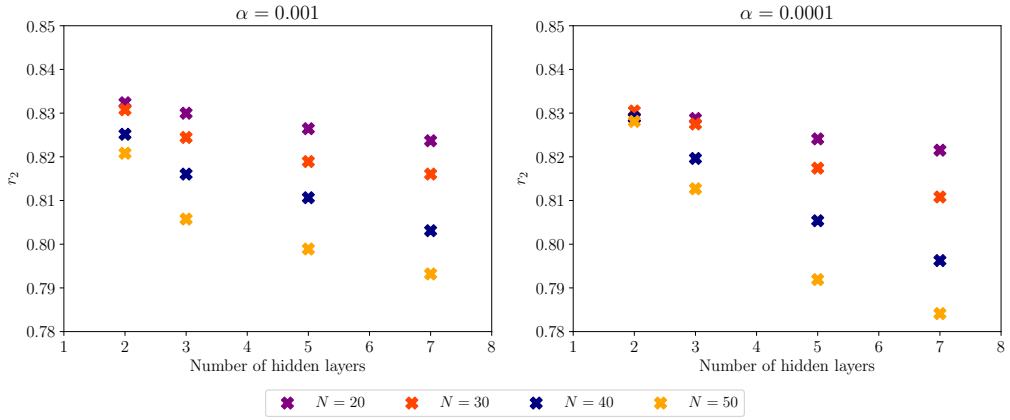

Figure 2.29: Hyperparameters search using the gridsearch algorithm combined with five-fold cross validation for the neural network using neighboring stencil mapping with $\mathbb{M}3$.

Therefore, we can conclude that we need additional input features such as velocity gradients to improve the prediction with point-to-point mapping.

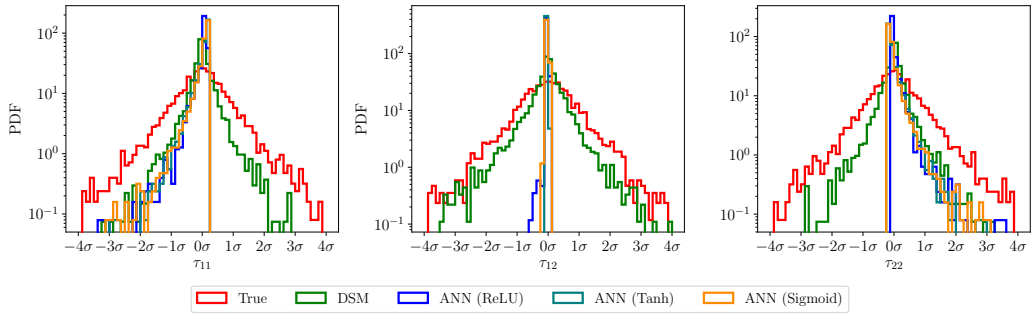

Figure 2.30: Probability density function for SGS stress distribution with point-to-point mapping. The ANN is trained using $\mathbb{M}1$ with different activation functions. The training set consists of 70 time snapshots from time $t = 0.0$ to $t = 3.5$ and the model is tested for $400th$ snapshot at $t = 4.0$.

The CNN architecture has similar hyperparameters as the ANN. Additionally, we need to select the kernel shape and strides for CNNs. Stride is the amount by which the kernel should shift as it convolves around the volume. We use the stride=1 in both x and y directions. We use 3×3 shaped kernel in our CNN architecture. We check the performance of CNN architecture for different number of hidden layers $L = 2, 4, 6, 8$, different number of filters $N = 8, 16, 24, 32$, and two learning rates. Figure 2.31 displays the performance index of CNN for different hyperparameters. The performance of CNN is more sensitive to the learning rate and we observe stable performance for the learning rate $\alpha = 0.001$. The performance is almost similar for

$L = 6, 8, 10$ with different number of kernels. We can select $L = 6$ and $N = 16$, which has performance index of 0.76. Additionally, we test the CNN architecture with $L = 6$ and $[16, 8, 8, 8, 8, 16]$ distribution for the number of kernels along hidden layers and we observed the performance index of 0.75 at less computational cost. Therefore, we apply $L = 6$, $N = [16, 8, 8, 8, 8, 16]$, and $\alpha = 0.001$ as our hyperparameters for the CNN architecture.

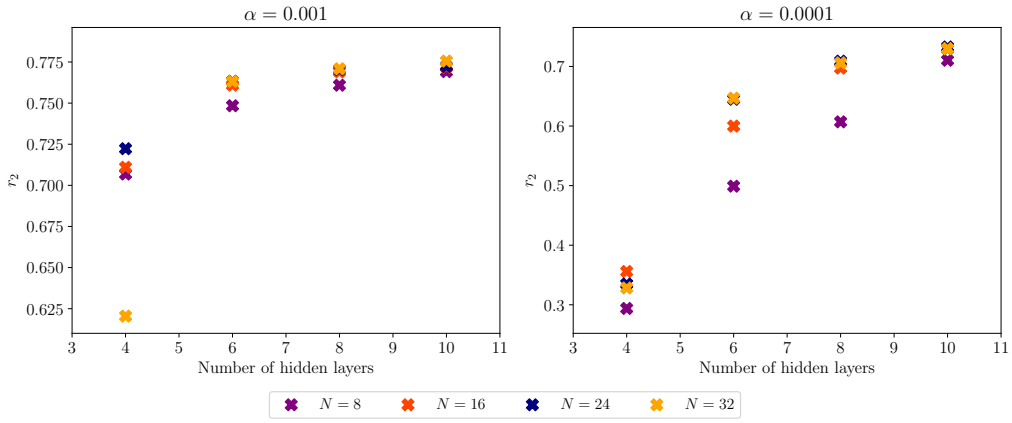

Figure 2.31: Hyperparameters search using the gridsearch algorithm combined with five-fold cross validation for CNN mapping with model $\mathbb{M}3$.

2.C CPU Time Measurements

In this study, the pseudo-spectral solver used for DNS is written in Python programming language. The code for coarsening of variables from fine to coarse grid, dynamic Smagorinsky model code is all written in Python. We use vectorization to get faster computational performance. The machine learning library Keras is also available in Python and is used for developing all data-driven closure models. Therefore, the CPU time reported in our analysis is for codes, which are all developed on the same platform. We would like to highlight that when the trained model is deployed, it makes the function for first time and hence it takes slightly more time. Once the function is created, the CPU time for deployment is less. Therefore, in all our Tables, we report the CPU time for running the predict function second time since initializing CUDA kernels might yield a startup overhead as shown in Listing 2.1, where t1 here has some idle time due to initializing kernels. In our study, we report t2, and we further verified that t3 - t2 = t2, which illustrate that the reported CPU times are consistent.

```
test_time_init = tm.time()
y_test = model.predict(ftest)
t1 = tm.time() - test_time_init

test_time_init = tm.time()
y_test = model.predict(ftest)
t2 = tm.time() - test_time_init

test_time_init = tm.time()
y_test = model.predict(ftest)
y_test = model.predict(ftest)
t3 = tm.time() - test_time_init
```

Listing 2.1: Code sample to check the CPU time for data-driven models.

2.D ANN and CNN Architectures

We use open-source Keras library to build our neural networks. It uses TensorFlow at the backend. Keras is widely used for fast prototyping, advanced research and production due to its simplicity and faster learning rate. Keras library provides different options for optimizers, neural network architectures, activation functions, regularization, dropuout, etc. Any simple neural network architecture can be coded with few lines of code. The sample code for ANN and CNN used in this work are listed in Listings 2.2 and 2.3.

```
model = Sequential()

input_layer = Input(shape=(nf,))
x = Dense(40, activation='relu',  use_bias=True)(input_layer)
x = Dense(40, activation='relu',  use_bias=True)(x)
output_layer = Dense(nl, activation='linear', use_bias=True)(x)
model = Model(input_layer, output_layer)
adam = optimizers.Adam(lr=lr, beta_1=0.9, beta_2=0.999, epsilon=None
    , decay=0.0, amsgrad=False)
model.compile(loss='mse', optimizer=adam, metrics=[cod])
```

Listing 2.2: Sample code for the ANN used in this study.

```
inputf = Input(shape=(nx,ny,nci))

x = Conv2D(16, (3, 3), activation='relu', padding='same')(inputf)
x = Conv2D(8, (3, 3), activation='relu', padding='same')(x)
x = Conv2D(8, (3, 3), activation='relu', padding='same')(x)
x = Conv2D(8, (3, 3), activation='relu', padding='same')(x)
x = Conv2D(8, (3, 3), activation='relu', padding='same')(x)
x = Conv2D(16, (3, 3), activation='relu', padding='same')(x)

output = Conv2D(nco, (3, 3), activation='linear', padding='same')(x)
model = Model(inputf, output)
adam = optimizers.Adam(lr=lr, beta_1=0.9, beta_2=0.999, epsilon=None
    , decay=0.0, amsgrad=False)
model.compile(loss='mse', optimizer=adam, metrics=[cod])
```

Listing 2.3: Sample code for the CNN used in this study.

CHAPTER III

Data assimilation Empowered Deep Learning for Subgrid Processes in Geophysical Flows

The contents of this chapter has been published in Physical Review Fluids (PRF)[1].

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 a set of applications of the multiscale Lorenz 96 system for which the parameterization model for unresolved scales is exactly known, and the two-dimensional Kraichnan turbulence system for which the parameterization model for unresolved scales is not known a priori. 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.

[1]Pawar, S., & San, O. (2021). Data assimilation empowered neural network parameterizations for subgrid processes in geophysical flows. Physical Review Fluids, 6(5), 050501.

3.1 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 $O(10$ km$)$ to $O(100$ 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 [359, 84, 311]. 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 [343, 82, 143].

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 [156, 157]. The nonlinear and multiscale nature of geophysical flows makes this tuning procedure cumbersome and can impede accurate climate prediction [424]. A recent development in machine learning, particularly deep learning [198], along with the huge volume of data gathered from high-resolution numerical simulations [262] and remote sensors measurements [187] offers an alternative to the physics-based parameterization schemes and can pave the way for improved climate and weather models. Deep learning approaches have been demonstrated to be successful for different scientific tasks in Earth system science, such as extreme weather pattern detection [219], precipitation nowcasting [346], transport process modeling [77], and many more. Deep learning has also been utilized to represent subgrid scale processes in climate models. Rasp et al. [316] trained a deep neural network (DNN) to emulate a cloud resolving model and formulated a procedure to produce stable results in the online deployments close to the original super-parameterized global circulation model. Gentine et al. [117] used an ensemble of random forests as a machine learning (ML) algorithm to parameterize the moist convection and implemented it in a global circulation model. They demonstrated the stable and robust performance of ML based parameterization in capturing important climate statistics including precipitation extremes.

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 [50] 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, [46] 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 [87, 340, 112, 239]. Ling et al. [213] proposed a novel neural network architecture with embedded Galilean invariance for the prediction of Reynolds stress anisotropy tensor. Wang et al. [388] employed random forest as an ML algorithm to reconstruct the discrepancy in 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, subgrid scale closure modeling of Kraichnan turbulence [238], decaying homogeneous isotropic turbulence [28], forced isotropic turbulence [414], compressible isotropic turbulence [413], and wall-bounded turbulence [354]. The feasibility of deep learning has been investigated to produce a predictive model for turbulent fluxes, such as heat fluxes [173] and anomalous fluxes in drift-wave turbulence [136]. In a recent work, Novati et al. [267] 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 very successful for challenging problems such as super-resolution of turbulent flows [108, 158, 364], data-driven modeling of chaotic systems [281, 383, 387], reduced order modeling of high-dimensional multiphysics systems [201, 247, 303, 307], and developing forecast models for complex physical systems [389, 14, 66].

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 [98, 385]. 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 [318].

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 [309, 364, 308, 405, 94]. Another way is to change the structure of neural network architecture to enforce physical conservation laws as hard constraints [249, 227]. 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 [318, 77, 166]. 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 [316, 314]. Similarly, Brenowitz et al. [48] found that the nonphysical correlations learned by neural networks were the cause of instabilities in their online deployment within the global circulation model [47] and developed an approach to ensure stability. Wu et al. [407] 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 [112, 388]. Wu et al. [407] 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 [206, 348, 97, 411, 423, 17] and can be applied to achieve accurate prediction in hybrid models that employ data-driven model as a sub-model 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 [191, 254, 74, 72]. 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 a 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 [209, 207, 370, 38, 44] and we demonstrate this through our study.

In this work, we propose a neural network closure framework in developing hybrid physics-ML models supplemented with DA for multiscale systems. In particular, we advocate the use of sequential DA techniques to improve the state estimate of the system by incorporating observations into a model equipped with neural network parameterization schemes for unresolved physics. To this end, we use real-time measurements to regularize ML empowered predictive tools through ensemble Kalman filter based approach. Our first example a two-level Lorenz 96 model [222] 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 [196, 163, 137, 378]. 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. We further extend our framework to Kraichnan turbulence [181], where it is shown that the DA improves the state estimate of the hybrid physics-ML model and this leads to better prediction for statistical properties like kinetic energy spectra and vorticity structure functions in comparison with high-fidelity direct numerical simulation (DNS). 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.

This chapter is structured as follows. In Section 3.2, we discuss the problem of parameterizations using a two-level Lorenz 96 model and Kraichnan turbulence as a prototypical examples. Section 3.3 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 sequential data assimilation and ensemble Kalman filter based algorithms in Section 3.4. In Section 3.5, we discuss the findings of our numerical experiments with a two-level Lorenz 96 model and Kraichnan turbulence. Finally, we conclude with the summary and direction for future work in Section 3.6.

3.2 Parameterizations in Multiscale Systems

3.2.1 Two-level Lorenz 96 model

In this section, we describe the two-level variant of the Lorenz 96 model proposed by Lorenz [222]. This model has been extensively investigated to study stochastic parameterization schemes[277, 403, 73], scale-adaptive parameterizations[381], and neural network parameterizations[314, 402]. The two-level Lorenz 96 model can be written as

$$\frac{dX_i}{dt} = -X_{i-1}(X_{i-2} - X_{i+1}) - X_i - \frac{hc}{b}\sum_{j=1}^{J} Y_{j,i} + F, \tag{3.1}$$

$$\frac{dY_{j,i}}{dt} = -cbY_{j+1,i}(Y_{j+2,i} - Y_{j-1,i}) - cY_{j,i} + \frac{hc}{b}X_i, \tag{3.2}$$

where Equation 3.1 represents the evolution of slow, high-amplitude variables X_i ($i = 1, \ldots, n$), and Equation 3.2 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. The boundary conditions for the slow and fast variables are detained in Section 3.5 along with the generation of initial condition for the two-level Lorenz 96 system.

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 3.1 can be constructed with the parameterization for unresolved variables as follows

$$\frac{d\widetilde{X}_i}{dt} = -\widetilde{X}_{i-1}(\widetilde{X}_{i-2} - \widetilde{X}_{i+1}) - \widetilde{X}_i - \frac{hc}{b}G_i + F, \tag{3.3}$$

where the tilde is used to denote the fact that the parameterization G_i is used to represent the effect of unresolved variables. Typically, the parameterizations is a function of resolved variables and can be written mathematically as

$$\sum_{j=1}^{J} Y_{j,i} :\approx G_i = \mathbf{N}(\widetilde{\mathbf{X}}), \tag{3.4}$$

where $\mathbf{N}(\cdot)$ is the nonlinear mapping of resolved variables to the parameterizations at the i^{th} grid point. This mapping can be based on certain physical arguments or can also be learned with any data-driven methods. Therefore in parameterization research for multiscale systems, the underlying physical laws governing the dynamics of resolved variables are assumed to be known exactly, and the effect of unresolved variables is considered through parameterizations G_i. If we use data-driven methods to represent the parameterization G_i, then the forecast model given in Equation 3.3 can be considered as a hybrid model. Our main objective in this work is to improve the forecasting capability of multiscale systems that are represented by a hybrid model embedded with data-driven parameterizations and we achieve this through data assimilation techniques.

3.2.2 Kraichnan Turbulence

Here, we summarize the mathematical background of subgrid-scale parameterizations in the LES of two-dimensional turbulence. Even though two-dimensional turbulence cannot be realized in practice, it is extensively used for modeling geophysical flows in the atmosphere and ocean [42, 39]. The confinement of fluid turbulence to two spatial

dimensions leads to Kraichnan–Batchelor–Leith (KBL) theory of dual cascade with an inverse energy cascade to larger scales and direct enstrophy cascade to smaller scales. The non-dimensional vorticity transport equation for incompressible flows can be written as

$$\frac{\partial \omega}{\partial t} + J(\omega, \psi) = \frac{1}{\mathrm{Re}} \nabla^2 \omega, \tag{3.5}$$

$$J(\omega, \psi) = \frac{\partial \omega}{\partial x}\frac{\partial \psi}{\partial y} - \frac{\partial \omega}{\partial y}\frac{\partial \psi}{\partial x}, \tag{3.6}$$

where ω is the vorticity, ψ is the stremfunction, J is the Jacobian (or the nonlinear term), and Re is the Reynolds number of the flow. The vorticity and streamfunction are related to each other through the Poisson equation given by

$$\nabla^2 \psi = -\omega. \tag{3.7}$$

The governing equations for LES are obtained by applying a low-pass filtering operation, and the filtered vorticity transport equation can be written as

$$\frac{\partial \bar{\omega}}{\partial t} + \overline{J(\omega, \psi)} = \frac{1}{\mathrm{Re}} \nabla^2 \bar{\omega}. \tag{3.8}$$

The above equation can be rewritten as

$$\frac{\partial \bar{\omega}}{\partial t} + J(\bar{\omega}, \bar{\psi}) = \frac{1}{\mathrm{Re}} \nabla^2 \bar{\omega} + \Pi, \tag{3.9}$$

where the overbar quantities represent filtered variables and are evolved on a grid which is significantly coarse than required for the DNS. The effect of the unresolved scales due to truncation of high wavenumber flow scales is encompassed in a subgrid-scale source term Π and must be modeled. Mathematically, the true source term Π can be expressed as

$$\Pi = J(\bar{\omega}, \bar{\psi}) - \overline{J(\omega, \psi)}. \tag{3.10}$$

The approximation of subgrid processes plays an important role in determining the accuracy of large-scale flows and therefore the subgrid-scale parameterizations are critical to accurate LES simulations of geophysical flows [103]. Different models have been proposed in the literature for subgrid-scale parameterizations in geophysical flows [350, 204, 102, 90, 337, 234], and remains an active area of research due to complexity of the subgrid-scale closure modeling. In the present work, we put forth a data-driven

framework based on a neural network to predict the approximate value of the source term Π as a function of resolved flow variables on the coarser grid. One of the main advantages of data-driven closure modeling is that they are computationally faster than dynamic closure modeling procedure that involves several test filtering operations [340, 284, 276].

3.3 Neural Network Parameterizations

The parameterization problem in multiscale flows can be posed as a regression problem where the mapping between resolved scales and unresolved scales has to be determined. We consider supervised class of machine learning algorithms, where the optimal map between inputs and outputs is learned. In this section, we describe an artificial neural network (ANN) also called as multilayer perceptron, and convolutional neural network (CNN) to build data-driven parameterization models.

3.3.1 Artificial Neural Network

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

$$S^l = \mathbf{W}^l \mathcal{X}^{l-1}, \tag{3.11}$$

where $\mathcal{X}^{l-1}$ is the output of the $(l-1)^{\text{th}}$ layer, $\mathbf{W}^l$ 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

$$\mathcal{X}^l = \zeta(S^l + B^l), \tag{3.12}$$

where B^l is the vector of biasing parameters for the l^{th} layer and ζ 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 follows

$$\tilde{\mathcal{Y}} = \zeta_L(\mathbf{W}^L, B^L, \ldots, \zeta_2(\mathbf{W}^2, B^2, \zeta_1(\mathbf{W}^1, B^1, \mathcal{X}))), \tag{3.13}$$

where $\mathcal{X}$ and $\tilde{\mathcal{Y}}$ 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 $\mathbf{W}$ 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 [175] provide a rapid way to learn optimal weights. The training procedure for ANN can be summarized as:

- 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 $\tilde{\mathcal{Y}}$ whose true label is $\mathcal{Y}$.
- 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 [413, 417, 112]. Another method is to include the information at neighboring grid points to determine the output label at a single point [238, 284]. For a two-level Lorenz system, 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

$$\text{ANN-3} : \{X_{i-1}, X_i, X_{i+1}\} \in \mathbb{R}^3 \rightarrow \{\tilde{G}_i\} \in \mathbb{R}^1, \tag{3.14}$$

$$\text{ANN-5} : \{X_{i-2} \ldots, X_{i+2}\} \in \mathbb{R}^5 \rightarrow \{\tilde{G}_i\} \in \mathbb{R}^1, \tag{3.15}$$

$$\text{ANN-7} : \{X_{i-3} \ldots, X_{i+3}\} \in \mathbb{R}^7 \rightarrow \{\tilde{G}_i\} \in \mathbb{R}^1, \tag{3.16}$$

where $\tilde{G}_i$ is the predicted 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 3.1 and Equation 3.2 in a coupled manner. For all ANN architectures used in this study, we apply two hidden layers with 40 neurons and ReLU activation function for all hidden layers. For the output layer, the linear activation function is used. The ANN is trained using an Adam optimizer for 300 iterations.

3.3.2 Convolutional Neural Network

The convolutional neural network (CNN) is particularly attractive when the data is in the form of two-dimensional images [199]. 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

$$S^l_{ij} = \sum_{p=-\Delta_i/2}^{\Delta_i/2} \sum_{q=-\Delta_j/2}^{\Delta_j/2} \sum_{r=-\Delta_k/2}^{\Delta_k/2} \mathbf{W}^l_{pqr} \mathcal{X}^{l-1}_{i+p\ j+q\ k+r} + B_{pqr}, \tag{3.17}$$

where Δ_i, Δ_j, Δ_k are the sizes of filter in each direction, $\mathbf{W}^l_{pqr}$ are the entries of the filter for l^{th} Conv layer, B_{pqr} is the biasing parameter, and $\mathcal{X}^{l-1}_{ijk}$ is the input from $(l-1)^{\text{th}}$ 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

$$\mathcal{X}^l_{ijk} = \zeta(S^l_{ijk}), \tag{3.18}$$

where ζ 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 for learning parameterizations in a two-level Lorenz model can be mathematically presented as

$$\text{CNN} : \{X_1, \ldots, X_n\} \in \mathbb{R}^n \rightarrow \{\tilde{G}_1, \ldots \tilde{G}_n\} \in \mathbb{R}^n, \tag{3.19}$$

where X_i is the resolved variable and $\tilde{G}_i$ is the predicted 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 7×1 shape. We apply ReLU activation function for all hidden layers and the linear activation function for the output layer. Also, the zero-padding is used to keep the input and output shape the same. The CNN is trained with an Adam optimizer for 400 iterations. The hyperparameters of both ANN and CNN architectures were obtained through parametric study for a different number of neurons/filters and the number of hidden layers. 80% of the total data selected randomly was used for training and the remaining 20% of the data was used for validation. The selection of hyperparameters is done in such a way that the mean squared error between the actual and predicted parameterization drops

smoothly for both training and validation dataset so that overfitting is avoided and our model generalizes well to the unseen data. There are other methods like regularization, dropout, ensembling from different models, early stopping that can be adopted to prevent overfitting. An extensive hyperparameter search can be carried out for complex geophysical flows using neural architecture search packages like DeepHyper[229] and Tune[211].

For the two-dimensional turbulence, we learn the source term Π as a function of resolved flow variables, i.e., the vorticity ω, the streamfunction ψ, and two eddy-viscosity kernels as input features. The use of eddy-viscosity kernels as input features can be considered as a feature engineering step where certain important quantities are pre-computed and the neural network is presented with it as raw input features so as to facilitate the faster training and robust prediction. The CNN map for two-dimensional turbulence can be mathematically written as

$$\text{CNN} : \{\bar{\omega}, \bar{\psi}, |\bar{S}|, |\nabla\bar{\omega}|\} \rightarrow \{\tilde{\Pi}\}, \tag{3.20}$$

where $\tilde{\Pi}$ is the predicted source term, $|\bar{S}|$ is the Smagorinsky kernel, and $|\nabla\bar{\omega}|$ is the Leith kernel. The Smagorinsky and Leith kernels are computed as follow

$$|\bar{S}| = \sqrt{4\left(\frac{\partial^2\bar{\psi}}{\partial x\partial y}\right)^2 + \left(\frac{\partial^2\bar{\psi}}{\partial x^2} - \frac{\partial^2\bar{\psi}}{\partial y^2}\right)^2}, \tag{3.21}$$

$$|\nabla\bar{\omega}| = \sqrt{\left(\frac{\partial\bar{\omega}}{\partial x}\right)^2 + \left(\frac{\partial\bar{\omega}}{\partial y}\right)^2}. \tag{3.22}$$

The CNN architecture to learn the parameterization model for Kraichnan turbulence consist of 6 hidden layers and 16 filters in each hidden layers. The size of the filter in each hidden layer is 3×3 and the ReLU activation function is utilized for hidden layers. The training is performed for 800 epochs using the Adam optimizer. We note here that the predicted source term by the CNN is further post-processed during the deployment before it is injected into the vorticity transport equation to ensure numerical stability and we detail that procedure in Section 3.5.

3.4 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 [316, 316,

47, 48, 407]. From our numerical experiments with the two-level Lorenz system, 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 for the prediction of future state. 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[206, 348, 97, 115, 397] and here we discuss only sequential data assimilation problem and then outline the algorithm procedure for perturbed observations ensemble Kalman filter (EnKF), and the deterministic ensemble Kalman filter (DEnKF).

We consider the dynamical system whose evolution can be represented as

$$\mathbf{x}_{k+1} = \mathbf{M}_{t_k \to t_{k+1}}(\mathbf{x}_k) + \mathbf{w}_{k+1}, \tag{3.23}$$

where $\mathbf{x}_k \in \mathbb{R}^n$ is the state of the dynamical system at discrete time t_k, $\mathbf{M} : \mathbb{R}^n \to \mathbb{R}^n$ is the nonlinear model operator that defines the evolution of the system. The term $\mathbf{w}_{k+1}$ 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 $\mathbf{z}_k \in \mathbb{R}^m$ be observations of the state vector obtained from noisy measurements and cane be written as

$$\mathbf{z}_k = \mathbf{h}(\mathbf{x}_k) + \mathbf{v}_k, \tag{3.24}$$

where $\mathbf{h}(\cdot)$ is a nonlinear function that maps $\mathbb{R}^n \to \mathbb{R}^m$, and $\mathbf{v}_k \in \mathbb{R}^m$ is the measurement noise. We assume that the measurement noise is a white Gaussian noise with zero mean and the covariance matrix $\mathbf{R}_k$, i.e., $\mathbf{v}_k \sim \mathcal{N}(0, \mathbf{R}_k)$. Additionally, the noise vectors $\mathbf{w}_k$ and $\mathbf{v}_k$ are assumed to be uncorrelated to each other at all time steps. The sequential data assimilation can be considered as a problem of estimating the state $\mathbf{x}_k$ of the system given the observations up to time t_k, i.e., $\mathbf{z}_1, \ldots, \mathbf{z}_k$. 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 $\widehat{\mathbf{x}}_k$ 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 $\mathbf{x}_k^f$.

The ensemble Kalman filter (EnKF) [54] follows the Monte Carlo approach to approximate the probability distribution in the Kalman filter equations [161]. We start by initializing the state of the system for different ensemble members as follows

$$\widehat{\mathbf{X}}_0(i) = \mathbf{m}_0 + \mathbf{y}_0(i), \quad i = 1 \dots N \tag{3.25}$$

where $\mathbf{y}_0(i) \sim \mathcal{N}(0, \mathbf{P}_0)$, $\mathbf{m}_0$ is some assumed mean state of the system, $\mathbf{P}_0$ is the initial covariance error matrix, and N is the number of ensemble members. The propagation of the state for each ensemble over the time interval $[t_k, t_{k+1}]$ can be written as

$$\mathbf{X}^f_{k+1}(i) = \mathbf{M}_{t_k \to t_{k+1}}(\widehat{\mathbf{X}}_k(i)) + \mathbf{w}_{k+1}. \tag{3.26}$$

The term $\mathbf{w}_{k+1}$ accounting for model imperfections is usually assumed to be Gaussian noise. In this study, we consider the model error by means of multiplicative inflation [9] and without loss of generality we set $\mathbf{w}_{k+1} = 0$. The prior state and the prior covariance matrix are approximated using the sample mean and error covariance matrix $\mathbf{P}^f_{k+1}$ as follows

$$\mathbf{x}^f_{k+1} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{X}^f_{k+1}(i), \tag{3.27}$$

$$\mathbf{E}^f_{k+1}(i) = \mathbf{X}^f_{k+1}(i) - \mathbf{x}^f_{k+1}, \tag{3.28}$$

$$\mathbf{P}^f_{k+1} = \frac{1}{N-1} \sum_{i=1}^{N} \mathbf{E}^f_{k+1}(i)(\mathbf{E}^f_{k+1}(i))^{\mathrm{T}}. \tag{3.29}$$

Once the observations are available at time t_{k+1}, we generate N realizations of perturbed observations as follows

$$\mathbf{Z}_{k+1}(i) = \mathbf{z}_{k+1} + \mathbf{v}_{k+1}(i), \tag{3.30}$$

where $\mathbf{v}_{k+1}(i) \sim \mathcal{N}(0, \mathbf{R}_{k+1})$. Each member of the forecast ensemble $\mathbf{X}^f_{k+1}(i)$ is analyzed using the Kalman filter formulae as shown below

$$\widehat{\mathbf{X}}_{k+1}(i) = \mathbf{X}^f_{k+1}(i) + \mathbf{K}_{k+1}[\mathbf{Z}_{k+1}(i) - \mathbf{h}(\mathbf{X}^f_{k+1}(i))], \tag{3.31}$$

$$\mathbf{K}_{k+1} = \mathbf{P}^f_{k+1}\mathbf{H}^{\mathrm{T}}_{k+1}[\mathbf{H}_{k+1}\mathbf{P}^f_{k+1}\mathbf{H}^{\mathrm{T}}_{k+1} + \mathbf{R}_{k+1}]^{-1}, \tag{3.32}$$

where $\mathbf{H} \in \mathbb{R}^{m \times n}$ is the Jacobian of observation function $\mathbf{h}(\cdot)$. The analysis state estimate at time t_{k+1} is computed using the sample mean of all ensemble members as

$$\widehat{\mathbf{x}}_{k+1} = \frac{1}{N} \sum_{i=1}^{N} \widehat{\mathbf{X}}_{k+1}(i). \tag{3.33}$$

In order to take model imperfections into account, all ensemble members are updated by applying inflation to all ensemble anomalies as follows

$$\widehat{\mathbf{X}}_{k+1}(i) \leftarrow \widehat{\mathbf{x}}_{k+1} + \lambda \cdot (\widehat{\mathbf{X}}_{k+1}(i) - \widehat{\mathbf{x}}_{k+1}), \tag{3.34}$$

where λ is the inflation factor. The inflation also helps to address the problem of covariance underestimation due to small number of ensembles [97]. The inflation factor can either be a scalar or it can be made space and time dependent to improve the filter performance [7, 18]. In this study, we use the constant value of the inflation factor over the entire space at all times.

As a variant of the low-rank sequential nonlinear filtering framework, we also utilize the deterministic EnKF (DEnKF) algorithm proposed by Sakov et al. [331] for the data assimilation. We start the DEnKF algorithm by initializing the state estimate for all ensemble members similar to the EnKF algorithm as given in Equation 3.25. The anomalies between the forecast estimate of all ensembles and its sample mean (calculated using Equation 3.27) is

$$\mathbf{A}^{f}_{k+1}(i) = \mathbf{X}^{f}_{k+1}(i) - \mathbf{x}^{f}_{k+1}. \tag{3.35}$$

Once the observations are available at time t_{k+1}, the forecast state estimate is assimilated using the Kalman filter analysis equation as follows

$$\widehat{\mathbf{x}}_{k+1} = \mathbf{x}^{f}_{k+1} + \mathbf{K}_{k+1}[\mathbf{z}_{k+1} - \mathbf{h}(\mathbf{x}^{f}_{k+1})]. \tag{3.36}$$

Here, the Kalman gain matrix is computed using its square root version (without storing or computing $\mathbf{P}^{f}_{k+1}$ explicitly) as follows

$$\mathbf{K}_{k+1} \quad = \quad \frac{\mathcal{A}^{f}_{k+1}(\mathbf{H}_{k+1}\mathcal{A}^{f}_{k+1})^{\mathrm{T}}}{N-1} \left[\frac{(\mathbf{H}_{k+1}\mathcal{A}^{f}_{k+1})(\mathbf{H}_{k+1}\mathcal{A}^{f}_{k+1})^{\mathrm{T}}}{N-1} \quad + \quad \mathbf{R}_{k+1} \right]^{-1}, \tag{3.37}$$

where $\mathbf{H} \in \mathbb{R}^{m \times n}$ is the Jacobian of the observation operator (i.e., $H_{kl} = \frac{\partial h_k}{\partial x_l}$), and

the matrix $\mathcal{A}^f_{k+1} \in \mathbb{R}^{n\times N}$ is concatenated as follows

$$\mathcal{A}^f_{k+1} = [\mathbf{A}^f_{k+1}(1), \mathbf{A}^f_{k+1}(2), \ldots, \mathbf{A}^f_{k+1}(N)]. \tag{3.38}$$

The anomalies for all ensemble members are then updated separately with half the Kalman gain as shown below

$$\widehat{\mathbf{A}}_{k+1}(i) = \mathbf{A}^f_{k+1}(i) - \frac{1}{2}\mathbf{K}_{k+1}\mathbf{H}_{k+1}\mathbf{A}^f_{k+1}(i). \tag{3.39}$$

The analysis state for all ensemble members is obtained by adding ensemble anomalies and can be written as

$$\widehat{\mathbf{X}}_{k+1}(i) = \widehat{\mathbf{x}}_{k+1} + \lambda \cdot \widehat{\mathbf{A}}_{k+1}(i), \tag{3.40}$$

where λ is the inflation factor. We validate our implementation of the DEnKF algorithm using the one-level Lorenz 96 model and is detailed in Appendix 3.A.

3.5 Numerical Experiments

In the following, we present the findings of our numerical experiments with a two-level Lorenz 96 model and Kraichnan turbulence.

3.5.1 Two-level Lorenz 96 model

In this subsection, 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., $X_{i-n} = X_{i+n} = X_i$. The fast variables are extended by letting $Y_{j,i-n} = Y_{j,i+n} = Y_{j,i}$, $Y_{j-J,i} = Y_{j,i-1}$, and $Y_{j+J,i} = Y_{j,i+1}$. 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 $i \in 1, 2, \ldots, n$. We perturb the equilibrium solution for the 18^{th} 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 3.1 and Equation 3.2 in a coupled manner up to time $t = 0$. With this initial condition (i.e., at $t = 0$), 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 $t = 10$ to $t = 20$. Therefore, there is no overlap between the data used for training and testing. Since the neural 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 3.3.1. Figure 3.1 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 3.1. 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 sequential 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 $\mathbf{R}_k$, i.e., $\mathbf{v}_k \sim \mathcal{N}(0, \mathbf{R}_k)$. We use $\mathbf{R}_k = \sigma_b^2 \mathbf{I}$, where σ_b is the standard deviation of measurement noise and is set at $\sigma_b = 1$. We assume that observations are sparse in space and are collected at every 10^{th} time step. The number of ensemble members used for all numerical experiments is $N = 30$. We present two levels of observation density in space for the DA. For the first case, we employ observations at $[X_4, X_8, \ldots, X_{36}] \in R^9$ for the assimilation. The second set of observations consists of 50% of the full state of the system, i.e., $[X_2, X_4, \ldots, X_{36}] \in R^{18}$. In Figure 3.2, 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 3.2 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

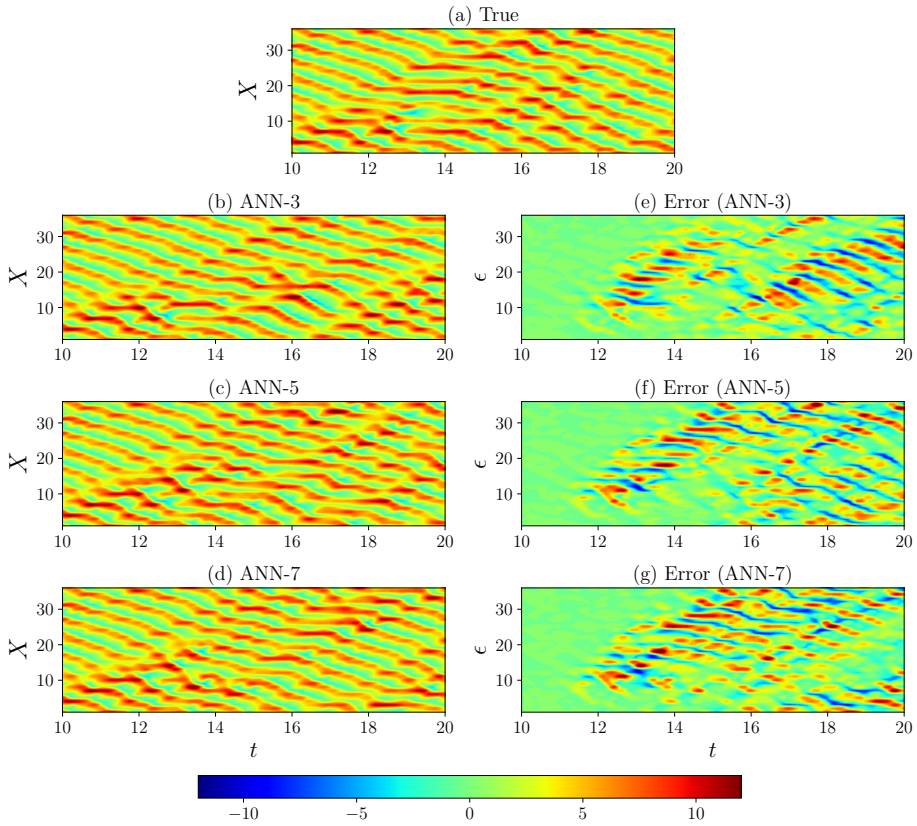

Figure 3.1: Full state trajectory of the multiscale Lorenz 96 model with the closure term computed using the different neighboring stencil mapping feedforward ANN architecture.

measurements coming from various types of sensor networks and can find applications in different fields like climate modeling, turbulence closure modeling where the subgrid scale parameterizations are unavoidable.

Figure 3.3 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 3.3, 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 3.2 and Figure 3.3, 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 reason for the inaccurate forecast can be attributed to

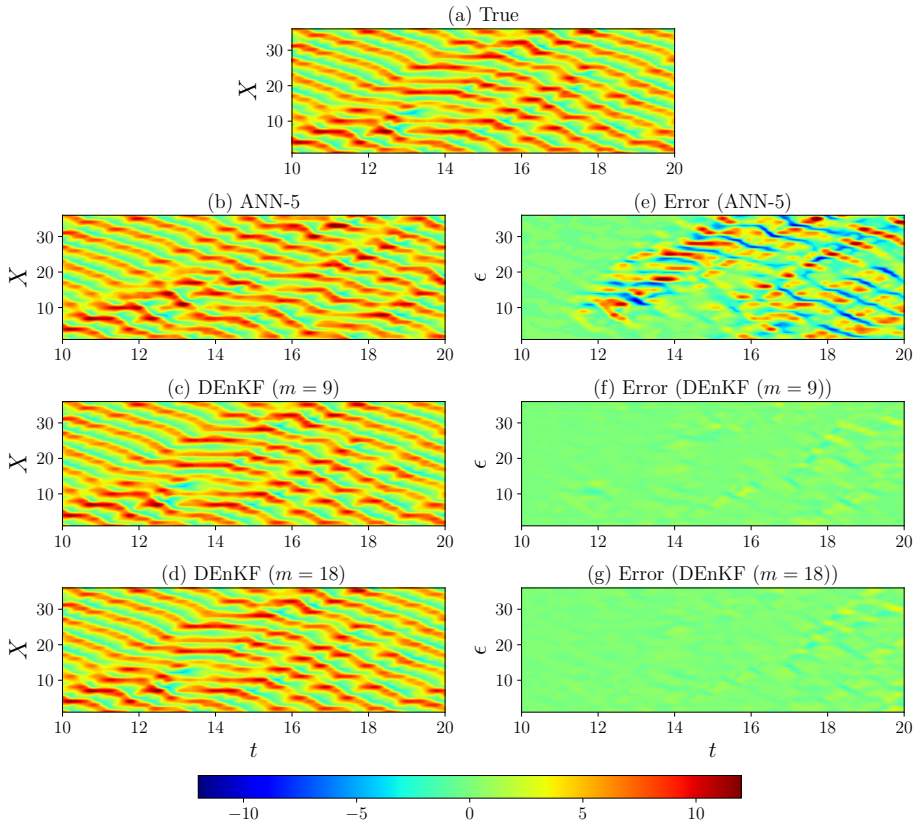

Figure 3.2: Full state trajectory of the multiscale Lorenz 96 model with the closure term computed using the five-point neighboring stencil mapping feedforward ANN architecture and the DEnKF used for data assimilation.

the uncertainty in the prediction of parameterizations by CNN. We highlight here that both ANN-5 and CNN architectures used in this study have similar number of trainable parameters. However, we see a better performance of the ANN-5 architecture over CNN due to a more number of training examples in the case of the ANN. For the ANN, every single point of the two-level Lorenz 96 system is one training example and therefore a single time snapshot of the training data leads to 36 samples for training. However, in the case of CNN, the total number of training samples is equal to the total number of time snapshots available for training. Therefore, we observe the better performance of the ANN-5 over CNN.

Another 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. To isolate the source of error, we integrate the forecast model for a two-level Lorenz 96 model without any parameterizations. The two-level Lorenz 96 model with no parameterizations is equivalent to setting

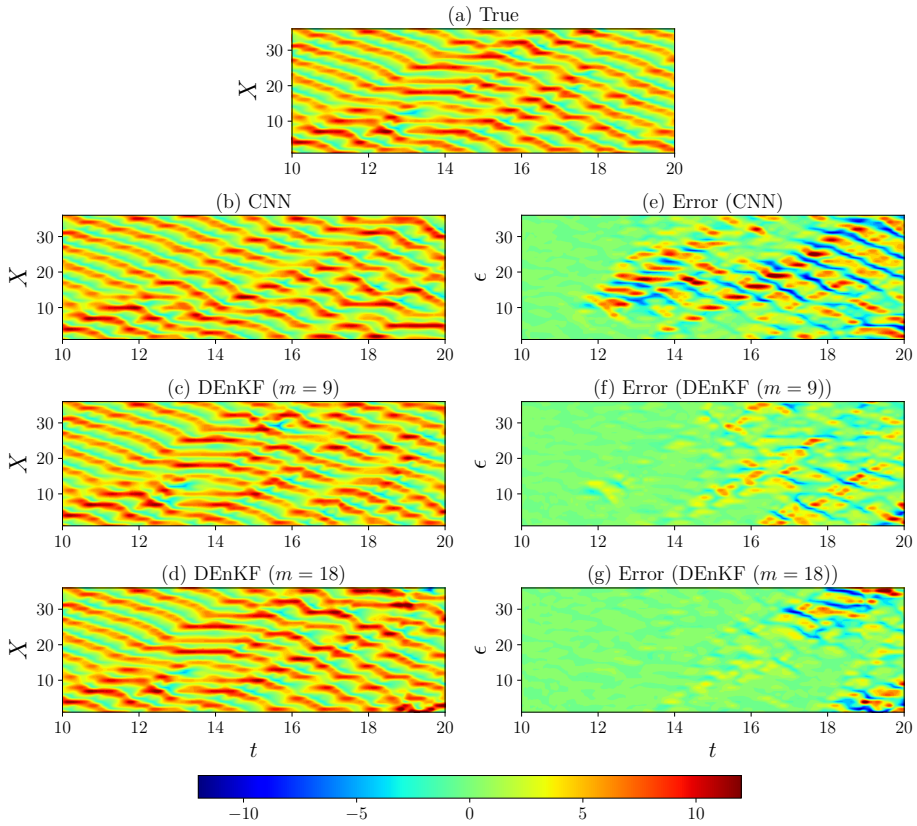

Figure 3.3: Full state trajectory of the multiscale Lorenz 96 model with the closure term computed using the CNN architecture and the DEnKF used for data assimilation

the coupling coefficient $h = 0$ in Equation 3.1 and it reduces to one-level Lorenz 96 model as presented in Equation 3.46. 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., $\frac{hc}{b}\sum_{j=1}^{J} Y_{j,i}$ term in Equation 3.1) can be considered as an added noise to the true state of the system for a one-level Lorenz 96 model presented in Equation 3.46.

In Figure 3.4, 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 10^{th} 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 3.1.

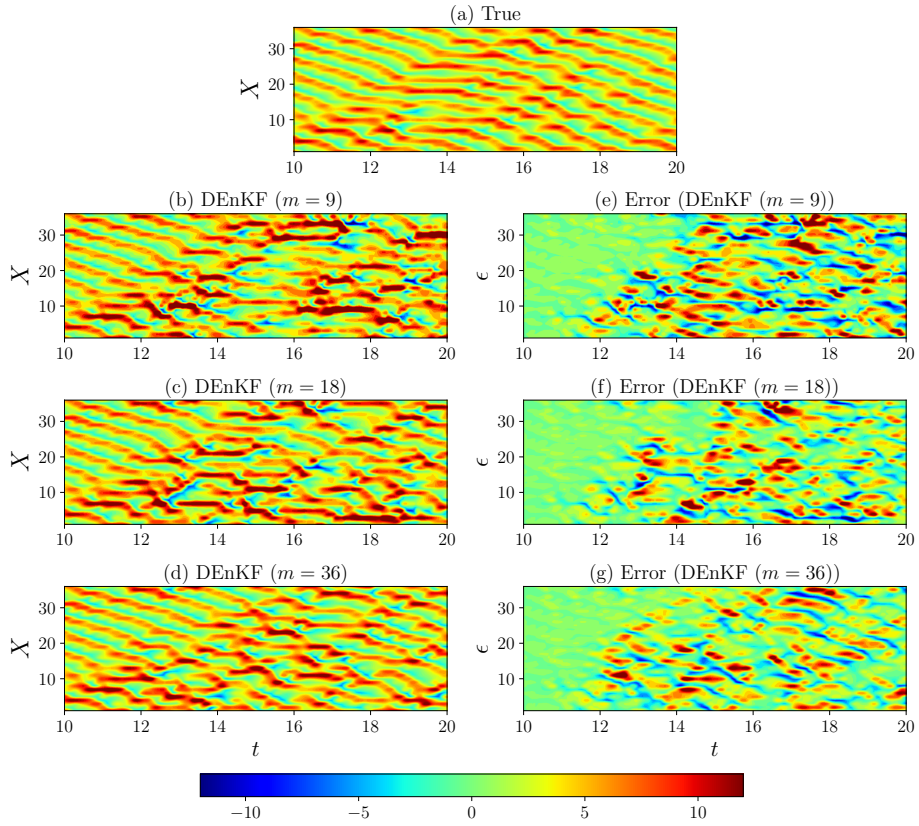

Figure 3.4: Full state trajectory of the multiscale Lorenz 96 model with no closure for subgrid processes. The observation data for the DEnKF algorithm is obtained by adding measurement noise to the exact solution of the multiscale Lorenz 96 system.

In this numerical experiment with the truncated model, 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 3.4, 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 3.2 and Figure 3.3 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 [208, 8, 10, 429, 58, 266]. 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

$$\text{RMSE} = \sqrt{\frac{1}{n}\frac{1}{n_t}\sum_{i=1}^{n}\sum_{k=1}^{n_t}\left(X_i^{\text{T}}(t_k) - X_i^{\text{P}}(t_k)\right)^2}, \tag{3.41}$$

where X_i^{T} is the true state of the system and X_i^{P} is the predicted state of the system. Table 3.1 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.

We highlight here that in the previous numerical experiment with the truncated model, we assumed that our forecast model is a true model. However, often the forecast models in DA are imperfect, and the model error introduced due to truncation of the sub-model is usually either modeled using the Gaussian noise or covariance inflation. Indeed, the ensemble Kalman filter framework is very well established and, to the extent that if modeling errors can be represented as zero-mean with a simple correlation structure, then the DA is very effective at correcting model errors. For

Table 3.1: Quantitative assessment of different neural network parameterizations for subgrid scale processes using the total root mean square error given by Equation (3.41).

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

example, Brajard et al. [44] utilized a Gaussian noise with zero mean and a certain value of standard deviation (optimized by tuning experiments) to account for the model error arising due to truncation of the parameterizations in a two-level Lorenz system. Similarly, Attia et al. [18] proposed a variational framework for adaptive tuning of inflation and localization parameters and demonstrated its successful performance for a two-level Lorenz system. For a fair comparison with neural network-based parameterizations, we repeat the numerical experiments with the truncated model for different values of inflation factor. We keep the number of ensembles fixed at $N = 30$, and the inflation factor is varied from 1.0 to 1.05 with an increment of 0.01. Figure 3.5 reports the RMSE for the truncated model for different inflation factors, and we can notice that with the proper choice of inflation factor and sufficient observations, the truncated model can also predict the true state of the two-level Lorenz system. In contrast to Figure 3.4, Figure 3.6 depicts the full state trajectory of the two-level Lorenz system estimated using the DEnKF algorithm with the inflation factor $\lambda = 1.03$ for three sets of observations. Overall, the results presented in Figure 3.5 suggest that the true state of the two-level Lorenz system can be determined when more than 50% of the state is observable, and a proper value of inflation factor is employed for the DA with the truncated model. Moreover, the prediction of the true state of the system

with neural network-based parameterization can be further improved by applying the inflation and the RMSE for different values of the inflation factor for CNN-based parameterization model are also shown in Figure 3.5. It can be clearly seen that there is a significant accuracy gain by adding the CNN based parameterizations for almost all configurations.

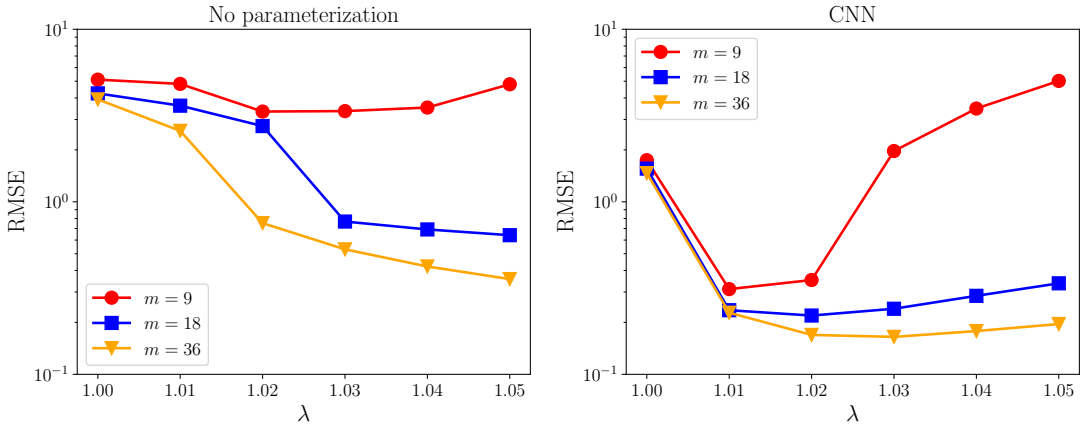

Figure 3.5: The root mean squared error for different values of the inflation factor for three sets of observations. The number of ensembles is kept fixed at $N = 30$ for all sets of observations.

3.5.2 Kraichnan Turbulence

We now characterize the performance of the EnKF algorithm, as described in Section 3.4, to estimate the state of the two-dimensional turbulence system when observation from high fidelity simulation are available. This test set-up is particularly challenging because of the modeling of unresolved scales in the LES solver, which is employed as the forecast model for two-dimensional turbulence. The performance of the EnKF algorithm is impacted by the choice of the model and the forecast model should be accurate enough for error control techniques like covariance inflation, covariance localization, stochastic forcing, etc. to work. As we will see, if the effect of unresolved scales are not modeled, even the EnKF algorithm with high value of the inflation factor does not improve the state estimate of the two-dimensional turbulence system.

The governing equations for the two-dimensional turbulence are numerically solved using the second-order finite difference discretization. The nonlinear Jacobian term is discretized with the energy-conserving Arakawa [12] numerical scheme. A third-order

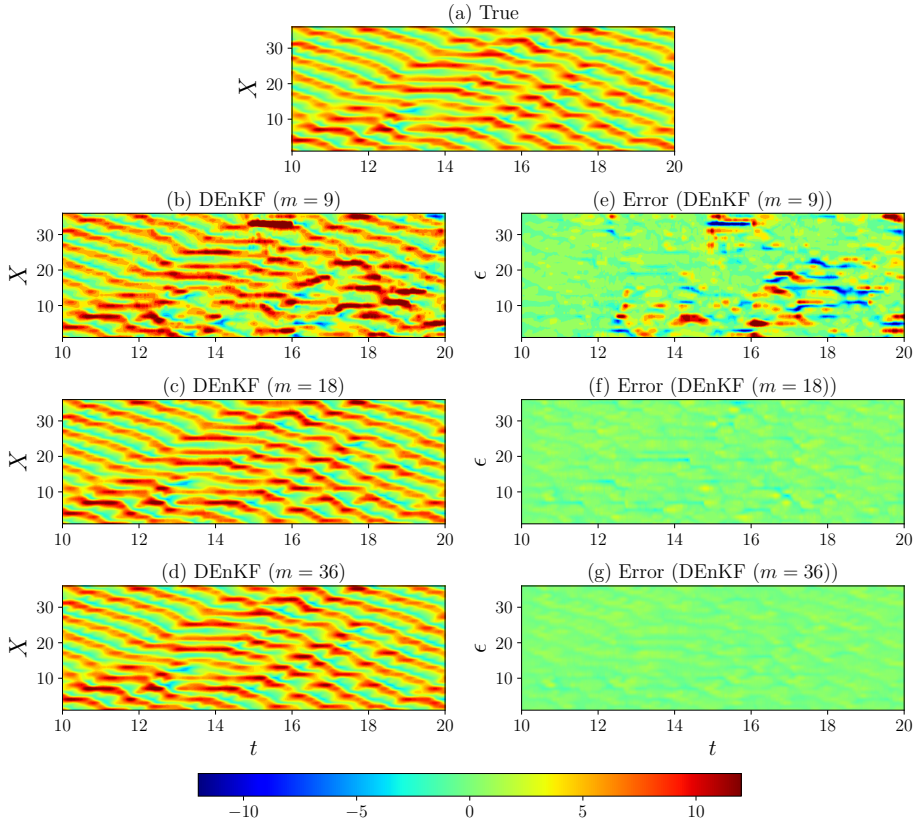

Figure 3.6: Full state trajectory of the multiscale Lorenz 96 model with no closure for subgrid processes and for the inflation factor $\lambda = 1.03$. The observation data for the DEnKF algorithm is obtained by adding measurement noise to the exact solution of the multiscale Lorenz 96 system.

total-variation-diminishing Runge-Kutta scheme is used for the temporal integration and a spectral Poisson solver is utilized to update streamfunction from the vorticity [124]. The computational domain is square in shape with dimensions $[0, 2\pi] \times [0, 2\pi]$ in x and y directions, respectively, and the periodic boundary condition is applied in both x and y directions. The training data for CNN is generated by carrying out the DNS at $\mathrm{Re} = 8000$ for two different initial conditions (independent of the truth model) on a grid resolution of 512×512 and then collecting total 800 snapshots (400 for each initial condition) between time $t = -2$ to $t = 2$. The initial condition is assigned in such a way that the maximum value of the initial energy spectra occurs at wavenumber $K_p = 10$. Further details of the randomization process for the initial condition can be found in related work [336]. The DNS data is coarsened to the 64×64 grid resolution using the spectral cutoff filter. The coarsened flow variables are then used to compute input features and labels for developing the data-driven

subgrid-scale parameterization model as discussed in Section 3.3.2. Once the CNN is trained, it is deployed in the forecast model from time $t = 0$ to $t = 4$. We highlight that the data from time $t = 2$ to $t = 4$ is not seen during the training.

The predicted source term $\tilde{\Pi}$ has negative eddy viscosities embedded in it and needs to be post-processed before directly injecting it into the solver [238]. The numerical stability is ensured during the *a posteriori* deployment by truncating the learned source term $\tilde{\Pi}$ corresponding to negative numerical viscosities as follows

$$\Pi = \begin{cases} \tilde{\Pi}, & \text{if } (\nabla^2\bar{\omega})(\tilde{\Pi}) > 0 \\ 0, & \text{otherwise} \end{cases} \tag{3.42}$$

In addition to truncating the source term corresponding to negative eddy viscosity, the truncation is also applied at points where the local eddy viscosity is greater than the local-average average eddy viscosity. This truncation scheme can be mathematically expressed as

$$\Pi_{i,j} = \begin{cases} \tilde{\Pi}_{i,j}, & \text{if } \bar{\nu}_{i,j} > \nu_{i,j} \\ 0, & \text{otherwise} \end{cases} \tag{3.43}$$

where the eddy viscosity ν is computed as

$$\nu_{i,j} = \frac{\tilde{\Pi}_{i,j}}{\nabla^2\bar{\omega}}, \tag{3.44}$$

and the local-averaged eddy viscosity $\bar{\nu}_{i,j}$ is calculated using the mean filtering kernel of size 3×3. This additional truncation scheme given in Equation 3.43 aids in preserving the statistical quantities like the kinetic energy spectra close to the DNS solution as compared to utilizing only negative eddy viscosity truncation scheme. In terms of the computational cost, the data-driven subgrid-scale parameterization model is significantly fast compared to the dynamic Smagorinksy model (DSM) [285] and we observed up to 30% reduction in computational speed in the *a posteriori* runs with the CNN based closure model.

The 'truth' solution for the data assimilation is obtained by solving the vorticity transport equation with a grid resolution 512×512 for the Reynolds number $\mathrm{Re} = 8000$ and then applying the spectral cutoff filter to get the filtered DNS solution on the coarse grid with resolution 64×64. Other methods like multigridding can also be adopted to relax the solution from fine grid to coarse grid [299]. The DNS solution is

generated from time $t = -2$ to $t = 0$ with $\Delta t = 1 \times 10^{-3}$. The assimilation is started at time $t = 0$ once the turbulence is developed and the initial transience from $t = -2$ to $t = 0$ is discarded. The observations are assimilated at every 10^{th} time step of the forecast model. The synthetic observations are generated by sampling vorticity field at 32×32 (corresponding to 25% of the full state of the system) equidistant points in x and y directions from the filtered DNS solution and then contaminating them with the Gaussian noise, i.e., $\mathbf{v}_k \sim \mathcal{N}(0, \mathbf{R}_k)$, where $\mathbf{R}_k = \sigma_b^2 \mathbf{I}$. We set observation noise at $\sigma_b^2 = 2$.

The initialization of the ensemble members also plays an important role in the performance of the EnKF algorithm [148], especially in the initial period of the DA. There are different ways that have been used for the initialization of ensemble members in DA of turbulent flows, such as, using the solution field separated by a certain time from the turbulent flow simulation [72], adding random perturbation to mean flow solution [74]. We initialize all ensemble ensemble members by adding a random perturbations drawn from $\mathcal{N}(0, \mathbf{P}_0)$ to the filtered DNS solution at time $t = 0$. The initial covariance matrix is set at $\mathbf{P}_0 = \sigma_0^2 \mathbf{I}$, where $\sigma_0^2 = 1$.

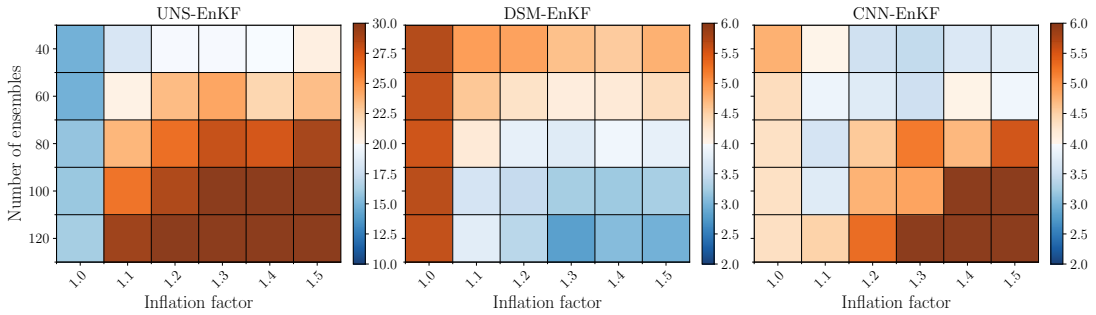

Figure 3.7: Average RMSE of the compensated energy spectra for different combinations of the inflation factor and number of ensembles. The RMSE is averaged over the inertial range (i.e., from $K = 8$ to $K = 32$) considering data from $t = 2$ to $t = 4$.

We illustrate the performance of the EnKF algorithm for three different types of forecast models. The first forecast model is the unresolved numerical simulation (UNS), where subgrid-scale parameterization is completely discarded. In the second forecast model, the dynamic Smagorinsky model (DSM) [118, 212] is used for modeling the source term in LES simulation. The third forecast model consists of utilizing the CNN based subgrid-scale parameterization. For a fair comparison of the application of the EnKF algorithm to three different models, we run experiments with different combinations of the number of ensemble members and the inflation factor. The number

of ensemble members is increased from 40 to 120 with an increment of 20 and the inflation factor is varied from 1.0 to 1.5 with an increment of 0.1. This gives us 30 different numerical experiments for each model. The performance of each numerical experiment is evaluated by computing the RMSE between the compensated kinetic energy spectra for the state estimated by the EnKF algorithm and the filtered DNS solution. The energy spectra is considered over the inertial range, i.e., between $K = 8$ to $K = 32$ and for 200 snapshots stored over the last half of the experiment timespan (from time $t = 2$ to $t = 4$). The RMSE results for these numerical experiments are shown in Figure 3.7. Results in Figure 3.7 suggest that the RMSE for the UNS model is significantly higher than the DSM and CNN model. The solution field predicted by UNS has a significant error due to the truncation of subgrid-scale parameterization and an increase in the number of ensembles or the inflation factor does not seem to help improve the state of the system predicted by the UNS model. The average RMSE for both DSM and CNN models is of a similar magnitude. Moreover, Figure 3.7 indicates that the lower RMSE occurs at less number of ensembles for the CNN model with a moderate inflation factor (i.e., 1.2-1.3).

We evaluate the performance of different models through kinetic energy spectra calculation and second-order vorticity structure functions. We compute the vorticity structure function using the formula given by [127] for two-dimensional turbulence and is shown below

$$S_\omega(\mathbf{r}) =< |\bar{\omega}(\mathbf{x}+\mathbf{r}) - \bar{\omega}(\mathbf{x})|^2 >, \tag{3.45}$$

where $<>$ indicates ensemble averaging, $\mathbf{x}$ is the position on the grid, and $\mathbf{r}$ is certain distance from this location.

Figure 3.8 displays the kinetic energy spectra at final time $t = 4$ obtained with different models for $\mathrm{Re} = 8000$. We can observe that there is an accumulation of the energy near grid cutoff wavenumber in the case of the UNS model. The UNS-EnKF model is not able to correct the state estimate of the system due to very high noise in the forward model. We see an improvement in the energy spectra predicted by the DSM-EnKF and CNN-EnKF model compared to utilizing only the parameterization model. We note here that the kinetic energy spectra for the filtered DNS solution is identical to the DNS spectra till the grid cutoff wavenumber due to the use of a spectral cutoff filter. Figure 3.9 depicts the second-order vorticity structure functions at final time $t = 4$ where the evaluation with the FDNS shows that the EnKF is successful in improving the prediction of the vorticity structure function for both

DSM and CNN model. We do not observe any improvement in the vorticity structure function prediction for the UNS model, which again emphasizes the importance of using an accurate forecast model in data assimilation.

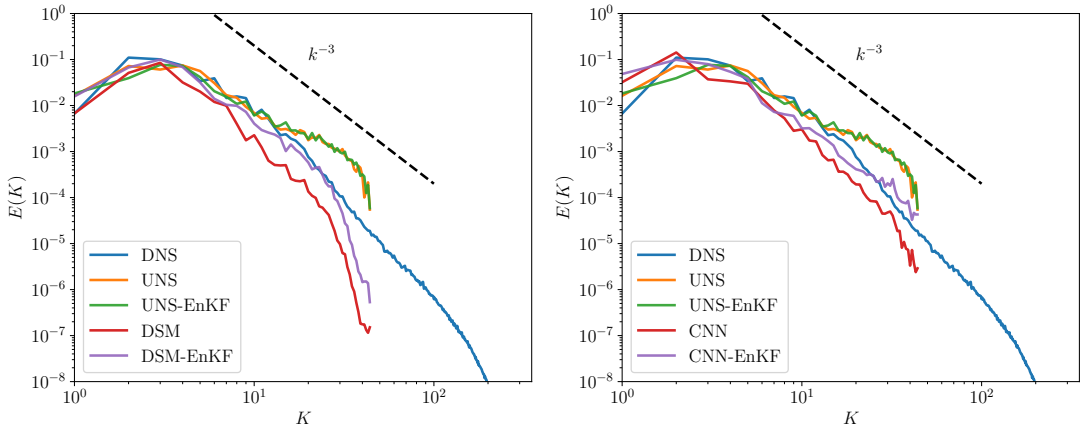

Figure 3.8: *A posteriori* kinetic energy spectra at $t = 4$ at $N_x \times N_y = 64 \times 64$ grid resolution for different models. The number of ensembles and the inflation factor for the EnKF algorithm for different models corresponds to minimum value of the average RMSE between the compensated energy spectra for the filtered DNS solution and the solution predicted with different models. The EnKF related parameters are $N = 40, \lambda = 1.0$ for UNS-EnKF, $N = 120, \lambda = 1.3$ for DSM-EnKF, and $N = 40, \lambda = 1.3$ for CNN-EnKF.

3.6 Concluding Remarks

The data-driven methods are successful in discovering model-free parameterizations from high-fidelity numerical simulations or experimental measurements and offers an alternative to parameterization models based on empirical or phenomenological arguments. The data-driven parameterization models are also computationally faster and are suitable for sequential data assimilation where multiple forward runs of a forecast model are required. To this end, 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

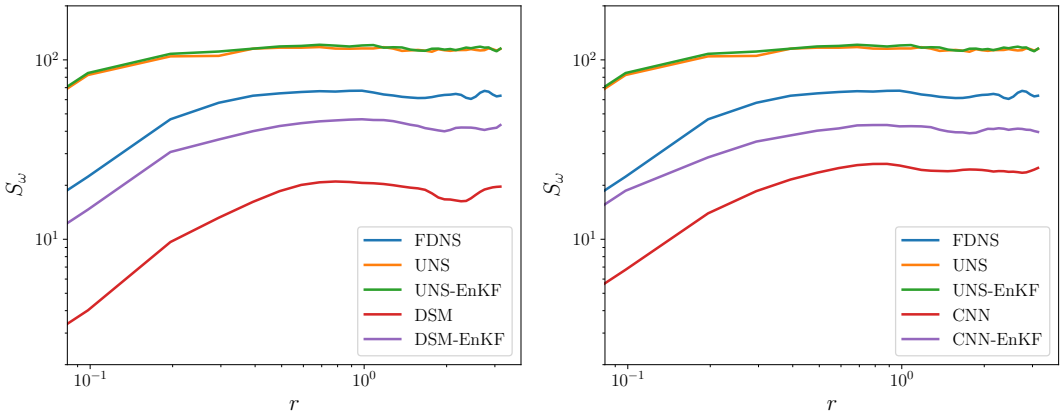

Figure 3.9: *A posteriori* second-order vorticity structure functions plotted against r for $Re = 8000$ at $N_x \times N_y = 64 \times 64$ grid resolution with different models used for the subgrid scale closure.

period and then there is a large discrepancy between true and predicted state of the system. To address this issue and to improve the long-term prediction, we exploit the sparse observations data through data assimilation.

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 and for Kraichnan turbulence where the parameterization model for unresolved scales is not known a priori. We obtain a considerable improvement in the prediction for both test cases 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. We also highlight that the inaccuracy associated with data-driven parameterizations can be tackled with data assimilation error control techniques like covariance inflation, covariance localization, stochastic forcing, etc.

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 ensemble Kalman filter based algorithms

for data assimilation. 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.

3.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

$$\frac{dX_i}{dt} = -X_{i-1}(X_{i-2} - X_{i+1}) - X_i + F, \tag{3.46}$$

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 1×10^{-2}. The observations are generated for data assimilation by adding a measurement noise from the Gaussian distribution with zero mean and the variance of $\sigma_b^2 = 1$ (i.e, $\mathbf{R}_k = \mathbf{I}$) to the true state of the system. The observations are assumed to be available at every 10^{th} time step, similar to the two-level variant of the Lorenz 96 model.

As depicted in Figure 3.10, 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.

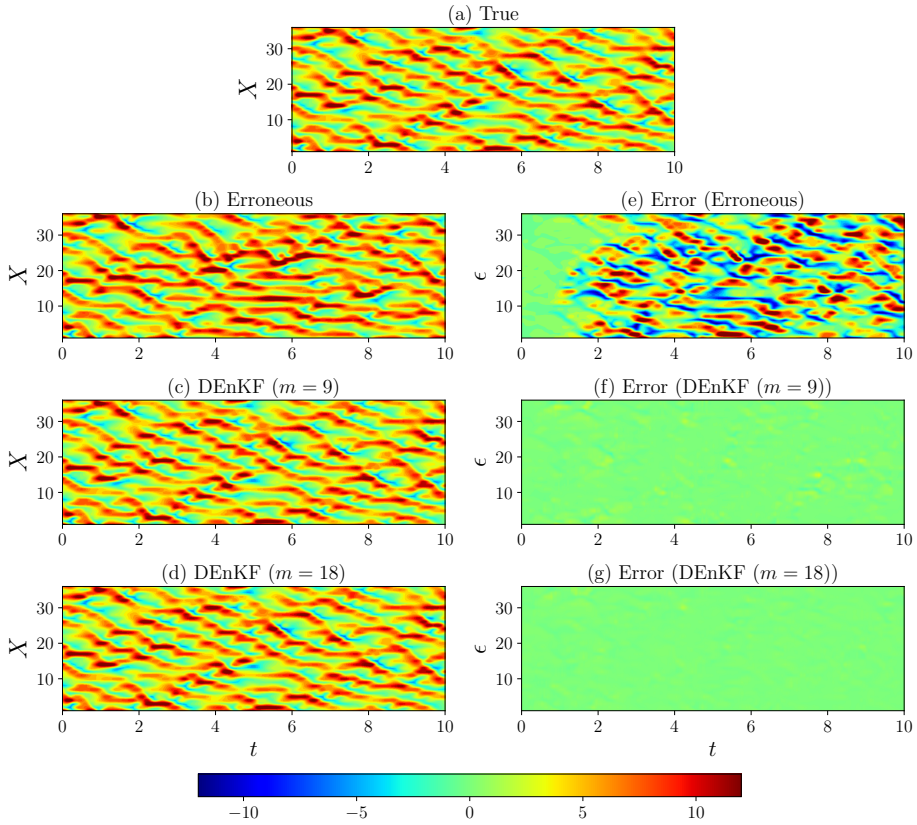

Figure 3.10: Full state trajectory of the Lorenz 96 model with the the DEnKF algorithm.

CHAPTER IV

Frame-invariant Neural Network Closures for Kraichnan Turbulence

The contents of this chapter is under review at Physica A: Statistical Mechanics and its Applications.[1].

Abstract: The numerical simulation of geophysical and atmospheric flows has to rely on the parameterization of subgrid scale processes due to their limited spatial resolution. Despite substantial progress in developing parameterization (or closure) models for subgrid scale (SGS) processes using semi-empirical physical principles, they remain imperfect and can lead to inaccurate predictions. In recent years, machine learning has been successful in extracting complex patterns from high-resolution spatio-temporal data, leading to improved parameterization models, and ultimately better coarse grid prediction. However, the inability to satisfy known physics and poor generalization hinders the application of these models for real-world problems. In this work, we propose a frame invariant closure approach to improve the accuracy and generalizability of deep learning-based subgrid scale closure models by embedding physical symmetries directly into the structure of the neural network. Specifically, we utilized specialized layers within the convolutional neural network in such a way that desired constraints are theoretically guaranteed without the need for any regularization terms. We demonstrate our framework for a two-dimensional decaying turbulence test case mostly characterized by the forward enstrophy cascade. We show that our frame invariant SGS model (i) accurately predicts the subgrid scale forcing, (ii) respects the physical symmetries such as translation, Galilean, and rotation invariance, (iii) is numerical stable when implemented in coarse-grid simulation, and (iv) generalizes to different initial conditions and Reynolds number. This work builds a bridge between extensive physics-based theories and data-driven modeling paradigms, and thus represents a promising step towards the development of physically consistent data-driven turbulence closure models.

[1]Pawar, S., San, O., Rasheed, A., & Vedula, P. (2022). Frame invariant neural network closures for Kraichnan turbulence. arXiv preprint arXiv:2201.02928.

4.1 Introduction

Computational modeling of turbulent flows remains a key issue in many engineering and geophysical applications. Turbulence involves a wide range of spatio-temporal scales that makes the direct numerical simulation (DNS) computationally infeasible for many complex systems. Coarse-graining approaches like large eddy simulation (LES) alleviate the computational burden by resolving only large and intermediate scales of the flow. The non-linearity of the Navier-Stokes equations introduces a subgrid scale (SGS) closure problem in LES which can be addressed via modeling of unresolved scales on the resolved flow quantities. The choice of the SGS model directly affects the accuracy of LES-based solution, and, therefore, the SGS modeling has been an active area of research for the past few decades [297, 330, 33, 244]. The development of SGS models has largely been driven by physical insights, mathematical considerations, and often problem-specific intuition. More recently, the availability of data from observations and high-resolution simulation along with advances in hardware and algorithms has fuelled interest in the development of data-driven turbulence models [87, 29, 50, 46].

The initial efforts towards data-driven SGS modeling include training a neural network to predict computationally expensive SGS model for channel flow [340] with the aim to speed-up LES. Similar frameworks includes applying neural network to determine the eddy-viscosity of the dynamic Smagorinsky model [276, 284], SGS model classification and blending [236], and reinforcement learning to predict SGS dissipation coefficient [268]. Deep learning (DL) has been applied to discovering new SGS models from the DNS data without any assumption of prior structural or functional form of the model [112, 238, 413, 391, 316, 110, 283]. The data-driven approach that employs convolutional neural network for learning the SGS model has also been used for different problems like two-dimensional decaying turbulence [284, 286, 128], three-dimensional decaying homogeneous isotropic turbulence [28], momentum forcing in ocean models [40], and subgrid-scale scalar flux modeling [105]. Moreover, neural networks have also been utilized to learn the optimal map between filtered and unfiltered variables in the approximate deconvolution framework for SGS modeling [237, 420]. Apart from SGS closure modeling, machine learning (ML) and in particular DL is being increasingly applied for different problems in fluid mechanics, like superresolution of turbulent flows [108, 170], Reynolds-Average Navier-Stokes (RANS) closure modeling [406, 279, 356], data assimilation [408, 15, 224, 53], spatio-temporal forecasting of

fluid flows [66, 65, 67] and reduced-order modeling [412, 409, 16, 2].

Despite their early success, ML models are faced with an array of challenges, such as poor generalization, lack of interpretability, and in some cases, violation of the known governing laws of the physical systems. For example, the SGS model derived through supervised learning may be numerically unstable, and diverge from the original trajectory, and this issue is exposed in many studies [238, 407, 28, 258]. These issues can be addressed by leveraging our prior knowledge about the physical systems into an ML model. Readers are referred to recent review articles on physics-informed machine learning [164, 167] that detail different methods of incorporating physics into ML models and discuss the capabilities and limitations of these methods. In the context of SGS modeling, there are many ways to embed physical constraints into the ML model. One such method for constructing a robust and generalizable SGS model is through the selection of suitable non-dimensionalized input and output quantities of the ML model to ensure that the known symmetries are respected [300]. Another class of methods pertains to the customized neural network architectures that encode the prior physical or mathematical knowledge as hard constraints. Some of the examples of this methods applied in fluid dynamics are tensor basis neural network [213], transformation invariant neural network [105], physics-embedded neural network [249], spatial transformer [62, 179], and equivariant networks [390, 347, 129].

In this work, we address the challenges associated with data-driven turbulence modeling by introducing a frame invariant convolutional neural network (FI-CNN) for SGS closure model discovery. Specifically, we select model inputs that are Galilean invariant, and replace the convolution operation with group convolutions [70, 395] to embed rotation invariance. Therefore, the FI-CNN preserves various symmetries, including translation, Galilean, and rotation both during training and inference. This makes the FI-CNN framework physically consistent and robust to extrapolation, and consequently, it produces accurate and stable results in their *a posteriori* deployment. We demonstrate our framework for two-dimensional turbulence which is often used as a prototypical test case for large-scale geophysical flows [39, 42]. Although we focus on SGS closure model development in this study, this framework has a promising application for many scientific problems where physical symmetries are very common. For example, there are several invariant finite-difference schemes based on equivariant moving frames that preserve Lie symmetries that have been developed for the solution of partial differential equations (PDEs) via consideration of modified forms of the underlying PDEs [275, 273, 274]. These symmetries can be exploited along with

data-driven discretization [21] to design numerical schemes that are more accurate than their non-invariant counterpart.

This chapter is organized as follows. In Sec 4.2, the symmetries of Navier-Stokes equations and the SGS closure modeling problem for two-dimensional turbulence is introduced. The detailed procedure on how to embed frame symmetries, including translation, Galilean, and rotation invariance within the CNN is provided in Sec 4.3. In Sec 4.4, the details on data generation and training are discussed. The performance of the FI-CNN in the *a priori* and *a posteriori* settings along with a detailed discussion of the results are presented in Sec 4.5. Finally, the concluding remarks and summary of the work are given in Sec 4.6.

4.2 SGS Closure Modeling

4.2.1 Symmetries of Navier-Stokes Equations

The Navier-Stokes equations governing incompressible fluid flows can be written in primitive variable (velocity–pressure) form as

$$\nabla \cdot \boldsymbol{u} = 0, \tag{4.1}$$

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} = -\frac{1}{\rho}\nabla p + \nu \nabla^2 \boldsymbol{u}, \tag{4.2}$$

where $\boldsymbol{u}$ is the velocity, p is the pressure, ρ is the density, and ν is the kinematic viscosity of the fluid. The governing equations for LES (also called as the filtered Navier-Stokes equations) are obtained by applying a spatial filter operation and it can be written as follows

$$\nabla \cdot \bar{\boldsymbol{u}} = 0, \tag{4.3}$$

$$\frac{\partial \bar{\boldsymbol{u}}}{\partial t} + \bar{\boldsymbol{u}} \cdot \nabla \bar{\boldsymbol{u}} = -\frac{1}{\rho}\nabla \bar{p} + \nu \nabla^2 \bar{\boldsymbol{u}} + \nabla \cdot \underbrace{(\overline{\boldsymbol{u}\boldsymbol{u}} - \bar{\boldsymbol{u}}\bar{\boldsymbol{u}})}_{\boldsymbol{\tau}_{\mathrm{SGS}}}, \tag{4.4}$$

where the overbar is used to denote the filtered variables and $\boldsymbol{\tau}_{\mathrm{SGS}}$ is subgrid-scale stress tensor. The problem of determining subgrid-scale stress tensor $\boldsymbol{\tau}_{\mathrm{SGS}}$ using the filtered variables is called the subgrid scale closure problem in LES.

There are many possible SGS closure models and any mathematical, physical constraints will lead to a specific type of SGS model. Requirements such as frame

invariance, realizability, finite kinetic energy can act as guiding principles for a satisfactory SGS closure model, and readers are referred to [33] for more details. The frame invariance constraint on the SGS model is derived by enforcing the symmetry of the original Navier-Stokes equations [330, 297, 269, 33] upon the filtered Navier-Stokes equations with the SGS closure model. Let G denote a group of transformation acting on space-time functions $\boldsymbol{u}(\boldsymbol{x}, t)$. We say that the group G is a symmetry group of the Navier-Stokes equations if, for all $\boldsymbol{u}$ which are solutions of the Navier-Stokes equations, and all $g \in G$, the function $g\boldsymbol{u}$ is also a solution [107]. The frame invariance constraint involves preservation of the symmetry property of the original Navier-Stokes equations to translation, Galilean, and rotation transformations, and they can be written as follows

- Space-translation: $g^{\text{space}}_{\boldsymbol{\delta}}\boldsymbol{u}(\boldsymbol{x}, t) = \boldsymbol{u}(\boldsymbol{x} - \boldsymbol{\delta}, t),\ \forall \boldsymbol{\delta} \in \mathbb{R}^3$, where $g^{\text{space}}_{\boldsymbol{\delta}}$ is the space-translation operator with the arbitrary displacement $\boldsymbol{\delta}$.
- Galilean transformation: $g^{\text{Gal}}_{\boldsymbol{\alpha}}\boldsymbol{u}(\boldsymbol{x}, t) = \boldsymbol{u}(\boldsymbol{x} - \boldsymbol{\alpha}t, t) + \boldsymbol{\alpha},\ \forall \boldsymbol{\alpha} \in \mathbb{R}^3$, where $g^{\text{Gal}}_{\boldsymbol{\alpha}}$ is the Galilean operator and $\boldsymbol{\alpha}$ is a fixed but arbitrary constant vector.
- Space-rotations: $g^{\text{rot}}_{\boldsymbol{A}}\boldsymbol{u}(\boldsymbol{x}, t) = \boldsymbol{A}\boldsymbol{u}(\boldsymbol{A}^{-1}\boldsymbol{x}, t)$, where $g^{\text{rot}}_{\boldsymbol{A}}$ is the rotation operation and $\boldsymbol{A} \in SO(3)$.

Imposing the symmetry preservation constraint give some structure to the the SGS model, and this insights have been extensively used in turbulence models [353, 330, 33]. We make use of these symmetries as physical constraints while building a frame invariant data-driven SGS model.

4.2.2 Two-dimensional Turbulence

In this work, we are interested in the SGS modeling for two-dimensional turbulence that is usually applied for modeling geophysical flows in the atmosphere and ocean [42, 39] where rotation and stratification dominate, and the most efficient way to model it is using the vorticity transport equation. Taking the curl of Eq. 4.2 yields the Navier-Stokes equations in vorticity-velocity formulation, and, for incompressible fluid flows, it can be written as follows

$$\frac{\partial \omega}{\partial t} + (\boldsymbol{u} \cdot \nabla)\omega = \nu \nabla^2 \omega, \tag{4.5}$$

where ω is the vorticity, and for two-dimensional flows, we have $\omega = \partial v/\partial x - \partial u/\partial y$. A scalar function called the streamfunction is defined in such a way that the continuity equation is satisfied if the velocity expressed in terms of the streamfunction is substituted in the continuity equation. This leads to the definition of velocity in terms of the streamfunction as follows

$$u = \frac{\partial \psi}{\partial y}, \quad v = -\frac{\partial \psi}{\partial x}, \tag{4.6}$$

where ψ is the streamfunction. The Poisson equation relating the vorticity and streamfunction is obtained by substituting the above velocity components in the definition of vorticity. Thus, we have

$$\nabla^2 \psi = -\omega. \tag{4.7}$$

It is convenient to write Eq. 4.5 in the vorticity-streamfunction formulation as follows

$$\frac{\partial \omega}{\partial t} + J(\omega, \psi) = \frac{1}{\mathrm{Re}} \nabla^2 \omega, \tag{4.8}$$

$$J(\omega, \psi) = \frac{\partial \omega}{\partial x}\frac{\partial \psi}{\partial y} - \frac{\partial \omega}{\partial y}\frac{\partial \psi}{\partial x}, \tag{4.9}$$

where $J(\cdot, \cdot)$ is the Jacobian (or the nonlinear term), and Re is the Reynolds number of the flow. The above equation is also called the vorticity transport equation.

The filtered Navier-Stokes equations for two-dimensional turbulence is obtained by applying a spatial filtering operation to Eq 4.8 as follows

$$\frac{\partial \overline{\omega}}{\partial t} + \overline{J(\omega, \psi)} = \frac{1}{\mathrm{Re}} \nabla^2 \overline{\omega}. \tag{4.10}$$

The above equation can be rewritten as

$$\frac{\partial \overline{\omega}}{\partial t} + J(\overline{\omega}, \overline{\psi}) = \frac{1}{\mathrm{Re}} \nabla^2 \overline{\omega} + \Pi, \tag{4.11}$$

where the overbar quantities represent filtered variables and are evolved on a grid that is significantly coarse compared to the DNS resolution. The effect of the unresolved scales due to truncation of high wavenumber flow scales is encapsulated in a subgrid scale (SGS) source term Π and must be modeled solely based on the resolved variables

$(\bar{\omega}, \bar{\psi})$. Mathematically, the true SGS source term Π can be expressed as

$$\Pi = J(\bar{\omega}, \bar{\psi}) - \overline{J(\omega, \psi)}. \tag{4.12}$$

The functional and structural models are the most commonly used approaches for modeling the SGS closure term in LES of turbulent flows [330]. The functional models are based on the concept of eddy viscosity where the effect of unresolved scales are approximated by artificial dissipation [350, 204]. The functional models can be further improved by dynamic adaptations of the coefficients that control the dissipation of the model and are determined adaptively by the use of a low-pass spatial test filter [118, 212, 102]. Although the dynamic formulation allows for spatial and temporal variation of coefficients in the eddy viscosity model, the ensemble averaging procedure does not allow for true back-scattering in order to limit the growth of numerical instabilities during the *a posteriori* testing [176, 153]. The structural models on the other hand aim at obtaining an accurate approximation of the SGS term and are based on the approximate deconvolution procedure [362, 337] and scale-similarity arguments [22]. Scale-similarity models address the SGS closure term by extrapolation from the smallest resolved scales to unresolved scales and have found to be the most accurate in *a priori* testing [330, 341]. However, numerical instabilities have been reported with scale-similarity models, and this has led to development of many mixed models with additional eddy viscosity term for stability reasons [197, 217, 234]. More recently, data-driven methods are emerging as a new paradigm to build turbulence closure models by extracting information from the data, and are seen as the potential applications to address the limitations of existing SGS models [87, 29, 86].

4.3 Frame Invariant SGS Closure Model

In this work, we consider the frame invariance property that must be satisfied by any SGS model and demonstrate how to include them within a neural network as hard constraints. The SGS source term Π is approximated using a neural network as shown below

$$\widetilde{\Pi} \approx \mathcal{M}(\bar{\omega}, \bar{\psi}), \tag{4.13}$$

where $\mathcal{M}$ is a neural network-based model, and $\widetilde{\Pi}$ is the approximation of true SGS source term Π. We remark here that the vorticity is defined using the spatial derivative of the velocity field, and, therefore it is invariant to Galilean transformations.

Additionally, the streamfunction is computed using the vorticity, and therefore both the inputs to our model are Galilean invariant. We now discuss how to embed the translation and rotational invariance/symmetry properties into the neural network-based model.

4.3.1 Translation Invariance

In this work, we employ the convolutional neural network (CNN) for learning the SGS closure model based on filtered vorticity and streamfunction as the model inputs. The CNN is an attractive choice for high-dimensional data and it does not suffer from the curse of dimensionality due to its weight-sharing feature. The CNN is composed of many convolutional layers and each of the layers is parameterized by filters, also called kernels, that has to be learned through training. Let $\underline{f}, \underline{k} : \mathbb{R}^2 \to \mathbb{R}^{N_c}$ be vector-valued two-dimensional features and kernel, i.e., $\underline{f} = (f_1, \cdots, f_{N_c})$ and $\underline{k} = (k_1, \cdots, k_{N_c})$, then the convolutional operation can be defined as

$$(\underline{k} \star \underline{f})(\mathbf{x}) = \sum_{c=1}^{N_c} \int_{\mathbb{R}^2} k_c(\mathbf{x} - \mathbf{x}') f_c(\mathbf{x}') d\mathbf{x}', \tag{4.14}$$

where $\mathbf{x}'$ is a dummy variable spanning over $\mathbb{R}^2$ space. The convolutional layer maps a feature vector $\underline{f}^{(l-1)} : \mathbb{R}^2 \to \mathbb{R}^{N_{l-1}}$ with N_{l-1} channels to feature vector $\underline{f}^{(l)} : \mathbb{R}^2 \to \mathbb{R}^{N_l}$ using a set of N_l kernels $\mathbf{k}^{(l)} := (\underline{k}_1^{(l)}, \cdots, \underline{k}_{N_l}^{(l)})$ and this operation can be defined as

$$\underline{f}^{(l)} = \zeta(\mathbf{k}^{(l)} \star \underline{f}^{(l-1)}) := \zeta(\underline{k}_1^{(l)} \star \underline{f}^{(l-1)}, \cdots, \underline{k}_{N_l}^{(l)} \star \underline{f}^{(l-1)}), \tag{4.15}$$

where ζ is an activation function. The parameters of the kernel are shared for the whole image as the kernel is convolved relative to the position about $\mathbf{x}$ and this aspect of the relative motion makes the CNN translation invariant. Although we present the convolution operation with continuous kernels, convolutional layers are equipped with discretized-filtering operations in their practical implementation. From here on, we refer to the model build using convolutional layers and nonlinear activation function as $\mathcal{M}_{\text{CNN}}$. The inputs to our model are the vorticity and streamfunction and the output is the SGS source term. Therefore, the learning map for $\mathcal{M}_{\text{CNN}}$ can be expressed as

$$\mathcal{M}_{\text{CNN}} : \{\bar{\omega}, \bar{\psi}\} \in \mathbb{R}^2 \to \mathbb{R}^2 \mapsto \{\widetilde{\Pi}\} \in \mathbb{R}^2 \to \mathbb{R}^1, \tag{4.16}$$

where $\widetilde{\Pi}$ is the predicted SGS source term.

4.3.2 Rotation Invariance

The rotational invariance of the SGS model requires that it maps as a tensor under the coordinate rotation [330]. As discussed in Section 4.3.1, the CNN is often invariant to only translation and not for other groups of transformations. However, there are recent developments on this front to exploit polar mapping of input images to convert rotation to translation [171]. In this work, we apply the group equivariant convolutions within the *E(2)-CNN* framework [394] for embedding rotational symmetry. The first roto-translation equivariant CNN was called the group convolutional neural network (GCNN) and it considered the rotations by multiples of $\pi/2$ [70]. The GCNN was further augmented by defining filters in terms of the steerable basis that are equivariant to rotations by multiples of $2\pi/N$, with $N > 4$ [396]. The *E(2)-CNN* library is based on the framework of steerable CNNs [71, 395] and it has different options for the group that takes the form of the semi-direct group $H = \mathbb{R}^2 \rtimes G$ where the group $G \leq O(2)$ (here $O(2)$ is the group of continuous rotations and reflections). For example, the group $H = \mathbb{R}^2 \rtimes SO(2) = SE(2)$ is the semi-direct product of the group of planar translations $\mathbb{R}^2$ and continuous rotations $SO(2)$. In this work, we utilize the cyclic group $G = C_N$ containing the discrete rotations of $2\pi/N$ (i.e., $H = \mathbb{R}^2 \rtimes C_N$). For a large value of N, the difference between continuous rotations and discrete rotations is indistinguishable due to space discretization.

A full understanding of the steerable CNNs requires some knowledge of the group representation theory, but the implementation of the steerable CNNs is similar to ordinary CNNs. Readers are suggested to read Weiler and Cesa [394] and references therein for a more comprehensive discussion on the general framework of steerable CNNs. Here, we briefly explain the G-equivariant convolutions. A G-convolution between a vector-valued two-dimensional image $\underline{f} : \mathbb{R}^2 \to \mathbb{R}^{N_c}$ and a filter $\underline{k} : \mathbb{R}^2 \to \mathbb{R}^{N_c}$ where $\underline{f} = (f_1, \cdots, f_{N_c})$ and $\underline{k} = (k_1, \cdots, k_{N_c})$ can be expressed as follows

$$(\underline{k} \tilde{\star} \underline{f})(g) = \sum_{c=1}^{N_c} \int_{\mathbb{R}^2} k_c(g^{-1}\mathbf{x}') f_c(\mathbf{x}') d\mathbf{x}', \tag{4.17}$$

where $g = (\mathbf{x}, \theta) \in H = \mathbb{R}^2 \rtimes C_N$, $\mathbf{x}' \in \mathbb{R}^2$, and $\tilde{\star}$ denotes the group correlation operation under joint translation and rotation. This operation corresponds to lifting of the data on two-dimensional space to the data that lives on a three-dimensional position orientation space H. The first layer maps a two-dimensional image $\underline{f}^{(l-1)} : \mathbb{R}^2 \to \mathbb{R}^{N_{l-1}}$ with N_{l-1} channels at $(l-1)$th layer to H vector image $\underline{F}^{(l)} : H \to \mathbb{R}^{N_l}$ using a set of

N_l kernels $\mathbf{k}^{(l)} := (\underline{k}_1^{(l)}, \cdots, \underline{k}_{N_l}^{(l)})$ as follows

$$\underline{F}^{(l)} = \zeta(\mathbf{k}^{(l)} \tilde{\star} \underline{f}^{(l-1)}) := \zeta(\underline{k}_1^{(l)} \tilde{\star} \underline{f}^{(l-1)}, \cdots, \underline{k}_{N_l}^{(l)} \tilde{\star} \underline{f}^{(l-1)}). \tag{4.18}$$

Since the $\underline{F}$ is a function on H, the filters from the second layer onward should also be functions on H. The subsequent group convolutions are defined as [396, 30]

$$(\underline{K} \tilde{\star} \underline{F})(g) = \sum_{c=1}^{N_c} \int_H K_c(g^{-1}h) F_c(h) dh. \tag{4.19}$$

A group convolution layer is defined by a set of H kernels $\mathbf{K} := (\underline{K}_1^{(l)}, \cdots \underline{K}_{N_l}^{(l)})$ that maps $\underline{F}^{(l-1)}$ with N_{l-1} channels to $\underline{F}^{(l)}$ with N_l channels as shown below

$$\underline{F}^{(l)} = \zeta(\mathbf{K}^{(l)} \tilde{\star} \underline{F}^{(l-1)}) := \zeta(\underline{K}_1^{(l)} \tilde{\star} \underline{F}^{(l-1)}, \cdots, \underline{K}_{N_l}^{(l)} \tilde{\star} \underline{F}^{(l-1)}). \tag{4.20}$$

Finally, the feature field at the last layer can be synthesized from H space to $\mathbb{R}^2$ space. The user interface of the *E(2)-CNN* library [394] hides most of the intricacies of group theory, solutions of the steerable kernels space constraints, and requires users to specify only the transformation laws of the feature spaces. We use the regular representation for all hidden layers and the action of regular representation is given by permutation matrices (Appendix B in [394]). From here on, the model built using the equivariant CNN is called as $\mathcal{M}_{\text{FI-CNN}}$. The learning map for $\mathcal{M}_{\text{FI-CNN}}$ is same as the $\mathcal{M}_{\text{CNN}}$ given in Eq. 4.16.

4.4 Data Generation and Training

The parameters of the neural network based SGS models are learned through supervised training that requires a set of labeled inputs and outputs, usually obtained from direct numerical simulation (DNS). The dataset should encompass a range of dynamics that is expected to be reproduced by the SGS model. The data for training is generated from DNS of two-dimensional Kraichnan turbulence in a doubly periodic square domain with $L_x \times L_y = [0, 2\pi] \times [0, 2\pi]$, and the domain is discretized using 2048^2 degrees of freedom. Our DNS solver is based on a second-order accurate energy-conserving Arakawa scheme [12] for the nonlinear Jacobian and second-order accurate finite-difference scheme for the Laplacian of the vorticity. The elliptic equation for the relationship between the streamfunction and vorticity is solved using a second-order

accurate FFT-based Poisson solver, and the time integration is performed with a third-order accurate Runge-Kutta method. The vorticity distribution at the start of the simulation is initialized based on the energy spectrum given by [271]

$$E(k) = Ak^4 \exp\left(- \left(\frac{k}{k_p} \right)^2 \right), \tag{4.21}$$

where $A = 4k_p^{-5}/3\pi$ and $k = |\mathbf{k}| = \sqrt{k_x^2 + k_y^2}$. For our numerical experiments, we use $k_p = 10$. The initial vorticity distribution in Fourier space is obtained through the introduction of random phase as follows

$$\tilde{\omega}(\mathbf{k}) = \sqrt{\frac{k}{\pi} E(k)}\ \mathrm{e}^{\mathbf{i}\xi(\mathbf{k})}, \tag{4.22}$$

where the phase function is given by $\xi(\mathbf{k}) = \phi(\mathbf{k}) + \eta(\mathbf{k})$. Here, $\phi(\mathbf{k})$ and $\eta(\mathbf{k})$ are independent random values chosen in $[0, 2\pi]$ at each grid point in the first quadrant of the $k_x - k_y$ plane (i.e., $k_x, k_y \geq 0$). The phase function for other quadrants is obtained through conjugate relations as follows

$$\xi(\mathbf{k}) = -\phi(\mathbf{k}) + \eta(\mathbf{k}) \text{ for } k_x < 0 \text{ and } k_y \geq 0, \tag{4.23}$$

$$\xi(\mathbf{k}) = -\phi(\mathbf{k}) - \eta(\mathbf{k}) \text{ for } k_x < 0 \text{ and } k_y < 0, \tag{4.24}$$

$$\xi(\mathbf{k}) = \phi(\mathbf{k}) - \eta(\mathbf{k}) \text{ for } k_x \geq 0 \text{ and } k_y < 0, \tag{4.25}$$

$$\tag{4.26}$$

Further details on the problem setup and the numerical schemes can be found in our previous work [234]. Different realizations of the initial vorticity field can be obtained by using different phase functions with a different seed for random value generation.

The DNS is performed from time $t = 0$ to $t = 4$ with the time step $\Delta t = 5 \times 10^{-4}$. In the Kraichnan turbulence problem, the initial vorticity field is dominated by a population of vortices and small-scale structure starts appearing as the flow evolves. The initial spin-up time from $t = 0$ to 0.5 is neglected and we start collecting the data for training from time $t = 0.5$. From time $t \approx 0.5$, the flow has started following Kraichnan–Batchelor–Leith (KBL) theory [181, 24, 204] of energy cascade where energy is transferred from the smaller scales to the larger scales. From time $t \approx 0.5$ onward, large coherent vortices start emerging through vortex merging mechanism and viscous dissipation of small-scale structures. The vorticity field and angle-averaged

energy spectrum are displayed in Fig. 4.1 and we can see that the energy spectrum has started exhibiting k^{-3} scaling from approximately $t = 0.5$.

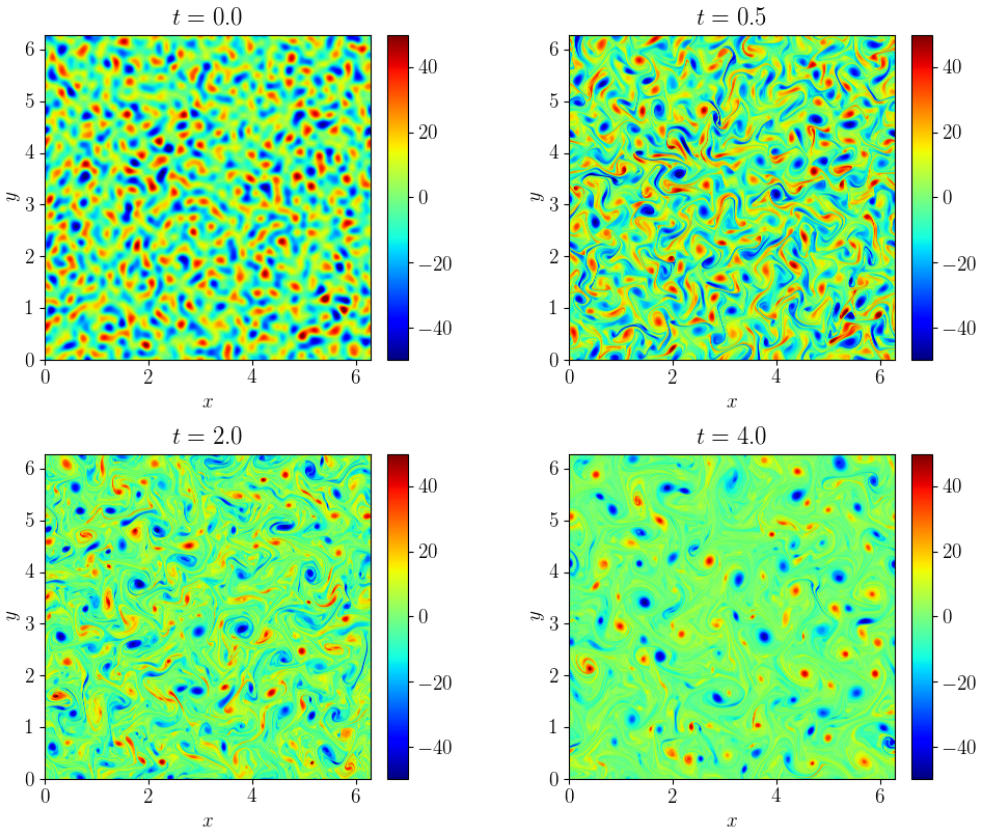

Figure 4.1: Visualization of the vorticity field and energy spectrum at different time instances for Re = 16000 with grid resolution 2048×2048.

The data for training a neural network-based SGS model is stored at every $20\Delta t$, i.e., we have 350 snapshots of the vorticity and streamfunction between $t = 0.5$ to 4.0. We emphasize here that the neural network-based SGS model is trained only for a single Reynolds number Re = 16000 and we assess the performance of the model for Reynolds number up to Re = 128000. The filtered DNS data for training is obtained by first applying a Gaussian filter transfer function to the DNS data and then coarse-graining the filtered solution to the LES grid [421, 128]. The Gaussian filter provides a smooth transition between resolved and subgrid scales and is also positive definite in physical and wave space [294, 324]. Additionally, our numerical solver is in physical space, and therefore we select the Gaussian filter instead of a spectral cut-off filter. The coarse-grid level for LES is 256^2 which corresponds to 64 times fewer spatial degrees of freedom compared to DNS.

We do not pre-process the filtered DNS data before training as the DNS data is

generated from a non-dimensionalized vorticity transport equation. The total data is divided into 80% of the data for training and 20% for the validation set. While the input and output of both $\mathcal{M}_{\text{CNN}}$ and $\mathcal{M}_{\text{FI-CNN}}$ are the same, the user needs to specify the type of representation for intermediate feature field while constructing an FI-CNN [394], similar to the number of kernels for CNN. We use the kernel size of 5×5, six hidden layers and ReLU activation function for the $\mathcal{M}_{\text{CNN}}$ and $\mathcal{M}_{\text{FI-CNN}}$. The number of kernels for the CNN and FI-CNN models is set to 30 and 16, respectively. With these hyperparameters, the number of trainable parameters is roughly the same around $O(1.1 \times 10^5)$ for both models. Both the models are trained for 100 iterations using an Adam optimizer. Fig. 4.2 shows the history of training loss versus iterations for both neural network-based SGS models and we can observe that the loss for $\mathcal{M}_{\text{FI-CNN}}$ is almost one order magnitude less than the loss for $\mathcal{M}_{\text{CNN}}$. This can be attributed to rotational invariances incorporated in the $\mathcal{M}_{\text{FI-CNN}}$ against $\mathcal{M}_{\text{CNN}}$, which is only invariant to translation and Galilean transformation. For both neural network-based SGS models, we use the parameters (i.e., weights) corresponding to minimum validation loss obtained while training the neural network.

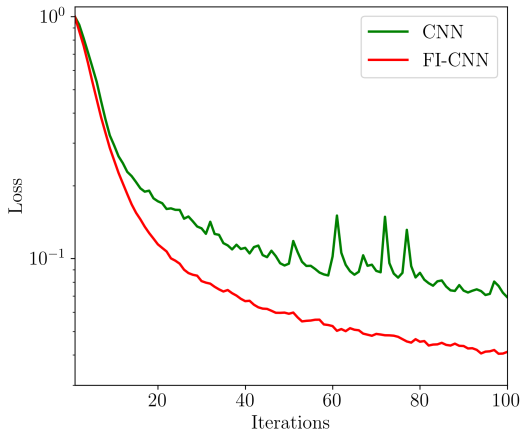

Figure 4.2: History of the training loss versus iterations for $\mathcal{M}_{\text{CNN}}$ and $\mathcal{M}_{\text{FI-CNN}}$.

4.5 Numerical Results

In this section, we first outline the numerical results of our framework in the *a priori* settings where the neural network-based models are utilized in predicting the SGS source term. We analyze the capability of $\mathcal{M}_{\text{CNN}}$ and $\mathcal{M}_{\text{FI-CNN}}$ in incorporating the frame-invariance property over the testing data. Then, we present the results of *a posteriori* LES coupled with neural network-based SGS models and evaluate their

performance using numerous statistical metrics.

Before we present our results for the Kraichnan turbulence problem, we illustrate the capability of CNN and FI-CNN models to satisfy rotational equivariance using a simple example of a vortex merging test case. In this test case, two vortices of the same sign are separated by a certain distance from each other along a parallel axis and the merging of vortices occurs ending as a single, nearly axisymmetric, final vortex [384]. The CNN and FI-CNN models are trained using the data generated for horizontally aligned vortices initial condition, i.e., the vorticity field shown in the top row and first column in Figure 4.3. During the inference, both CNN and FI-CNN models are able to predict the SGS source term with a sufficient level of accuracy. However, if we test the CNN model for vertically aligned vortices (bottom row and first column in Figure 4.3), we observe that the predicted SGS source term is inaccurate and the rotational equivarince is not satisfied. However, the FI-CNN model is successful in preserving the rotational symmetry and correctly captures the 90° rotation of the SGS source term. Therefore, by designing a model that respects certain physical symmetries, we can guarantee that the network is robust and generalizes well to different scenarios.

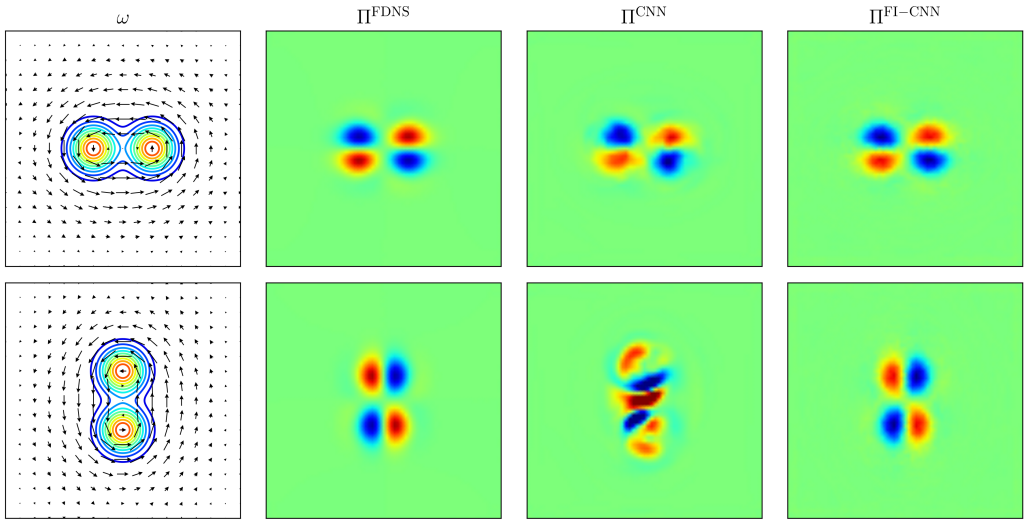

Figure 4.3: Illustration of rotational symmetry for the initial vorticity field of two Gaussian-distributed vortices. The CNN and FI-CNN models are trained using the data generated for horizontally aligned vortices, i.e., the top row and first column. When data-driven models are used for vertically aligned vortices, the CNN model fails to capture the rotational equivariance (the bottom row and third column), while the FI-CNN model correctly captures the rotational equivariance (the bottom row and fourth column).

4.5.1 *A Priori* Investigation

Here, we assess the performance of neural network-based models in predicting the SGS source term compared to the true SGS source term for the out-of-training data. The out-of-training data is obtained for a different initial condition and corresponds to 70 snapshots stored randomly between time $t = 0.5$ to $t = 4.0$. We remark here that the initial energy spectrum for the testing data is also given by Eq. 4.21 and the difference is due to a different phase function. Fig. 4.4 displays the probability distribution function and cumulative distribution function for the test data. There is a very good agreement between the true SGS source term and the predicted SGS source term from both models. However, we notice that the $\mathcal{M}_{\text{FI-CNN}}$ is more accurate near the tails of the distribution (Fig 4.4, left) compared to $\mathcal{M}_{\text{CNN}}$. This difference is also observable in the cumulative distribution function of true and predicted SGS source terms and is highlighted in the zoom-in portion (Fig 4.4, right). Based on these results, we may conclude that both neural network-based SGS model has learned the relationship between filtered quantities and the SGS source term. Both models are able to produce viable physical results for the completely unseen data with similar physics.

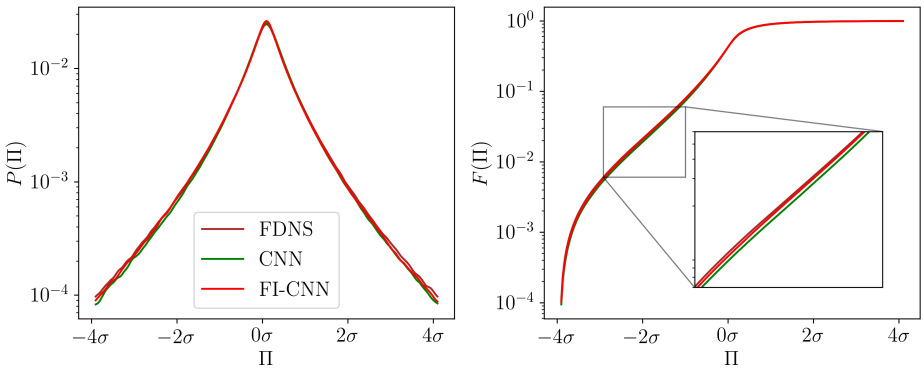

Figure 4.4: Probability distribution function (left) and cumulative distribution function (right) of the SGS source term over the entire testing dataset. The testing dataset corresponds to 70 snapshots selected randomly between time $t = 0.5$ to $t = 4.0$ for the initial condition different from the one used in training and σ is the standard deviation of the data.

Next, we evaluate neural network-based models in respecting rotational symmetry on the test data. Specifically, we perturbed the test data based on the rotation transformation, and generate multiple test datasets. Then, we compute the root mean squared error (RMSE) for each dataset, and calculate the expected value and variance

Table 4.1: Evaluation of the rotational symmetry constraints provided by $\mathcal{M}_{\text{CNN}}$ and $\mathcal{M}_{\text{FI-CNN}}$. The expected value and variance of the root mean squared error on the SGS source term predicted by both models is computed from many realizations (20 ensembles) on the testing data. The testing dataset corresponds to 70 snapshots selected randomly for the initial condition different from the one used for training. The rotational angle $\boldsymbol{A}$ is sampled uniformly between $[0°, 360°]$ in the multiple of 90° and is used in the rotational operator $g^{\text{rot}}_{\boldsymbol{A}}$. The Pearson's cross-correlation coefficient between the predicted SGS source term and the filtered DNS solution is computed as $\mathcal{P}(X, Y) = \text{cov}(X, Y)/\sigma_X \sigma_Y$.

Metric	$\mathcal{M}_{\text{CNN}}$	$\mathcal{M}_{\text{FI-CNN}}$
$\text{E}[\mathcal{L}]$	10.6941	7.3462
$\sigma[\mathcal{L}]$	4.2442×10^{-2}	5.6587×10^{-8}
$\mathcal{P}(X, Y)$	0.9600	0.9776

for all test datasets. The rotation matrix $\mathbf{A}$ is sampled uniformly between $[0°, 360°]$ in the multiple of 90°. If the rotation symmetry constraint is satisfied strictly, then the RMSE will be the same for each test dataset leading to zero variance for the RMSE metric. The results in Table 4.1 demonstrate the remarkable ability of $\mathcal{M}_{\text{FI-CNN}}$ to respect the rotation symmetry in contrast to $\mathcal{M}_{\text{CNN}}$ which violates this symmetry. Furthermore, the expected value of RMSE for $\mathcal{M}_{\text{FI-CNN}}$ is one order of magnitude lower than $\mathcal{M}_{\text{CNN}}$ and is consistent with the training loss (Fig. 4.2). The performance of both models is substantially accurate in terms of the Pearson's cross-correlation coefficient, with $\mathcal{M}_{\text{FI-CNN}}$ slightly better than $\mathcal{M}_{\text{CNN}}$. We note here that it is relatively straightforward to embed Galilean invariance constraint within neural network-based SGS model through intelligent selection of model inputs, and translation invariance through simple CNN. However, incorporating rotational symmetry in a neural network-based SGS model is more complex and requires special consideration. Although a relatively simple method like data augmentation can be utilized to impose the rotation symmetry as a soft constraint, it does not satisfy rotation invariance strictly [105]. The strict enforcement of rotation symmetry is challenging and requires the use of tailored neural network architecture, such as equivariant CNN.

4.5.2 *A Posteriori* Deployment

We now evaluate the performance of neural network-based SGS models in the LES of Kraichnan turbulence. The spatial resolution for LES is reduced by a factor of eight in each direction and this gives us 256^2 degrees of freedom. The time step for

LES simulation is ten times larger compared to the DNS, i.e., $\Delta t_{\text{LES}} = 5 \times 10^{-3}$. The performance of neural network-based SGS models is compared with the widely used dynamic Smagorinsky model (DSM) [118, 212]. The *a posteriori* deployment is a rigorous task for any data-driven SGS model due to the presence of numerical instabilities, and the challenges and remedies have been highlighted in many studies [238, 236, 235, 128, 28, 360, 426]. For example, Maulik et al. [238] and Zhou et al. [426] achieved the stable LES results by truncating SGS source term corresponding to negative eddy viscosity. Stoffer et al. [360] attained stable *a posteriori* results by resorting to artificially introducing additional dissipation (via eddy-viscosity models). Guan et al. [128] provided sufficient amount of data during training to obtain a stable *a posteriori* results. While the exact reason for this behavior is unknown, several issues such as error accumulation, aliasing errors, numerical instability, extrapolation beyond the training data, chaotic nature of turbulence, presence of multiple attractors might be responsible for unstable *a posteriori* simulation [28, 360, 29, 258].

From our *a posteriori* simulation, it is revealed that $\mathcal{M}_{\text{CNN}}$ is unstable, while $\mathcal{M}_{\text{FI-CNN}}$ is able to produce a stable and physical solution without any kind of clipping or by adding artificial dissipation. We note here that, perhaps $\mathcal{M}_{\text{CNN}}$ can also achieve stable *a posteriori* simulation, provided there is sufficient data available for training or some kind of post-processing is carried out for the predicted SGS source term. However, our main motivation in this work is to construct a physically consistent data-driven SGS model that can be trained in a data-sparse regime and is also stable in the *a posteriori* simulation. We assess the performance of our *a posteriori* simulation using several statistical metrics and compare it with the statistics from filtered DNS solution. The turbulent kinetic energy at time t_k is computed as follows

$$TKE(t_k) = \lambda(u_f^2(t_k) + v_f^2(t_k)), \tag{4.27}$$

where u_f and v_f are the fluctuating components of velocity given by

$$u_f = \bar{u} - \lambda(\bar{u}), \tag{4.28}$$

$$v_f = \bar{v} - \lambda(\bar{v}), \tag{4.29}$$

where $\lambda(a)$ represents the spatial average of the field variable a. The velocity $\bar{u}$, and $\bar{v}$

are computed by spectral differentiation of streamfunction as shown below

$$\bar{u} = \frac{\partial \bar{\psi}}{\partial y}; \quad \bar{v} = -\frac{\partial \bar{\psi}}{\partial x}. \tag{4.30}$$

The vorticity variance at each time step is computed as

$$\sigma^2 = \lambda((\bar{\omega} - \lambda(\bar{\omega}))^2). \tag{4.31}$$

We compare the kinetic-energy spectra and the vorticity structure function at intermediate time $t = 2.0$ and at final time $t = 4.0$ with the k^{-3} scaling which is observed in two-dimensional turbulence based on the classical KBL theory. The vorticity structure function is calculated using the formula given by [127] for two-dimensional turbulence as follows

$$S_\omega =< |\bar{\omega}(\mathbf{x} + \mathbf{r}) - \bar{\omega}(\mathbf{x})|^2 >, \tag{4.32}$$

where $<>$ indicates ensemble averaging, $\mathbf{x}$ is the position on the grid, and $\mathbf{r}$ is certain distance from this location. The PDF of the vorticity increment is utilized to assess the capability of SGS models in predicting the coherent vortices in the flow. The vorticity increments at different separations $\mathbf{r}$ is defined as

$$\delta\omega(\mathbf{r}) = \omega(\mathbf{x} + \mathbf{r}) - \omega(\mathbf{r}). \tag{4.33}$$

We reiterate here that neural network-based SGS models are trained using the data for Reynolds number Re = 16000 and a single initial condition. Once the models are trained, the LES coupled with SGS models is performed for Reynolds number up to Re = 128000 and for five different initial conditions. Fig. 4.5 shows the evolution of turbulent kinetic energy and vorticity variance for LES runs with five different initial conditions and for several Reynolds numbers. For all the LES runs, we initialize the vorticity field at $t = 0.5$ after the initial spin-up period using the filtered DNS solution. We can observe that the model $\mathcal{M}_{\text{CNN}}$ is stable only for short time and quickly becomes unstable after $t \approx 2.0$ even for Reynolds number Re = 16000 which was included in the training. In contrast to $\mathcal{M}_{\text{CNN}}$, model $\mathcal{M}_{\text{FI-CNN}}$ is stable for all test cases conducted here without any post-processing of the predicted SGS source term. The ensemble averaging procedure in DSM leads to highly dissipative results and is noticeable in the overprediction of the energy decay rate. The results of the

LES runs with $\mathcal{M}_{\text{FI-CNN}}$ have the best agreement with filtered DNS solution for both turbulent kinetic energy and the vorticity variance.

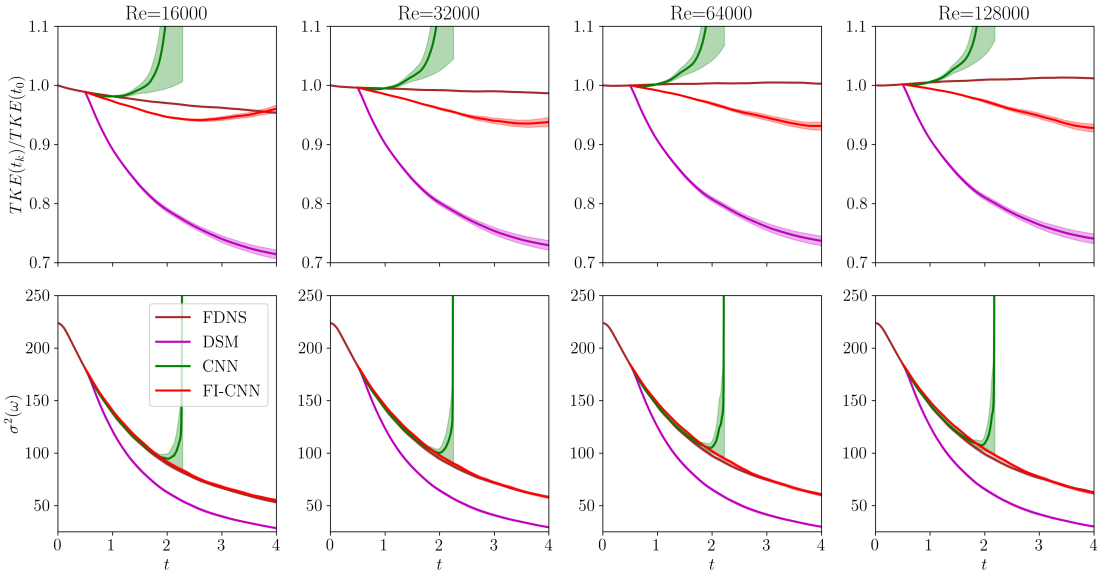

Figure 4.5: The time evolution of the turbulent kinetic energy $TKE(t_k)$ normalized by the initial turbulent kinetic energy $TKE(t_0)$ (top row) and vorticity variance (bottom row) for different Reynolds numbers at 256^2 grid resolution. The solid line shows the mean from LES runs for five different initial conditions and the shaded area corresponds to one standard deviation. The LES simulation starts at $t = 0.5$ after the initial spin-up time (i.e., once the turbulence has set in). The CNN and FI-CNN models are trained using the data generated from a single initial condition at Reynolds number Re = 16000.

Fig. 4.6 displays the kinetic-energy spectra at intermediate time $t = 2.0$ and at final time $t = 4.0$ obtained from LES runs with five different initial conditions for multiple Reynolds number. Although the LES runs coupled with $\mathcal{M}_{\text{CNN}}$ is stable at $t = 2.0$, the solution is unphysical as seen by the energy pile up near grid cutoff wavenumbers. This behavior is also illustrated in Fig. 4.7 through a large value of vorticity structure function at $t = 2.0$ across all Reynolds numbers. The LES runs with $\mathcal{M}_{\text{CNN}}$ has diverged around $t \approx 2.5$ (see Fig. 4.5), and, therefore the kinetic-energy spectra and vorticity structure function are missing at $t = 4.0$ in Fig. 4.6 and Fig. 4.7, respectively. There is a very good agreement between the kinetic-energy spectra for LES runs with $\mathcal{M}_{\text{FI-CNN}}$ and filtered DNS solution, especially in the inertial subrange and k^{-3} theoretical scaling is captured accurately. From Fig. 4.7, we can see that the model $\mathcal{M}_{\text{FI-CNN}}$ is successful in producing the $r^{3/2}$ scaling [183] for the vorticity structure function at small scales and it gradually flattens near the large scales. The excessive

dissipation of DSM is also illustrated in Fig. 4.6 and Fig. 4.7 via mismatch between kinetic-energy spectra and vorticity structure function between DSM and filtered DNS solution. The successful performance of LES runs with $\mathcal{M}_{\text{FI-CNN}}$ demonstrates that incorporating frame symmetries as hard constraints has been effective in stabilizing the coarse-grid simulation and in ensuring generalized learning across different initial conditions and Reynolds numbers.

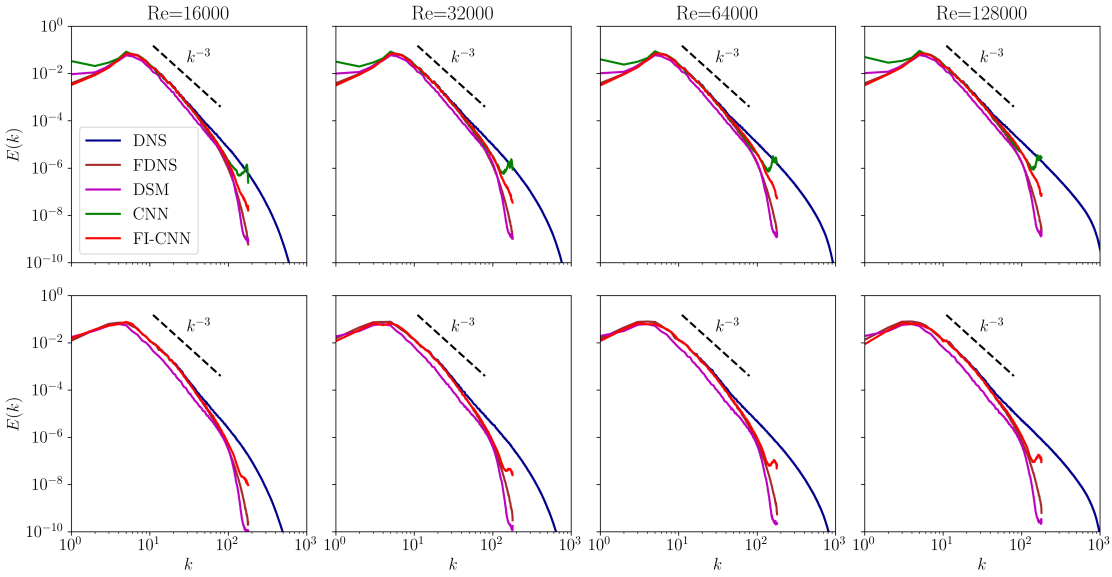

Figure 4.6: *A posteriori* kinetic-energy spectra for different Reynolds numbers at $t = 2.0$ (top row) and $t = 4.0$ (bottom row). These results are obtained from LES runs with five different initial conditions and only mean kinetic energy spectrum is shown. Note here that the CNN model has diverged and the kinetic-energy spectra for the CNN model is missing at the final time $t = 4.0$ (bottom row).

Figs. 4.8-4.11 provides the visualization of vorticity field and probability density function (PDF) of vorticity increments for Reynolds number Re = 16000 to Re = 128000 computed using the filtered DNS solution, LES with DSM model, and LES with $\mathcal{M}_{\text{FI-CNN}}$ at final time $t = 4.0$. We remark here that these results correspond to only one initial condition that is different from the one used for training the neural network-based SGS models. Even though the LES with DSM model is successful in capturing large-scale structures in the flow, it fails to capture the small-scale structure due to excessive dissipation. The LES with $\mathcal{M}_{\text{FI-CNN}}$ is able to capture both large- and small-scale structures in the flow, and this can be ascertained to the stabilizing property of $\mathcal{M}_{\text{FI-CNN}}$ in the *a posteriori* deployment without any post-processing of the predicted SGS source term. Qualitatively, the vorticity field obtained from LES

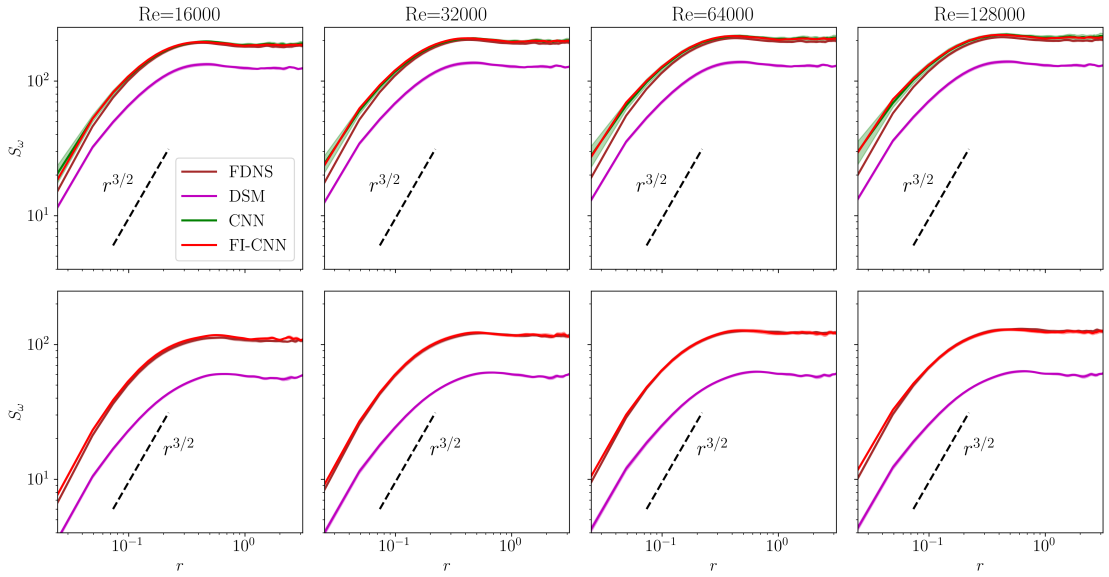

Figure 4.7: *A posteriori* second-order vorticity structure for different Reynolds numbers at $t = 2.0$ (top row) and $t = 4.0$ (bottom row). These results are obtained from LES runs with five different initial conditions and the solid line shows the mean vorticity structure and the shaded area corresponds to one standard deviation. Note here that the CNN model has diverged and the vorticity structure for the CNN model is not present at the final time $t = 4.0$ (bottom row).

with model $\mathcal{M}_{\text{FI-CNN}}$ is very similar to the filtered DNS solution. The similarity in the shape of the PDF of vorticity increments as shown in Figs. 4.8-4.11 suggests the scale-invariant statistics of turbulence at all Reynolds numbers investigated in this study. The shape of the PDF of vorticity increments predicted by the LES with $\mathcal{M}_{\text{FI-CNN}}$ matches with the shape of the filtered DNS solution, and the heavy exponential tails in the PDF are related to the presence of coherent vortices in the flow. These heavy exponential tails are missing in the PDF of the solution obtained from LES with DSM, and it follows the Gaussian distribution.

Next, we examine the robustness of the neural network-based SGS model by training an ensemble of neural networks using randomization-based approaches where different random initialization of weights are utilized for generating ensembles. Specifically, we train five neural networks for both models $\mathcal{M}_{\text{CNN}}$ and $\mathcal{M}_{\text{FI-CNN}}$ using the same dataset as discussed in Section 4.4. This method is also applied to quantify the model-form uncertainty in deep learning [192]. Fig. 4.12 shows the time evolution of turbulent kinetic energy and vorticity variance at different Reynolds numbers. The time evolution of the TKE in Fig. 4.12 implies that the weights of the neural networks

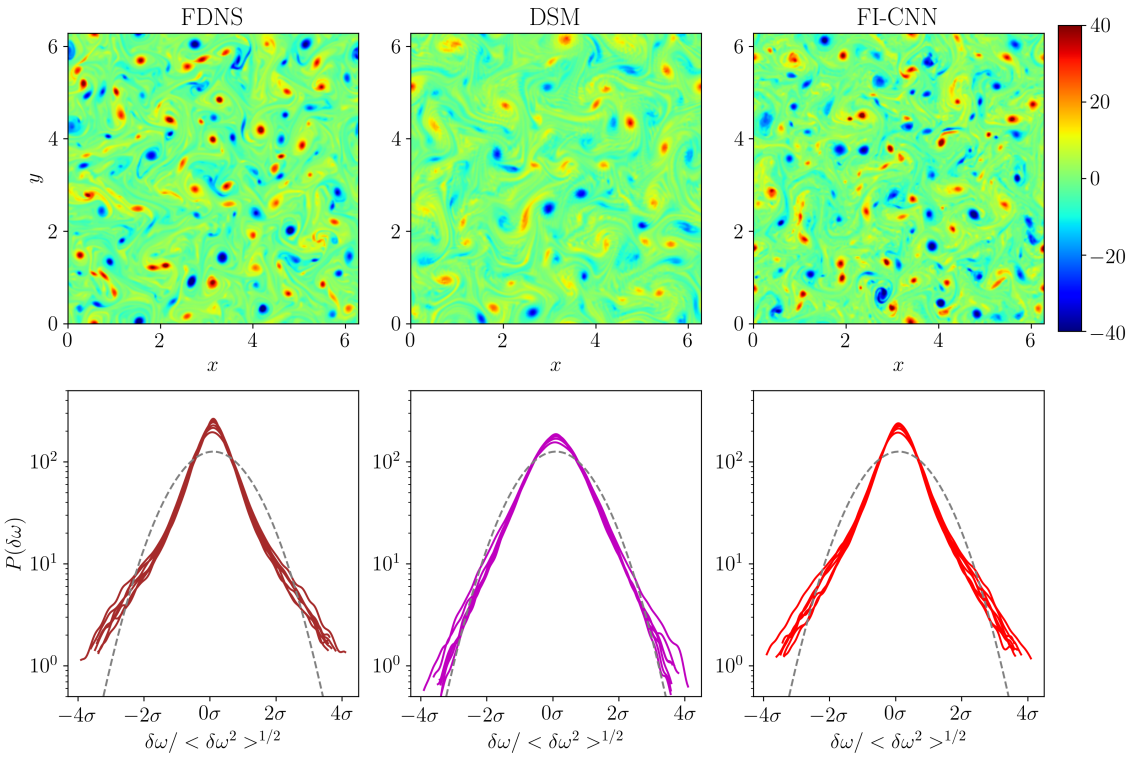

Figure 4.8: Snapshots of the vorticity distribution for Re = 16000 taken at final time $t = 4.0$ (top row) from different models and compared qualitatively against the FDNS solution. The bottom row displays the probability density function $P(\delta\omega)$ of the vorticity increments $\delta\omega$ for separations $r = 2\pi/256, 2\pi/128, 2\pi/64, 2\pi/32, 2\pi/16, 2\pi/8,$ and$2\pi/4$ computed from different models at Re = 16000. A Gaussian distribution is given in gray dashed line for comparison. We also note that the plain vanilla CNN becomes numerically unstable and unbounded before $t = 4.0$.

for $\mathcal{M}_{\text{FI-CNN}}$ are learned in such a way that the final models are overall dissipative in nature (as indicated by the solid line for mean from different LES runs). The vorticity variance predicted by the model $\mathcal{M}_{\text{FI-CNN}}$ is more accurate compared to DSM and is very close to the filtered DNS solution.

Fig. 4.13 depicts the kinetic-energy spectra at intermediate time $t = 2.0$ and at final time $t = 4.0$ obtained from LES runs for a single initial condition with different network-based SGS models for multiple Reynolds numbers. We observe the energy pile up near grid cutoff wavenumbers for the LES runs coupled with $\mathcal{M}_{\text{CNN}}$ and this suggests that the solution is unphysical. This behavior is also demonstrated in Fig. 4.14 through a large value of vorticity structure function at $t = 2.0$ across all Reynolds numbers. The LES runs with $\mathcal{M}_{\text{CNN}}$ has diverged around $t \approx 2.5$ (as seen by large TKE in Fig. 4.12), and, therefore the kinetic-energy spectra and vorticity

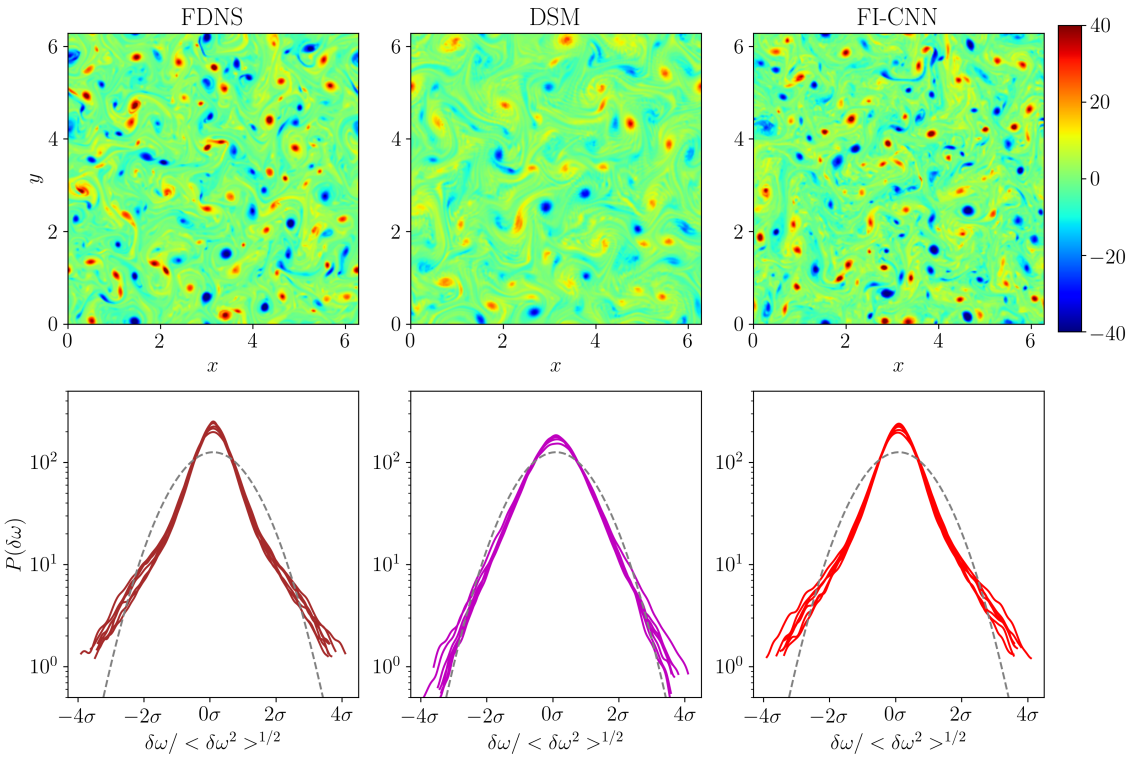

Figure 4.9: Snapshots of the vorticity distribution for Re = 32000 taken at final time $t = 4.0$ (top row) from different models and compared qualitatively against the FDNS solution. The bottom row displays the probability density function $P(\delta\omega)$ of the vorticity increments $\delta\omega$ for separations $r = 2\pi/256, 2\pi/128, 2\pi/64, 2\pi/32, 2\pi/16, 2\pi/8,$ and$2\pi/4$ computed from different models at Re = 32000. A Gaussian distribution is given in gray dashed line for comparison. We also note that the plain vanilla CNN becomes numerically unstable and unbounded before $t = 4.0$.

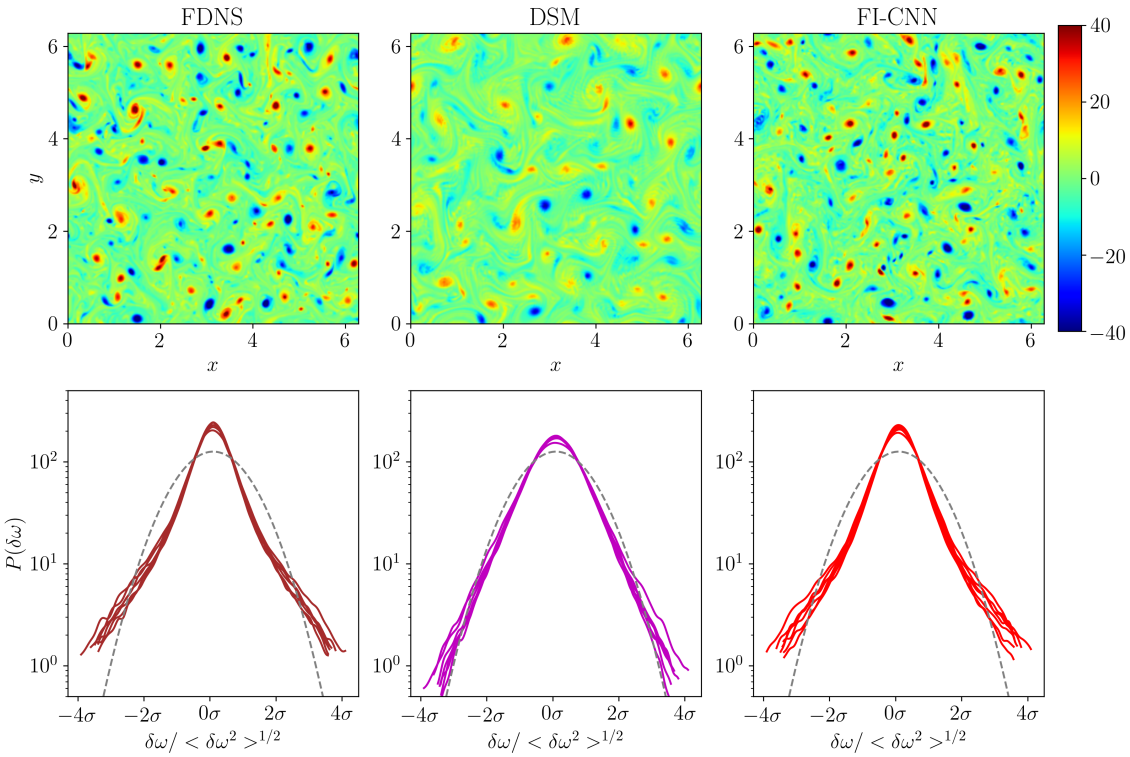

Figure 4.10: Snapshots of the vorticity distribution for Re = 64000 taken at final time $t = 4.0$ (top row) from different models and compared qualitatively against the FDNS solution. The bottom row displays the probability density function $P(\delta\omega)$ of the vorticity increments $\delta\omega$ for separations $r = 2\pi/256, 2\pi/128, 2\pi/64, 2\pi/32, 2\pi/16, 2\pi/8,$ and$2\pi/4$ computed from different models at Re = 64000. A Gaussian distribution is given in gray dashed line for comparison. We also note that the plain vanilla CNN becomes numerically unstable and unbounded before $t = 4.0$.

structure function are missing at $t = 4.0$ in Fig. 4.13 and Fig. 4.14, respectively. The kinetic-energy spectra for LES runs with $\mathcal{M}_{\text{FI-CNN}}$ is highly accurate and shows an excellent agreement with the filtered DNS solution, especially in the inertial subrange. The small uncertainty band also suggests that an ensemble of neural networks have produced very similar statistics for $\mathcal{M}_{\text{FI-CNN}}$. Fig. 4.14 shows that the model $\mathcal{M}_{\text{FI-CNN}}$ is successful in capturing the $r^{3/2}$ scaling for the vorticity structure function at small scales and flattening near large scales. With this numerical experiment, we can establish that the $\mathcal{M}_{\text{FI-CNN}}$ is robust, trustworthy, and stable in the LES, and it also ensures generalizable learning across different initial conditions and Reynolds numbers.

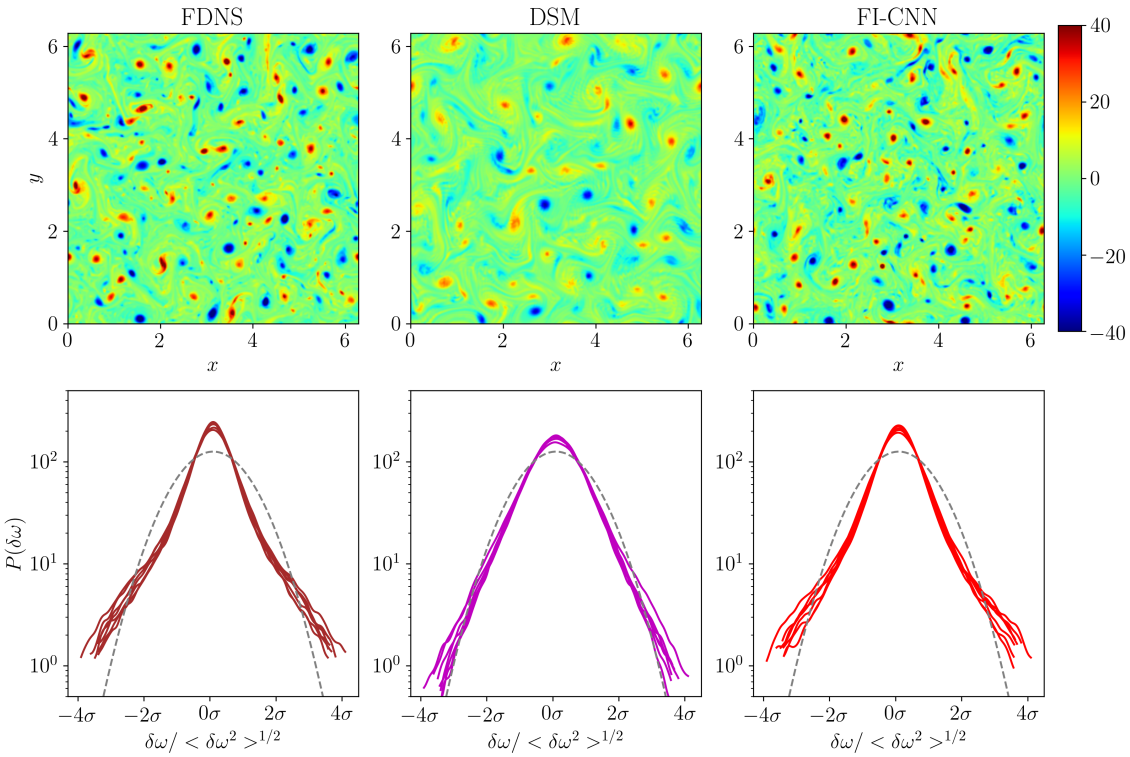

Figure 4.11: Snapshots of the vorticity distribution for Re = 128000 taken at final time $t = 4.0$ (top row) from different models and compared qualitatively against the FDNS solution. The bottom row displays the probability density function $P(\delta\omega)$ of the vorticity increments $\delta\omega$ for separations $r = 2\pi/256, 2\pi/128, 2\pi/64, 2\pi/32, 2\pi/16, 2\pi/8,$ and$2\pi/4$ computed from different models at Re = 128000. A Gaussian distribution is given in gray dashed line for comparison. We also note that the plain vanilla CNN becomes numerically unstable and unbounded before $t = 4.0$.

4.6 Concluding Remarks

Closure modeling in fluid dynamics simulations refers to parameterizing the interactions between high-fidelity and coarse-fidelity descriptions. In this study, we explore data-driven closure modeling strategies to improve both the accuracy and generalizability of such residual models. The motivation behind data-driven closure modeling stems from the fact that most of the existing SGS models are derived based on physical and mathematical considerations, and might not account for the important transfer of kinetic energy from small scales to large scales (i.e., back-scatter) [140]. However, pure data-driven models can lead to physically inconsistent results [164] and might violate the known physical constraints. To address this limitation of pure data-driven models, we apply a frame invariant neural network architecture aiming at embedding

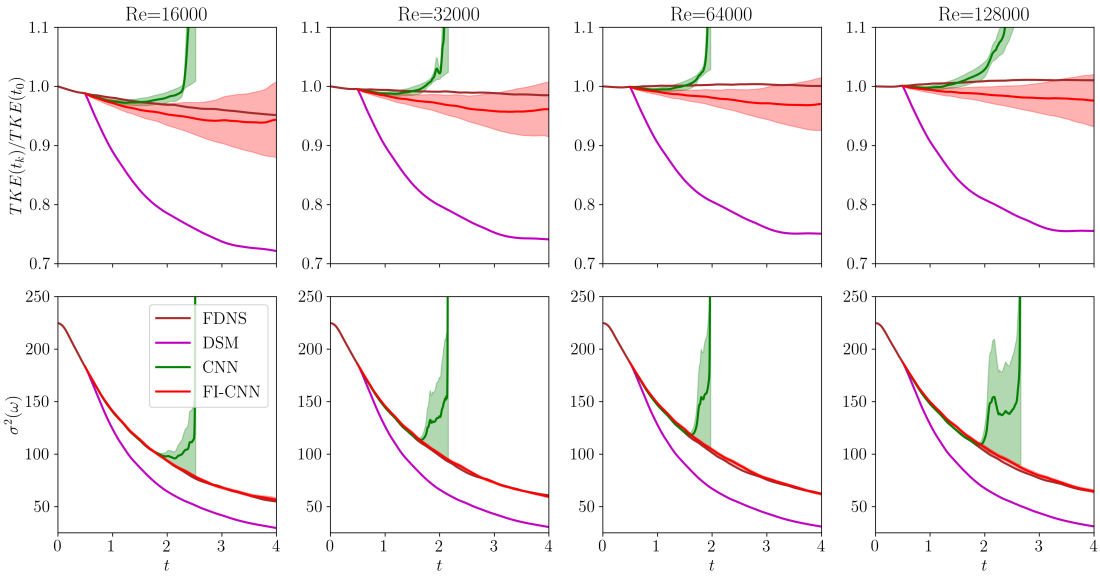

Figure 4.12: The time evolution of the turbulent kinetic energy $TKE(t_k)$ normalized by the initial turbulent kinetic energy $TKE(t_0)$ (top row) and vorticity variance (bottom row) for different Reynolds numbers at 256^2 grid resolution. The LES simulation starts at $t = 0.5$ after the initial spin-up time (i.e., once the turbulence has set in). For the CNN and FI-CNN models, an ensemble of neural networks is trained using the data generated from a single initial condition at Reynolds number Re = 16000 with different weights initialization. The solid line shows the mean from LES runs for a single initial condition with different trained networks and the shaded area corresponds to one standard deviation.

physical symmetries directly into the structure of the convolutional neural networks. Thus, our model theoretically guarantees the frame symmetries, including translation, Galilean, and rotation invariance both during training and inference. The embedding of physical symmetries as hard constraints not only improves the accuracy of the model but notably improves the generalization of the model, and eventually makes the model stable in their *a posteriori* deployment without any clipping.

We test the proposed framework for subgrid-scale modeling of Kraichnan turbulence in *a priori* and *a posteriori* settings. The performance of the proposed framework is evaluated using several metrics like kinetic energy spectra, vorticity structure, and vorticity increments. Based on our analysis, we concluded that symmetry preservation has the potential to improve the accuracy, generalizability, and stability of the SGS model, besides embedding important geometric properties of the underlying PDEs into deep learning models. This work also illustrates a broader lesson on how to combine machine learning with physics for scientific computing. It may be argued

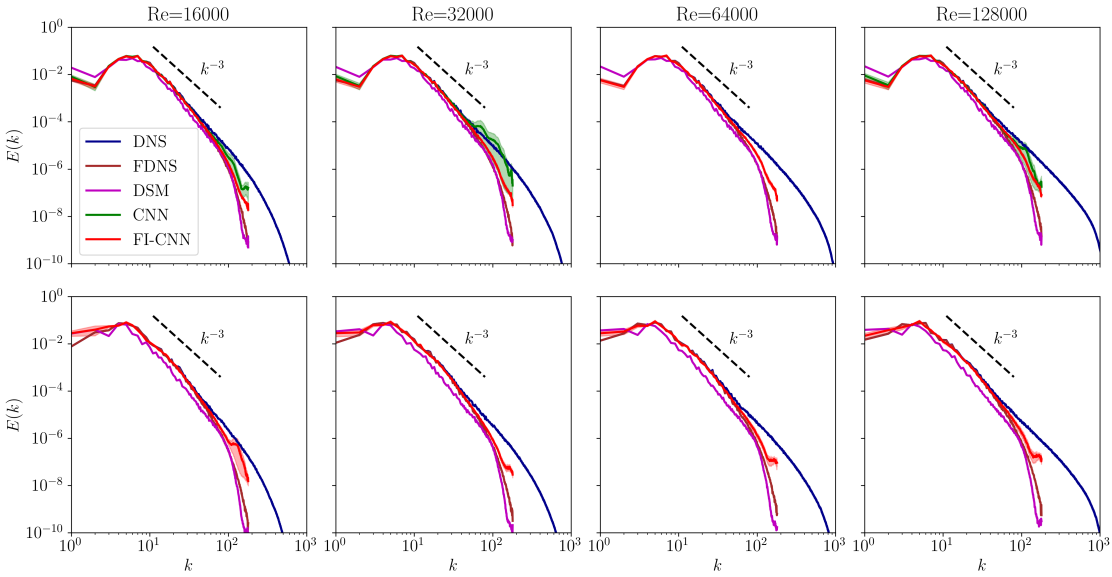

Figure 4.13: *A posteriori* kinetic-energy spectra for different Reynolds numbers at $t = 2.0$ (top row) and $t = 4.0$ (bottom row). The solid line shows the mean from LES runs for a single initial condition with different trained networks and the shaded area corresponds to one standard deviation. Note here that the CNN model has already diverged and the kinetic-energy spectra for the CNN model is missing at the final time $t = 4.0$ (bottom row).

that two-dimensional turbulence is far from reality. However, it is generally considered as a canonical testbed for geophysical turbulence in the atmosphere and oceans. Our future development will be focused on scaling up the proposed frame invariant closure modeling framework to solve more realistic three-dimensional turbulent flows. Another interesting avenue is to apply this framework for learning parameterization models for realistic geophysical flows with stratification, the Earth rotation, and Beta plane effects, paving the way for improved weather and climate prediction.

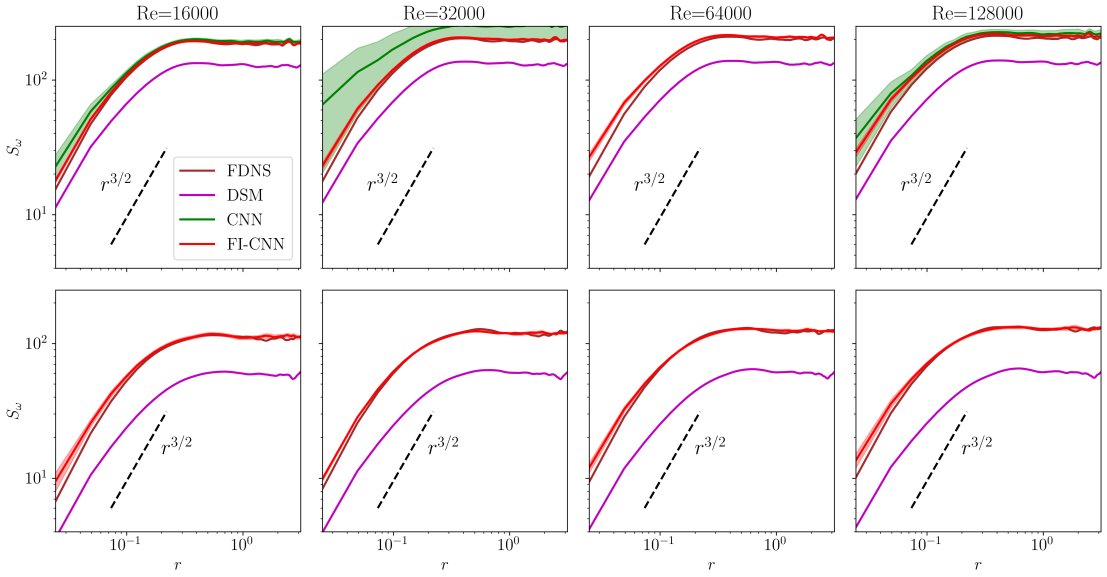

Figure 4.14: *A posteriori* second-order vorticity structure for different Reynolds numbers at $t = 2.0$ (top row) and $t = 4.0$ (bottom row). The solid line shows the mean from LES runs for a single initial condition with different trained networks and the shaded area corresponds to one standard deviation. Note here that the CNN model has diverged and the vorticity structure for the CNN model is not present at the final time $t = 4.0$ (bottom row).

CHAPTER V

Concatenated Neural Networks for Multi-fidelity Information Fusion

The contents of this chapter has been published at Scientific Reports[1].

Abstract: Recently, computational modeling has shifted towards the use of statistical inference, deep learning, and other data-driven modeling frameworks. Although this shift in modeling holds promise in many applications like design optimization and real-time control by lowering the computational burden, training deep learning models needs a huge amount of data. This big data is not always available for scientific problems and leads to poorly generalizable data-driven models. This gap can be furnished by leveraging information from physics-based models. Exploiting prior knowledge about the problem at hand, this study puts forth a physics-guided machine learning (PGML) approach to build more tailored, effective, and efficient surrogate models. For our analysis, without losing its generalizability and modularity, we focus on the development of predictive models for laminar and turbulent boundary layer flows. In particular, we combine the self-similarity solution and power-law velocity profile (low-fidelity models) with the noisy data obtained either from experiments or computational fluid dynamics simulations (high-fidelity models) through a concatenated neural network. We illustrate how the knowledge from these simplified models results in reducing uncertainties associated with deep learning models applied to boundary layer flow prediction problems. The proposed multi-fidelity information fusion framework produces physically consistent models that attempt to achieve better generalization than data-driven models obtained purely based on data. While we demonstrate our framework for a problem relevant to fluid mechanics, its workflow and principles can be adopted for many scientific problems where empirical, analytical, or simplified models are prevalent. In line with grand demands in novel PGML principles, this work builds a bridge between extensive physics-based theories and data-driven modeling paradigms and paves the way for using hybrid physics and machine learning modeling approaches for next-generation digital twin technologies.

[1]Pawar, S., San, O., Vedula, P., Rasheed, A., & Kvamsdal, T. (2022). Multi-fidelity information fusion with concatenated neural networks. Scientific Reports, 12(1), 1-13.

5.1 Introduction

The modeling of spatiotemporal dynamics of multiscale and multiphysics systems is an open problem relevant to many scientific and engineering applications. For instance, wind energy is a highly complex system whose dynamics is governed by global atmospheric processes to turbulent boundary layer formed around the blades that span nine orders of magnitudes [379]. Over the past several decades, we have improved our understanding of such multiphysics systems by developing accurate numerical models for governing equations of the system, such as Navier-Stokes equations for fluid flows. However, these numerical models can be computationally prohibitive, especially for nonlinear multiscale systems, and their use in real-time optimization and control is scarce. The recent advancement in machine learning (ML) and deep learning (DL) holds the great potential for tackling the challenge of modeling and analysis of high-dimensional systems and has been successful in diverse applications, such as fluid mechanics [51], earth science [318], and material science [342]. These advances have been driven by a vast amount of data generated from high-resolution numerical simulations, experimental and satellite measurements, and computing power along with the emergence of effective and efficient algorithms that can extract relevant patterns from the data. ML/DL techniques has been successfully applied for turbulence closure modeling [87], super-resolution of climate data [358], predicting clustered weather patterns [61], reduced-order modeling [104], and many more.

ML/DL models are capable of providing insights from data, exploiting these insights in building predictive tools, and continuously updating themselves as the new streams of data get available. Despite these advantages, ML/DL techniques lack interpretability and suffer from the curse of dimensionality. The interpretability issue can be addressed by understanding the physical implications of ML/DL models [240], and understanding the neural network correlations discovered from the data [255, 89]. By the curse of dimensionality, we mean that DL models are data-hungry in nature. For instance, Bonavita and Laloyaux [41] showed that the amount of the training data for nonlinear dynamical systems grows exponentially with the dimensionality of the system. Furthermore, pure ML/DL models lead to poor extrapolation/generalization, i.e., they fit the observations data very well, but predictions may be poor and physically inconsistent for data beyond the distribution of the training dataset. To this end, physics-informed learning algorithms that leverage prior knowledge based on the physical and mathematical understanding of the system are proposed in several studies

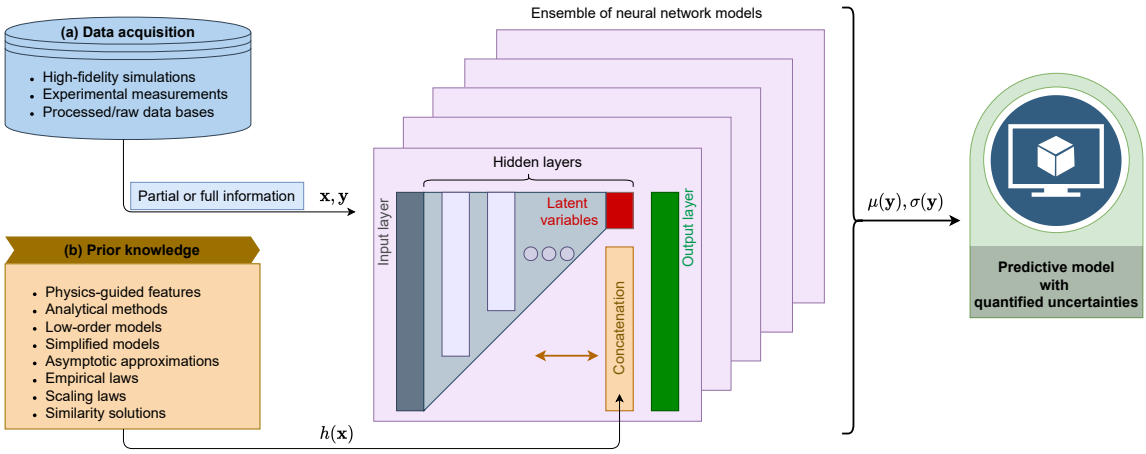

Figure 5.1: The proposed multi-fidelity information fusion framework. The prediction from the low-fidelity models (e.g., self-similarity solution or law of the wall) is concatenated with the latent variables at the certain hidden layer of the neural network. An ensemble of neural networks is trained using the negative log-likelihood loss function to estimate the uncertainty associated with the prediction. Here, $\mathbf{x}$ refers to independent variables or design parameters, or $\mathbf{y}$ indicates the quantities of interest, and $h(\mathbf{x})$ represents a low-fidelity or simplified model that is fast-to-compute.

[164].

There are two main techniques based on inductive biases, and learning biases to embed physics into ML/DL models in combination with the observations data [164]. Inductive biases techniques relate to building tailored ML/DL model architecture that exploits the prior knowledge about the problem at hand to build physically consistent data-driven models. The representative examples includes embedding invariance property into neural network architecture [213, 421], imposing conservation laws of physical quantities or analytical constraints into neural network [249, 34, 126], using prediction from simplified models as the bias [287], enhancing feature space [238], and adopting equivariant transformer networks [367]. The other approach based on learning biases imposes constraints such as governing equations in a soft manner by penalizing the loss function of ML/DL models. Some of the examples of this approach pertain to physics-informed neural networks [309], statistically constrained generative adversarial networks [405], and Bayesian framework with auto-regressive model [116]. There is also a class of hybrid analysis and modeling approaches that utilizes pure data-driven and physics-based models in tandem [335]. While many methods have been demonstrated to be successful in enforcing physics into ML/DL models, they offer many possibilities to fuse domain knowledge to improve the generalizability and data efficiency of data-driven models [167].

In this work, we introduce a physics-guided machine learning (PGML) fraamework based on tailored neural network architecture that utilizes a concatenation layer for surrogate modeling of high-fidelity data given the solution from a computationally inexpensive low-fidelity model. There is a hierarchy of numerical models ranging from simple empirical relations to highly accurate numerical discretization-based models across all scientific disciplines. For example, flow around airfoils can be modeled using tools extending from panel methods [139] to the fully resolved direct numerical simulation (DNS) [252]. Another example is the wake modeling of wind turbines, where fast but typically inaccurate analytical models are adopted for tasks like layout optimization and wind farm control [13]. However these models are insufficient to take the unsteady nature of interactions of turbine wake with other wakes as well as atmospheric turbulence into account, and such effects can be modeled with computationally demanding but accurate models like large-eddy simulation (LES) [49]. Our work draws inspiration from these multi-fidelity modeling approaches and exploits the real-time prediction from low-fidelity models to inform a DL model of high-fidelity observations. The information fusion from the multi-fidelity sources of data leads to a robust and generalizable surrogate model in comparison to a purely data-driven model trained solely on high-fidelity data.

Figure 5.1 graphically illustrates the proposed data fusion approach from multi-modal data streams in the process of generating PGML models. As shown in Figure 5.1, a data-driven model can be constructed using the data obtained from high-resolution simulations and experimental measurements. The PGML framework further augments the ML model with the information from a low-fidelity representation of the system (such as analytical models, scaling laws, empirical relations, etc.). Ensemble of PGML models is trained using both the sources of data along with the uncertainty quantification mechanism that finally gives the predictive model to be employed in online tasks. In a nutshell, our work puts forth a novel data-driven framework to take prior knowledge about the system into account when generating a black-box deep learning predictive model. How should we inject physics and domain knowledge into machine learning models? How deep learning can be constructed as a trustworthy approach toward more accurate real-time prediction of nonlinear complex systems such as turbulent flows? These are the fundamental research questions that we tackle in this paper, and provide our insights about these questions.

We demonstrate the application of our framework for boundary layer flows. Boundary layer phenomenon is one of the most important flows and is of engineering concern

in many scientific and industrial applications [351]. The behavior of flow in the boundary layer has implications on the drag force in ship hulls and aircraft, the energy required to move oil through pipes, and the distribution of heat in the atmosphere [159, 36, 410]. Given that boundary layer flows are prevalent in engineering applications, building a computationally efficient and accurate surrogate model is of paramount importance for online tasks like boundary layer control to achieve lift enhancement, noise mitigation, drag reduction, and wall cooling. Additionally, boundary layer flows can be described using hierarchies of models that have different levels of fidelity spanning analytical models to DNS and hence represent an interesting test case for illustrating the effectiveness of PGML framework. While in this work we consider only two levels of fidelity, the proposed framework can be applied for blending information from various levels of fidelity. Moreover, the neural architecture search tools can be utilized to discover more complex and optimal architectures automatically [92].

5.2 Methods

In this Section, we first provide the details of the concatenated neural network architecture as a PGML framework that blends the information from models of different levels of fidelity and then present the deep ensemble method used to quantify the uncertainty associated with the prediction. Then, we discuss the multi-fidelity data fusion for laminar and turbulent boundary layer flows over a flat plate.

5.2.1 Multi-fidelity Concatenated Neural Network

A neural network is a computational graph composed of several layers consisting of the predefined number of neurons. Each neuron is associated with certain coefficients called weights and some bias. The input from the previous layer is multiplied by a weight matrix as shown below

$$\mathbf{S}^l = \mathbf{W}^l \mathbf{X}^{l-1}, \tag{5.1}$$

where $\mathbf{X}^{l-1}$ is the output of the $(l-1)$th layer, $\mathbf{W}^l$ is the matrix of weights for the lth layer, and $\mathbf{S}^l$ is the input-weight product. 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 lth layer can be written as

$$\mathbf{X}^l = \zeta(\mathbf{S}^l + \mathbf{B}^l), \tag{5.2}$$

where $\mathbf{B}^l$ is the vector of biasing parameters for the lth layer and ζ is the activation function. If there are N_L layers between the input and the output in a neural network, then the neural network mapping $\mathcal{F} : \mathbf{X} \to \mathbf{Y}(\mathbf{X})$ can be represented as follows

$$\mathbf{Y} = \zeta_{N_L}(\cdot; \mathbf{\Theta}_{N_L}) \circ \cdots \circ \zeta_2(\cdot; \mathbf{\Theta}_2) \circ \zeta_1(\mathbf{X}; \mathbf{\Theta}_1) \tag{5.3}$$

where $\mathbf{\Theta}$ represents the weight and bias of the corresponding layer of the neural network and $\mathbf{Y}$ is the final output of the neural network. For the concatenated neural network, the information from the low-fidelity model is injected at a certain intermediate layer of the neural network as follows

$$\mathbf{Y} = \zeta_{N_L}(\cdot; \mathbf{\Theta}_{N_L}) \circ \cdots \circ \underbrace{\mathcal{C}(\zeta_i(\cdot; \mathbf{\Theta}_i), h(\mathbf{X}))}_{\text{Concatenation layer}} \circ \cdots \circ \zeta_1(\mathbf{X}; \mathbf{\Theta}_1), \tag{5.4}$$

where $\mathcal{C}(\cdot, \cdot)$ represents the concatenation operation and the information from the low-fidelity model, i.e., $h(\mathbf{X})$ is injected at ith layer, and $\mathbf{\Theta}_i$ are the trainable parameters of the corresponding layer. The concatenation operator takes the latent variables at a particular layer and combines them with information from the low-fidelity model to return a vector. Specifically, if the ith layer has D_i neurons and $h(\mathbf{X}) \in \mathbb{R}^{D_h}$, the input to the $(i+1)$th layer will be in $\mathbb{R}^{D_i + D_h}$. During the training of the neural network, the weights of the neural network are updated using the backpropagation and gradient descent algorithm. The backpropagation algorithm involves computation of the gradient of the loss function with respect to each of the trainable parameters and the trainable parameters are updated using a gradient descent algorithm. In a concatenated neural network, the prediction from the low-fidelity model is injected at an intermediate layer of the network. Therefore, the low-fidelity model prediction is also the input feature to the network, and the training does not involve computing the gradient of the loss function with respect to this injected feature. However, the number of trainable parameters is increased and the change in the number of trainable parameters will depend on the dimension of the low-fidelity data and the number of neurons in the subsequent hidden layer.

One important caveat in a concatenated neural network is the selection of an

appropriate intermediate layer at which to inject the low-fidelity model information, and tools like automated machine learning (AutoML) [152] can be applied to automate this search. The concatenation operator given by Equation 5.4 can also be applied to all hidden layers simultaneously as

$$\mathbf{Y} = \zeta_{N_L}(\cdot;\boldsymbol{\Theta}_{N_L}) \circ \underbrace{\mathcal{C}(\zeta_{N_L-1}(\cdot;\boldsymbol{\Theta}_{N_L-1}), h(\mathbf{X})) \circ \cdots \circ \mathcal{C}(\zeta_i(\cdot;\boldsymbol{\Theta}_i), h(\mathbf{X})) \circ \cdots \circ \mathcal{C}(\zeta_1(\cdot;\boldsymbol{\Theta}_1), h(\mathbf{X}))}_{\text{Concatenation layers}}. \tag{5.5}$$

We highlight here that the proposed PGML framework is modular and Equation 5.5 can be generalized for fusing information from multiple low-fidelity models. However, we dedicate this study to investigate the feasibility of the proposed PGML framework for only two levels of approximations. An additional optimization problem can be constructed to search the best architecture in terms of relevant hyperparameters such as the number of hidden layers, and the location and sparsity of the concatenation structure. Such auto PGML investigations will be a topic that we will pursue in our future works.

5.2.2 Deep Ensembles: Training and Prediction

Deep learning algorithms like neural networks approximate the mapping from inputs to outputs using trainable parameters called weights and biases. The parameters of the neural network are determined through the minimization of the loss function. The prediction from the neural network is usually a point estimate, i.e., continuous outputs for regression tasks and discrete classes for classification problems. However, the information about the confidence in the model's prediction might be crucial for many scientific applications [5]. The uncertainty estimates can also be useful for applications like sensor placement and Bayesian optimization. In this study, we apply the deep ensembles algorithm for estimating the probabilistic distribution function (PDF) of output conditioned on the inputs [192]. While there are state-of-the-art methods like Bayesian neural networks [261] that quantifies uncertainty by learning the distribution of weights, deep ensembles is adopted due to their simplicity and scalability.

Here, we briefly discuss the uncertainty quantification mechanism of deep ensembles. We assume that our training dataset $\mathcal{D}$ consists of N samples $\mathcal{D} = \{\mathbf{X}_i, \mathbf{Y}_i\}_{i=1}^{N}$, where $\mathbf{X} \in \mathbb{R}^P$ represents the $P-$dimensional features and the label is $Q-$dimensional, i.e., $\mathbf{Y} \in \mathbb{R}^Q$. A neural network is trained by minimizing the loss function $\mathcal{L}(\mathbf{Y}, \tilde{\mathbf{Y}}(\mathbf{X};\underline{\Theta}))$, where $\tilde{\mathbf{Y}}$ is the predicted label from a neural network parameterized by $\underline{\Theta}$ (i.e.,

trainable parameters of the whole neural network). The most common loss function for regression tasks is the mean squared error (MSE) between true and predicted labels averaged over all samples in the dataset. The MSE loss function does not give an estimate of the probability distribution of $\mathcal{P}(\tilde{\mathbf{Y}}|\mathbf{X})$ and hence the uncertainty estimate is usually absent with the prediction from neural networks.

In order to quantify the predictive uncertainty, the neural network is trained to output the mean and variance of the Gaussian distribution in the output layer. The weights of the neural network are determined by minimizing the negative log-likelihood $\mathcal{L}$ as follows

$$\underline{\Theta} = \arg\min_{\underline{\Theta}}[\mathcal{L}], \quad \text{where} \quad \mathcal{L} = \sum_{i=1}^{N} \frac{1}{2}\log\sigma^2(\mathbf{X}_i) + \frac{(\mathbf{Y}_i - \mu(\mathbf{X}_i))^2}{2\sigma^2(\mathbf{X}_i)}, \tag{5.6}$$

where the mean μ and the variance σ^2 are parameterized by the neural network. The positivity constraint is enforced for the variance by passing the output corresponding to variance of distribution though the *softplus* function $\log(1 + \exp(\cdot))$, and adding a minimum variance (for example 10^{-6}) for numerical stability. If we assume the variance to be *constant* in Equation 5.6 (i.e., it does not depend on input features), then the negative log-likelihood loss function becomes analogous to the MSE loss function. Therefore, from a probabilistic point of view, minimizing the MSE is equivalent to minimizing negative log-likelihood with an assumption of Gaussian distribution with constant standard deviation [76, 263].

The ensemble of neural networks has been demonstrated to be successful in improving the predictive performance of machine learning models [80]. There are broadly two methods of generating ensembles, (i) randomization-based approaches where the ensembles can be trained in parallel without any interaction, and (ii) boosting-based approaches where the ensembles are trained sequentially [99]. The randomization procedure for generating ensembles of neural networks should be such that prediction from individual models are de-correlated and each individual models are strong (i.e., high accuracy). In this work, the random initialization of weights of the neural network is used for generating ensembles. There are other schemes such as bagging where the ensembles of neural networks are trained on a different subset of the original training data. However, random initialization is better than bagging for improving predictive accuracy and uncertainty [192, 203]. This simple and yet robust randomization approach is highly scalable as it allows for distributed training

of neural networks and can be applied to many scientific problems. For computing the predictive probability distribution, we approximate the ensemble prediction as a Gaussian whose mean and variance are computed as follows

$$\mu_*(\mathbf{X}) = \frac{1}{M}\sum_{j=1}^{M}\mu_{\underline{\Theta}_j}(\mathbf{X}), \tag{5.7}$$

$$\sigma_*^2(\mathbf{X}) = \frac{1}{M}\sum_{j=1}^{M}(\sigma_{\underline{\Theta}_j}^2(\mathbf{X}) + \mu_{\underline{\Theta}_j}^2(\mathbf{X})) - \mu_*^2(\mathbf{X}), \tag{5.8}$$

where $\mu_{\underline{\Theta}_j}$ and $\sigma_{\underline{\Theta}_j}$ is the mean and standard deviation of predicted probability distribution by the jth neural network. We employ an ensemble of five neural networks in this study (i.e., $M = 5$).

5.2.3 Multi-fidelity Data Fusion for Laminar Boundary Layer

The first test case considered in this study is the laminar boundary layer flow. Boundary layer flows can be characterized by dividing the flow into two regions, one inside the boundary layer where the viscosity dominates and one outside the boundary layer where the effect of viscosity can be neglected. The low-fidelity model considered for laminar flow is the steady-state two-dimensional laminar boundary layer described using Blasius equation [400]. The core idea behind Blasius equation is transforming a partial differential equation (PDE) comprised of the flat plate boundary layer equations, with zero pressure gradient, into a single ordinary differential equation (ODE) by using a similarity solution approach. The derivation of the Blasius equation can be found in many texts on fluid mechanics and we describe only the final form. The Blasius equation and its boundary conditions can be written as

$$2f''' + ff'' = 0, \tag{5.9}$$

$$f(0) = f'(0) = 0,\ f'(\infty) = 1, \tag{5.10}$$

where $f(\eta)$ is a function of similarity variable η. The similarity variable η is defined as $\eta = y\sqrt{u_\infty/(x\nu)}$, where y is the direction normal to the plate, x is the direction along its length with zero being the leading edge, u_∞ is the freestream velocity, and ν is the kinematic viscosity of the fluid. The third-order ODE is first split into a coupled system of three first-order ODEs. Then we apply the shooting method to determine the initial value for $f''(0)$, and the first-order ODEs are numerically integrated with

the fourth-order Runge-Kutta scheme [251]. The velocity profile from the Blasius solution can be determined using the relation $\bar{u} = u_\infty f'$, where overbar symbol is used to indicate the low-fidelity model estimate. The high-fidelity observations are generated by solving the RANS equations with the PISO algorithm [155] available in OpenFoam. We get the velocity (components along streamwise and wall-normal directions) and the pressure distribution from CFD simulation. The Reynolds number based on the length of the flat plate used for generating data is $\mathrm{Re}_L = 5 \times 10^4$, where L is the length of the flat plate. The training data for the concatenated neural network is sampled from the whole domain and the velocity field is contaminated by adding a white Gaussian noise with zero mean and a standard deviation of 0.05. Advanced sampling methods like Latin hypercube sampling, clustered sampling can be utilized to reduce the number of samples required for training and we will consider this as part of our future work.

For the laminar boundary layer reconstruction task, the input to the neural network is the location of the sensor, i.e., $\mathbf{X} = [x, y]$, where x and y are the positions of the sensors in streamwise and wall-normal directions. The output of the neural network is the probability distribution of u, v, p represented by their mean and standard deviation, where u is the velocity in the streamwise direction, v is the velocity in the wall-normal direction, and p is the pressure at the sensor's location. Additionally, the velocity profile obtained from the Blasius solution is used as the low-fidelity model, i.e., $h(\mathbf{X}) = [\bar{u}]$. Following our previous discussion, the problem formulation can be written as

$$\{\mu(\mathbf{X}), \sigma(\mathbf{X})\} = \mathcal{F}(\mathbf{X}, h(\mathbf{X})), \quad \mathcal{P}(\mathbf{Y}|\mathbf{X}, h(\mathbf{X})) = \mathcal{N}(\mu_*(\mathbf{X}), \sigma_*(\mathbf{X})). \tag{5.11}$$

5.2.4 Multi-fidelity Data Fusion for Turbulent Boundary Layer

The boundary layer around the flat plate transitions to turbulence at high Reynolds number. Before the advent of supercomputing, it was not possible to numerically solve the Navier-Stokes equations for turbulent flows and fluid dynamicists had to resort to experimental studies to derive empirical relations for high Reynolds number flows. In this study, the low-fidelity approximation for the turbulent boundary layer is obtained using the one-seventh power law. The one-seventh power law [59] for computing the mean (or ensemble-averaged) velocity profile for flat-plate turbulent boundary layer is

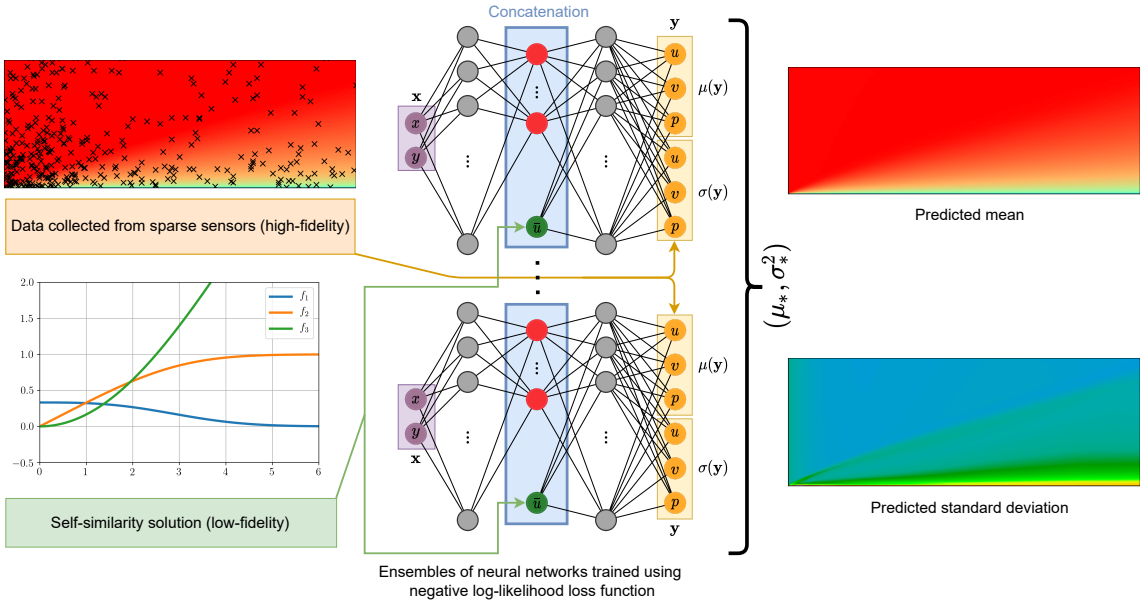

Figure 5.2: Illustration of the multi-fidelity data-fusion framework applied to the laminar flat plate boundary layer prediction task. The self-similarity Blasius solution is replaced by the one-seventh power law when applying to the turbulent boundary layer flows.

given as follows

$$\frac{\bar{u}}{u_\infty} \approx \begin{cases} \left(\frac{y}{\delta}\right)^{1/7} & \text{for} \quad y \leq \delta, \\ 1 & \text{for} \quad y > \delta, \end{cases} \tag{5.12}$$

where u_∞ is the freestream velocity, y is the direction normal to the plate, and the turbulent boundary layer thickness δ is computed as follows [59]

$$\delta \approx \frac{0.38x}{(\mathrm{Re}_x)^{1/5}}, \tag{5.13}$$

where Re_x is the Reynolds number at a given x-location. There are many such empirical relations available to approximate turbulent boundary layers such as the log law, and Spalding's law of the wall [352]. The high-fidelity data is generated for the flat-plate turbulent boundary layer with zero pressure gradient by solving the incompressible RANS equations with the SIMPLE algorithm and $k-\omega$-SST turbulent model implemented in OpenFoam. The Reynolds number based on the length of the flat plate for turbulent boundary layer simulation is $\mathrm{Re}_L = 1 \times 10^7$. One important parameter in turbulence modeling is the dimensionless distance in the normal direction called wall y^+ and is defined as $y^+ = \sqrt{yu_\tau/\nu}$, where u_τ is the friction velocity. The friction velocity is calculated based on the wall shear stress as $u_\tau = \sqrt{\tau_w/\rho}$. The

mesh is refined near the flat plate in such a way that the near-wall y^+ is below 5. The locations for collecting the data are sampled in such a way that more points are clustered near the leading edge and the wall, and are contaminated by adding a white Gaussian noise with zero mean and a standard deviation of 1.0 to mimic the measurement error. The formulation of turbulent boundary layer reconstruction is similar to the laminar boundary reconstruction as given in Equation 5.11 except for the low-fidelity model. The low-fidelity model prediction for turbulent boundary layer flow is calculated using Equation 5.12.

For both laminar and turbulent boundary layer reconstruction problem, we use a neural network with three hidden layers and twenty neurons in each hidden layer. The three hidden layers were found to be enough for providing sufficiently accurate prediction and hence we chose this architecture to reduce the computational overhead of training and inference. The prediction from the low-fidelity model is concatenated at the second hidden layer. Specifically, the equation for the concatenated neural network employed in this study can be written as follows

$$\mathbf{Y} = \zeta_4(\cdot;\boldsymbol{\Theta}_4) \circ \zeta_3(\mathbf{X};\boldsymbol{\Theta}_3) \circ \underbrace{\mathcal{C}(\zeta_2(\cdot;\boldsymbol{\Theta}_2), h(\mathbf{X}))}_{\text{Concatenation layer}} \circ \zeta_1(\mathbf{X};\boldsymbol{\Theta}_1), \tag{5.14}$$

where $\zeta_1,\ \zeta_2,\ \zeta_3$ are the ReLU activation functions, and ζ_4 is the linear activation function. The neural network architecture shown in Figure 5.2 is representative of the network used within the proposed multi-fidelity data-fusion framework for the boundary layer reconstruction task. In terms of the trainable parameters, the ML model has 1,026 parameters, and the PGML model has 1,046 parameters.

5.3 Results and Discussion

In this section, we demonstrate the capability of the proposed approach presented in Methods section to reconstruct laminar and turbulent boundary layer flows around the flat plate.

Laminar flow past a flat plate

We refer to a simple feed-forward neural network as the machine learning (ML) model and the concatenated neural network augmented with low-fidelity data is called the physics-guided machine learning (PGML) model. The ML model is trained solely

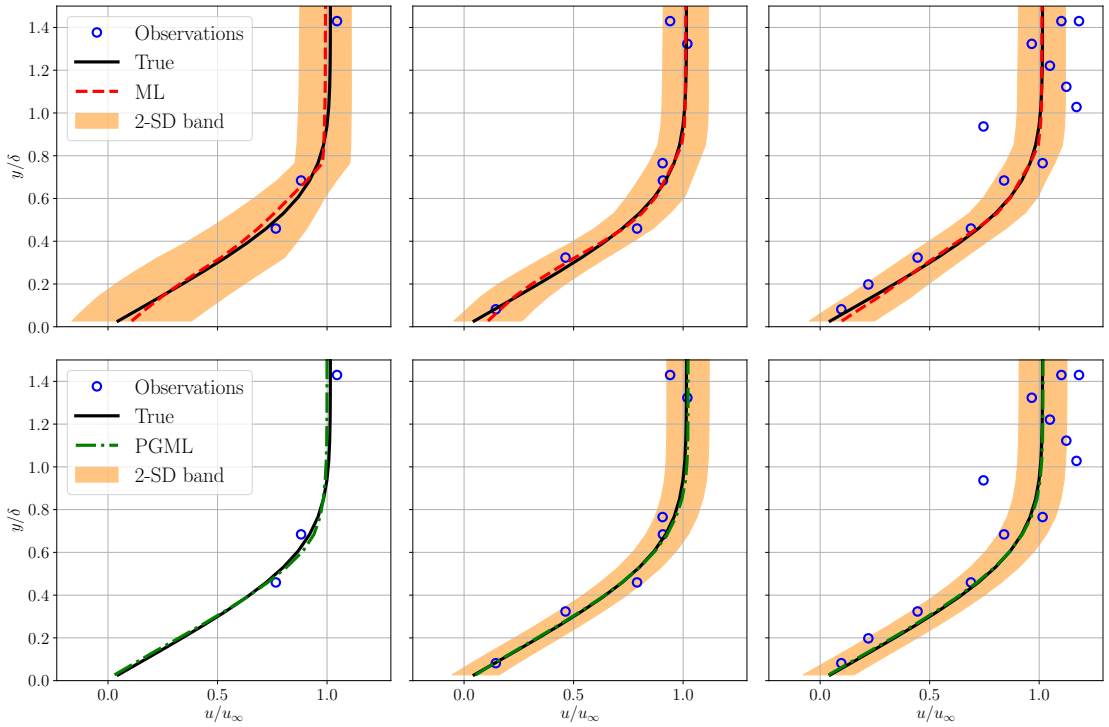

Figure 5.3: Boundary layer prediction for laminar flat plate flow at $x/L = 0.5$ along with the observations used for training the ML and PGML model. The amount of observations data used for training the model is 10% (left), 30% (middle), and 50% (right). The shaded area corresponds to two standard deviation (2-SD) band.

based on the high-fidelity data, while the PGML model uses the prediction from a physics-based low-fidelity model (Blasius equations, see Equation 5.9 and 5.10) along with the high-fidelity data. Figure 5.3 shows the profile of the horizontal component of velocity versus distance from the wall, at the location $x/L = 0.5$, for different amounts of data used for training the ML and PGML models. The velocity is normalized with the freestream velocity and the vertical distance is normalized using the boundary layer thickness for laminar flow over the flat plate. The ML model fails to capture the accurate velocity profile when the velocity field information at 10% of locations within the computational domain is utilized for training. The mean velocity profile predicted by the PGML model is highly accurate even with just 10% of the observations. The predicted mean velocity profile is also accompanied by a confidence interval spanning one standard deviation (SD) on either side of the mean velocity profile and is an outcome of the uncertainty quantification mechanism built into deep ensembles. Deep ensembles achieve uncertainty quantification by training an ensemble of neural networks with the negative log-likelihood loss function. The uncertainty

estimate associated with the PGML model is lower than the ML model and this is particularly notable near the wall within the boundary layer, i.e., for $y/\delta < 0.8$. However, we note that the prediction of laminar flow past a flat plate is a relatively simple task and therefore even the ML model is giving sufficiently accurate prediction. The improvement in the prediction by the PGML framework is noticeable for the turbulent flow past a flat plate, which will be presented in the following section.

Turbulent flow past a flat plate

Next, we evaluate the performance of the proposed PGML approach for reconstruction of turbulent boundary layer flow over a (smooth) flat plate. The prior knowledge we concatenate in this case is the one-seventh power law velocity profile (i.e., see Equation 5.12). Figure 5.4 displays the variation of the normalized velocity profile in the vertical direction at $x/L = 0.5$ for different amounts of data used for training the ML and PGML models. We can observe that the ML model performs very poorly for the data-sparse regime (i.e., 5% and 10% of the observations). Such situations are very common in scientific applications where a collection of high-fidelity data either from experiments or numerical simulations can be prohibitive. The PGML model on the other hand leads to an accurate prediction by exploiting the correlation between low- and high-fidelity data. The prediction from the ML model is also not reliable as indicated by the high width of the confidence band. Figure 5.5 shows the velocity profile in the *near-wall region* to illustrate how well the boundary condition is satisfied by the prediction obtained from both ML and PGML models. The velocity field predicted by the ML model is highly inaccurate in the *near-wall region* even when 30% of the observations are available for training. The PGML model provides very accurate prediction even in the low-data regime, and the prediction improves further as more data is used for training. The PGML model captures the slope of the velocity profile at the wall with very high accuracy. The slope of the boundary layer profile $\partial u/\partial y$ at the wall determines the skin friction drag along the wall. This quantity is not predicted accurately with the ML model and this can lead to poor estimation of the quantity of interests like the total drag. The PGML model is successful in predicting the correct slope of the velocity profile at the wall and therefore will lead to a more accurate estimation of total drag.

Figures 5.6- 5.8 shows the spatial variation of the predicted mean of the velocity field, confidence interval of two standard deviations, and the error with respect to

the true velocity field near the wall region for 5%, 10%, and 30% of the training data, respectively. As the training data increases, the error decreases for both ML and PGML models. The confidence estimate associated with the PGML model is substantially higher (i.e., lower uncertainty) than the ML model for all three datasets. Moreover, the error of the PGML model is greatly reduced compared to the ML model. One other benefit of constructing a PGML approach is its modular nature that can provide an opportunity of bridging the gap between domain-specific knowledge and physics-agnostic models.

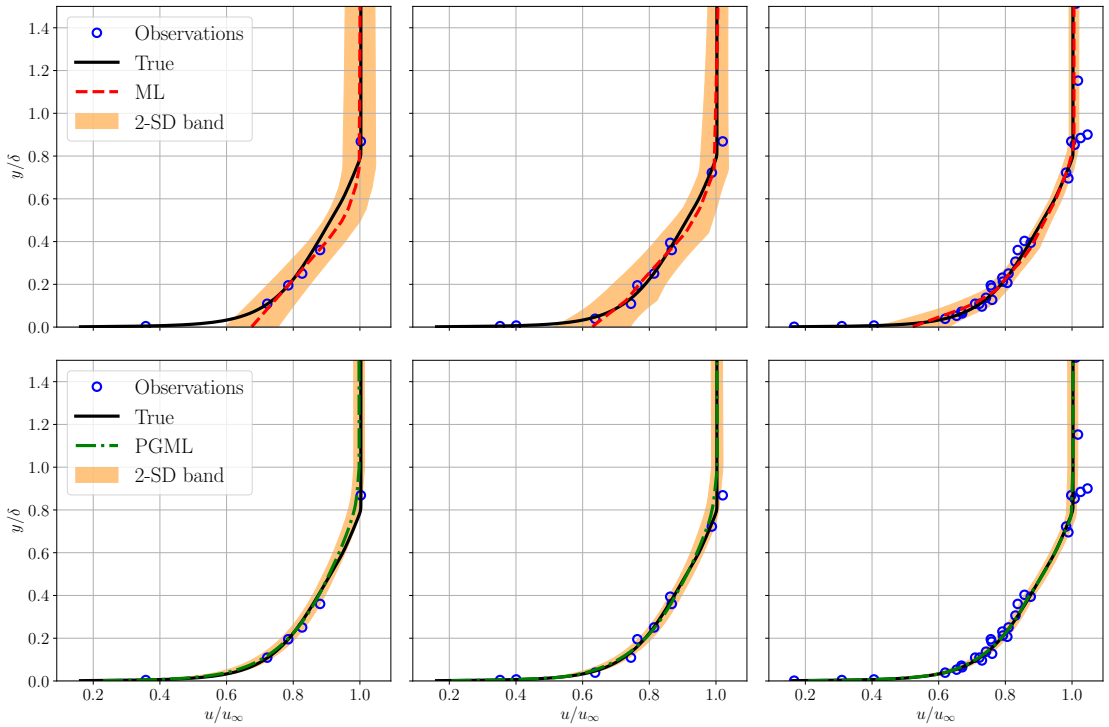

Figure 5.4: Boundary layer prediction for turbulent flat plate flow at $x/L = 0.5$ along with the observations used for training the ML and PGML model. The amount of observations data used for training the model is 5% (left), 10% (middle), and 30% (right). The shaded area corresponds to two standard deviation (2-SD) band.

In our previous numerical experiments, we focused on the reconstruction task within the interpolation region. Both ML and PGML models were trained using the data sampled from the whole domain, i.e., up to $L = 2.0$, where L is the length of the flat plate. In our next numerical experiments, we sample observations only from the region till $L = 1.5$. Therefore, the region between $L = 1.5$ to $L = 2.0$ corresponds to the extrapolation region. We quantify the performance of the ML and PGML model using the variation of root mean squared error (RMSE) percentage along the

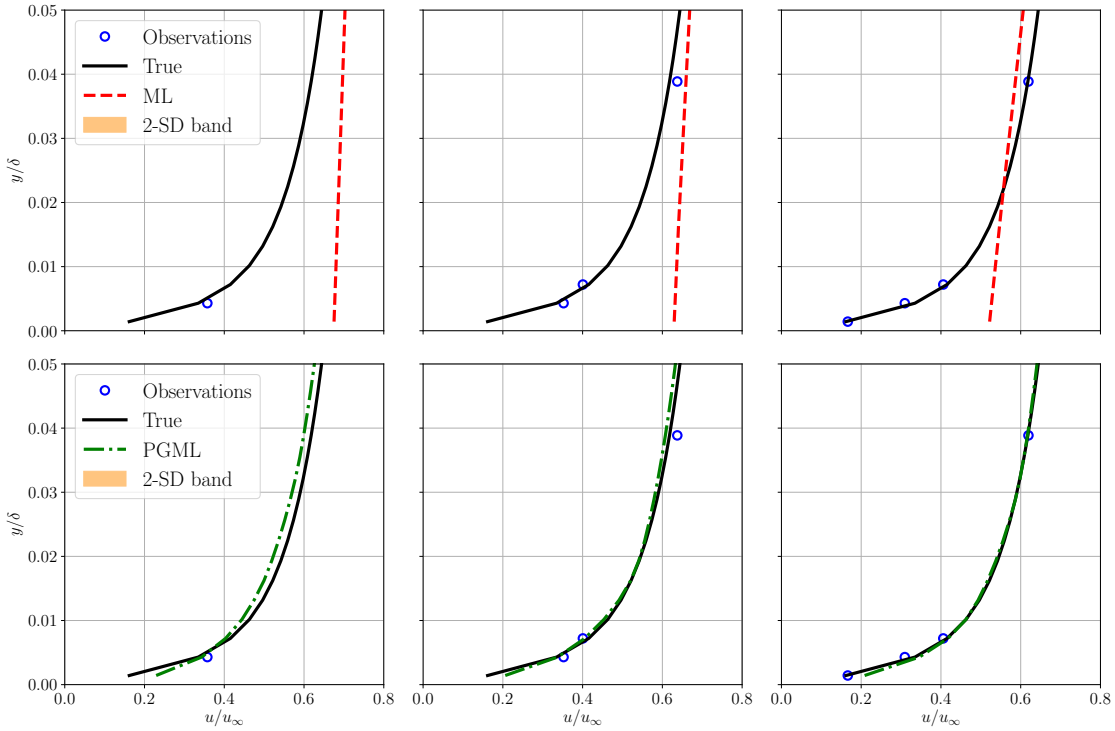

Figure 5.5: Boundary layer prediction in the *near-wall region* for turbulent flat plate flow at $x/L = 0.5$ along with the observations used for training the ML and PGML model. The amount of observations data used for training the model is 5% (left), 10% (middle), and 30% (right). The shaded area corresponds to two standard deviation (2-SD) band.

streamwise direction as follows

$$\text{RMSE}(x) = 100 \times \left(\frac{1}{N_y} \sum_{j=1}^{N_y} \left(\frac{u_T(y_j) - u_P(y_j)}{u_T(y_j)} \right)^2 \right)^{1/2} \tag{5.15}$$

where u_T is the velocity of the high-fidelity model, u_P is the velocity predicted from the data-driven model, N_y is the spatial resolution in the wall-normal direction. From Figure 5.9, we can see that the RMSE increases substantially in the extrapolation region for the ML model, especially when the observations are very sparse, i.e., 5% of the data. This is a well-known limitation of DL models to extrapolate poorly in the absence of dense data. The PGML model on the other hand has RMSE almost one order of magnitude less than the ML model in the interpolation region. Additionally, the increase in RMSE is not significant in the extrapolation region. This shows that the PGML model is robust for the unseen condition, and it performs well for out-of-distribution examples.

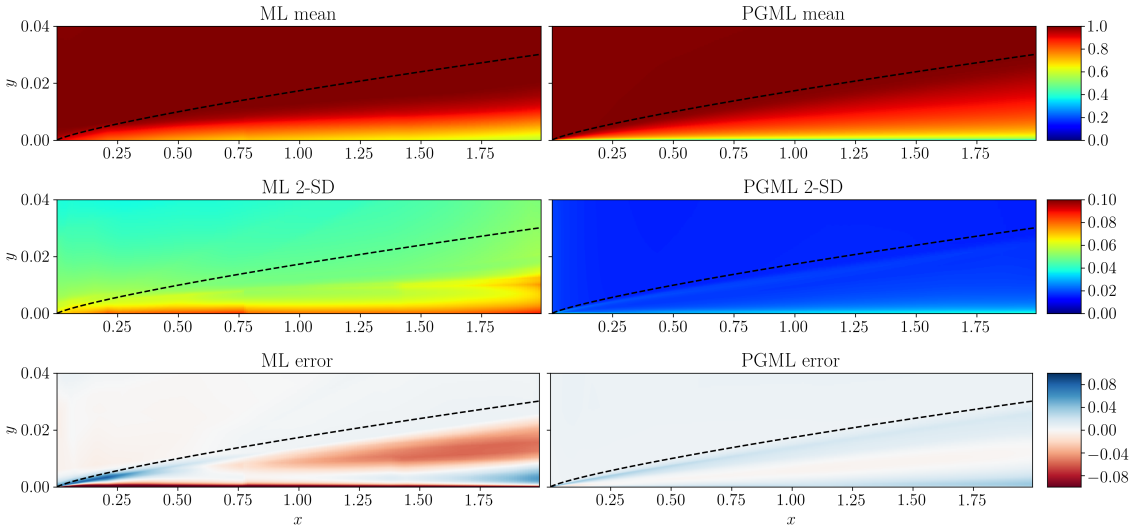

Figure 5.6: Prediction of the turbulent flat plate boundary layer with 5% of data used for training the ML and PGML model. The error is calculated as the difference between the true flow field and the flow field predicted by ML and PGML models. The figure is created by Matplotlib v3.5.1 [151].

5.4 Concluding Remarks

This study aims to develop a physics-guided machine learning (PGML) framework to improve data-driven models using prior knowledge from low-fidelity models. The PGML is a new deep neural network architecture that makes it possible to inject known physics during the training and *deployment* processes to reduce uncertainty and consequently improve the trade-off between efficiency and accuracy. Our design of a hierarchically sequential learning algorithm allows us to embed simplified theories, low order models, or empirical laws directly into deep learning models. These physics-based injections assist the neural network models in constraining the output to a manifold of the physically consistent solution and leads to improved reliability and generalizability. The PGML model trained using the deep ensembles algorithm provides us an estimate of the uncertainty associated with the prediction. This uncertainty information can be used for several applications like active learning, sensor placement, and optimization. Some of the questions addressed in this study are as follows

- How prior information on the physics of the problem can be used to improve black-box machine learning models?
- Can a concatenated neural network architecture augmented with a simplified or empirical model outperform a pure data-driven reconstruction model?

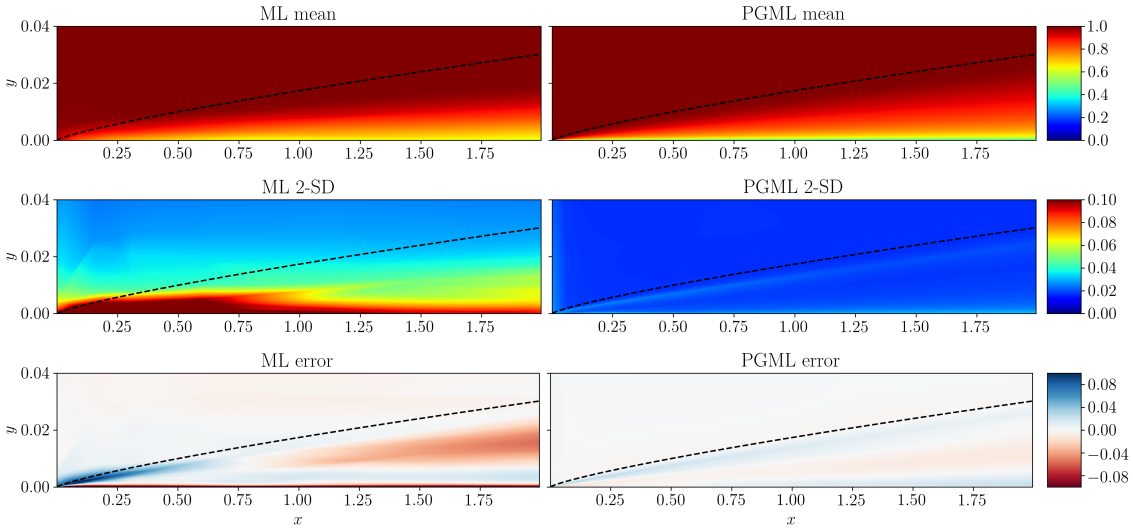

Figure 5.7: Prediction of the turbulent flat plate boundary layer with 10% of data used for training the ML and PGML model. The error is calculated as the difference between the true flow field and the flow field predicted by ML and PGML models. The figure is created by Matplotlib v3.5.1 [151].

- How can these data-driven predictive models be quantified regarding their uncertainties?
- What is the generalizability of predictive performance across ML and PGML architectures, when these are applied to unseen conditions?

To provide a proof-of-concept of the PGML framework, we use the laminar and turbulent flow over a flat plate as the prototypical test cases where the self-similar solution is used as the low-fidelity model, and Reynolds-Averaged Navier-Stokes equations (RANS) is a high-fidelity model. Although our notion of high-fidelity here is relative to the selected low-fidelity model, the chief idea, which is scalable with different notions, is to use low-fidelity models to restrict the neural network to a manifold, such that less data is required to train the network and to improve its predictive capability under extrapolation. Our analysis indicates that an injection of the empirical relations like *one-seventh power law* improves the predictive model significantly for estimating canonical turbulent flat plate boundary layer flows. We found that the PGML outperforms its ML counterpart by reducing the RMSE to nearly an order of magnitude lower levels. We also demonstrated that the proposed PGML framework substantially reduces the model uncertainty even when only sparse observations are available. Furthermore, generalizability of results is also supported when we integrate our predictive models for unseen conditions. Specifically, the RMSE

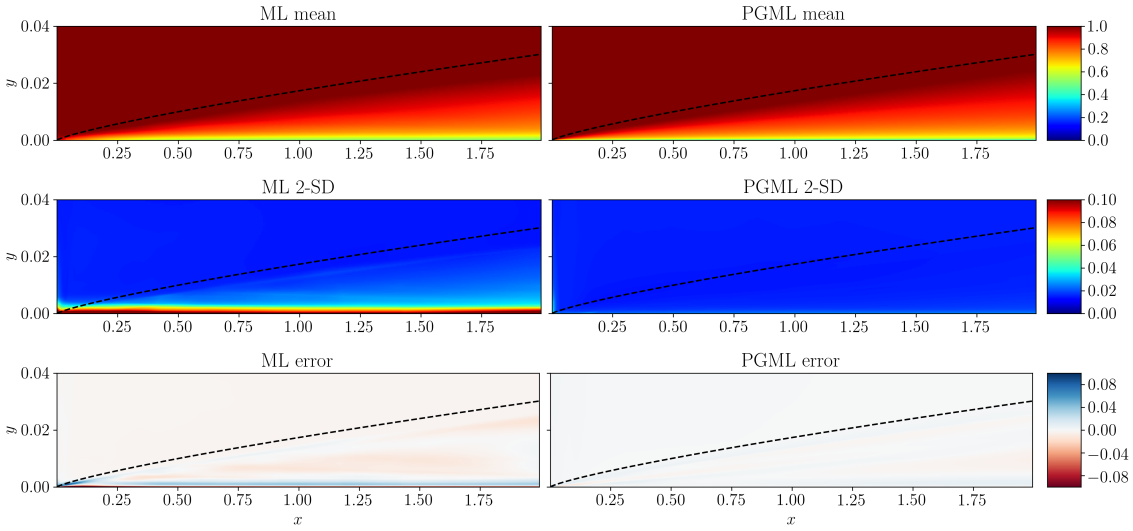

Figure 5.8: Prediction of the turbulent flat plate boundary layer with 30% of data used for training the ML and PGML model. The error is calculated as the difference between the true flow field and the flow field predicted by ML and PGML models. The figure is created by Matplotlib v3.5.1 [151].

distributions for the ML model (with respect to true data) are around 20% and 30% for interpolation and extrapolation regions, respectively. However, the PGML model has superior performance in both interpolation and extrapolation regions with the RMSE distribution in the range of 3%. For laminar flows, we also showed that a PGML approach (via injecting the Blasius solution) outperforms the ML approach. The Blasius approximation can also be extended for heat transfer problems (based on Falkner-Skan solutions) and for strongly nonlinear problems using non-similarity solutions [210].

The concatenated neural networks are capable of discovering a correlation between low- and high-fidelity data allowing for smaller training dataset sizes, training time, and improved extrapolation performance. Therefore, the PGML model has a great potential for a vast number of physical systems where a hierarchy of models is commonly used. The proposed framework is very modular and can be applied to a wide range of problems in fluid mechanics. Additionally, the framework is compatible with different neural network architectures making it suitable for complex high-dimensional problems. For example, one can treat the flow over a two-dimensional cylinder as the low-fidelity model for reconstructing the flow around a three-dimensional cylinder. Another interesting example is the wake prediction behind wind turbines, where analytical wake models can serve as the low-fidelity model for high-fidelity models like

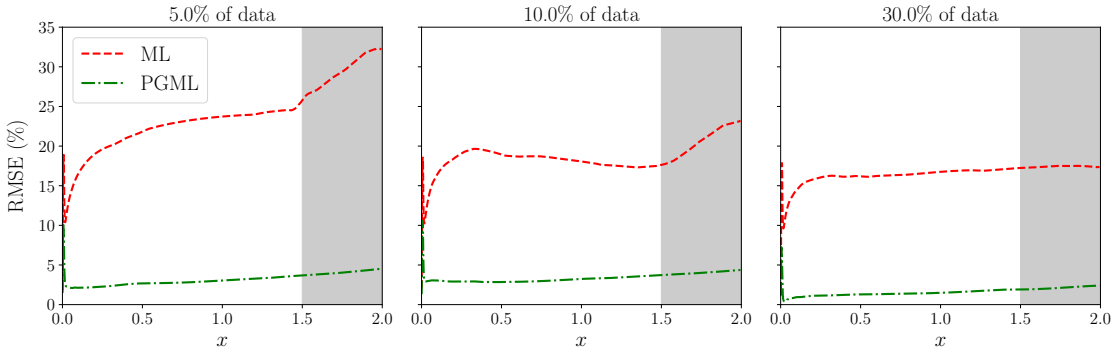

Figure 5.9: Variation of the normalized RMSE (in percentage) along the streamwise direction for ML and PGML models. The amount of observations data used for training the model is 5% (left), 10% (middle), and 30% (right). The gray color shaded area represents the extrapolation region.

RANS solver. One of the challenges with the PGML framework is coupling of steady and unsteady solver, i.e., RANS as the low-fidelity model and large eddy simulation (LES) as the high-fidelity model. If one is interested only in the time-averaged quantities from LES simulation, injecting information from the RANS solution within the neural network will help in pruning the space of possible solutions. We plan to extend the PGML framework for these high-dimensional systems with more complex low- and high-fidelity data fusion in our future studies. We also highlight that the PGML approach could be useful for generating physics consistent initial conditions to accelerate large-scale high-fidelity computations with eliminating non-physical initial transient time.

CHAPTER VI

Concatenated Neural Networks for Projection-based Reduced Order Modeling

The contents of this chapter has been published in Physical of Fluids (POF)[1].

Abstract: The unprecedented amount of data generated from experiments, field observations, and large-scale numerical simulations at a wide range of spatio-temporal scales have enabled the rapid advancement of data-driven and especially deep learning models in the field of fluid mechanics. Although these methods are proven successful for many applications, there is a grand challenge of improving their *generalizability*. This is particularly essential when data-driven models are employed within outer-loop applications like optimization. In this work, we put forth a physics-guided machine learning (PGML) framework that leverages the interpretable physics-based model with a deep learning model. Leveraging a concatenated neural network design from multi-modal data sources, the PGML framework is capable of enhancing the generalizability of data-driven models and effectively protect against or inform about the inaccurate predictions resulting from extrapolation. We apply the PGML framework as a novel model fusion approach combining the physics-based Galerkin projection model and long-short term memory (LSTM) network for parametric model order reduction of fluid flows. We demonstrate the improved generalizability of the PGML framework against a purely data-driven approach through the injection of physics features into intermediate LSTM layers. Our quantitative analysis shows that the overall model uncertainty can be reduced through the PGML approach especially for test data coming from a distribution different than the training data. Moreover, we demonstrate that our approach can be used as an inverse diagnostic tool providing a confidence score associated with models and observations. The proposed framework also allows for multi-fidelity computing by making use of low-fidelity models in the online deployment of quantified data-driven models.

[1]Pawar, S., San, O., Nair, A., Rasheed, A., & Kvamsdal, T. (2021). Model fusion with physics-guided machine learning: Projection-based reduced-order modeling. Physics of Fluids, 33(6), 067123.

6.1 Introduction

Data-driven approaches are emerging as the new paradigm in computational modeling of various problems in different branches of science and engineering, including fluid dynamics [51, 46]. The universal approximation capability of neural networks [145] makes them the powerful algorithm for complicated problems like turbulence closure modeling [87], spatio-temporal super-resolution [109], state-estimation [259], and nonlinear model order reduction [202]. One of the key issues with deep learning is that they exhibit poor *generalizability*, i.e., they produce inaccurate prediction when the test data is from a distribution far from that of the training data. This adversely affects the trustworthiness of neural networks for scientific applications and embedding the known physics to enhance the generalizability is a challenge and opportunity for data-driven algorithm developments [166, 318, 313, 167].

While many techniques have sought to enforce physics into data-driven models, such as regularizing the neural network with the governing equations or statistical constraints [309, 405, 116], imposing conservation of physical quantities directly into neural network [35, 249], embedding invariance property into neural network architecture [213, 421], and incorporating inductive biases to respect exact conservation laws [126], they offer many possibilities to fuse domain knowledge to improve the generalizability and explainability of these models [167]. In this work, we propose the physics-guided machine learning (PGML) framework that can answer the question of how to hybridize data-driven (i.e., non-intrusive) and first-principles (i.e., intrusive) approaches that are often utilized in fluid dynamics applications. The PGML framework is based on a composite modeling strategy where the physics-based model is integrated within the neural network architecture leading to a more generalizable learning engine.

The high-dimensional and multiscale nature of fluid flows makes the numerical discretization methods computationally infeasible for many practical applications. There are several techniques that aim at constructing the lower-order representation of high-dimensional systems that can capture the essential features and are computationally orders of magnitude faster than full order model [326, 31, 369]. These reduced order models (ROMs) are particularly appealing for outer-loop applications, such as data assimilation [56, 357, 242], uncertainty quantification [131], model predictive control [113, 149] that require multiple evaluations of a model for multiple inputs. Proper orthogonal decomposition (POD) [144] and dynamic mode decomposition [189] enabled ROMs are some of the most widely used methods among the fluid dynamics

community. POD is often combined with Galerkin projection (GP) to model the dynamics of the ROM [319, 325]. One of the limitations of intrusive approaches like GP is that it requires the complete knowledge about the equations governing the system's dynamics. However, for many physical systems, the exact governing equations are not available due to imperfect knowledge about the system. For example, in many geophysical flows we might not have accurate models for processes such as wind forcing, bottom friction, stratification, and other parameterization [184].

Lately, equation-free or non-intrusive reduced order modeling (NIROM) has drawn great attention in the fluid mechanics community due to its flexibility and efficiency for systems with incomplete dynamics and is undergoing rapid development with various algorithms emerging from different perspectives [419]. In most of the NIROMs, the main idea is to employ deep learning methods to construct nonlinear manifold [122, 202, 231] and to model the temporal dynamics of ROMs [415, 306, 392, 130]. Despite the success of NIROMs for many complex nonlinear problems, the naive deployment of NIROMs in multi-query applications is limited because they lack connection with the physics-based model and might suffer from poor performance in extrapolation regime [223]. Therefore, the hybrid modeling approach that can overcome these drawbacks is warranted and to this end, we apply the PGML framework that exploits the Galerkin projection model for the known physics and long-short term memory (LSTM) neural network to model the unknown physics. We remark here that the ideas from other approaches like physics-reinforced neural network (PRNN) [64] and physics-embedded convolutional autoencoder (PhyCAE) [249] can be easily integrated within the proposed PGML framework. Furthermore, improving model predictions with composite models involving the composition (e.g., addition) of two models has been shown to be effective, for example, in the field of computer experiments using the nonstationary Gaussian processes [20, 79, 216, 75]. Such composite models use one model to learn the global trends in the data and another to learn the local volatility, thereby together enhancing predictions. In this case, the second model is trained to predict the residual of the first model. We also refer readers to an interesting discussion on improving predictions of such computer experiments using a small-scale measurement error term called nugget [125], where the authors highlighted that the nugget term can lead to enhanced surrogate models with better statistical properties.

In this study, we consider a dynamical system whose evolution is governed by

partial differential equations (PDEs) such as the Navier-Stokes equations as follows

$$\dot{\boldsymbol{u}} = \boldsymbol{f}(\boldsymbol{u}; \boldsymbol{x}, t; \boldsymbol{\mu}), \tag{6.1}$$

where $\boldsymbol{u}$ is the prognostic variable that depends on a set of parameters, $\boldsymbol{f}$ is the nonlinear function that fully represents the physical processes and conservation laws, and $\boldsymbol{\mu} \in \mathbb{R}^{N_p}$ is the parameterization such as Reynolds or Rayleigh numbers, and N_p refers to the number of parameters. Then, we can split $\boldsymbol{f}$ using a notion of known-physics (i.e., $\hat{\boldsymbol{f}}$) and unknown-physics (i.e, $\boldsymbol{\pi}$) parts. More precisely, $\hat{\boldsymbol{f}}$ refers to the model's dynamical core (i.e., a semi-discretized function of PDEs of mass, momentum, energy conservation laws etc). However, there could be unknown physics that might not be covered within the physics-based dynamical core model $\hat{\boldsymbol{f}}$. For example, $\boldsymbol{\pi}$ might refer to the physical and chemical processes that are either unknown or too complex to be modeled (e.g., precipitation, long and short wave atmospheric radiation, clouds, constituency transport, and chemical reactions, or any forcing function that is not modeled in $\hat{\boldsymbol{f}}$). The augmented representation can then be written at an abstract level as

$$\dot{\boldsymbol{u}} = \hat{\boldsymbol{f}}(\boldsymbol{u}; \boldsymbol{x}, t; \boldsymbol{\mu}) + \boldsymbol{\pi}(\boldsymbol{u}; \boldsymbol{x}, t; \boldsymbol{\mu}), \tag{6.2}$$

where $\boldsymbol{\pi}$ encompasses all the unknown processes and parameterizations. To differentiate the solutions between the full and partial physics, let's assume that we have a physics-based (but incomplete) model

$$\dot{\boldsymbol{u}} = \hat{\boldsymbol{f}}(\boldsymbol{u}; \boldsymbol{x}, t; \boldsymbol{\mu}). \tag{6.3}$$

Here, we can define $\delta\boldsymbol{u}$ as a natural way to quantify the unknown physics $\boldsymbol{\pi}$ as follows

$$\delta\boldsymbol{u} = \sqrt{\frac{||\boldsymbol{u}_1 - \boldsymbol{u}_2||^2}{||\boldsymbol{u}_2||^2}}, \tag{6.4}$$

where $\boldsymbol{u}_1$ is the solution of Eq. 6.3 (i.e., without $\boldsymbol{\pi}$), $\boldsymbol{u}_2$ is the solution of Eq. 6.2 with $\boldsymbol{\pi}$, and $||\cdot||$ is the L_2 norm. One can easily quantify the level of unknown processes using the definition given by Eq. 6.4, i.e. $\delta\boldsymbol{u} \to 0$ when $\boldsymbol{\pi} \to 0$. On the other hand, a higher value of $\delta\boldsymbol{u}$ means that our known physical model $\hat{\boldsymbol{f}}$ is poor.

The chief motivation behind the PGML framework is to exploit domain knowledge to produce generalizable data-driven models. We demonstrate the application of the

PGML framework to fuse intrusive and non-intrusive parametric projection-based ROMs for incomplete dynamical systems that can be represented using Eq. 6.3. Apart from improving the extrapolation performance of data-driven models, the PGML framework can also be considered as an enabler for multi-fidelity computing through a synergistic combination of low-cost/low-fidelity models with the high-fidelity data. The field of multi-fidelity computing is gaining attention in computational science and engineering, as it provides a way to utilize computationally cheap approximate models with high-fidelity observations [290, 292]. This is particularly relevant to fluid mechanics applications where there is a wealth of simplified models like potential flow theory, Blasius boundary layer model, and continuum Darcy/pipe flow model. One of the major challenges in multi-fidelity computing is determining the correlation between low and high fidelity models, and the neural networks employed within the PGML framework are capable of discovering this nonlinear relation. This first step in the assessment of the new PGML model shows that the proposed hybrid reduced order modeling framework could represent a viable tool for emerging digital twin applications [313], where low fidelity models are often employed for computational efficiency. Nonetheless, with available streams of observational data from the real physical asset, PGML can enhance the fidelity of the whole system.

This chapter is structured as follows. We introduce the parametric ROM framework in Section 6.2. We then discuss the a specific formulation of the PGML framework in Section 6.3 for combining the domain knowledge with data. The numerical results for the two-dimensional vorticity transport equation are detailed in Section 6.4. Finally, Section 6.5 will present conclusions and ideas for the future work.

6.2 Parametric ROM Framework

The first step in parametric ROMs is the dimensionality reduction. We choose proper orthogonal decomposition (POD) to compute the linear orthogonal basis functions. POD is one of the most popular technique for dimensionality reduction in the fluid mechanics community [349, 32]. POD provides the linear basis functions that minimizes the error between the true data and its projection in the L_2 sense compared to all linear basis functions of the same dimension. Given the N_s snapshots of the data for a variable of interest in the physical system, we can form the matrix $\boldsymbol{A}$ as follows

$$\boldsymbol{A} = \left[\boldsymbol{u}^{(1)}, \boldsymbol{u}^{(2)}, \ldots, \boldsymbol{u}^{(N_s)}\right] \in \mathbb{R}^{N \times N_s} \tag{6.5}$$

where $\boldsymbol{u}^{(k)} \in \mathbb{R}^N$ corresponds to individual snapshot of the solution in time. Then, we perform the singular value decomposition (SVD) as follows

$$\boldsymbol{A} = \boldsymbol{W}\boldsymbol{\Sigma}\boldsymbol{V}^T = \sum_{k=1}^{N_s} \sigma_k \boldsymbol{w}_k \boldsymbol{v}_k^T, \tag{6.6}$$

where $\boldsymbol{W} \in \mathbb{R}^{N \times N_s}$ is a matrix with orthonormal columns which represent the left-singular vectors, $\boldsymbol{V} \in \mathbb{R}^{N_s \times N_s}$ is matrix with orthonormal columns representing the right-singular vectors, and $\boldsymbol{\Sigma} \in \mathbb{R}^{N_s \times N_s}$ is a matrix with non-negative diagonal entries, called singular values, arranged such that $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_{N_s} \geq 0$. In dimensionality reduction, only the first N_r columns of $\boldsymbol{W}$ and $\boldsymbol{V}$ are retained along with the upper-left $N_r \times N_r$ sub-matrix of $\boldsymbol{\Sigma}$. The singular values σ_i give a measure of the quality of information that is retailed in N_r-order approximation of the matrix $\boldsymbol{A}$. The N_r columns of $\boldsymbol{W}$ are referred to as the POD modes or basis functions, denoted as $\boldsymbol{\Phi}_{\boldsymbol{\mu}} = \{\phi_k\}_{k=1}^{N_r}$. The amount of the energy retained by POD modes can be computed using the quantity called the relative information content (RIC) as follows

$$\text{RIC}(N_r) = \frac{\sum_{j=1}^{N_r} \sigma_j^2}{\sum_{j=1}^{N_s} \sigma_j^2} \tag{6.7}$$

We utilize the Grassmann manifold interpolation [6, 428] to compute the basis functions for a new set of parameters from available POD basis function computed in the offline phase. In the offline phase, we calculate a set of POD basis functions $\{\boldsymbol{\Phi}_i\}_{i=1}^{N_c}$ corresponding to a set of parameters $\{\boldsymbol{\mu}_i\}_{i=1}^{N_c}$, where N_c is the number of parameterized data sets (e.g., $N_c = 4$ in this study since we utilize data snapshots from Re = $\{200, 400, 600, 800\}$ in our training). The Grassman manifold interpolation consists of choosing a reference point S_0 corresponding to the basis set $\boldsymbol{\Phi}_0$, and then mapping each point S_i to a matrix $\boldsymbol{\Gamma}_i$ which represents a point on the tangent space at S_0 using logarithmic map Log_{S_0} as follows

$$(\boldsymbol{\Phi}_i - \boldsymbol{\Phi}_0\boldsymbol{\Phi}_0^T\boldsymbol{\Phi}_i)(\boldsymbol{\Phi}_0^T\boldsymbol{\Phi}_i)^{-1} = \boldsymbol{W}_i\boldsymbol{\Sigma}_i\boldsymbol{V}_i^T, \tag{6.8}$$

$$\boldsymbol{\Gamma}_i = \boldsymbol{W}_i \tan^{-1}(\boldsymbol{\Sigma}_i)\boldsymbol{V}_i^T. \tag{6.9}$$

The matrix $\boldsymbol{\Gamma}_t$ corresponding to target operating point $\boldsymbol{\mu}_t$ is computed by interpolating the corresponding entries of matrices $\{\boldsymbol{\Gamma}_i\}_{i=1}^{N_c}$. When $N_p = 1$ (e.g., $\boldsymbol{\mu} \in \{\text{Re}\}$ in this

work), typically Lagrange-type interpolation method is applied as follows

$$\boldsymbol{\Gamma}_t = \sum_{i=1}^{N_c} \left(\prod_{\substack{j=1 \\ j \neq i}}^{N_c} \frac{\mu_t - \mu_j}{\mu_i - \mu_j} \right) \boldsymbol{\Gamma}_i. \tag{6.10}$$

The POD basis functions $\boldsymbol{\Phi}_t$ corresponding to the test parameter $\boldsymbol{\mu}_t$ is computed using the exponential mapping as follows

$$\boldsymbol{\Gamma}_t = \boldsymbol{W}_t \boldsymbol{\Sigma}_t \boldsymbol{V}_t^T, \tag{6.11}$$

$$\boldsymbol{\Phi}_t = [\boldsymbol{\Phi}_0 \boldsymbol{V}_t \cos(\boldsymbol{\Sigma}_t) + \boldsymbol{W}_t \sin(\boldsymbol{\Sigma}_t)] \boldsymbol{V}_t^T, \tag{6.12}$$

where the trigonometric operators apply only to diagonal elements.

Once the basis functions corresponding to the test operating point are computed, the reduced order representation of the high-dimensional state $\boldsymbol{u}(\boldsymbol{x}, t; \boldsymbol{\mu})$ is

$$\boldsymbol{u}(\boldsymbol{x}, t; \boldsymbol{\mu}) \approx \boldsymbol{u}_r(\boldsymbol{x}, t; \boldsymbol{\mu}) = \boldsymbol{\Phi}_{\boldsymbol{\mu}} \boldsymbol{a}(t; \boldsymbol{\mu}), \tag{6.13}$$

where $\boldsymbol{a}(t; \boldsymbol{\mu}) \in \mathbb{R}^{N_r}$ is the reduced state on the basis functions and is also called as the modal coefficients and $\boldsymbol{\Phi}_{\boldsymbol{\mu}}$ is the POD basis functions for a set of parameters $\boldsymbol{\mu}$. The ROM is obtained by substituting the low-rank approximation given in Eq. 6.13 into the full order model defined in Eq. 6.2 and then taking the inner product with test basis functions to yield a system of N_r ordinary differential equations (ODEs). For many fluid mechanics problems, the mechanistic description of some variables or processes is not available or insufficient for the desired task (i.e., the term $\boldsymbol{\pi}(\cdot)$ is unknown). Therefore, the physics-based intrusive approaches like Galerkin projection ROM (GROM) provide us

$$\dot{\boldsymbol{a}} = \boldsymbol{\Phi}_{\boldsymbol{\mu}}^T \boldsymbol{f}(\boldsymbol{\Phi}_{\boldsymbol{\mu}} \boldsymbol{a}; \boldsymbol{x}, t; \boldsymbol{\mu}). \tag{6.14}$$

We note here that in GROM, the test basis functions are the same as the trial basis which allows us to make use of the orthonormality, i.e., $\boldsymbol{\Phi}_{\boldsymbol{\mu}}^T \boldsymbol{\Phi}_{\boldsymbol{\mu}} = \boldsymbol{I}_{N_r}$.

One of the limitation with intrusive ROMs like GROM is that the governing equations of the system have to be known exactly. If the governing equations are not known exactly, the effect of unknown dynamics is truncated in GROMs and accurate prediction is not achieved. This issue is mitigated in NIROMs that exploit machine

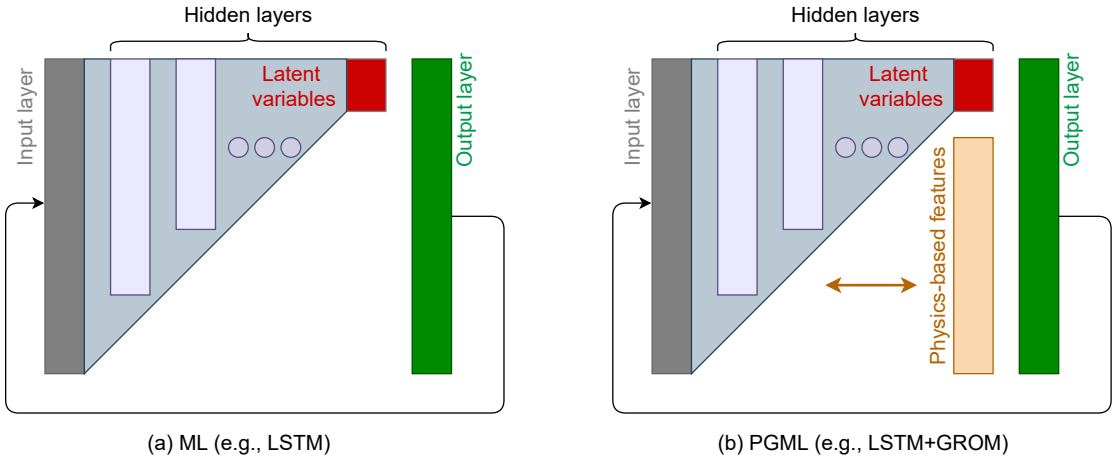

Figure 6.1: General schematic of the PGML framework. The PGML is a modular deep neural network architecture that makes it possible to inject known physics during the training and deployment processes to reduce uncertainty, and consequently improve the trade-off between efficiency and accuracy. Of particular interest, in this chapter we consider (a) an end-to-end non-intrusive model reduction approach using a recurrent neural network architecture based on the LSTM, and (b) its PGML version with the GROM features embedded into the LSTM layers. As one of the main features of the PGML framework, we can concatenate additional features into the middle layers of neural networks without breaking the neural network training process.

learning (ML) algorithms to learn the reduced order dynamics from the observed data [419]. The training data for the ML algorithm can be generated by using the same data used for computing the POD basis functions as follows

$$\boldsymbol{\alpha}(t;\boldsymbol{\mu}) = \langle \boldsymbol{u}(\boldsymbol{x},t;\boldsymbol{\mu}), \boldsymbol{\Phi}_{\boldsymbol{\mu}} \rangle, \tag{6.15}$$

where $\boldsymbol{\alpha}$ is the data projection coefficient, and the angle-parentheses refer to the Euclidean inner product defined as $\langle \boldsymbol{p}, \boldsymbol{q} \rangle = \boldsymbol{p}^T \boldsymbol{q}$. The neural network is one of the most successful algorithms for learning the dynamics on reduced order basis functions. However, as the complexity of the problems increases, the number of trainable parameters quickly explodes and neural networks suffer from high epistemic uncertainty associated with limited training data. We address this problem by hybridizing intrusive and non-intrusive approaches through our PGML framework. More specifically, in Section 6.3 we present a PGML approach to combine a data-driven neural network model and a projection-based ROM for improved model generalizability.

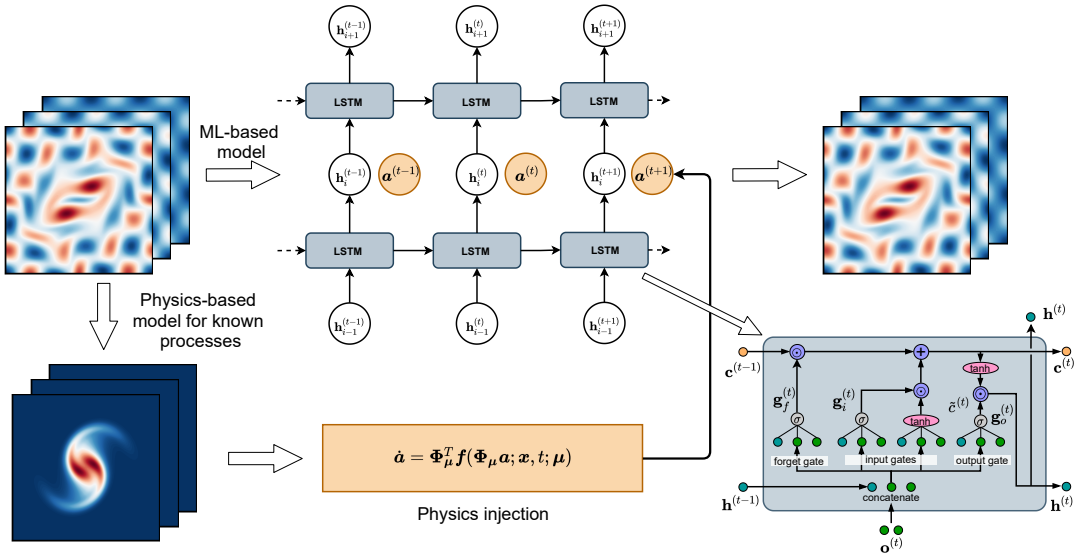

Figure 6.2: Proposed physics-guided machine learning (PGML) framework for reduced order modeling, where the physics-based features are directly embedded into hidden layers of the neural network along with learned latent variables. Here, the physics-based features obtained by GROM assist the LSTM based neural network models in constraining the output to a manifold of the physically more realizable solution, and lead to improved generalizability for the extrapolation regime beyond the training data sets.

6.3 Physics-guided Machine Learning (PGML) Framework

In this Section, we detail different components of the proposed PGML framework for modeling the reduced order representation of the high-dimensional nonlinear dynamical system. In a broader sense, the PGML is a modular approach to leverage the interpretable physics based model with a deep learning model. Thus, the PGML makes it possible to inject known physics during the training and deployment processes to reduce uncertainty, and consequently improve the trade-off between efficiency and accuracy. Since it relies on injecting additional features into the internal layers of neural networks as shown in Figure 6.1, it can be also viewed as a highly generic approach to embed simplified theories, physics-based kernels or features directly into deep neural network models. For example, in our recent study on aerodynamics load calculations [287], we show that embedding a panel method significantly improves the deep learning based surrogate models (e.g., achieved approximately 75% reduction in uncertainty). In this chapter, our aim is to demonstrate how the PGML combines physical information with POD based GROM, and explore numerically how these physics-based features assist the neural network models in constraining the output to a

manifold of the physically more realizable solution and lead to improved generalizability for the extrapolation regime beyond the training data sets. Moreover, we study the sensitivity of the layer-wise location of the physics injection. Here, we highlight that there are more advanced hyperparameter tuning methods on designing automated neural network architectures (e.g., using genetic algorithms [377]), and these automated machine learning approaches can be utilized to find the optimal layer(s) to inject the physics in the PGML architectures, a topic we would like to explore in our future studies.

Different components of the PGML framework for ROM are illustrated in Figure 6.2. First, we present the formulation of the LSTM neural network for time series modeling and then discuss its application within the PGML framework. In a typical ML algorithm for time series modeling, the model is trained on a time series of observable $\{\mathbf{o}^{(1)}, \ldots, \mathbf{o}^{(T)}\}$, where $\mathbf{o} \in \mathbb{R}^{d_o}$, sampled at a fixed time interval. The LSTM neural network is one of the most popular ML algorithms for time-series modeling due to its ability to model long-term temporal dependencies without suffering from the vanishing gradient problem of recurrent neural network [142]. Recently, LSTM has also been very successful in modeling high-dimensional spatio-temporal chaotic systems and for ROMs [281, 382, 230, 383, 133]. The general functional form of the models used for time series forecasting can be written as

$$\mathbf{h}^{(t)} = f_h^h(\mathbf{o}^{(t)}, \mathbf{h}^{(t-1)}), \tag{6.16}$$

$$\tilde{\mathbf{o}}^{(t+1)} = f_h^o(\mathbf{h}^{(t)}), \tag{6.17}$$

where $\mathbf{o}^{(t)}$ is the current observable and $\tilde{\mathbf{o}}^{(t+1)}$ is the forecast for the observable at the next time step $\mathbf{o}^{(t+1)}$ and $\mathbf{h}^{(t)} \in \mathbb{R}^{d_h}$ is the hidden state. Here. f_h^h is the hidden-to-hidden mapping and f_h^o is the hidden-to-output mapping.

The LSTM mitigates the issue with vanishing (or exploding) gradient by employing the gating mechanism that allows information to be forgotten. The equations that implicitly define the hidden-to-hidden mapping from hidden state from the previous time step (i.e., $\mathbf{h}^{(t-1)}$) and input vector at the current time step (i.e., $\mathbf{o}^{(t)}$) to the

forecast hidden state (i.e., $\mathbf{h}^{(t)}$) can be written as

$$\mathbf{g}_f^{(t)} = \sigma_f(\mathbf{W}_f[\mathbf{h}^{(t-1)}, \mathbf{o}^{(t)}] + \mathbf{b}_f), \tag{6.18}$$

$$\mathbf{g}_i^{(t)} = \sigma_i(\mathbf{W}_i[\mathbf{h}^{(t-1)}, \mathbf{o}^{(t)}] + \mathbf{b}_i), \tag{6.19}$$

$$\tilde{\mathbf{c}}^{(t)} = \tanh(\mathbf{W}_c[\mathbf{h}^{(t-1)}, \mathbf{o}^{(t)}] + \mathbf{b}_c), \tag{6.20}$$

$$\mathbf{c}^{(t)} = \mathbf{g}_f^{(t)} \odot \mathbf{c}^{(t-1)} + \mathbf{g}_i^{(t)} \odot \tilde{\mathbf{c}}^{(t)}, \tag{6.21}$$

$$\mathbf{g}_o^{(t)} = \sigma_o(\mathbf{W}_o[\mathbf{h}^{(t-1)}, \mathbf{o}^{(t)}] + \mathbf{b}_o), \tag{6.22}$$

$$\mathbf{h}^{(t)} = \mathbf{g}_o^{(t)} \odot \tanh(\mathbf{c}^{(t)}), \tag{6.23}$$

where $\mathbf{g}_f^{(t)},\ \mathbf{g}_i^{(t)},\ \mathbf{g}_o^{(t)} \in \mathbb{R}^{d_h}$ are the forget gate, input gate, and output gate, respectively. The $\mathbf{o}^{(t)} \in \mathbb{R}^{d_o}$ is the input vector at time t, $\mathbf{h}^{(t)} \in \mathbb{R}^{d_h}$ is the hidden state, $\mathbf{c}^{(t)} \in \mathbb{R}^{d_h}$ is the cell state, $\mathbf{W}_f,\ \mathbf{W}_i,\ \mathbf{W}_c,\ \mathbf{W}_o \in \mathbb{R}^{d_h \times (d_h + d_o)}$ are the weight matrices, and $\mathbf{b}_f,\ \mathbf{b}_i,\ \mathbf{b}_c,\ \mathbf{b}_o \in \mathbb{R}^{d_h}$ are the bias vectors. The symbol $\odot$ denotes the element-wise multiplication, and σ is the sigmoid activation function. The activation functions $\sigma_f, \sigma_i, \sigma_o$ are sigmoids. The hidden-to-output mapping is given by a fully connected layer with a linear activation function as follows

$$\tilde{\mathbf{o}}^{(t+1)} = \mathbf{W}_o \mathbf{h}^t, \tag{6.24}$$

where $\mathbf{W}_o \in \mathbb{R}^{d_o \times d_h}$.

When we apply the LSTM neural network for building NIROMs, the reduced order state of the system at a future time step, i.e., $\boldsymbol{\alpha}^{(t+1)}$, is learned as the function of a short history of d past temporally consecutive reduced order states as follows

$$\boldsymbol{\alpha}^{(t+1)} = \mathcal{F}(\underbrace{\boldsymbol{z}^{(t)}, \boldsymbol{z}^{(t-1)}, \ldots, \boldsymbol{z}^{(t-d+1)}}_{\boldsymbol{z}^{(t):(t-d+1)}}; \boldsymbol{\theta}), \tag{6.25}$$

where $\mathcal{F}(\cdot\ ; \boldsymbol{\theta})$ is the nonlinear function parameterized by a set of parameters $\boldsymbol{\theta}$, and $\boldsymbol{z}$ refers to input features consisting of the POD modal coefficients and a set of parameters governing the system, i.e., $\boldsymbol{z} \in \mathbb{R}^{N_r + N_p}$. The LSTM is trained using the backpropagation through time (BPTT) algorithm and the parameters are optimized as

$$\boldsymbol{\theta}^* = \arg\min_\theta \frac{1}{N - d + 1} \sum_{n=d}^{N} ||\mathcal{F}(\underbrace{\boldsymbol{z}^{(t):(t-d+1)}}_{\zeta}; \boldsymbol{\theta}) - \boldsymbol{\alpha}^{(t+1)}||_2^2. \tag{6.26}$$

We employ $(N_l - 1)$ LSTM layers and a single dense layer with a linear activation

function as the last layer. Therefore, the ML model can be written as

$$\mathcal{F}_{\text{ML}}(\boldsymbol{\zeta};\boldsymbol{\theta}) = \boldsymbol{h}_{N_l}(\cdot;\boldsymbol{\Theta}_{N_l}) \circ \cdots \circ \boldsymbol{h}_2(\cdot;\boldsymbol{\Theta}_2) \circ \boldsymbol{h}_1(\boldsymbol{\zeta};\boldsymbol{\Theta}_1), \tag{6.27}$$

where the output of each LSTM layer ($i = 1, \ldots, N_l - 1$) is $\boldsymbol{h}_i(\cdot;\boldsymbol{\Theta}_i) \in \mathbb{R}^{d \times N_h}$ and the last dense layer maps the final hidden state to the output, i.e., $\boldsymbol{h}_{N_l}(\cdot;\boldsymbol{\Theta}_{N_l}) : \mathbb{R}^{N_h} \to \mathbb{R}^{N_r}$. Here, N_h is the number of hidden units in the LSTM cell and we use the constant number of hidden units across all LSTM layers.

In the PGML framework, the features extracted from the physics-based model are embedded into the ith intermediate hidden layer along with the latent variables as follows

$$\mathcal{F}_{\text{PGML}}(\boldsymbol{\zeta};\boldsymbol{\theta}) = \boldsymbol{h}_{N_l}(\cdot;\boldsymbol{\Theta}_{N_l}) \circ \cdots \circ \underbrace{\mathcal{C}(\boldsymbol{h}_i(\cdot;\boldsymbol{\Theta}_i), \boldsymbol{a}^{(t):(t-d+1)})}_{\text{Physics injection}} \circ \cdots \circ \boldsymbol{h}_1(\boldsymbol{\zeta};\boldsymbol{\Theta}_1), \tag{6.28}$$

where $\mathcal{C}(\cdot,\cdot)$ represents the concatenation operation and $\mathbf{a}$ are the physics-based features (i.e., the GROM modal coefficients). Therefore, the output of ith layer will be in $\mathbb{R}^{d \times (N_r + N_h)}$. We highlight here that the physics-based features are embedded only at a single layer of the neural network. Therefore, at which layer shall the physics-based features be embedded becomes another hyperparameter of the PGML framework. While some guidelines or rule of thumb can be established based on the interplay between known and unknown physics, a thorough systematic investigation is needed. Methods like layer-wise relevance propagation can be adopted to bring interpretability to complex neural networks in the PGML framework and we consider it as part of our future work. The physics-based features assist the LSTM network in constraining the output to a manifold of the physically realizable solution and leads to improved generalizability for the extrapolation regime. Even though we demonstrate the PGML framework for POD-ROM, we emphasize here that the PGML framework is highly modular and can also be applied to nonlinear, cluster-based, and network-based ROMs. For example, we do not need to restrict to only linear basis construction methods like POD and the PGML framework can be easily extended to nonlinear manifold generation methods like convolutional autoencoders [122, 202, 91].

6.4 Vortex Merging Experiments

We apply the PGML framework to the two-dimensional vorticity transport equation for the vortex merger problem as a prototypical test case. This problem involves

the study of the merging of two co-rotating vortices. The two-dimensional vorticity transport equation can be written as

$$\frac{\partial \omega}{\partial t} + J(\omega, \psi) = \frac{1}{\mathrm{Re}} \nabla^2 \omega + \boldsymbol{\pi}(\boldsymbol{x}, t; \boldsymbol{\mu}), \tag{6.29}$$

$$\nabla^2 \psi = -\omega, \tag{6.30}$$

where ω is the vorticity defined as $\omega = \nabla \times \mathbf{u}$, $\mathbf{u} = [u, v]^T$ is the velocity vector, ψ is the streamfunction, Re is the Reynolds number parameterizing the system and $\boldsymbol{\pi}(\cdot)$ represent the unknown physics. Using the notion introduced in Eq. 6.3, we can rewrite Eq. 6.29 as

$$\frac{\partial \omega}{\partial t} = \hat{\boldsymbol{f}}(\boldsymbol{x}, t; \boldsymbol{\mu}) + \boldsymbol{\pi}(\boldsymbol{x}, t; \boldsymbol{\mu}), \tag{6.31}$$

where

$$\hat{\boldsymbol{f}}(\boldsymbol{x}, t; \boldsymbol{\mu}) = \frac{1}{\mathrm{Re}} \nabla^2 \omega - J(\omega, \psi). \tag{6.32}$$

The two terms on the right hand side of Eq. 6.32, i.e., the linear Laplacian term and the nonlinear Jacobian term represent the known physics of the system and are defined as follows

$$\nabla^2 \omega = \frac{\partial^2 \omega}{\partial x^2} + \frac{\partial^2 \omega}{\partial y^2}, \tag{6.33}$$

$$J(\omega, \psi) = \frac{\partial \psi}{\partial y} \frac{\partial \omega}{\partial x} - \frac{\partial \psi}{\partial x} \frac{\partial \omega}{\partial y}. \tag{6.34}$$

To demonstrate the proposed PGML framework for ROMs, we assume that $\boldsymbol{\pi}$ is unknown. Therefore, for the best case scenario, the known processes can be represented by

$$\frac{\partial \omega}{\partial t} + J(\omega, \psi) = \frac{1}{\mathrm{Re}} \nabla^2 \omega, \tag{6.35}$$

together with the kinematic relationship given by Eq. 6.30. As discussed in Section 6.2, the reduced order approximation of the vorticity and streamfunction fields can be written as

$$\omega(x, y, t) \approx \sum_{k=1}^{N_r} a_k(t) \phi_k^{\omega}(x, y), \tag{6.36}$$

$$\psi(x, y, t) \approx \sum_{k=1}^{N_r} a_k(t) \phi_k^{\psi}(x, y), \tag{6.37}$$

The vorticity and streamfunction share the same time dependent modal coefficients as they are both related by the kinematic relationship given in Eq. 6.30. Furthermore, the POD basis functions of the streamfunction can be obtained by solving the below Poisson equation

$$\nabla^2 \phi_k^{\psi} = -\phi_k^{\omega}. \tag{6.38}$$

The GROM for the vorticity transport equations considering only the known physics can be written as

$$\dot{\boldsymbol{a}} = \mathfrak{L}\boldsymbol{a} + \boldsymbol{a}^T \mathfrak{N} \boldsymbol{a}, \tag{6.39}$$

or more explicitly,

$$\frac{\mathrm{d}a_k}{\mathrm{d}t} = \sum_{i=1}^{N_r} \mathfrak{L}_k^i a_i + \sum_{i=1}^{N_r}\sum_{j=1}^{N_r} \mathfrak{N}_k^{ij} a_i a_j, \tag{6.40}$$

where $\mathfrak{L}$ and $\mathfrak{N}$ are the linear and nonlinear operators. We highlight that the model provided by Eq. 6.40 does not have access to the unknown processes $\boldsymbol{\pi}$. Since it has access to the transport operators given by Eq. 6.32, it is thus referred to a physics-based model (i.e., an intrusive reduced order model) in this work, even though its basis functions are generated from data. Instead of the POD basis function expansion provided by Eq. 6.36-6.37, one can generalize the model using a standard Fourier Galerkin approach [334], completely eliminating the dependency on a pre-recorded data set on building a physics based ROM. The linear and nonlinear operators for the vorticity transport equation are as follows

$$\mathfrak{L}_{ik} = \left\langle \frac{1}{\mathrm{Re}} \nabla^2 \phi_i^{\omega}, \phi_k^{\omega} \right\rangle, \tag{6.41}$$

$$\mathfrak{N}_{ijk} = \left\langle -J(\phi_i^{\omega}, \phi_j^{\psi}), \phi_k^{\omega} \right\rangle, \tag{6.42}$$

where the angle-parentheses refer to the Euclidean inner product defined as $\langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x}^T\mathbf{y} = \sum_{i=1}^{N} x_i y_i$. We apply the third-order Adams-Bashforth (AB3) numerical scheme to solve the system of ODEs given in Eq. 6.40. In discrete sense, the update formula can be written as

$$a_k^{(n+1)} = a_k^{(n)} + \Delta t \sum_{q=0}^{s} \beta_q G(a_k^{(n-q)}), \tag{6.43}$$

where s and β_q are the constants corresponding to AB3 scheme, which are $s = 2,\ \beta_0 = 23/12,\ \beta_1 = -16/12$, and $\beta_2 = 5/12$. Here, the operator $G(a_k)$ is the right hand side

of Eq. 6.40 as follows

$$G(a_k) = \sum_{i=1}^{N_r} \mathfrak{L}_k^i a_i + \sum_{i=1}^{N_r}\sum_{j=1}^{N_r} \mathfrak{N}_k^{ij} a_i a_j. \tag{6.44}$$

We start with an initial vorticity field of two Gaussian-distributed vortices,

$$\begin{aligned}\omega(x, y, 0) = &\exp\left(-\pi\left[(x-x_1)^2 + (y-y_1)^2\right]\right) \\ &+ \exp\left(-\pi\left[(x-x_2)^2 + (y-y_2)^2\right]\right),\end{aligned} \tag{6.45}$$

where their centers are initially located at $(x_1, y_1) = (3\pi/4, \pi)$ and $(x_2, y_2) = (5\pi/4, \pi)$. We utilize an arbitrary array of decaying Taylor-Green vortices as the unknown processes parameterized by $\mu = \{\mathrm{Re}\}$ i.e., we have

$$\boldsymbol{\pi}(x, y, t; \mathrm{Re}) = -\gamma\ e^{-t/\mathrm{Re}} \cos(3x)\cos(3y). \tag{6.46}$$

The parameter γ controls the strength of unknown physics compared to the known physics. The synthetic data for building the ROM is generated by solving Eq. 6.29 on the computational domain $(x, y) \in [0, 2\pi]$ with periodic boundary conditions and 256^2 spatial discretization. The system is evolved for 20 time units with a time-step size of $\Delta t = 1 \times 10^{-3}$. The data snapshots for the training are obtained for $\mathrm{Re} = \{200, 400, 600, 800\}$ and the trained models are evaluated for $\mathrm{Re} = \{1000, 1500, 2000, 3000\}$. Specifically, the physical variable of interest is the vorticity field, i.e., $\boldsymbol{u} = \{\omega\}$ and the unknown source term $\boldsymbol{\pi}(\cdot)$ is given in Eq. 6.46. We retain $N_r = 8$ POD modes for all Reynolds numbers and these POD modes captures more than 99% of the energy of the system as illustrated in Fig. 6.3. Instantaneous snapshots of the vorticity field for different Reynolds numbers at different time instances are displayed in Fig. 6.4. We note here that the vorticity field shown in Fig. 6.4 is obtained by solving the vorticity transport equation assuming that there is no unknown physics, i.e., setting $\gamma = 0.0$.

The LSTM network in the ML model consists of $N_l = 5$ layers with the first four layers as the LSTM layers and the dense layer with a linear activation function as the output layer. Each LSTM layer has $N_h = 80$ hidden units and tanh activation function is applied. From our experimentation with the hyperparameters, we found that the results were not improving by employing deeper network architecture and in some cases, deeper networks led to poor performance due to overfitting. We utilize $d = 3$,

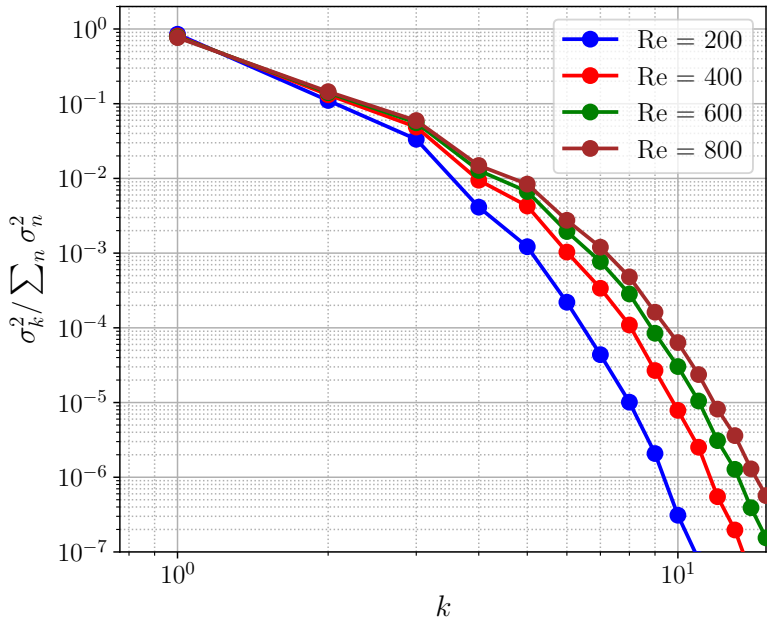

Figure 6.3: Amount of the kinetic energy captured by each POD modes.

i.e., the state of the system for three previous time steps are used to predict the future state. Since we are using the information of only d past temporally consecutive states as the input, the LSTM can capture dependencies up to d previous time steps. During the online deployment, the initial condition for the first d time steps is provided. This information is used to predict the forecast state at $(d+1)$th time step. Then the state of the system from $2-(d+1)$ is used to predict the forecast state at $(d+2)$th time step. This procedure is continued until the final time step. The PGML framework also uses the same LSTM network architecture and the features from the Galerkin projection model (i.e., GROM modal coefficients) are embedded into the third LSTM layer. We reiterate here that the layer at which the physics based features should be embedded is another hyperparameter of PGML framework and methods like neural architecture search can be used for complex problems [229]. The online deployment of the PGML framework is similar to the ML framework. We characterize the generalizability of the ML and PGML model through uncertainty estimate computed using the deep ensembles method where ensemble of neural networks are trained with different initialization of parameters [372, 138, 193]. In deep ensembles method, the prediction from the ensemble of neural networks is interpreted as the epistemic uncertainty. Despite the simplicity of this method, it is appealing due to its scalability and strong empirical results showing that the uncertainty estimate is as good as the Bayesian

neural networks [193]. The quantitative performance for ML and PGML framework is measured through the statistics of the weighted root mean squared error (WRMSE) defined as

$$\epsilon(i) = \sum_{k=1}^{N_r} \lambda_k \left(\frac{1}{N_t} \sum_{n=1}^{N_t} (\alpha_k^{(n)} - \tilde{\alpha}_k^{(n)}(i))^2 \right)^{1/2}, \tag{6.47}$$

where $\lambda_k = \sigma_k^2 / (\sum_{j=1}^{N_r} \sigma_j^2)$ represents the energy in each POD mode and $\tilde{\alpha}_k(i)$ is the prediction of the modal coefficients for ith ensemble. Each neural network in the ensemble set is initiated by using a different initial seed. The statistics of the WRMSE, i.e., the mean and the standard deviation are computed as follows

$$\bar{\epsilon} = \frac{1}{N_e} \sum_{i=1}^{N_e} \epsilon(i), \quad s_\epsilon = \left(\frac{\sum_{i=1}^{N_e} (\epsilon(i) - \bar{\epsilon})^2}{N_e - 1} \right)^{1/2}, \tag{6.48}$$

where N_e is the size of ensemble of the neural network. We set $N_e = 30$ for both ML and PGML models in our study.

Fig. 6.5 shows the temporal evolution of selected modal coefficients predicted by true projection, GP, ML, and PGML models at Re = 1500 for $\gamma = 0.01$. The ensemble-averaged modal coefficients predicted by the PGML model matches very accurately with the true projection compared to the GP and ML model. We also observe that the ensemble-average prediction by the ML model is also improved compared to the GP model. However, the prediction of the ML model is marred by high uncertainty compared to the PGML model prediction. From Fig. 6.4, we can notice that the flow is topologically very similar at different Reynolds numbers and the main difference is in the amplitude of the vorticity field. Therefore, the task of extrapolation for this problem is not very challenging for neural networks. However, in this study, we demonstrate the proof of concept to show the benefit of PGML over a purely data-driven ML model, and evaluating the PGML framework for topologically different data sets is a part of our future work. Fig. 6.6 depicts the evolution of selected modal coefficients predicted by true projection, GP, ML, and PGML models at Re = 3000 for $\gamma = 0.01$. The ensemble-averaged modal coefficient predicted by the ML model is very inaccurate with high uncertainty as the test Reynolds number is significantly different from the training Reynolds numbers. The magnitude of the unknown source term is relatively small for $\gamma = 0.01$, and the GP model can predict the modal coefficient with sufficient accuracy. The PGML model achieves the most

accurate prediction with less uncertainty as shown in Fig. 6.6. Here, we would like to emphasize that the accuracy of any supervised ML approach mostly depends on the quality and relevance of the training data set. Even though all ML models are trained from the same data set, the underpinning optimization problem yields different predictor model (i.e., a different local minimum) at each realization due to random initialization of the weights. One of our chief goals in developing the PGML approach is to reduce the uncertainty of these predictor models through an information or knowledge injection process. Our numerical results support that the uncertainty of the ML models could be improved significantly through injection of the GROM features that are derived from the known part of the governing equations.

In Fig. 6.7, the temporal evolution of the last mode (i.e., a_8) predicted by true projection, GP, ML, and PGML model is displayed for different magnitudes of the unknown source term. Overall, we can observe that the PGML framework can predict the modal coefficients with high confidence up to $\gamma = 0.04$ and the prediction for $\gamma = 0.1$ is less certain. One of the reasons behind inaccurate prediction at $\gamma = 0.1$ can be due to a very poor correlation between the known physics in the GP model and the actual physics of the system.

We also evaluate the capability of GP, ML, and PGML framework in reconstructing the vorticity field using the RMSE between the true vorticity field and the predicted vorticity field. The RMSE is defined as

$$\text{RMSE}(t) = \left(\frac{1}{N} \sum_{i=1}^{N} \left(\omega_T(\boldsymbol{x}_i, t) - \omega_R(\boldsymbol{x}_i, t) \right)^2 \right)^{1/2}, \tag{6.49}$$

where N is the spatial resolution (i.e., $N = N_x \times N_y$), ω_T is the true projection of the vorticity field, and ω_R is the vorticity field predicted by ROM (computed using Eq. 6.36). The ω_R is computed using the ensemble-averaged modal coefficients predicted by ML and PGML model. Fig. 6.8 shows the time evolution of RMSE for different magnitudes of the source term at Re = 1000 and Re = 1500. We can observe that when the complete physics of the vortex-merging process is known (i.e., $\gamma = 0.0$), the RMSE for both ML and PGML framework is less than the GP model at Re = 1000. However, at Re = 1500, the RMSE for the ML model is higher than the GP model. This is due to the poor extrapolation capability of a purely data-driven ML model. On the other hand, the PGML model can successfully use the physics-based information from the GP model leading to less RMSE compared to both GP and ML

models. The vorticity field predicted by the GP model quickly diverges from the true vorticity field in the presence of unknown physics and overall the PGML prediction is better than the ML model as illustrated in Fig. 6.8. The difference between the vorticity field at the final time $t = 20$ predicted by true projection modal coefficients and GP, ML, and PGML models is depicted in Fig. 6.9.

Table 6.1 reports the quantitative analysis of GP, ML, and PGML models at $\text{Re} = \{1000, 1500\}$ for different magnitudes of the unknown source term. For the ML and PGML models, the WRMSE mentioned in Table 6.1 are calculated using Eq. 6.50 and its statistics is computed with Eq. 6.48. For the GP model, we report WRMSE computed by using

$$\epsilon = \sum_{k=1}^{N_r} \lambda_k \left(\frac{1}{N_t} \sum_{n=1}^{N_t} (\alpha_k^{(n)} - a_k^{(n)})^2 \right)^{1/2}, \tag{6.50}$$

where α_k is the set of true projection coefficients, and a_k refers to the GROM coefficients. In Table 6.1, $\gamma = 0.0$ corresponds to the special case where the physics is completely available to us. In that case, if the closure error is also negligible there is no need to build such a stochastic learning machine (i.e., a neural network derived from true projection coefficients), so a physics-based model (e.g., GP) will provide an accurate prediction. Moreover, as verified by Table 6.1, we would expect that the GP approach should indeed yield more accurate results than the ML (or PGML) approach drawn from true projection coefficients, since the GP captures almost 100% of RIC at $\gamma = 0.0$. Furthermore, from Table 6.1 we can see that the WRMSE for the ML model is considerably higher for $\text{Re} = 1500$ at $\gamma = 0.0$ due to the poor extrapolation ability of pure data-driven models. However, the WRMSE for the PGML model is significantly lower than the ML. Yet, the WRMSE for the PGML model slightly higher than the GP model even though the reconstructed vorticity field is more accurate as illustrated in Fig. 6.8. The WRMSE assigns more weightage to the first few modes and therefore the inaccurate prediction for dominant modes leads to a higher value of WRMSE for the PGML model. Being the primary focus of this study, for higher values of γ, the dynamics of the system is not known exactly and it gives us an opportunity to introduce a stochastic neural network model as a predictive engine. As demonstrated in Table 6.1, the trustworthiness of the pure data-driven ML model is less due to the high value of the uncertainty in the prediction (i.e., high standard deviation as shown in parenthesis). However, once we incorporate physics-based features from the GP model in the PGML framework, the uncertainty in the prediction is reduced by almost

one order magnitude.

We also consider the hypothetical situation where $\delta \boldsymbol{u} = 0.94$ (so the physics is almost completely unknown and as shown in Figure 6.9, and unknown physics severely dominates the vortex merging processes), then it is not surprising that the PGML framework would not be in favor of the standard ML because of the injection of the unrepresentative physics. Considering a false-negative analogy, this further verifies the success of the PGML methodology and will automatically indicate that the injected physics is not representative of the underlying processes. In fact, this feature of the PGML framework could act as a diagnostic approach in many digital twin applications to physical systems. There are numerous simulation packages specifically designed for modeling and computing $\hat{\boldsymbol{f}}$ in these systems. On the other hand, exhaustive data sets become more and more available to generate versatile machine learning based predictive models. To this end, a confidence score $\delta \boldsymbol{u}$ defined in Eq. 6.4 can be utilized to assess the relative modeling strength of principled and data-driven models, where lower $\delta \boldsymbol{u}$ might indicate that the principled model is a reliable tool and the physics-based features from the principled model should be incorporated into data-driven models. This diagnostic aspect and the investigation of such an adaptive composite model will be the subject of our future studies.

To further investigate the limits of the proposed PGML approach, we carry out the quantitative analysis for Re = 2000 and Re = 3000. Table 6.2 lists the WRMSE and its statistics defined by Eq. 6.48 for higher Reynolds numbers. It can be observed that the WRMSE and the uncertainty in the prediction are significantly lower for the PGML model compared to the ML model for both Reynolds numbers. For all the numerical experiments conducted at different $\delta \boldsymbol{u}$ levels, it can be concluded that the PGML approach leads to significantly more accurate results compared to the ML approach, which clearly demonstrates the feasibility of the proposed physics injection approach for the projection based reduced order models.

Furthermore, we reiterate here that GROM features can be embedded at any layer of the neural network or perhaps at all hidden layers. We investigate the effect of embedding GROM features at the specific layer of the neural network and present the quantitative results in Table 6.3 for $\gamma = 0.01$ at different Reynolds numbers. We notice that the ensemble-averaged (i.e., $N_e = 30$ ensembles from different initialization) WRMSE and the uncertainty estimate is minimum at all Reynolds numbers when the GROM features are embedded at each of the hidden layers of the neural network. Although our systematic comparison in Table 6.3 highlights the underlying trends, the

γ	Re = 1000				Re = 1500			
	$\delta\boldsymbol{u}$	GP	ML	PGML	$\delta\boldsymbol{u}$	GP	ML	PGML
0.00	0.00	0.075	0.262 (0.190)	0.064 (0.012)	0.00	0.138	0.959 (0.632)	0.192 (0.024)
0.01	0.25	1.564	0.258 (0.194)	0.110 (0.012)	0.26	1.563	0.852 (0.521)	0.300 (0.029)
0.02	0.46	6.033	0.316 (0.113)	0.181 (0.026)	0.47	6.061	0.998 (0.458)	0.498 (0.040)
0.03	0.61	12.809	0.465 (0.387)	0.259 (0.040)	0.62	12.905	1.352 (0.582)	0.784 (0.092)
0.04	0.72	20.843	0.726 (0.591)	0.312 (0.047)	0.72	21.082	1.828 (0.992)	1.116 (0.149)
0.10	0.94	70.935	1.910 (0.804)	1.424 (0.356)	0.94	71.923	5.252 (2.387)	5.842 (6.227)

Table 6.1: The mean of weighted root mean square error $\bar{\epsilon}$ (and its standard deviation s_ϵ) defined by Eq. 6.48 in predicting the modal coefficients for the vortex merging flows at Re $= \{1000, 1500\}$. The training has been performed using data for Re $= \{200, 400, 600, 800\}$. Note that the control parameter γ adjusts the relative strength between known-physics and unknown-physics (i.e., $\gamma = 0$ refers to the special case when physics is fully known), and $\delta\boldsymbol{u}$ quantifies the level of unknown processes.

optimal neural network architecture for the PGML framework can be discovered using techniques like AutoML [135] and we consider this as part of our future work. Moreover, it is also worth mentioning that we intentionally use the same architecture in our comparisons between the ML and PGML models throughout our study. Intuitively, a PGML model might result in a simpler (fewer hyperparameters) model than a physics-agnostic ML model.

6.5 Discussion and Conclusion

Although advanced machine learning (ML) models like deep learning networks are powerful tools for finding patterns in complicated datasets, the number of trainable parameters (weights) quickly explodes as these neural network models become complex, adversely affecting their interpretability and hence their trustworthiness. Our chief aim in this study is to illustrate how physics can be injected into these neural networks to improve the trustworthiness of the overall system. With this in mind, we introduce a

γ	Re = 2000				Re = 3000			
	$\delta \boldsymbol{u}$	GP	ML	PGML	$\delta \boldsymbol{u}$	GP	ML	PGML
0.00	0.00	0.221	4.087 (2.179)	0.331 (0.062)	0.00	0.360	6.017 (0.833)	0.595 (0.147)
0.01	0.26	1.564	3.542 (2.003)	0.503 (0.086)	0.27	1.563	5.636 (0.904)	0.867 (0.219)
0.02	0.48	6.033	4.412 (2.297)	0.809 (0.095)	0.48	6.061	7.309 (1.036)	1.542 (0.474)
0.03	0.63	12.809	5.046 (2.364)	1.441 (0.342)	0.64	12.905	10.813 (2.811)	3.345 (1.071)
0.04	0.73	20.843	6.402 (3.409)	2.384 (0.589)	0.74	21.082	15.439 (3.931)	5.813 (1.519)
0.10	0.94	70.935	17.672 (14.194)	11.353 (6.816)	0.95	71.923	45.863 (13.155)	31.468 (9.623)

Table 6.2: The mean of weighted root mean square error $\bar{\epsilon}$ (and its standard deviation s_ϵ) defined by Eq. 6.48 in predicting the modal coefficients for the vortex merging flows at Re $= \{2000, 3000\}$. The training has been performed using data for Re $= \{200, 400, 600, 800\}$.

physics-guided ML (PGML) framework that fuses first-principles with neural-network models, and explore complementary physics versus statistics issues that arise in the development of the PGML framework. The PGML framework puts particular emphasis on the physics-based features and embeds them at an intermediate layer of the neural network. The robustness of the PGML framework is successfully demonstrated for the vortex-merging process in the presence of an array of Taylor-Green vortices as the unknown source term. Our results indicate that the PGML framework can give considerably accurate prediction in the extrapolation regime compared to its counterpart, i.e., pure data-driven ML model. The physics-based features from the GP model also ensures that the model uncertainty in the neural network prediction is reduced. The PGML framework is able to achieve very accurate prediction with a very small uncertainty at Reynolds number substantially different from the training. Finally, we also emphasize that the PGML framework is highly modular, and it naturally allows for the multi-fidelity model fusion in different branches of science and engineering.

The natural extension of the present study is to assess the PGML framework for much more complex flows, where there is a significant variation between the train

Model	Re = 1000	Re = 1500	Re = 2000	Re = 3000
PGML-1	0.108 (0.018)	0.259 (0.030)	0.446 (0.071)	1.119 (0.507)
PGML-2	0.114 (0.015)	0.286 (0.023)	0.450 (0.055)	0.774 (0.157)
PGML-3	0.110 (0.012)	0.300 (0.029)	0.503 (0.086)	0.867 (0.219)
PGML-A	0.108 (0.012)	0.267 (0.033)	0.397 (0.057)	0.596 (0.093)

Table 6.3: The mean of weighted root mean square error $\bar{\epsilon}$ (and its standard deviation s_ϵ) defined by Eq. 6.48 in predicting the modal coefficients for the vortex merging flows in case of $\gamma = 0.01$ at different Reynolds numbers. The training has been performed using data for Re $= \{200, 400, 600, 800\}$ and the physics-based features are embedded at different intermediate layer of the neural network. For example, in PGML-1 model, the GROM features are embedded at the first hidden layer. For the PGML-A model, GROM features are embedded at each hidden layer of the neural network.

and test data. Other unanswered questions are understanding how neural network assigns importance to physics-based features, at which layer shall the physics-based features be embedded, and methods from machine learning community can be utilized for this [323, 255]. Another direction for future work will be to use other uncertainty quantification methods like Monte Carlo dropouts, Bayesian neural networks to quantify model uncertainty.

Moreover, the PGML approach might lead to modular data-driven workflows across multiple scientific domains. Such design of a hierarchically sequential learning algorithm allows the embedding of simplified theories, possible symmetries, space-time correlations, scaling laws, and other physics-based kernels, multimodal data sources or features directly into deep neural network models. These physics-based features often assist the deep neural network models in constraining the output to a manifold of the physically realizable solution. Consequently, this concatenated neural network design can lead to improved generalizability for the extrapolation regime beyond the training data sets, as illustrated in this chapter in the context of projection based model reduction. In our future studies, we will also exploit how to develop an automatic-PGML framework to determine the optimal architecture for multifidelity physics injection.

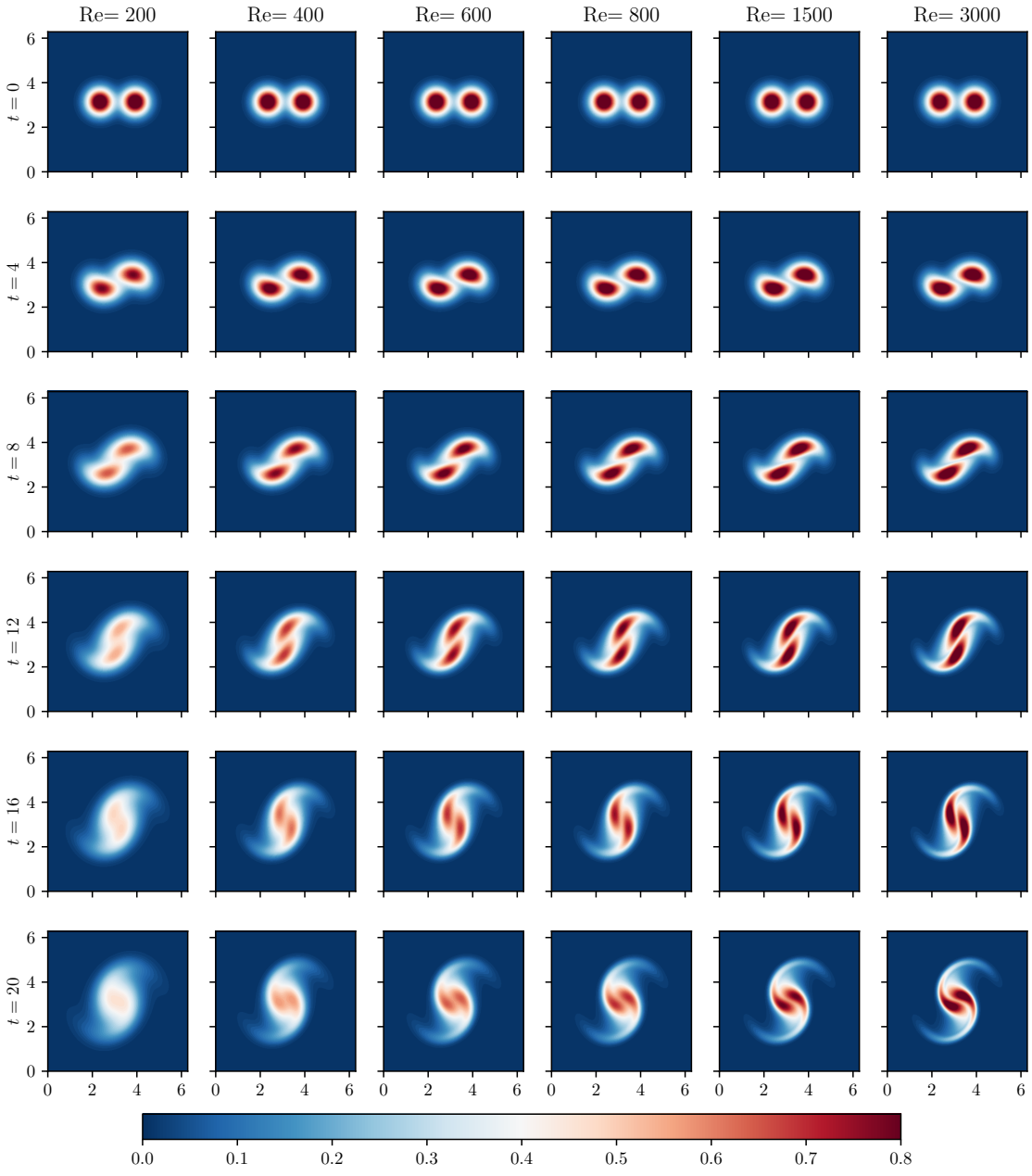

Figure 6.4: Temporal evolution of the vorticity field at different time instances for several Reynolds number investigated in this study. We note here that the vorticity field is presented for $\gamma = 0.0$.

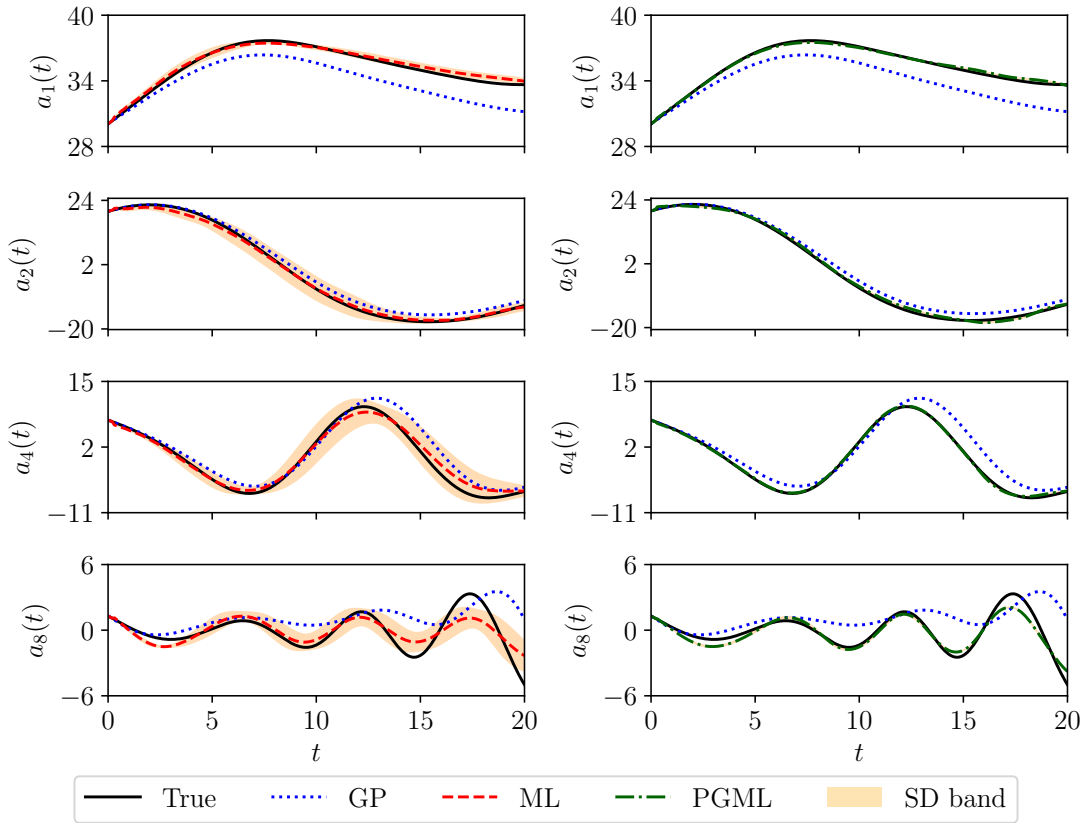

Figure 6.5: Temporal evolution of selected modal coefficients for the vortex-merger test case at $\mathrm{Re} = 1500$ and $\gamma = 0.01$. The ML (left) and PGML (right) represent the average of the modal coefficients predicted by all neural networks trained using MSE loss function with different seeds for initialization of the parameters. Although the ensemble-averaged ML model provides more accurate predictions than the GP model, there is a large standard deviation (SD) band over all ML model predictions.

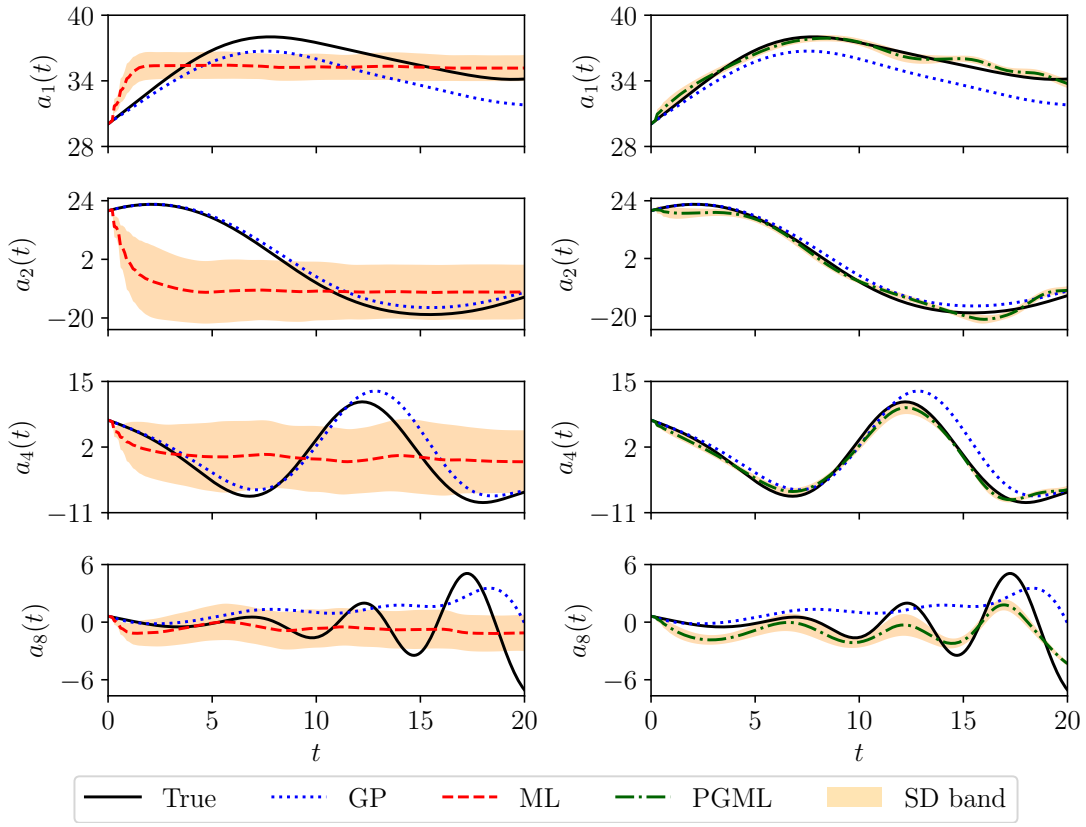

Figure 6.6: Temporal evolution of selected modal coefficients for the vortex-merger test case at Re = 3000 and $\gamma = 0.01$. The ML (left) and PGML (right) represent the average of the modal coefficients predicted by all neural networks trained using MSE loss function with different seeds for initialization of the parameters. The ensemble-averaged ML model provides very inaccurate predictions than the GP model and there is a large standard deviation (SD) band over all ML model predictions.

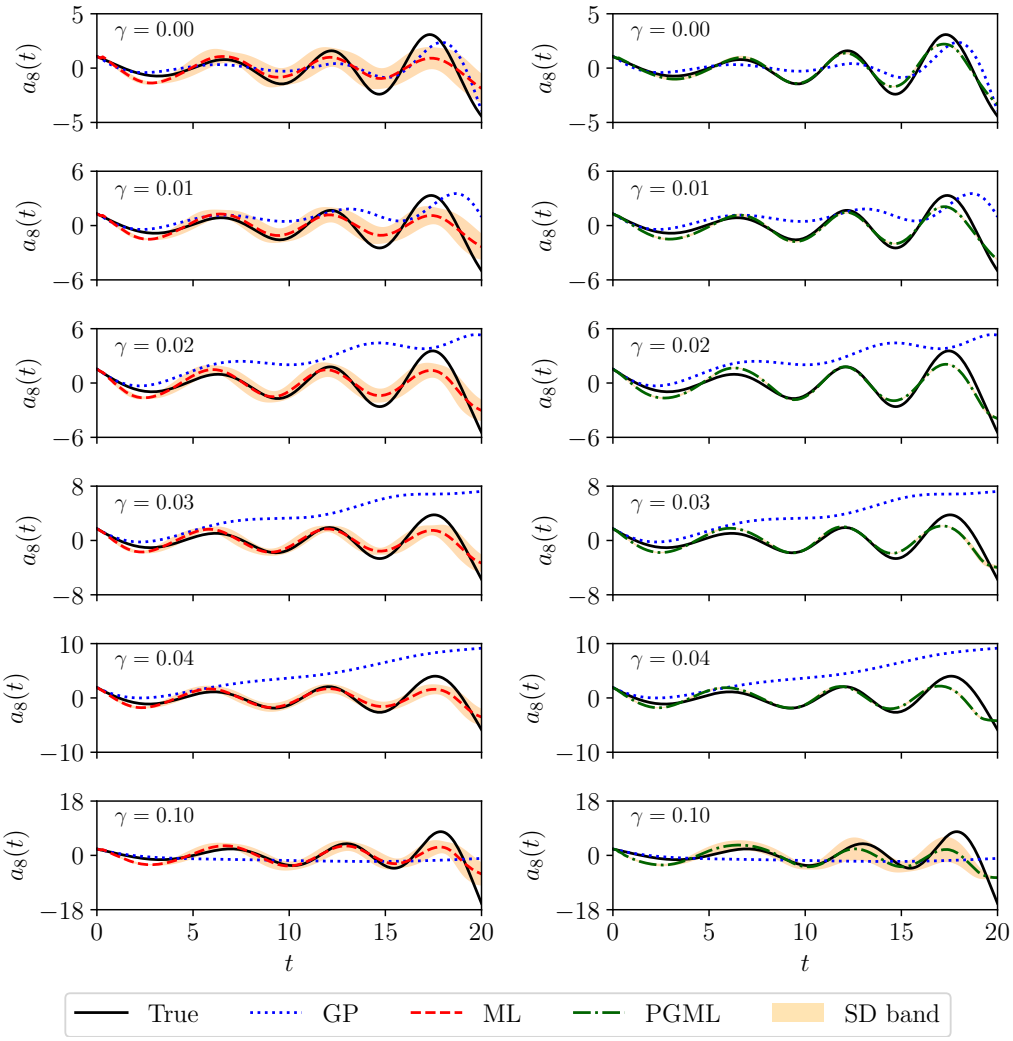

Figure 6.7: Temporal evolution of the last modal coefficient for the vortex-merger test case at Re = 1500 and for different magnitudes of the source term. The dashed red curve represent the average of the modal coefficients predicted by ML model (left), the PGML model (right).

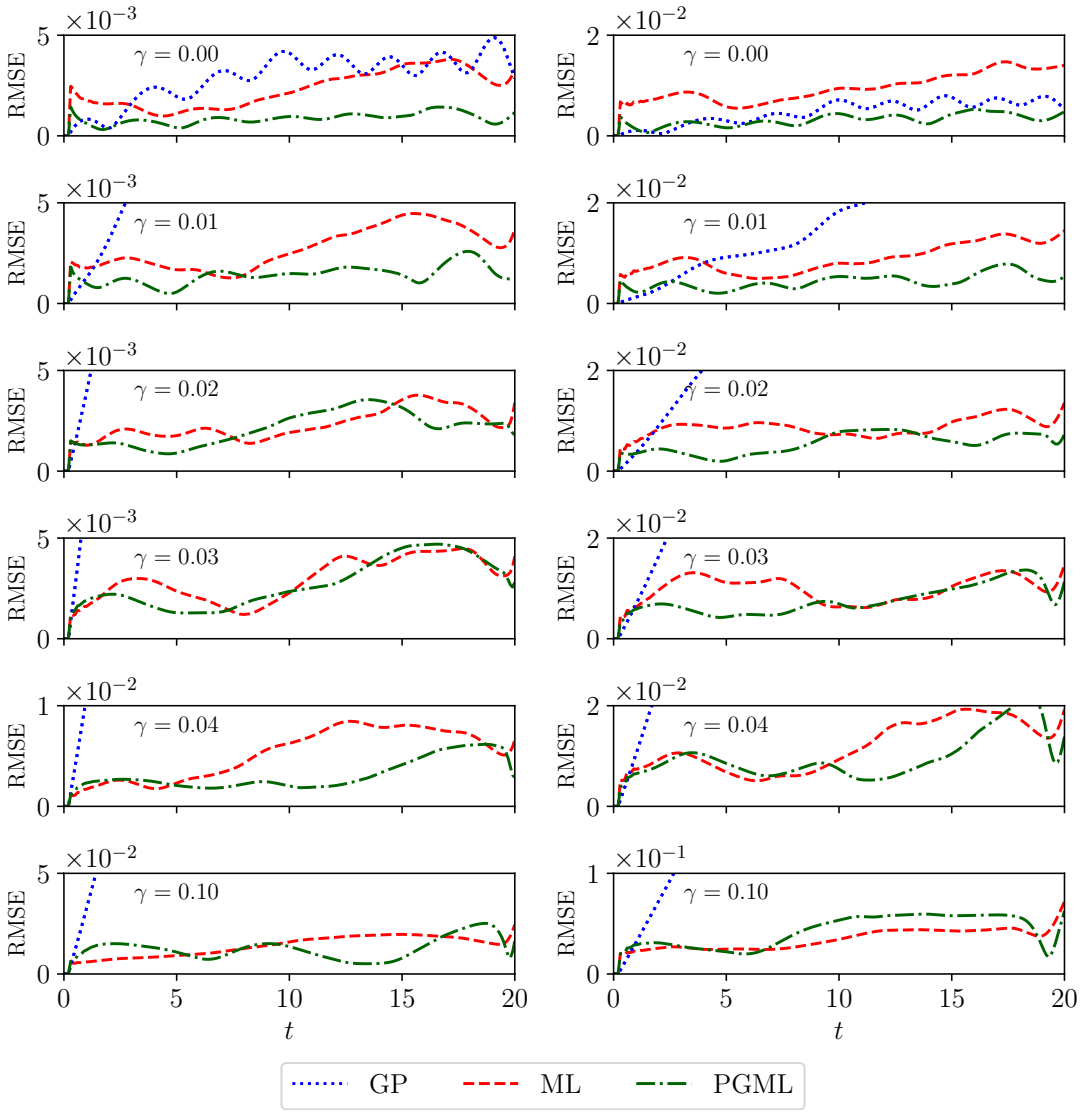

Figure 6.8: Temporal evolution of the RMSE of the vorticity field (defined in Eq. 6.49) for the vortex-merger test case at Re = 1000 (left) and Re = 1500 (right) for different magnitudes of the source term. The dashed red curve represent the average of the modal coefficients predicted by ML model (left), the PGML model (right).

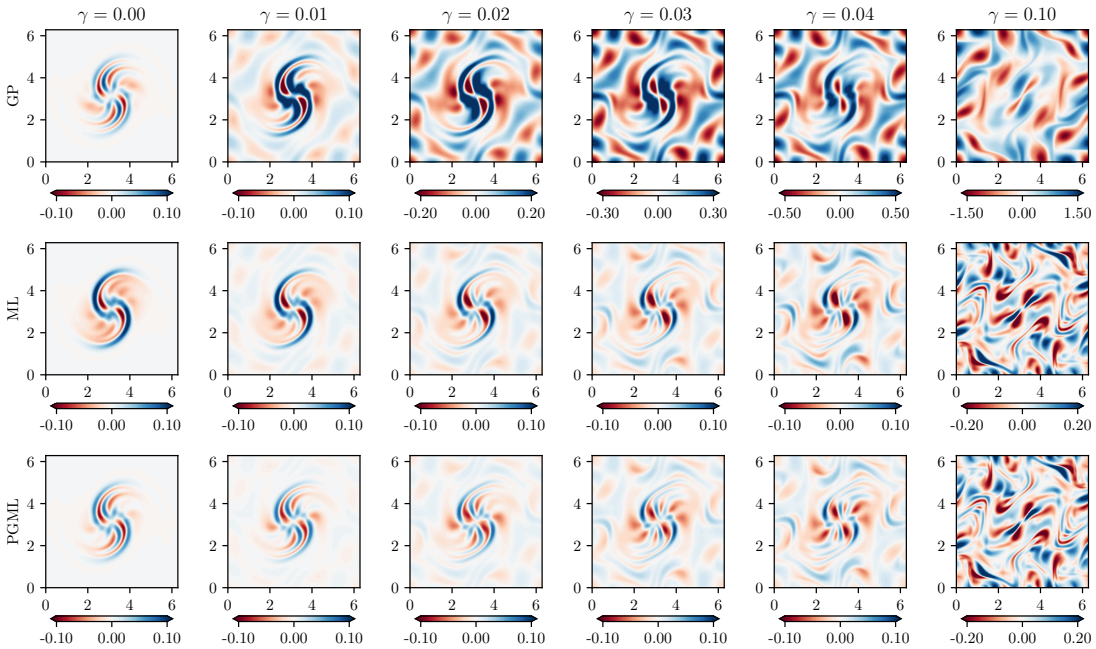

Figure 6.9: The error in the prediction of the vorticity field at the final time $t = 20$ for the GP model (top), ML model (middle), and PGML model (bottom). The error corresponds to the vortex-merger test case at Re = 1500.

CHAPTER VII

Data Assimilation in Latent Space with Non-intrusive Reduced Order Models

The contents of this chapter is under review at Journal of Advances in Modeling Earth Systems[1].

Abstract: There is a growing interest in developing data-driven reduced-order models for atmospheric and oceanic flows that are trained on data obtained either from high-resolution simulations or satellite observations. The data-driven models are non-intrusive in nature and offer significant computational savings compared to large-scale numerical models. These low-dimensional models can be utilized to reduce the computational burden of generating forecasts and estimating model uncertainty without losing the key information needed for data assimilation to produce accurate state estimates. This paper aims at exploring an equation-free surrogate modeling approach at the intersection of machine learning and data assimilation in Earth system modeling. With this objective, we introduce an end-to-end non-intrusive reduced-order modeling (NIROM) framework equipped with contributions in modal decomposition, time series prediction, optimal sensor placement, and sequential data assimilation. Specifically, we use proper orthogonal decomposition (POD) to identify the dominant structures of the flow, and a long short-term memory network to model the dynamics of the POD modes. The NIROM is integrated within the deterministic ensemble Kalman filter (DEnKF) to incorporate sparse and noisy observations at optimal sensor locations obtained through QR pivoting. The feasibility and the benefit of the proposed framework are demonstrated for the NOAA Optimum Interpolation Sea Surface Temperature (SST) V2 dataset. Our results indicate that the NIROM is stable for long-term forecasting and can model dynamics of SST with a reasonable level of accuracy. Furthermore, the prediction accuracy of the NIROM gets improved by almost one order of magnitude by the DEnKF algorithm.

[1]Pawar, S., & San, O. (2022). Equation-free surrogate modeling of geophysical flows at the intersection of machine learning and data assimilation. arXiv preprint arXiv:2205.13410.

7.1 Introduction

The integration of models and observations has greatly transformed the predictions of the Earth system, including weather forecasts and climate projections [119, 26]. Observations can be either used to estimate the initial condition for the prediction that is consistent with the present state of the Earth system through the process of data assimilation or to reduce model errors by advancing the representation of certain processes within a model. Data assimilation (DA) is a well-established method that involves combining information coming from the forecast model with available observations and are used extensively in numerical weather prediction (NWP) [221, 305, 260]. DA can also be viewed from the Bayesian perspective that involves fusing data (observations) with the prior knowledge (i.e., mathematical representation of physical processes; model output) to obtain an estimate of the distribution of the true state of the process [401, 162].

DA methods are usually classified into two types, variational approaches and sequential approaches [206]. Variational DA is formulated as the constrained optimization problem defined by a cost function to minimize the discrepancy between the prior knowledge (i.e., the computational model) and observations. On the other hand, the sequential DA involves evolving the state of the system with background information until observations get available, and then updating the system's state. One of the key components of the DA cycle is the forecast model. The forecast models used within DA are based on solving the governing equations or the best approximation of physical processes numerically using spatial and temporal discretization on a computational grid [27]. Despite their success, the current forecast models have difficulty representing complex processes like turbulence, convection and clouds [418, 312] leading to high uncertainty in the prediction [422]. Additionally, the computational cost of ensemble forecasting (using many realizations with perturbed initial conditions) with numerical models is huge. As a result, the forecast model is one of the major limiting factors in DA, and this has spurred interest in using data-driven methods for the Earth system modeling that can deliver both computational efficiency and better representation of physical processes derived from data [85, 43].

Data-driven methods have been applied to a wide range of problems in Earth system modeling [318]. For example, machine learning approaches have been explored for parameterization of subgrid-scale processes [316, 110, 40], precipitation nowcasting [345], superresolution of wind and solar data [358], and weather forecasting [399].

Some of the studies with the data-driven weather prediction (DDWP) have already started showing promising results with similar if not superior performance compared to state-of-the-art NWP models [344, 282, 317]. These DDWP models leverage deep learning methods like a convolutional neural network, Fourier neural operator, and recurrent neural network that are trained on data obtained from reanalysis products [315]. Furthermore, data-driven models can be augmented by taking up the information from physical knowledge, or at least respecting the conservation properties [167, 164, 60, 154]. Some of the work towards these directions include enforcing the physical laws or statistical constraint into the loss function [309, 405], tailoring the architecture design to enforce certain symmetry or conservation laws [249, 34, 288], symbolic regression for equation discovery [421], and end-to-end learning strategy [106].

These studies indicate that ML has the potential to improve scientific knowledge (and hence models), especially when we cannot express our understanding in the form of mathematical equations in physics-based models [114, 52]. DA could benefit a lot from ML, where a forecast model is replaced with a hybrid model that incorporates a neural network as a component of the physical model [150, 286] or as a complete replacement with a data-driven model for forecasting and state estimation [63, 291]. Another advantage of an ML-based emulator is the quick computation of backpropagation gradient with automatic differentiation that can be used as a replacement for expensive adjoint solvers within variational DA [232, 69]. Similarly, some of the challenges with ML such as handling uncertain and sparsely sampled data can be mitigated with DA. Brajard et al. [45] proposed a two-step process to learn the parameterization of unresolved processes where a neural network was trained to learn the model error obtained from the analysis state of the truncated model. Some other works at the intersection of DA and ML include iterative application of neural network and DA to learn the chaotic dynamics [44], enforcing conservation of mass constraint in DA [327], and for building tangent-linear and adjoint models of parameterization schemes in variational DA [134].

In this work, we seek to build a data-driven surrogate model trained on the analysis data and integrate it with a sequential DA cycle. More specifically, we will focus on a class of data-driven methods that are based on projection-based reduced-order modeling (ROM). These methods first extract the recurrent spatial structures from the high-dimensional data using singular value decomposition or convolutional autoencoder [51, 256]. The next step is to either solve the projected governing equations on the

dominant modes [333, 57] or learn the projected dynamics with data-driven methods [392, 306, 278]. The recurrent neural network is one of the most popular algorithms in modeling the dynamics of lower-dimensional latent space, and has been applied for a wide range of applications [419, 2]. One of the advantages of using projection-based ROM as a surrogate model is that it allows for DA on a reduced-order space rather than on a full-state and this latent assimilation framework leads to substantial speed improvement in contrast to DA [304, 4, 215, 293]. The key challenge with latent assimilation is to link the real-time observations in physical space with the latent observation space [68]. Our approach for latent observation space construction is by first identifying the near-optimal sensor locations using QR pivoting and then using the measurements at these discrete locations for reconstruction on a tailored basis [226].

This chapter is organized as follows. In Section 7.2.1, the surrogate modeling with proper orthogonal decomposition (POD) for dimensionality reduction, and a long short-term memory (LSTM) network for modeling dynamics is discussed. The detailed discussion on the deterministic ensemble Kalman filter (DEnKF) is provided in Section 7.2.2. The computation of the latent observation space is described in Section 7.2.3. In Section 7.3, the performance of the proposed framework is demonstrated for the NOAA Optimum Interpolation (OI) Sea Surface Temperature (SST) V2 dataset. Finally, concluding remarks are provided in Section 7.4.

7.2 Methods

In this section, we introduce different components of the proposed framework including surrogate modeling, sequential data assimilation, and optimal sensor placement strategy. The surrogate model is based on the projection-based model order reduction which relies on proper orthogonal decomposition (POD) and a recurrent neural network to model the dynamics of the latent space. This surrogate model is integrated within the sequential data assimilation to correct the forecast using real-time observations. The latent observation space is obtained by first reconstructing the full state from point sensor measurements and then projecting it onto POD bases.

7.2.1 Surrogate Modeling

Our surrogate model is constructed using the projection-based reduced-order modeling where the high-dimensional system is first compressed to a low-dimensional system and

the dynamics of the low-dimensional system is modeled. The compression is achieved by projecting the high-dimensional data onto a set of optimal linear basis functions obtained via POD [349, 32]. POD provides the optimal linear basis functions as they minimize the error between the true data and its projection in the L_2 sense compared to all other linear basis functions of the same dimension. Given the N_s snapshots of the data for a state variable $\boldsymbol{x} \in \mathbb{R}^n$, we can form the matrix $\boldsymbol{A}$ as follows

$$\boldsymbol{A} = \left[\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(N_s)} \right] \in \mathbb{R}^{n \times N_s}, \tag{7.1}$$

where $\boldsymbol{x}^{(k)}$ corresponds to an individual snapshot in time. Then, we perform the reduced singular value decomposition (SVD) as follows

$$\boldsymbol{A} = \boldsymbol{U} \boldsymbol{\Sigma} \boldsymbol{V}^T = \sum_{k=1}^{N_s} \sigma_k \boldsymbol{u}_k \boldsymbol{v}_k^T, \tag{7.2}$$

where $\boldsymbol{U} \in \mathbb{R}^{n \times N_s}$ is a matrix with orthonormal columns which corresponds to the left-singular vectors, $\boldsymbol{V} \in \mathbb{R}^{N_s \times N_s}$ is matrix with orthonormal columns representing the right-singular vectors, and $\boldsymbol{\Sigma} \in \mathbb{R}^{N_s \times N_s}$ is a matrix with non-negative diagonal entries, called singular values, and are arranged such that $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_{N_s} \geq 0$. For the dimensionality reduction task, only the first N_r columns of $\boldsymbol{U}$ and $\boldsymbol{V}$ (denoted as $\overline{\boldsymbol{U}}$ and $\overline{\boldsymbol{V}}$) are retained along with the upper-left $N_r \times N_r$ sub-matrix of $\boldsymbol{\Sigma}$ (denoted as $\overline{\boldsymbol{\Sigma}}$). The reduced-order approximation of the matrix $\boldsymbol{A}$ can be written as follows

$$\overline{\boldsymbol{A}} = \overline{\boldsymbol{U}}\ \overline{\boldsymbol{\Sigma}}\ \overline{\boldsymbol{V}}^T. \tag{7.3}$$

The total L_2 error between the snapshot data matrix $\boldsymbol{A}$ and its reduced-order approximation $\overline{\boldsymbol{A}}$ satisfies the following equalities [376]

$$\|\boldsymbol{A} - \overline{\boldsymbol{A}}\|_2 \ = \ \inf_{\substack{\boldsymbol{B} \in \mathbb{R}^{n \times N_s} \\ \text{rank}(\boldsymbol{B}) \leq N_r}} \|\boldsymbol{A} - \boldsymbol{B}\|_2 \ = \ \sigma_{N_r+1}, \tag{7.4}$$

where $\|\cdot\|_2$ is the matrix-2 norm. Equation 7.4 states that across all possible matrices $\boldsymbol{B} \in \mathbb{R}^{n \times N_s}$ of rank N_r or less, the matrix $\overline{\boldsymbol{A}}$ provides the best approximation in the L_2 sense and the error between $\boldsymbol{A}$ and its N_r-order approximation $\overline{\boldsymbol{A}}$ equals σ_{N_r+1}. From here on, the N_r columns of $\boldsymbol{U}$ are called as the POD modes or basis functions and we denote them as $\boldsymbol{\Phi} = \{\phi_k\}_{k=1}^{N_r}$. Once the POD modes are obtained, the compressed

latent space for a single state vector $\boldsymbol{x}$ can be written as

$$\boldsymbol{\alpha} = \boldsymbol{\Phi}^T \boldsymbol{x}, \tag{7.5}$$

where $\boldsymbol{\alpha}$ is the reduced-order approximation of the full state vector $\boldsymbol{x}$ and is also referred to as the POD modal coefficients. The reconstruction of the state vector is then computed as

$$\overline{\boldsymbol{x}} \;=\; \boldsymbol{\Phi}\boldsymbol{\alpha} \;=\; \boldsymbol{\Phi}\boldsymbol{\Phi}^T \boldsymbol{x}, \tag{7.6}$$

where $\overline{\boldsymbol{x}}$ is the optimal reconstruction of full state vector $\boldsymbol{x}$. The number of retained modes, i.e., N_r, is decided based on the variance of the data captured by the retained modes. The singular values σ_i give a measure of the quality of information that is retailed in N_r-order approximation of the matrix $\boldsymbol{A}$. The amount of the energy retained by POD modes can be calculated using the quantity called the relative information content (RIC) as follows

$$\mathrm{RIC}(N_r) = \frac{\sum_{j=1}^{N_r} \sigma_j^2}{\sum_{j=1}^{N_s} \sigma_j^2} \tag{7.7}$$

The second step of the surrogate modeling is to model the evolution of the latent variables. If the exact equations governing the physical system are known, intrusive approaches like Galerkin projection can be applied to build the ROM. The ROM is obtained by substituting the low-rank approximation into the full order model and then taking the inner product with test basis functions to yield a system of N_r ordinary differential equations (ODEs). However, for geophysical systems, the exact governing equations are unavailable or insufficient for the desired purpose (for example due to coarse grid resolution). On the other hand, the reasonably accurate and complete observational data for the evolution of the state of the system has been collected for many decades. This situation makes the equation-free techniques for predicting the future state of the dynamical system very attractive for multiscale and chaotic dynamical system modeling [168, 382, 281]. Recently, machine learning methods based on recurrent neural network have been applied successfully in many studies to build the non-intrusive ROM [419].

We apply the long short-term memory (LSTM) neural network to model the evolution of the latent variables. LSTM has been successfully applied in many studies dealing with modeling high-dimensional spatio-temporal chaotic dynamics of physical

systems [306, 248, 383]. LSTM is a type of recurrent neural network (RNN) that can capture the long-term dependencies in the evolution of time series data [142]. RNNs contain loops that allow them to persist information from one time step to another and can be expressed as

$$\mathbf{h}^{(t)} = f_{h\to h}(\mathbf{o}^{(t)}, \mathbf{h}^{(t-1)}), \tag{7.8}$$

$$\tilde{\mathbf{o}}^{(t+1)} = f_{h\to o}(\mathbf{h}^{(t)}), \tag{7.9}$$

where $\mathbf{h}^{(t)} \in \mathbb{R}^{d_h}$ is the hidden state at time t, $\mathbf{o}^{(t)} \in \mathbb{R}^{d_h}$ is the input vector at time t, $f_{h\to h}$ is the hidden to hidden mapping, and $f_{h\to o}$ is the hidden to output mapping. The output of the model is the forecast $\tilde{\mathbf{o}}^{(t+1)}$ at time step $t+1$.

One of the limitations of RNNs is vanishing (or exploding) gradient to capture the long-term dependencies. This problem occurs because the gradient is multiplied with the weight matrix repetitively during backpropagation through time (BPTT) [398]. The LSTM mitigates this issue by employing a memory cell composed of gating mechanism that decides which information to memorized or forgotten. The equations that implicitly define the mapping from hidden state of the previous time step (i.e., $\mathbf{h}^{(t-1)}$) and input vector at the current time step (i.e., $\mathbf{o}^{(t)}$) to the forecast hidden state (i.e., $\mathbf{h}^{(t)}$) can be written as follows

$$\mathbf{g}_f^{(t)} = \sigma(\mathbf{W}_f[\mathbf{h}^{(t-1)}, \mathbf{o}^{(t)}] + \mathbf{b}_f), \tag{7.10}$$

$$\mathbf{g}_i^{(t)} = \sigma(\mathbf{W}_i[\mathbf{h}^{(t-1)}, \mathbf{o}^{(t)}] + \mathbf{b}_i), \tag{7.11}$$

$$\tilde{\mathbf{c}}^{(t)} = \tanh(\mathbf{W}_c[\mathbf{h}^{(t-1)}, \mathbf{o}^{(t)}] + \mathbf{b}_c), \tag{7.12}$$

$$\mathbf{c}^{(t)} = \mathbf{g}_f^{(t)} \odot \mathbf{c}^{(t-1)} + \mathbf{g}_i^{(t)} \odot \tilde{\mathbf{c}}^{(t)}, \tag{7.13}$$

$$\mathbf{g}_o^{(t)} = \sigma(\mathbf{W}_o[\mathbf{h}^{(t-1)}, \mathbf{o}^{(t)}] + \mathbf{b}_o), \tag{7.14}$$

$$\mathbf{h}^{(t)} = \mathbf{g}_o^{(t)} \odot \tanh(\mathbf{c}^{(t)}), \tag{7.15}$$

where $\mathbf{g}_f^{(t)},\ \mathbf{g}_i^{(t)},\ \mathbf{g}_o^{(t)} \in \mathbb{R}^{d_h}$ are the forget gate, input gate, and output gate, respectively. The $\mathbf{o}^{(t)} \in \mathbb{R}^{d_i}$ is the input vector at time t, $\mathbf{h}^{(t)} \in \mathbb{R}^{d_h}$ is the hidden state, $\mathbf{c}^{(t)} \in \mathbb{R}^{d_h}$ is the cell state, $\mathbf{W}_f,\ \mathbf{W}_i,\ \mathbf{W}_c,\ \mathbf{W}_o \in \mathbb{R}^{d_h \times (d_h + d_i)}$ are the weight matrices, and $\mathbf{b}_f,\ \mathbf{b}_i,\ \mathbf{b}_c,\ \mathbf{b}_o \in \mathbb{R}^{d_h}$ are the bias vectors. The symbol $\odot$ denotes the element-wise multiplication, and σ is the sigmoid activation function, i.e., $\sigma(x) = (1/(1+e^{-x}))$. The above set of equations are unfolded in time to model the temporal dependencies in predicting future state $\mathbf{o}^{(t+1)}$ given $\mathbf{o}^{(t)}, \mathbf{o}^{(t-1)}, \cdots, \mathbf{o}^{(t-l)}$. The l is referred to as the

lookback window which governs how much amount of the old temporal information is required to forecast the future state of the system accurately.

When we utilize the LSTM network for constructing a surrogate model, the reduced-order state of the system at a future time step, i.e., $\boldsymbol{\alpha}^{(k+1)}$, is learned as the function of a short history of l past temporally consecutive reduced-order states as follows

$$\boldsymbol{\alpha}^{(k+1)} = \boldsymbol{F}(\underbrace{\boldsymbol{\alpha}^{(k)}, \boldsymbol{\alpha}^{(k-1)}, \ldots, \boldsymbol{\alpha}^{(k-l+1)}}_{\boldsymbol{\alpha}^{(k):(k-l+1)}}; \boldsymbol{\theta}), \tag{7.16}$$

where $\boldsymbol{F}(\cdot\,; \boldsymbol{\theta})$ is the nonlinear function parameterized by a set of parameters $\boldsymbol{\theta}$, and $\boldsymbol{\alpha}$ is the low-dimensional approximation of the full state vector, i.e., the POD modal coefficients given in Eq. 7.5. Since the surrogate model does not use any governing equations of the system, it is also referred to as the non-intrusive reduced-order model (NIROM).

7.2.2 Data Assimilation

We consider the dynamical system whose evolution can be represented as

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{M}_{t_k \to t_{k+1}}(\boldsymbol{x}^{(k)}) + \boldsymbol{w}^{(k+1)}, \tag{7.17}$$

where $\boldsymbol{x}^{(k)} \in \mathbb{R}^n$ is the state of the system at discrete time t_k, and $\boldsymbol{M}_{t_k \to t_{k+1}} : \mathbb{R}^n \to \mathbb{R}^n$ is the nonlinear model operator that defines the evolution of the system over the interval $[t_k, t_{k+1}]$. The term $\boldsymbol{w}^{(k+1)}$ denotes the model error that takes into account any type of uncertainty in the model that can be attributed to boundary conditions, imperfect models, etc. Let $\boldsymbol{z}^{(k)} \in \mathbb{R}^m$ be observations of the state vector obtained from sparse, noisy measurements and can be written as

$$\boldsymbol{z}^{(k+1)} = \boldsymbol{q}(\boldsymbol{x}^{(k+1)}) + \boldsymbol{v}^{(k+1)}, \tag{7.18}$$

where $\boldsymbol{q}(\cdot)$ is a nonlinear function that maps $\mathbb{R}^n \to \mathbb{R}^m$, and $\boldsymbol{v}^{(k+1)} \in \mathbb{R}^m$ is the measurement noise. We assume that the measurement noise is a white Gaussian noise with zero mean and the covariance matrix $\boldsymbol{R}^{(k+1)}$, i.e., $\boldsymbol{v}^{(k+1)} \sim \mathcal{N}(0, \boldsymbol{R}^{(k+1)})$. Additionally, the noise vectors $\boldsymbol{w}^{(k+1)}$ and $\boldsymbol{v}^{(k+1)}$ are assumed to be uncorrelated to each other at all time steps.

The sequential DA can be considered as a Bayesian inference framework that estimates the state $\boldsymbol{x}^{(k+1)}$ of the system given the observations up to time t_{k+1}, i.e.,

$\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(k+1)}$. When we utilize observations to estimate the state of the system, we say that the data are assimilated into the model, and use the notation $\widehat{\boldsymbol{x}}^{(k+1)}$ to denote an analyzed state estimate of the system at time t_{k+1}. When all the observations before (but not including) time t_{k+1} are applied for estimating the state of the system, then we call it the forecast estimate and denote it as $\boldsymbol{x}_f^{(k+1)}$. The ensemble Kalman filter (EnKF) [54] follows the Monte Carlo estimation method to approximate the covariance matrix in the Kalman filter equations [161]. Instead of modeling the exact evolution of a probability density function under nonlinear dynamics, ensemble methods maintain an empirical approximation to the target distribution in the form of a set of ensemble members $\widehat{\boldsymbol{X}}^{(k)}(i)$ for $i = 1 \ldots N$. We begin by initializing the state of the system for different ensemble members $\widehat{\boldsymbol{X}}^{(0)}(i)$ drawn from the distribution $\mathcal{N}(\widehat{\boldsymbol{x}}^{(0)}, \boldsymbol{P}^{(0)})$, where $\widehat{\boldsymbol{x}}^{(0)}$ represents the best-known state estimate at time t_0, and $\boldsymbol{P}^{(0)}$ is the initial covariance error matrix.

The propagation of the state for each ensemble member over the time interval $[t_k, t_{k+1}]$ can be written as

$$\boldsymbol{X}_f^{(k+1)}(i) = \boldsymbol{M}_{t_k \to t_{k+1}}(\widehat{\boldsymbol{X}}^{(k)}(i)) + \boldsymbol{w}^{(k+1)}. \tag{7.19}$$

The prior state and the covariance matrix are approximated using the sample mean and error covariance matrix $\boldsymbol{P}_f^{(k+1)}$ as follows

$$\boldsymbol{x}_f^{(k+1)} = \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{X}_f^{(k+1)}(i), \tag{7.20}$$

$$\boldsymbol{A}_f^{(k+1)}(i) = \boldsymbol{X}_f^{(k+1)}(i) - \boldsymbol{x}_f^{(k+1)}, \tag{7.21}$$

$$\boldsymbol{P}_f^{(k+1)} = \frac{1}{N-1} \sum_{i=1}^{N} \boldsymbol{A}_f^{(k+1)}(i)(\boldsymbol{A}_f^{(k+1)}(i))^{\mathrm{T}}, \tag{7.22}$$

where the superscript T denotes the transpose, and $\boldsymbol{A}_f^{(k+1)}(i)$ is the anomalies between the forecast estimate for the ith ensemble and the sample mean. Once the observations get available at time t_{k+1}, the forecast state estimate is assimilated using the Kalman filter analysis equation as follows

$$\widehat{\boldsymbol{x}}^{(k+1)} = \boldsymbol{x}_f^{(k+1)} + \boldsymbol{K}^{(k+1)}[\boldsymbol{z}^{(k+1)} - \boldsymbol{q}(\boldsymbol{x}_f^{(k+1)})]. \tag{7.23}$$

Unlike the EnKF algorithm, the DEnKF does not employ any perturbed observa-

tions. The Kalman gain matrix is computed using its square root version (without storing or computing $\boldsymbol{P}_f^{(k+1)}$ explicitly) as follows

$$\boldsymbol{K}^{(k+1)} = \frac{\mathcal{A}_f^{(k+1)}(\boldsymbol{Q}^{(k+1)}\mathcal{A}_f^{(k+1)})^{\mathrm{T}}}{N-1}\left[\frac{(\boldsymbol{Q}^{(k+1)}\mathcal{A}_f^{(k+1)})(\boldsymbol{Q}^{(k+1)}\mathcal{A}_f^{(k+1)})^{\mathrm{T}}}{N-1} + \boldsymbol{R}^{(k+1)}\right]^{-1}, \tag{7.24}$$

where $\boldsymbol{Q} \in \mathbb{R}^{m\times n}$ is the Jacobian of the observation operator (i.e., $Q_{kl} = \frac{\partial q_k}{\partial x_l}$), and the matrix $\mathcal{A}_f^{(k+1)} \in \mathbb{R}^{n\times N}$ is concatenated as follows

$$\mathcal{A}_f^{(k+1)} = [\boldsymbol{A}_f^{(k+1)}(1), \boldsymbol{A}_f^{(k+1)}(2), \ldots, \boldsymbol{A}_f^{(k+1)}(N)]. \tag{7.25}$$

The anomalies for all ensemble members are then updated separately with half the Kalman gain as shown below

$$\widehat{\boldsymbol{A}}^{(k+1)}(i) = \boldsymbol{A}_f^{(k+1)}(i) - \frac{1}{2}\boldsymbol{K}^{(k+1)}\boldsymbol{Q}^{(k+1)}\boldsymbol{A}_f^{(k+1)}(i). \tag{7.26}$$

The state for all ensemble members is updated by adding ensemble anomalies to analysis state estimate and can be written as

$$\widehat{\boldsymbol{X}}^{(k+1)}(i) = \widehat{\boldsymbol{x}}^{(k+1)} + \lambda \cdot \widehat{\boldsymbol{A}}^{(k+1)}(i), \tag{7.27}$$

where λ is the inflation factor to account for modeling errors. Inflation and covariance localization approaches are usually used in the EnKF framework to mitigate small number of ensembles [147, 19, 1]. The above ensembles are used as initial ensembles for the next assimilation cycle and the procedure is continued.

The DA procedure described so far corresponds to the full state vector of the system. However, in this study, the forward model is replaced with the surrogate model, and therefore, we perform the data assimilation in the latent space. The similar ideas have also been used in other studies that deals with the data assimilation for reduced-order models [4, 293, 232]. Once the LSTM network is trained, it is used to forecast the future state of the POD modal coefficients in an auto-regressive manner [283]. The evolution of the reduced-order model over the interval $[t_k, t_{k+1}]$ with the LSTM network is given as follows

$$\boldsymbol{\alpha}_f^{(k+1)} = \boldsymbol{F}_{t_k \to t_{k+1}}(\widehat{\boldsymbol{\alpha}}^{(k)}), \tag{7.28}$$

where $\boldsymbol{\alpha}_f^{(k+1)}$ is the low-dimensional forecast estimate of the system, and $\widehat{\boldsymbol{\alpha}}^{(k)}$ is the low-dimensional analyzed state of the system at time t_k. Once the observations at time t_{k+1} gets available, they need to be processed to obtain the *latent* observations $\widetilde{\boldsymbol{\alpha}}^{(k+1)}$. Usually, the observations are available at very few sparse locations, i.e., $m << n$, and therefore, they need to be mapped from observation-space $\mathbb{R}^m$ to state-space $\mathbb{R}^n$ through some reconstruction technique. Peyron et al. [293] used the simple interpolation to learn this mapping while applying data assimilation to the indoor air quality problem. We utilize the POD reconstruction augmented with QR pivoting for learning the map from observations to full state, and its further details are provided in Section 7.2.3. The analysis equation in the latent space can be written as follows

$$\widehat{\boldsymbol{\alpha}}^{(k+1)} = \boldsymbol{\alpha}_f^{(k+1)} + \widetilde{\boldsymbol{K}}^{(k+1)}[\widetilde{\boldsymbol{\alpha}}^{(k+1)} - \boldsymbol{q}(\widetilde{\boldsymbol{\alpha}}_f^{(k+1)})]. \tag{7.29}$$

The Kalman gain matrix in Eq.7.29 is calculated as follows

$$\widetilde{\boldsymbol{K}}^{(k+1)} = \frac{\widetilde{\mathcal{A}}_f^{(k+1)}(\widetilde{\boldsymbol{Q}}^{(k+1)}\widetilde{\mathcal{A}}_f^{(k+1)})^{\mathrm{T}}}{N-1}\left[\frac{(\widetilde{\boldsymbol{Q}}^{(k+1)}\widetilde{\mathcal{A}}_f^{(k+1)})(\widetilde{\boldsymbol{Q}}^{(k+1)}\widetilde{\mathcal{A}}_f^{(k+1)})^{\mathrm{T}}}{N-1} + \widetilde{\boldsymbol{R}}^{(k+1)}\right]^{-1}, \tag{7.30}$$

where $\widetilde{\boldsymbol{Q}} \in \mathbb{R}^{N_r \times N_r}$ is the Jacobian of the *latent* observation operator, and the matrix $\widetilde{\mathcal{A}}_f^{(k+1)} \in \mathbb{R}^{N_r \times N}$ is formed in latent space similar to Eq.7.25.

7.2.3 Reconstruction from Discrete Sensor Locations

As discussed previously, we need to reconstruct the full state of the system from limited number of discrete sensor locations. The low-rank approximation methods based on POD modes are one of the most popular method for the reconstruction task, where the sensor data is used to estimate the POD modal coefficients [83, 226]. Additionally, the QR decomposition with column pivoting is used for the near-optimal sensor placement in contrast to random sensor placement. We are interested in estimating the full state of the system given the sensor measurements $\boldsymbol{s} \in \mathbb{R}^m$ at discrete locations. For this case, we have $s_i = x_j$ for some $1 \leq i \leq m$ and $1 \leq j \leq n$. We can write this as follows

$$\boldsymbol{s} = \boldsymbol{\Theta}\boldsymbol{x}, \tag{7.31}$$

where $\boldsymbol{\Theta} \in \mathbb{R}^{m \times n}$ is constructed by taking m rows of $n \times n$ identity matrix. Therefore, each row of the matrix $\boldsymbol{\Theta}$ will consist of all zeros except for the corresponding observation location, where it will have the value of one. We use the first m POD modes obtained from singular value decomposition of the matrix $\boldsymbol{A}$ given in Eq. 7.1 for the reconstruction task and is denoted as $\boldsymbol{\Psi} \in \mathbb{R}^{n \times m}$ (i.e., the first m columns of $\boldsymbol{U}$). The $\boldsymbol{\Psi}$ is also referred to as the tailored basis functions in the literature. The approximation of the full state is given by

$$\boldsymbol{x} \approx \widetilde{\boldsymbol{x}} = \boldsymbol{\Psi}\mathbf{a}, \tag{7.32}$$

where $\mathbf{a} \in \mathbb{R}^m$. Once the sensor measurements gets available, the POD modal coefficients for the new sample can be calculated as follows

$$\boldsymbol{s} \approx \boldsymbol{\Theta}\boldsymbol{\Psi}\mathbf{a}, \tag{7.33}$$

$$\mathbf{a} = (\boldsymbol{\Theta}\boldsymbol{\Psi})^{-1}\boldsymbol{s}. \tag{7.34}$$

Once the POD modal coefficients for the new sample is determined, the full state of the system can be reconstructed using Eq. 7.32. The *latent* observations for data assimilation are computed by projecting the reconstructed data onto the POD basis functions $\boldsymbol{\Phi}$ as follows

$$\widetilde{\boldsymbol{\alpha}} = \boldsymbol{\Phi}^T\widetilde{\boldsymbol{x}}. \tag{7.35}$$

So far, we did not discuss the choice of the matrix $\boldsymbol{\Theta}$. The choice of the sensor locations can have a significant impact on the accuracy of the full state reconstruction. The sensor locations can be chosen either randomly or based on some heuristics or intuition of the physical problem. There are several techniques like optimal experimental design [160] and Bayesian criteria [185] that can be used to determine the optimal sensor locations for moderately sized problems. The QR factorization is a powerful and robust data-driven method to determine the near-optimal sensor locations solely based on the data [83, 226]. In this method, we perform the QR decomposition with column pivoting of the matrix $\boldsymbol{\Psi}^T$. The QR decomposition calculates a column permutation matrix $\boldsymbol{\mathcal{C}}^T \in \mathbb{R}^{n \times n}$, an orthogonal matrix $\boldsymbol{\mathcal{Q}} \in \mathbb{R}^{m \times m}$, and an upper triangular matrix $\boldsymbol{\mathcal{R}} \in \mathbb{R}^{m \times n}$ such that $\boldsymbol{\Psi}^T\boldsymbol{\mathcal{C}}^T = \boldsymbol{\mathcal{Q}}\boldsymbol{\mathcal{R}}$. The greedy approximation of the optimal sensor locations is obtained from the first m rows of matrix $\boldsymbol{\mathcal{C}}$, i.e., by setting $\boldsymbol{\Theta} = \overline{\boldsymbol{\mathcal{C}}}$, where $\overline{\boldsymbol{\mathcal{C}}}$ corresponds to first m rows of the matrix $\boldsymbol{\mathcal{C}}$ [226, 404]. We note here that one can also use the QR decomposition method for oversampled case,

where the number of sensor exceeds the number of tailored basis functions used for reconstruction.

7.3 Results and Discussion

We first describe the NOAA Optimum Interpolation (OI) Sea Surface Temperature (SST) V2 dataset used in this study for building a surrogate model and then integrating it within the data assimilation cycle. Then we present our numerical experimental results where we analyze the performance of our surrogate model and surrogate model assisted data assimilation framework.

7.3.1 Dataset and Preprocessing

We use the NOAA OI SST V2 analysis dataset for building our surrogate model. The analysis uses in situ (ship and buoy) and satellite SSTs plus SSTs simulated by the sea-ice cover. Before computing the analysis, the satellite data is adjusted for biases using the method of Reynolds [321] and Reynolds and Marsico [322]. This dataset consists of the weekly average sea surface temperature on a 1° latitude $\times$ 1° longitude global grid (180×360). The SST dataset exhibits a strong periodic structure due to seasonal fluctuations. Despite this seasonal periodicity, complex ocean dynamics lead to rich flow physics in this dataset. This dataset has been used in number of recent studies on flow reconstruction [55], geophysical emulation [229], and dynamic mode decomposition [190]. Here, we use the data from October 1981 to December 2000 (1000 snapshots) for building a surrogate model and the data from January 2001 to June 2018 (914 snapshots) for comparing the performance of the surrogate model for forecasting.

7.3.2 Numerical Experiments

In this subsection, we detail the numerical experiment design to assess the performance of NIROM and the integration of NIROM with DA, i.e., NIROM-DA. First, we use the data from October 1981 to December 2000 (1000 snapshots) to identify the POD basis functions. The masking operation is used to remove the data that correspond to the land area prior to its utilization for surrogate modeling. The temporal mean is also subtracted from the data and the POD is carried out for the unsteady component of the SST field. The first four POD modes capture approximately 90% of the variance

of the data and are enough to capture the long-term and seasonal trends in the SST. Therefore, we fix $N_r = 4$ in this study. The first four POD bases computed with POD are shown in Fig. 7.1. The next step is to train the LSTM network to learn the dynamics of a reduced-order system and a lookback window of $l = 4$ is used. Once the LSTM is trained, the model is deployed in auto-regressive manner for dynamical system forecasting. In the auto-regressive deployment, the initial condition for l time steps is used to predict the forecast state of the system at $(l+1)$th time step. Then the state of the system from $(2) - (l+1)$ is used to determine the forecast state at $(l+2)$th time step. The LSTM network architecture used in this study employs skip-connection after every hidden layer and is displayed in Fig. 7.2. The input and output data for training the LSTM network is scaled between $[-1, 1]$ to accelerate the training. The LSTM network utilizes three layers of stacked cells, 80 neurons per LSTM cell, and a ReLU activation function. The LSTM network is trained using an Adam optimizer with the learning rate of 1×10^{-3} and the weights of the network are initialized with the random normal initializer. The hyperparameters of the network are determined based on previous studies [289], and considerations of computational efficiency.

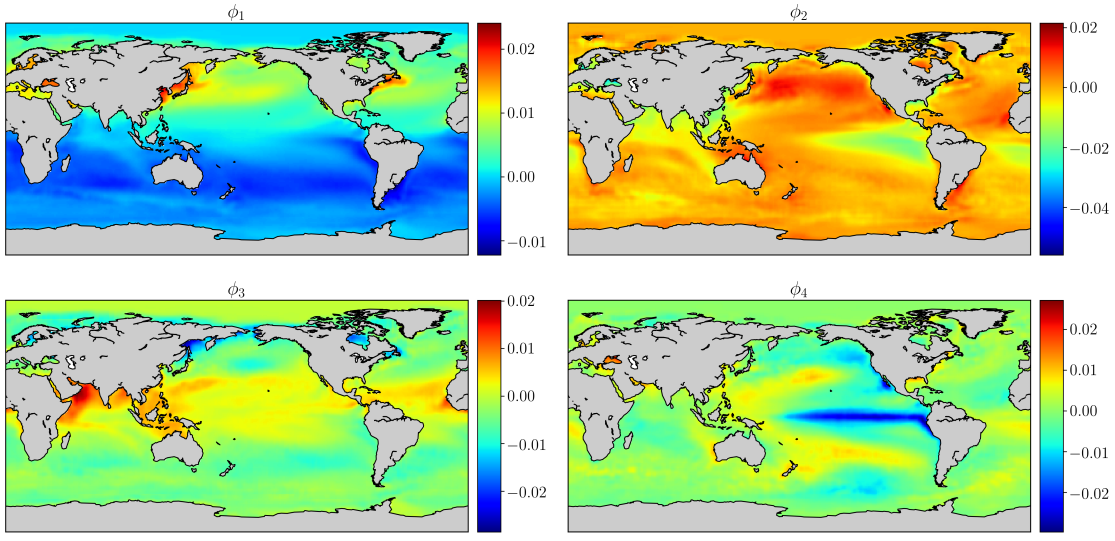

Figure 7.1: The first four leading POD modes extracted from the SST dataset. The temporal mean was subtracted from the data and the POD modes are computed for he unsteady component of the SST field.

Once the surrogate model for SST is built, it is integrated into the DA cycle. The performance of the NIROM-DA framework is analyzed using a twin experiment for

the period of January 2001 to June 2018. This data was not utilized in any stage of the NIROM construction. This ensures that there is no overlap in the training and online deployment of the surrogate model. For our twin experiment, the observations are generated by adding noise drawn from the Gaussian distribution with zero mean and the covariance matrix $\boldsymbol{R}_k$, i.e., $\boldsymbol{v}_k \sim \mathcal{N}(0, \boldsymbol{R}_k)$. We use $\boldsymbol{R}_k = \sigma_b^2 \boldsymbol{I}$, where σ_b is the standard deviation of measurement noise and is set at $\sigma_b = 1$. The observations are assumed to be collected only at discrete locations and the sensor locations are determined using the QR pivoting. We utilize only 300 discrete sensors for the reconstruction and this corresponds to less than 1% of the full state of the system. This number is also very close to the optimal rank truncation threshold determined for this dataset from previous studies [226]. The number of tailored basis functions is also set equal to the number of sensors to avoid the intractable computation associated with the oversampled case. As discussed in Section 7.2.3, the tailored basis functions are just the POD modes, and we can measure the variance of the data captured by these modes using their singular values. The first 300 POD modes capture approximately more than 99.5% of the variance of the data and this leads to the improved reconstruction of small-scale features. Fig. 7.3 depicts the near-optimal sensor locations determined through QR pivoting.

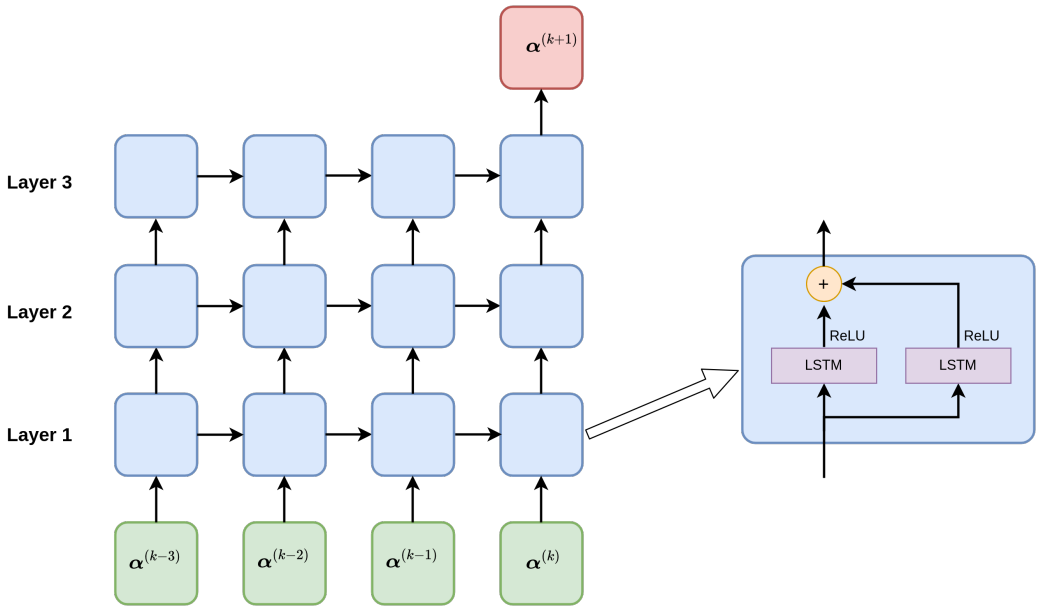

Figure 7.2: LSTM network architectures used for modeling the dynamics of reduced-order dynamical system. The LSTM is trained to predict the future forecast state of the system based on the previous four consecutive states of the system.

The initialization of ensembles is very important for the sequential DA [148]. We

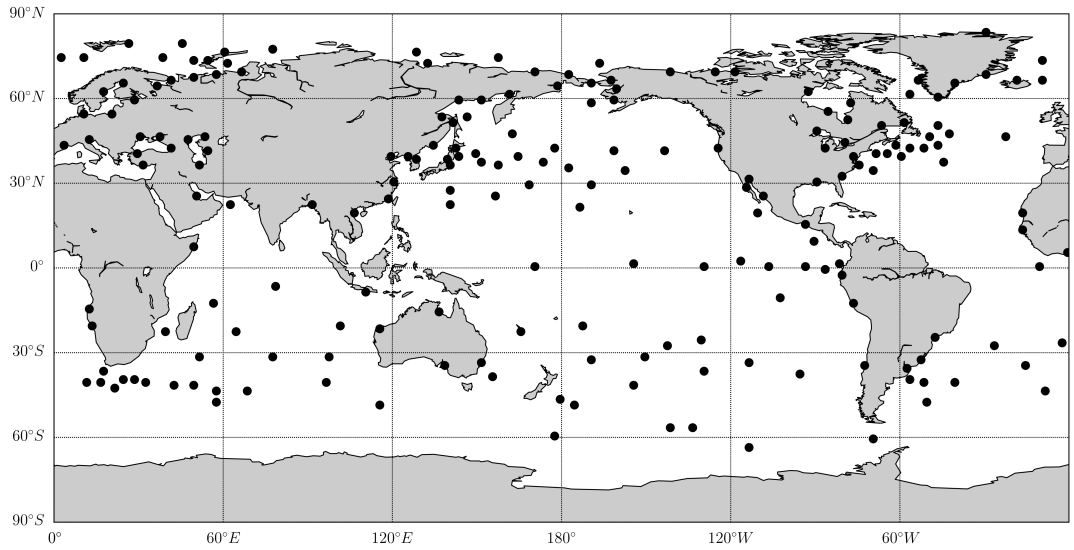

Figure 7.3: Optimal sensor locations for reconstruction obtained using QR. These locations are informative about the ocean dynamics in contrast to random selection of sensor locations.

experimented with three methods for initialization of ensembles, the first method is using random snapshots collected over some time to initialize the ensembles, the second one is adding random perturbation to the full state and then projecting it onto POD basis functions, and the third method is to add a random perturbation to the reduced-order state of the system. The first method is not suitable for our problem because of the seasonality in the dataset and this leads to the phase difference between the initial condition of ensembles. The second method did not lead to sufficient variability in the initial condition of the ensembles. The third approach ensures that the initial condition for all ensembles is sufficiently different without having any phase difference. Since we are doing both the forecast and data assimilation in latent space, the approach of adding a random perturbation to the true latent space is more suitable for our problem. Once all the ensembles are initialized, the forecasting is started using the NIROM in an auto-regressive manner. The observations are assimilated every third week to correct the initial condition for the future forecast. As discussed previously, the latent observations are obtained by first reconstructing the full state using the information at discrete sensor locations and tailored basis functions. The full state is then projected onto POD modes to compute latent observations for assimilation. The number of ensembles is set at 40 and the inflation factor of 1.5 is used to account for model error.

Fig. 7.4 shows the evolution of POD modal coefficients for the forecasting period,

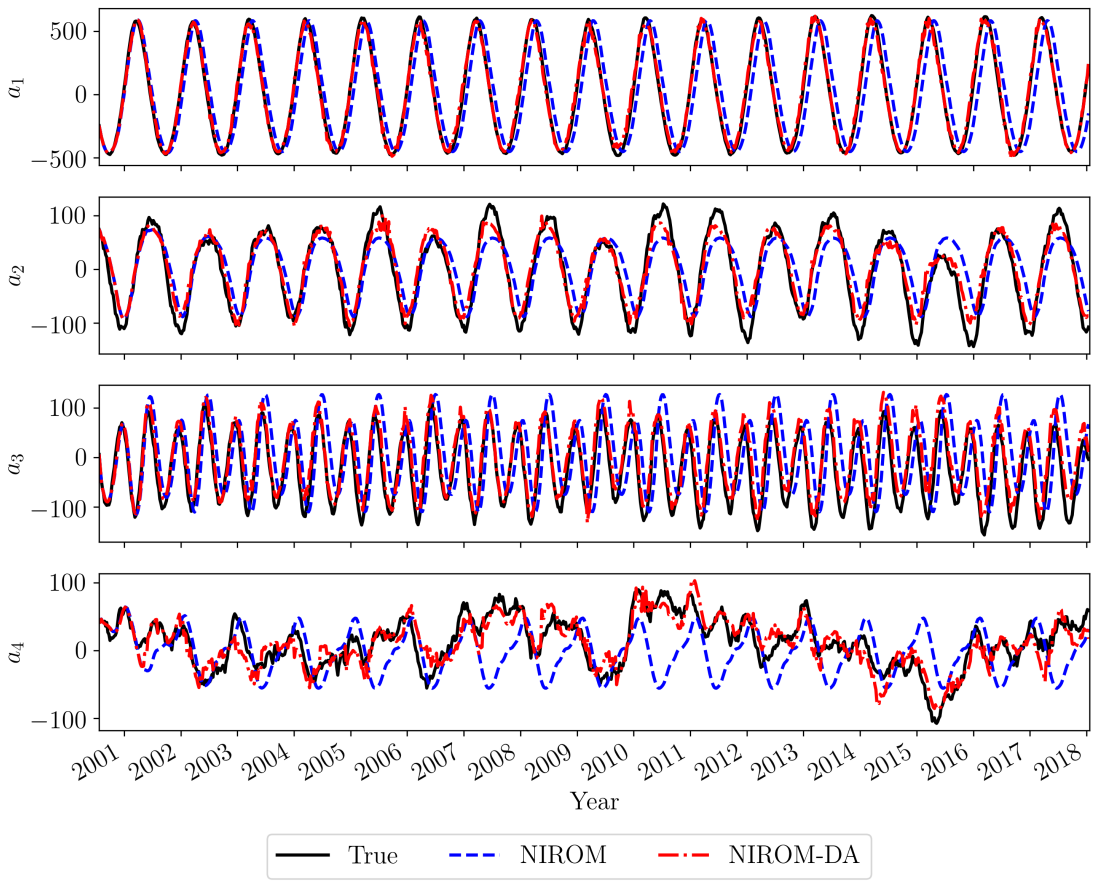

Figure 7.4: Evolution of the POD modal coefficients for the forecasting (i.e., 2001-2018 period) of the sea surface temperature data. Predictions are started with an ensemble of 40 noisy observations.

i.e., 2001-2018. The POD modal coefficients can be interpreted as the contribution of different spatial frequencies (i.e., basis functions) in the evolving flow. One can observe that the first three modal coefficients are responsible for capturing the seasonal dynamics, and hence do not exhibit strong chaotic behavior. The fourth modal coefficient is significantly more chaotic than the first three modal coefficients due to the stochastic nature of small-scale fluctuations. The prediction from the NIROM is quite accurate in the initial period and the error gradually increases with time. This is a well-known limitation of the auto-regressive deployment of the LSTM for modeling the evolution of dynamical systems [339, 229] and can be attributed to the error accumulation over time. The difference is particularly considerable for the first modal coefficient where there is a large phase difference between the true dynamics and predicted dynamics near the final time. Similar errors are also observed for the second and third modal coefficients along with a very inaccurate prediction for the

fourth modal coefficient. The NIROM-DA framework can provide an analysis state that is very close to the true modal coefficients. The phase difference between the true and analyzed modal coefficients is very small leading to accurate modeling of seasonal dynamics. The analysis state of the fourth modal coefficient is also very close to the true state, meaning that the small-scale fluctuations are captured accurately with the NIROM-DA framework.

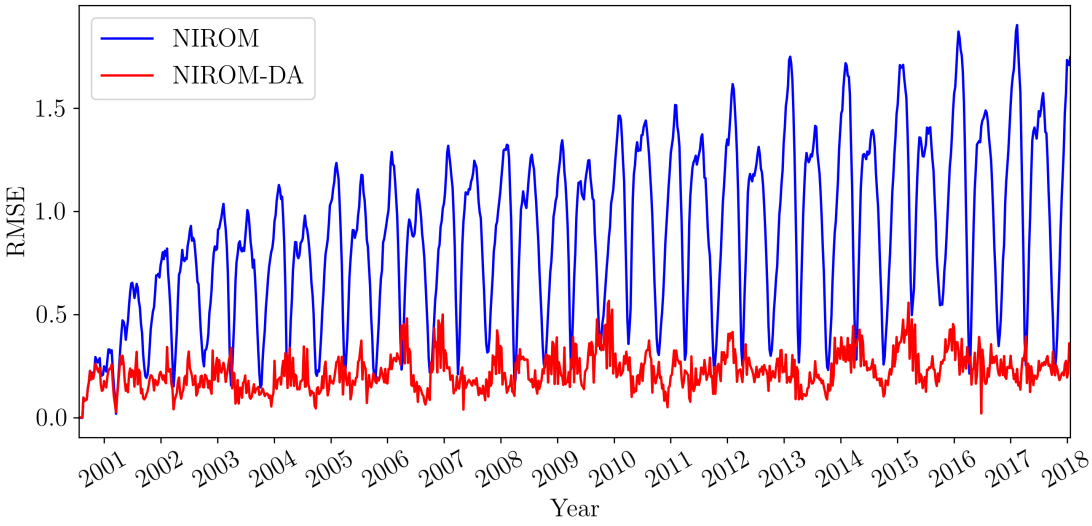

Figure 7.5: The RMSE (in degree Celsius) for the forecast (i.e., 2001-2018 period) between the projection of observed weekly average SST data onto four POD modes and the forecast obtained from NIROM, and NIROM-DA approaches.

Next, we analyze the performance of NIROM and NIROM-DA frameworks in the reconstruction of the temperature field. The temperature field is reconstructed using the predicted modal coefficients and the POD basis functions with Eq. 7.6. We note here that the performance of our surrogate model strategy is limited by the number of POD modes used and the energy captured by those POD modes. Thus, we can at the most recover the true projection (TP) of the analysis temperature field (also referred to as the FOM) onto the POD modes. One can also interpret TP as the filtered version of the analysis temperature field. Thus, for our analysis, the root mean squared error (RMSE) is computed between the TP temperature field and the reconstructed temperature field from NIROM and NIROM-DA frameworks. Fig. 7.5 displays the evolution of RMSE in degrees Celsius for the forecasting period. Overall the RMSE for NIROM-DA is one order of magnitude less than using only NIROM for the prediction. The probability density of the RMSE for NIROM and NIROM-DA

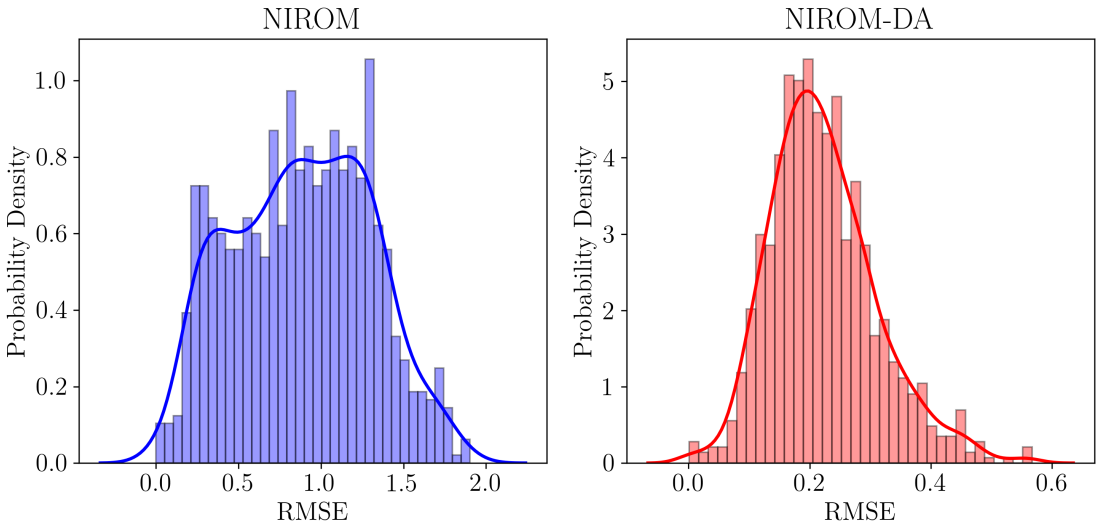

Figure 7.6: The distribution of the forecast RMSE in degree Celsius for NIROM (left) and NIROM-DA (right).

approaches is displayed in Fig. 7.6. The RMSE for NIROM lies mostly between 0.5° Celsius to 1.5° Celsius, while the RMSE for NIROM-DA is mostly centered around 0.2° Celsius. This shows that a pure data-driven surrogate model can be built by exploiting the observational data collected over many decades and it can provide a sufficient level of accuracy in forecasting. Furthermore, the prediction from the forecast model can be improved using online sensor measurement in a computationally efficient manner using latent assimilation.

The quantitative findings for the NIROM and NIROM-DA frameworks are further supported by the visualization of the temperature field at two different times. Fig. 7.7 displays the temperature field on September 14, 2009 and the temperature field on June 21, 2018 is shown in Fig. 7.8. As we are retaining only four POD modes for the surrogate model, some of the small-scale features of the exact temperature field (i.e., FOM) are not captured by TP. While these small-scale features can be captured by retaining a large number of POD modes, training a recurrent neural network to learn the dynamics of lower-energy POD modes will require additional treatment. One potential solution for this can be the non-linear proper orthogonal decomposition (NLPOD) where an autoencoder is used to compress a large number of POD modal coefficients [3]. From Fig. 7.7, it is seen that even though larger structures in the flow are captured by NIROM, the error is higher for NIROM compared to NIROM-DA, especially in the northern hemisphere region. Similar observations are also noted

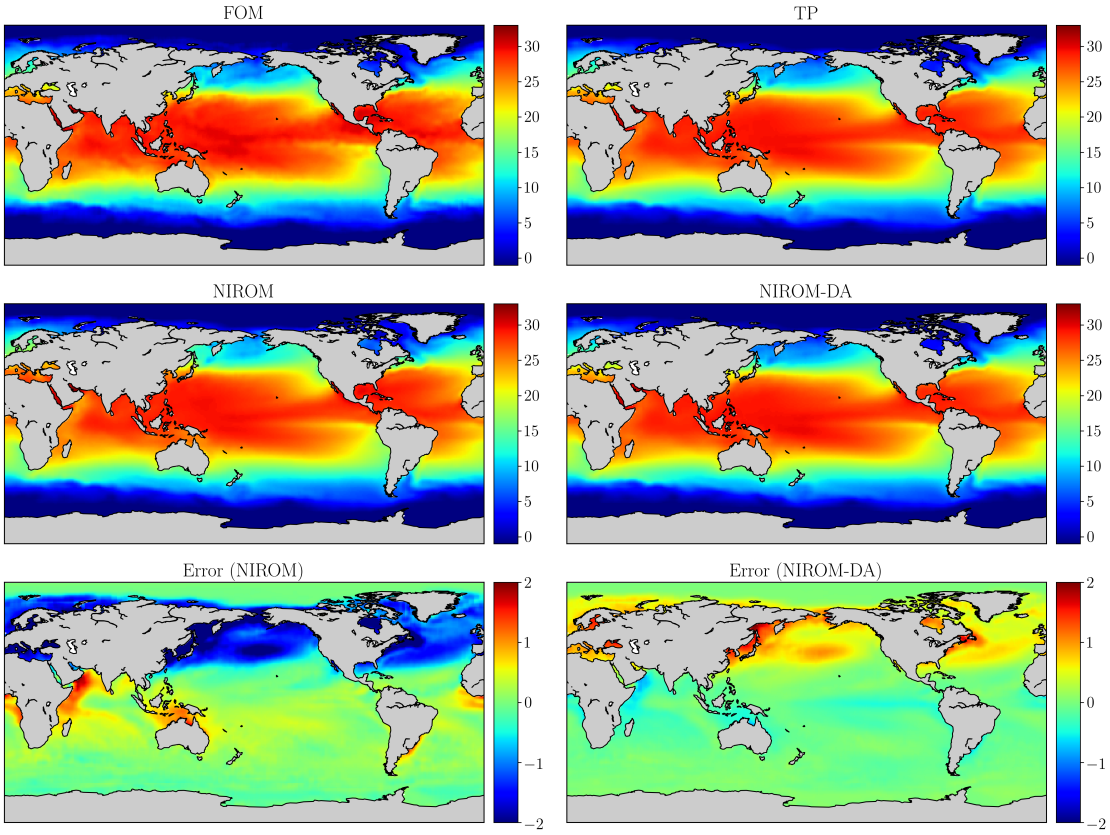

Figure 7.7: Sample averaged temperature forecasts in degrees Celsius for the week of September 14, 2009. The FOM corresponds to actual observed averaged sea surface temperature and the TP corresponds to the projection of the FOM data onto four POD modes. The error for NIROM and NIROM-DA are calculated as the difference between the TP field and the predicted field.

in Fig. 7.8, where the error for NIROM is larger than the NIROM-DA in both the northern and southern hemisphere regions.

7.4 Concluding Remarks

In this chapter, we propose a novel framework to construct a non-intrusive reduced order model (NIROM) for geophysical flow emulation and integrate this low-dimensional model within the deterministic ensemble Kalman filter (DEnKF) algorithm to assimilate sparse and noisy observations in the latent space. The NIROM is based on a proper-orthogonal decomposition to identify the dominant modes from the data and then uses a recurrent neural network to learn the dynamics of low-dimensional latent space. This surrogate modeling strategy exploits the archival of observational data without relying on any kind of governing equations. One of the critical components

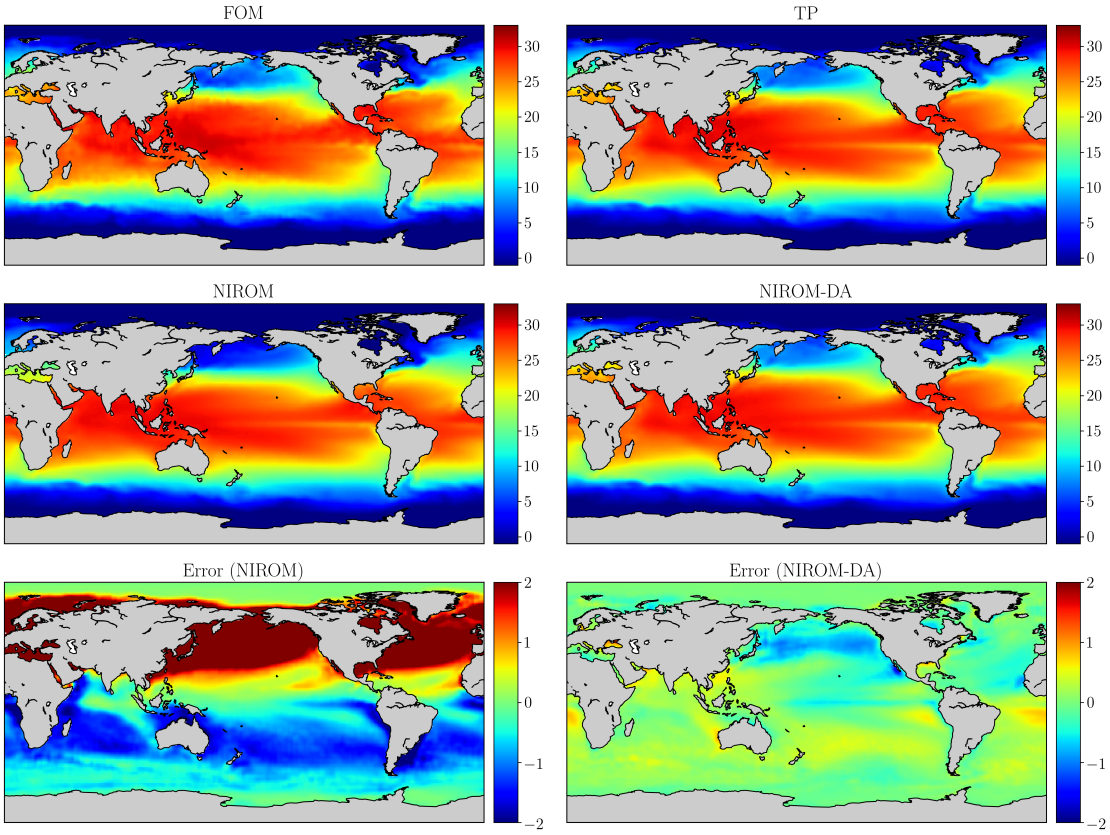

Figure 7.8: Sample averaged temperature forecasts in degrees Celsius for the week of June 21, 2018. The FOM corresponds to actual observed averaged sea surface temperature and the TP corresponds to the projection of the FOM data onto four POD modes. The error for NIROM and NIROM-DA are calculated as the difference between the TP field and the predicted field.

of the proposed framework is the reconstruction of full state form sparse and noisy discrete sensor measurement. This is achieved using the least square estimation to map the information at near-optimal sensor locations determined through QR pivoting to the full state of the system. The latent observations for assimilation are generated by projecting this full state onto the POD modes.

We demonstrate the performance of our framework for the NOAA Optimum Interpolation Sea Surface Temperature (SST) analysis dataset. The NIROM is able to achieve a stable long-range forecast along with predicting larger structures in the temperature field with a sufficient level of accuracy. Once the NIROM is integrated within the DEnKF, the prediction is improved quantitatively by almost one order of magnitude. The results show that the NIROM can be readily coupled with the DA and latent space trajectory can be corrected every time sparse and noisy observations

get available. With this framework, the computational saving is achieved through the replacement of a forward numerical solver with a data-driven model, and assimilation in latent space in contrast to the full state of the system. The proposed framework is extremely flexible and other algorithms can be easily accommodated. For example, one can use algorithms like convolutional autoencoder for dimensionality reduction, or shallow decoder for the reconstruction of full state from sparse discrete measurements. Furthermore, the NIROM can be easily differentiated with automatic differentiation and can be utilized in variational DA settings.

Finally, we point out that there is a number of future research directions that one can take. One such future direction is to improve the performance of the surrogate model by incorporating more modes to capture the low-energy content and frameworks like non-linear proper orthogonal decomposition [3] can be exploited for that. Another future research direction is to enhance the robustness of a long short-term memory network by reducing error accumulation during its auto-regressive deployment [339]. Our future work will also include exploring multi-fidelity data assimilation where a few ensembles of high-fidelity numerical solvers will be complemented with a large ensemble of data-driven models for forecasting [298] and employ a shallow decoder for full state reconstruction [94].

CHAPTER VIII

Conclusions and Future Work

This dissertation detailed different physics-guided machine learning (PGML) frameworks relevant to two key problems in modeling turbulent fluid flows, i.e., closure model discovery and model order reduction. The PGML-based closure modeling has emerged as a promising alternative for large eddy simulation (LES) of geophysical turbulent flows that do not rely on any prior assumption about turbulence physics and learns from big data using physical constraints. The PGML-based model order reduction framework along with uncertainty quantification reduces the computational cost of a partial-differential equation solver and can be integrated with many online tasks like data assimilation, optimization, and optimal control. Figure 8.1 summarizes different components of physics-guided machine learning approaches proposed in this dissertation and applied to different scientific benchmark problems.

8.1 Summary of Study

In **Chapter II**, we investigated different data-driven subgrid scale (SGS) closure models for two-dimensional Kraichnan turbulence in the *a priori* settings. We illustrate the numerically stable deployment of CNN-based SGS closure models in *a posteriori* simulations and recover average turbulence statistics similar to the dynamic Smagorinsky model. The data-driven SGS model is computationally much faster compared to the dynamic Smagorinsky model and can be further augmented with the known physics.

In **Chapter III**, we presented a data assimilation (DA) framework to integrate sparse and noisy observations into a hybrid model where the dynamical core of the system is modeled using known governing equations and the subgrid processes are considered with a deep learning model. This framework is demonstrated for the two-level Lorenz 96 system and Kraichnan turbulence. This framework leads to significant improvement in the long-term prediction for both systems against using only the truncated model with DA and just using a hybrid model without DA.

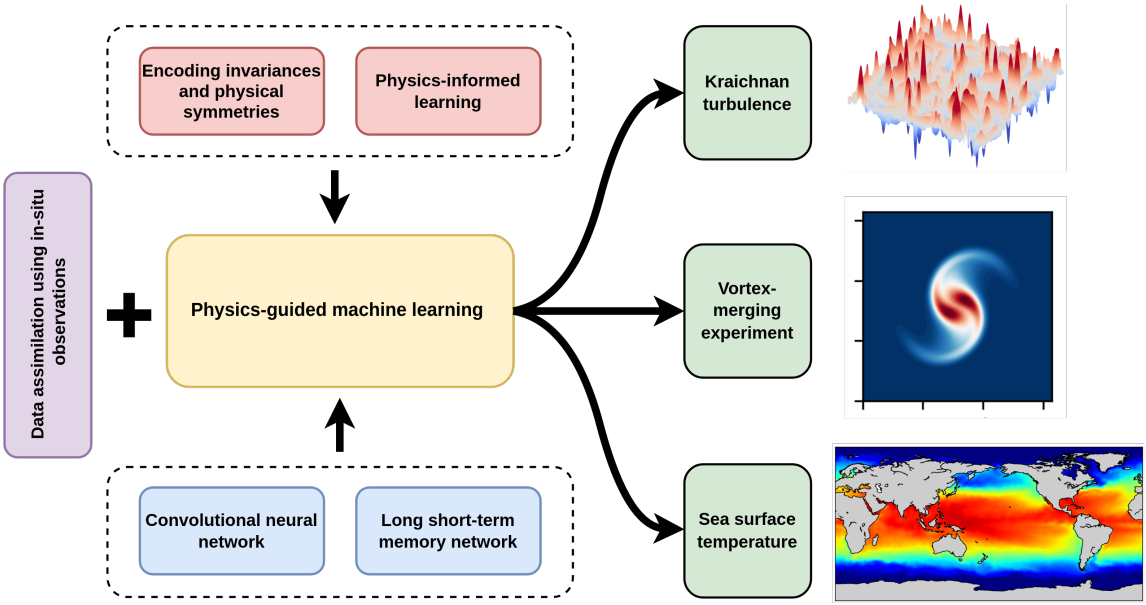

Figure 8.1: Summary of physics-guided machine learning models applied to different scientific applications.

In **Chapter IV**, we propose a novel neural network-based SGS model for LES of turbulent fluid flows. This model is based on customized convolutional neural network architecture that incorporates translation, Galilean, and rotation invariances as hard constraints and guarantees them up to the discretization error. This framework is demonstrated for two-dimensional decaying turbulence test case and we show that our frame invariant SGS model (i) accurately predicts the subgrid scale forcing, (ii) respects the physical symmetries such as translation, Galilean, and rotation invariance, (iii) is numerical stable when implemented in coarse-grid simulation, and (iv) generalizes to different initial conditions and Reynolds number. Our numerical results show that the frame invariant SGS model provides the best agreement for several statistical metrics with the filtered DNS solution.

In **Chapter V**, a concatenated neural network architecture is proposed as a model fusion framework to exploit the data coming from models with different levels of fidelity. Additionally, the framework predicts the uncertainty associated with the prediction, and thus safeguards against the inaccurate prediction in the extrapolation region. This framework is applied to laminar and turbulent boundary layer flow over a flat plate, and improved results are obtained with very sparse data. In **Chapter VI**, the concatenated neural network framework with uncertainty quantification is extended to projection-based reduced-order models and the framework is illustrated for the

vortex-merging experiment. It is shown that the framework can provide accurate prediction even in the extrapolation region, while the pure data-driven model fails.

Chapter VII presents an end-to-end framework to integrating non-intrusive reduced-order model with sequential data assimilation. Particularly, measurements at optimal sensor locations are used for obtaining the full state using the linear reconstruction method, and then the full state is mapped to latent observational space for assimilation. The use of a surrogate model instead of a full order model and assimilation in latent space makes this framework computationally efficient. The complete framework is demonstrated for the forecasting of weekly average sea surface temperature by using the previously stored satellite observations to construct a surrogate model.

8.2 Future Work

The studies presented in this dissertation opens some interesting questions and provides many opportunities for future research.

- The turbulence closure model approaches proposed in this study are flow specific, meaning that the closure model developed using the data generated for one type of flow may not generalize well to a different type of flow. This is a fundamental challenge with any data-driven method. Transfer learning and meta-learning are two techniques that can be explored to tackle this issue. Transfer learning is a method that focuses on storing knowledge gained while solving one problem and applying it to solve a different but related problem. Using transfer learning, the accuracy of a neural network trained using data generated for one Reynolds number can be improved by using very few snapshots of training data for different Reynolds numbers [363, 128, 129]. There are many open questions while applying transfer learning to turbulence closure discovery problems, such as, what are the optimal layers to retrain; what exactly is learned during transfer learning; and how much variation should be there in the data. Another promising approach is meta-learning which is a training framework for generalizing deep learning models to different problems [380]. The main idea of meta-learning is to extract prior information from a set of tasks that allow for efficient learning on a new task. Within the context of meta-learning, one can train the model using the data for different canonical turbulence datasets, and then during the adaptation phase, we can focus on the quick acquisition of knowledge to learn task-specific

parameters.

- The uncertainty quantification (UQ) will continue to become more important for surrogate modeling before it can be used for critical applications and decision-making [301]. There are many sources of uncertainties with PGML modeling, such as noisy and limited data, selection of hyperparameters or overparametrization of ML models, and incorrect specification of physical constraints. The systematic modeling of different sources of uncertainties using advanced deep learning methods like Bayesian neural network and generative adversarial network is a fruitful avenue for future research. The UQ information from these deep learning models can be exploited with active learning, i.e., re-adjusting the mean and uncertainty prediction in the light of new data. This can play a key role in dealing with a lack of labeled data which is particularly important for rare and extreme events.

- The interpretability is another major challenge with data-driven methods. While incorporating physics is closely coupled to model interpretability, it is important to interpret how the model has optimized the data. The model introspection challenge is even more pronounced for nonlinear methods like deep learning which usually has thousands of parameters and sufficient nonlinearity. The techniques such as feature importance, feature visualization [270, 165], and neural network dissection [25] can be investigated in future studies to decompose the decision process of PGML models into human-interpretable steps. The interpretable steps may include connecting abstract representations to known physical laws or prior knowledge about the problem.

- The PGML approaches proposed in this study are validated for two-dimensional benchmark problems with a structured grid. Most of the atmospheric and oceanic flows are well approximated with two-dimensional turbulence. However, for some applications, three-dimensional modeling is necessary to capture the behavior of rotating, stratified fluids. Furthermore, an unstructured mesh is usually preferred for complex geometries, especially in engineering applications. Graph neural network is a potential choice for simulation of fluid mechanics system that requires unstructured domain and meshes [338] and can be investigated in future studies for both closure model discovery and model order reduction problems. When it comes to applying ML for high-dimensional systems, the key

computational challenges are the large data involved and the associated training time and scalability. Both model- and data-parallelization can be used in future studies to assess the scalability of training ML models for scientific problems. Another interesting future research direction can be to map ML parallelization techniques which are mostly developed for cloud and distributed systems to high-performance computing machines. The integration of python-based ML libraries with legacy CFD simulation codes which are mostly written in Fortran or C/C++ has unique challenges and opportunities that can be explored in future studies.

References

[1] Shady E Ahmed, Suraj Pawar, and Omer San. PyDA: A Hands-On Introduction to Dynamical Data Assimilation with Python. *Fluids*, 5(4):225, 2020.

[2] Shady E Ahmed, Suraj Pawar, Omer San, Adil Rasheed, Traian Iliescu, and Bernd R Noack. On closures for reduced order models—a spectrum of first-principle to machine-learned avenues. *Physics of Fluids*, 33(9):091301, 2021.

[3] Shady E Ahmed, Omer San, Adil Rasheed, and Traian Iliescu. Nonlinear proper orthogonal decomposition for convection-dominated flows. *Physics of Fluids*, 33(12):121702, 2021.

[4] Maddalena Amendola, Rossella Arcucci, Laetitia Mottet, Cesar Quilodran Casas, Shiwei Fan, Christopher Pain, Paul Linden, and Yi-Ke Guo. Data assimilation in the latent space of a neural network. *arXiv preprint arXiv:2012.12056*, 2020.

[5] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.

[6] David Amsallem and Charbel Farhat. Interpolation method for adapting reduced-order models and application to aeroelasticity. *AIAA Journal*, 46(7):1803–1813, 2008.

[7] Jeffrey L Anderson. An adaptive covariance inflation error correction algorithm for ensemble filters. *Tellus A: Dynamic meteorology and oceanography*, 59(2):210–224, 2007.

[8] Jeffrey L Anderson. A non-Gaussian ensemble filter update for data assimilation. *Monthly Weather Review*, 138(11):4186–4198, 2010.

[9] Jeffrey L Anderson and Stephen L Anderson. A Monte Carlo implementation of the nonlinear filtering problem to produce ensemble assimilations and forecasts. *Monthly Weather Review*, 127(12):2741–2758, 1999.

[10] 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.

[11] Akio Arakawa. Computational design for long-term numerical integration of the equations of fluid motion: Two-dimensional incompressible flow. part i. *Journal of computational physics*, 1(1):119–143, 1966.

[12] Akio Arakawa. Computational design for long-term numerical integration of the equations of fluid motion: Two-dimensional incompressible flow. Part I. *Journal of Computational Physics*, 135(2):103–114, 1997.

[13] Cristina L Archer, Ahmadreza Vasel-Be-Hagh, Chi Yan, Sicheng Wu, Yang Pan, Joseph F Brodie, and A Eoghan Maguire. Review and evaluation of wake loss models for wind energy applications. *Applied Energy*, 226:1187–1207, 2018.

[14] Troy Arcomano, Istvan Szunyogh, Jaideep Pathak, Alexander Wikner, Brian R. Hunt, and Edward Ott. A machine learning-based global atmospheric forecast model. *Geophysical Research Letters*, 47(9), 2020.

[15] Rossella Arcucci. Effective data assimilation with machine learning. In *Data Science and Big Data Analytics in Smart Environments*, pages 27–46. CRC Press, 2021.

[16] Rossella Arcucci, César Quilodrán Casas, Dunhui Xiao, Laetitia Mottet, Fangxin Fang, Pin Wu, and Christopher Pain. A Domain Decomposition Reduced Order Model with Data Assimilation (DD-RODA). IOS Press BV, 2020.

[17] 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.

[18] Ahmed Attia and Emil Constantinescu. An optimal experimental design framework for adaptive inflation and covariance localization for ensemble filters. *arXiv preprint arXiv:1806.10655*, 2018.

[19] Ahmed Attia and Adrian Sandu. Dates: a highly extensible data assimilation testing suite v1. 0. *Geoscientific Model Development*, 12(2):629–649, 2019.

[20] Shan Ba and V Roshan Joseph. Composite gaussian process models for emulating expensive functions. *The Annals of Applied Statistics*, pages 1838–1860, 2012.

[21] Yohai Bar-Sinai, Stephan Hoyer, Jason Hickey, and Michael P Brenner. Learning data-driven discretizations for partial differential equations. *Proceedings of the National Academy of Sciences*, 116(31):15344–15349, 2019.

[22] Jorge Bardina, J Ferziger, and WC Reynolds. Improved subgrid-scale models for large-eddy simulation. In *13th fluid and plasmadynamics conference*, page 1357, 1980.

[23] Brian R Bartoldson, Ari S Morcos, Adrian Barbu, and Gordon Erlebacher. The generalization-stability tradeoff in neural network pruning. *arXiv preprint arXiv:1906.03728*, 2019.

[24] George K Batchelor. Computation of the energy spectrum in homogeneous two-dimensional turbulence. *The Physics of Fluids*, 12(12):II–233, 1969.

[25] David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6541–6549, 2017.

[26] Peter Bauer, Bjorn Stevens, and Wilco Hazeleger. A digital twin of earth for the green transition. *Nature Climate Change*, 11(2):80–83, 2021.

[27] Peter Bauer, Alan Thorpe, and Gilbert Brunet. The quiet revolution of numerical weather prediction. *Nature*, 525:47–55, 09 2015.

[28] Andrea Beck, David Flad, and Claus-Dieter Munz. Deep neural networks for data-driven les closure models. *Journal of Computational Physics*, 398:108910, 2019.

[29] Andrea Beck and Marius Kurz. A perspective on machine learning methods in turbulence modeling. *GAMM-Mitteilungen*, 44(1):e202100002, 2021.

[30] Erik J Bekkers, Maxime W Lafarge, Mitko Veta, Koen AJ Eppenhof, Josien PW Pluim, and Remco Duits. Roto-translation covariant convolutional networks for medical image analysis. In *International conference on medical image computing and computer-assisted intervention*, pages 440–448. Springer, 2018.

[31] Peter Benner, Serkan Gugercin, and Karen Willcox. A survey of projection-based model reduction methods for parametric dynamical systems. *SIAM Review*, 57(4):483–531, 2015.

[32] Gal Berkooz, Philip Holmes, and John L Lumley. The proper orthogonal decomposition in the analysis of turbulent flows. *Annual Review of Fluid Mechanics*, 25(1):539–575, 1993.

[33] Luigi C. Berselli, Traian Iliescu, and William J. Layton. *Mathematics of Large Eddy Simulation of Turbulent Flows*. Scientific Computation, Springer-Verlag, Berlin, 2006.

[34] Tom Beucler, Michael Pritchard, Stephan Rasp, Jordan Ott, Pierre Baldi, and Pierre Gentine. Enforcing analytic constraints in neural networks emulating physical systems. *Physical Review Letters*, 126(9):098302, 2021.

[35] Tom Beucler, Stephan Rasp, Michael Pritchard, and Pierre Gentine. Achieving conservation of energy in neural network emulators for climate modeling. *arXiv preprint arXiv:1906.06622*, 2019.

[36] Kiran Bhaganagar, John Kim, and Gary Coleman. Effect of roughness on wall-bounded turbulence. *Flow, turbulence and combustion*, 72(2):463–492, 2004.

[37] Saakaar Bhatnagar, Yaser Afshar, Shaowu Pan, Karthik Duraisamy, and Shailendra Kaushik. Prediction of aerodynamic flow fields using convolutional neural networks. *Computational Mechanics*, 64:525–545, 2019.

[38] 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.

[39] Guido Boffetta and Robert E Ecke. Two-dimensional turbulence. *Annual Review of Fluid Mechanics*, 44:427–451, 2012.

[40] Thomas Bolton and Laure Zanna. Applications of deep learning to ocean data inference and subgrid parameterization. *Journal of Advances in Modeling Earth Systems*, 11(1):376–399, 2019.

[41] Massimo Bonavita and Patrick Laloyaux. Machine learning for model error inference and correction. *Journal of Advances in Modeling Earth Systems*, 12(12):e2020MS002232, 2020.

[42] Freddy Bouchet and Antoine Venaille. Statistical mechanics of two-dimensional and geophysical flows. *Physics Reports*, 515(5):227–295, 2012.

[43] Sid-Ahmed Boukabara, Vladimir Krasnopolsky, Stephen G Penny, Jebb Q Stewart, Amy McGovern, David Hall, John E Ten Hoeve, Jason Hickey, Hung-Lung Allen Huang, John K Williams, et al. Outlook for exploiting artificial intelligence in the earth and environmental sciences. *Bulletin of the American Meteorological Society*, 102(5):E1016–E1032, 2021.

[44] 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.

[45] Julien Brajard, Alberto Carrassi, Marc Bocquet, and Laurent Bertino. Combining data assimilation and machine learning to infer unresolved scale parametrization. *Philosophical Transactions of the Royal Society A*, 379(2194):20200086, 2021.

[46] MP Brenner, JD Eldredge, and JB Freund. Perspective on machine learning for advancing fluid mechanics. *Physical Review Fluids*, 4(10):100501, 2019.

[47] Noah D Brenowitz and Christopher S Bretherton. Prognostic validation of a neural network unified physics parameterization. *Geophysical Research Letters*, 45(12):6289–6298, 2018.

[48] Noah D Brenowitz and Christopher S Bretherton. Spatially extended tests of a neural network parametrization trained by coarse-graining. *Journal of Advances in Modeling Earth Systems*, 11(8):2728–2744, 2019.

[49] S-P Breton, J Sumner, Jens Nørkær Sørensen, Kurt Schaldemose Hansen, Sasan Sarmast, and Stefan Ivanell. A survey of modelling methods for high-fidelity wind farm simulations using large eddy simulation. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 375(2091):20160097, 2017.

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

[51] Steven L Brunton, Bernd R Noack, and Petros Koumoutsakos. Machine learning for fluid mechanics. *Annual Review of Fluid Mechanics*, 52:477–508, 2020.

[52] Caterina Buizza, César Quilodrán Casas, Philip Nadler, Julian Mack, Stefano Marrone, Zainab Titus, Clémence Le Cornec, Evelyn Heylen, Tolga Dur, Luis Baca Ruiz, et al. Data learning: Integrating data assimilation and machine learning. *Journal of Computational Science*, 58:101525, 2022.

[53] Caterina Buizza, César Quilodrán Casas, Philip Nadler, Julian Mack, Stefano Marrone, Zainab Titus, Clémence Le Cornec, Evelyn Heylen, Tolga Dur, Luis Baca Ruiz, Claire Heaney, Julio Amador Díaz Lopez, K.S. Sesh Kumar, and Rossella Arcucci. Data learning: Integrating data assimilation and machine learning. *Journal of Computational Science*, 58:101525, 2022.

[54] Gerrit Burgers, Peter Jan van Leeuwen, and Geir Evensen. Analysis scheme in the ensemble Kalman filter. *Monthly Weather Review*, 126(6):1719–1724, 1998.

[55] Jared L Callaham, Kazuki Maeda, and Steven L Brunton. Robust flow reconstruction from limited measurements via sparse representation. *Physical Review Fluids*, 4(10):103907, 2019.

[56] Yanhua Cao, Jiang Zhu, I Michael Navon, and Zhendong Luo. A reduced-order approach to four-dimensional variational data assimilation using proper orthogonal decomposition. *International Journal for Numerical Methods in Fluids*, 53(10):1571–1583, 2007.

[57] Kevin Carlberg, Charbel Bou-Mosleh, and Charbel Farhat. Efficient non-linear model reduction via a least-squares petrov–galerkin projection and compressive tensor approximations. *International Journal for numerical methods in engineering*, 86(2):155–181, 2011.

[58] 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.

[59] Yunus A.. Çengel and John M.. Cimbala. *Fluid Mechanics: Fundamentals and Applications*. McGraw-Hill, New York, 2010.

[60] Matthew Chantry, Hannah Christensen, Peter Dueben, and Tim Palmer. Opportunities and challenges for machine learning in weather and climate modelling: hard, medium and soft ai, 2021.

[61] Ashesh Chattopadhyay, Pedram Hassanzadeh, and Saba Pasha. Predicting clustered weather patterns: A test case for applications of convolutional neural networks to spatio-temporal climate data. *Scientific Reports*, 10(1):1–13, 2020.

[62] Ashesh Chattopadhyay, Mustafa Mustafa, Pedram Hassanzadeh, Eviatar Bach, and Karthik Kashinath. Towards physically consistent data-driven weather forecasting: Integrating data assimilation with equivariance-preserving spatial transformers in a case study with era5. *Geoscientific Model Development Discussions*, pages 1–23, 2021.

[63] Ashesh Chattopadhyay, Mustafa Mustafa, Pedram Hassanzadeh, Eviatar Bach, and Karthik Kashinath. Towards physics-inspired data-driven weather forecasting: integrating data assimilation with a deep spatial-transformer-based u-net in a case study with era5. *Geoscientific Model Development*, 15(5):2221–2237, 2022.

[64] Wenqian Chen, Qian Wang, Jan S Hesthaven, and Chuhua Zhang. Physics-informed machine learning for reduced-order modeling of nonlinear problems. *Journal of Computational Physics*, 446:110666, 2021.

[65] M Cheng, F Fang, CC Pain, and I M Navon. An advanced hybrid deep adversarial autoencoder for parameterized nonlinear fluid flow modelling. *Computer Methods in Applied Mechanics and Engineering*, 372:113375, 2020.

[66] M Cheng, Fangxin Fang, Christopher 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.

[67] Meiling Cheng, Fangxin Fang, I M Navon, and CC Pain. A real-time flow forecasting with deep convolutional generative adversarial network: Application to flooding event in Denmark. *Physics of Fluids*, 33(5):056602, 2021.

[68] Sibo Cheng, Jianhua Chen, Charitos Anastasiou, Panagiota Angeli, Omar K Matar, Yi-Ke Guo, Christopher C Pain, and Rossella Arcucci. Generalised latent

assimilation in heterogeneous reduced spaces with machine learning surrogate models. *arXiv preprint arXiv:2204.03497*, 2022.

[69] Austin Chennault, Andrey A Popov, Amit N Subrahmanya, Rachel Cooper, Anuj Karpatne, and Adrian Sandu. Adjoint-matching neural network surrogates for fast 4d-var data assimilation. *arXiv preprint arXiv:2111.08626*, 2021.

[70] Taco Cohen and Max Welling. Group equivariant convolutional networks. In *International conference on machine learning*, pages 2990–2999. PMLR, 2016.

[71] Taco S Cohen and Max Welling. Steerable cnns. *arXiv preprint arXiv:1612.08498*, 2016.

[72] CH Colburn, JB Cessna, and TR Bewley. State estimation in wall-bounded flow systems. Part 3. The ensemble Kalman filter. *Journal of Fluid Mechanics*, 682:289, 2011.

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

[74] Andre FC da Silva and Tim Colonius. Ensemble-based state estimator for aerodynamic flows. *AIAA Journal*, 56(7):2568–2578, 2018.

[75] Casey B Davis, Christopher M Hans, and Thomas J Santner. Prediction of non-stationary response functions using a bayesian composite gaussian process. *Computational Statistics & Data Analysis*, 154:107083, 2021.

[76] Anthony Christopher Davison. *Statistical models*, volume 11. Cambridge University Press, Cambridge, 2003.

[77] 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.

[78] James W Deardorff. A numerical study of three-dimensional turbulent channel flow at large Reynolds numbers. *Journal of Fluid Mechanics*, 41(2):453–480, 1970.

[79] X Deng, C Devon Lin, K-W Liu, and RK Rowe. Additive gaussian process for computer models with qualitative and quantitative factors. *Technometrics*, 59(3):283–292, 2017.

[80] Thomas G Dietterich. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15. Springer, Berlin, 2000.

[81] Chao Dong, Chen Change Loy, and Xiaoou Tang. Accelerating the super-resolution convolutional neural network. In *European conference on computer vision*, pages 391–407. Springer, 2016.

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

[83] Zlatko Drmac and Serkan Gugercin. A new selection operator for the discrete empirical interpolation method—improved a priori error bound and extensions. *SIAM Journal on Scientific Computing*, 38(2):A631–A648, 2016.

[84] 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.

[85] Peter D Dueben and Peter Bauer. Challenges and design choices for global weather and climate models based on machine learning. *Geoscientific Model Development*, 11(10):3999–4009, 2018.

[86] Karthik Duraisamy. Perspectives on machine learning-augmented reynolds-averaged and large eddy simulation models of turbulence. *Physical Review Fluids*, 6(5):050504, 2021.

[87] Karthik Duraisamy, Gianluca Iaccarino, and Heng Xiao. Turbulence modeling in the age of data. *Annual Review of Fluid Mechanics*, 51:357–377, 2019.

[88] Paul A Durbin. Near-wall turbulence closure modeling without "damping functions". *Theoretical and Computational Fluid Dynamics*, 3(1):1–13, 1991.

[89] Imme Ebert-Uphoff and Kyle Hilburn. Evaluation, tuning, and interpretation of neural networks for working with images in meteorological applications. *Bulletin of the American Meteorological Society*, 101(12):E2149–E2170, 2020.

[90] Carsten Eden and Richard J Greatbatch. Towards a mesoscale eddy closure. *Ocean Modelling*, 20(3):223–239, 2008.

[91] Hamidreza Eivazi, Hadi Veisi, Mohammad Hossein Naderi, and Vahid Esfahanian. Deep neural networks for nonlinear model order reduction of unsteady flows. *Physics of Fluids*, 32(10):105104, 2020.

[92] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *The Journal of Machine Learning Research*, 20(1):1997–2017, 2019.

[93] N Benjamin Erichson, Lionel Mathelin, Zhewei Yao, Steven L Brunton, Michael W Mahoney, and J Nathan Kutz. Shallow learning for fluid flow reconstruction with limited sensors and limited data. *arXiv preprint arXiv:1902.07358*, 2019.

[94] N Benjamin Erichson, Lionel Mathelin, Zhewei Yao, Steven L Brunton, Michael W Mahoney, and J Nathan Kutz. Shallow neural networks for fluid flow reconstruction with limited sensors. *Proceedings of the Royal Society A*, 476(2238):20200097, 2020.

[95] N Benjamin Erichson, Michael Muehlebach, and Michael W Mahoney. Physics-informed autoencoders for Lyapunov-stable fluid flow prediction. *arXiv preprint arXiv:1905.10866*, 2019.

[96] Gordon Erlebacher, M Yousuff Hussaini, Charles G Speziale, and Thomas A Zang. Toward the large-eddy simulation of compressible turbulent flows. *Journal of Fluid Mechanics*, 238:155–185, 1992.

[97] Geir Evensen. *Data assimilation: the ensemble Kalman filter*. Springer Science & Business Media, 2009.

[98] James H Faghmous, Arindam Banerjee, Shashi Shekhar, Michael Steinbach, Vipin Kumar, Auroop R Ganguly, and Nagiza Samatova. Theory-guided data science for climate change. *Computer*, 47(11):74–78, 2014.

[99] Artur J Ferreira and Mário AT Figueiredo. Boosting algorithms: A review of methods, theory, and applications. In *Ensemble machine learning*, pages 35–85. Springer, Berlin, 2012.

[100] Matthias Fey, Jan Eric Lenssen, Frank Weichert, and Heinrich Müller. SplineCNN: Fast geometric deep learning with continuous B-spline kernels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 869–877, 2018.

[101] AM Foley. Uncertainty in regional climate modelling: A review. *Progress in Physical Geography*, 34(5):647–670, 2010.

[102] Jorgen S Frederiksen and Steven M Kepert. Dynamical subgrid-scale parameterizations from direct numerical simulations. *Journal of the Atmospheric Sciences*, 63(11):3006–3019, 2006.

[103] Jorgen S Frederiksen, Terence J O'Kane, and Meelis J Zidikheri. Subgrid modelling for geophysical flows. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 371(1982):20120166, 2013.

[104] Stefania Fresca, Luca Dede, and Andrea Manzoni. A comprehensive deep learning-based approach to reduced order modeling of nonlinear time-dependent parametrized pdes. *Journal of Scientific Computing*, 87(2):1–36, 2021.

[105] Hugo Frezat, Guillaume Balarac, Julien Le Sommer, Ronan Fablet, and Redouane Lguensat. Physical invariance in neural networks for subgrid-scale scalar flux modeling. *Physical Review Fluids*, 6(2):024607, 2021.

[106] Hugo Frezat, Julien Le Sommer, Ronan Fablet, Guillaume Balarac, and Redouane Lguensat. A posteriori learning for quasi-geostrophic turbulence parametrization. *arXiv preprint arXiv:2204.03911*, 2022.

[107] Uriel Frisch and Andreĭ Nikolaevich Kolmogorov. *Turbulence: the legacy of AN Kolmogorov*. Cambridge university press, 1995.

[108] Kai Fukami, Koji Fukagata, and Kunihiko Taira. Super-resolution reconstruction of turbulent flows with machine learning. *Journal of Fluid Mechanics*, 870:106–120, 2019.

[109] Kai Fukami, Koji Fukagata, and Kunihiko Taira. Machine-learning-based spatio-temporal super resolution reconstruction of turbulent flows. *Journal of Fluid Mechanics*, 909:A9, 2021.

[110] David John Gagne, Hannah M Christensen, Aneesh C Subramanian, and Adam H Monahan. Machine learning for stochastic parameterization: Generative adversarial networks in the lorenz'96 model. *Journal of Advances in Modeling Earth Systems*, 12(3):e2019MS001896, 2020.

[111] Boris Galperin and Steven A Orszag. *Large eddy simulation of complex engineering and geophysical flows*. Cambridge University Press, 1993.

[112] Masataka Gamahara and Yuji Hattori. Searching for turbulence models by artificial neural network. *Physical Review Fluids*, 2(5):054604, 2017.

[113] Carlos E Garcia, David M Prett, and Manfred Morari. Model predictive control: Theory and practice—a survey. *Automatica*, 25(3):335–348, 1989.

[114] AJ Geer. Learning earth system models from observations: machine learning or data assimilation? *Philosophical Transactions of the Royal Society A*, 379(2194):20200089, 2021.

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

[116] Nicholas Geneva and Nicholas Zabaras. Modeling the dynamics of pde systems with physics-constrained deep auto-regressive networks. *Journal of Computational Physics*, 403:109056, 2020.

[117] 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.

[118] Massimo Germano, Ugo Piomelli, Parviz Moin, and William H Cabot. A dynamic subgrid-scale eddy viscosity model. *Physics of Fluids A: Fluid Dynamics*, 3(7):1760–1765, 1991.

[119] Andrew Gettelman, Alan J Geer, Richard M Forbes, Greg R Carmichael, Graham Feingold, Derek J Posselt, Graeme L Stephens, Susan C van den Heever, Adam C Varble, and Paquita Zuidema. The future of earth system prediction: Advances in model-data fusion. *Science Advances*, 8(14):eabn3488, 2022.

[120] Sandip Ghosal, Thomas S Lund, Parviz Moin, and Knut Akselvoll. A dynamic localization model for large-eddy simulation of turbulent flows. *Journal of Fluid Mechanics*, 286:229–255, 1995.

[121] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

[122] Francisco J Gonzalez and Maciej Balajewicz. Deep convolutional recurrent autoencoders for learning low-dimensional feature dynamics of fluid systems. *arXiv preprint arXiv:1808.01346*, 2018.

[123] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.

[124] Sigal Gottlieb and Chi-Wang Shu. Total variation diminishing Runge-Kutta schemes. *Mathematics of Computation*, 67(221):73–85, 1998.

[125] Robert B Gramacy and Herbert KH Lee. Cases for the nugget in modeling computer experiments. *Statistics and Computing*, 22(3):713–722, 2012.

[126] Sam Greydanus, Misko Dzamba, and Jason Yosinski. Hamiltonian neural networks. *arXiv preprint arXiv:1906.01563*, 2019.

[127] Siegfried Grossmann and Peter Mertens. Structure functions in two-dimensional turbulence. *Zeitschrift für Physik B Condensed Matter*, 88(1):105–116, 1992.

[128] Yifei Guan, Ashesh Chattopadhyay, Adam Subel, and Pedram Hassanzadeh. Stable a posteriori les of 2d turbulence using convolutional neural networks: Backscattering analysis and generalization to higher re via transfer learning. *arXiv preprint arXiv:2102.11400*, 2021.

[129] Yifei Guan, Adam Subel, Ashesh Chattopadhyay, and Pedram Hassanzadeh. Learning physics-constrained subgrid-scale closures in the small-data regime for stable and accurate LES. *arXiv preprint arXiv:2201.07347*, 2022.

[130] Mengwu Guo and Jan S Hesthaven. Data-driven reduced order modeling for time-dependent problems. *Computer methods in applied mechanics and engineering*, 345:75–99, 2019.

[131] Sofia Guzzetti, LA Mansilla Alvarez, PJ Blanco, Kevin Thomas Carlberg, and A Veneziani. Propagating uncertainties in large-scale hemodynamics models

via network uncertainty quantification and reduced-order modeling. *Computer Methods in Applied Mechanics and Engineering*, 358:112626, 2020.

[132] Martin T Hagan, Howard B Demuth, Mark H Beale, and Orlando De Jesús. *Neural network design*, volume 20. Pws Pub. Boston, 1996.

[133] Kazuto Hasegawa, Kai Fukami, Takaaki Murata, and Koji Fukagata. Cnn-lstm based reduced order modeling of two-dimensional unsteady flows around a circular cylinder at different reynolds numbers. *Fluid Dynamics Research*, 52(6):065501, 2020.

[134] Sam Hatfield, Matthew Chantry, Peter Dueben, Philippe Lopez, Alan Geer, and Tim Palmer. Building tangent-linear and adjoint models for data assimilation with neural networks. *Journal of Advances in Modeling Earth Systems*, 13(9):e2021MS002521, 2021.

[135] Xin He, Kaiyong Zhao, and Xiaowen Chu. Automl: A survey of the state-of-the-art. *Knowledge-Based Systems*, 212:106622, 2021.

[136] RA Heinonen and PH Diamond. Turbulence model reduction by deep learning. *Physical Review E*, 101(6):061201, 2020.

[137] 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.

[138] Tom Heskes. Practical confidence and prediction intervals. In *Advances in neural information processing systems*, pages 176–182, 1997.

[139] John L Hess. Panel methods in computational fluid dynamics. *Annual Review of Fluid Mechanics*, 22(1):255–274, 1990.

[140] Helene T Hewitt, Malcolm Roberts, Pierre Mathiot, Arne Biastoch, Ed Blockley, Eric P Chassignet, Baylor Fox-Kemper, Pat Hyder, David P Marshall, Ekaterina Popova, et al. Resolving and parameterising the ocean mesoscale in earth system models. *Current Climate Change Reports*, 6:137–152, 2020.

[141] Tony Hey, Keith Butler, Sam Jackson, and Jeyarajan Thiyagalingam. Machine learning and big scientific data. *Philosophical Transactions of the Royal Society A*, 378(2166):20190054, 2020.

[142] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[143] 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.

[144] Philip Holmes, John L Lumley, and Gal Berkooz. *Turbulence, coherent structures, dynamical systems and symmetry*. Cambridge University Press, New York, 1998.

[145] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

[146] Wei Hou, Darwin Darakananda, and Jeff Eldredge. Machine learning based detection of flow disturbances using surface pressure measurements. In *AIAA Scitech 2019 Forum*, page 1148, 2019.

[147] Peter L Houtekamer and Herschel L Mitchell. Ensemble kalman filtering. *Quarterly Journal of the Royal Meteorological Society: A journal of the atmospheric sciences, applied meteorology and physical oceanography*, 131(613):3269–3289, 2005.

[148] Peter L Houtekamer and Fuqing Zhang. Review of the ensemble Kalman filter for atmospheric data assimilation. *Monthly Weather Review*, 144(12):4489–4532, 2016.

[149] Svein Hovland, Jan Tommy Gravdahl, and Karen E Willcox. Explicit model predictive control for large-scale systems via model reduction. *Journal of guidance, control, and dynamics*, 31(4):918–926, 2008.

[150] William W Hsieh and Benyang Tang. Applying neural network models to prediction and data analysis in meteorology and oceanography. *Bulletin of the American Meteorological Society*, 79(9):1855–1870, 1998.

[151] J. D. Hunter. Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[152] Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren. *Automated machine learning: methods, systems, challenges*. Springer Nature, 2019.

[153] Traian Iliescu and Paul Fischer. Backscatter in the rational LES model. *Computers & Fluids*, 33(5-6):783–790, 2004.

[154] Christopher Irrgang, Niklas Boers, Maike Sonnewald, Elizabeth A Barnes, Christopher Kadow, Joanna Staneva, and Jan Saynisch-Wagner. Towards neural earth system modelling by integrating artificial intelligence in earth system science. *Nature Machine Intelligence*, 3(8):667–674, 2021.

[155] Raad I Issa. Solution of the implicitly discretised fluid flow equations by operator-splitting. *Journal of Computational Physics*, 62(1):40–65, 1986.

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

[157] 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.

[158] 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.

[159] Javier Jiménez. Cascades in wall-bounded turbulence. *Annual Review of Fluid Mechanics*, 44:27–45, 2012.

[160] Siddharth Joshi and Stephen Boyd. Sensor selection via convex optimization. *IEEE Transactions on Signal Processing*, 57(2):451–462, 2008.

[161] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45, 03 1960.

[162] Eugenia Kalnay. *Atmospheric modeling, data assimilation and predictability*. Cambridge University Press, Cambridge, 2003.

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

[164] George Em Karniadakis, Ioannis G Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.

[165] Andrej Karpathy, Justin Johnson, and Li Fei-Fei. Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078*, 2015.

[166] Anuj Karpatne, Gowtham Atluri, James H Faghmous, Michael Steinbach, Arindam Banerjee, Auroop Ganguly, Shashi Shekhar, Nagiza Samatova, and Vipin 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.

[167] K Kashinath, M Mustafa, A Albert, JL Wu, C Jiang, S Esmaeilzadeh, K Azizzadenesheli, R Wang, A Chattopadhyay, A Singh, et al. Physics-informed machine learning: case studies for weather and climate modelling. *Philosophical Transactions of the Royal Society A*, 379(2194):20200093, 2021.

[168] Ioannis G Kevrekidis and Giovanni Samaey. Equation-free multiscale computation: Algorithms and applications. *Annual review of physical chemistry*, 60:321–344, 2009.

[169] Sina Khani and Michael L Waite. Large eddy simulations of stratified turbulence: the dynamic smagorinsky model. *Journal of Fluid Mechanics*, 773:327–344, 2015.

[170] Hyojin Kim, Junhyuk Kim, Sungjin Won, and Changhoon Lee. Unsupervised deep learning for super-resolution reconstruction of turbulence. *Journal of Fluid Mechanics*, 910, 2021.

[171] Jinpyo Kim, Wooekun Jung, Hyungmo Kim, and Jaejin Lee. CyCNN: a rotation invariant CNN using polar mapping and cylindrical convolution layers. *arXiv preprint arXiv:2007.10588*, 2020.

[172] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Accurate image super-resolution using very deep convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1646–1654, 2016.

[173] Junhyuk Kim and Changhoon Lee. Prediction of turbulent heat transfer using convolutional neural networks. *Journal of Fluid Mechanics*, 882:A18, 2020.

[174] Ryan King, Oliver Hennigh, Arvind Mohan, and Michael Chertkov. From deep to physics-informed learning of turbulence: Diagnostics. *arXiv preprint arXiv:1810.07785*, 2018.

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

[176] Gokhan Kirkil, Jeff Mirocha, Elie Bou-Zeid, Fotini Katopodes Chow, and Branko Kosović. Implementation and evaluation of dynamic subfilter-scale stress models for large-eddy simulation using wrf. *Monthly Weather Review*, 140(1):266–284, 2012.

[177] Vassili Kitsios, Jorgen S Frederiksen, and Meelis J Zidikheri. Theoretical comparison of subgrid turbulence in atmospheric and oceanic quasi-geostrophic models. *Nonlinear Processes in Geophysics*, 23(2), 2016.

[178] Jan Kleissl, Vijayant Kumar, Charles Meneveau, and Marc B Parlange. Numerical study of dynamic Smagorinsky models in large-eddy simulation of the atmospheric boundary layer: Validation in stable and unstable conditions. *Water Resources Research*, 42(6), 2006.

[179] Simon Kneer, Taraneh Sayadi, Denis Sipp, Peter Schmid, and Georgios Rigas. Symmetry-aware autoencoders: s-pca and s-nlpca. *arXiv preprint arXiv:2111.02893*, 2021.

[180] Robert H Kraichnan. The structure of isotropic turbulence at very high Reynolds numbers. *Journal of Fluid Mechanics*, 5(4):497–543, 1959.

[181] Robert H Kraichnan. Inertial ranges in two-dimensional turbulence. *The Physics of Fluids*, 10(7):1417–1423, 1967.

[182] Robert H Kraichnan and David Montgomery. Two-dimensional turbulence. *Reports on Progress in Physics*, 43(5):547, 1980.

[183] W Kramer, GH Keetels, HJH Clercx, and GJF van Heijst. Structure-function scaling of bounded two-dimensional turbulence. *Physical Review E*, 84(2):026310, 2011.

[184] Vladimir M Krasnopolsky and Michael S Fox-Rabinovitz. Complex hybrid models combining deterministic and machine learning components for numerical climate modeling and weather prediction. *Neural Networks*, 19(2):122–134, 2006.

[185] Andreas Krause, Ajit Singh, and Carlos Guestrin. Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. *Journal of Machine Learning Research*, 9(2), 2008.

[186] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

[187] 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.

[188] J Nathan Kutz. Deep learning in fluid dynamics. *Journal of Fluid Mechanics*, 814:1–4, 2017.

[189] J Nathan Kutz, Steven L Brunton, Bingni W Brunton, and Joshua L Proctor. *Dynamic mode decomposition: data-driven modeling of complex systems*. SIAM, 2016.

[190] J Nathan Kutz, Xing Fu, and Steven L Brunton. Multiresolution dynamic mode decomposition. *SIAM Journal on Applied Dynamical Systems*, 15(2):713–735, 2016.

[191] 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.

[192] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6405–6416, Red Hook, NY, USA, 2017. Curran Associates Inc.

[193] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6405–6416, Red Hook, NY, USA, 2017. Curran Associates Inc.

[194] Corentin J Lapeyre, Antony Misdariis, Nicolas Cazard, Denis Veynante, and Thierry Poinsot. Training convolutional neural networks to estimate turbulent sub-grid scale reaction rates. *Combustion and Flame*, 203:255–264, 2019.

[195] Brian Edward Launder, G Jr Reece, and W Rodi. Progress in the development of a Reynolds-stress turbulence closure. *Journal of Fluid Mechanics*, 68(3):537–566, 1975.

[196] 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.

[197] William Layton and Roger Lewandowski. A simple and stable scale-similarity model for large eddy simulation: energy balance and existence of weak solutions. *Applied Mathematics Letters*, 16(8):1205–1209, 2003.

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

[199] 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.

[200] Kookjin Lee and Kevin Carlberg. Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders. *arXiv preprint arXiv:1812.08373*, 2018.

[201] Kookjin Lee and Kevin Carlberg. Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders. *Journal of Computational Physics*, 404:108973, 2020.

[202] Kookjin Lee and Kevin T Carlberg. Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders. *Journal of Computational Physics*, 404:108973, 2020.

[203] Stefan Lee, Senthil Purushwalkam, Michael Cogswell, David Crandall, and Dhruv Batra. Why m heads are better than one: Training a diverse ensemble of deep networks. *arXiv preprint arXiv:1511.06314*, 2015.

[204] CE Leith. Atmospheric predictability and two-dimensional turbulence. *Journal of the Atmospheric Sciences*, 28(2):145–161, 1971.

[205] A Leonard. Energy cascade in large-eddy simulations of turbulent fluid flows. In *Advances in Geophysics*, volume 18, pages 237–248. Elsevier, 1975.

[206] John M Lewis, Sivaramakrishnan Lakshmivarahan, and Sudarshan Dhall. *Dynamic data assimilation: a least squares approach*, volume 104. Cambridge University Press, Cambridge, 2006.

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

[208] 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.

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

[210] Shi-Jun Liao. A general approach to get series solution of non-similarity boundary-layer flows. *Communications in Nonlinear Science and Numerical Simulation*, 14(5):2144–2159, 2009.

[211] Richard Liaw, Eric Liang, Robert Nishihara, Philipp Moritz, Joseph E. Gonzalez, and Ion Stoica. Tune: A research platform for distributed model selection and training, 2018.

[212] Douglas K Lilly. A proposed modification of the Germano subgrid-scale closure method. *Physics of Fluids A: Fluid Dynamics*, 4(3):633–635, 1992.

[213] 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.

[214] Julia Ling, Anthony Ruiz, Guilhem Lacaze, and Joseph Oefelein. Uncertainty analysis and data-driven model advances for a jet-in-crossflow. *Journal of Turbomachinery*, 139(2):021008, 2017.

[215] C Liu, R Fu, D Xiao, R Stefanescu, P Sharma, C Zhu, S Sun, and C Wang. Enkf data-driven reduced order assimilation system. *Engineering Analysis with Boundary Elements*, 139:46–55, 2022.

[216] Changhui Liu, Tao Liu, Juan Du, Yansong Zhang, Xinmin Lai, and Jianjun Shi. Hybrid nonlinear variation modeling of compliant metal plate assemblies considering welding shrinkage and angular distortion. *Journal of Manufacturing Science and Engineering*, 142(4), 2020.

[217] Shewen Liu, Charles Meneveau, and Joseph Katz. On the properties of similarity subgrid-scale models as deduced from measurements in a turbulent jet. *Journal of Fluid Mechanics*, 275:83–119, 1994.

[218] Shewen Liu, Charles Meneveau, and Joseph Katz. Experimental study of similarity subgrid-scale models of turbulence in the far-field of a jet. *Applied Scientific Research*, 54(3):177–190, 1995.

[219] 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.

[220] Zichao Long, Yiping Lu, Xianzhong Ma, and Bin Dong. PDE-net: Learning PDEs from data. *arXiv preprint arXiv:1710.09668*, 2017.

[221] Andrew C. Lorenc. The potential of the ensemble kalman filter for nwp—a comparison with 4d-var. *Quarterly Journal of the Royal Meteorological Society*, 129(595):3183–3203, 2003.

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

[223] Hugo F. S. Lui and William R. Wolf. Construction of reduced-order models for fluid flows using deep feedforward neural networks. *Journal of Fluid Mechanics*, 872:963–994, 2019.

[224] Julian Mack, Rossella Arcucci, Miguel Molina-Solana, and Yi-Ke Guo. Attention-based convolutional autoencoders for 3d-variational data assimilation. *Computer Methods in Applied Mechanics and Engineering*, 372:113291, 2020.

[225] Jim Magiera, Deep Ray, Jan S Hesthaven, and Christian Rohde. Constraint-aware neural networks for Riemann problems. *arXiv preprint arXiv:1904.12794*, 2019.

[226] Krithika Manohar, Bingni W Brunton, J Nathan Kutz, and Steven L Brunton. Data-driven sparse sensor placement for reconstruction: Demonstrating the benefits of exploiting known patterns. *IEEE Control Systems Magazine*, 38(3):63–86, 2018.

[227] Pablo Márquez-Neila, Mathieu Salzmann, and Pascal Fua. Imposing hard constraints on deep networks: Promises and limitations. *arXiv preprint arXiv:1706.02025*, 2017.

[228] PJ Mason and NS Callen. On the magnitude of the subgrid-scale eddy coefficient in large-eddy simulations of turbulent channel flow. *Journal of Fluid Mechanics*, 162:439–462, 1986.

[229] Romit Maulik, Romain Egele, Bethany Lusch, and Prasanna Balaprakash. Recurrent neural network architecture search for geophysical emulation. *arXiv preprint arXiv:2004.10928*, 2020.

[230] Romit Maulik, Bethany Lusch, and Prasanna Balaprakash. Non-autoregressive time-series methods for stable parametric reduced-order models. *Physics of Fluids*, 32(8):087115, 2020.

[231] Romit Maulik, Bethany Lusch, and Prasanna Balaprakash. Reduced-order modeling of advection-dominated systems with recurrent neural networks and convolutional autoencoders. *Physics of Fluids*, 33(3):037106, 2021.

[232] Romit Maulik, Vishwas Rao, Jiali Wang, Gianmarco Mengaldo, Emil Constantinescu, Bethany Lusch, Prasanna Balaprakash, Ian Foster, and Rao Kotamarthi. Aieada 1.0: Efficient high-dimensional variational data assimilation with machine-learned reduced-order models. *Geoscientific Model Development Discussions*, pages 1–20, 2022.

[233] Romit Maulik and Omer San. A neural network approach for the blind deconvolution of turbulent flows. *Journal of Fluid Mechanics*, 831:151–181, 2017.

[234] Romit Maulik and Omer San. A stable and scale-aware dynamic modeling framework for subgrid-scale parameterizations of two-dimensional turbulence. *Computers & Fluids*, 158:11–38, 2017.

[235] Romit Maulik, Omer San, and Jamey D Jacob. Spatiotemporally dynamic implicit large eddy simulation using machine learning classifiers. *Physica D: Nonlinear Phenomena*, 406:132409, 2020.

[236] Romit Maulik, Omer San, Jamey D Jacob, and Christopher Crick. Sub-grid scale model classification and blending through deep learning. *Journal of Fluid Mechanics*, 870:784–812, 2019.

[237] Romit Maulik, Omer San, Adil Rasheed, and Prakash Vedula. Data-driven deconvolution for large eddy simulations of kraichnan turbulence. *Physics of Fluids*, 30(12):125109, 2018.

[238] 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.

[239] 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.

[240] Amy McGovern, Ryan Lagerquist, David John Gagne, G Eli Jergensen, Kimberly L Elmore, Cameron R Homeyer, and Travis Smith. Making the black box more transparent: Understanding the physical implications of machine learning. *Bulletin of the American Meteorological Society*, 100(11):2175–2199, 2019.

[241] O Mcmillan, J Ferziger, and R Rogallo. Tests of subgrid-scale models in strained turbulence. In *13th Fluid and Plasma Dynamics Conference*, page 1339, 1980.

[242] Marcello Meldi and Alexandre Poux. A reduced order model based on kalman filtering for sequential data assimilation of turbulent flows. *Journal of Computational Physics*, 347:207–234, 2017.

[243] George L Mellor and Tetsuji Yamada. Development of a turbulence closure model for geophysical fluid problems. *Reviews of Geophysics*, 20(4):851–875, 1982.

[244] Charles Meneveau and Joseph Katz. Scale-invariance and turbulence models for large-eddy simulation. *Annual Review of Fluid Mechanics*, 32(1):1–32, 2000.

[245] Charles Meneveau, Thomas S Lund, and William H Cabot. A Lagrangian dynamic subgrid-scale model of turbulence. *Journal of Fluid Mechanics*, 319:353–385, 1996.

[246] Milano, Michele and Koumoutsakos, Petros. Neural network modeling for near wall turbulent flow. *Journal of Computational Physics*, 182(1):1–26, 2002.

[247] 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.

[248] Arvind T Mohan and Datta V Gaitonde. A deep learning based approach to reduced order modeling for turbulent flow control using lstm neural networks. *arXiv preprint arXiv:1804.09269*, 2018.

[249] 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.

[250] Muhammad Mohebujjaman, Leo G Rebholz, and Traian Iliescu. Physically constrained data-driven correction for reduced-order modeling of fluid flows. *International Journal for Numerical Methods in Fluids*, 89(3):103–122, 2019.

[251] Parviz Moin. *Fundamentals of engineering numerical analysis*. Cambridge University Press, Cambridge, 2010.

[252] Parviz Moin and Krishnan Mahesh. Direct numerical simulation: a tool in turbulence research. *Annual Review of Fluid Mechanics*, 30(1):539–578, 1998.

[253] Parviz Moin, Kyle Squires, W Cabot, and Sangsan Lee. A dynamic subgrid-scale model for compressible turbulence and scalar transport. *Physics of Fluids A: Fluid Dynamics*, 3(11):2746–2757, 1991.

[254] Vincent Mons, J-C Chassaing, Thomas Gomez, and Pierre Sagaut. Reconstruction of unsteady viscous flows using data assimilation schemes. *Journal of Computational Physics*, 316:255–280, 2016.

[255] Grégoire Montavon, Alexander Binder, Sebastian Lapuschkin, Wojciech Samek, and Klaus-Robert Müller. Layer-wise relevance propagation: an overview. *Explainable AI: interpreting, explaining and visualizing deep learning*, pages 193–209, 2019.

[256] Changhong Mou, Zhu Wang, David R Wells, Xuping Xie, and Traian Iliescu. Reduced order models for the quasi-geostrophic equations: A brief survey. *Fluids*, 6(1):16, 2020.

[257] Takaaki Murata, Kai Fukami, and Koji Fukagata. Nonlinear mode decomposition with convolutional neural networks for fluid dynamics. *Journal of Fluid Mechanics*, 882, 2020.

[258] BT Nadiga and D Livescu. Instability of the perfect subgrid model in implicit-filtering large eddy simulation of geostrophic turbulence. *Physical Review E*, 75(4):046303, 2007.

[259] Nirmal J Nair and Andres Goza. Leveraging reduced-order models for state estimation using deep learning. *Journal of Fluid Mechanics*, 897:R1, 2020.

[260] Ionel M Navon. Data assimilation for numerical weather prediction: a review. *Data assimilation for atmospheric, oceanic and hydrologic applications*, pages 21–65, 2009.

[261] Radford M Neal. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012.

[262] 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.

[263] Michael A Nielsen. *Neural networks and deep learning*, volume 25. Determination Press, San Francisco, 2015.

[264] Zacharias M Nikolaou, Charalambos Chrysostomou, Luc Vervisch, and Stewart Cant. Modelling turbulent premixed flames using convolutional neural networks: application to sub-grid scale variance and filtered reaction rate. *arXiv preprint arXiv:1810.07944*, 2018.

[265] ZM Nikolaou, C Chrysostomou, L Vervisch, and S Cant. Progress variable variance and filtered rate modelling using convolutional neural networks and flamelet methods. *Flow, Turbulence and Combustion*, pages 1–17, 2019.

[266] 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.

[267] Guido Novati, Hugues Lascombes de Laroussilhe, and Petros Koumoutsakos. Automating turbulence modeling by multi-agent reinforcement learning. *arXiv preprint arXiv:2005.09023*, 2020.

[268] Guido Novati, Hugues Lascombes de Laroussilhe, and Petros Koumoutsakos. Automating turbulence modelling by multi-agent reinforcement learning. *Nature Machine Intelligence*, 3(1):87–96, 2021.

[269] M Oberlack. Invariant modeling in large-eddy simulation of turbulence. *Annual Research Briefs*, pages 3–22, 1997.

[270] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2(11):e7, 2017.

[271] Paolo Orlandi. *Fluid flow phenomena: a numerical toolkit*, volume 55. Springer Science & Business Media, 2000.

[272] Paolo Orlandi. *Fluid flow phenomena: a numerical toolkit*, volume 55. Springer Science & Business Media, 2012.

[273] Ersin Ozbenli and Prakash Vedula. High order accurate finite difference schemes based on symmetry preservation. *Journal of Computational Physics*, 349:376–398, 2017.

[274] Ersin Ozbenli and Prakash Vedula. Numerical solution of modified differential equations based on symmetry preservation. *Physical Review E*, 96(6):063304, 2017.

[275] Ersin Ozbenli and Prakash Vedula. Construction of invariant compact finite-difference schemes. *Physical Review E*, 101(2):023303, 2020.

[276] Anikesh Pal. Deep learning parameterization of subgrid scales in wall-bounded turbulent flows. *arXiv preprint arXiv:1905.12765*, 2019.

[277] 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.

[278] Shaowu Pan and Karthik Duraisamy. Long-time predictive modeling of nonlinear dynamical systems using neural networks. *Complexity*, 2018, 2018.

[279] Eric J Parish and Karthik Duraisamy. A paradigm for data-driven predictive modeling using field inversion and machine learning. *Journal of Computational Physics*, 305:758–774, 2016.

[280] Noma Park and Krishnan Mahesh. Reduction of the Germano-identity error in the dynamic Smagorinsky model. *Physics of Fluids*, 21(6):065106, 2009.

[281] 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.

[282] Jaideep Pathak, Shashank Subramanian, Peter Harrington, Sanjeev Raja, Ashesh Chattopadhyay, Morteza Mardani, Thorsten Kurth, David Hall, Zongyi Li, and Kamyar Azizzadenesheli. Fourcastnet: A global data-driven high-resolution weather model using adaptive fourier neural operators. *arXiv preprint arXiv:2202.11214*, 2022.

[283] S Pawar, O San, A Rasheed, and I M Navon. A nonintrusive hybrid neural-physics modeling of incomplete dynamical systems: Lorenz equations. *GEM - International Journal on Geomathematics*, 2021.

[284] S Pawar, O San, A Rasheed, 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.

[285] Suraj Pawar, SM Rahman, H Vaddireddy, Omer San, Adil Rasheed, and Prakash Vedula. A deep learning enabler for nonintrusive reduced order modeling of fluid flows. *Physics of Fluids*, 31(8):085101, 2019.

[286] Suraj Pawar and Omer San. Data assimilation empowered neural network parametrizations for subgrid processes in geophysical flows. *Physical Review Fluids*, 6(5):050501, 2021.

[287] Suraj Pawar, Omer San, Burak Aksoylu, Adil Rasheed, and Trond Kvamsdal. Physics guided machine learning using simplified theories. *Physics of Fluids*, 33(1):011701, 2021.

[288] Suraj Pawar, Omer San, Adil Rasheed, and Prakash Vedula. Frame invariant neural network closures for Kraichnan turbulence. *arXiv preprint arXiv:2201.02928*, 2022.

[289] Suraj Pawar, Omer San, and Gary G Yen. Hyperparameter search using the genetic algorithm for surrogate modeling of geophysical flows. In *AIAA SCITECH 2022 Forum*, page 0459, 2022.

[290] Benjamin Peherstorfer, Karen Willcox, and Max Gunzburger. Survey of multifidelity methods in uncertainty propagation, inference, and optimization. *SIAM Review*, 60(3):550–591, 2018.

[291] Stephen G Penny, Timothy A Smith, T-C Chen, Jason A Platt, H-Y Lin, Michael Goodliff, and Henry DI Abarbanel. Integrating recurrent neural networks with data assimilation for scalable data-driven state estimation. *Journal of Advances in Modeling Earth Systems*, 14(3):e2021MS002843, 2022.

[292] Paris Perdikaris, Maziar Raissi, Andreas Damianou, Neil D Lawrence, and George Em Karniadakis. Nonlinear information fusion algorithms for data-efficient multi-fidelity modelling. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 473(2198):20160751, 2017.

[293] Mathis Peyron, Anthony Fillion, Selime Gürol, Victor Marchais, Serge Gratton, Pierre Boudier, and Gael Goret. Latent space data assimilation by using deep learning. *Quarterly Journal of the Royal Meteorological Society*, 147(740):3759–3777, 2021.

[294] Ugo Piomelli, William H Cabot, Parviz Moin, and Sangsan Lee. Subgrid-scale backscatter in turbulent and transitional flows. *Physics of Fluids A: Fluid Dynamics*, 3(7):1766–1771, 1991.

[295] Ugo Piomelli, Parviz Moin, and Joel H Ferziger. Model consistency in large eddy simulation of turbulent channel flows. *The Physics of Fluids*, 31(7):1884–1891, 1988.

[296] SB Pope. A more general effective-viscosity hypothesis. *Journal of Fluid Mechanics*, 72(2):331–340, 1975.

[297] Stephen B Pope and Stephen B Pope. *Turbulent flows*. Cambridge university press, 2000.

[298] Andrey A Popov, Changhong Mou, Adrian Sandu, and Traian Iliescu. A multifidelity ensemble kalman filter with reduced order control variates. *SIAM Journal on Scientific Computing*, 43(2):A1134–A1162, 2021.

[299] Andrey A Popov, Adrian Sandu, Elias D Nino-Ruiz, and Geir Evensen. A stochastic covariance shrinkage approach in ensemble transform kalman filtering. *arXiv preprint arXiv:2003.00354*, 2020.

[300] Aviral Prakash, Kenneth E Jansen, and John A Evans. Invariant data-driven subgrid stress modeling in the strain-rate eigenframe for large eddy simulation. *arXiv preprint arXiv:2106.13410*, 2021.

[301] Apostolos F Psaros, Xuhui Meng, Zongren Zou, Ling Guo, and George Em Karniadakis. Uncertainty quantification in scientific machine learning: Methods, metrics, and comparisons. *arXiv preprint arXiv:2201.07766*, 2022.

[302] Vladimir Puzyrev, Mehdi Ghommem, and Shiv Meka. pyROM: A computational framework for reduced order modeling. *Journal of Computational Science*, 30:157–173, 2019.

[303] Elizabeth Qian, Boris Kramer, Benjamin Peherstorfer, and Karen Willcox. Lift & learn: Physics-informed machine learning for large-scale nonlinear dynamical systems. *Physica D: Nonlinear Phenomena*, 406:132401, 2020.

[304] César Andres Quilodrán Casas. *Fast ocean data assimilation and forecasting using a neural-network reduced-space regional ocean model of the north Brazil current*. PhD thesis, Imperial College London, 2018.

[305] Florence Rabier. Overview of global data assimilation developments in numerical weather-prediction centres. *Quarterly Journal of the Royal Meteorological Society*, 131(613):3215–3233, 2005.

[306] Sk M Rahman, Suraj Pawar, Omer San, Adil Rasheed, and Traian Iliescu. Nonintrusive reduced order modeling framework for quasigeostrophic turbulence. *Physical Review E*, 100(5):053306, 2019.

[307] Sk Mashfiqur Rahman, Suraj Pawar, Omer San, Adil Rasheed, and Traian Iliescu. Nonintrusive reduced order modeling framework for quasigeostrophic turbulence. *Physical Review E*, 100:053306, 2019.

[308] Maziar Raissi and George Em Karniadakis. Hidden physics models: Machine learning of nonlinear partial differential equations. *Journal of Computational Physics*, 357:125–141, 2018.

[309] 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.

[310] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Numerical gaussian processes for time-dependent and nonlinear partial differential equations. *SIAM Journal on Scientific Computing*, 40(1):A172–A198, 2018.

[311] David A Randall. Cloud parameterization for climate modeling: Status and prospects. *Atmospheric research*, 23(3-4):345–361, 1989.

[312] David A. Randall, Cecilia M. Bitz, Gokhan Danabasoglu, A. Scott Denning, Peter R. Gent, Andrew Gettelman, Stephen M. Griffies, Peter Lynch, Hugh Morrison, Robert Pincus, and John Thuburn. 100 years of earth system model development. *Meteorological Monographs*, 59:12.1–12.66, 2018.

[313] Adil Rasheed, Omer San, and Trond Kvamsdal. Digital twin: Values, challenges and enablers from a modeling perspective. *IEEE Access*, 8:21980–22012, 2020.

[314] 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.

[315] Stephan Rasp, Peter D Dueben, Sebastian Scher, Jonathan A Weyn, Soukayna Mouatadid, and Nils Thuerey. Weatherbench: a benchmark data set for data-driven weather forecasting. *Journal of Advances in Modeling Earth Systems*, 12(11):e2020MS002203, 2020.

[316] 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.

[317] Stephan Rasp and Nils Thuerey. Data-driven medium-range weather prediction with a resnet pretrained on climate simulations: A new model for weatherbench. *Journal of Advances in Modeling Earth Systems*, 13(2):e2020MS002405, 2021.

[318] 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.

[319] D Rempfer. On low-dimensional Galerkin models for fluid flow. *Theoretical and Computational Fluid Dynamics*, 14(2):75–88, 2000.

[320] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, pages 91–99, 2015.

[321] Richard W Reynolds. A real-time global sea surface temperature analysis. *Journal of climate*, 1(1):75–87, 1988.

[322] Richard W Reynolds and Diane C Marsico. An improved real-time global sea surface temperature analysis. *Journal of climate*, 6(1):114–119, 1993.

[323] Haakon Robinson, Adil Rasheed, and Omer San. Dissecting deep neural networks. *arXiv preprint arXiv:1910.03879*, 2019.

[324] Robert S Rogallo and Parviz Moin. Numerical simulation of turbulent flows. *Annual review of fluid mechanics*, 16(1):99–137, 1984.

[325] Clarence W Rowley, Tim Colonius, and Richard M Murray. Model reduction for compressible flows using POD and Galerkin projection. *Physica D: Nonlinear Phenomena*, 189(1-2):115–129, 2004.

[326] Clarence W Rowley and Scott TM Dawson. Model reduction for flow analysis and control. *Annual Review of Fluid Mechanics*, 49:387–417, 2017.

[327] Yvonne Ruckstuhl, Tijana Janjić, and Stephan Rasp. Training a convolutional neural network to conserve mass in data assimilation. *Nonlinear Processes in Geophysics*, 28(1):111–119, 2021.

[328] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[329] Samuel H Rudy, Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Data-driven discovery of partial differential equations. *Science Advances*, 3(4):e1602614, 2017.

[330] Pierre Sagaut. *Large eddy simulation for incompressible flows: an introduction*. Springer Science & Business Media, 2006.

[331] 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.

[332] Omer San. A dynamic eddy-viscosity closure model for large eddy simulations of two-dimensional decaying turbulence. *International Journal of Computational Fluid Dynamics*, 28(6-10):363–382, 2014.

[333] Omer San and Traian Iliescu. A stabilized proper orthogonal decomposition reduced-order model for large scale quasigeostrophic ocean circulation. *Advances in Computational Mathematics*, 41(5):1289–1319, 2015.

[334] Omer San and Romit Maulik. Neural network closures for nonlinear model order reduction. *Advances in Computational Mathematics*, 44(6):1717–1750, 2018.

[335] Omer San, Adil Rasheed, and Trond Kvamsdal. Hybrid analysis and modeling, eclecticism, and multifidelity computing toward digital twin revolution. *GAMM-Mitteilungen*, page e202100007, 2021.

[336] Omer San and Anne E Staples. High-order methods for decaying two-dimensional homogeneous isotropic turbulence. *Computers & Fluids*, 63:105–127, 2012.

[337] Omer San, Anne E Staples, and Traian Iliescu. Approximate deconvolution large eddy simulation of a stratified two-layer quasigeostrophic ocean model. *Ocean Modelling*, 63:1–20, 2013.

[338] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter Battaglia. Learning to simulate complex physics with graph networks. In *International Conference on Machine Learning*, pages 8459–8468. PMLR, 2020.

[339] Matteo Sangiorgio and Fabio Dercole. Robustness of LSTM neural networks for multi-step forecasting of chaotic time series. *Chaos, Solitons & Fractals*, 139:110045, 2020.

[340] 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.

[341] F Sarghini, U Piomelli, and E Balaras. Scale-similar models for large-eddy simulations. *Physics of Fluids*, 11(6):1596–1607, 1999.

[342] Jonathan Schmidt, Mário RG Marques, Silvana Botti, and Miguel AL Marques. Recent advances and applications of machine learning in solid-state materials science. *NPJ Computational Materials*, 5(1):1–36, 2019.

[343] 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.

[344] MG Schultz, Clara Betancourt, Bing Gong, Felix Kleinert, Michael Langguth, LH Leufen, Amirpasha Mozaffari, and Scarlet Stadtler. Can deep learning beat numerical weather prediction? *Philosophical Transactions of the Royal Society A*, 379(2194):20200097, 2021.

[345] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. *Advances in neural information processing systems*, 28, 2015.

[346] 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.

[347] B Siddani, S Balachandar, and Ruogu Fang. Rotational and reflectional equivariant convolutional neural network for data-limited applications: Multiphase flow demonstration. *Physics of Fluids*, 33(10):103323, 2021.

[348] Dan Simon. *Optimal state estimation: Kalman, H infinity, and nonlinear approaches*. John Wiley & Sons, New York, 2006.

[349] Lawrence Sirovich. Turbulence and the dynamics of coherent structures. I. Coherent structures. *Quarterly of Applied Mathematics*, 45(3):561–571, 1987.

[350] Joseph Smagorinsky. General circulation experiments with the primitive equations: I. the basic experiment. *Monthly Weather Review*, 91(3):99–164, 1963.

[351] Alexander J Smits and Ivan Marusic. Wall-bounded turbulence. *Phys. Today*, 66(9):25–30, 2013.

[352] D. B. Spalding. A Single Formula for the "Law of the Wall". *Journal of Applied Mechanics*, 28(3):455–458, 1961.

[353] Charles G Speziale. Galilean invariance of subgrid-scale stress models in the large-eddy simulation of turbulence. *Journal of Fluid Mechanics*, 156:55–62, 1985.

[354] 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.

[355] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[356] Vishal Srivastava and Karthik Duraisamy. Generalizable physics-constrained modeling using learning and inference assisted by feature space engineering. *arXiv preprint arXiv:2103.16042*, 2021.

[357] Răzvan Ştefănescu, Adrian Sandu, and Ionel Michael Navon. POD/DEIM reduced-order strategies for efficient four dimensional variational data assimilation. *Journal of Computational Physics*, 295:569–595, 2015.

[358] Karen Stengel, Andrew Glaws, Dylan Hettinger, and Ryan N King. Adversarial super-resolution of climatological wind and solar data. *Proceedings of the National Academy of Sciences*, 117(29):16805–16815, 2020.

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

[360] R. Stoffer, C. M. van Leeuwen, D. Podareanu, V. Codreanu, M. A. Veerman, M. Janssens, O. K. Hartogensis, and C. C. van Heerwaarden. Development of a large-eddy simulation subgrid model based on artificial neural networks: a case study of turbulent channel flow. *Geoscientific Model Development*, 14(6):3769–3788, 2021.

[361] S Stolz and Nikolaus A Adams. An approximate deconvolution procedure for large-eddy simulation. *Physics of Fluids*, 11(7):1699–1701, 1999.

[362] S Stolz, Nikolaus A Adams, and Leonhard Kleiser. An approximate deconvolution model for large-eddy simulation with application to incompressible wall-bounded flows. *Physics of fluids*, 13(4):997–1015, 2001.

[363] Adam Subel, Ashesh Chattopadhyay, Yifei Guan, and Pedram Hassanzadeh. Data-driven subgrid-scale modeling of forced burgers turbulence using deep learning with generalization to higher reynolds numbers via transfer learning. *Physics of Fluids*, 33(3):031702, 2021.

[364] 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.

[365] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International Conference on Machine Learning*, pages 1139–1147, 2013.

[366] Patrick Tabeling. Two-dimensional turbulence: a physicist approach. *Physics Reports*, 362(1):1–62, 2002.

[367] Kai Sheng Tai, Peter Bailis, and Gregory Valiant. Equivariant transformer networks. In *International Conference on Machine Learning*, pages 6086–6095. PMLR, 2019.

[368] Kunihiko Taira. Revealing essential dynamics from high-dimensional fluid flow data and operators. *arXiv preprint arXiv:1903.01913*, 2019.

[369] Kunihiko Taira, Maziar S Hemati, Steven L Brunton, Yiyang Sun, Karthik Duraisamy, Shervin Bagheri, Scott TM Dawson, and Chi-An Yeh. Modal analysis of fluid flows: Applications and outlook. *AIAA Journal*, pages 1–25, 2019.

[370] 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.

[371] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. KPConv: flexible and deformable convolution for point clouds. *arXiv preprint arXiv:1904.08889*, 2019.

[372] Robert Tibshirani. A comparison of some error estimates for neural network models. *Neural Computation*, 8(1):152–163, 1996.

[373] Brendan Tracey, Karthik Duraisamy, and Juan Alonso. Application of supervised learning to quantify uncertainties in turbulence and combustion modeling. In *51st AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, page 259, 2013.

[374] Brendan D Tracey, Karthikeyan Duraisamy, and Juan J Alonso. A machine learning strategy to assist turbulence model development. In *53rd AIAA Aerospace Sciences Meeting*, page 1287, 2015.

[375] Nathaniel Trask, Ravi G Patel, Ben J Gross, and Paul J Atzberger. GMLS-Nets: A framework for learning from unstructured data. *arXiv preprint arXiv:1909.05371*, 2019.

[376] Lloyd N Trefethen and David Bau III. *Numerical linear algebra*, volume 50. Siam, 1997.

[377] Anh Truong, Austin Walters, Jeremy Goodsitt, Keegan Hines, C Bayan Bruss, and Reza Farivar. Towards automated machine learning: Evaluation and comparison of automl approaches and tools. In *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 1471–1479. IEEE, 2019.

[378] Dirk L van Kekem and Alef E Sterk. Symmetries in the Lorenz-96 model. *International Journal of Bifurcation and Chaos*, 29(01):1950008, 2019.

[379] Paul Veers, Katherine Dykes, Eric Lantz, Stephan Barth, Carlo L Bottasso, Ola Carlson, Andrew Clifton, Johney Green, Peter Green, Hannele Holttinen, et al. Grand challenges in the science of wind energy. *Science*, 366(6464):eaau2027, 2019.

[380] Ricardo Vilalta and Youssef Drissi. A perspective view and survey of meta-learning. *Artificial intelligence review*, 18(2):77–95, 2002.

[381] 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.

[382] Pantelis R Vlachas, Wonmin Byeon, Zhong Y Wan, Themistoklis P Sapsis, and Petros Koumoutsakos. Data-driven forecasting of high-dimensional chaotic systems with long short-term memory networks. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 474(2213):20170844, 2018.

[383] 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.

[384] J Von Hardenberg, JC McWilliams, A Provenzale, A Shchepetkin, and JB Weiss. Vortex merging in quasi-geostrophic flows. *Journal of Fluid Mechanics*, 412:331–353, 2000.

[385] Nicholas Wagner and James M Rondinelli. Theory-guided machine learning in materials science. *Frontiers in Materials*, 3:28, 2016.

[386] Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In *International Conference on Machine Learning*, pages 1058–1066, 2013.

[387] 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.

[388] 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.

[389] 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.

[390] Rui Wang, Robin Walters, and Rose Yu. Incorporating symmetry into deep dynamics models for improved generalization. In *International Conference on Learning Representations*, 2021.

[391] Yunpeng Wang, Zelong Yuan, Chenyue Xie, and Jianchun Wang. Artificial neural network-based spatial gradient models for large-eddy simulation of turbulence. *AIP Advances*, 11(5):055216, 2021.

[392] Zheng Wang, Dunhui Xiao, Fangxin Fang, Rajesh Govindan, Christopher C Pain, and Yike Guo. Model identification of reduced order fluid dynamics systems using deep learning. *International Journal for Numerical Methods in Fluids*, 86(4):255–268, 2018.

[393] Zhuo Wang, Kun Luo, Dong Li, Junhua Tan, and Jianren Fan. Investigations of data-driven closure for subgrid-scale stress in large-eddy simulation. *Physics of Fluids*, 30(12):125101, 2018.

[394] Maurice Weiler and Gabriele Cesa. General $e(2)$-equivariant steerable cnns. *arXiv preprint arXiv:1911.08251*, 2019.

[395] Maurice Weiler, Mario Geiger, Max Welling, Wouter Boomsma, and Taco Cohen. 3d steerable cnns: Learning rotationally equivariant features in volumetric data. *arXiv preprint arXiv:1807.02547*, 2018.

[396] Maurice Weiler, Fred A Hamprecht, and Martin Storath. Learning steerable filters for rotation equivariant cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 849–858, 2018.

[397] Greg Welch and Gary Bishop. *An introduction to the Kalman filter*. Citeseer, 1995.

[398] Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

[399] Jonathan A Weyn, Dale R Durran, and Rich Caruana. Can machines learn to predict weather? using deep learning to predict gridded 500-hpa geopotential height from historical weather data. *Journal of Advances in Modeling Earth Systems*, 11(8):2680–2693, 2019.

[400] Frank M White and Joseph Majdalani. *Viscous fluid flow*, volume 3. McGraw-Hill, New York, 2006.

[401] Christopher K Wikle and L Mark Berliner. A bayesian tutorial for data assimilation. *Physica D: Nonlinear Phenomena*, 230(1-2):1–16, 2007.

[402] Alexander Wikner, Jaideep Pathak, Brian Hunt, Michelle Girvan, Troy Arcomano, Istvan Szunyogh, Andrew Pomerance, and Edward Ott. Combining machine learning with knowledge-based modeling for scalable forecasting and subgrid-scale closure of large, complex, spatiotemporal systems. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 30(5):053111, 2020.

[403] 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.

[404] Jan Williams, Olivia Zahn, and J Nathan Kutz. Data-driven sensor placement with shallow decoder networks. *arXiv preprint arXiv:2202.05330*, 2022.

[405] 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.

[406] Jin-Long Wu, Heng Xiao, and Eric Paterson. Physics-informed machine learning approach for augmenting turbulence models: A comprehensive framework. *Physical Review Fluids*, 3(7):074602, 2018.

[407] 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.

[408] Pin Wu, Xuting Chang, Wenyan Yuan, Junwu Sun, Wenjie Zhang, Rossella Arcucci, and Yike Guo. Fast data Assimilation (FDA): Data assimilation by machine learning for faster optimize model state. *Journal of Computational Science*, 51:101323, 2021.

[409] Pin Wu, Junwu Sun, Xuting Chang, Wenjie Zhang, Rossella Arcucci, Yike Guo, and Christopher C Pain. Data-driven reduced order model with temporal convolutional neural network. *Computer Methods in Applied Mechanics and Engineering*, 360:112766, 2020.

[410] Xiaohua Wu and Parviz Moin. Direct numerical simulation of turbulence in a nominally zero-pressure-gradient flat-plate boundary layer. *Journal of Fluid Mechanics*, 630:5–41, 2009.

[411] 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.

[412] D Xiao, F Fang, CE Heaney, IM Navon, and CC Pain. A domain decomposition method for the non-intrusive reduced order modelling of fluid flow. *Computer Methods in Applied Mechanics and Engineering*, 354:307–330, 2019.

[413] 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.

[414] 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.

[415] Jiayang Xu and Karthik Duraisamy. Multi-level convolutional autoencoder networks for parametric prediction of spatio-temporal dynamics. *Computer Methods in Applied Mechanics and Engineering*, 372:113379, 2020.

[416] Yifan Xu, Tianqi Fan, Mingye Xu, Long Zeng, and Yu Qiao. Spidercnn: Deep learning on point sets with parameterized convolutional filters. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 87–102, 2018.

[417] 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.

[418] Jun-Ichi Yano, Michał Z. Ziemiański, Mike Cullen, Piet Termonia, Jeanette Onvlee, Lisa Bengtsson, Alberto Carrassi, Richard Davy, Anna Deluca, Suzanne L. Gray, Víctor Homar, Martin Köhler, Simon Krichak, Silas Michaelides, Vaughan T. J. Phillips, Pedro M. M. Soares, and Andrzej A. Wyszogrodzki. Scientific challenges of convective-scale numerical weather prediction. *Bulletin of the American Meteorological Society*, 99(4):699–710, 2018.

[419] Jian Yu, Chao Yan, and Mengwu Guo. Non-intrusive reduced-order modeling for fluid problems: A brief review. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, 233(16):5896–5912, 2019.

[420] Zelong Yuan, Chenyue Xie, and Jianchun Wang. Deconvolutional artificial neural network models for large eddy simulation of turbulence. *Physics of Fluids*, 32(11):115106, 2020.

[421] Laure Zanna and Thomas Bolton. Data-driven equation discovery of ocean mesoscale closures. *Geophysical Research Letters*, 47(17):e2020GL088376, 2020.

[422] Mark D Zelinka, Timothy A Myers, Daniel T McCoy, Stephen Po-Chedley, Peter M Caldwell, Paulo Ceppi, Stephen A Klein, and Karl E Taylor. Causes of higher climate sensitivity in cmip6 models. *Geophysical Research Letters*, 47(1):e2019GL085782, 2020.

[423] 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.

[424] 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.

[425] Ye Zhou. Turbulence theories and statistical closure approaches. *Physics Reports*, 935:1–117, 2021.

[426] Zhideng Zhou, Guowei He, Shizhao Wang, and Guodong Jin. Subgrid-scale model for large-eddy simulation of isotropic turbulent flows using an artificial neural network. *Computers & Fluids*, 195:104319, 2019.

[427] Linyang Zhu, Weiwei Zhang, Jiaqing Kou, and Yilang Liu. Machine learning methods for turbulence modeling in subsonic flows around airfoils. *Physics of Fluids*, 31(1):015105, 2019.

[428] Ralf Zimmermann, Benjamin Peherstorfer, and Karen Willcox. Geometric subspace updates with applications to online adaptive nonlinear model reduction. *SIAM Journal on Matrix Analysis and Applications*, 39(1):234–261, 2018.

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

VITA

Suraj Pawar

Candidate for the Degree of

Doctor of Philosophy

Dissertation: PHYSICS-GUIDED MACHINE LEARNING FOR TURBULENCE CLOSURE AND REDUCED-ORDER MODELING

Major Field: Mechanical & Aerospace Engineering

Biographical:

Personal Data: Born in Pune, MH, India in June 1992.

Education:

Completed the requirements for the degree of Doctor of Philosophy with a major in Mechanical & Aerospace Engineering at Oklahoma State University in July 2022.
Completed the requirements for the degree of Master of Science with a major in Mechanical Engineering at Virginia Polytechnic Institute & State University in December 2018.
Received a Bachelor of Engineering in Mechanical Engineering at VJTI Mumbai, India in June 2014.

Experience:

Graduate Research Assistant Jan 2019 - July 2022
Oklahoma State University
Graduate Research Assistant Jan 2017 - Dec 2018
Virginia Polytechnic Institute & State University

Professional Membership:

American Physical Society (APS)
Society for Industrial and Applied Mathematics (SIAM)