UNIVERSITY OF CALIFORNIA SAN DIEGO

Physics-Guided Deep Learning for Dynamics Forecasting

A dissertation submitted in partial satisfaction of the
requirements for the degree Doctor of Philosophy

in

Computer Science and Engineering

by

Rui Wang

Committee in charge:

Professor Rose Yu, Chair
Professor Christos Faloutsos
Professor Sicun Gao
Professor Nicholas Lutsko
Professor Lawrence Saul

2023

University of California San Diego

2023

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I am deeply grateful to my advisor, Prof. Rose Yu, for her guidance and encouragement throughout my PhD journey. Her expertise and mentorship have been invaluable, especially in teaching me the value of enthusiasm and optimism in research. I also feel very fortunate to work with Robin Walters, who has been a great role model for me. He has shared valuable knowledge and insights in the fields of machine learning and mathematics, for which I am extremely grateful.

I am deeply grateful to my family, particularly my parents and my uncle Ke and aunt Tong, for their unwavering support, love, and encouragement throughout my entire PhD journey. Their belief in me and their constant presence have been a main source of strength and motivation. I would also like to express my gratitude to my girlfriend, Yeqing, whose understanding and care have been invaluable during stressful times. Her love and encouragement have played a vital role in keeping me motivated. I feel incredibly fortunate to have her by my side, and I wish her all the best in her studies at NEU in the future. I extend my gratitude to my best friends, Tony and Oliver, for dragging me away from the computer for much-needed fun and relaxation.

I would like to acknowledge a difficult period during my third year of research when I experienced self-doubt and questioned the significance of my work. I am grateful for the strength I found within myself to persevere and not give up. This experience taught me the importance of resilience and the understanding that the value of research often unfolds over time. I have come to realize that the true meaning of life lies in the process of seeking and discovering purpose. I want to express my appreciation for the sport of basketball and the discipline of weight training. These activities have provided me with a sense of balance, fulfillment, and accomplishment.

Since coming to the US for my master's and doctoral degrees, I have been fortunate to meet kind-hearted individuals who have generously supported me. Their impact on my academic and personal life is deeply appreciated. Finally, I express my aspiration to make meaningful contributions to scientific discovery in the future. This dissertation marks a significant milestone in my journey, and I am committed to continuing my pursuit of knowledge and innovation to drive progress and make a positive impact in the scientific community.

VITA

2017	Bachelor of Science, Huazhong University of Science and Technology
2019	Master of Science, Northeastern University
2023	Doctor of Philosophy, University of California San Diego

PUBLICATIONS

Rui Wang, Rose Yu., "Physics-Guided Deep Learning for Dynamical Systems: A survey", *In submission to ACM Computing Survey*, 2022.

Rui Wang, Yihe Dong, Sercan Arik, Rose Yu., "Koopman Neural Forecaster for Time-series with Temporal Distributional Shifts", *International Conference on Learning Representations (ICLR)*, 2023.

Rui Wang, Robin Walters, Rose Yu., " Meta-Learning Dynamics Forecasting Using Task Inference", *the Conference on Neural Information Processing Systems (NeurIPS)*, 2022.

Rui Wang, Robin Walters, Rose Yu., "Approximately Equivariant Networks for Imperfectly Symmetric Dynamics", *International Conference on Machine Learning (ICML)*, 2022.

Rui Wang, Robin Walters, Rose Yu., "Data Augmentation vs. Equivariant Networks: A Theory of Generalization on Dynamics Forecasting", *ICML Principles of Distribution Shift Workshop*, 2022.

Rui Wang, Robin Walters, Rose Yu., "Incorporating Symmetry into Deep Dynamics Models for Improved Generalization", *International Conference on Learning Representations (ICLR)*, 2021.

Rui Wang, Karthik Kashinath, Mustafa Mustafa, Adrian Albert, and Rose Yu., "Towards Physics-Informed Deep Learning for Turbulent Flow Prediction", *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2020.

ABSTRACT OF THE DISSERTATION

Physics-Guided Deep Learning for Dynamics Forecasting

by

Rui Wang

in Computer Science and Engineering

University of California San Diego, 2023

Professor Rose Yu, Chair

Modeling complex dynamics and forecasting the future is a fundamental task in science, such as turbulence modeling and weather forecasting. Physics-based models, which rely on mathematical principles, can accurately predict dynamics but can be computationally intensive, and the underlying physical principles may not be entirely understood. Deep Learning (DL) provides efficient alternatives to simulating dynamics but it lacks physical consistency and struggles with generalization. Thus, there is a growing need for integrating prior physics knowledge with deep learning to take the best of both types of approaches to better solve scientific problems. Thus, the study of physics-guided DL emerged and has made great progress.

In this thesis, we describe the physics-guide DL for dynamics forecasting and presented several approaches to improving the physical consistency, accuracy, and generalization of DL models for dynamics forecasting. The approaches include (1) incorporating prior physical knowledge into the design of model architecture and loss functions for improved physical consistency and accuracy, (2) leveraging model-based meta-learning for improved generalization across heterogeneous domains, (3) simplifying nonlinear dynamics with Koopman theory for improved generalization over temporal distributional shifts, and (4) incorporating symmetries into deep dynamics models for improved generalization across relevant symmetry groups and consistency with conservation laws. In the end, we summarize the challenges in this field and discuss the emerging opportunities for future research.

Chapter 1

Introduction

1.1 Background

Modeling complex physical dynamics over a wide range of spatial and temporal scales is a fundamental task in a wide range of fields including, for example, fluid dynamics [205], cosmology [220], economics[43], and neuroscience [93]. Dynamical systems are mathematical objects that are used to describe the evolution of phenomena over time and space occurring in nature. Dynamical systems are commonly described with differential equations which are equations related to one or more unknown functions and their derivatives.

Definition 1. *Fix an integer $k \geq 1$ and let U denote an open subset of $\mathbb{R}^n$. Let $u : U \mapsto \mathbb{R}^m$ and we write $\boldsymbol{u} = (u^1, ..., u^m)$, where $x \in U$. Then an expression of the form*

$$\mathscr{F}(D^k\boldsymbol{u}(x), D^{k-1}\boldsymbol{u}(x), ..., D\boldsymbol{u}(x), \boldsymbol{u}(x), x) = 0 \tag{1.1}$$

is called a k-th-order system of partial differential equation (or ordinary differential equation when $n = 1$), where $\mathscr{F} : \mathbb{R}^{mn^k} \times \mathbb{R}^{mn^{k-1}} \times ... \times \mathbb{R}^{mn} \times \mathbb{R}^m \times U \mapsto \mathbb{R}^m$.

$\mathscr{F}$ models the dynamics of a n-dimensional state $x \in \mathbb{R}^n$ and it can be either a linear or non-linear operator. Since most dynamics evolve over time, one of the variables of u is usually the time dimension. In general, one must specify appropriate boundary and initial conditions of Equ.1.1 to ensure the existence of a solution. Learning dynamical systems is to search for a model $\mathscr{F}$ that can accurately describe the behavior of the physical process insofar as we are interested.

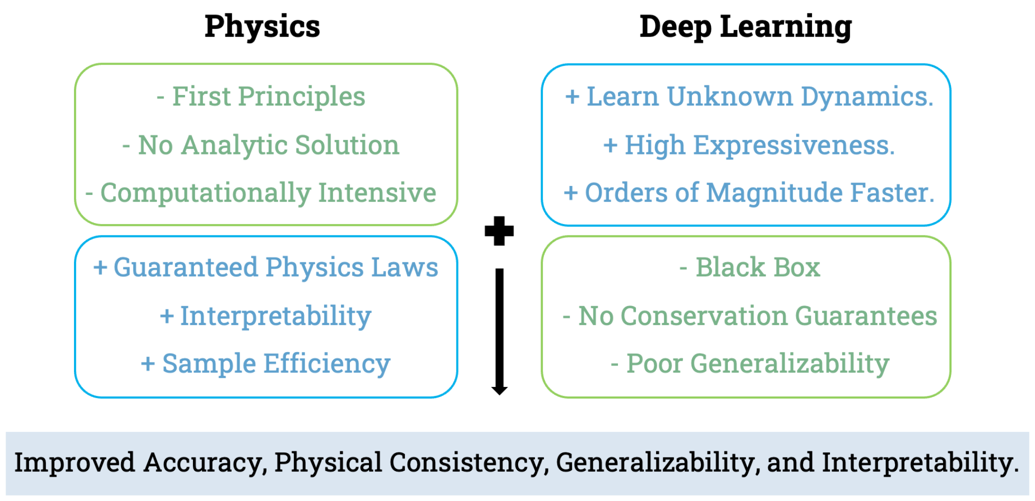
Physics
Deep Learning
- First Principles
- No Analytic Solution
- Computationally Intensive
+ Learn Unknown Dynamics.
+ High Expressiveness.
+ Orders of Magnitude Faster.
+ Guaranteed Physics Laws
+ Interpretability
+ Sample Efficiency
- Black Box
- No Conservation Guarantees
- Poor Generalizability
Improved Accuracy, Physical Consistency, Generalizability, and Interpretability.

Figure 1.1. The goal of physics-guided deep learning is to combine the strengths of both types of methods while overcoming their limitations by integrating existing knowledge of physics or physics-based modeling as inductive biases in the design of neural networks.

Physics as a discipline has a long tradition of using first principles to describe spatiotemporal dynamics. The laws of physics have greatly improved our understanding of the physical world. Many physics laws are described by systems of highly nonlinear differential equations that have direct implications for understanding and predicting physical dynamics. However, these equations are usually too complicated to be solvable. The current paradigm of numerical methods for solution approximation is purely physics-based: known physical laws encoded in systems of coupled differential equations are solved over space and time via numerical differentiation and integration schemes [85, 89, 133, 94, 151, 184]. However, these methods are tremendously computationally intensive, requiring significant computational resources and expertise. An alternative is to look for simplified models that are based on certain assumptions and can describe the dynamics, such as Reynolds-averaged Navier-stokes equations for turbulent flows and Euler equations for gas dynamics [30, 121, 210]. But it is highly nontrivial to obtain a simplified model that can describe a phenomenon with satisfactory accuracy. More importantly, for many complex real-world phenomena, only partial knowledge of their dynamics is known. The equations may not fully represent the true system states.

Deep Learning (DL) provides efficient alternatives to learn high-dimensional spatiotemporal dynamics from massive datasets. It achieves this by directly predicting the input-output mapping and bypassing numerical integration. Recent works have shown that DL can generate realistic predictions and significantly accelerate the simulation of physical dynamics relative to numerical solvers, from turbulence modeling to weather prediction [224, 111, 105, 103, 109]. This opens up new opportunities at the intersection of DL and physical sciences, such as molecular dynamics[196, 198], epidemiology[242], cardiology[132, 223] and material science [134, 27].

Despite the tremendous progress, DL is purely data-driven by nature, which has many limitations. DL models still adhere to the fundamental rules of statistical inference. The nonlinear and chaotic nature of real-world dynamics poses significant challenges to existing DL frameworks. Without explicit constraints, DL models are prone to make physically implausible forecasts, violating the governing laws of physical systems. Additionally, DL models often struggle with generalization: models trained on one dataset cannot adapt properly to unseen scenarios with different distributions, known as distribution shift. For dynamics learning, the distribution shift occurs not only because the dynamics are non-stationary and nonlinear, but also due to the changes in system parameters, such as external forces and boundary conditions. In a word, the current limitation of DL models for learning complex dynamics is their lack of ability to understand the system solely from data and cope with the distributional shifts that naturally occur across systems and time.

Neither DL alone nor purely physics-based approaches are sufficient for learning complex dynamical systems in scientific domains. Therefore, there is a growing need for integrating traditional physics-based approaches with DL models so that we can take the best from both types of approaches, as shown in Figure 1.1. There is already a vast amount of work about physics-guided DL [236, 52, 22, 117, 103, 176, 23], but the focus on deep learning for dynamical systems is still nascent. Physics-guided DL offers a set of tools to blend these physical concepts such as differential equations and symmetry with deep neural networks. On one hand, these DL models offer great computational benefits over traditional numerical solvers. On the other hand, the physical constraints impose appropriate inductive biases on the DL models, leading to accurate simulation, scientifically valid predictions, reduced sample complexity, and guaranteed improvement in generalization to unknown environments.

Figure 1.2 provides an overview of this thesis. We will commence with an overview of the significance of physics-guided DL, followed by the formulation of four key problems in this field. Subsequently, we will present four of our contributions towards the prediction of dynamic systems, and conclude by discussing potential avenues for future research.

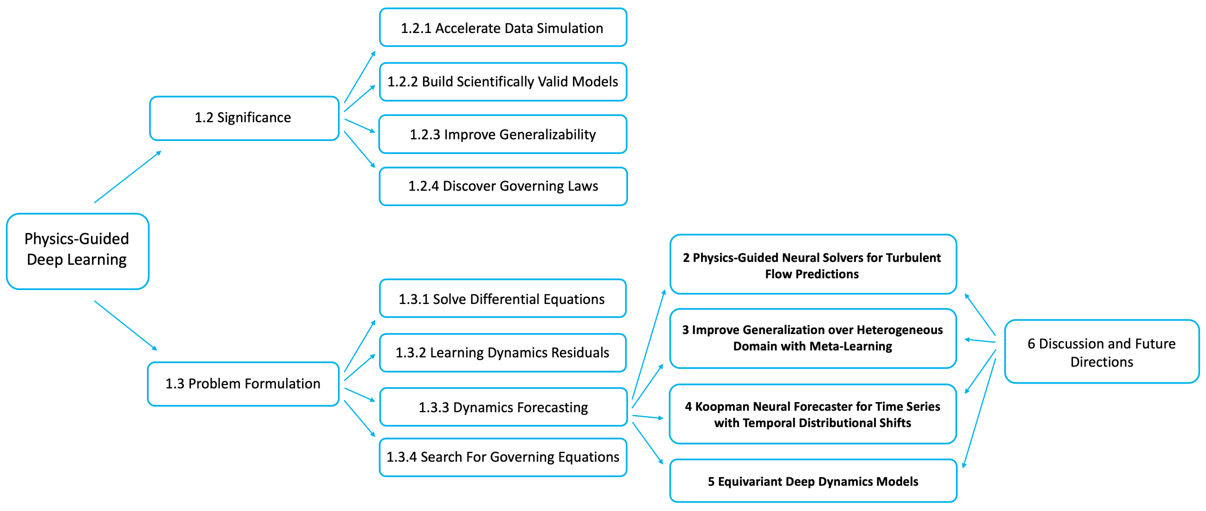
Physics-Guided Deep Learning
1.2 Significance
1.2.1 Accelerate Data Simulation
1.2.2 Build Scientifically Valid Models
1.2.3 Improve Generalizability
1.2.4 Discover Governing Laws
1.3 Problem Formulation
1.3.1 Solve Differential Equations
1.3.2 Learning Dynamics Residuals
1.3.3 Dynamics Forecasting
1.3.4 Search For Governing Equations
2 Physics-Guided Neural Solvers for Turbulent Flow Predictions
3 Improve Generalization over Heterogeneous Domain with Meta-Learning
4 Koopman Neural Forecaster for Time Series with Temporal Distributional Shifts
5 Equivariant Deep Dynamics Models
6 Discussion and Future Directions

Figure 1.2. The organization of this thesis.

1.2 Significance of Physics-Guided Deep Learning

This subsection provides an overview of the motivations and significance of physics-guided DL for learning dynamical systems. By incorporating physical principles, governing equations, mathematical modeling, and domain knowledge into DL models, the rapidly growing field of physics-guided DL can potentially (1) accelerate data simulation (2) build physically scientifically valid models (3) improve the generalizability of DL models (4) discover governing equations.

1.2.1 Accelerate Data Simulation.

Simulation is an important method of analyzing, optimizing, and designing real-world processes, which are easily verified, communicated, and understood. It serves as the surrogate modeling and digital twin and provides valuable insights into complex physical systems. Traditional physics-based simulations often rely on running numerical methods: known physical laws encoded in systems of coupled differential equations are solved over space and time via numerical differentiation and integration schemes [85, 133, 94, 151, 184]. Although the governing equations of many physical systems are known, finding approximate solutions using numerical algorithms and computers is still prohibitively expensive. Because the discretization step size is usually confined to be very small due to stability constraints when the dynamics are complex. Moreover, the performance of numerical methods can highly depend on the initial guesses of unknown parameters [92].

Recently, DL has demonstrated great success in the automation, acceleration, and streamlining of highly compute-intensive workflows for science [176, 210, 111]. Deep dynamics models can directly approximate high-dimensional spatiotemporal dynamics by directly forecasting the future states and bypassing numerical integration [226, 45, 186, 166, 186, 164, 129]. These models are trained to make forward predictions given the historic frames as input with one or more steps of supervision and can roll out up to hundreds of steps during inference. DL models are usually faster than classic numerical solvers by orders of magnitude since DL is able to take much larger space or time steps than classical solvers [166]. Another common approach is that deep neural networks can directly approximate the solution of complex coupled differential equations via gradient-based optimization, which is the so-called physics-informed neural networks (PINNs). This approach has shown success in approximating a variety of PDEs [173, 171, 29, 75]. Additionally, deep generative models, such as diffusion models and score-based models, have been shown effective in accurate molecule graph generation [67, 84].

1.2.2 Build Scientifically Valid Models.

Despite the tremendous progress of DL for science, e.g., atmospheric science [176], computational biology [2], material science [27], quantum chemistry [192], it remains a grand challenge to incorporate physical principles in a systematic manner to the design, training, and inference of such models. DL models are essentially statistical models that learn patterns from the data they are trained on. Without explicit constraints, DL models, when trained solely on data, are prone to make scientifically implausible predictions, violating the governing laws of physical systems. In many scientific applications, it is important that the predictions made by DL models are consistent with the known physical laws and constraints. For example, in fluid dynamics, a model that predicts the velocity field of a fluid must satisfy the conservation of mass and momentum. In materials science, a model that predicts the properties of a material must obey the laws of thermodynamics and the principles of quantum mechanics.

Thus, to build trustworthy predictive models for science and engineering, we need to leverage known physical principles to guide DL models to learn the correct underlying dynamics instead of simply fitting the observed data. For instance, [101, 95, 244, 72, 40] improve the physical and statistical consistency of DL models by explicitly regularising the loss function with physical constraints. Hybrid DL models, e.g., [140, 7, 31] integrate differential equations in DL for temporal dynamics forecasting and achieve promising performance. [131] and [57] studied tensor invariant neural networks that can learn the Reynolds stress tensor while preserving Galilean invariance. We will also discuss our deep surrogate model, *TF-Net*[224], that combines the numerical RANS-LES coupling method with a custom-designed U-net in detail in Chapter 2. The model uses the temporal and spatial filters in the RANS-LES coupling method to guide the U-net in learning both large and small eddies. This approach improves the both accuracy and physical consistency of the model, making it more effective at representing the complex flow phenomena observed in many fluid dynamics applications.

1.2.3 Improve the generalizability of DL models

DL models often struggle with generalization: models trained on one dataset cannot adapt properly to unseen scenarios with distributional shifts that may naturally occur in dynamical systems [114, 162, 139, 3, 225]. Because they learn to represent the statistical patterns in the training data, rather than the underlying causal relationships. In addition, most current approaches are still trained to model a specific system and multiple systems with close distributions, making it challenging to meet the needs of the scientific domain with heterogeneous environments. Thus, it is imperative to develop generalizable DL models that can learn and generalize well across systems with various parameter domains.

Prior physical knowledge can be considered as an inductive bias that can place a prior distribution on the model class and shrink the model parameter search space. With the guide of inductive bias, DL models can better capture the underlying dynamics from the data that are consistent with physical laws. Across different data domains and systems, the laws of physics stay constant. Hence, integrating physical laws in DL enables the models to generalize outside of the training domain and even to different systems. For instance, embedding symmetries into DL models is one way to improve the generalization, which we will discuss in detail in subsection 5. For example, [226] designed deep equivariant dynamics models that respect the rotation, scaling, and uniform motion symmetries in fluid dynamics. The models are both theoretically and experimentally robust to distributional shifts by symmetry group transformations and enjoy favorable sample complexity compared with data augmentation.

1.2.4 Discover Governing Equations

The discovery of governing equations is crucial as it enables us to comprehend the underlying physical laws that regulate complex systems. By identifying the mathematical models that describe the behavior of a system, we can optimize the performance of engineering systems, improve the precision of weather forecasts, and understand the mechanisms behind biological processes, among other applications [50, 18, 52, 51]. However, discovering governing equations is a challenging task for various reasons. Firstly, real-world systems are frequently complex and involve many interdependent variables, making it difficult to identify the relevant variables and their relationships. Secondly, many systems are nonlinear and involve interactions between variables that are hard to model using linear equations. Thirdly, the available data may be noisy or incomplete, making it challenging to extract meaningful patterns and relationships. Despite these challenges, recent advances in deep learning have made it possible to automate the process of governing equations discovery and identify complex, nonlinear models from data.

Discovering governing equations from data is often accomplished by defining a large set of possible mathematical basis functions and learning the coefficients. [25, 26, 98, 18, 190] proposed to find ordinary differential equations by creating a dictionary of possible basis functions and discovering sparse, low-dimensional, and nonlinear models from data using the sparse identification. More recent work, such as [122, 178], incorporated neural networks to further augment the dictionary to model more complex dynamics. [28] contributed to this trend by introducing an efficient first-order conditional gradient algorithm for solving the optimization problem of finding the best sparse fit to observational data in a large library of potential nonlinear models. Alternatively, [146, 185] presented a shallow neural network approach, *EQL* to identify concise equations from data. They replaced the activation functions with predefined basis functions, including identity and trigonometry functions, and used specially designed division units to model division relationships in the potential governing equations.

1.3 Problem Formulation

In light of the motivation and significance of physics-guided deep learning we discuss in the previous section, the primary research efforts in this field have been aimed at tackling the following four fundamental problems.

1.3.1 Solving Differential Equations

When $\mathscr{F}$ in Eq. 1.1 is *known* but Eq. 1.1 is too complicated to be solvable, researchers tend to directly solve the differential Eq.ations by approximating solution of $\boldsymbol{u}(x)$ with a deep neural network, and enforcing the governing equations as a soft constraint on the output of the neural nets during training at the same time[172, 173, 115]. This approach can be formulated as the following optimization problem,

$$\min_\theta \mathscr{L}(\boldsymbol{u}) + \lambda_{\mathscr{F}} \mathscr{L}_{\mathscr{F}}(\boldsymbol{u}) \tag{1.2}$$

$\mathscr{L}(\boldsymbol{u})$ denotes the misfit of neural net predictions and the training data points. θ denotes the neural net parameters. $\mathscr{L}_{\mathscr{F}}(\boldsymbol{u})$ is a constraint on the residual of the differential equation system under consideration and $\lambda_{\mathscr{F}}$ is a regularization parameter that controls the emphasis on this residual. The goal is then to train the neural nets to minimize the loss function in Eq. 1.2.

1.3.2 Learning Dynamics Residuals

When $\mathscr{F}$ in Eq. 1.1 is *partially know*, we can use neural nets to learn the errors or residuals made by physics-based models [44, 249, 100]. The key is to learn the bias of physics-based models and correct it with the help of deep learning. The final prediction of the state is composed of the simulation from the physics-based models and the residual prediction from neural nets as below,

$$\hat{\boldsymbol{u}} = \hat{\boldsymbol{u}}_{\mathscr{F}} + \hat{\boldsymbol{u}}_{\text{NN}}. \tag{1.3}$$

where $\hat{\boldsymbol{u}}_{\mathscr{F}}$ is the prediction obtained by numerically solving $\mathscr{F}$, $\hat{\boldsymbol{u}}_{\text{NN}}$is the prediction from neural networks and $\hat{u}$ is the final prediction made by hybrid physics-DL models.

This learning problem generally involves two training strategies: 1) joint training: optimizing the parameters in the differential equations and the neural networks at the same time by minimizing the prediction errors of the system states. 2) two-stage training: we first fit differential equations on the training data and obtain the residuals, then directly optimize the neural nets on predicting the residuals.

1.3.3 Dynamics Forecasting

When $\mathscr{F}$ in Eq. 1.1 is *unknown* or numerically solving Eq. 1.1 requires too much computation, many works studied learning high-dimensional spatiotemporal dynamics by directly predicting the input-output system state mapping and bypassing numerical discretization and integration [45, 103, 196, 226]. If we assume the first dimension x_1 of $\boldsymbol{u}$ in Eq. 1.1 is the time dimension t, then the problem of dynamics forecasting can be defined as learning a map $f : \mathbb{R}^{n\times k} \mapsto \mathbb{R}^{n\times q}$ that maps a sequence of historic states to future states of the dynamical system,

$$f\big(\boldsymbol{u}(t-k+1,\cdot),...,\boldsymbol{u}(t,\cdot)\big) = \boldsymbol{u}(t+1,\cdot),...,\boldsymbol{u}(t+q,\cdot) \tag{1.4}$$

where k is the input length and q is the output length. f is commonly approximated with purely data-driven or physics-guided neural nets and the neural nets are optimized by minimizing the prediction errors of the state $\mathscr{L}(\boldsymbol{u})$.

1.3.4 Search for Governing Equations

When $\mathscr{F}$ in Eq. 1.1 is *unknown* and it is necessary to determine the precise governing equations to solve practical problems, numerous efforts have been made to discover the exact mathematical formulation of $\mathscr{F}$. The most common approach is to select from a wide range of possible candidate functions and choose the model that minimizes fitting errors on observation data. More specifically, based on Definition 1, the goal of discovering governing equations is to find an approximate function $\hat{\mathscr{F}} = \Phi(\boldsymbol{u}(x),x)\boldsymbol{\theta} \approx \mathscr{F}$, where $\Phi(\boldsymbol{u}(x),x) = [\phi_1(\boldsymbol{u}(x),x), \phi_2(\boldsymbol{u},x),\ldots,\phi_p(\boldsymbol{u},x)]$ is a library of candidate functions, such as polynomials and trigonometric functions, and $\boldsymbol{\theta} \in \mathbb{R}^p$ is a sparse vector indicating which candidate functions are active in the dynamics. This problem can be formulated as an optimization problem, where we aim to minimize the following cost function over a set of observed data $\{\boldsymbol{y}_i\}_{i=1}^n$ of $\boldsymbol{u}$:

$$\mathscr{L}(\boldsymbol{\theta}) = \sum_{i=1}^{n} ||\Phi(\boldsymbol{y}_i, x)\boldsymbol{\theta}||^2 \tag{1.5}$$

1.4 Contributions of the Thesis

In this thesis, we focus on the task of dynamics forecasting defined in Section 1.3.3. We will first discuss how to improve the physical consistency of DL models by incorporating prior physical knowledge into the design of model architecture and loss functions. We believe the fundamental difficulty of DL for learning dynamics as neural nets have trouble coping with the distributional shifts due to the change of system parameters and data transformation across either tasks or time. To address this challenge, we present a series of works that enable DL models to generalize across diverse dynamics, temporal distributional shifts as well as symmetry transformations.

In Chapter 2, we show how to incorporate prior physical knowledge into the design of model architecture and loss functions by introducing `Turbulent Flow Net`. It unifies a Computational Fluid Dynamics (CFD) technique, RANS-LES coupling, with a custom-designed U-net to learn the multi-scale behavior of turbulence and enforces zero divergence on predicted velocity fields by a divergence-free regularizer. It not only achieves state-of-art prediction accuracy but also preserves desired physical quantities on the task of turbulent flow prediction.

In Chapter 3, we present a physics-guided model-based meta-learning method called `Dynamical Adaption Networks (DyAd)` which can generalize across heterogeneous domains by partitioning them into different tasks. `DyAd` has two parts: an encoder which infers the time-invariant hidden features of the task with weak supervision from prior knowledge, and a forecaster which learns the shared dynamics of the entire domain. The encoder adapts and controls the forecaster during inference using adaptive instance normalization and adaptive padding. `DyAd` outperforms state-of-the-art approaches to forecasting complex physical dynamics including turbulent flow, real-world sea surface temperature, and ocean currents.

In Chapter 4, we propose a novel deep sequence model based on the Koopman theory for time series forecasting: `Koopman Neural Forecaster (KNF)`. We focus on real-world time series with temporal distributional shifts (i.e. underlying dynamics changing over time). `KNF` leverages DNNs to learn the linear Koopman space and the coefficients of chosen measurement functions. It also imposes appropriate inductive biases for improved robustness against distributional shifts, employing both a global operator to learn shared characteristics, and a local operator to capture changing dynamics, as well as a specially-designed feedback loop to continuously update the learnt operators over time for rapidly varying behaviors. We demonstrate that `KNF` achieves the superior performance compared to the alternatives, on multiple time series datasets that are shown to suffer from distribution shifts.

In Chapter 5, we propose to improve accuracy and generalization by incorporating symmetries into convolutional neural networks. Specifically, we employ a variety of methods each tailored to enforce a different symmetry. Our models are both theoretically and experimentally robust to distributional shifts by symmetry group transformations and enjoy favorable sample complexity. Despite the fact that physical laws obey many symmetries, real-world dynamical data rarely conforms to strict mathematical symmetry either due to noisy or incomplete data or to symmetry-breaking features in the underlying dynamical system. Thus, we further explore approximately equivariant networks which are biased towards preserving symmetry but are not strictly constrained to do so. We demonstrate the advantage of both strictly and approximately equivariant dynamics models on a variety of physical dynamics and derive the generalization bounds for data augmentation and equivariant networks, characterizing their effect on learning in a unified framework.

In Chapter 6, we summarize the challenges in this field and discuss the emerging opportunities for future research.

Chapter 2

Physics-Guided Neural Solvers for Turbulent Flow Predictions

2.1 Introduction

2.1.1 Turbulence Modeling

In this chapter, we investigate the problem of predicting the evolution of two-dimensional spatiotemporal turbulent flow, governed by the high-dimensional non-linear Navier-Stokes equations shown below. Let $\boldsymbol{w}(t)$ be the vector velocity field of the flow with two components $(u(t), v(t))$, velocities along x and y directions,

$$\begin{aligned}
&\nabla \cdot \boldsymbol{w} = 0 && \text{Continuity Equation}\\
&\frac{\partial \boldsymbol{w}}{\partial t} + (\boldsymbol{w} \cdot \nabla)\boldsymbol{w} = -\frac{1}{\rho_0}\nabla p + \nu \nabla^2 \boldsymbol{w} + f && \text{Momentum Equation}\\
&\frac{\partial T}{\partial t} + (\boldsymbol{w} \cdot \nabla)T = \kappa \nabla^2 T && \text{Temperature Equation}
\end{aligned}$$

where p and T are pressure and temperature respectively, κ is the coefficient of heat conductivity, ρ_0 is density at temperature at the beginning, α is the coefficient of thermal expansion, ν is the kinematic viscosity, f the external force such as gravity.

In contrast to modeling sea surface temperature or lake dynamics, turbulent flows are characterized by chaotic motions and intermittency, which are difficult to predict. The temporal evolution of the turbulent flow is exceedingly sensitive to initial conditions and there is no analytical theory to characterize its dynamics. Furthermore, turbulent fluctuations occur over a wide range of length and time scales with high correlations between these scales.

Computational Fluid Dynamics (CFD) allows simulating complex turbulent flows, however, the wide range of scales makes it very challenging to accurately resolve all the scales. More precisely, fully resolving a complex turbulent flow numerically, known as direct numerical simulations (DNS), requires a very fine discretization of space-time, which makes the computation prohibitive even with advanced high-performance computing. Hence most CFD methods, like Reynolds-Averaged Navier-Stokes and Large Eddy Simulations ([151, 167, 152], resort to resolving the large scales whilst modeling the small scales, using various averaging techniques and/or low-pass filtering of the governing equations. However, the unresolved processes and their interactions with the resolved scales are extremely challenging to model.

Deep learning (DL) is poised to accelerate and improve turbulent flow simulations because well-trained DL models can generate realistic instantaneous flow fields without solving the complex nonlinear coupled PDEs that govern the system [210, 172]. For instance, deep dynamics models that directly forecast the future states and bypass numerical integration are usually faster than classic numerical solvers by orders of magnitude since DL is able to take much larger space or time steps than classical solvers [166, 186, 129]. These models are trained to make forward predictions given the historic frames as input with multiple steps of supervision and can roll out up to hundreds of steps during inference.

However, DL models are hard to train and are often used as "black boxes" as they lack knowledge of the underlying physics and are very hard to interpret. While these DL models may achieve low prediction errors they often lack scientific consistency and do not respect the physics of the systems. Therefore, it is critical to infuse known physics laws and design efficient turbulent flow prediction DL models that are not only accurate but also physically meaningful.

2.1.2 Computational Fluid Dynamics

Computational techniques are at the core of present-day turbulence investigations. Direct Numerical Simulations (DNS) are accurate but not computationally feasible for practical applications. Great emphasis was placed on the alternative approaches including Large-Eddy Simulation (LES) and Reynolds-averaged Navier–Stokes (RANS). See [151] for details.

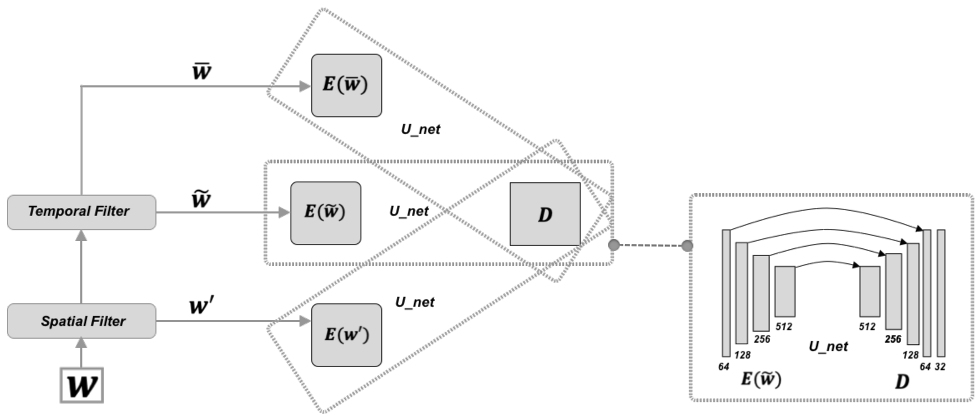

Figure 2.1. Turbulent Flow Net: three identical encoders to learn the transformations of the three components of different scales, and one shared decoder that learns the interactions among these three components to generate the predicted 2D velocity field at the next instant.

Reynolds-averaged Navier–Stokes (RANS) decomposes the turbulent flow $\boldsymbol{w}$ into two separable time scales: a time-averaged mean flow $\bar{\boldsymbol{w}}$ and a fluctuating quantity $\boldsymbol{w}'$. The resulting RANS equations contain a closure term, the Reynolds stresses, that require modeling, the classic closure problem of turbulence modeling. While this approach is a good first approximation to solving a turbulent flow, RANS does not account for broadband unsteadiness and intermittency, characteristic of most turbulent flows. Further, closure models for the unresolved scales are often inadequate, making RANS solutions to be less accurate. n here is the moving average window size.

$$\boldsymbol{w}(\boldsymbol{x},t) = \bar{\boldsymbol{w}}(\boldsymbol{x},t) + \boldsymbol{w}'(\boldsymbol{x},t), \quad \bar{\boldsymbol{w}}(\boldsymbol{x},t) = \frac{1}{n}\int_{t-n}^{t} T(s)\boldsymbol{w}(\boldsymbol{x},s)ds \tag{2.1}$$

Large Eddy Simulation (LES) is an alternative approach based on low-pass filtering of the Navier-Stokes equations that solves a part of the multi-scale turbulent flow corresponding to the most energetic scales. In LES, the large-scale component is a spatially filtered variable $\tilde{\boldsymbol{w}}$, which is usually expressed as a convolution product by the filter kernel S. The kernel S is often taken to be a Gaussian kernel. Ω_i is a subdomain of the solution and depends on the filter size [183].

$$\boldsymbol{w}(\boldsymbol{x},t) = \tilde{\boldsymbol{w}}(\boldsymbol{x},t) + \boldsymbol{w}'(\boldsymbol{x},t), \quad \tilde{\boldsymbol{w}}(\boldsymbol{x},t) = \int_{\Omega_i} S(\boldsymbol{x}|\boldsymbol{\xi})\boldsymbol{w}(\boldsymbol{\xi},t)d\boldsymbol{\xi} \tag{2.2}$$

The key difference between RANS and LES is that RANS is based on time averaging, leading to simpler steady equations, whereas LES is based on a spatial filtering process which is more accurate but also computationally more expensive.

Hybrid RANS-LES Coupling combines both RANS and LES approaches in order to take advantage of both methods [53, 30]. It decomposes the flow variables into three parts: mean flow, resolved fluctuations and unresolved (subgrid) fluctuations. RANS-LES coupling applies the spatial filtering operator S and the temporal average operator T sequentially. We can define $\bar{\boldsymbol{w}}$ in discrete form with using $\boldsymbol{w}^*$ as an intermediate term,

$$\boldsymbol{w}^*(\boldsymbol{x},\boldsymbol{t}) = S * \boldsymbol{w} = \sum_{\boldsymbol{\xi}} S(\boldsymbol{x}|\boldsymbol{\xi})\boldsymbol{w}(\boldsymbol{\xi},t) \tag{2.3}$$

$$\bar{\boldsymbol{w}}(\boldsymbol{x},\boldsymbol{t}) = T * \boldsymbol{w}^* = \frac{1}{n}\sum_{s=t-n}^{t} T(s)\boldsymbol{w}^*(\boldsymbol{x},s) \tag{2.4}$$

then $\tilde{\boldsymbol{w}}$ can be defined as the difference between $\boldsymbol{w}^*$ and $\bar{\boldsymbol{w}}$:

$$\tilde{\boldsymbol{w}} = \boldsymbol{w}^* - \bar{\boldsymbol{w}}, \quad \boldsymbol{w}' = \boldsymbol{w} - \boldsymbol{w}^* \tag{2.5}$$

Finally, we can have the three-level decomposition of the velocity field.

$$\boldsymbol{w} = \bar{\boldsymbol{w}} + \tilde{\boldsymbol{w}} + \boldsymbol{w}^{'} \tag{2.6}$$

Figure 2.2 shows this three-level decomposition in wavenumber space [53]. k is the wavenumber, the spatial frequency in the Fourier domain. $E(k)$ is the energy spectrum describing how much kinetic energy is contained in eddies with wavenumber k. Small k corresponds to large eddies that contain most of the energy. The slope of the spectrum is negative and indicates the transfer of energy from large scales of motion to small scales. This hybrid approach combines the computational efficiency of RANS with the resolving power of LES to provide a technique that is less expensive and more tractable than pure LES.

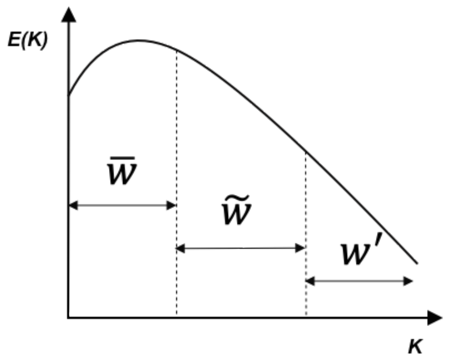

Figure 2.2. Three level spectral decomposition of velocity $\boldsymbol{w}$, $E(k)$ is the energy spectrum and k is wavenumber.

2.1.3 Related Works

In recent years, DL has been widely applied to accelerate and improve the simulation of turbulent flows. For example, [130, 57] studied tensor invariant neural networks to learn the Reynolds stress tensor while preserving Galilean invariance, but Galilean invariance only applies to flows without external forces. In our case, RBC flow has gravity as an external force. Most recently, [107] studied unsupervised generative modeling of turbulent flows but the model is not able to make real-time future predictions given the historic data. [153] proposed a purely data-driven DL model for turbulence, compressed convolutional LSTM, but the model lacks physical constraints and interpretability. [244] and [13] introduced statistical and physical constraints in the loss function to regularize the predictions of the model. However, their studies only focused on spatial modeling without temporal dynamics, besides regularization being ad-hoc and difficult to tune the hyper-parameters.

Moreover, [186] designed a deep encoder-processor-decoder graphic architecture for simulating fluid dynamics under Lagrangian description. The rich physical states are represented by graphs of interacting particles, and complex interactions are approximated by learned message-passing among nodes. [166] utilized the same architecture to learn mesh-based simulation. The authors directly construct graphs on the irregular meshes constructed in the numerical simulation methods. In addition, they proposed an adaptive re-meshing algorithm that allows the model to accurately predict dynamics at both large and small scales. [128] proposed a *Neural Operator* approach that learns the mapping between function spaces, and is invariant to different approximations and grids. More specifically, it used the message-passing graph network to learn Green's function from the data, and then the learned Green's function can be used to compute the final solution of PDEs. [129] further extended it to *Fourier Neural Operator* by replacing the kernel integral operator with a convolution operator defined in Fourier space, which is much more efficient than *Neural Operator*.

In parallel, the computer graphics community has also investigated using deep learning to speed up numerical simulations for generating realistic animations of fluids such as water and smoke. For example, [210] used an incompressible Euler's equation with a customized Convolutional Neural Network (CNN) to predict velocity updates within a finite difference method solver. [36] propose double CNN networks to synthesize high-resolution flow simulation based on reusable space-time regions. [245] and [211] developed deep learning models in the context of fluid flow animation, where physical consistency is less critical. [203] proposed a method for the data-driven inference of temporal evolutions of physical functions with deep learning. However, fluid animation emphasizes on the realism of the simulation rather than the physical consistency of the predictions or physics metrics and diagnostics of relevance to scientists.

The physical system we investigate in this work is two-dimensional Rayleigh-Bénard convection (RBC), a model for turbulent convection, with a horizontal layer of fluid heated from below so that the lower surface is at a higher temperature than the upper surface. Turbulent convection is a major feature of the dynamics of the oceans, and the atmosphere, as well as engineering and industrial processes, which has motivated numerous experimental and theoretical studies for many years. The RBC system serves as an idealized model for turbulent convection

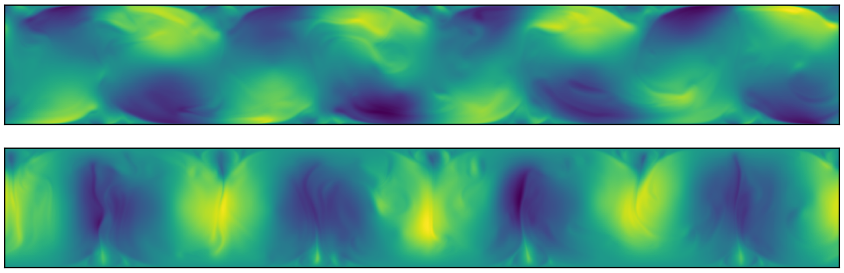

Figure 2.3. A snapshot of the Rayleigh-Bénard convection flow, the velocity fields along x direction (top) and y direction (bottom) [35]. The spatial resolution is 1792 x 256 pixels.

that exhibits the full range of dynamics of turbulent convection for sufficiently large temperature gradients. Figure 2.3 shows a snapshot in our RBC flow dataset.

We aim to infuse CFD principles into deep neural networks so that the physics-guided model can correctly model the multi-scale behaviors of turbulence and preserve desired physical properties. The global idea behind our method is to decompose the turbulent flow into components of different scales with trainable modules for simulating each component. First, we provide a brief introduction of the CFD techniques which are built on this basic idea.

2.2 Turbulent Flow Net

Inspired by techniques used in hybrid RANS-LES Coupling to separate scales of a multi-scale system, we propose a hybrid DL framework, `Turbulent Flow Net (TF-Net)`, based on the multi-level spectral decomposition of the turbulent flow.

Specifically, we decompose the velocity field into three components of different scales using two scale separation operators, the spatial filter S and the temporal filter T. In traditional CFD, these filters are usually pre-defined, such as the Gaussian spatial filter. In `Turbulent Flow Net (TF-Net)`, both filters are *trainable* neural networks. The spatial filtering process is instantiated as one layer convolutional neural network with a single 5×5 filter to each input image. The temporal filter is also implemented as a convolutional layer with a single 1×1 filter applied to every T image. The motivation for this design is to explicitly guide the DL model to learn the non-linear dynamics of both large and small eddies as relevant to the task of spatiotemporal prediction.

We design three identical encoders to encode the three scale components separately. We use a shared decoder to learn the interactions among these three components and generate the final prediction. Each encoder and the decoder can be viewed as a U-net without duplicate layers and middle layers in the original architecture [159]. The encoder consists of four convolutional layers with double the number of feature channels of the previous layer and stride 2 for down-sampling. The decoder consists of one output layer and four deconvolutional layers with a summation of the corresponding feature channels from the three encoders and the output of the previous layer as input. Figure 2.1 shows the overall architecture of our hybrid model `TF-net`.

To generate multi-step forecasts, we perform one-step ahead prediction and roll out the predictions autoregressively. Furthermore, we also consider a variant of `TF-net` by explicitly adding a physical constraint to the loss function. Since the turbulent flow under investigation has zero divergences ($\nabla \cdot \boldsymbol{w}$ should be zero everywhere), we include $||\nabla \cdot \boldsymbol{w}||^2$ as a regularizer to constrain the predictions, leading to a constrained `TF-net`, or `Con TF-Net`.

2.3 Experiments Setup

2.3.1 Dataset

The dataset for our experiments comes from two-dimensional turbulent flow simulated using the Lattice Boltzmann Method [35]. We use only the velocity vector fields, where the spatial resolution of each image (snapshots in time) is 1792 x 256. Each image has two channels, one is the turbulent flow velocity along x direction and the other one is the velocity along y direction. The physics parameters relevant to this numerical simulation are: Prandtl number $= 0.71$, Rayleigh number $= 2.5 \times 10^8$ and the maximum Mach number $= 0.1$. We use 1500 images for our experiments. The task is to predict the spatiotemporal velocity fields up to 60 steps ahead given 10 initial frames.

We divided each 1792 by 256 image into 7 square sub-regions of size 256 x 256, then

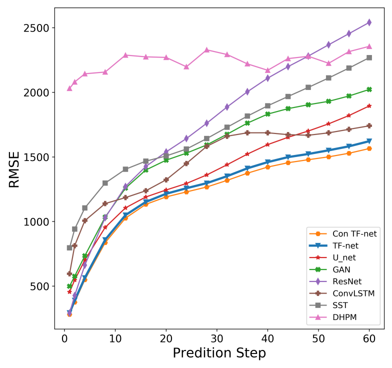

Figure 2.4. Root mean square errors of different models' predictions at varying forecasting horizon

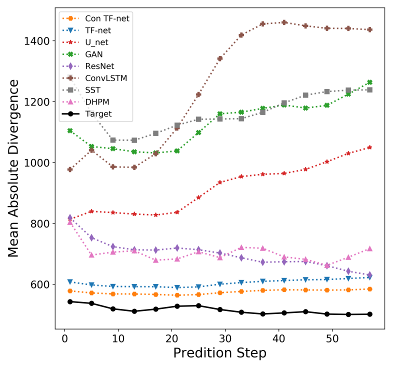

Figure 2.5. Mean absolute divergence of different models' predictions at varying forecasting horizon

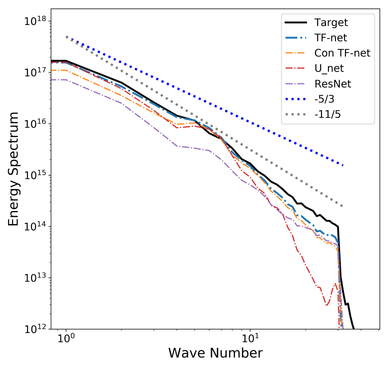

Figure 2.6. The Energy Spectrum of `TF-net`, `U-net` and `ResNet` on the leftmost square sub-region.

downsample them into 64 x 64 pixels sized images. We use a sliding window approach to generate 9,870 samples of sequences of velocity fields: 6,000 training samples, 1,700 validation samples and 2,170 test samples. The DL model is trained using back-propagation through prediction errors accumulated over multiple steps. We use a validation set for hyper-parameters tuning based on the average error of predictions up to six steps ahead. All results are averaged over three runs with random initialization.

2.3.2 Baselines

We compare our model with a series of state-of-the-art baselines, including pure data-driven video predictions models, `ResNet` [79], `ConvLSTM` [197], `U-Net` [159] and `GAN`, as well as hybrid physics-informed DL, `SST` [45] and `DHPM` [170].

2.3.3 Evaluation Metrics

Even though Root Mean Square Error (RMSE) is a widely accepted metric for quantifying the prediction performance, it only measures pixel differences. We need to check whether the predictions are physically meaningful and preserve desired physical quantities, such as Turbulence Kinetic Energy, Divergence and Energy Spectrum. Therefore, we include a set of additional metrics for evaluation.

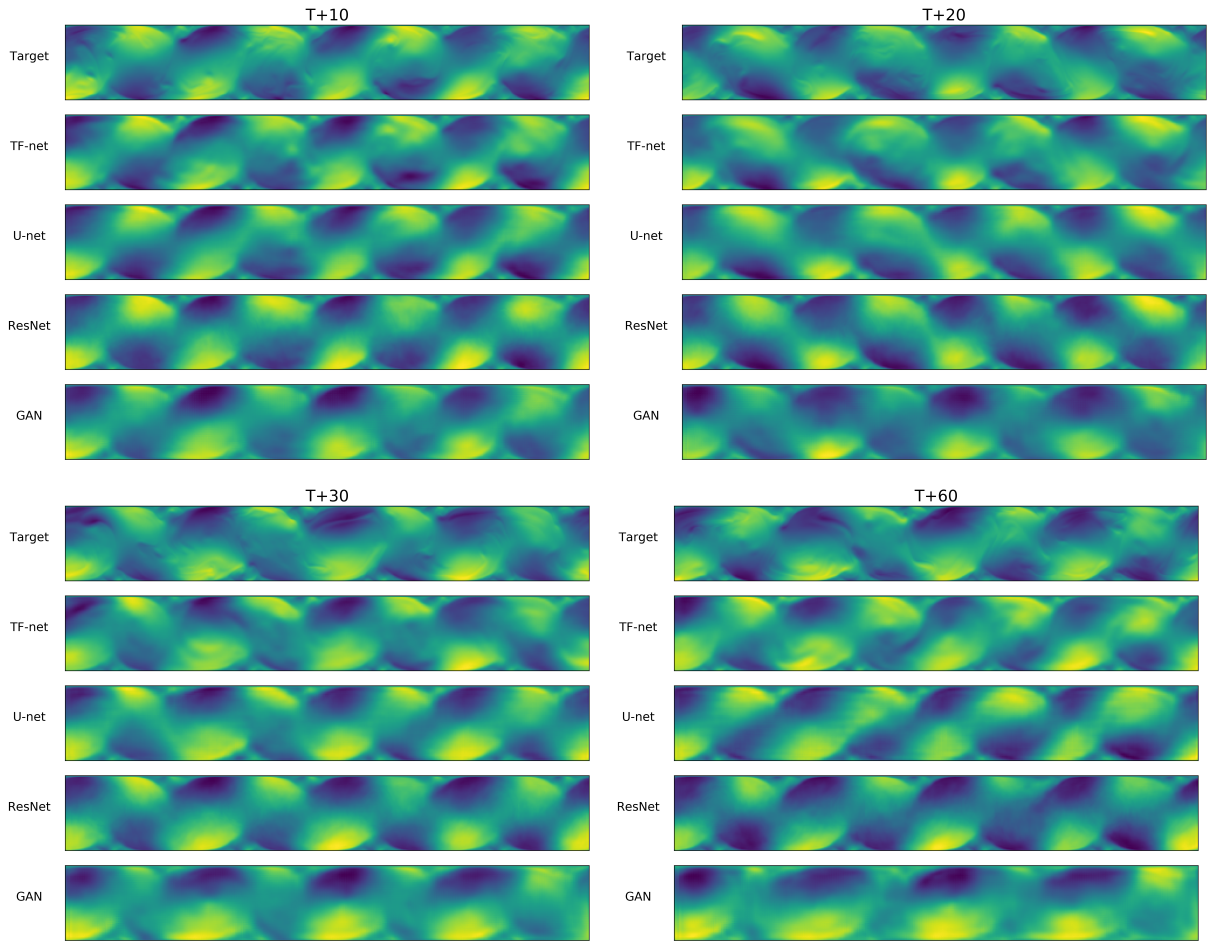

Figure 2.7. Ground truth (target) and predicted u velocities by `TF-net` and three best baselines (U-Net, ResNet and GAN) at time $T+10$, $T+20$, $T+30$ to $T+60$ (suppose T is the time step of the last input frame).

Root Mean Square Error: We calculate the RMSE of all predicted values from the ground truth for each pixel.

Divergence: Since we investigate incompressible turbulent flows in this work, which means the divergence, $\nabla \cdot \mathbf{w}$, at each pixel should be zero, we use the average of absolute divergence over all pixels at each prediction step as an additional evaluation metric.

Turbulence Kinetic Energy: In fluid dynamics, turbulence kinetic energy is the mean kinetic energy per unit mass associated with eddies. Physically, the turbulence kinetic energy is characterized by measured root mean square velocity fluctuations,

$$((\overline{(u')^2} + \overline{(v')^2})/2, \quad \overline{(u')^2} = \frac{1}{T}\sum_{t=0}^{T}(u(t) - \bar{u})^2 \tag{2.7}$$

where t is the time step, u and v represent the velocity along x and y directions. We calculate the turbulence kinetic energy for each predicted sample of 60 velocity fields.

Energy Spectrum: The energy spectrum of turbulence, $E(k)$, is related to the mean turbulence kinetic energy as

$$\int_0^\infty E(k)dk = (\overline{(u')^2} + \overline{(v')^2}), \tag{2.8}$$

where k is the wavenumber, the spatial frequency in the 2D Fourier domain. We calculate the Energy Spectrum on the Fourier transformation of the Turbulence Kinetic Energy fields. The large eddies have low wavenumbers and the small eddies correspond to high wavenumbers. The spectrum indicates how much kinetic energy is contained in eddies with wavenumber k.

2.4 Results on the Rayleigh-Bénard Convection Dataset

2.4.1 Accuracy

Figure 2.4 shows the growth of RMSE with a prediction horizon up to 60 time steps ahead. `TF-net` consistently outperforms all baselines, and constraining it with a divergence-free regularizer can further improve the performance. We also found `DHPM` is able to overfit the training set but performs poorly when tested outside of the training domain. Neither Dropout nor regularization techniques can improve its performance. Also, the warping scheme of the [45] relies on the simplified linear assumption, which was too limiting for our non-linear problem.

During Inference, we apply the trained `TF-net` to the entire input domain instead of square sub-regions. Figure 2.7 shows the ground truth and the predicted u velocity fields from all models from time step 0 to 60. We see that the predictions by our `TF-net` model are the

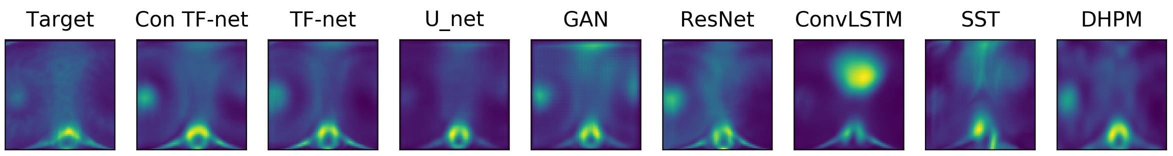

Figure 2.8. Turbulence kinetic energy of all models' predictions at the leftmost square field in the original rectangular field

closest to the target based on the shape and the frequency of the motions. Baselines generate smooth predictions and miss the details of the small-scale motion. `U-net` is the best-performing data-driven video prediction baseline. [209] also found the U-net architecture is quite effective in modeling dynamics flows. Nevertheless, there is still room for improvement in long-term prediction for all the models.

2.4.2 Physical Consistency

Figure 2.5 shows the averages of absolute divergence over all pixels at each prediction step. `TF-net` has lower divergence than other models even without additional divergence-free constraints for varying prediction steps. It is worth mentioning that there is a subtle trade-off between RMSE and divergence. Even though constraining the model with the divergence-free regularizer can reduce the divergence of the model predictions, too much constraint also has the side effect of smoothing out the small eddies, which results in a larger RMSE.

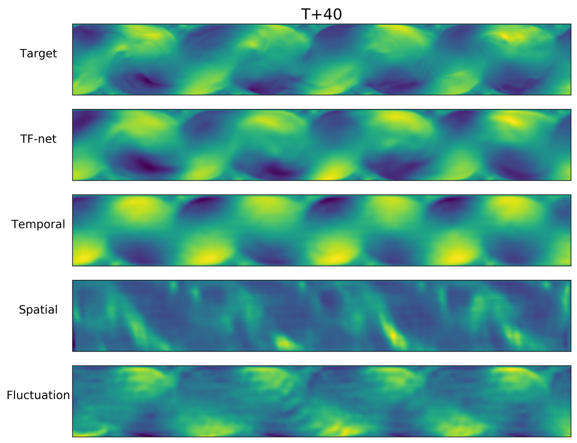

Figure 2.9. Ablation study: From top to bottom are the target, the predictions of `TF-net`, and the outputs of each small U-net of `TF-net` while the other two encoders are zeroed out at time $T+40$

Figure 2.8 displays predicted TKEs of all models at the leftmost square field in the original rectangular field. Figure 2.6 shows the energy spectrum of our model and two best baseline at the leftmost square sub-field. While the turbulence kinetic energy of `TF-net`, `U-net` and `ResNet` appear to be similar in Figure 2.8, from the energy spectrum in Figure 2.6, we can see that `TF-net` predictions are in fact much closer to the target. Extra divergence-free constraint does not affect the energy spectrum of predictions. By Incorporating physics principles in DL, `TF-net` is able to generate predictions that are physically consistent with the ground truth.

2.4.3 Ablation Study

We also perform an additional ablation study of `TF-net` to understand each component of `TF-net` and investigate whether the `TF-net` has actually learned the flow with different scales. During inference, we applied each small U-net in `TF-net` with the other two encoders removed to the entire input domain. Figure 2.9 includes the predictions of `TF-net`, and the outputs of each small U-net while the other two encoders are zeroed out at $T+40$. We observe that the outputs of each small u-net are the flow with different scales, which demonstrates that `TF-net` can learn multi-scale behaviors of turbulent flows.

2.5 Conclusion

We presented a novel hybrid DL model, `TF-net`, that unifies representation learning and turbulence simulation techniques. `TF-net` exploits the multi-scale behavior of turbulent flows to design trainable scale-separation operators to model different ranges of scales individually. We provide exhaustive comparisons of `TF-net` and baselines and observe significant improvement in both the prediction error and desired physical quantifies, including divergence, turbulence kinetic energy, and energy spectrum. We argue that different evaluation metrics are necessary to evaluate a DL model's prediction performance for physical systems that include both accuracy and physical consistency. A key contribution of this work is the combination of state-of-the-art turbulent flow simulation paradigms with DL.

However, this work and many related works only focus on a specific system and train on past data in order to predict the future. These models usually fail to generalize new systems with different parameters and boundary conditions and a new model must be trained to predict a system with different dynamics. Therefore, it is necessary to develop generalizable models for turbulence that can learn and predict well over a large heterogeneous domain. We will present our framework, `Dynamical Adaption Networks`, that enable deep dynamics models to generalize across diverse dynamics in the next chapter.

Chapter 3

Improving Generalization over Heterogeneous Domain with Meta-Learning

3.1 Introduction

3.1.1 Generalization Issues in Learning Dynamics

Modeling dynamical systems with DL has shown great success in a wide range of systems from climate science, Internet of Things to infectious diseases [90, 211, 33, 129, 149, 88]. However, the main limitation of previous works is limited generalizability. Most approaches only focus on a specific system and train on past data in order to predict the future. Thus, a new model must be trained to predict a system with different dynamics. Consider, for example, learning fluid dynamics; shown in Fig. 3.1 are two fluid flows with different degrees of turbulence. Although the flows are governed by the same equations, the difference in buoyant forces would require two separate DL models to forecast. Therefore, it is imperative to develop *generalizable* DL models for dynamical systems that can learn and predict well over a large heterogeneous domain.

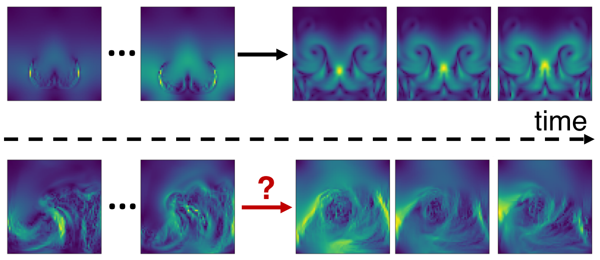

Figure 3.1. Meta-learning dynamic forecasting on turbulent flow. The model needs to generalize to a flow with a very different buoyant force.

Meta-learning [208, 11, 58], or learning to learn, improves generalization by learning multiple tasks from the environment. Recent developments in meta-learning have been successfully applied to few-shot classification [156, 206], active learning [250], and reinforcement learning [74, 86]. However, meta-learning in the context of forecasting high-dimensional physical dynamics has not been studied before. The challenges with meta-learning dynamical systems are unique in that (1) we need to efficiently infer the latent representation of the dynamical system given observed time series data, (2) we need to account for changes in unknown initial and boundary conditions, and (3) we need to model temporal dynamics across heterogeneous domains.

Our approach is inspired by the fact that similar dynamical systems may share time-invariant hidden features. Even the slightest change in these features may lead to vastly different phenomena. For example, in climate science, fluids are governed by a set of differential equations called the Navier-Stokes equations. Some features such as kinematic viscosity and external forces (e.g. gravity), are time-invariant and determine the flow characteristics. By inferring this latent representation, we can model diverse system behaviors from smoothly flowing water to atmospheric turbulence.

Inspired by neural style transfer [102], we propose a model-based meta-learning method, called `DyAd`, which can rapidly adapt to systems with varying dynamics. `DyAd` has two parts, an encoder g and a forecaster f. The encoder maps different dynamical systems to time-invariant hidden features representing constants of motion, boundary conditions, and external forces which characterize the system. The forecaster f then takes the learned hidden features from the encoder and the past system states to forecast the future system state. Controlled by the time-invariant hidden features, the forecaster has the flexibility to adapt to a wide range of systems with heterogeneous dynamics.

Unlike gradient-based meta-learning techniques such as MAML [58], `DyAd` automatically adapts during inference using an encoder and does not require retraining. Similar to model-based meta-learning methods such as MetaNets [156], we employ a two-part design with an adaptable learner which receives task-specific weights. However, for time-series forecasting, since input and output come from the same domain, a support set of labeled data is unnecessary to define the task. The encoder can infer the task directly from the query input.

3.1.2 Meta-learning in Dynamics Forecasting

Let $x \in \mathbb{R}^d$ be a d-dimensional state of a dynamical system governed by parameters ψ. The problem of dynamics forecasting is that given a sequence of past states $(\mathbf{x}_1, \ldots, \mathbf{x}_t)$, we want to learn a map f such that: $f : (\mathbf{x}_1, \ldots, \mathbf{x}_t) \longmapsto (\mathbf{x}_{t+1}, \ldots \mathbf{x}_{t+k})$.

Here l is the length of the input series, and k is the forecasting horizon in the output. Existing approaches for dynamics forecasting only predict future data for a specific system as a single task. Here a task refers to forecasting for a specific system with a given set of parameters. The resulting models often generalize poorly to different system dynamics. Thus a new model must be trained to predict for each specific system.

To perform meta-learning, we identify each forecasting task by some parameters $c \subset \psi$, such as constants of motion, external forces, and boundary conditions. We learn multiple tasks simultaneously and infer the task from data. Here we use c for a subset of system parameters ψ, because we usually do not have the full knowledge of the system dynamics. In the turbulent flow example, the state $\mathbf{x}_t$ is the velocity field at time t. Parameters c can represent Reynolds number, mean vorticity, mean magnitude, or a vector of all three.

Let μ be the data distribution over $\mathscr{X} \times \mathscr{Y}$ representing the function $f\colon \mathscr{X} \to \mathscr{Y}$ where $\mathscr{X} = \mathbb{R}^{d \times t}$ and $\mathscr{Y} = \mathbb{R}^{d \times k}$. Our main assumption is that the domain $\mathscr{X}$ can be partitioned into separate tasks $\mathscr{X} = \cup_{c \in \mathscr{C}} \mathscr{X}_c$, where $\mathscr{X}_c$ is the domain for task c and $\mathscr{C}$ is the set of all tasks. The data in the same task share the same set of parameters. Let μ_c be the conditional distribution over $\mathscr{X}_c \times \mathscr{Y}$ for task c.

During training, the model is presented with data drawn from a subset of tasks $\{(x,y) : (x,y) \sim \mu_c, c \sim C\}$. Our goal is to learn the function $f\colon \mathscr{X} \to \mathscr{Y}$ over the whole domain $\mathscr{X}$ which can thus generalize across all tasks $c \in \mathscr{C}$. To do so, we need to learn the map $g\colon \mathscr{X} \to \mathscr{C}$

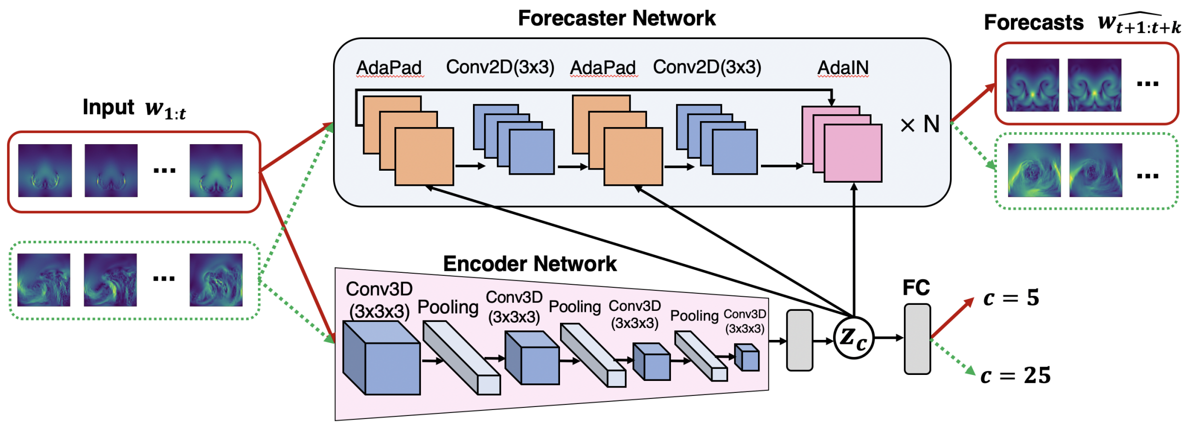

Figure 3.2. Overview of `DyAd` applied to two inputs of fluid turbulence, one with small external forces and one with larger external forces. The encoder infers the time-shift invariant characteristic variable z which is used to adapt the forecaster network.

taking $x \in \mathscr{X}_c$ to c in order to infer the task with minimal supervision.

3.1.3 Related Work

Multi-task learning [216] focuses on learning shared representations from multiple related tasks. Architecture-based MTL methods can be categorized into encoder-focused [135] and decoder-focused [246]. There are also optimization-based MTL methods, such as task balancing methods [104]. But MTL assumes tasks are known a priori instead of inferring the task from data. On the other hand, the aim of meta-learning [208] is to leverage the shared representation to fast adapt to unseen tasks. Based on how the meta-level knowledge is extracted and used, meta-learning methods are classified into model-based [156, 1, 160, 195, 256], metric-based [218, 201] and gradient-based [58, 181, 71, 247]. Most meta-learning approaches are not designed for forecasting with a few exceptions. [237] proposed to train a domain classifier with weak supervision to help domain adaptation but focused on low-dimensioal time series classification problems. [160] designed a residual architecture for time series forecasting with a meta-learning parallel. [1] proposed a modular meta-learning approach for continuous control. But forecasting physical dynamics poses unique challenges to meta-learning as it requires encoding physical knowledge into our model.

Our approach is inspired by neural style transfer techniques. In style transfer, a generative network is controlled by an external style vector through adaptive instance normalization between convolutional layers. Our hidden representation bears affinity with the "style" vector in style transfer techniques. Rather than aesthetic style in images, our hidden representation encodes time-invariant features. Style transfer initially appear in non-photorealistic rendering [120]. Recently, neural style transfer [97] has been applied to image synthesis [65, 136], videos generation [177], and language translation [168]. For dynamical systems, [187] adapts texture synthesis to transfer the style of turbulence for animation. [107] studies unsupervised generative modeling of turbulent flows but for super-resolution reconstruction rather than forecasting.

3.2 Dynamic Adaptation Network

We propose a model-based meta-learning approach for dynamics forecasting. Given multiple forecasting tasks, we propose to learn the function f in two stages. That is, by first inferring the task c from the input x, and then adapting to a specialized forecaster $f_c \colon \mathscr{X}_c \to \mathscr{Y}$ for each task. An alternative is to use a single deep neural network to directly model f in one step over the whole domain. But this requires the training set to have good and uniform coverage of the different tasks. If the data distribution μ is highly heterogeneous or the training set is not sampled i.i.d. from the whole domain, then a single model may struggle with generalization.

We hypothesize that by partitioning the domain into different tasks, the model would learn to pick up task-specific features without requiring uniform coverage of the training data. Furthermore, by separating task inference and forecasting into two stages, we allow the forecaster to rapidly adapt to new tasks that never appeared in the training set.

As shown in Fig. 3.2, our model consists of two parts: an encoder g and a forecaster f. We introduce z_c as a time-invariant hidden feature for task c. We assume that c depends linearly on the hidden feature for simplicity and easy interpretation. We design the encoder to infer the hidden feature z_c given the input x. We then use z_c to adapt the forecaster f to the specific task, i.e., model $y = f_c(x)$ as $y = f(x, z_c)$. As the system dynamics are encoded in the input sequence x, we can feed the same input sequence x to a forecaster and generate predictions $\hat{y} = f_c(x)$.

3.2.1 Encoder Network

The encoder maps the input x to the hidden features z_c that are time-invariant. To enforce this inductive bias, we encode time-invariance both in the architecture and in the training objective.

Time-Invariant Encoder. The encoder is implemented using 4 Conv 3D layers, each followed by `BatchNorm`, `LeakyReLU`, and max-pooling. Note that theoretically, max-pooling is not perfectly shift invariant since 2x2x2 max-pooling is equivariant to shifts of size 2 and only approximately invariant to shifts of size 1. But standard convolutional architectures often include max-pooling layers to boost performance. We convolve both across spatial and temporal dimensions. After that, we use a global mean-pooling layer and a fully connected layer to estimate the hidden feature $\hat{z}_c$. The task parameter depends linearly on the hidden feature. We use a fully connected layer to compute the parameter estimate $\hat{c}$.

Since convolutions are equivariant to shift (up to boundary frames) and mean pooling is invariant to shift, the encoder is shift-invariant. In practice, shifting the time sequence one frame forward will add one new frame at the beginning and drop one frame at the end. This creates some change in output value of the encoder. Thus, practically, the encoder is only approximately shift-invariant.

Encoder Training. The encoder network g is trained first. To combat the loss of shift invariance from the change from the boundary frames, we train the encoder using a time-invariant loss. Given two training samples $(x^{(i)}, y^{(i)})$ and $(x^{(j)}, y^{(j)})$ and their task parameters c, we have loss

$$\mathscr{L}_{\texttt{enc}} = \sum_{c \sim \mathscr{C}} \|\hat{c} - c\|^2 + \alpha \sum_{i,j,c} \|\hat{z}_c^{(i)} - \hat{z}_c^{(j)}\|^2 + \beta \sum_{i,c} \| \|\hat{z}_c^{(i)}\|^2 - m \|^2 \tag{3.1}$$

where $\hat{z}^{(i)} = g(x^{(i)})$ and $\hat{z}^{(j)} = g(x^{(j)})$ and $\hat{c}^{(i)} = W\hat{z}_c^{(i)} + b$ is an affine transformation of z_c.

The first term $||\hat{c}-c||^2$ uses weak supervision of the task parameters whenever they are available. Such weak supervision helps guide the learning of hidden feature z_c for each task. While not all parameters of the dynamical system are known, we can compute approximate values in the datum $c^{(i)}$ based on our domain knowledge. For example, instead of the Reynolds number of the fluid flow, we can use the average vorticity as a surrogate for task parameters.

The second term $||\hat{z}_c^{(i)} - \hat{z}_c^{(j)}||^2$ is the time-shift invariance loss, which penalizes the changes in latent variables between samples from different time steps. Since the time-shift invariance of convolution is only approximate, this loss term drives the time-shift error even lower. The third term $|||\hat{z}_c^{(i)}|| - m|^2$ (m is a positive value) prevents the encoder from generating small $\hat{z}_c^{(i)}$ due to time-shift invariance loss. It also helps the encoder to learn more interesting z, even in the absence of weak supervision.

Hidden Features. The encoder learns time-invariant hidden features. These hidden features resemble the time-invariant dimensionless parameters [116] in physical modeling, such as Reynolds number in fluid mechanics. The hidden features may also be viewed as partial disentanglement of the system state. As suggested by [138, 158], our disentanglement method is guided by inductive bias and training objectives. Unlike complete disentanglement, as in e.g. [148], in which the latent representation is factored into time-invariant and time-varying components, we focus only on time-shift-invariance. Nonetheless, the hidden features can control the forecaster which is useful for generalization.

3.2.2 Forecaster Network.

The forecaster incorporates the hidden feature z_c from the encoder and adapts to the specific forecasting task $f_c = f(\cdot, z_c)$. In what follows, we use z for z_c. We use two specialized layers, adaptive instance normalization (AdaIN) and adaptive padding (AdaPad). AdaIN has been used in neural style transfer [102, 87] to control generative networks. Here, AdaIN may adapt for specific coefficients and external forces. We also introduce a new layer, AdaPad(x,z), which is designed for encoding the boundary conditions of dynamical systems. The backbone of the forecaster network can be any sequence prediction model. We use a design that is similar to `ResNet` for spatiotemporal sequences.

AdaIN. We employ AdaIN to adapt the forecaster network. Denote the channels of input x by x_i and let $\mu(x_i)$ and $\sigma(x_i)$ be the mean and standard deviation of channel i. For each AdaIN layer, a particular style is computed $s = (\mu_i, \sigma_i)_i = Az + b$, where the linear map A and bias b are learned weights. Adaptive instance normalization is then defined as $y_i = \sigma_i \frac{x_i - \mu(x_i)}{\sigma(x_i)} + \mu_i$. In essence, the channels are renormalized to the style s.

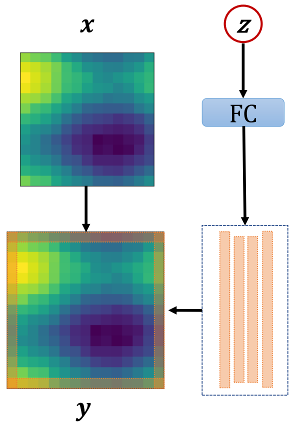

Figure 3.3. Illustration of the AdaPad operation.

For dynamics forecasting, the hidden feature z encodes data analogous to the coefficients of a differential equation and external forces on the system. In numerical simulation of a differential equation these coefficients enter as scalings of different terms in the equation and the external forces are added to the combined force equation [94]. Thus in our context AdaIN, which scales channels and adds a global vector, is well-suited to injecting this information.

AdaPad. To complement AdaIN, we introduce AdaPad, which encodes the boundary conditions of each specific dynamical system. Generally when predicting dynamical systems, error is introduced along the boundaries since it is unknown how the dynamics interact with the boundary of the domain, and there may be unknown inflows or outflows. In our method, the inferred hidden feature z may contain the boundary information. AdaPad uses the hidden features to compute the boundary conditions via a linear layer. Then it applies the boundary conditions as padding immediately outside the spatial domain in each layer, as shown in Fig. 3.3.

Forecaster Training The forecaster is trained separately after the encoder. The kernels of the convolutions and the mappings of the AdaIN and AdaPad layers are all trained simultaneously as the forecaster network is trained. Denoting the true state as y and the predicted state as $\hat{y}$, we compute the loss per time step $||\hat{y} - y||^2$ for each example. We accumulate the loss over different time steps and generate multi-step forecasts in an autoregressive fashion. In practice, we observe separate training achieves better performance than training end-to-end; see the experiments setup section for details.

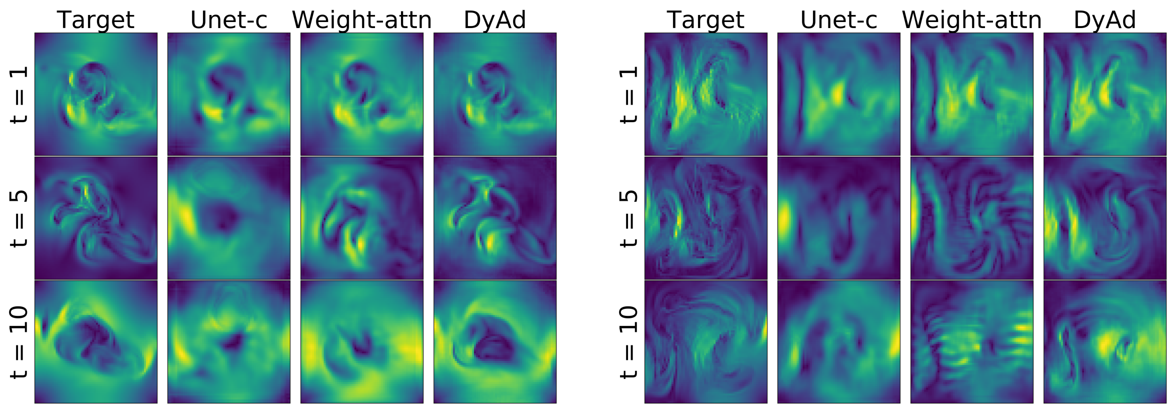

Figure 3.4. Target and predictions by `Unet-c`, `Modular-wt` and `DyAd` at time 1, 5, 10 for turbulent flows with buoyancy factors 9 (left) and 21 (right) respectively. `DyAd` can easily generate predictions for various flows while baselines have trouble understanding and disentangling buoyancy factors.

Table 3.1. Prediction RMSE on the turbulent flow and sea surface temperature datasets. Prediction RMSE and ESE (energy spectrum errors) on the future and domain test sets of ocean currents dataset.

Model	Turbulent Flows		Sea Temperature		Ocean Currents	
	future	domain	future	domain	future	domain
`ResNet`	0.94±0.10	0.65±0.02	0.73±0.14	0.71±0.16	9.44±1.55 | 0.99±0.15	9.65±0.16 | 0.90±0.16
`ResNet-c`	0.88±0.03	0.64±0.01	0.70±0.08	0.71±0.06	9.71±0.01 | 0.81±0.03	9.15±0.01 | 0.73±0.03
`U-Net`	0.92±0.02	0.68±0.02	0.57±0.05	0.63±0.05	7.64±0.05 | 0.83±0.02	7.61±0.14 | 0.86±0.03
`Unet-c`	0.86±0.07	0.68±0.03	0.47±0.02	0.45±0.06	**7.26±0.01** | 0.94±0.02	7.51±0.03 | 0.87±0.04
`PredRNN`	0.75±0.02	0.75±0.01	0.67±0.12	0.99±0.07	8.49±0.01 | 1.27±0.02	8.99±0.03 | 1.69±0.01
`VarSepNet`	0.67±0.05	0.63±0.06	0.63±0.14	0.49±0.09	9.36±0.02 | 0.63±0.04	7.10±0.01 | 0.58±0.02
`Mod-attn`	0.63±0.12	0.92±0.03	0.89±0.22	0.98±0.17	8.08±0.07 | 0.76±0.11	8.31±0.19 | 0.88±0.14
`Mod-wt`	0.58±0.03	0.60±0.07	0.65±0.08	0.64±0.09	10.1±0.12 | 1.19±0.72	8.11±0.19 | 0.82±0.19
`MetaNet`	0.76±0.13	0.76±0.08	0.84±0.16	0.82±0.09	10.9±0.52 | 1.15±0.18	11.2±0.16 | 1.08±0.21
`MAML`	0.63±0.01	0.68±0.02	0.90±0.17	0.67±0.04	10.1±0.21 | 0.85±0.06	10.9±0.79 | 0.99±0.14
`DyAd+ResNet`	**0.42±0.01**	**0.51±0.02**	0.42±0.03	0.44±0.04	7.28±0.09 | **0.58±0.02**	**7.04±0.04** | **0.54±0.03**
`DyAd+Unet`	0.58±0.01	0.59±0.01	**0.35±0.03**	**0.42±0.05**	7.38±0.01 | 0.70±0.04	7.46±0.02 | 0.70±0.07

3.3 Experiments Setup

3.3.1 Datasets

We experiment on three datasets: synthetic turbulent flows, real-world sea temperature, and ocean currents. These are difficult to forecast using numerical methods due to unknown external forces and complex dynamics not fully captured by simplified mathematical models.

Turbulent Flow with Varying Buoyancy: We generate a synthetic dataset of turbulent flows with a numerical simulator, PhiFlow[1]. It contains 64$\times$64 velocity fields of turbulent flows in which we vary the buoyant force acting on the fluid from 1 to 25. Each buoyant force corresponds to a forecasting task and there are 25 tasks in total. We use the mean vorticity of each task as partial supervision c as we can directly calculate it from the data. Vorticity can characterize formation and circular motion of turbulent flows.

Sea Surface Temperature: We evaluate on a real-world sea surface temperature data generated by the NEMO ocean engine [143][2]. We select an area from Pacific ocean range from 01/01/2018 to 12/31/2020. The corresponding latitude and longitude are (-150$\sim$-120, -20$\sim$-50). This area is then divided into 25 64$\times$64 subregions, each is a task since the mean temperature varies a lot along longitude and latitude. For the encoder training, we use season as an additional supervision signal besides the mean temperature of each subregion. In other words, the encoder should be able to infer the mean temperature of the subregion as well as to classify four seasons given the temperature series.

Ocean Currents: We also experiment with the velocity fields of ocean currents from the same region and use the same task division as the sea surface temperature data set. Similar to the turbulent flow data set, we use the mean vorticity as well as the location of each subregion as the weak-supervision signal.

For all datasets, we use a sliding-window approach to generate samples of sequences. We evaluate on two generalization scenarios. For test-future, we train and test on the same task but different time steps. For test-domain, we train and test on different tasks with an 80-20 split. All models are trained to make next-step predictions given the history. We forecast in an autoregressive manner to generate multi-step ahead predictions. All results are averaged over 3 runs with random initialization. Apart from the root mean square error (RMSE), we also report

[1]https://github.com/tum-pbs/PhiFlow

[2]The data are available at https://resources.marine.copernicus.eu/?option=com_csw&view=details&product_id=GLOBAL_ANALYSIS_FORECAST_PHY_001_024

the energy spectrum error (ESE) for ocean current prediction, which quantifies the physical consistency. ESE indicates whether the predictions preserve the correct statistical distribution and obey the energy conservation law, which is a critical metric for physical consistency.

3.3.2 Baselines

We include several SoTA baselines from meta-learning and dynamics forecasting.

- `ResNet` [79]: A widely adopted video prediction model [226].
- `U-net` [159]: Originally developed for biomedical image segmentation, adapted for dynamics forecasting [45].
- `ResNet/Unet-c`: Above `ResNet` and `Unet` with an additional final layer that generates task parameter c and trained with weak-supervision and forecasting loss altogether.
- `PredRNN` [230]: A state-of-art RNN-based spatiotemporal forecasting model.
- `VarSepNet` [49]: A convolutional dynamics forecasting model based on spatiotemporal disentanglement.
- `Mod-attn`[1]: A modular meta-learning method which combines the outputs of modules to generalize to new tasks using attention.
- `Mod-wt`: A modular meta-learning variant which uses attention weights to combine the parameters of the convolutional kernels in different modules for new tasks.
- `MetaNet` [156]: A model-based meta-learning method which requires a few samples from test tasks as a support set to adapt.
- `MAML` [58]: A optimization-based meta-learning approach. We replaced the original classifier with a ResNet for regression.

Note that both `ResNet-c` and `Unet-c` have access to task parameters c. `Modular-attn` has a convolutional encoder f that takes the same input x as each module M to generate attention weights, $\sum_{l=1}^{m} \frac{\exp[f(x)(l)]}{\sum_{k=1}^{m}\exp[f(x)(k)]} M_l(x)$. `Modular-wt` also has the same encoder but to generate weights for combining the convolution parameters of all modules. `MetaNet` requires samples from test tasks as a support set and MAML needs adaptation retraining on test tasks, while other models do not need any information from the test domains. Thus, we use additional samples of up to 20% of the test set from test tasks. MetaNet uses these as a support set. MAML is retrained on these samples for 10 epoch for adaptation. To demonstrate the generalizability of DyAd, we experimented with ResNet and U-net as our forecaster.

3.3.3 Evaluation Metrics

Apart from the root mean square error (RMSE), we also report the energy spectrum error (ESE) for ocean current prediction, which quantifies the physical consistency. ESE indicates whether the predictions preserve the correct statistical distribution and obey the energy conservation law, which is a critical metric for physical consistency.

3.4 Results

3.4.1 Prediction Performance.

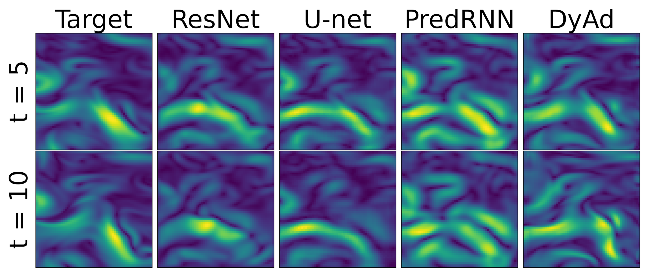

Figure 3.5. DyAd, ResNet, U-net, PredRNN velocity norm ($\sqrt{u^2+v^2}$) predictions on an ocean current sample in the future test set.

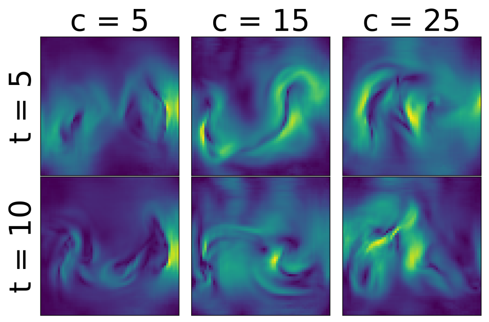

Figure 3.6. Outputs from DyAd while we vary encoder input but keep the forecaster input fixed. From left to right, the encoder is fed with flow with different buoyancy factor $c = 5, 15, 25$. the forecaster network input has fixed buoyancy $c = 15$.

Table 3.1 shows the RMSE of multi-step predictions on Turbulent Flows (20 steps), Sea Surface Temperature (10 steps), and Ocean Currents (10 step) in two testing scenarios. We observe that DyAd makes the most accurate predictions in both scenarios across all datasets. Comparing ResNet/Unet-c with DyAd, we observe a clear advantage of task inference with separate training. VarSepNet achieves competitive performances on Ocean Currents (second best) through spatiotemporal disentanglement but cannot adapt to future data. Table 3.1 also reports ESEs on real-world Ocean Currents. DyAd not only has small RMSE but also obtains the smallest ESE, suggesting it captures the statistical distribution of ocean currents well.

Figure 3.4 shows the target and predicted velocity norm fields ($\sqrt{u^2+v^2}$) by Unet-c, Modular-wt and DyAd at time step 1, 5, 10 for Turbulent Flows with buoyancy factors 9 and 21 respectively. We can see that DyAd can generate realistic flows with the corresponding characteristics while the baselines have trouble understanding and disentangling the buoyancy factor. Figure 3.5 shows DyAd, ResNet, U-net, PredRNN predictions on an ocean current sample in the future test set, and we see the shape of predictions by DyAd is closest to the target. These results demonstrate that DyAd not only forecasts well but also accurately captures the physical characteristics of the system.

3.4.2 Controllable Forecast.

DyAd infers the hidden features from data, which allows direct control of the latent space in the forecaster. We vary the encoder input while keeping the forecaster input fixed. Figure 3.6 shows the forecasts from DyAd when the encoder is fed with flows having different buoyancy factors $c = 5, 15, 25$. As expected, with higher buoyancy factors, the predictions from the forecaster become more turbulent. This demonstrates that the encoder can successfully disentangle the latent representation of difference tasks, and control the predictions of the forecaster.

3.4.3 Ablation Study.

We performed several ablation studies of DyAd on the turbulence dataset to understand the contribution of each model component.

We first performed an ablation study of the model architecture, shown in Table 3.2. We removed the encoder from DyAd while keeping the same forecaster network (No_enc). The resulting model degrades but still outperforms ResNet. This demonstrates the effectiveness of AdaIN and AdaPad for forecasting. We also tested DyAd with AdaIN only (No_AdaPad), and the performance without AdaPad was slightly worse.

Another notable feature of our model is the ability to infer tasks with weakly supervised signals c. It is important to have a c that is related to the task domain. As an ablative study, we fed the encoder in DyAd with a random c, leading to Wrong_enc. We can see that having the totally wrong supervision may slightly hurt the forecasting performance. We also trained the encoder and the forecaster in DyAd altogether (End2End) but observed worse performance. This validates our hypothesis about the significance of domain partitioning and separate training strategy.

Table 3.2. Ablation study: prediction RMSE of DyAd and its variations with different components removed.

Model	future	domain
DyAd	**0.42**±**0.01**	**0.51**±**0.02**
No_enc	0.63±0.03	0.60±0.02
No_AdaPad	0.47±0.01	0.54±0.02
Wrong_enc	0.66±0.02	0.62±0.03
End2End	0.45±0.01	0.54±0.01

3.5 Conclusion

We propose a model-based meta-learning method, DyAd, to forecast physical dynamics. DyAd uses an encoder to infer the parameters of the task and a prediction network to adapt and forecast giving the inferred task. Our model can also leverage any weak supervision signals that can help distinguish different tasks, allowing the incorporation of additional domain knowledge. On challenging turbulent flow prediction and real-world ocean temperature and currents forecasting tasks, we observe the superior performance of our model across heterogeneous dynamics.

Although this work focuses on distributional shifts across different domains, temporal distribution shifts are also common in real-world time-series applications. Therefore, it is crucial to develop models that can handle distribution shifts over time caused by factors such as a highly dynamic and non-stationary environment, unforeseeable abrupt changes, or constantly evolving trends in the underlying data distribution. To address this, we introduce the `Koopman Neural Forecaster`, inspired by Koopman theory, which achieves high accuracy in forecasting highly nonstationary time series.

Chapter 4

Koopman Neural Forecaster for Time Series with Temporal Distribution Shifts

4.1 Introduction

4.1.1 Challenges in Highly Nonstationary Time Series Forecasting

Aside from the distributional shifts among different domains we discussed in the last chapter, temporal distribution shifts frequently occur in real-world time-series applications, from forecasting stock prices to detecting and monitoring sensory measures to predicting fashion trend-based sales. Such distribution shifts over time may be due to the data being generated in a highly-dynamic and non-stationary environment, abrupt changes that are difficult to predict, or constantly evolving trends in the underlying data distribution [63].

Temporal distribution shifts pose a fundamental challenge for time-series forecasting [119]. There are two scenarios of distribution shifts. When the distribution shifts only occur between the training and test domains, meta-learning and transfer-learning approaches [96, 161] have been developed. The other scenario is much more challenging: distribution shifts occur continuously over time. This scenario is closely related to "concept drift" [141] and non-stationary processes [41] but has received less attention from the deep learning community. In this work, we focus on the second scenario.

To tackle temporal distribution shifts, various statistical estimation methods have been studied, including spectral density analysis [41], sample reweighting [12, 150] and Bayesian state-space models [235]. However, these methods are limited to low-capacity auto-regressive models and are typically designed for short-horizon forecasting. For large-scale complex time series data, deep learning models [161, 239, 212, 258] now increasingly outperform traditional statistical methods. Yet, most deep learning approaches are designed for stationary time-series data (with i.i.d. assumption), such as electricity usage, sales and air quality, that have clear seasonal and trend patterns. For distribution shifts, DNNs have been shown to be problematic in forecasting data with varying distributions [114, 225].

DNNs are black-box models and often require a large number of samples to learn. For time series with continuous distribution shifts, the number of samples from a given distribution is small, thus DNNs would struggle to adapt to the changing distribution. Furthermore, the non-linear dependencies in a DNN are difficult to interpret or manipulate. Directly modifying the parameters based on the change in dynamics may lead to undesirable effects [219]. Therefore, if we can reduce non-linearity and simplify dynamics modeling, then we would be able to model time series in a much more interpretable and robust manner. Koopman theory [113] provides convenient tools to simplify dynamics modeling. It states that any nonlinear dynamics can be modeled by a *linear* Koopman operator acting on the space of measurement functions [24], thus the dynamics can be manipulated by simply modifying the Koopman matrix.

In this paper, we propose a novel approach for accurate forecasting for time series with distribution shifts based on Koopman theory: Koopman Neural Forecaster (`KNF`). Our model has three main features: 1) we combine predefined measurement functions with learnable coefficients to introduce appropriate inductive biases into the model. 2) our model employs both global and local Koopman operators to approximate the forward dynamics: the global operator learns the shared characteristics; the local operator captures the local changing dynamics. 3) we also integrate a feedback loop to cope with distribution shifts and maintain the model's long-term forecasting accuracy. The feedback loop continuously updates the learned operators over time based on the current prediction error.

Leveraging Koopman theory brings multiple benefits to time series forecasting with distribution shifts: 1) using predefined measurement functions (e.g., exponential, polynomial) provides sufficient expressivity for the time series without requiring a large number of samples. 2) since the Koopman operator is linear, it is much easier to analyze and manipulate. For instance, we can perform spectral analysis and examine its eigenfunctions, reaching a better understanding of the frequency of oscillation. 3) Our feedback loop makes the Koopman operator adaptive to a non-stationary environment. This is fundamentally different from previous works that learn a single and fixed Koopman operator [76, 207, 8].

4.1.2 Koopman Theory

Time series data $\{\boldsymbol{x}_t\}_{t=1}^{T}$ can be considered as observations of a dynamical system states – consider the following discrete form: $\boldsymbol{x}_{t+1} = \boldsymbol{F}(\boldsymbol{x}_t)$, where $\boldsymbol{x} \in \mathscr{X} \subseteq \mathbb{R}^d$ is the system state, and $\boldsymbol{F}$ is the underlying governing equation. We focus on multi-step forecasting task of predicting the future states given a sequence of past observations. Formally, we seek a function map f such that:

$$f : (\boldsymbol{x}_{t-q+1}, \ldots, \boldsymbol{x}_t) \longrightarrow (\boldsymbol{x}_{t+1}, \ldots, \boldsymbol{x}_{t+h}), \tag{4.1}$$

where q is the lookback window length and h is the forecasting window length.

Koopman theory [113] shows that any nonlinear dynamic system can be modeled by an infinite-dimensional linear Koopman operator acting on the space of all possible measurement functions. More specifically, there exists a linear infinite-dimensional operator $\mathscr{K} : \mathscr{G}(\mathscr{X}) \mapsto \mathscr{G}(\mathscr{X})$ that acts on a space of real-valued measurement functions $\mathscr{G}(\mathscr{X}) := \{g : \mathscr{X} \mapsto \mathbb{R}\}$. The Koopman operator maps between function spaces and advances the observations of the state to the next step:

$$\mathscr{K} g(\boldsymbol{x}_t) = g(\boldsymbol{F}(\boldsymbol{x}_t)) = g(\boldsymbol{x}_{t+1}). \tag{4.2}$$

4.1.3 Related Work

DNNs for time-series forecasting. DNNs are shown to increasingly outperform traditional statistical methods (such as exponential smoothing (ETS) [64] and ARIMA [6]) for time series forecasting. For example, [212, 231] proposed to use DNNs to learn the local and global representations of time series separately, showing high accuracy on sales and weather data. [239] leverages inductive biases in different architectures and also specially designed contrastive loss to learn disentangled seasonal and trend representations. [194] utilized a global TCN to avoid normalization before training when there are wide variations in scale. But it focuses mainly on better modeling the relationships between time series rather than advances in modeling over time as ours. Transformer-based approaches are particularly effective in time series forecasting, particularly on datasets including electricity and traffic [258, 243, 257], which are relatively stationary and have clear seasonality and trend dynamics.

Robustness against temporal distribution shifts. Non-stationarity poses a great challenge for time series forecasting. To cope with varying distributions, one approach is to stationarize the input data. [108] proposes a reversible instance normalization technique applied on data to alleviate the temporal distribution shift problem. Similarly, [163] utilizes a DNN to adaptively stationarize input time series. But these approaches did not improve the generalizability of DNNs. [137] proposes a normalization-denormalization technique to stationarize time series, but only for transformer-based models. [5] proposes test-time adaptation with a self-supervised objective to better adapt against distribution shifts. Another line of work is to combine DNNs and statistical approaches, for better accuracy on non-stationary time series data [144, 145]. [200] combines ETS with a RNN, where the seasonality and smoothing coefficients, are fitted concurrently with the RNN weights using the same gradient descent optimization. [160] proposes a deep residual architecture for univariant time series data, in which each block used MLP to learn coefficients of basis functions, such as polynomials and sine functions. Indeed, we show that our model outperforms both [200] and [160] on the M4 financial time series dataset [144].

Koopman theory. Koopman theory [113, 205] shows that a nonlinear dynamical system can be represented as an infinite-dimensional linear Koopman operator acting on a space of measurement functions. The spectral decomposition of the linear Koopman operator can provide insights on the behaviors of nonlinear systems. Traditionally, dynamic mode decomposition (DMD) [21] is commonly used for approximating the Koopman Operator. However, it is highly nontrivial to find appropriate measurement functions as well as the Koopman operator. Many works at the intersection of machine learning and Koopman theory employ DNNs to learn measurement functions from data [76, 127, 155, 126, 207, 56]. [248] and [142] propose to use fully connected DNNs to learn the transformation between the observations and measurement functions that span a Koopman invariant subspace and the Koopman operator is approximated by a linear layer. [8] also designed an autoencoder architecture based on Koopman theory to forecast fluid dynamics. Different from these approaches, we allow the Koopman matrix to evolve over time for each time series to adapt to the changing distribution.

4.2 Koopman Neural Forecaster

We propose Koopman Neural Forecasting (`KNF`), a deep sequence model based on Koopman theory to forecast highly non-stationary time series, as shown in Fig. 4.1. It instantiates an encoder-decoder architecture for each time series segment. The encoder takes in observations from multiple time steps as the underlying dynamics may contain higher-order time derivatives. Our model has three main features: 1) we use predefined measurement functions with learned coefficients to map time series to the functional space. 2) the model employs both a global Koopman operator to learn the shared characteristics and a local Koopman operator to capture the local changing dynamics. 3) we also integrate a feedback loop to update the learned operators over time-based on forecasting error, maintaining the model's long-term forecasting performance.

Leveraging Predefined Measurement Functions We define a set of measurement functions $\mathcal{G} := [g_1, \cdots, g_n]$ that spans the Koopman space, where each $g_i : \mathbb{R} \mapsto \mathbb{R}$. For example,

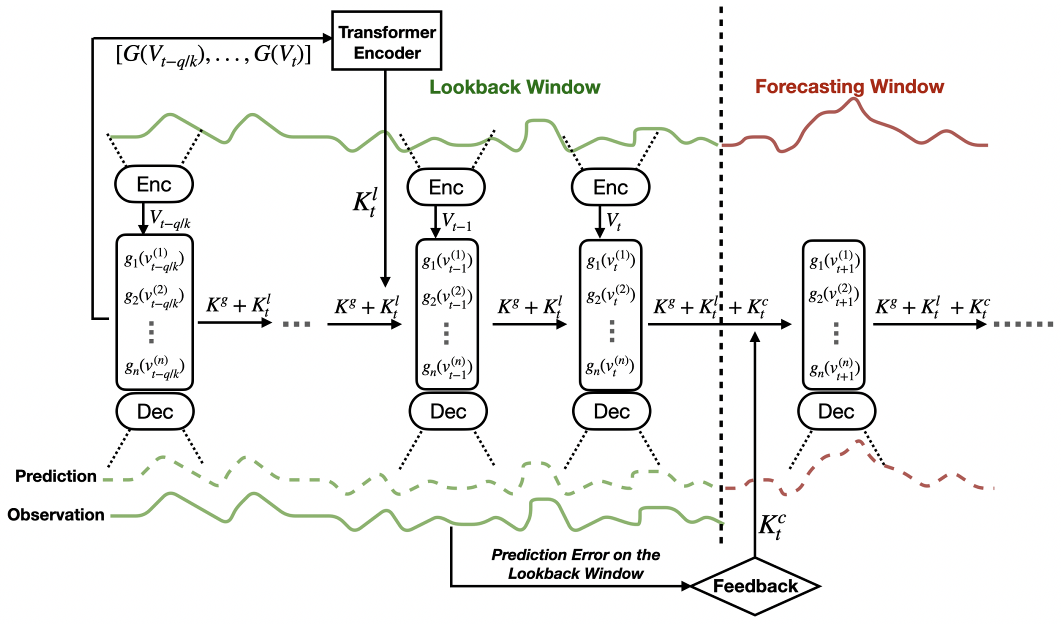

Figure 4.1. Architecture of the proposed KNF model. It encodes every multiple steps of observations into a measurement vector and utilizes a global operator and an adaptive local operator to model the evolution of measurements. An additional adjustment operator is also learned on the forecasting window, .

$g_1(x) = \sin(x)$. These functions are canonical nonlinear functions and are often used to model complex dynamical systems, such as Duffing oscillator and fluid dynamics [24, 118]. They also provide a sample-efficient approach to represent highly nonlinear behavior that may be difficult to learn for DNNs.

We use an encoder to generate the coefficients of the measurement functions $\Psi(\boldsymbol{X}_t)$, such as the frequencies of sine functions. Let n be the number of measurement functions for each feature, d be the number of features in a time series and k be the number of steps encoded by the encoder $\Psi : \mathbb{R}^{d\times k} \mapsto \mathbb{R}^{n\times d\times k}$ every time. The lookback window length q is a multiple of k and we denote $\boldsymbol{x}_{tk:(t+1)k}$ as $\boldsymbol{X}_t \in \mathbb{R}^{d\times k}$ for simplicity.

As shown in the Eq.4.3 below, we first obtain a latent matrix $\boldsymbol{V}_t = [\boldsymbol{v}_t^{(1)}, \boldsymbol{v}_t^{(2)}, \cdots, \boldsymbol{v}_t^{(n)}] \in \mathbb{R}^{n\times d}$. Every vector $\boldsymbol{v}_i \in \mathbb{R}^d$ is a different linear transformation of the observations, where the weights are learnt by the encoder Ψ:

$$\boldsymbol{V}_t[i,j] = \sum_l \Psi(\boldsymbol{X}_t)[i,j,l]\boldsymbol{X}_t[j,l]; \quad 1 \leq i \leq n,\ 1 \leq j \leq d,\ 1 \leq l \leq k. \tag{4.3}$$

Our measurement functions are defined in the latent space rather than the observational space. We apply a set of predefined measurement functions $\mathscr{G}$ to the latent matrix $\boldsymbol{V}_t$:

$$\mathscr{G}(\boldsymbol{V}_t) = [g_1(\boldsymbol{v}_t^{(1)}), g_2(\boldsymbol{v}_t^{(2)}), \ldots, g_n(\boldsymbol{v}_t^{(n)})] \in \mathbb{R}^{n\times d} \tag{4.4}$$

In our implementation, we flatten $\mathscr{G}(\boldsymbol{V}_t)$ into a vector and then finite Koopman operator should be a $nd \times nd$ matrix. Finally, we use a decoder $\Phi : \mathbb{R}^{n\times d} \mapsto \mathbb{R}^{k\times d}$ to reconstruct the observations from the measurements:

$$\hat{\boldsymbol{X}}_t = \Phi(\mathscr{G}(\boldsymbol{V}_t)). \tag{4.5}$$

Here, the encoder Ψ and the decoder Φ can be any DNN architecture, for which we use multi-layer perceptron (MLP). The set of measurement functions $\mathscr{G}$ contains polynomials, exponential functions, trigonometric functions as well as interaction functions. These predefined measurement functions are useful in imposing inductive biases into the model and help capture the non-linear behaviors of time series. The encoder model needs to approximate only the parameters of these functions without the need to directly learn non-stationary characteristics. Ablation studies demonstrate that using predefined measurement functions significantly outperforms the model with learned measurement functions in the previous works.

Global and Local Koopman Operators Dynamic mode decomposition (DMD) [213] is traditionally used to find the Koopman operator that best propagates the measurements. But for time series with a temporal distribution shift, we need to compute spectral decomposition and learn a Koopman matrix for every sample (i.e. slice of a trajectory), which is computationally expensive. So we utilize DNNs to learn Koopman operators.

In classic Koopman theory, the measurement vector $\mathscr{G}(\boldsymbol{V}_t)$ is infinite-dimensional, which is impossible to learn. We assume that encoder is learning a finite approximation and $\mathscr{G}(\boldsymbol{V}_t)$ forms a finite Koopman-invariant subspace. Thus, the Koopman operator $\mathscr{K}$ that we need to find is the finite matrix that best advances the measurements forward in time.

While the Koopman matrix should vary across samples and time in our case, it should also capture the global behaviors. Thus, we propose to use both a global operator $\mathscr{K}^g$ and a local operator $\mathscr{K}_t^l$ to model the propagation of dynamics in the Koopman space, defined as below:

$$\mathscr{K}\mathscr{G}(\boldsymbol{V}_t) = (\mathscr{K}^g + \mathscr{K}_t^l)\mathscr{G}(\boldsymbol{V}_t) = \mathscr{G}(\boldsymbol{V}_{t+1}), \;\; t \geq 0. \tag{4.6}$$

The global operator $\mathscr{K}^g$ is an $nd \times nd$ trainable matrix that is shared across all time series. We use the global operator to learn the shared behavior such as trend. The local Koopman operator $\mathscr{K}_t^l$, on the other hand, is based on the measurement functions on the lookback window for each sample, shown in Fig. 4.1. The local operator should capture the local dynamics specific to each sample. Since we generate the forecasts in an autoregressive way, the local operator depends on time t and varies across autoregressive steps, adapting to the distribution changes along with prediction. We use a Transformer architecture with a single head as the encoder, to capture the relationships between measurements at different steps. We use the attention weight matrix in the last layer as the local Koopman operator.

Feedback Loop Suppose an abrupt distributional shift occurs in the middle of the lookback window, the model would still try to fit two distributions before and after the shift but a single propagation matrix is never good enough to model multiple distributions. This will result in the inaccurate operator used for the forecasting window. To address it, we add an additional feedback closed-loop, in which we employ an MLP module Γ to learn the adjustment operator $\mathscr{K}_t^c$ based on the prediction errors in the lookback window. It is directly added to other operators when making predictions on the forecasting window, as shown in Fig. 4.1. More specifically, we apply global and local operators recursively to the measurements at the first step in the lookback window to obtain predictions:

$$\hat{\boldsymbol{X}}_{t-q/k+i} = \Phi((\mathscr{K}^g + \mathscr{K}_t^l)^i \mathscr{G}(\boldsymbol{V}_{t-q/k})), \quad 0 < i \leq q/k. \tag{4.7}$$

Then, the feedback module Γ uses the difference between the predictions on the lookback window and the observed data to generate additional adjustment operator $\mathscr{K}_t^c$, which is a diagonal matrix:

$$\mathscr{K}_t^c = \Gamma(\hat{\boldsymbol{X}}_{t-q/k:t} - \boldsymbol{X}_{t-q/k:t}) = \Gamma(\hat{\boldsymbol{x}}_{t-q:t} - \boldsymbol{x}_{t-q:t}) \tag{4.8}$$

If the predictions deviate significantly from the ground truth within the lookback window, the operator $\mathscr{K}_t^c$ would learn the temporal change in the underlying dynamics and correspondingly adjust the other two operators. Thus, for forecasting, the sum of all three operators is used:

$$\hat{\boldsymbol{X}}_{t+i} = \Phi((\mathscr{K}^g + \mathscr{K}_t^l + \mathscr{K}_t^c)^i \mathscr{G}(\boldsymbol{V}_t)), \quad i > 0. \tag{4.9}$$

In a word, the feedback module is designed to detect the distributional shifts in the lookback window and adapt the global+local operator to the latest distribution in the lookback window.

Loss Functions `KNF` is trained in an end-to-end fashion, using superposition of three loss terms $L = L_{\text{rec}} + L_{\text{back}} + L_{\text{forw}}$. Denote l as a distance metric for which we use the L2 loss. The first term is the reconstruction loss, to ensure the decoder Φ can reconstruct the time series from the measurements:

$$L_{\text{rec}} = l(\boldsymbol{X}_t, \Phi(\mathscr{G}(\Psi(\boldsymbol{X}_t)\boldsymbol{X}_t))), \quad t \geq 0. \tag{4.10}$$

The second term is the prediction loss on the lookback window to ensure the sum of global and local operators is the best-fit propagation matrix on the lookback window.

$$L_{\text{back}} = l(\boldsymbol{X}_{t-q/k+i}, \Phi((\mathscr{K}^g + \mathscr{K}_t^l)^i \mathscr{G}(\Psi(\boldsymbol{X}_{t-q/k})\boldsymbol{X}_{t-q/k})), \quad 0 < i \leq q/k. \tag{4.11}$$

The third term is for prediction accuracy in the forecasting window to guide the feedback loop to learn the correct adjustment placed on the Koopman operator.

$$L_{\text{forw}} = l(\boldsymbol{X}_{t+i}, \Phi((\mathscr{K}^g + \mathscr{K}_t^l + \mathscr{K}_t^c)^i \mathscr{G}(\boldsymbol{V}_t)), \quad i > 0. \tag{4.12}$$

4.3 Experiments Setup

4.3.1 Datasets

We benchmark our method on three time series datasets: M4 [144], Crypto [5] and Basketball Player Trajectories [125]. These time series are particularly chosen as they are difficult to forecast due to high nonstationarity and abrupt temporal changes, as analyzed in Sec. 4.2.

M4. It contains 10000 highly nonstationary univariate time series with different frequencies from hourly to yearly and different categories from financials to demographics. The forecasting horizon varies across different frequencies. We directly compare KNF with the M4 competition winner [200], the second place [154] and the ensemble N-Beats-I+G [160] that has achieved the competitive prediction performance on M4.

Crypto.[1] This multivariate dataset contains 8 features on historical trades, such as open and close prices, for 14 cryptocurrencies. The original challenge is to predict 3-step ahead 15-minute relative future returns. Since we focus on long-term forecasting, we train all models to make 15-step predictions of 15-minute relative future returns. We use the original training set from the competition and do an 80%-10%-10% training-validation-test split.

Player Trajectory.[2] This dataset contains basketball player movement trajectories from NBA games in 2016. We randomly sample 300 player trajectories for training and validation and 50 trajectories for testing. All models are trained to yield 30-step ahead predictions.

As a way of motivating for the improvement scenarios of our method, we show that the

[1]https://www.kaggle.com/competitions/g-research-crypto-forecasting/data
[2]https://github.com/linouk23/NBA-Player-Movements

three datasets we focus on are much more difficult to predict than commonly-used datasets like Electricity [78] with the following three metrics: (1) Forecastability [70]: it is one minus the entropy of Fourier decomposition of the time series; (2) Lyapunov exponents (LEs) [48, 193]: a measure of how sensitive a dynamical system is to initial conditions.(3) Trend: the slope of the linear regression fitted on the time series scaled by its own magnitude; and (4) Seasonality: we use ACF test [238] to test if there is clear seasonality. We report the mean forecastability, mean trend and the percentage of the slices that have seasonality in Table 4.1, averaged over slices with length 20. We can see that seasonality is dominant for Electricity dataset, which also has significantly higher forecastability compared to the datasets we experiment with. Moreover, we show that KNF achieves the state-of-the-art prediction performance on those datasets that have high Lyapunov exponents, low forecastability, no clear trends and seasonality, including Crypto, Player Trajectories, M4-monthly, M4-weekly and M4-daily.

Table 4.1. The forecastability, Lyapunov exponents, trend and seasonality of the datasets that we used for our experiments, compared with a commonly-used Electricity benchmark. The boldface datasets are what we use for experiments in this paper.

	Electricity	**Crypto**	**Player Traj.**	**M4-monthly**	**M4-weekly**	**M4-daily**	M4-hourly	M4-yearly	M4-quarterly
Forecastability	0.77	0.35	0.49	0.44	0.43	0.44	0.46	0.58	0.47
LEs	0.005	0.026	0.052	0.011	0.013	0.020	0.003	0.004	0.003
Trend	0.00	0.02	0.01	0.48	0.13	0.05	0.02	4.32	1.06
Seasonality	100%	0.00%	0.00%	66.34%	0.00%	0.00%	99.76%	0.00%	84.51%

For all datasets, we use a sliding window approach to generate training samples. On Crypto and Player Trajectory datasets, we perform a grid search of hyperparameters, including learning rate, input length, hidden dimension, number of predictions made in each autoregressive step, etc. for all models. The default set of measurement functions used in KNF includes polynomials up to the order of four, one exponential function and trigonometric functions with the same number of input steps for each feature, as well as pairwise product interaction functions between features if the time series data is multivariate.

4.3.2 Baselines

For M4, we compare with the winner solution [200], the second best [154] and N-beats [160]. On the Crypto and Player Trajectory datasets, we compare KNF with four different types of models.

- **Vector ARIMA** (VARIMA) [204]: It is an extension of the classic ARIMA model for multivariate time series.
- **Multi-layer Perceptron** (MLP) that maps the historic observations to the future and rolls out autoregressively to yield long-term predictions. It is a commonly-used DL model for time series forecasting [5, 225, 55].
- **Random Forest** (RF) applied autoregressively same as MLP. It is a traditional ML model that have achieved competitive performance on many time series benchmarks [69, 147, 99, 91].
- **Long Expressive Memory** (LEM) [180] : An SOTA recurrent model to learn long-term sequential dependencies.
- **FedFormer** (FedFormer) [258] is a state-of-the-art transformer-based model, which has outperformed other transformer-based models, such as Informer [257] and Autoformer [243], on many datasets, including electricity, traffic, weather, etc.
- **Variational Beam Search** (VBS) [125] is a Bayesian online learning model proposed to detect and adapt to temporal distributional shifts. Since VBS is not designed for time series forecasting, we modify it by feeding its own prediction of next step back to the input (instead of the ground truth) to yield multi-step predictions.

We find that the following two training strategies can improve the prediction accuracy of models by varying degrees. The first one is the reversible instance normalization ReVIN [108], which normalizes the input sequence and de-normalizes the predictions at every autoregressive step for every instance. We use ReVIN for MLP and KNF since it can improve their prediction

Table 4.2. sMAPE for KNF and baselines on M4 datasets. The numbers in parentheses are the number of prediction steps. KNF achieves state-of-the-art prediction performance at Weekly, Daily, and Monthly frequencies.

sMAPE	Monthly(18)	Weekly(13)	Daily(14)
Montero	12.639	7.625	3.097
Smyl	12.126	7.817	3.170
Nbeats-I+G	12.024	-	-
KNF	**11.930**	**7.254**	**2.990**

accuracy. An ablation study of ReVIN on KNF can be found in Table 4.3. The second one is, temporal bundling TB [17] that asks autoregressive models to generate multiple-step predictions instead of just one on every call, to reduce the number of model calls and therefore error propagation speed. We observe this strategy can improve the prediction accuracy of both MLP and RF.

4.3.3 Evaluation Metrics

We report the mean and standard deviation averaged across five runs. We follow the literature and use RMSE for the evaluation on Player Trajectories and the weighted RMSE on Crypto, where the weight corresponds to the importance of each cryptocurrency same as in the competition. Since VBS is deterministic once its inverse temperature parameter is fixed, we do not report its standard deviation. On M4, we evaluate models with the sMAPE metric [144][1] used in the original competition. Ensembling is used by most models in the M4-competition and N-Beats [160], so we ensemble five KNF with best hyperparameters but trained with random seeds. Since the evaluation in the M4 competition is only based on a single submission, we also report the sMAPE of a single ensembled prediction without standard deviation.

[1]sMAPE is the mean absolute error scaled by the magnitude of the predictions and target.

4.4 Results

4.4.1 Prediction Performance

Results on M4 Table 4.2 reports the prediction sMAPE of KNF and three baselines, including the M4 winner [200], the second place [154] and the ensembled N-beats-I+G [160], on six datasets with different frequency in M4. [160] does not report the breakdown sMAPE of N-beats on weekly, daily and hourly, so we did not include them in the table. The number behind each frequency in the table is the number of prediction steps required in the competition. We can see that KNF achieves state-of-the-art accuracy on Weekly, Daily and Monthly data that have longer forecasting horizons, and no clear trends and seasonality. This highlights the value proposition of KNF for accurate long-term forecasting accuracy especially for time series with nonstationary characteristics and low forecastability.

Table 4.3. Ablation study of KNF on M4-weekly data. It shows the importance of every component in the model architecture.

M4-Weekly	sMAPE
KNF-base-G	14.175
KNF-base-I	9.122
+RevIN	8.435
+RevIN+$\mathscr{K}^g$	7.500
+RevIN+$\mathscr{K}^g$+feedback	**7.254**

Results on Crypto and Player Trajectories Table 4.4 shows the Prediction RMSE on Crypto and Player Trajectories datasets. KNF consistently achieves the best performance on both, across different forecasting horizons. Fig. 4.2 exemplifies the predictions of KNF and best-performing baselines. We observe that KNF can capture both overall trends and small fluctuations in a superior way, yielding much higher accuracy. Fig. 4.2 right visualizes the predictions on a trajectory with changing direction. KNF correctly predicts the change of moving direction while FedFormer fails.

Regarding baselines, RevIN and TB greatly improve MLP, and MLP+RevIN+TB is the best

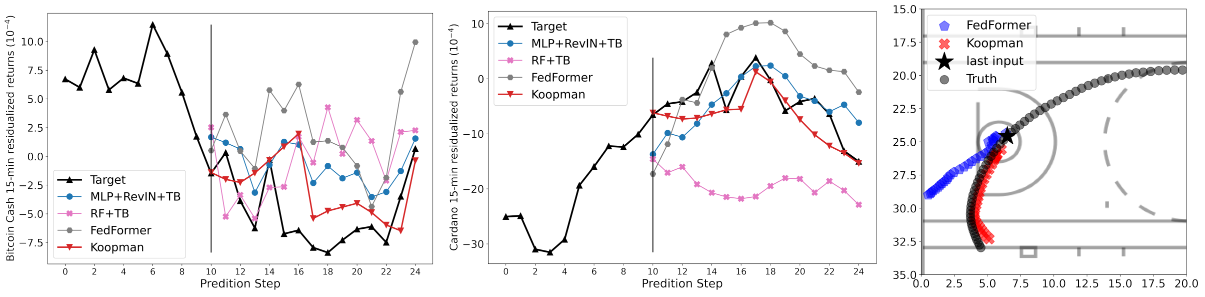

Figure 4.2. Left: Example forecasts on Crypto dataset. Right: Example forecasts on NBA player trajectory dataset. We observe that `KNF` can be superior in capturing complex non-stationary patterns.

performing one on Crypto, and `FedFormer` performs better on Player Trajectory, that are less chaotic and irregular. Though `VBS` was shown to perform well on the online change point detection task, it does not appear to be as successful for time series forecasting based on its performance on these two datasets.

4.4.2 Ablation Study

We performed an ablation study of `KNF` on the M4-Weekly data to understand the contribution of each component in our model. Table 4.3 shows the sMAPE of ensembled predictions from five funs of each variant. We denote `KNF`-base as the basic backbone of our model that only has an encoder-decoder architecture and a local Koopman operator. `KNF`-base-G uses purely data-driven measurements, which is similar to the Koopman autoencoders proposed in [248, 8]. This can be considered a baseline for our model. `KNF`-base-I uses predefined measurement functions, which significantly outperforms `KNF`-base-G. This demonstrates the

Table 4.4. Prediction RMSE on Cryptos and Player Trajectories datasets.

Model	Crypto (Weighted RMSE 10^{-3})				Basketball Player Trajectory (RMSE)			
	(1∼5)	(6∼10)	(11∼15)	Total	(1∼10)	(11∼20)	(21∼30)	Total
`VARIMA`	6.09±0.00	8.83±0.00	10.74±0.00	8.76±0.00	**0.22±0.00**	0.90±0.00	1.98±0.00	1.26±0.00
`MLP`	6.68±1.53	7.95±0.33	8.64±0.55	7.85±0.35	0.73±0.20	1.64±0.31	2.77±0.42	1.91±0.32
`MLP+RevIN+TB`	**5.03±0.08**	7.16±0.13	8.41±0.06	7.01±0.08	0.37±0.02	1.16±0.03	2.25±0.04	1.48±0.25
`RF+TB`	6.62±1.30	7.99±0.2s4	8.51±1.19	7.84±0.04	0.86±0.01	2.10±0.01	3.48±0.02	2.40±0.01
`FedFormer`	5.61±0.05	7.50±0.03	8.89±0.03	7.46±0.04	0.43±0.02	0.92±0.03	1.97±0.04	1.29±0.03
`LEM`	5.27±0.02	7.23±0.06	8.23±0.05	7.02±0.04	0.33±0.01	1.08±0.04	2.18±0.02	1.42±0.02
`VBS`	15.23	14.46	26.49	19.52	0.90	2.84	9.24	5.60
`KNF`	5.24±0.01	**7.03±0.01**	**7.63±0.01**	**6.91±0.01**	0.26±0.01	**0.84±0.01**	**1.81±0.01**	**1.16±0.01**

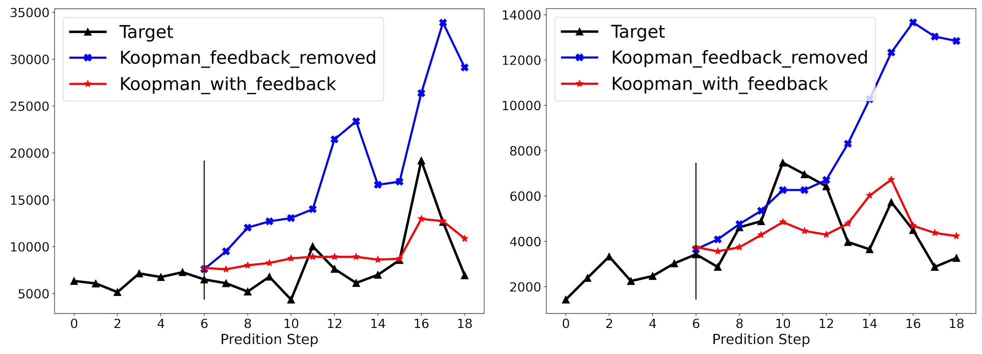

Figure 4.3. Predictions on M4-weekly data from original `KNF` and with its version without the feedback loop. We observe that the feedback loop can help provide robustness against temporal distributional shifts and maintain long-term prediction accuracy.

benefits of leveraging hard-coded functions. Moreover, adding any of ReVIN, the global Koopman operator and the feedback loop also brings great improvement based on the results shown in Table 4.3.

To further demonstrate the effectiveness of the feedback loop, we visualize the predictions on M4-weekly data from `KNF` and `KNF` with the feedback module removed after training in Figure 4.3. We can observe that the predictions from the model with the feedback loop removed start to deviate from the ground truth after a few steps. That means the feedback loop can cope with the temporal distributional shifts and thus improve the long-horizon forecasting accuracy.

4.4.3 Interpretability Analysis

`KNF` can offer unique interpretability capabilities via spectral analysis of the Koopman operators. We perform eigendecomposition on $\mathscr{K}^g + \mathscr{K}_t^l$ for `KNF` trained on M4-weekly data to investigate what individual eigenfunctions have learned. Fig. 4.4 shows the predictions in the lookback window with different eigenfunctions. From the top to the

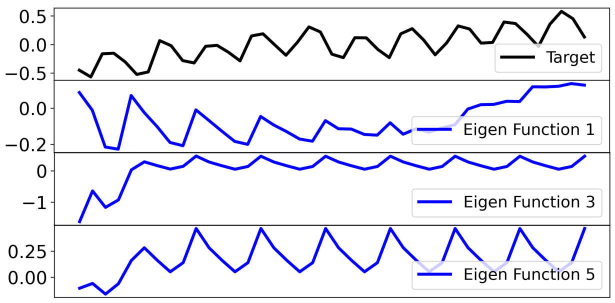

Figure 4.4. Reconstructions from `KNF` on the lookback window with only one of eigenfunctions of $\mathscr{K}^g + \mathscr{K}_t^l$ on a M4 time series.

bottom, the plot shows the target, the reconstruction from KNF with only the first eigenfunction in the lookback window and the reconstruction only using the second eigenfunction. We observe that some eigenfunctions capture the trend while some other functions focus on seasonality.

4.5 Conclusion

This chapter presents the Koopman Neural Forecaster (KNF), a novel model based on the Koopman theory that accurately forecasts non-stationary time series with temporal distribution shifts. KNF uses predefined measurement functions to capture the nonlinear and nonstationary characteristics that are challenging for neural nets to learn. The model employs a global operator to learn shared characteristics and a local operator to capture changing dynamics. Furthermore, we employ a feedback loop to continuously update the learned operators over time for rapidly varying behaviors. Our experiments show that KNF achieves state-of-the-art performance on a wide range of time series datasets known to suffer from distribution shifts, such as M4, Cryptos, and NBA player trajectory datasets, and provides interpretable results.

In addition to guiding DL models to understand domain or temporal distributional shifts due to changing dynamics, it is also crucial to incorporate the symmetries in the governing laws of dynamics into the design of DL models to make them generalize automatically to symmetry transformations. More importantly, DL models that respect symmetries obey conservation laws, thus generating physically meaningful predictions. In the next chapter, we discuss how to design equivariant neural nets for different symmetries and empirically and theoretically demonstrate the advantages of these nets in learning dynamics.

Chapter 5

Equivariant Deep Dynamics Models

5.1 Introduction

5.1.1 Symmetry and Equivariance

Exploiting symmetry in dynamical systems is a powerful way to improve the generalization of DL. The model learns to be equivariant or invariant to group transformations and hence is more robust to distribution shift. Symmetry has long been implicitly used in DL to design networks with known invariances and equivariances. Convolutional neural networks enabled breakthroughs in computer vision by leveraging translation equivariance [252, 123, 253]. Similarly, recurrent neural networks [179, 81], graph neural networks [189, 110], and capsule networks [182, 80] all impose symmetries.

While the equivariant DL models have achieved remarkable success in image and text data [37, 232, 38, 34, 124, 112, 9, 241, 61, 234, 47, 66, 202], the study of equivariant nets in learning dynamical systems has become increasingly popular recently [130, 226, 199, 84, 217, 198]. Since the symmetries can be integrated into neural nets through not only loss functions but also the design of neural net layers and there has been a large volume of works about equivariant and invariant DL

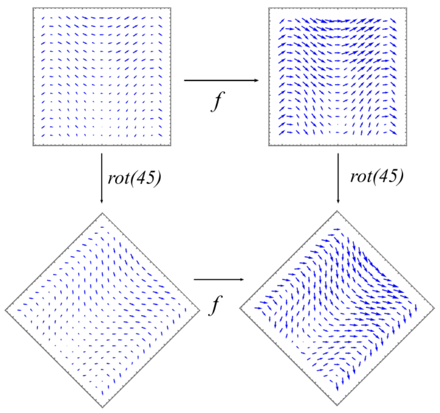

Figure 5.1. Illustration of equivariance: $f(x) = 2x$ w.r.t $T = \text{rot}(\pi/4)$

models for physical dynamics, we discuss this topic separately in this section.

In physics, there is a deep connection between symmetries and physics. Noether's law gives a correspondence between conserved quantities and groups of symmetries. For instance, translation symmetry corresponds to the conservation of energy and rotation symmetry corresponds to the conservation of angular momentum. By building a neural network that inherently respects a given symmetry, we thus make conservation of the associated quantity more likely and consequently the model's prediction more physically accurate. Furthermore, by designing a model that is inherently equivariant to transformations of its inputs, we can guarantee that our model generalizes automatically across these transformations, making it robust to distributional shifts.

A **group of symmetries** or simply **group** consists of a set G together with an associative composition map $\circ\colon G \times G \to G$. The composition map has an identity $1 \in G$ and composition with any element of G is required to be invertible. A group G has an **action** on a set S if there is an action map $\cdot\colon G \times S \to S$ which is compatible with the composition law. We say further that $\rho : G \mapsto GL(V)$ is a G-**representation** if the set V is a vector space and each group element $g \in G$ is represented by a linear map (matrix) $\rho(g)$ that acts on V. Formally, a function $f\colon X \to Y$ may be described as respecting the symmetry coming from a group G using the notion of equivariance.

Definition 2. *Assume a group representation ρ_{in} of G acts on X and ρ_{out} acts on Y. We say a function f is **G-equivariant** if*

$$f(\rho_{in}(g)(x)) = \rho_{out}(g)f(x) \tag{5.1}$$

*for all $x \in X$ and $g \in G$. The function f is **G-invariant** if $f(\rho_{in}(g)(x)) = f(x)$ for all $x \in X$ and $g \in G$. This is a special case of equivariance for the case $\rho_{\text{out}}(g) = 1$. See Figure 5.1 for an illustration of a rotation equivariant function.*

5.1.2 Symmetries of Differential Equations

By classifying the symmetries of a system of differential equations, the task of finding solutions is made far simpler, since the space of solutions will exhibit those same symmetries. Let G be a group equipped with an action on 2-dimensional space $X = \mathbb{R}^2$ and 3-dimensional spacetime $\hat{X} = \mathbb{R}^3$. Let $V = \mathbb{R}^d$ be a G-representation. Denote the set of all **V-fields** on $\hat{X}$ as $\hat{\mathscr{F}}_V = \{\boldsymbol{w} \colon \hat{X} \to V : \boldsymbol{w} \text{ smooth}\}$. Define $\mathscr{F}_V$ similarly to be V-fields on X. Then G has an induced action on $\hat{\mathscr{F}}_V$ by $(g\boldsymbol{w})(x,t) = g(\boldsymbol{w}(g^{-1}x, g^{-1}t))$ and on $\mathscr{F}_V$ analogously.

Consider a system of differential operators $\mathscr{D}$ acting on $\hat{\mathscr{F}}_V$. Denote the set of solutions $\mathrm{Sol}(\mathscr{D}) \subseteq \hat{\mathscr{F}}_V$. We say G is **a symmetry group of** $\mathscr{D}$ if G preserves $\mathrm{Sol}(\mathscr{D})$. That is, if φ is a solution of $\mathscr{D}$, then for all $g \in G$, $g(\varphi)$ is also. In order to forecast the evolution of a system $\mathscr{D}$, we model the forward prediction function f. Let $\boldsymbol{w} \in \mathrm{Sol}(\mathscr{D})$. The input to f is a collection of k snapshots at times $t-k, \ldots, t-1$ denoted $\boldsymbol{w}_{t-i} \in \mathscr{F}_d$. The prediction function $f \colon \mathscr{F}_d^k \to \mathscr{F}_d$ is defined $f(\boldsymbol{w}_{t-k}, \ldots, \boldsymbol{w}_{t-1}) = \boldsymbol{w}_t$. It predicts the solution at a time t based on the solution in the past. Let G be a symmetry group of $\mathscr{D}$. Then for $g \in G$, $g(\boldsymbol{w})$ is also a solution of $\mathscr{D}$. Thus $f(g\boldsymbol{w}_{t-k}, \ldots, g\boldsymbol{w}_{t-1}) = g\boldsymbol{w}_t$. Consequently, f is G-equivariant.

We mainly focus on the symmetries in fluid systems governed by Navier-Stokes equations that are invariant under the following five different transformations. Individually each of these types of transformations generates a group of symmetries in the system.

- Space translation: $T_{\boldsymbol{c}}^{\mathrm{sp}} \boldsymbol{w}(\boldsymbol{x}, t) = \boldsymbol{w}(\boldsymbol{x} - \boldsymbol{c}, t), \quad \boldsymbol{c} \in \mathbb{R}^2,$
- Time translation: $T_{\tau}^{\mathrm{time}} \boldsymbol{w}(\boldsymbol{x}, t) = \boldsymbol{w}(\boldsymbol{x}, t - \tau), \quad \tau \in \mathbb{R},$
- Galilean: $T_{\boldsymbol{c}}^{\mathrm{um}} \boldsymbol{w}(\boldsymbol{x}, t) = \boldsymbol{w}(\boldsymbol{x} - \boldsymbol{c}t, t) + \boldsymbol{c}, \quad \boldsymbol{c} \in \mathbb{R}^2,$
- Rotation/Reflection: $T_R^{\mathrm{rot}} \boldsymbol{w}(\boldsymbol{x}, t) = R\boldsymbol{w}(R^{-1}\boldsymbol{x}, t),\ R \in O(2),$
- Scaling: $T_{\lambda}^{sc} \boldsymbol{w}(\boldsymbol{x}, t) = \lambda \boldsymbol{w}(\lambda \boldsymbol{x}, \lambda^2 t), \quad \lambda \in \mathbb{R}_{>0}.$

5.1.3 Equivariant CNNs

We primarily employ convolutional-based models for our dynamics forecasting problem, with a specific emphasis on the following two types of equivariant CNNs.

G-Equivariant Group Convolution [38] A G-equivariant group convolution takes as input a c_{in}-dimensional feature map $f\colon G \to \mathbb{R}^{c_{\text{in}}}$ and convolves it with kernel $\Psi\colon G \to \mathbb{R}^{c_{\text{out}} \times c_{\text{in}}}$ over a group G,

$$[f \star_G \Psi](g) = \sum_{h \in G} f(h)\Psi(g^{-1}h). \tag{5.2}$$

Here we assume G finite, however, G may also be taken to be compact if the sum is replaced with an integral. Group convolution achieves equivariance by weight sharing since the kernel weight Ψ at (g,h) depends only on $g^{-1}h$ and thus pairs with equal $g^{-1}h$ share weights. Notice that both f and the filter Ψ are functions on the plane $\mathbb{Z}^m$ in the first layer. But they are functions on discrete group G for all layers after.

G-Steerable Convolution [39] Let f be the input feature map $f\colon \mathbb{R}^2 \to \mathbb{R}^{c_{\text{in}}}$ with group $H \subset O(2)$ acting on the base space $\mathbb{R}^2$ by matrix multiplication and on the channel space $\mathbb{R}^{c_{\text{in}}}$ by ρ_{in}. Also fix an H-action ρ_{out} on $\mathbb{R}^{c_{\text{out}}}$. Let $\Phi\colon \mathbb{R}^2 \to \mathbb{R}^{c_{\text{out}} \times c_{\text{in}}}$ be a kernel, then by [233], the standard 2D convolution $f \star_{\mathbb{R}^2} \Phi$ is H-equivariant and $\mathbb{R}^2$-translation equivariant when

$$\Phi(hx) = \rho_{\text{out}}(h)\Phi(x)\rho_{\text{in}}(h^{-1}) \; \forall h \in H. \tag{5.3}$$

This linear constraint induces dependence in the weights which is called weight tying. Solving for a basis of solutions to (5.3) gives an equivariant kernel basis $\{\Phi_l\}_{l=1}^{L}$ which can be combined using trainable coefficients $\Phi = \sum_{l=1}^{L} w_l \Phi_l$ to learn any kernel solving (5.3).

5.1.4 Related Work

Equivariant Convolutional Neural Networks Apart from incorporating symmetries into regular convolution we have discussed in the previous sections, there has been a surge of interest in designing equivariant continuous convolution models. This is due to the fact that continuous convolution allows for convolutional operations to be performed on a continuous input domain. For instance, [192] proposed *SchNet*, which is a continuous convolution framework that generalizes the CNN approach to continuous convolutions to model particles at arbitrary positions. Continuous convolution kernels are generated by dense neural networks that operate on the interatomic distances, which ensures rotational and translation invariance of the energy. In a traffic forecasting application, [221] proposed a novel model, *Equivariant Continuous COnvolution (ECCO)* that uses rotationally equivariant continuous convolutions to embed the symmetries of the system for improved trajectory prediction. The rotational equivariance is achieved by a weight-sharing scheme within kernels in polar coordinates. *ECCO* achieves superior performance to baselines on two real-world trajectory prediction datasets, Argoverse and TrajNet++.

Equivariant Graph Neural Networks In addition to the equivariant convolution, numerous equivariant graph neural nets have also been developed, particularly for modeling atomic systems and molecular dynamics. This is due to the pervasive presence of symmetry in molecular physics, as evidenced by roto-translation equivariance in molecular conformations and coordinates. [188] designed E(n)-equivariant graph neural network for predicting molecular properties. It updates edge features with the Euclidean distance between nodes and updates the coordinates of particles with the weighted sum of relative differences of all neighbors. [196] proposed to use a score-based generative model for generating molecular conformation. The authors used equivariant graph neural networks to estimate the score function, which is the gradient fields of the log density of atomic coordinates because it is roto-translation equivariant. [4] designed *Cormorant*, a rotationally covariant neural network architecture for learning the behavior and properties of complex many-body physical systems. *Cormorant* achieves promising results in learning molecular potential energy surfaces on the MD-17 dataset and learning the geometric, energetic, electronic, and thermodynamic properties of molecules on the GDB-9 dataset. [214] further designed a series SE(3)-equivariant operations and building blocks for DL architectures operating on geometric point cloud data, which was used to construct *PhiSNet*, a novel architecture capable of accurately predicting wavefunctions and electronic densities. [16] proposed a lie point symmetry data augmentation method for training graph neural PDE solvers and this method enables these neural solvers to preserve multiple symmetries.

Symmetry Discovery There has been also an emerging area that is symmetry discovery, the key idea of which is to find the weight-sharing patterns in neural networks that have been trained on data with symmetries. For instance, [255] factorized the weight matrix in a fully connected layer into a symmetry (i.e. weight-sharing) matrix and a vector of filter parameters. The two parts are learned separately in the inner and outer loop training with the Model-Agnostic Meta-Learning algorithm (MAML) [59], which is an optimization-based meta-learning method so that the symmetry matrix can learn the weight-sharing pattern from the data. Furthermore, [46] proposed *Lie Algebra Convolutional Network (L-conv)*, a novel architecture that can learn the Lie algebra basis and automatically discover symmetries from data. It can be considered as an infinitesimal version of group convolution.

5.2 Incorporating Symmetry into Deep Dynamics Models for Improved Generalization

5.2.1 Deep Equivariant Dynamics Models

We prescribe equivariance by training within function classes containing only equivariant functions. Our models can thus be theoretically guaranteed to be equivariant up to discretization error. We incorporate equivariance into two state-of-the-art architectures for dynamics prediction, `ResNet` and `U-net` [224]. Below, we describe how we modify the convolution operation in these models for different symmetries G to form four $\texttt{Equ}_G$`-ResNet` and four $\texttt{Equ}_G$`-Unet` models.

The key to building equivariant networks is that the composition of equivariant functions is equivariant. Hence, if the maps between layers of a neural network are equivariant, then the whole network will be equivariant. Note that both the linear maps and activation functions must be equivariant. An important consequence of this principle is that the hidden layers must also carry a G-action. Thus, the hidden layers are not collections of scalar channels but vector-valued G-representations.

Rotational Equivariance To incorporate rotational symmetry, we model f using SO(2)-equivariant convolutions and activations within the `E(2)-CNN` framework of [232]. In practice, we use the cyclic group $G = C_n$ instead of $G = \mathrm{SO}(2)$ as for large enough n the difference is practically indistinguishable due to space discretization. We use powers of the regular representation $\rho = \mathbb{R}[C_n]^m$ for hidden layers. The representation $\mathbb{R}[C_n]$ has basis given by elements of C_n and C_n-action by permutation matrices. It has good descriptivity since it contains all irreducible representations of C_n, and it is compatible with any activation function applied channel-wise.

Uniform Motion Equivariance Uniform motion is part of Galilean invariance and is relevant to all non-relativistic physics modeling. For a vector field $X\colon \mathbb{R}^2 \to \mathbb{R}^2$ and vector $c \in \mathbb{R}^2$, uniform motion transformation is adding a constant vector field to the vector field $X(v)$, $T^{\mathrm{um}}_{\boldsymbol{c}}(X)(v) = X(v) + \boldsymbol{c}, \boldsymbol{c} \in \mathbb{R}^2$.

We make CNNs equivariant to uniform motion by conjugating the model with shifted input distribution. For each sliding local block in each convolutional layer, we shift the mean of input tensor to zero and shift the output back after convolution and activation function per sample. In other words, if the input is $\mathscr{P}_{b\times d_{in}\times s\times s}$ and the output is $\mathscr{Q}_{b\times d_{out}} = \sigma(\mathscr{P}\cdot K)$ for one sliding local block, where b is batch size, d is number of channels, s is the kernel size, and K is the kernel, then

$$\boldsymbol{\mu}_i = \mathrm{Mean}_{jkl}\left(\mathscr{P}_{ijkl}\right); \quad \mathscr{P}_{ijkl} \mapsto \mathscr{P}_{ijkl} - \boldsymbol{\mu}_i; \quad \mathscr{Q}_{ij} \mapsto \mathscr{Q}_{ij} + \boldsymbol{\mu}_i. \tag{5.4}$$

This will allow the convolution layer to be equivariant with respect to uniform motion. If the input is a vector field, we apply this operation to each element.

Proposition 1. *A residual block $f(\boldsymbol{x}) + \boldsymbol{x}$ is uniform motion equivariant if the residual connection*

f is uniform motion invariant.

Proof. We denote the uniform motion transformation by $\boldsymbol{c}$ by $T^{\mathrm{um}}_{\boldsymbol{c}}(\boldsymbol{w}) = \boldsymbol{w} + \boldsymbol{c}$. Let f be an invariant residual connection which is a composition of convolution layers and activation functions. Then we compute

$$\begin{aligned} f(T^{\mathrm{um}}_{\boldsymbol{c}}(\boldsymbol{w})) + T^{\mathrm{um}}_{\boldsymbol{c}}(\boldsymbol{w}) &= f(\boldsymbol{w}) + \boldsymbol{w} + \boldsymbol{c} \\ &= (f(\boldsymbol{w}) + \boldsymbol{w}) + \boldsymbol{c} \\ &= T^{\mathrm{um}}_{\boldsymbol{c}}(f(\boldsymbol{w}) + \boldsymbol{w}). \end{aligned}$$

as desired. □

By the proposition 1 above, within `ResNet`, residual mappings should be *invariant*, not equivariant, to uniform motion. That is, the skip connection $f^{(i,i+2)} = I$ is equivariant and the residual function $f^{(i,i+1)}$ should be invariant. Hence, for the first layer in each residual block, we omit adding the mean back to the output $\mathcal{Q}_{ij}$. In the case of `Unet`, when upscaling, we pad with the mean to preserve the overall mean.

Scale Equivariance Scale equivariance in dynamics is unique as the physical law dictates the scaling of magnitude, space and time simultaneously. This is very different from scaling in images regarding resolutions [240]. For example, the Navier-Stokes equations are preserved under a specific scaling ratio of time, space, and velocity given by the transformation

$$T_\lambda : \boldsymbol{w}(\boldsymbol{x}, t) \mapsto \lambda \boldsymbol{w}(\lambda \boldsymbol{x}, \lambda^2 t), \tag{5.5}$$

where $\lambda \in \mathbb{R}_{>0}$. We implement two different approaches for scale equivariance, depending on whether we tie the physical scale with the resolution of the data.

If the physical scale of the data is fixed, then scaling corresponds to a change in resolution and time step size. To achieve this, we replace the convolution layers with group correlation layers over the group $G = (\mathbb{R}_{>0}, \cdot) \ltimes (\mathbb{R}^2, +)$ of scaling and translations. In convolution, we translate a kernel K across an input $\boldsymbol{w}$ as such $\boldsymbol{v}(\boldsymbol{p}) = \sum_{\boldsymbol{q}\in\mathbb{Z}^2} \boldsymbol{w}(\boldsymbol{p}+\boldsymbol{q})K(\boldsymbol{q})$. The G-correlation upgrades this operation by both translating *and* scaling the kernel relative to the input,

$$\boldsymbol{v}(\boldsymbol{p}, s, \mu) = \sum_{\lambda\in\mathbb{R}_{>0}, t\in\mathbb{R}, \boldsymbol{q}\in\mathbb{Z}^2} \lambda \boldsymbol{w}(\lambda \boldsymbol{p} + \boldsymbol{q}, \lambda^2 t, \lambda\mu) K(\boldsymbol{q}, s, t, \lambda), \tag{5.6}$$

where s and t denote the indices of output and input channels respectively. We add an axis to the tensors corresponding the scale factor μ. Note that we treat the channel as a time dimension both with respective to our input and scaling action. As a consequence, as the number of channels increases in the lower layers of `Unet` and `ResNet`, the temporal resolution increases, which is analogous to temporal refinement in numerical methods [106, 133]. For the input $\tilde{\boldsymbol{w}}$ of first layer where $\tilde{\boldsymbol{w}}$ has no levels originally, $\boldsymbol{w}(p, s, \lambda) = \lambda \tilde{\boldsymbol{w}}(\lambda p, \lambda^2 s)$.

Our model builds on the methods of [240], but with important adaptations for the physical domain. Our implementation of group correlation Eqn. 5.6 directly incorporates the physical scaling law (5.5) of the system Eqn. 2.1.1. This affects time, space, and magnitude. (For heat, we drop the magnitude scaling.) The physical scaling law dictates our model should be equivariant to both up and down scaling and by any $\lambda \in \mathbb{R}_{>0}$. Practically, the sum is truncated to 7 different $1/3 \leq \lambda \leq 3$ and discrete data is continuously indexed using interpolation. Note (5.5) demands we scale *anisotropically*, i.e. differently across time and space.

Magnitude Equivariance We fix the resolution and scale the magnitude of the input by varying the discretization step size. An input $\boldsymbol{w} \in \mathscr{F}^k_{\mathbb{R}^2}$ with step size $\Delta_x(\boldsymbol{w})$ and $\Delta_t(\boldsymbol{w})$ can be scaled $\boldsymbol{w}' = T^{sc}_\lambda(\boldsymbol{w}) = \lambda \boldsymbol{w}$ by scaling the magnitude of vector alone, provided the discretization constants are now assumed to be $\Delta_x(\boldsymbol{w}') = 1/\lambda \Delta_x(\boldsymbol{w})$ and $\Delta_t(\boldsymbol{w}') = 1/\lambda^2 \Delta_t(\boldsymbol{w})$. We refer to this as *magnitude* equvariance hereafter.

To obtain magnitude equivariance, we divide the input tensor by the MinMax scaler (the maximum of the tensor minus the minimum) and scale the output back after convolution and activation per sliding block. We found that the standard deviation and mean L2 norm may work

Table 5.1. The RMSE and ESE of the `ResNet(Unet)` and `Equ-ResNets(Unets)` predictions on the original and transformed test sets of Rayleigh-Bénard Convection. `Augm` is `ResNet(Unet)` trained on additional samples applied with random transformations from the relevant symmetry group.

	Root Mean Square Error(10^3)					Energy Spectrum Errors				
	Orig	*UM*	*Mag*	*Rot*	*Scale*	*Orig*	*UM*	*Mag*	*Rot*	*Scale*
`ResNet`	0.67±0.24	2.94±0.84	4.30±1.27	3.46±0.39	1.96±0.16	0.46±0.19	0.56±0.29	0.26±0.14	1.59±0.42	4.32±2.33
`Augm`		1.10±0.20	1.54±0.12	0.92±0.09	1.01±0.11		1.37±0.02	1.14±0.32	1.92±0.21	1.55±0.14
`Equ`$_{\texttt{UM}}$	0.71±0.26	**0.71±0.26**				0.33±0.11	**0.33±0.11**			
`Equ`$_{\texttt{Mag}}$	0.69±0.24		**0.67±0.14**			0.34±0.09		**0.19±0.02**		
`Equ`$_{\texttt{Rot}}$	**0.65±0.26**			**0.76±0.02**		0.31±0.06			**1.23±0.04**	
`Equ`$_{\texttt{Scal}}$	0.70±0.02				**0.85±0.09**	0.44±0.22				**0.68±0.26**
`U-net`	0.64±0.24	2.27±0.82	3.59±1.04	2.78±0.83	1.65±0.17	0.50±0.04	0.34±0.10	0.55±0.05	0.91±0.27	4.25±0.57
`Augm`		0.75±0.28	1.33±0.33	0.86±0.04	1.11±0.07		0.96±0.23	0.44±0.21	1.24±0.04	1.47±0.11
`Equ`$_{\texttt{UM}}$	0.68±0.26	**0.71±0.24**				0.23±0.06	**0.14±0.05**			
`Equ`$_{\texttt{Mag}}$	0.67±0.11		**0.68±0.14**			0.42±0.04		**0.34±0.06**		
`Equ`$_{\texttt{Rot}}$	0.68±0.25			**0.74±0.01**		**0.11±0.02**			**1.16±0.05**	
`Equ`$_{\texttt{Scal}}$	0.69±0.13				**0.90±0.25**	0.45±0.32				**0.89±0.29**

as well but are not as stable as the MinMax scaler. Specifically,

$$\sigma_i = \text{MinMax}_{jkl}\left(\mathscr{P}_{ijkl}\right); \quad \mathscr{P}_{ijkl} \mapsto \mathscr{P}_{ijkl}/\sigma_i; \quad \mathscr{Q}_{ij} \mapsto \mathscr{Q}_{ij} \cdot \sigma_i. \tag{5.7}$$

5.2.2 Experiments on Simulated Rayleigh-Bénard Convection Dynamics

Data Description. Rayleigh-Bénard Convection occurs in a horizontal layer of fluid heated from below and is a major feature of the El Niño dynamics. The dataset comes from two-dimensional turbulent flow simulated using the Lattice Boltzmann Method [35] with Rayleigh number 2.5×10^8. We divide each 1792 $\times$ 256 image into 7 square subregions of size 256 $\times$ 256, then downsample to 64 $\times$ 64 pixels. To test the models' generalization ability, we generate additional four test sets : 1) *UM*: added random vectors drawn from $U(-1,1)$; 2) *Mag*: multiplied by random values sampled from $U(0,2)$; 3) *Rot*: randomly rotated by the multiples of $\pi/2$; 4) *Scale*: scaled by λ sampled from $U(1/5,2)$. Due to lack of a fixed reference frame, real-world data would be transformed relative to training data. We use transformed data to mimic this scenario.

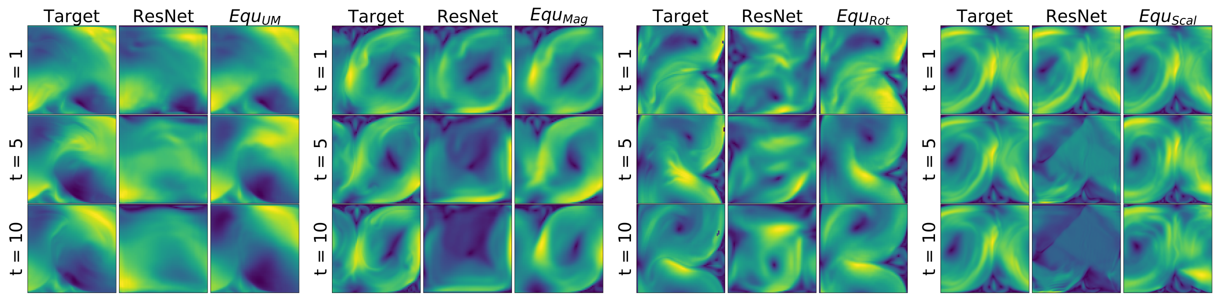

Figure 5.2. The ground truth and the predicted velocity norm fields $\|\boldsymbol{w}\|_2$ at time step 1, 5 and 10 by the `ResNet` and four `Equ-ResNets` on the four transformed test samples. The first column is the target, the second is `ResNet` predictions, and the third is predictions by `Equ-ResNets`.

Prediction Performance. Table 5.1 shows the prediction RMSE and ESE on the original and four transformed test sets by the non-equivariant `ResNet(Unet)` and four `Equ-ResNets(Unets)`. `Augm` is `ResNet(Unet)` trained on the augmented training set with additional samples with random transformations applied from the relevant symmetry group. The augmented training set contains additional transformed samples and is three times the size of the original training set. Each column contains the prediction errors by the non-equivariant and equivariant models on each test set. On the original test set, all models have similar RMSE, yet the equivariant models have lower ESE. This demonstrates that incorporating symmetries preserves the representation powers of CNNs and even improves models' physical consistency.

Table 5.2. Model Equivariance Errors

$\text{EE}_T(10^3)$	*UM*	*Mag*	*Rot*	*Scale*
`ResNets`	2.01	1.88	5.90	1.66
$\texttt{Equ}_{\texttt{ResNet}}$	0.0	0.0	1.19	0.58
`Unets`	1.07	0.20	1.55	1.81
$\texttt{Equ}_{\texttt{Unet}}$	0.0	0.0	0.79	0.48

On the transformed test sets, we can see that `ResNet(Unet)` fails, while `Equ-ResNets(Unets)` performs even much better than `Augm-ResNets(Unets)`. This demonstrates the value of equivariant models over data augmentation for improving generalization. Figure 5.2 shows the ground truth and the predicted velocity fields at time step 1, 5 and 10 by the `ResNet` and four `Equ-ResNets` on the four transformed test samples.

Table 5.3. Performance comparison on the transformed train and test sets.

	RMSE	ESE
`ResNet`	1.03±0.05	0.96±0.10
$\texttt{Equ}_{\texttt{UM}}$	**0.69±0.01**	**0.35±0.13**
`ResNet`	1.50±0.02	0.55±0.11
$\texttt{Equ}_{\texttt{Mag}}$	**0.75±0.04**	**0.39±0.02**
`ResNet`	1.18±0.05	1.21±0.04
$\texttt{Equ}_{\texttt{Rot}}$	**0.77±0.01**	**0.68±0.01**
`ResNet`	0.92±0.01	1.34±0.07
$\texttt{Equ}_{\texttt{Scal}}$	**0.74±0.03**	**1.02±0.02**

Generalization. In order to evaluate models' generalization ability with respect to the extent of distributional shift, we created additional test sets with different scale factors from $\frac{1}{5}$ to 1. Figure 5.3 shows ResNet and $\texttt{Equ}_{\texttt{Scal}}$-ResNet prediction RMSEs (left) and ESEs (right) on the test sets upscaled by different factors. We observed that $\texttt{Equ}_{\texttt{Scal}}$-ResNet is very robust across various scaling factors while ResNet does not generalize.

We also compare ResNet and Equ-ResNet when both train and test sets have random transformations from the relevant symmetry group applied to each sample. This mimics real-world data in which each sample has unknown reference frame. As shown in Table 5.3 shows Equ-ResNet outperforms ResNet on average by 34% RMSE and 40% ESE.

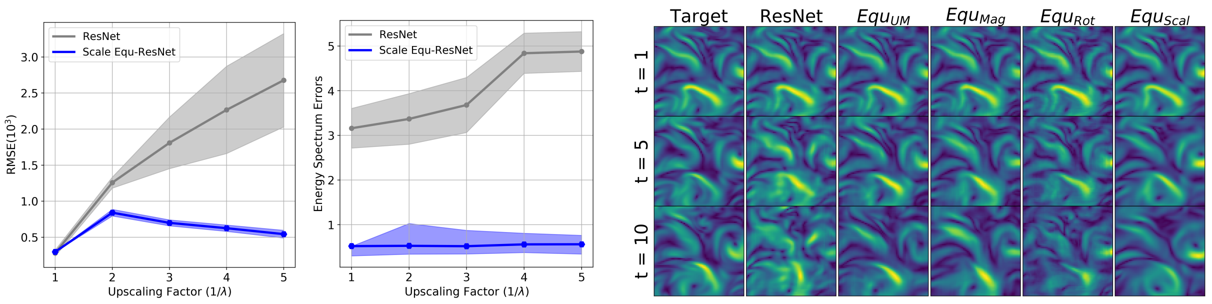

Figure 5.3. Left: Prediction RMSE and ESE over five runs of ResNet and $\texttt{Equ}_{\texttt{Scal}}$-ResNet on the Rayleigh-Bénard Convection test set upscaled by different factors. Right: The ground truth and predicted ocean currents $\|\boldsymbol{w}\|_2$ by ResNet and four Equ-ResNets on the test set of future time.

5.2.3 Experiments on Real World Ocean Dynamics

Data Description. We use the reanalysis ocean current velocity data generated by the NEMO ocean engine [143].[1] We selected an area from each of the Atlantic, Indian and North Pacific Oceans from 01/01/2016 to 08/18/2017 and extracted 64×64 sub-regions for our experiments. The corresponding latitude and longitude ranges for the selected regions are (-44∼-23, 25∼46), (55∼76, -39∼-18) and (-174∼-153, 5∼26) respectively. We not only test all models on the future data but also on a different domain (-180∼-159, -40∼-59) in South Pacific Ocean from 01/01/2016 to 12/15/2016.

[1]The data are available at https://resources.marine.copernicus.eu/?option=com_csw&view=details&product_id=GLOBAL_ANALYSIS_FORECAST_PHY_001_024

Prediction Performance. Table 5.4 shows the RMSE and ESE of ResNets(Unets), and equivariant Equ-ResNets(Unets) on the test sets with different time range and spatial domain from the training set. All the equivariant models outperform the non-equivariant baseline on RMSE, and Equ$_{\mathtt{Scal}}$-ResNet achieves the lowest RMSE. For ESE, only the Equ$_{\mathtt{Mag}}$-ResNet(Unet) is worse than the baseline. Also, it is remarkable that the Equ$_{\mathtt{Rot}}$ models have significantly lower ESE than others, suggesting that they correctly learn the statistical distribution of ocean currents.

Table 5.4. Prediction RMSE and ESE comparison on the two ocean currents test sets.

	RMSE		ESE	
	Test$_{time}$	*Test*$_{domain}$	*Test*$_{time}$	*Test*$_{domain}$
ResNet	0.71±0.07	0.72±0.04	0.83±0.06	0.75±0.11
Augm$_{\mathtt{UM}}$	0.70±0.01	0.70±0.07	1.06±0.06	1.06±0.04
Augm$_{\mathtt{Mag}}$	0.76±0.02	0.71±0.01	1.08±0.08	1.05±0.8
Augm$_{\mathtt{Rot}}$	0.73±0.01	0.69±0.01	0.94±0.01	0.86±0.01
Augm$_{\mathtt{Scal}}$	0.97±0.06	0.92±0.04	0.85±0.03	0.95±0.11
Equ$_{\mathtt{UM}}$	0.68±0.06	0.68±0.16	0.75±0.06	0.73±0.08
Equ$_{\mathtt{Mag}}$	0.66±0.14	**0.68±0.11**	0.84±0.04	0.85±0.14
Equ$_{\mathtt{Rot}}$	0.69±0.01	0.70±0.08	**0.43±0.15**	**0.28±0.20**
Equ$_{\mathtt{Scal}}$	**0.63±0.02**	0.68±0.21	0.44±0.05	0.42±0.12
U-net	0.70±0.13	0.73±0.10	0.77±0.12	0.73±0.07
Augm$_{\mathtt{UM}}$	0.68±0.02	0.68±0.01	0.85±0.04	0.83±0.04
Augm$_{\mathtt{Mag}}$	0.69±0.02	0.67±0.10	0.78±0.03	0.86±0.02
Augm$_{\mathtt{Rot}}$	0.79±0.01	0.70±0.01	0.79±0.01	0.78±0.02
Augm$_{\mathtt{Scal}}$	0.71±0.01	0.77±0.02	0.84±0.01	0.77±0.02
Equ$_{\mathtt{UM}}$	0.66±0.10	0.67±0.03	0.73±0.03	0.82±0.13
Equ$_{\mathtt{Mag}}$	**0.63±0.08**	**0.66±0.09**	0.74±0.05	0.79±0.04
Equ$_{\mathtt{Rot}}$	0.68±0.05	0.69±0.02	**0.42±0.02**	0.47±0.07
Equ$_{\mathtt{Scal}}$	0.65±0.09	0.69±0.05	0.45±0.13	**0.43±0.05**

Comparison with Data Augmentation. We also compare Equ-ResNets(Unets) ResNets(Unets) that are trained with data-augmentation (Augm) in Table 5.4. In all cases, equivariant models outperforms the baselines trained with data augmentation. We find that data augmentation sometimes improves slightly on RMSE but not as much as the equivariant models. And, in fact, ESE is uniformly worse for models trained with data augmentation than even the baselines. In contrast, the equivariant models have much better ESE than the baselines with or without augmentation. We believe data augmentation presents a trade-off in learning. Though the model may be less sensitive to the various transformations we consider, we need to train bigger models longer on many more samples. The models may not have enough capacity to learn the symmetry from the augmented data and the dynamics of the fluids at the same time. By comparison, equivariant architectures do not have this issue.

Figure 5.3 shows the ground truth and the predicted ocean currents at time step $1, 5, 10$ by different models. We can see that equivariant models' predictions are more accurate and contain more details than the baselines. Thus, incorporating symmetry into deep learning models can improve the prediction accuracy of ocean currents.

5.3 Approximately Equivariant Networks for Imperfectly Symmetric Dynamics

5.3.1 Approximate Symmetry

Existing equivariant networks assume perfect symmetry in the data. The network is approximating a function that is strictly invariant or equivariant under a given group action. However, many real-world dynamics may not satisfy the strict equivariance. Since many of the underlying dynamics will contain symmetry, the resulting system may still be approximately equivariant. For example, while the heat equation itself is fully rotationally symmetric, in practice, imperfections in the thickness of the metal or compositional of the metal may lead to imperfect symmetry, as shown in Figure 5.4.

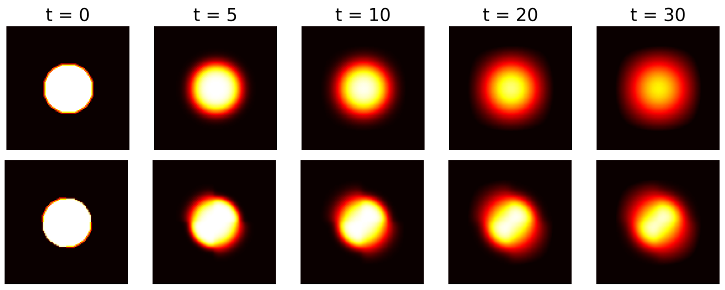

Figure 5.4. Simulated diffusion of heat in a metal plate with (top) uniform diffusion coefficient resulting in perfect symmetry and (bottom) slightly varying diffusion coefficient resulting in approximate symmetry.

In turbulence modeling, even though the governing equations of turbulence satisfy many different symmetries such as scale-invariance [83], varying external forces, certain boundary conditions as well as the presence of missing values would break these symmetries to varying degrees. This significantly hinders the potential applications of equivariant networks.

Figure 5.5 left shows that when symmetry is a good inductive bias, prediction performance increases as equivariance or invariance is imposed in the model. But real-world data is very rarely perfectly symmetric, and so relaxing the strict constraint in equivariant networks to balance

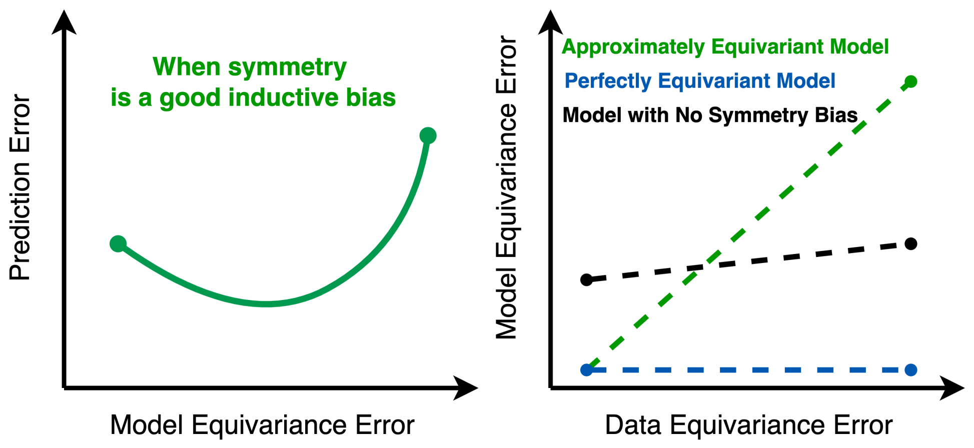

Figure 5.5. Left: Relaxing the strict constraint in equivariant networks to balance inductive bias and expressivity can further improve predictive performance. **Right**: An ideal model should be approximately equivariant and automatically learn the correct amount of symmetry in the data.

inductive bias and expressivity can further improve predictive performance. The right figure suggests that highly flexible models have trouble achieving zero equivariance error without the guide of appropriate symmetry biases when the data is symmetric. Perfectly equivariant models maintain zero equivariance error, which is overly restricted when data is not perfectly symmetric. An ideal model for real world dynamics should be approximately equivariant and automatically learn the correct amount of symmetry in the data.

Relaxing the rigid assumption in equivariant networks to balance inductive bias and expressivity in deep learning has been the pursuit of a few recent works. For example, [54] showed that spatial invariance may be overly restrictive, and relaxing the spatial weight sharing in the standard convolution can improve image classification accuracy. [42] enforce convolutional inductive bias in self-attention layer at *initialization* to improve vision Transformers. Residual Pathway Priors [60] convert hard architectural constraints into soft priors by placing a higher likelihood on the "residual". The residual explains the difference between the structure in the data and the inductive bias encoded by an equivariant model. [222] proposed Lift Expansion which factorizes the data into equivariant and non-equivariant components and models them jointly. Despite progress, a formal definition of approximate symmetry does not exist. While existing research focuses on translation symmetry, the rich groups of symmetry in high-dimensional dynamics learning problems are unexplored.

In this section, we first define approximate symmetry. It gives rise to a new class of approximately equivariant networks that avoid stringent symmetry constraints while maintaining favourable inductive biases for learning. Specifically, we generalizes the weight relaxation scheme originally proposed by [54]. We study three symmetries that are common in dynamics: rotation $\mathrm{SO}(2)$, scaling $\mathbb{R}_{>0}$, and Euclidean $E(2)$. For group convolution, we relax the weight-sharing scheme by expressing the kernel as a weighted combination of multiple filter banks. For steerable CNNs, we introduce dependencies on the input to the kernel basis. We apply our approximate symmetry networks to the challenging problem of forecasting fluid flow and observe significant improvements for both synthetic and real-world datasets.

Below, we give the formal definition of approximate symmetry:

Definition 1 (Approximate Equivariance)**.** Let $f\colon X \to Y$ be a function and G be a group. Assume that G acts on X and Y via representations ρ_X and ρ_Y. We say f is ε-approximately G-equivariant if for any $g \in G$,

$$\|f(\rho_X(g)(x)) - \rho_Y(g)f(x)\| \leq \varepsilon.$$

Note that strictly equivariant functions are $\varepsilon = 0$ approximately equivariant.

5.3.2 Relaxed Equivariant CNNs

Symmetry in equivariant networks is enforced by strict constraints on the weights. Here we propose relaxing weight-sharing and weight-tying to model approximate symmetries. [54] showed that relaxing the spatial weight sharing in the standard convolution neural nets can improve image classification accuracy. Whereas 2D convolutions are shift equivariant, this relaxed 2D convolution is only approximately equivariant. Our method generalizes this approach to other symmetry groups including rotation $\mathrm{SO}(2)$, scaling $\mathbb{R}_{>0}$, and Euclidean $E(2)$. Specifically, we relax the strict weight-sharing and -tying constraints in both group convolution and steerable CNNs.

Relaxing Weight Sharing in Group Convolution The G-equivariance of group convolution results from the shared kernel $\Psi(g^{-1}h)$. To relax this and consequently relax the G-equivariance, we replace the single kernel Ψ with a set of kernels $\{\Psi_l\}_{l=1}^{L}$. We define the new kernel Ψ as a linear combination of Ψ_l with coefficients that vary with h and thus introduce symmetry-breaking dependence on the specific pair (g,h),

$$\Psi(g,h) = \sum_{l=1}^{L} w_l(h)\Psi_l(g^{-1}h). \tag{5.8}$$

We define relaxed group convolution by multiplication with Ψ as such

$$f\tilde{\star}_G\Psi](g) = \sum_{h\in G} f(h)\Psi(g,h) = \sum_{h\in G}\sum_{l=1}^{L} f(h)w_l(h)\Psi_l(g^{-1}h). \tag{5.9}$$

By varying the number of kernels L, we can control the degree of relaxation of equivariance. The weights $w_i(h) \in \mathbb{R}$ and the kernels $\Psi_l(g^{-1}h) \in \mathbb{R}^{c_{\text{out}}\times c_{\text{in}}}$ can be learnt from data. Relaxed group convolution reduces to group convolution and is fully equivariant if and only if $g_1^{-1}h_1 = g_2^{-1}h_2$ implies $\Psi(g_1,h_1) = \Psi(g_2,h_2)$. In particular, this occurs if $w_l(h_1) = w_l(h_2)$ for all $h_1,h_2 \in G$ and for all l.

In dynamics learning, we consider inputs which are velocity vectors. To apply group convolution over the discrete rotation group C_n, we first lift these to feature maps $f\colon C_n \to \mathbb{R}$ as described in Table 1 in [221]. Given $v = (a,b) \in \mathbb{R}^2$, for $i \in C_n$ we define $f(i) = ca\cos(2\pi i/n) + cb\sin(2\pi i/n)$ where $c \in \mathbb{R}$ is a trainable weight. This amounts to mapping the irreducible ρ_1 representation of C_n to the regular representation. This process can also be extended to lifting velocity fields to features $f\colon C_4 \ltimes (\mathbb{Z}^2,+) \to \mathbb{R}$ over the group of discrete rotations and translations. As an additional advantage, the feature maps f are compatible with element-wise non-linearity.

Relaxed G-Steerable 2D Convolution Although relaxed group convolution has the advantage that we do not need to precompute an equivariant kernel basis, it is limited to discrete (or compact) groups and is not efficient when the group order is large. Thus, we also propose relaxed steerable convolutions.

We first write out explicitly the formula for G-steerable 2D convolution described by (5.3) in order to clarify the distinction with the relaxed convolution. Let $\{\Phi_l\}_{l=1}^{L}$ be an equivariant kernel basis of L non-trainable kernels that satisfy (5.3) for given input and output representations ρ_{in} and ρ_{out}. Denote $K = \{-k, \dots, k\}$. Let the input signal be $f_{\text{in}} \colon \mathbb{Z}^2 \to \mathbb{R}^{c_{\text{in}}}$ and predetermined equivariant kernels $\Phi_l \colon K^2 \to \mathbb{R}^{c_{\text{out}} \times c_{\text{in}}}$ and trainable weights $w \in \mathbb{R}^{c_{\text{out}} \times c_{\text{in}} \times L}$. Then (non-relaxed) steerable convolution defines output $f_{\text{out}} = f_{\text{in}} \star_{\mathbb{R}^2} \Phi \colon \mathbb{Z}^2 \to \mathbb{R}^{c_{\text{out}}}$

$$f_{\text{out}}(\mathbf{x}) = \sum_{\mathbf{y} \in \mathbb{Z}^2} \sum_{l=1}^{L} (w_l \odot \Phi_l(\mathbf{y})) f_{\text{in}}(\mathbf{x} + \mathbf{y}) \tag{5.10}$$

for $\mathbf{x} \in \mathbb{Z}^2$ a position in the input. Here $\odot$ denotes element-wise product in $\mathbb{R}^{c_{\text{out}} \times c_{\text{in}}}$. The index $\mathbf{y}$ refers to spatial locations in the kernel.

We relax (5.10) and break symmetry by introducing dependence on $\mathbf{y}$ into the weights w. Since w is freely trainable, this breaks the strict positional dependence imposed by (5.3). Since the same weights w are shared across all kernel basis elements Φ_l, this is not as relaxed as 2-D convolution with no constraints. Formally, let $w \colon K^2 \to \mathbb{R}^{c_{\text{out}} \times c_{\text{in}} \times L}$ be the weights, we define relaxed steerable convolution $f_{\text{in}} \hat{\star}_H \Phi$ by

$$f_{\text{out}}(\mathbf{x}) = \sum_{\mathbf{y} \in \mathbb{Z}^2} \sum_{l=1}^{L} (w_l(\mathbf{y}) \odot \Phi_l(\mathbf{y})) f_{\text{in}}(\mathbf{x} + \mathbf{y}). \tag{5.11}$$

When G is a rotation group and $k > 0$, we can define $w_l(\mathbf{y}) = w_l(\theta)$, where $\theta = \text{arctan2}(\mathbf{y})$. Since the weight depends only on the angle of vector $\mathbf{y}$, we use fewer parameters. To avoid the model learning being over-relaxed, we initialize $w_l(\mathbf{y})$ equally for every $\mathbf{y}$ and penalize the value differences in $w_l(\mathbf{y})$ during training, which we describe in Section 5.3.2. Moreover, we can also relax the translation equivariance at the same time by allowing

$w\colon \mathbb{Z}^2 \times K^2 \to \mathbb{R}^{c_{\text{out}} \times c_{\text{in}} \times L}$ to vary across input spatial dimensions as well

$$f_{\text{out}}(\mathbf{x}) = \sum_{\mathbf{y}\in\mathbb{Z}^2} \sum_{l=1}^{L} (w_l(\mathbf{x},\mathbf{y}) \odot \Phi_l(\mathbf{y})) f_{\text{in}}(\mathbf{x}+\mathbf{y}). \tag{5.12}$$

However, the above equation has too large a trainable weight space to be practical. Thus, we use a low-rank factorization of w,

$$w_l(\mathbf{x},\mathbf{y}) = \sum_{r=1}^{R} a_r(\mathbf{x}) b_{r,l}(\mathbf{y})$$

where $a_r\colon \mathbb{Z}^2 \to \mathbb{R}$ and $b_{r,l} : K^2 \to\colon \mathbb{R}^{c_{\text{out}} \times c_{\text{in}}}$. Note this equation effectively combines relaxed translation group convolution with our method for relaxed rotation or scale steerable kernels.

Soft Equivariance Regularization To encourage equivariance and avoid the model becoming over-relaxed, we add regularization terms to the loss function on the symmetry-independent weights w during training. In the case of relaxed group convolution, we add the following regularizer to constrain w in (5.8),

$$L_{\text{gconv}} = \alpha \sum_{i=1}^{L} \sum_{g,h\in G} \|w_i(h) - w_i(g)\|.$$

For the relaxed steerable convolution, we impose the following loss term to prevent the $w\colon K^2 \to \mathbb{R}^{c_{\text{out}} \times c_{\text{in}} \times L}$ in (5.11) from varying too much across the kernel spatial domains,

$$L_{\text{sconv}} = \alpha \left(\left\| \frac{\partial w(m,n)}{\partial m} \right\| + \left\| \frac{\partial w(m,n)}{\partial n} \right\| \right).$$

5.3.3 Experiments on Synthetic Smoke Plumes

Baselines We compare it with several state-of-the-art (SoTA) methods from those with no symmetry bias to perfect symmetry, and SoTA approximately symmetric models.

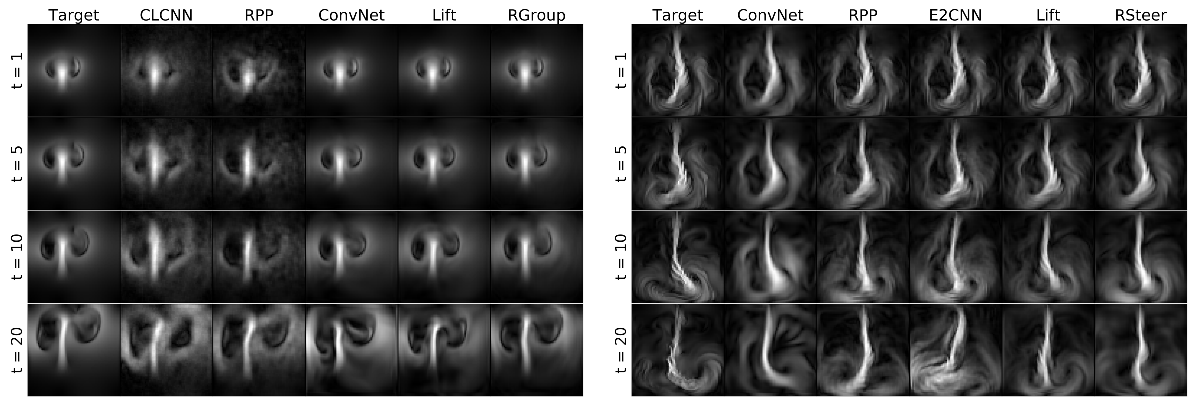

Figure 5.6. Target (ground truth) and model predictions comparison at time step 1, 5, 10, 20 for smoke simulation with approximate translation (left) and rotation (right) symmetries.

- `MLP`: multi-layer perceptrons, a non-equivariant baseline with a weaker inductive bias than convolution neural nets.
- `ConvNet`: standard convolutional neural nets that have the full translation symmetry.
- `Equiv`: fully equivariant convolutional models. It is same as `ConvNet` for translation symmetry. We use `E2CNN` [232] for rotation and `SESN` [202] for scaling symmetry.
- `Rpp` [60]: Residual Pathway Priors, a SOTA approximate equivariance model that sums up the outputs from an equivariant and non-equivariant layer in each block while posing constraints on the non-equivariant layer in loss function. We use the combination of `MLP` and `ConvNet` for translation, `ConvNet` and `E2CNN` for rotation, and `ConvNet` and `SESN` for scaling.
- `Combo`: models that start with non-equivariant layers followed by equivariant layers. The early layers of the model map observations with approximate symmetries to a space with explicit symmetry actions.
- `CLCNN`: locally connected neural networks with equivariance constraints imposed in the loss function.

- `Lift` [222]: Lift Expansion for modeling partial symmetry. The model uses a non-equivariant encoder that is tiled across the equivariant dimensions of the feature map as additional channels in equivariant neural nets. This method can also model approximate symmetry when both the non-equivariant encoder and the main equivariant backbone are fed with the same input.

EMLP [62] is also a SoTA equivariant model, but it cannot handle large data like images as stated in the paper, so we do not include it as a baseline.

Experiments Setup All models are trained to make a forward prediction of raw velocity fields given historical data. For all datasets, we use a sliding window approach to generate samples of sequences. We perform a grid hyperparameters search, including learning rate, batch size, hidden dimension, number of layers, and number of steps of prediction errors for training. We also tune the number of filter banks for group convolution-based models and the coefficient of weight constraints for relaxed weight-sharing models. The input length is fixed as 10. In the meanwhile, we make sure the total number of trainable parameters for every model is fewer than 10^7 for a fair comparison.

We test all models under two scenarios. For *test-future*, we train and test on the same tasks but different time steps. For *test-domain*, we train and test on different simulations/regions with an 80%-20% split. All models are trained to make the next step prediction given the previous steps as input. The first scenario evaluates the models' ability to extrapolate into the future for the same task. The second scenario estimates the capability of the models to generalize across different simulations/regions. We forecast in an autoregressive manner to generate multi-step predictions during inference and evaluate them based on 20-step prediction RMSEs. All results are averaged over 3 runs with random initialization.

Data Description: The synthetic 64×64 2-D smoke datasets are generated by PhiFlow [82] and contain smoke simulations with different initial conditions and external forces. We explore three symmetry groups: 1) *Translation*: 35 smoke simulations with different inflow

Table 5.5. Prediction RMSE on three synthetic smoke plume datasets with approximate symmetries. *Future* means testing data lies in the future time of the training data. *Domain* means training and test data are from different spatial domains.

Model		`MLP`	`Conv`	`Equiv`	`Rpp`	`Combo`	`CLCNN`	`Lift`	`RGroup`	`RSteer`
Translation	Future	1.56±0.08	—	0.94±0.02	0.92±0.01	1.02±0.02	0.92±0.01	0.87±0.03	**0.71±0.01**	—
	Domain	1.79±0.13	—	0.68±0.05	0.93±0.01	0.98±0.01	0.89±0.01	0.70±0.00	**0.62±0.02**	—
Rotation	Future	1.38±0.06	1.21±0.01	1.05±0.06	0.96±0.10	1.07±0.00	0.96±0.05	0.82±0.08	0.82±0.01	**0.80±0.00**
	Domain	1.34±0.03	1.10±0.05	0.76±0.02	0.83±0.01	0.82±0.02	0.84±0.10	0.68±0.09	0.73±0.02	**0.67±0.01**
Scaling	Future	2.40±0.02	0.83±0.01	0.75±0.03	0.81±0.09	0.78±0.04	1.03±0.01	0.85±0.01	0.76±0.04	**0.70±0.01**
	Domain	1.81±0.18	0.95±0.02	0.87±0.02	0.86±0.05	0.85±0.01	0.83±0.05	0.77±0.02	0.86±0.12	**0.73±0.01**

positions. We also horizontally split the entire domain into two separate sub-domains that have different buoyant forces. Although the inflow positions are translation equivariant, the closed boundary and the two different buoyant forces would break the equivariance. 2) *Rotation*: 40 simulations with different inflow positions and buoyant forces. Both inflow location and the direction of buoyant forces have perfect rotation symmetry with respect to C_4 group, but the buoyancy factor varies with the inflow positions to break the rotation symmetry. 3) *Scaling*: It contains 40 simulations generated with different spatial step Δx and temporal step Δt. And the buoyant force varies across the simulations to break the scaling symmetry.

Prediction Performance: Table 5.5 shows prediction RMSEs on three synthetic smoke plume datasets with different approximate symmetries by our proposed models and baselines. Since CNNs are already translation equivariant, so we do not have relaxed steerable model for translation. We can see, on the approximate translation dataset, our relaxed group convolution (`RGroup`) significantly outperforms baselines on both test sets. And for rotation and scaling, the proposed relaxed steerable convolution (`RSteer`) always achieves the lowest RMSEs and `RGroup` can outperform most baselines.

Figure 5.6 shows the target and predictions of our proposed models and the best baselines at time step 1, 5, 10, 20 for smoke simulation with approximate translation (left) and rotation (right) symmetries. From the shape and frequency of the flows, predictions from our approximately equivariant models are much closer to the target than the baselines. Moreover, we can see that `E2CNN` predicts the smoke flowing in the wrong direction at time step 20, which could be a consequence of over-constraining from equivariance.

5.3.4 Learning Different Levels of Equivariance

We use PhiFlow [82] to create 10 small smoke plume datasets with different levels of rotational equivariance. In each dataset, both inflow location and the direction of buoyant forces have perfect rotation symmetry with respect to C_4 group. By varying the amount of difference in buoyant force between simulations with different inflow positions, we can control the amount of equivariance error in the data. The data equivariance error of each dataset is the mean absolute error between the simulations after all being rotated back to the same inflow position.

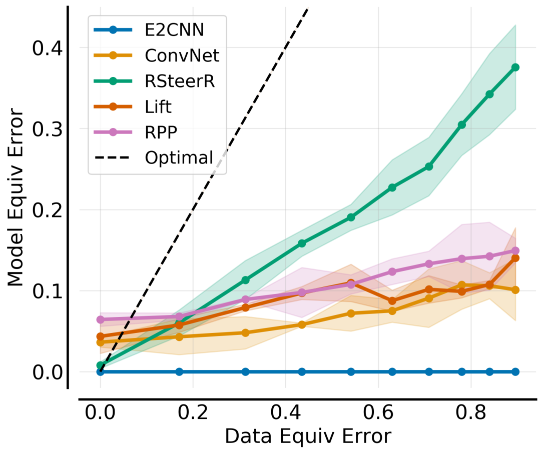

Figure 5.7. Model equivariance errors vs. data equivariance errors on synthetic smoke plume with different levels of rotational equivariance. We see that our `RSteer` model can always learn the correct level of equivariance in different datasets while non-equivariant `ConvNet` and fully equivariant `E2CNN` cannot.

We trained two-layer `ConvNet`, `E2CNN` and our relaxed rotation equivariant steerable convolution `RSteerR` on these 10 datasets. We calculate the equivariance error of each well-trained model based on Definition 1, where $G = C_4$ and the norm is L1 norm. From Figure 5.7, we see that `E2CNN` always has zero equivariance error due to the overly restrictive symmetry constraint even if the data does not have perfect symmetry. And our `RSteerR` can always correctly learn different levels of equivariance in the data while `ConvNet` cannot. Since the prediction errors are not zeros, the equivariance errors in the model and data are not the same.

Table 5.6. Prediction RMSEs on experimental jet flow data for different models. Proposed `RSteer` and `RSteer` are designed for the corresponding assumed symmetry group.

Model	`Conv`	`Lift`	`RGroup`	`E2CNN`	`Lift`	`RSteer`	`SESN`	`Rpp`	`RSteer`
	Translation			Rotation			Scaling		
Future	0.22±0.06	0.17±0.02	0.15±0.00	0.21±0.02	0.18±0.02	0.17±0.01	0.15±0.00	0.16±0.06	0.14±0.01
Domain	0.23±0.06	0.18±0.02	0.16±0.01	0.27±0.03	0.21±0.04	0.16±0.01	0.16±0.01	0.16±0.07	0.15±0.00

This experiment demonstrates that our proposed methods based on relaxed weight sharing can learn the *correct* amount of inductive biases from data while avoiding the stringent symmetry constraints.

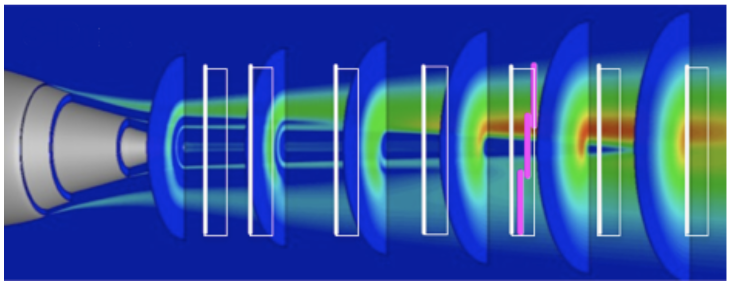

Figure 5.8. Visualization of axial measurement locations. White boxes show fields of view acquired on streamwise planes at the jet centerline for multistream flows. There are three vertical stations at each axial location/white box, as illustrated by the pink lines. Figure taken from [19].

5.3.5 Experiments on Experimental Jet Flow Data

Data Description. We use the real experimental data of 2D turbulent velocity in multi-stream jets from NASA that is measured using the time-resolved particle image velocimetry [19]. Figure 5.8 visualizes the measurement system of the jet flow. White boxes show fields of view acquired on the streamwise plane at the jet centerline for multistream flows. There are three vertical stations at each axial location/white box, as illustrated by the pink lines. In other words, the dataset was acquired by 24 different stations at different locations. Since the data collected at the different locations are not acquired concurrently, we do not have the complete velocity fields of entire jet flows at each time step. Thus, we trained and tested models on 24 62×23 sub-regions of jet flows.

Prediction Performance. We compare three equivariant models, three best-performing approximately equivariant baselines in the previous experiment as well as our proposed relaxed steerable convolution and relaxed group convolution. Table 5.6 shows prediction RMSEs on the jet flow dataset and we group the results by each symmetry in the table. For each symmetry, our models based on relaxed weight sharing achieve lower errors than not only the fully equivariant model but also approximately equivariant baselines. We also experimented with combining relaxed translation group convolution with relaxed rotation and scale steerable convolution, which correspond to `RSteerTR` and `RSteerTS` respectively in the table. We observe that `RSteerTR` outperforms both `RGroup` with relaxed group convolution and `RSteer` with relaxed steerable convolution. This implies relaxing more than one equivariance constraint can potentially lead to even better performance. Figure 5.9 visualizes the target and mean absolute errors between model jet flow predictions and the ground truth (target), and we can see that our relaxed steerable CNNs achieve the lowest errors.

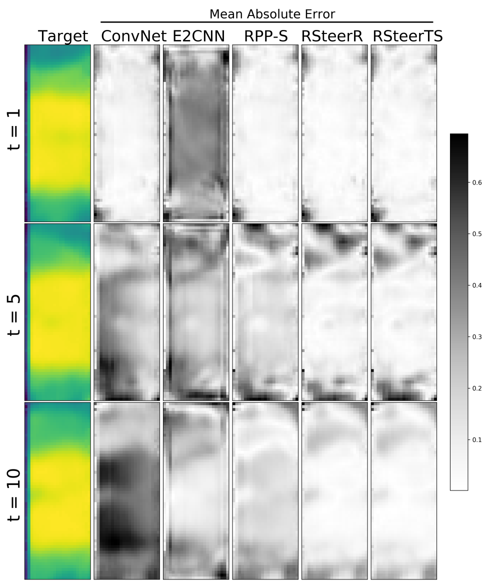

Figure 5.9. Target jet flow velocity norm fields and the prediction errors (MAE) of different models over 10 time steps.

5.4 Data Augmentation vs. Equivariant Networks: A Theory of Generalization on Dynamics Forecasting

In this work, we present a theory of generalization for dynamics forecasting, where the data are non-stationary and non-mixing time series. We theoretically analyze and compare the generalization strength of data augmentation versus equivariant networks. We show that when the underlying dynamics is symmetric, equivariant networks achieve a tighter generalization bound than data augmentation. Furthermore, when the symmetries in the data are only approximate, the generalization bound for approximately equivariant networks [228] is further improved.

5.4.1 Statistical Learning Theory for Time Series.

We consider forecasting deterministic dynamics where the learner receives N observed time series $\{X^{(1)}, ..., X^{(N)}\}$ with length T of a dynamical system [227, 225]. Each time series $X^{(i)}$ is a sample from a dynamical system where the system parameters are drawn i.i.d. from a given distribution. Even though the system parameters are independently sampled, each time series can be highly non-stationary and exhibit complex dependencies.

Denote $Z_t^{(i)} = (X_{t-k-1:t-1}^{(i)}, X_t^{(i)}) \in \mathscr{X}^k \times \mathscr{X}$ as a training sample (a subsequence of time series i at time t). $X_{t-k-1:t-1}^{(i)}$ and $X_t^{(i)}$ are the input and output of a forecasting model. For a loss function $\mathscr{L} : \mathscr{X} \times \mathscr{X} \to [0, \infty)$ and a hypothesis set $\mathscr{F}$ of functions that map from $\mathscr{X}^k$ to $\mathscr{X}$, we want to minimize its empirical risk:

$$R_n(\theta) = \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} q_t \mathscr{L}(f_\theta(X_{t-k-1:t-1}^{(i)}), X_t^{(i)})$$

where θ represent the parameters in f. We use $L(\theta, Z_t^{(i)})$ to denote $\mathscr{L}(f_\theta(X_{t-k-1:t-1}^{(i)}), X_t^{(i)})$ for simplicity.

Note that $q_1, .., q_T$ are real numbers, which in the standard statistical learning scenarios are chosen to be all equal to $\frac{1}{T}$. We follow the time series forecasting setting in [119]. For non-stationary dynamics, different Z_t may follow different distributions, and thus distinct weights could be assigned to the errors made on different sample points, depending on their relevance to forecasting the future Z_{T+1}. The learning objective is to find a θ that achieves a small test error, $\mathbb{E}L(\theta, Z_{T+1})$.

To derive the generalization bound, [15] and [174] generalizes the classic Rademacher Complexity [68] to time series learning, as defined below,

Definition 2 (Sequential Rademacher Complexity, [15, 174])**.** Given a function class $\mathscr{G} \subset \mathbb{R}^{\mathscr{Z}}$, we define the sequential Rademacher complexity of class $\mathscr{G}$ as:

$$\mathscr{R}_T^{sq}(\mathscr{G}) = \mathbb{E}_{\boldsymbol{z}}\mathbb{E}_{\boldsymbol{\sigma}}[\sup_{g\in\mathscr{G}}\sum_{t=1}^{T}\sigma_t q_t g(\boldsymbol{z}_t(\boldsymbol{\sigma}))]$$

where $\boldsymbol{z}$ is a real-valued complete binary tree that is a sequence $(z_1,...,z_T)$ of T mappings $z_t : \{\pm 1\}^{t-1} \to \mathbb{R}$ for $t \in [1,...,T]$, and $\boldsymbol{\sigma}$ is a sequence of Rademacher random variables, which is also a path in the tree $\boldsymbol{\sigma} = (\sigma_1,...,\sigma_{T-1}) \in \{\pm 1\}^{T-1}$.

In our forecasting setting, the sequential Rademacher complexity of a loss class can be defined more specifically as:

$$\mathscr{R}_T^{sq}(L \circ \Theta) = \mathbb{E}_{\boldsymbol{\sigma}}\left[\sup_{\theta\in\Theta}\frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\sigma_t q_t L(\theta, Z_t^{(i)}))\right]$$

5.4.2 Equivariance and Invariance.

Based on Def 2 and Def 1, we further define equivariance error, which quantifies the amount of symmetry the function f contains.

Definition 3 (Equivariance Error)**.** Let $f\colon X \to Y$ be a function and G be a group. Assume that G acts on X and Y via representation ρ_{in} and ρ_{out}. Then the **equivariance error** of f is

$$\|f\|_{EE} = \sup_{x,g} \|f(\rho_{\text{in}}(g)(x)) - \rho_{\text{out}}(g)f(x)\|.$$

For strictly equivariant functions, we have $\varepsilon = 0$. But for real-world dynamics, the symmetry is often approximately equivariant, defined below: Several recent work have designed approximately equivariant networks [228, 215, 60] to learn the approximate functions. In this work, we assume the equivariance errors of trained approximately equivariant models is less or equal to the true data equivariance errors.

Data Aug. Introduces Symmetry. Consider a finite group G that acts on the observed time series, we assume that for any $g \in G$, there is a certain amount of symmetry in the distribution, that is $Z_t^{(i)} \approx_d gZ_t^{(i)}, Z_t^{(i)} \sim \mathbb{P}$. We assume the group transformations are norm-preserving, i.e. $\|g\| = 1 \; \forall g \in G$, such as rotation and translation.

Definition 4 (Data Augmentation)**.** Given a finite group G, we assume the augmented samples are the original samples applied with transformations uniformly sampled from the group. In other words, for every sample $Z_t^{(i)}$ in the original training set, we add samples $\{gZ_t^{(i)}, g \in G\}$. Then the augmented training set is the $|G|$ times bigger than the original training set.

We derive generalization bounds for data augmentation and equivariant networks. We show that the strictly equivariant networks can outperform data augmentation. When the underlying dynamics are approximately symmetric, approximately equivariant estimator can outperform both data augmentation estimator and strictly equivariant networks.

5.4.3 Population and Empirical Risk Minimizers

We first define the population and the empirical risk minimizers for data augmentation, perfectly equivariant models and approximately equivariant models based on the dynamic forecasting setting defined in the previous section.

- Population minimizer: $\theta^* = \text{argmin}_{\theta \in \Theta} \mathbb{E}[L(\theta, Z)]$
- Empirical minimizer:

$$\hat{\theta}_n = \text{argmin}_{\theta \in \Theta} \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} q_t L(\theta, Z_t^{(i)})$$

- Empirical minimizer for data augmentation:

$$\begin{aligned} \hat{\theta}_G &= \text{argmin}_{\theta \in \Theta} \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} q_t \mathbb{E}_G[L(\theta, gZ_t^{(i)})] \\ &= \text{argmin}_{\theta} \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} q_t \bar{L}(\theta, Z_t^{(i)}) \end{aligned}$$

where $\bar{L}$ is the orbit-averaging loss because of data augmentation based on the definition in [32].

- Empirical minimizer for perfectly equivariant models:

$$\hat{\theta}_E = \text{argmin}_{\theta \in \Theta_E} \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} q_t L(\theta, Z_t^{(i)}),$$

$$\Theta_E = \{\theta : f_\theta(\rho_{\text{in}}(g)(x)) = \rho_{\text{out}}(g) f_\theta(x), \forall g \in G\}$$

- Empirical minimizer for approximately equivariant nets:

$$\hat{\theta}_{AE} = \text{argmin}_{\theta} \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} q_t L(\theta, Z_t^{(i)}),$$

$$\Theta_{AE} = \{\theta : \sup_{g \in G} \| f_\theta(\rho_{\text{in}}(g)(x)) = \rho_{\text{out}}(g) f_\theta(x) \|_2 \leq \varepsilon\}$$

where Θ is the parameter space without symmetry inductive biases imposed, Θ_E is the parameter space of all equivariant functions, and Θ_{AE} is the parameter space of all approximately equivariant functions.

5.4.4 Generalization Bound for Dynamics Forecasting

[119] presented a data-dependent learning bound for the general scenario of nonstationary non-mixing stochastic processes. Yet, our focus is forecasting deterministic dynamics. Since the dynamics are nonstationary, we define a discrepancy measure to characterize the distributional shift between the training and test sets:

Definition 5 (Discrepancy Measure)**.** We use $\text{disc}_T(\boldsymbol{q})$ to denote the discrepancy between the target distribution and the distribution of the training samples.

$$\text{disc}_T(\boldsymbol{q}) = \sup_{\theta \in \Theta} \left| \mathbb{E}\left[\sum_{t=1}^{T} q_t L(\theta, Z_t) \right] - \mathbb{E} L(\theta, Z_{T+1}) \right|$$

We prove that the upper bound of the generalization error on dynamics forecasting is controlled by the sequential Rademacher complexity and the discrepancy measure of the temporal distributional shift.

Theorem 2. *For any $\delta > 0$, with probability at least $1-\delta$, the following inequality holds for all $\hat{\theta} \in \Theta$ and all $\alpha = \|\boldsymbol{q}\|_2/2 > 0$:*

$$\mathbb{E}\mathscr{L}(\hat{\theta}, Z_{T+1}) - \mathbb{E}\mathscr{L}(\theta^*, Z_{T+1}) \leq 2disc_T(\boldsymbol{q}) + 6M\sqrt{4\pi \log T}\mathscr{R}_T^{sq}(L \circ \Theta) + \sqrt{\frac{2\log(2/\sigma)}{N}} + \|\boldsymbol{q}\|_2(M\sqrt{8log\frac{1}{\delta}} + 1)$$

Note that our result is consistent with the conclusion in [119] for stochastic dynamics. Full proof can be found in Appendix .1 Theorem 7.

5.4.5 Effect of Symmetry

We derive generalization bounds for forecasting nonstationary dynamics with data augmentation, perfectly equivariant networks, and approximately equivariant networks based on Theorem 2. Following [32], we use the Wasserstein distance to measure the closeness of the original distribution to the distribution under group transformations.

We generalize Theorem 3.4 in [32] from the i.i.d case to non-stationary dynamics forecasting:

Corollary 3 (Data Augmentation)**.** *Let $L(\theta, \cdot)$ be uniformly Lipschitz w.r.t. θ with Lipschitz constant $\|L\|_{Lip}$. For any $\delta > 0$, with probability at least $1-\delta$, the following inequality holds:*

$$\begin{aligned}&\mathbb{E}\mathscr{L}(\hat{\theta}_G, Z_{T+1}) - \mathbb{E}\mathscr{L}(\theta^*, Z_{T+1})\\ &\leq 2disc_T(\boldsymbol{q}) + 6M\sqrt{4\pi \log T}\mathscr{R}_T^{sq}(\bar{L} \circ \Theta) + \sigma\\ &+ max_{t,i}\, \|L\|_{Lip} \cdot \mathbb{E}_G[\mathscr{W}(Z_{T+1}, gZ_{T+1}) + q_t \mathscr{W}(Z_t^{(i)}, gZ_t^{(i)})]\end{aligned}$$

where $\sigma = \sqrt{\frac{2\log(2/\sigma)}{N}} + \|\boldsymbol{q}\|_2(M\sqrt{8log\frac{1}{\delta}} + 1)$.

We can see that the performance gain of data augmentation is governed by a bias term $\max_{t,i} \|L\|_{\mathrm{Lip}} \cdot \mathbb{E}_G[\mathscr{W}(Z_{T+1}, gZ_{T+1}) + q_t \mathscr{W}(Z_t^{(i)}, gZ_t^{(i)})]$, which vanishes under exact symmetry and the sequential Rademacher complexity reduction because of the group orbit averaging over the loss function.

The difference in sequential Rademacher complexity between the data augmentation estimator and regular estimator can further be bounded as

$$\mathscr{R}_T^{sq}(\bar{L} \circ \Theta) - \mathscr{R}_T^{sq}(L \circ \Theta) \leq \Delta + \max_{t,i} \|L\|_{\mathrm{Lip}} \cdot \mathbb{E}_G[q_t \mathscr{W}(Z_t^{(i)}, gZ_t^{(i)})] \tag{5.13}$$

where $\Delta = \mathbb{E}_{\boldsymbol{\sigma}}[\sup_{\theta \in \Theta} \sum_{t=1}^T \sigma_t q_t \mathbb{E}_G L(\theta, gZ_t)] - \mathbb{E}_{\boldsymbol{\sigma}} \mathbb{E}_G[\sup_{\theta \in \Theta} \sum_{t=1}^T \sigma_t q_t L(\theta, gZ_t)] \leq 0$.

Here Δ corresponds to the "variance reduction term" defined in [32]. When Δ is small, data augmentation has a strong effect on improving generalizability.

To compare the generalization bounds of data augmentation and equivariant networks, we first need to prove the following lemma.

Lemma 4. $\mathscr{R}_T^{sq}(L \circ \Theta_E) \leq \mathscr{R}_T^{sq}(\bar{L} \circ \Theta)$

Next, we derive the generalization bound for strictly equivariant networks.

Corollary 5 (Equivariant Networks)**.** *Let $L(\theta, \cdot)$ be uniformly Lipschitz w.r.t. θ with Lipschitz constant $\|L\|_{Lip}$. For any $\delta > 0$, with probability at least $1 - \delta$, the following inequality holds:*

$$\begin{aligned}
&\mathbb{E}\mathscr{L}(\hat{\theta}_E, Z_{T+1}) - \mathbb{E}\mathscr{L}(\theta^*, Z_{T+1}) \\
&\leq 2disc_T(\boldsymbol{q}) + 6M\sqrt{4\pi \log T}\mathscr{R}_T^{sq}(L \circ \Theta_E) + \sigma \\
&+ \|L\|_{Lip} \cdot \mathbb{E}_G \mathscr{W}(Z_{T+1}, gZ_{T+1})
\end{aligned}$$

where $\sigma = \sqrt{\frac{2\log(2/\sigma)}{N}} + \|\boldsymbol{q}\|_2(M\sqrt{8log\frac{1}{\delta}} + 1)$.

From Lemma 4, we have $\mathscr{R}_T^{sq}(L \circ \Theta_E) \leq \mathscr{R}_T^{sq}(\bar{L} \circ \Theta)$. Hence, Corollary 5 indicates that equivariant networks have a tighter generalization bound than data augmentation. In particular, the generalization bound of data augmentation in Corollary 3 has an additional bias term $\max_{t,i} \|L\|_{\text{Lip}} \cdot \mathbb{E}_G[q_t \mathscr{W}(Z_t^{(i)}, gZ_t^{(i)})]$. This term vanishes when the data are perfectly symmetric.

However, in real-world scenarios, the data are very rarely perfect symmetric. We further analyze the generalization behavior of a class of approximate equivariant models:

Corollary 6 (Approximate Equivariance)**.** *Let $L(\theta, \cdot)$ be uniformly Lipschitz w.r.t. θ with a Lipschitz constant $\|L\|_{Lip}$. We assume $\|\hat{\theta}_{AE}\|_{EE} \leq \|\theta^*\|_{EE}$ and $\frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T} q_t L(\theta^*, Z_t^{(i)}) \leq \xi$. For any $\delta > 0$, with probability at least $1-\delta$, the following inequality holds:*

$$\begin{aligned}
&\mathbb{E}\mathscr{L}(\hat{\theta}_{AE}, Z_{T+1}) - \mathbb{E}\mathscr{L}(\theta^*, Z_{T+1}) \\
&\leq 2disc_T(\boldsymbol{q}) + 6M\sqrt{4\pi logT}\mathscr{R}_T^{sq}(L \circ \Theta_{AE}) + \sigma \\
&+ \|L\|_{Lip} \cdot \mathbb{E}_G \mathscr{W}(Z_{T+1}, gZ_{T+1}) - \|\boldsymbol{q}\|_1 \|\hat{\theta}_{AE}\|_{EE} + 2\xi
\end{aligned}$$

where $\sigma = \sqrt{\frac{2\log(2/\sigma)}{N}} + \|\boldsymbol{q}\|_2(M\sqrt{8log\frac{1}{\delta}} + 1)$.

To put it simply, when the data do not have perfect symmetries, approximately equivariant models may have better prediction performance than data augmentations and perfectly equivariant models because of the term $-\|\boldsymbol{q}\|_1\|\hat{\theta}_{AE}\|_{EE}$ in the bound. The empirical error of the population minimizer ξ can be small enough to be ignored.

If approximately equivariant estimators can learn the correct amount of symmetry in the data, which means that $\|\hat{\theta}_{AE}\|_{EE}$ is big and close to the true equivariance error in the data $\|\theta^*\|_{EE}$, then they tend to have better generalizability. On the contrary, the estimators trained on a uniformly augmented training set and perfectly equivariant estimators maintain zero equivariance error even when data are not perfectly symmetric, which is overly restricted.

Full proofs for corollary 3, 5, 6 and lemma 4 are in Appendix .1.

5.5 Conclusion

We develop novel deep learning methods to improve the generalization of deep sequence models for learning physical dynamics. We incorporate various symmetries by designing equivariant neural networks and demonstrate their superior performance on 2D time series prediction both theoretically and experimentally. Our designs obtain improved physical consistency for predictions. In the case of transformed test data, our models generalize significantly better than their non-equivariant counterparts. Importantly, all of our equivariant models can be combined and can be extended to 3D cases.

In addition, we propose a new class of approximately equivariant networks that avoid stringent symmetry constraints to better fit real-world scenarios. Our methods balance inductive biases and model flexibility by relaxing the weight-sharing and weight-tying schemes in group convolution and steerable convolution. Based on the experiments on smoke plume simulations and real-world jet flow data, we observe that our proposed approximate equivariant networks can outperform many state-of-the-art baselines with no symmetry bias or with overly strict symmetry constraints. Future work includes applying our relaxed weight-sharing design to graph neural networks and theoretical analysis for approximately equivariant networks, including universal approximation and generalization.

Theoretically, we also take the first steps in the theoretical understanding of data augmentation and equivariant networks on the task of non-stationary dynamics forecasting. We derive the generalization bounds and show that strictly equivariant networks have a tighter upper bound than data augmentation and that approximately equivariant estimators can outperform both data augmentation and perfectly equivariant networks on modeling imperfectly symmetric dynamics. A limitation of this work is that our theoretical comparison is only for upper bounds, which can be arbitrarily loose in practice. Future work includes improving the generalizing bounds with Pac-Bayesian analysis and deriving lower bounds for these approaches characterizing the hardness of learning for different model classes.

Chapter 6

Discussion and Future Directions

In this thesis, we described a physics-guide DL framework for dynamics forecasting and presented several approaches to improving the physical consistency, accuracy, and generalization of DL models for dynamics forecasting. The approaches include incorporating prior physical knowledge into the design of model architecture and loss functions for improved physical consistency and accuracy, leveraging model-based meta-learning for improved generalization across heterogeneous domains, simplifying nonlinear dynamics with Koopman theory for improved generalization over temporal distributional shifts, and incorporating symmetries into deep dynamics models for improved generalization across relevant symmetry groups and consistency with conservation laws. Nonetheless, there are still many challenges in this field and emerging opportunities for future research.

Improving Generalization. Generalization is a central problem in machine learning. One current limitation of deep learning models for learning complex dynamics is their inability to understand the system solely from data and handle distributional shifts that naturally occur. Most deep learning models for dynamics modeling are trained to model a specific system and still struggle with generalization. For example, in turbulence modeling, deep learning models trained with fixed boundaries and initial conditions often fail to generalize to fluid flows with different characteristics. To overcome this limitation, one approach is to build physics-guided deep learning models, where the physics part plays a dominant role while the neural networks focus on learning the unknown process [169]. Another promising direction is meta-learning. For instance, we proposed a model-based meta-learning method called *DyAd* in chapter 3 that can generalize across heterogeneous domains of fluid dynamics. However, this model can only generalize well on the dynamics with interpolated physical parameters and cannot extrapolate beyond the range of the physical parameters in the training set. Another idea is to transform data into a canonical distribution that neural networks can learn from and then restore the original data after predictions are made [108]. Since neural networks struggle with multiple distributions, this approach aims to find a single distribution that can represent the dynamics effectively. A trustworthy and reliable model for learning physical dynamics should be able to extrapolate to systems with various parameters, external forces, or boundary conditions while maintaining high accuracy. Therefore, further research into generalizable physics-guided deep learning is crucial.

Improving Robustness of Long-term Forecasting. Long-term forecasting of physical dynamics is a challenging task as it is prone to error accumulation and instability to perturbations in the input, which significantly affect the accuracy of neural networks over a long forecasting horizon. To address these issues, several training techniques have been proposed in recent years. One such technique involves adding noise to the input, which makes the models less sensitive to perturbations [17]. It also suggests when making predictions in an autoregressive manner, the neural nets should be trained to make multiple steps of predictions in each autoregressive call instead of just one step. Additionally, [254] proposed a time-based Lyapunov regularizer to the loss function of deep forecasters to avoid training error propagation and improve the trained long-term prediction. Moreover, [166, 10] utilized online normalization that is normalizing the current training sample using a running mean and standard deviation, which also increases the time horizon that the model can predict. These models are trained on a large amount of simulation data. However, for real-world problems, obtaining real-world data such as experimental data of jet flow can be expensive, which presents a significant challenge for improving the robustness of predictions on limited training data. In such cases, developing robust prediction models that can generalize well on limited training data is of great importance.

Learning Dynamics in Non-Euclidean Spaces. Spatiotemporal phenomena, from global ocean currents to the spread of infectious diseases, are examples of dynamics in non-Euclidean spaces, which means they cannot be easily represented using traditional Euclidean geometry. To address this issue, the field of geometric deep learning [20] has emerged. Geometric deep learning aims to generalize neural network models to non-Euclidean domains such as graphs and manifolds. However, most of the existing work in this field has been limited to static graph data. Thus, learning dynamics in non-Euclidean Ssaces is a promising direction, and geometric concepts, such as rent notions of distance, curvature, and parallel transport, must be taken into account when designing models. For example, when modeling the ocean dynamics on the earth, which is a sphere, we need to encode the gauge equivariance [37] in the design of neural nets since there is no canonical coordinate system on a sphere.

Theoretical Analysis. The majority of literature on learning dynamics with DL focuses on the methodological and practical aspects. Research into the theoretical analysis of generalization is lacking. Current statistical learning theory is based on the typical assumption that training and test data are identically and independently distributed (i.i.d.) samples from some unknown distribution [251, 157]. However, this assumption does not hold for most dynamical systems, where observations at different times and locations may be highly correlated. [119] provided the discrepancy-based generalization guarantees for time series forecasting. On the basis of this, [229] took the first step to derive generalization bounds for equivariant models and data augmentation in the dynamics forecasting setting. The derived upper bounds are expressed in terms of measures of distributional shifts and group transformations, as well as the Rademacher complexity. But these bounds are sometimes not very informative since many of the inequalities used can be loose. However, to better understand the performance of DL on learning dynamics, we need to derive generalization bounds expressed in terms of the characteristics of the dynamics, such as the order and Lyapunov exponents. Deriving lower generalization bounds are also necessary since they reveal the best performance scenarios. Theoretical studies can also inspire research into model design and algorithm development for learning dynamics.

Causal Inference in Dynamical Systems. A fundamental pursuit in science is to identify causal relationships. In the context of dynamical systems, we may ask which variables directly or indirectly influence other variables through intermediates. While traditional approaches to the discovery of causation involve conducting controlled real experiments [165, 14], data-driven approaches have been proposed to identify causal relations from observational data in the past few decades [77, 73]. However, most data-driven approaches do not directly address the challenge of learning causality with big data. Many questions remain open, such as using causality to improve deep learning models, disentangling complex and multiple treatments, and designing the environment to control the given dynamics. Additionally, we are also interested in understanding the system's response under interventions. For example, when using deep learning to model climate dynamics, we need to make accurate predictions under different climate policies, such as carbon pricing policies and the development of clean energy, to enable better decisions by governments for controlling climate change.

Search for Physical Laws. Another promising direction is to seek physics laws with the help of DL. The search for fundamental laws of practical problems is the main theme of science. Once the governing equations of dynamical systems are found, they allow for accurate mathematical modeling, increased interpretability, and robust forecasting. However, current methods are limited to selecting from a large dictionary of possible mathematical terms [175, 191, 25, 122, 178]. The extremely large search space, limited high-quality experimental data, and overfitting issues have been critical concerns. Another line of work is to discover symmetry from the observed data instead of the entire dynamics with the help of DL [46, 59]. But these works can only work well on synthetic data and discover known symmetries. Still, research on data-driven methods based on DL for discovering physics laws is quite preliminary.

Efficient Computation. Given the rapid growth in high-performance computation, we need to improve automation and accelerate streamlining of highly compute-intensive workflows for science. We should focus on how to efficiently train, test, and deploy complex physics-guided DL models on large datasets and high-performance computing systems, such that these models can be quickly utilized to solve real-world scientific problems. To really revolutionize the field, these DL tools need to become more scalable and transferable and converge into a complete pipeline for the simulation and analysis of dynamical systems. A simple example is that we can integrate machine learning tools into the existing numerical simulation platforms so that we do not need to move data between systems every time and we can easily use either or both types of methods for analyzing data.

In conclusion, given the availability of abundant data and rapid growth in computation, we envision that the integration of physics and DL will play an increasingly essential role in advancing scientific discovery and addressing important dynamics modeling problems.

Appendix

.1 Full proof for theorems in Section 5.4

Theorem 7. *For any $\delta > 0$, with probability at least $1 - \delta$, the following inequality holds for all $\hat{\theta} \in \Theta$ and all $\alpha = \|\boldsymbol{q}\|_2/2 > 0$:*

$$\mathbb{E}\mathscr{L}(\hat{\theta}, Z_{T+1}) - \mathbb{E}\mathscr{L}(\theta^*, Z_{T+1}) \leq 2disc_T(\boldsymbol{q}) + 6M\sqrt{4\pi \log T}\mathscr{R}_T^{sq}(L \circ \Theta) + \sqrt{\frac{2\log(2/\sigma)}{N}} + \|\boldsymbol{q}\|_2(M\sqrt{8log\frac{1}{\delta}} + 1)$$

Proof.

$$\mathbb{E}L(\hat{\theta}, Z_{T+1}) - \mathbb{E}L(\theta^*, Z_{T+1}) = I + II + III + IV$$

$$I = \mathbb{E}L(\hat{\theta}, Z_{T+1}) - \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T} q_t L(\hat{\theta}, Z_t^{(i)})$$

$$II = \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T} q_t L(\hat{\theta}, Z_t^{(i)}) - \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T} q_t L(\theta^*, Z_t^{(i)}) \leq 0 \text{(the model does not underfit the data)}$$

$$III = \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T} q_t L(\theta^*, Z_t^{(i)}) - \mathbb{E}\sum_{t=1}^{T} q_t L(\theta^*, Z_t) \leq \sqrt{\frac{2\log(2/\sigma)}{N}} \text{(time series are i.i.d sampled).}$$

$$IV = \mathbb{E}\sum_{t=1}^{T} q_t L(\theta^*, Z_t) - \mathbb{E}L(\theta^*, Z_{T+1}) \leq \sup_{\theta \in \Theta} |\mathbb{E}\sum_{t=1}^{T} q_t L(\theta, Z_t) - \mathbb{E}L(\theta, Z_{T+1})| = \text{disc}_T(\boldsymbol{q})$$

Now we only need to bound the first term $I = \mathbb{E}L(\hat{\theta}, Z_{T+1}) - \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T} q_t L(\hat{\theta}, Z_t^{(i)})$

$$
\begin{aligned}
&\mathbb{P}(I - \mathrm{disc}_T(\boldsymbol{q}) > \varepsilon) \\
&= \mathbb{P}(\mathbb{E}L(\hat{\theta}, Z_{T+1}) - \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T} q_t L(\hat{\theta}, Z_t^{(i)}) - \sup_{\theta\in\Theta}|\mathbb{E}L(\theta, Z_{T+1}) - \mathbb{E}\sum_{t=1}^{T} q_t L(\theta, Z_t)| > \varepsilon) \\
&\leq \mathbb{P}(\sup_{\theta\in\Theta}|\mathbb{E}L(\theta, Z_{T+1}) - \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T} q_t L(\theta, Z_t^{(i)})| - \sup_{\theta\in\Theta}|\mathbb{E}L(\theta, Z_{T+1}) - \mathbb{E}\sum_{t=1}^{T} q_t L(\theta, Z_t)| > \varepsilon) \\
&\leq \mathbb{P}(\sup_{\theta\in\Theta}|\mathbb{E}L(\theta, Z_{T+1}) - \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T} q_t L(\theta, Z_t^{(i)}) + \mathbb{E}\sum_{t=1}^{T} q_t L(\theta, Z_t) - \mathbb{E}L(\theta, Z_{T+1})| > \varepsilon) \\
&\leq \mathbb{P}(\sup_{\theta\in\Theta}|\mathbb{E}L(\theta, Z_{T+1}) - \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T} q_t L(\theta, Z_t^{(i)}) + \mathbb{E}\sum_{t=1}^{T} q_t L(\theta, Z_t) - \mathbb{E}L(\theta, Z_{T+1})| > \varepsilon) \\
&= \mathbb{P}(\sup_{\theta\in\Theta}|\mathbb{E}\sum_{t=1}^{T} q_t L(\theta, Z_t) - \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T} q_t L(\theta, Z_t^{(i)})| > \varepsilon) \\
&= \mathbb{P}(\exp(\lambda \sup_{\theta\in\Theta}|\mathbb{E}\sum_{t=1}^{T} q_t L(\theta, Z_t) - \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T} q_t L(\theta, Z_t^{(i)}))|)) > \exp(\lambda\varepsilon)) \\
&\leq \exp(-\lambda\varepsilon)\mathbb{E}[\exp(\lambda \sup_{\theta\in\Theta}(\mathbb{E}\sum_{t=1}^{T} q_t L(\theta, Z_t) - \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T} q_t L(\theta, Z_t^{(i)})))](\text{Markov's inequality}) \quad \blacksquare
\end{aligned}
$$

$$
\begin{aligned}
&\mathbb{E}[\exp(\lambda\ \sup_{\theta\in\Theta}(\mathbb{E}\sum_{t=1}^{T} q_t L(\theta,Z_t) - \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T} q_t L(\theta,Z_t^{(i)})))] \\
&= \mathbb{E}[\exp(\lambda\ \sup_{\theta\in\Theta}(\frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T} q_t(\mathbb{E}L(\theta,Z_t) - L(\theta,Z_t^{(i)}))))] \\
&= \mathbb{E}[\exp(\lambda\ \sup_{\theta\in\Theta}(\frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T} q_t(\mathbb{E}[L(\theta,Z_t)|Z_0] - L(\theta,Z_t^{(i)}))))] \\
&\leq \mathbb{E}[\mathbb{E}_{Z_0\sim\mathscr{X}^k\times\mathscr{X}}\exp(\lambda\ \sup_{\theta\in\Theta}[(\frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T} q_t(L(\theta,Z_t) - L(\theta,Z_t^{(i)}))|Z_0])] \\
&= \mathbb{E}[\exp(\lambda\ \sup_{\theta\in\Theta}[(\frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T} q_t(L(\theta,Z'_t) - L(\theta,Z_t^{(i)}))))] \\
&= \mathbb{E}[\exp(\lambda\ \sup_{\theta\in\Theta}[(\sum_{t=1}^{T} q_t(L(\theta,Z'_t) - L(\theta,Z_t^{*}))))] \\
&= \mathbb{E}\mathbb{E}_{\boldsymbol{\sigma}}[\exp(\lambda\ \sup_{\theta\in\Theta}[(\sum_{t=1}^{T} \sigma_t q_t(L(\theta,Z'_t) - L(\theta,Z_t^{*}))))] \\
&= \mathbb{E}_{\boldsymbol{z}^*,\boldsymbol{z}'}\mathbb{E}_{\boldsymbol{\sigma}}[\exp(\lambda\ \sup_{\theta\in\Theta}[(\sum_{t=1}^{T} \sigma_t q_t(f(\boldsymbol{z}_t^*(\boldsymbol{\sigma})) - f(\boldsymbol{z}'_t(\boldsymbol{\sigma})))))] \text{ replace } L(\theta,\cdot) \text{ with } f \text{ for simplicity.} \\
&\leq \mathbb{E}_{\boldsymbol{z}^*}\mathbb{E}_{\boldsymbol{\sigma}}[\exp(2\lambda\ \sup_{f\in\mathscr{F}}\sum_{t=1}^{T} \sigma_t q_t f(\boldsymbol{z}_t^*(\boldsymbol{\sigma})))]
\end{aligned}
$$

$\blacksquare$

Given $\boldsymbol{z}^*$, let C be the minimal α-cover of $\mathscr{F}$ on $\boldsymbol{z}^*$,

$$\sup_{f\in\mathscr{F}}\sum_{t=1}^{T} \sigma_t q_t f(\boldsymbol{z}_t^*(\boldsymbol{\sigma}))) \leq \max_{\boldsymbol{c}\in C}\sum_{t=1}^{T} \sigma_t q_t \boldsymbol{c}_t(\boldsymbol{\sigma}) + \alpha$$

Thus,

$$\begin{aligned}
&\mathbb{E}_{\boldsymbol{\sigma}}[\exp(2\lambda\ \sup_{f\in\mathscr{F}}\sum_{t=1}^{T}\sigma_t q_t f(\boldsymbol{z}_t^*(\boldsymbol{\sigma})))] \\
&\le \exp(2\lambda\alpha)\mathbb{E}_{\boldsymbol{\sigma}}[\exp(2\lambda\ \max_{\boldsymbol{c}\in C}\sum_{t=1}^{T}\sigma_t q_t \boldsymbol{c}_t(\boldsymbol{\sigma}))] \\
&\le \exp(2\lambda\alpha)\max_{\boldsymbol{c}\in C}\mathbb{E}_{\boldsymbol{\sigma}}[\exp(2\lambda\ \sum_{t=1}^{T}\sigma_t q_t \boldsymbol{c}_t(\boldsymbol{\sigma}))] \\
&= \exp(2\lambda\alpha)\max_{\boldsymbol{c}\in C}\mathbb{E}_{\boldsymbol{\sigma}}[\exp(2\lambda\ \sum_{t=1}^{T-1}\sigma_t q_t \boldsymbol{c}_t(\boldsymbol{\sigma}))\ \mathbb{E}_{\boldsymbol{\sigma}^T}[\exp(2\lambda\ \sigma_T q_T \boldsymbol{c}_T(\boldsymbol{\sigma}))|\boldsymbol{\sigma}_{1:T-1}]] \\
&\le \exp(2\lambda\alpha)\max_{\boldsymbol{c}\in C}\mathbb{E}_{\boldsymbol{\sigma}}[\exp(2\lambda\ \sum_{t=1}^{T-1}\sigma_t q_t \boldsymbol{c}_t(\boldsymbol{\sigma}))\ \exp(2\lambda^2 q_T^2 M^2)] \\
&\le \exp(2\lambda\alpha)\exp(2\lambda^2\|\boldsymbol{q}\|_2^2 M^2)\ \text{(Iterate the last inequality over } t\text{)}
\end{aligned}$$

Then we have

$$\mathbb{P}(I-\mathrm{disc}_T(\boldsymbol{q})>\varepsilon)\le \mathbb{E}_{\boldsymbol{z}}[\mathscr{N}_1(\alpha,\Theta,\boldsymbol{z})]\exp(2\lambda\alpha-\lambda\varepsilon+2\lambda^2\|\boldsymbol{q}\|_2^2M^2)$$

Optimize λ

$$\mathbb{P}(I-\mathrm{disc}_T(\boldsymbol{q})>\varepsilon)\le \mathbb{E}_{\boldsymbol{z}}[\mathscr{N}_1(\alpha,\Theta,\boldsymbol{z})]\exp(\frac{(\varepsilon-2\alpha)^2}{8\|\boldsymbol{q}\|_2^2M^2})$$

Finally, $\mathbb{E}_{\boldsymbol{z}}[\mathscr{N}_1(\alpha,\Theta,\boldsymbol{z})]$ can be further bounded by the sequential Rademacher complexity based on the Theorem 2 in [119]. □

Here are the full proofs for corollary 3, 5, 6 and lemma 4.

Corollary 8. *Let $L(\theta,\cdot)$ be Lipschitz uniformly over θ, with a Lipschitz constant $\|L\|_{Lip}$. Assume $\frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}q_tL(\theta^*,Z_t^{(i)})\le\xi$. For any $\delta>0$, with probability at least $1-\delta$, the following inequality holds for all $\alpha=\|\boldsymbol{q}\|_2/2>0$:*

$$\mathbb{E}L(\hat{\theta}_G, Z_{T+1}) - \mathbb{E}L(\theta^*, Z_{T+1}) \leq 2disc_T(\boldsymbol{q}) + 6M\sqrt{4\pi logT}\mathscr{R}_T^{sq}(L \circ \Theta_G) + \Delta$$
$$+ max_{t,i}\ \|L\|_{Lip} \cdot \mathbb{E}_G[\mathscr{W}(Z_{T+1}, gZ_{T+1}) + q_t\mathscr{W}(Z_t^{(i)}, gZ_t^{(i)})] \tag{1}$$

$$\mathbb{E}L(\hat{\theta}_E, Z_{T+1}) - \mathbb{E}L(\theta^*, Z_{T+1}) \leq 2disc_T(\boldsymbol{q}) + 6M\sqrt{4\pi logT}\mathscr{R}_T^{sq}(L \circ \Theta_E) + \Delta$$
$$+ max_{t,i}\ \|L\|_{Lip} \cdot \mathbb{E}_G\mathscr{W}(Z_t^{(i)}, gZ_t^{(i)}) \tag{2}$$

$$\mathbb{E}L(\hat{\theta}_{AE}, Z_{T+1}) - \mathbb{E}L(\theta^*, Z_{T+1}) \leq 2disc_T(\boldsymbol{q}) + 6M\sqrt{4\pi logT}\mathscr{R}_T^{sq}(L \circ \Theta_{AE}) + \Delta$$
$$+ \|L\|_{Lip} \cdot \mathbb{E}_G\mathscr{W}(Z_{T+1}, gZ_{T+1}) - \|\boldsymbol{q}\|_1\|\hat{\theta}_{AE}\|_{EE} + 2\xi \tag{3}$$

where $\Delta = \sqrt{\frac{2\log(2/\sigma)}{N}} + \|\boldsymbol{q}\|_2(M\sqrt{8log\frac{1}{\delta}} + 1)$.

Proof. When $\hat{\theta} = \hat{\theta}_G$, we only need to derive a bound for I in the previous proof.

$$\begin{aligned}
&I = \mathbb{E}L(\hat{\theta}, Z_{T+1}) - \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T} q_t L(\hat{\theta}, Z_t^{(i)}) = A + B + C \\
&A = \mathbb{E}L(\hat{\theta}, Z_{T+1}) - \mathbb{E}\mathbb{E}_G L(\hat{\theta}, gZ_{T+1}) \leq \mathbb{E}_G|\mathbb{E}L(\hat{\theta}_G, Z_{T+1}) - \mathbb{E}L(\hat{\theta}_G, gZ_{T+1})| \leq \|L\|_{\text{Lip}} \cdot \mathbb{E}_G\mathscr{W}(Z_{T+1}, gZ_{T+1}) \\
&B = \mathbb{E}\mathbb{E}_G L(\hat{\theta}, gZ_{T+1}) - \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T} q_t \mathbb{E}_G L(\hat{\theta}, gZ_t^{(i)}) \leq \text{disc}_T(\boldsymbol{q}) + 6M\sqrt{4\pi logT}\mathscr{R}_T^{sq}(L \circ \Theta_G) + \Delta \\
&C = \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T} q_t \mathbb{E}_G L(\hat{\theta}, gZ_t^{(i)}) - \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T} q_t L(\hat{\theta}, Z_t^{(i)}) \leq \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T} q_t[\mathbb{E}_G L(\hat{\theta}_G, gZ_t^{(i)}) - L(\hat{\theta}_G, Z_t^{(i)})] \\
&\quad \leq \max_{t,i}\ \|L\|_{\text{Lip}} \cdot \mathbb{E}_G\mathscr{W}(Z_t^{(i)}, gZ_t^{(i)})
\end{aligned} \tag{4}$$

When $\hat{\theta} = \hat{\theta}_{AE}$:

$$\mathbb{E}L(\hat{\theta}_{AE}, Z_{T+1}) - \mathbb{E}L(\theta^*, Z_{T+1}) = I + II + III + IV + V + VI$$

$$
\begin{aligned}
I &= \mathbb{E}L(\hat{\theta}_{AE}, Z_{T+1}) - \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T} q_t L(\hat{\theta}_{AE}, Z_t^{(i)}) \leq \mathrm{disc}_T(\boldsymbol{q}) + 6M\sqrt{4\pi logT}\mathscr{R}_T^{sq}(L \circ \Theta_{AE}) + \Delta \\
II &= \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T} q_t L(\hat{\theta}_{AE}, Z_t^{(i)}) - \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T} q_t L(\theta^*, Z_t^{(i)}) \leq 0 \text{ (The model does not underfit the data)} \\
IV &= \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T} q_t \mathbb{E}_G L(\theta^*, gZ_t^{(i)}) - \sum_{t=1}^{T} q_t \mathbb{E}\mathbb{E}_G L(\theta^*, gZ_t) \leq \sqrt{\frac{2\log(2/\sigma)}{N}} \\
V &= \sum_{t=1}^{T} q_t \mathbb{E}\mathbb{E}_G L(\theta^*, gZ_t^{(i)}) - \mathbb{E}\mathbb{E}_G L(\theta^*, gZ_{T+1}) = \sum_{t=1}^{T} q_t \mathbb{E}\bar{L}(\theta^*, gZ_t^{(i)}) - \mathbb{E}\bar{L}(\theta^*, gZ_{T+1}) \\
&\leq \sup_{\theta\in\Theta}\left|\mathbb{E}\left[\sum_{t=1}^{T} q_t \bar{L}(\theta, Z_t)\right] - \mathbb{E}\bar{L}(\theta, Z_{T+1})\right| \leq \mathrm{disc}_T(\boldsymbol{q}) \\
VI &= \mathbb{E}\mathbb{E}_G L(\theta^*, gZ_{T+1}) - \mathbb{E}L(\theta^*, Z_{T+1}) \leq \|L\|_{\mathrm{Lip}} \cdot \mathbb{E}_G \mathscr{W}(Z_{T+1}, gZ_{T+1}) \\
III &= \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T} q_t L(\theta^*, Z_t^{(i)}) - \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T} q_t \mathbb{E}_G L(\theta^*, gZ_t^{(i)}) \;\; (\text{let } x_t^{(i)} = X_{t-k-1:t-1}^{(i)} \text{ and } y_t^{(i)} = X_t^{(i)}) \\
&= \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T} q_t \mathbb{E}_G[\|y_t^{(i)} - f_{\theta^*}(x_t^{(i)})\| - \|gy_t^{(i)} - f_{\theta^*}(gx_t^{(i)})\|] \\
&= \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T} q_t \mathbb{E}_G[\|y_t^{(i)} - f_{\theta^*}(x_t^{(i)})\| - \|gy_t^{(i)} - gf_{\theta^*}(x_t^{(i)}) + gf_{\theta^*}(x_t^{(i)}) - f_{\theta^*}(gx_t^{(i)})\|] \\
&\leq \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T} q_t \mathbb{E}_G[-\|gf_{\theta^*}(x_t^{(i)}) - f_{\theta^*}(gx_t^{(i)})\| + \|y_t^{(i)} - f_{\theta^*}(x_t^{(i)})\| + \|gy_t^{(i)} - gf_{\theta^*}(x_t^{(i)})\|] \\
&\leq -\|\boldsymbol{q}\|_1 \|\hat{\theta}_{AE}\|_{EE} + 2\xi
\end{aligned} \tag{5}
$$

When $\hat{\theta} = \hat{\theta}_E$:

$$
\mathbb{E}L(\hat{\theta}_E, Z_{T+1}) - \mathbb{E}L(\theta^*, Z_{T+1}) = I + II + III + IV + V + VI
$$

$$
\begin{aligned}
&I = \mathbb{E}L(\hat{\theta}_E, Z_{T+1}) - \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T} q_t L(\hat{\theta}_E, Z_t^{(i)}) \leq \mathrm{disc}_T(\boldsymbol{q}) + 6M\sqrt{4\pi logT}\mathscr{R}_T^{sq}(L\circ\Theta_E) + \Delta \\
&II = \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T} q_t L(\hat{\theta}_E, Z_t^{(i)}) - \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T} q_t L(\theta^*, Z_t^{(i)}) \leq 0 \text{ (The model does not underfit the data)} \\
&III = \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T} q_t L(\theta^*, Z_t^{(i)}) - \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T} q_t \mathbb{E}_G L(\theta^*, gZ_t^{(i)}) = 0 \\
&IV = \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T} q_t \mathbb{E}_G L(\theta^*, gZ_t^{(i)}) - \sum_{t=1}^{T} q_t \mathbb{E}\mathbb{E}_G L(\theta^*, gZ_t) \leq \sqrt{\frac{2\log(2/\sigma)}{N}} \\
&V = \sum_{t=1}^{T} q_t \mathbb{E}\mathbb{E}_G L(\theta^*, gZ_t^{(i)}) - \mathbb{E}\mathbb{E}_G L(\theta^*, gZ_{T+1}) = \sum_{t=1}^{T} q_t \mathbb{E}\bar{L}(\theta^*, gZ_t^{(i)}) - \mathbb{E}\bar{L}(\theta^*, gZ_{T+1}) \\
&\quad \leq \sup_{\theta\in\Theta}\left|\mathbb{E}\left[\sum_{t=1}^{T} q_t \bar{L}(\theta, Z_t)\right] - \mathbb{E}\bar{L}(\theta, Z_{T+1})\right| \leq \mathrm{disc}_T(\boldsymbol{q}) \\
&VI = \mathbb{E}\mathbb{E}_G L(\theta^*, gZ_{T+1}) - \mathbb{E}L(\theta^*, Z_{T+1}) \leq \|L\|_{\mathrm{Lip}} \cdot \mathbb{E}_G \mathscr{W}(Z_{T+1}, gZ_{T+1})
\end{aligned} \tag{6}
$$

Combining the bounds for the six terms gives the desired result.

Moreover,

$$
\begin{aligned}
&\mathscr{R}_T^{sq}(L\circ\Theta_E) - \mathscr{R}_T^{sq}(\bar{L}\circ\Theta) \\
&\leq \mathbb{E}_{\boldsymbol{\sigma}}[\sup_{\theta_E\in\Theta_E}\frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\sigma_t q_t L(\theta_E, Z_t)] - \mathbb{E}_{\boldsymbol{\sigma}}[\sup_{\theta\in\Theta}\frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\sigma_t q_t \mathbb{E}_G L(\theta, gZ_t)] \\
&\leq \mathbb{E}_{\boldsymbol{\sigma}}[\sup_{\theta_E\in\Theta_E}\frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\sigma_t q_t \mathbb{E}_G L(\theta_E, gZ_t)] - \mathbb{E}_{\boldsymbol{\sigma}}[\sup_{\theta\in\Theta}\frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\sigma_t q_t \mathbb{E}_G L(\theta, gZ_t)] \\
&= \mathbb{E}_{\boldsymbol{\sigma}}\mathbb{E}_G[\sup_{\theta_E\in\Theta_E}\frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\sigma_t q_t L(\theta_E, gZ_t) - \sup_{\theta\in\Theta}\frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\sigma_t q_t L(\theta, gZ_t)] \\
&\leq \mathbb{E}_{\boldsymbol{\sigma}}\mathbb{E}_G[\sup_{\theta_E\in\Theta_E}\frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\sigma_t q_t L(\theta_E, gZ_t) - \sup_{\theta\in\Theta_E}\frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T}\sigma_t q_t L(\theta, gZ_t)] \\
&= 0
\end{aligned} \tag{7}
$$

□

Bibliography

[1] Ferran Alet, Tomas Lozano-Perez, and L. Kaelbling. Modular meta-learning. *arXiv preprint arXiv: 1806.10166*, 2018.

[2] B. Alipanahi, Andrew Delong, Matthew T. Weirauch, and B. Frey. Predicting the sequence specificities of dna- and rna-binding proteins by deep learning. *Nature Biotechnology*, 33:831–838, 2015.

[3] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2019.

[4] Brandon Anderson, Truong-Son Hy, and Risi Kondor. Cormorant: Covariant molecular neural networks. In *Advances in neural information processing systems (NeurIPS)*, 2019.

[5] Sercan O Arik, Nathanael C Yoder, and Tomas Pfister. Self-adaptive forecasting for improved deep learning on non-stationary time-series. *arXiv preprint arXiv:2202.02403*, 2022.

[6] Adebiyi A Ariyo, Adewumi O Adewumi, and Charles K Ayo. Stock price prediction using the arima model. In *2014 UKSim-AMSS 16th international conference on computer modelling and simulation*, pages 106–112. IEEE, 2014.

[7] Ibrahim Ayed, Emmanuel De Bézenac, Arthur Pajot, and Patrick Gallinari. Learning partially observed PDE dynamics with neural networks, 2019.

[8] Omri Azencot, N Benjamin Erichson, Vanessa Lin, and Michael Mahoney. Forecasting sequential data using consistent koopman autoencoders. In *International Conference on Machine Learning*, pages 475–485. PMLR, 2020.

[9] Erkao Bao and Linqi Song. Equivariant neural networks and equivarification. *arXiv preprint arXiv:1906.07172*, 2019.

[10] Brian R. Bartoldson, Rui Wang, Brenda M. Ng, Phuoc Chanh N Nguyen, Jose E. Cadena Pico, Phan Nguyen, David P. Widemann, and USDOE National Nuclear Security Administration. Meshgraphnets, 2021.

[11] Jonathan Baxter. Theoretical models of learning to learn. In *Learning to learn*, pages 71–94. Springer, 1998.

[12] Stefanos Bennett and Jase Clarkson. Time series prediction under distribution shift using differentiable forgetting. *arXiv preprint arXiv:2207.11486*, 2022.

[13] Tom Beucler, Michael Pritchard, Stephan Rasp, Pierre Gentine, Jordan Ott, and Pierre Baldi. Enforcing analytic constraints in neural-networks emulating physical systems. *arXiv:1909.00912v2*, 2019.

[14] E. Bollt, J. Sun, and J. Runge. Introduction to focus issue: Causation inference and information flow in dynamical systems: Theory and applications. *Chaos*, 2018.

[15] Olivier Bousquet, Stéphane Boucheron, and Gábor Lugosi. Introduction to statistical learning theory. In *Summer school on machine learning*, pages 169–207. Springer, 2003.

[16] Johannes Brandstetter, Max Welling, and Daniel E Worrall. Lie point symmetry data augmentation for neural pde solvers. In *International Conference on Machine Learning*, pages 2241–2256. PMLR, 2022.

[17] Johannes Brandstetter, Daniel E. Worrall, and Max Welling. Message passing neural PDE solvers. In *International Conference on Learning Representations*, 2022.

[18] Jure Brence, Ljupčo Todorovski, and Sašo Džeroski. Probabilistic grammars for equation discovery. *Knowledge-Based Systems*, 224:107077, 2021.

[19] James Bridges and Mark P Wernet. Measurements of turbulent convection speeds in multistream jets using time-resolved piv. In *23rd AIAA/CEAS aeroacoustics conference*, 2017.

[20] Michael M Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021.

[21] Bingni W Brunton, Lise A Johnson, Jeffrey G Ojemann, and J Nathan Kutz. Extracting spatial–temporal coherent patterns in large-scale neural recordings using dynamic mode decomposition. *Journal of neuroscience methods*, 258:1–15, 2016.

[22] S. Brunton, B. R. Noack, and P. Koumoutsakos. Machine learning for fluid mechanics. *ArXiv*, abs/1905.11075, 2019.

[23] Steven L. Brunton. Applying machine learning to study fluid mechanics. *arXiv preprint arXiv:2110.02083*, 2021.

[24] Steven L Brunton, Marko Budišić, Eurika Kaiser, and J Nathan Kutz. Modern koopman theory for dynamical systems. *arXiv preprint arXiv:2102.12086*, 2021.

[25] Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Discovering governing equations from data: Sparse identification of nonlinear dynamical systems. *arXiv preprint arXiv:1509.03580*, 2015.

[26] Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the national academy of sciences*, 113(15):3932–3937, 2016.

[27] Ruijin Cang, Hechao Li, Hope Yao, Yang Jiao, and Yi Ren. Improving direct physical properties prediction of heterogeneous materials from imaging data via convolutional neural network and a morphology-aware generative model. *arXiv: Computational Physics*, 2017.

[28] Alejandro Carderera, Sebastian Pokutta, Christof Schütte, and Martin Weiser. Cindy: Conditional gradient-based identification of non-linear dynamics – noise-robust recovery. *arXiv preprint arXiv:2101.02630*, 2021.

[29] G. Carleo and M. Troyer. Solving the quantum many-body problem with artificial neural networks. *Science*, 355:602 – 606, 2017.

[30] Bruno Chaoua. The state of the art of hybrid rans/les modeling for the simulation of turbulent flows. *Springer Netherlands*, 99:279–327, 2017.

[31] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 6571–6583. Curran Associates, Inc., 2018.

[32] Shuxiao Chen, Edgar Dobriban, and Jane Lee. A group-theoretic framework for data augmentation. *Advances in Neural Information Processing Systems*, 33:21321–21333, 2020.

[33] Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *Advances in neural information processing systems*, pages 6571–6583, 2018.

[34] Benjamin Chidester, Minh N. Do, and Jian Ma. Rotation equivariance and invariance in convolutional neural networks. *arXiv preprint arXiv:1805.12301*, 2018.

[35] Dragos Bogdan Chirila. *Towards lattice Boltzmann models for climate sciences: The GeLB programming language with applications*. PhD thesis, University of Bremen, 2018.

[36] Mengyu Chu and Nils Thuerey. Data-driven synthesis of smoke flows with cnn-based feature descriptors. *ACM Transactions on Graphics (TOG)*, 36(4):69, 2017.

[37] Taco S. Cohen, Maurice Weiler, Berkay Kicanaoglu, and Max Welling. Gauge equivariant convolutional networks and the icosahedral CNN. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, volume 97, pages 1321–1330, 2019.

[38] Taco S. Cohen and Max Welling. Group equivariant convolutional networks. In *International conference on machine learning (ICML)*, pages 2990–2999, 2016.

[39] Taco S. Cohen and Max Welling. Steerable cnns. In *International Conference on Learning Representations*, 2017.

[40] M. Cranmer, S. Greydanus, S. Hoyer, P. Battaglia, D. Spergel, and S. Ho. Lagrangian neural networks. *ArXiv*, abs/2003.04630, 2020.

[41] Rainer Dahlhaus. Fitting time series models to nonstationary processes. *The annals of Statistics*, 25(1):1–37, 1997.

[42] Stéphane d'Ascoli, Hugo Touvron, Matthew Leavitt, Ari Morcos, Giulio Biroli, and Levent Sagun. Convit: Improving vision transformers with soft convolutional inductive biases. *ICML*, 2021.

[43] Richard H. Day. Complex economic dynamics-vol. 1: An introduction to dynamical systems and market mechanisms. *MIT Press Books*, 1, 1994.

[44] Filipe de Avila Belbute-Peres, Thomas D. Economon, and J. Z. Kolter. Combining differentiable pde solvers and graph neural networks for fluid flow prediction. *ArXiv*, abs/2007.04439, 2020.

[45] Emmanuel de Bezenac, Arthur Pajot, and Patrick Gallinari. Deep learning for physical processes: Incorporating prior scientific knowledge. In *International Conference on Learning Representations*, 2018.

[46] Nima Dehmamy, Robin Walters, Yanchen Liu, Dashun Wang, and Rose Yu. Automatic symmetry discovery with lie algebra convolutional network. *Advances in Neural Information Processing Systems*, 34, 2021.

[47] Sander Dieleman, Jeffrey De Fauw, and Koray Kavukcuoglu. Exploiting cyclic symmetry in convolutional neural networks. In *International Conference on Machine Learning (ICML)*, 2016.

[48] Jonathan B Dingwell. Lyapunov exponents. *Wiley encyclopedia of biomedical engineering*, 2006.

[49] Jérémie Donà, Jean-Yves Franceschi, sylvain lamprier, and patrick gallinari. {PDE}-driven spatiotemporal disentanglement. In *International Conference on Learning Representations*, 2021.

[50] Sašo Džeroski, Pat Langley, and Ljupčo Todorovski. *Computational discovery of scientific knowledge*. Springer, 2007.

[51] Saso Dzeroski and Ljupco Todorovski. *Computational discovery of scientific knowledge: introduction, techniques, and applications in environmental and life sciences*, volume 4660. Springer, 2007.

[52] Weinan E, Jiequn Han, and Linfeng Zhang. Integrating machine learning with physics-based modeling. *arXiv:2006.02619*, 2019.

[53] P. Sagaut E. Labourasse. Advance in rans-les coupling, a review and an insight on the nlde approach. *Archives of Computational Methods in Engineering*, 11:199–256, 2004.

[54] Gamaleldin F. Elsayed, Prajit Ramachandran, Jonathon Shlens, and Simon Kornblith. Revisiting spatial invariance with low-rank local connectivity. In *Proceedings of the 37th International Conference of Machine Learning (ICML)*, 2020.

[55] Christos Faloutsos, Jan Gasthaus, Tim Januschowski, and Yuyang Wang. Forecasting big time series: old and new. *Proceedings of the VLDB Endowment*, 11(12):2102–2105, 2018.

[56] Fletcher Fan, Bowen Yi, David Rye, Guodong Shi, and Ian R Manchester. Learning stable koopman embeddings. In *2022 American Control Conference (ACC)*, pages 2742–2747. IEEE, 2022.

[57] Rui Fang, David Sondak, Pavlos Protopapas, and Sauro Succi. Deep learning for turbulent channel flow. *arXiv preprint arXiv:1812.02241*, 2018.

[58] Chelsea Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference of Machine Learning*, 2017.

[59] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017.

[60] Marc Finzi, Gregory Benton, and Andrew G Wilson. Residual pathway priors for soft equivariance constraints. *Advances in Neural Information Processing Systems*, 34, 2021.

[61] Marc Finzi, Samuel Stanton, Pavel Izmailov, and Andrew Gordon Wilson. Generalizing convolutional neural networks for equivariance to lie groups on arbitrary continuous data. *arXiv preprint arXiv:2002.12880*, 2020.

[62] Marc Finzi, Max Welling, and Andrew Gordon Wilson. A practical method for constructing equivariant multilayer perceptrons for arbitrary matrix groups. *International Conference on Machine Learning*, 2021.

[63] João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM computing surveys (CSUR)*, 46(4):1–37, 2014.

[64] Everette S Gardner Jr. Exponential smoothing: The state of the art. *Journal of forecasting*, 4(1):1–28, 1985.

[65] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2414–2423, 2016.

[66] Rohan Ghosh and Anupam K. Gupta. Scale steerable filters for locally scale-invariant convolutional neural networks. *arXiv preprint arXiv:1906.03861*, 2019.

[67] Dwaraknath Gnaneshwar, Bharath Ramsundar, Dhairya Gandhi, Rachel Kurchin, and Venkatasubramanian Viswanathan. Score-based generative models for molecule generation. In *International Conference on Machine Learning*, 2022.

[68] Giorgio Gnecco and Marcello Sanguineti. Approximation error bounds via rademacher complexity. 2008.

[69] Rakshitha Godahewa, Christoph Bergmeir, Geoffrey I. Webb, Rob J. Hyndman, and Pablo Montero-Manso. Monash time series forecasting archive. In *Neural Information Processing Systems Track on Datasets and Benchmarks*, 2021.

[70] Georg Goerg. Forecastable component analysis. In *International conference on machine learning*, pages 64–72. PMLR, 2013.

[71] E. Grant, Chelsea Finn, S. Levine, Trevor Darrell, and T. Griffiths. Recasting gradient-based meta-learning as hierarchical bayes. *ArXiv Preprint*, abs/1801.08930, 2018.

[72] S. Greydanus, Misko Dzamba, and J. Yosinski. Hamiltonian neural networks. *ArXiv*, abs/1906.01563, 2019.

[73] Ruocheng Guo, Lu Cheng, Jundong Li, P. R. Hahn, and Huan Liu. A survey of learning causality with data. *ACM Computing Surveys (CSUR)*, 53:1 – 37, 2020.

[74] Abhishek Gupta, Russell Mendonca, YuXuan Liu, Pieter Abbeel, and Sergey Levine. Meta-reinforcement learning of structured exploration strategies. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 5307–5316, 2018.

[75] Jiequn Han, Linfeng Zhang, and E. Weinan. Solving many-electron schrödinger equation using deep neural networks. *J. Comput. Phys.*, 399, 2019.

[76] Yiqiang Han, Wenjian Hao, and Umesh Vaidya. Deep learning of koopman representation for control. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pages 1890–1895. IEEE, 2020.

[77] Michael Harradon, Jeff Druce, and Brian E. Ruttenberg. Causal learning and explanation of deep neural networks via autoencoded activations. *ArXiv*, abs/1802.00541, 2018.

[78] Michael Harries and New South Wales. Splice-2 comparative evaluation: Electricity pricing. 1999.

[79] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[80] Geoffrey E Hinton, Alex Krizhevsky, and Sida D Wang. Transforming auto-encoders. In *International conference on artificial neural networks*, pages 44–51. Springer, 2011.

[81] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[82] Philipp M Holl, Kiwon Um, and Nils Thuerey. phiflow: A differentiable pde solving framework for deep learning via physical simulations. In *Workshop on Differentiable Vision, Graphics, and Physics in Machine Learning at NeurIPS*, 2020.

[83] Philip Holmes, John L Lumley, Gahl Berkooz, and Clarence W Rowley. *Turbulence, coherent structures, dynamical systems and symmetry*. Cambridge university press, 2012.

[84] Emiel Hoogeboom, Vıctor Garcia Satorras, Clément Vignac, and Max Welling. Equivariant diffusion for molecule generation in 3d. In *International Conference on Machine Learning*, pages 8867–8887. PMLR, 2022.

[85] B. Houska, F. Logist, M. Diehl, and J. Van Impe. A tutorial on numerical methods for state and parameter estimation in nonlinear dynamic systems. In D. Alberer, H. Hjalmarsson, and L. Del Re, editors, *Identification for Automotive Systems, Volume 418, Lecture Notes in Control and Information Sciences*, page 67–88. Springer, 2012.

[86] Yun Hua, Xiangfeng Wang, Bo Jin, Wenhao Li, Junchi Yan, Xiaofeng He, and Hongyuan Zha. Hmrl: Hyper-meta learning for sparse reward reinforcement learning problem. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 637–645, 2021.

[87] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1501–1510, 2017.

[88] Zijie Huang, Yizhou Sun, and Wei Wang. Coupled graph ode for learning interacting system dynamics. In *27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2021*, pages 705–715, 2021.

[89] Thomas JR Hughes. *The finite element method: linear static and dynamic finite element analysis*. Courier Corporation, 2012.

[90] Jessica Hwang, Paulo Orenstein, Judah Cohen, Karl Pfeiffer, and Lester Mackey. Improving subseasonal forecasting in the western us with machine learning. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2325–2335, 2019.

[91] Rob J Hyndman and George Athanasopoulos. *Forecasting: principles and practice*. OTexts, 2018.

[92] Arieh Iserles. *A first course in the numerical analysis of differential equations*. Number 44. Cambridge university press, 2009.

[93] Eugene M. Izhikevich. *Dynamical systems in neuroscience*. MIT press, 2007.

[94] J.C.Butcher. *Applied Numerical Mathematics*, volume 20. Elsevier B.V., 1996.

[95] Xiaowei Jia, Jared Willard, Anuj Karpatne, Jordan Read, Jacob Zwart, Michael Steinbach, and Vipin Kumar. Physics guided rnns for modeling dynamical systems: A case study in simulating lake temperature profiles. In *Proceedings of the 2019 SIAM International Conference on Data Mining*, pages 558–566. SIAM, 2019.

[96] Xiaoyong Jin, Youngsuk Park, Danielle C Maddix, Yuyang Wang, and Xifeng Yan. Domain adaptation for time series forecasting via attention sharing. *International Conference on Machine Learning*, 2021.

[97] Yongcheng Jing, Yezhou Yang, Zunlei Feng, Jingwen Ye, Yizhou Yu, and Mingli Song. Neural style transfer: A review. *IEEE transactions on visualization and computer graphics*, 26(11):3365–3385, 2019.

[98] Eurika Kaiser, J Nathan Kutz, and Steven L Brunton. Sparse identification of nonlinear dynamics for model predictive control in the low-data limit. *Proceedings of the Royal Society A*, 474(2219):20180335, 2018.

[99] Michael J Kane, Natalie Price, Matthew Scotch, and Peter Rabinowitz. Comparison of arima and random forest time series models for prediction of avian influenza h5n1 outbreaks. *BMC bioinformatics*, 15(1):1–9, 2014.

[100] J. Kani and A. H. Elsheikh. Dr-rnn: A deep residual recurrent neural network for model reduction. *ArXiv*, abs/1709.00939, 2017.

[101] Anuj Karpatne, William Watkins, Jordan Read, and Vipin Kumar. Physics-guided neural networks (pgnn): An application in lake temperature modeling. *arXiv Preprint arXiv:1710.11431*, 2017.

[102] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4401–4410, 2019.

[103] K. Kashinath, M. Mustafa, J-L. Wu A. Albert, C. Jiang, K. Azizzadenesheli S. Esmaeilzadeh, R. Wang, A. Chattopadhyay, A. Singh, A. Manepalli, D. Chirila, R. Yu, R. Walters, B. White, H. Xiao, H. A. Tchelepi, P. Marcus, A. Anandkumar, and P. Hassanzadeh. Physics-informed machine learning: case studies for weather and climate modelling. *Journal of Philosophical Transactions of the Royal Society A*, 2020.

[104] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7482–7491, 2018.

[105] B. Kim, V. C. Azevedo, N. Thürey, Theodore Kim, M. Gross, and B. Solenthaler. Deep fluids: A generative network for parameterized fluid simulations. *Computer Graphics Forum*, 38, 2019.

[106] Ihn S Kim and Wolfgang JR Hoefer. A local mesh refinement algorithm for the time domain-finite difference method using maxwell's curl equations. *IEEE Transactions on Microwave Theory and Techniques*, 38(6):812–815, 1990.

[107] Junhyuk Kim and Changhoon Lee. Deep unsupervised learning of turbulence for inflow generation at various reynolds numbers. *arXiv:1908.10515*, 2019.

[108] Taesung Kim, Jinhee Kim, Yunwon Tae, Cheonbok Park, Jang-Ho Choi, and Jaegul Choo. Reversible instance normalization for accurate time-series forecasting against distribution shift. In *International Conference on Learning Representations*, 2022.

[109] Junhyuk Kima and Changhoon Lee. Deep unsupervised learning of turbulence for inflow generation at various reynolds numbers. *arXiv:1908.10515v1*, 2019.

[110] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[111] D. Kochkov, J. A. Smith, A. Alieva, Qifeng Wang, M. Brenner, and Stephan Hoyer. Machine learning accelerated computational fluid dynamics. *arXiv preprint arXiv:2102.01010*, 2021.

[112] Risi Kondor and Shubhendu Trivedi. On the generalization of equivariance and convolution in neural networks to the action of compact groups. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, volume 80, pages 2747–2755, 2018.

[113] Bernard O Koopman. Hamiltonian systems and transformation in hilbert space. *Proceedings of the National Academy of Sciences*, 17(5):315–318, 1931.

[114] Wouter M. Kouw and Marco Loog. An introduction to domain adaptation and transfer learning. *arXiv preprint arXiv:1812.11806*, 2018.

[115] Aditi Krishnapriyan, Amir Gholami, Shandian Zhe, Robert Kirby, and Michael W Mahoney. Characterizing possible failure modes in physics-informed neural networks. *Advances in Neural Information Processing Systems*, 34, 2021.

[116] Josef Kunes. *Dimensionless physical quantities in science and engineering*. Elsevier, 2012.

[117] J. Kutz. Deep learning in fluid dynamics. *Journal of Fluid Mechanics*, 814:1–4, 2017.

[118] J Nathan Kutz, Joshua L Proctor, and Steven L Brunton. Koopman theory for partial differential equations. *arXiv preprint arXiv:1607.07076*, 2016.

[119] Vitaly Kuznetsov and Mehryar Mohri. Discrepancy-based theory and algorithms for forecasting non-stationary time series. *Annals of Mathematics and Artificial Intelligence*, 88(4):367–399, 2020.

[120] Jan Eric Kyprianidis, John Collomosse, Tinghuai Wang, and Tobias Isenberg. State of the" art": A taxonomy of artistic stylization techniques for images and video. *IEEE transactions on visualization and computer graphics*, 19(5):866–885, 2012.

[121] E. Labourasse and P. Sagaut. Advance in rans-les coupling, a review and an insight on the nlde approach. *Archives of Computational Methods in Engineering*, 11:199–256, 2004.

[122] John H. Lagergren, John T. Nardini, G. Michael Lavigne, E. Rutter, and K. Flores. Learning partial differential equations for biological transport models from noisy spatio-temporal data. *Proceedings of the Royal Society A*, 476, 2020.

[123] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[124] Karel Lenc and Andrea Vedaldi. Understanding image representations by measuring their equivariance and equivalence. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 991–999, 2015.

[125] Aodong Li, Alex Boyd, Padhraic Smyth, and Stephan Mandt. Detecting and adapting to irregular distribution shifts in bayesian online learning. *Advances in Neural Information Processing Systems*, 34:6816–6828, 2021.

[126] Mengnan Li and Lijian Jiang. Deep learning nonlinear multiscale dynamic problems using koopman operator. *Journal of Computational Physics*, 446:110660, 2021.

[127] Yunzhu Li, Hao He, Jiajun Wu, Dina Katabi, and Antonio Torralba. Learning compositional koopman operators for model-based control. 2020.

[128] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Graph kernel network for partial differential equations. *arXiv preprint arXiv:2003.03485*, 2020.

[129] Zongyi Li, Nikola Borislavov Kovachki, Kamyar Azizzadenesheli, Burigede liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*, 2021.

[130] Julia Ling, Andrew Kurzawski, and Jeremy Templeton. Reynolds averaged turbulence modeling using deep neural networks with embedded invariance. *Journal of Fluid Mechanics*, 807:155–166, 2016.

[131] Julia Ling, Andrew Kurzawskim, and Jeremy Templeton. Reynolds averaged turbulence modeling using deep neural networks with embedded invariance. *Journal of Fluid Mechanics*, 2017.

[132] Ori Linial, D. Eytan, and U. Shalit. Generative ode modeling with known unknowns. *Proceedings of the Conference on Health, Inference, and Learning*, 2021.

[133] Vadim Lisitsa, Galina Reshetova, and Vladimir Tcheverda. Finite-difference algorithm with local time-space grid refinement for simulation of waves. *Computational geosciences*, 16(1):39–54, 2012.

[134] Dehao Liu and Yan Wang. Multi-fidelity physics-constrained neural network and its application in materials modeling. *Journal of Mechanical Design*, 141, 2019.

[135] Shikun Liu, Edward Johns, and Andrew J. Davison. End-to-end multi-task learning with attention. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1871–1880, 2019.

[136] Xiao Liu, Spyridon Thermos, Alison Q. O'Neil, and Sotirios A. Tsaftaris. Semi-supervised meta-learning with disentanglement for domain-generalised medical image segmentation. In *MICCAI*, 2021.

[137] Yong Liu, Haixu Wu, Jianmin Wang, and Mingsheng Long. Non-stationary transformers: Rethinking the stationarity in time series forecasting. *arXiv preprint arXiv:2205.14415*, 2022.

[138] Francesco Locatello, Stefan Bauer, Mario Lucic, Gunnar Raetsch, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem. Challenging common assumptions in the unsupervised learning of disentangled representations. In *international conference on machine learning*, pages 4114–4124. PMLR, 2019.

[139] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan. Learning transferable features with deep adaptation networks. In *International conference on machine learning*, pages 97–105. PMLR, 2015.

[140] Yun Long, Xueyuan She, and Saibal Mukhopadhyay. Hybridnet: Integrating model-based and data-driven learning to predict evolution of dynamical systems. *ArXiv Preprint arXiv:1806.07439*, 2019.

[141] Jie Lu, Anjin Liu, Fan Dong, Feng Gu, Joao Gama, and Guangquan Zhang. Learning under concept drift: A review. *IEEE Transactions on Knowledge and Data Engineering*, 31(12):2346–2363, 2018.

[142] Bethany Lusch, J Nathan Kutz, and Steven L Brunton. Deep learning for universal linear embeddings of nonlinear dynamics. *Nature communications*, 9(1):1–10, 2018.

[143] Gurvan Madec. NEMO ocean engine, 2015. Technical Note. Institut Pierre-Simon Laplace (IPSL), France. https://epic.awi.de/id/eprint/39698/1/NEMO_book_v6039.pdf.

[144] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. The m4 competition: 100,000 time series and 61 forecasting methods. *International Journal of Forecasting*, 36(1):54–74, 2020.

[145] Andrey Malinin, Neil Band, German Chesnokov, Yarin Gal, Mark JF Gales, Alexey Noskov, Andrey Ploskonosov, Liudmila Prokhorenkova, Ivan Provilkov, and Vatsal Raina. Shifts: A dataset of real distributional shift across multiple large-scale tasks. *arXiv preprint arXiv:2107.07455*, 2021.

[146] Georg Martius and Christoph H Lampert. Extrapolation and learning equations. *arXiv preprint arXiv:1610.02995*, 2016.

[147] Ricardo P Masini, Marcelo C Medeiros, and Eduardo F Mendes. Machine learning advances for time series forecasting. *Journal of Economic Surveys*, 2021.

[148] Armand Comas Massague, Chi Zhang, Zlatan Feric, Octavia Camps, and Rose Yu. Learning disentangled representations of video with missing data. *arXiv preprint arXiv:2006.13391*, 2020.

[149] Yasuko Matsubara and Yasushi Sakurai. Dynamic modeling and forecasting of time-evolving data streams. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 458–468, 2019.

[150] Daniel McCarthy and Shane T Jensen. Power-weighted densities for time series data. *The Annals of Applied Statistics*, 10(1):305–334, 2016.

[151] J. M. McDonough. *Introductory Lectures on Turbulence*. Mechanical Engineering Textbook Gallery, 2007.

[152] James M McDonough. Introductory lectures on turbulence: physics, mathematics and modeling. 2007.

[153] Arvind Mohan, Don Daniel, Michael Chertkov, and Daniel Livescu. Compressed convolutional lstm: An efficient deep learning framework to model high fidelity 3d turbulence. *arXiv:1903.00033*, 2019.

[154] Pablo Montero-Manso, George Athanasopoulos, Rob J Hyndman, and Thiyanga S Talagala. Fforma: Feature-based forecast model averaging. *International Journal of Forecasting*, 36:86–92, 2020.

[155] Jeremy Morton, Freddie D Witherden, and Mykel J Kochenderfer. Deep variational koopman models: inferring koopman observations for uncertainty-aware dynamics modeling and control. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 3173–3179, 2019.

[156] Tsendsuren Munkhdalai and Hong Yu. Meta networks. *Proceedings of machine learning research*, 70:2554–2563, 2017.

[157] Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nati Srebro. Exploring generalization in deep learning. *Advances in neural information processing systems*, 30, 2017.

[158] Weili Nie, Tero Karras, Animesh Garg, Shoubhik Debhath, A. Patney, Ankit B. Patel, and Anima Anandkumar. Semi-supervised stylegan for disentanglement learning. In *International Conference on Machine Learning*, 2020.

[159] Thomas Brox Olaf Ronneberger, Philipp Fischer. U-net: Convolutional networks for biomedical image segmentation. *arXiv:1512.03385*, 2015.

[160] Boris N Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. N-beats: Neural basis expansion analysis for interpretable time series forecasting. In *International Conference on Learning Representations*, 2019.

[161] Boris N Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. Meta-learning framework with applications to zero-shot time-series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 9242–9250, 2021.

[162] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

[163] Nikolaos Passalis, Anastasios Tefas, Juho Kanniainen, Moncef Gabbouj, and Alexandros Iosifidis. Deep adaptive input normalization for time series forecasting. *IEEE transactions on neural networks and learning systems*, 31(9):3760–3765, 2019.

[164] 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.

[165] Judea Pearl. Causal inference in statistics: An overview. 2009.

[166] Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter Battaglia. Learning mesh-based simulation with graph networks. In *International Conference on Learning Representations*, 2021.

[167] Marc Terracol Pierre Sagaut, Sebastien Deck. *Multiscale and Multiresolution Approaches in Turbulence*. Imperial College Press, 2006.

[168] Shrimai Prabhumoye, Yulia Tsvetkov, Ruslan Salakhutdinov, and Alan W Black. Style transfer through back-translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 866–876, 2018.

[169] Christopher Rackauckas, Yingbo Ma, Julius Martensen, Collin Warner, Kirill Zubov, Rohit Supekar, Dominic Skinner, Ali Ramadhan, and Alan Edelman. Universal differential equations for scientific machine learning. *arXiv preprint arXiv:2001.04385*, 2020.

[170] Maziar Raissi. Deep hidden physics models: Deep learning of nonlinear partial differential equations. *Journal of Machine Learning Research*, 2018.

[171] Maziar Raissi and George Em Karniadakis. Hidden physics models: Machine learning of nonlinear partial differential equations. *Journal of Computational Physics*, 357:125–141, 2018.

[172] 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.

[173] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10561*, 2017.

[174] Alexander Rakhlin and Karthik Sridharan. Statistical learning and sequential prediction. *Book Draft*, 2014.

[175] Chengping Rao, Pu Ren, Yang Liu, and Hao Sun. Discovering nonlinear PDEs from scarce data with physics-encoded learning. In *International Conference on Learning Representations*, 2022.

[176] Markus Reichstein, Gustau Camps-Valls, Bjorn Stevens, Martin Jung, Joachim Denzler, Nuno Carvalhais, and Prabhat. Deep learning and process understanding for data-driven earth system science. *Nature*, 566(7743):195–204, 2019.

[177] Manuel Ruder, Alexey Dosovitskiy, and Thomas Brox. Artistic style transfer for videos. In *German conference on pattern recognition*, pages 26–36. Springer, 2016.

[178] Samuel H. Rudy, Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Data-driven discovery of partial differential equations. *arXiv preprint arXiv:1609.06401*, 2016.

[179] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[180] T. Konstantin Rusch, Siddhartha Mishra, N. Benjamin Erichson, and Michael W. Mahoney. Long expressive memory for sequence modeling. In *International Conference on Learning Representations*, 2022.

[181] Andrei A. Rusu, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, and Raia Hadsell. Meta-learning with latent embedding optimization. In *International Conference on Learning Representations*, 2019.

[182] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. Dynamic routing between capsules. *arXiv preprint arXiv:1710.09829*, 2017.

[183] Pierre Sagaut. *Large Eddy Simulation for Incompressible Flows*. Springer-Verlag Berlin Heidelberg, 2001.

[184] Pierre Sagaut, Sebastien Deck, and Marc Terracol. *Multiscale and Multiresolution Approaches in Turbulence*. Imperial College Press, 2006.

[185] Subham Sahoo, Christoph Lampert, and Georg Martius. Learning equations for extrapolation and control. In *International Conference on Machine Learning*, pages 4442–4450. PMLR, 2018.

[186] Alvaro Sanchez-Gonzalez, J. Godwin, T. Pfaff, Rex Ying, J. Leskovec, and P. Battaglia. Learning to simulate complex physics with graph networks. *ArXiv*, abs/2002.09405, 2020.

[187] Syuhei Sato, Yoshinori Dobashi, Theodore Kim, and Tomoyuki Nishita. Example-based turbulence style transfer. *ACM Transactions on Graphics (TOG)*, 37(4):1–9, 2018.

[188] Victor Garcia Satorras, Emiel Hoogeboom, and Max Welling. E (n) equivariant graph neural networks. In *International Conference on Machine Learning*, pages 9323–9332. PMLR, 2021.

[189] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

[190] Hayden Schaeffer. Learning partial differential equations via data discovery and sparse optimization. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 473(2197):20160446, 2017.

[191] Michael D. Schmidt and Hod Lipson. Distilling free-form natural laws from experimental data. *Science*, 324:81 – 85, 2009.

[192] Kristof Schütt, P. Kindermans, Huziel Enoc Sauceda Felix, Stefan Chmiela, A. Tkatchenko, and K. Müller. Schnet: A continuous-filter convolutional neural network for modeling quantum interactions. In *NIPS*, 2017.

[193] Christopher Schölzel. Nonlinear measures for dynamical systems. June 2019.

[194] Rajat Sen, Hsiang-Fu Yu, and Inderjit S Dhillon. Think globally, act locally: A deep neural network approach to high-dimensional time series forecasting. *Advances in neural information processing systems*, 32, 2019.

[195] Sungyong Seo, Chuizheng Meng, Sirisha Rambhatla, and Y. Liu. Physics-aware spatiotemporal modules with auxiliary tasks for meta-learning. *ArXiv*, abs/2006.08831, 2020.

[196] Chence Shi, Shitong Luo, and Minkai Xu1. Learning gradient fields for molecular conformation generation. *International Conference on Machine Learning*, 2021.

[197] 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. *arXiv:1506.04214*, 2015.

[198] Gregor N. C. Simm, Robert Pinsler, Gábor Csányi, and José Miguel Hernández-Lobato. Symmetry-aware actor-critic for 3d molecular design. In *International Conference on Learning Representations*, 2021.

[199] Tess E Smidt. Euclidean symmetry and equivariance in machine learning. *Trends in Chemistry*, 3(2):82–85, 2021.

[200] Slawek Smyl. A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting. *International Journal of Forecasting*, 36(1):75–85, 2020.

[201] J. Snell, Kevin Swersky, and R. Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, 2017.

[202] Ivan Sosnovik, Michał Szmaja, and Arnold Smeulders. Scale-equivariant steerable networks. In *International Conference on Learning Representations*, 2020.

[203] Nils Thuerey Steffen Wiewel, Moritz Becher. Latent-space physics: Towards learning the temporal evolution of fluid flow. *Computer Graphics Forum*, 38, 2019.

[204] James H Stock and Mark W Watson. Vector autoregressions. *Journal of Economic perspectives*, 15(4):101–115, 2001.

[205] Steven H. Strogatz. *Nonlinear dynamics and chaos: with applications to physics, biology, chemistry, and engineering*. CRC press, 2018.

[206] Qiuling Suo, Jingyuan Chou, Weida Zhong, and Aidong Zhang. Tadanet: Task-adaptive network for graph-enriched meta-learning. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1789–1799, 2020.

[207] Naoya Takeishi, Yoshinobu Kawahara, and Takehisa Yairi. Learning koopman invariant subspaces for dynamic mode decomposition. *Advances in Neural Information Processing Systems*, 30, 2017.

[208] Sebastian Thrun and Lorien Pratt. Learning to learn: Introduction and overview. In *Learning to learn*, pages 3–17. Springer, 1998.

[209] N. Thuerey, K. Weibenow, L. Prantl, and Xiangyu Hu. Deep learning methods for reynolds-averaged navier-stokes simulations of airfoil flows. *arXiv:1810.08217*, 2019.

[210] Jonathan Tompson, Kristofer Schlachter, Pablo Sprechmann, and Ken Perlin. Accelerating eulerian fluid simulation with convolutional networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3424–3433. JMLR. org, 2017.

[211] Jonathan Tompson, Kristofer Schlachter, Pablo Sprechmann, and Ken Perlin. Accelerating Eulerian fluid simulation with convolutional networks. In *ICML'17 Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 3424–3433, 2017.

[212] Sana Tonekaboni, Chun-Liang Li, Sercan O Arik, Anna Goldenberg, and Tomas Pfister. Decoupling local and global representations of time series. In *International Conference on Artificial Intelligence and Statistics*, pages 8700–8714. PMLR, 2022.

[213] JH Tu, CW Rowley, DM Luchtenburg, SL Brunton, and JN Kutz. On dynamic mode decomposition: theory and applications. *arXiv preprint arXiv:1312.0041*, 2013.

[214] Oliver Unke, Mihail Bogojeski, Michael Gastegger, Mario Geiger, Tess Smidt, and Klaus-Robert Müller. Se (3)-equivariant prediction of molecular wavefunctions and electronic densities. *Advances in Neural Information Processing Systems*, 34:14434–14447, 2021.

[215] Tycho FA van der Ouderaa, David W Romero, and Mark van der Wilk. Relaxing equivariance constraints with non-stationary continuous filters. *arXiv preprint arXiv:2204.07178*, 2022.

[216] Simon Vandenhende, Stamatios Georgoulis, Wouter Van Gansbeke, Marc Proesmans, Dengxin Dai, and Luc Van Gool. Multi-task learning for dense prediction tasks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, PP, 2021.

[217] Soledad Villar, David W Hogg, Kate Storey-Fisher, Weichi Yao, and Ben Blum-Smith. Scalars are universal: Equivariant machine learning, structured like classical physics. In *Advances in Neural Information Processing Systems*, 2021.

[218] Oriol Vinyals, Charles Blundell, T. Lillicrap, K. Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. In *NIPS*, 2016.

[219] Pantelis R Vlachas, Jaideep Pathak, Brian R Hunt, Themistoklis P Sapsis, Michelle Girvan, Edward Ott, and Petros Koumoutsakos. Backpropagation algorithms and reservoir computing in recurrent neural networks for the forecasting of complex spatiotemporal dynamics. *Neural Networks*, 126:191–217, 2020.

[220] John Wainwright. *Dynamical systems in cosmology*. Cambridge University Press, 2005.

[221] Robin Walters, Jinxi Li, and Rose Yu. Trajectory prediction using equivariant continuous convolution. In *International Conference on Learning Representations*, 2021.

[222] Dian Wang, Robin Walters, Xupeng Zhu, and Robert Platt. Equivariant q learning in spatial action spaces. In *5th Annual Conference on Robot Learning*, 2021.

[223] Rui Wang, E. Huang, Uma Chandrasekaran, and Rose Yu. Aortic pressure forecasting with deep learning. *2020 Computing in Cardiology*, pages 1–4, 2020.

[224] Rui Wang, Karthik Kashinath, Mustafa Mustafa, Adrian Albert, and Rose Yu. Towards physics-informed deep learning for turbulent flow prediction. *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2020.

[225] Rui Wang, Danielle Maddix, Christos Faloutsos, Yuyang Wang, and Rose Yu. Bridging physics-based and data-driven modeling for learning dynamical systems. In *Learning for Dynamics and Control*, pages 385–398. PMLR, 2021.

[226] Rui Wang, Robin Walters, and Rose Yu. Incorporating symmetry into deep dynamics models for improved generalization. 2021.

[227] Rui Wang, Robin Walters, and Rose Yu. Meta-learning dynamics forecasting using task inference. *arXiv preprint arXiv:2102.10271*, 2021.

[228] Rui Wang, Robin Walters, and Rose Yu. Approximately equivariant networks for imperfectly symmetric dynamics. *International Conference on Machine Learning (ICML)*, 2022.

[229] Rui Wang, Robin Walters, and Rose Yu. Data augmentation vs. equivariant networks: A theoretical study of generalizability on dynamics forecasting. *International Conference on Machine Learning, Principles of Distribution Shift Workshop*, 2022.

[230] Yunbo Wang, Mingsheng Long, Jianmin Wang, Zhifeng Gao, and Philip S Yu. PredRNN: Recurrent neural networks for predictive learning using spatiotemporal LSTMs. In *Advances in Neural Information Processing Systems*, pages 879–888, 2017.

[231] Yuyang Wang, Alex Smola, Danielle Maddix, Jan Gasthaus, Dean Foster, and Tim Januschowski. Deep factors for forecasting. In *International conference on machine learning*, pages 6607–6617. PMLR, 2019.

[232] Maurice Weiler and Gabriele Cesa. General E(2)-equivariant steerable CNNs. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 14334–14345, 2019.

[233] Maurice Weiler, Mario Geiger, Max Welling, Wouter Boomsma, and Taco Cohen. 3d steerable cnns: Learning rotationally equivariant features in volumetric data. In *NeurIPS*, 2018.

[234] Maurice Weiler, Fred A. Hamprecht, and Martin Storath. Learning steerable filters for rotation equivariant CNNs. *Computer Vision and Pattern Recognition (CVPR)*, 2018.

[235] Mike West and Jeff Harrison. *Bayesian forecasting and dynamic models*. Springer Science & Business Media, 2006.

[236] Jared Willard, Xiaowei Jia, Shaoming Xu, Michael S. Steinbach, and Vipin Kumar. Integrating physics-based modeling with machine learning: A survey. *CoRR*, abs/2003.04919, 2020.

[237] Garrett Wilson, Janardhan Rao Doppa, and Diane J Cook. Multi-source deep domain adaptation with weak supervision for time-series sensor data. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1768–1778, 2020.

[238] Annette Witt, Jürgen Kurths, and A Pikovsky. Testing stationarity in time series. *physical Review E*, 58(2):1800, 1998.

[239] Gerald Woo, Chenghao Liu, Doyen Sahoo, Akshat Kumar, and Steven Hoi. CoST: Contrastive learning of disentangled seasonal-trend representations for time series forecasting. In *International Conference on Learning Representations*, 2022.

[240] Daniel Worrall and Max Welling. Deep scale-spaces: Equivariance over scale. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 7364–7376, 2019.

[241] Daniel E Worrall, Stephan J Garbin, Daniyar Turmukhambetov, and Gabriel J Brostow. Harmonic networks: Deep translation and rotation equivariance. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5028–5037, 2017.

[242] Dongxian Wu, Liyao Gao, X. Xiong, Matteo Chinazzi, Alessandro Vespignani, Y. Ma, and Rose Yu. Deepgleam: a hybrid mechanistic and deep learning model for covid-19 forecasting. *arXiv preprint arXiv: 2102.06684*, 2021.

[243] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in Neural Information Processing Systems*, 34:22419–22430, 2021.

[244] Jin-Long Wu, Karthik Kashinath, Adrian Albert, Dragos Chirila, Prabhat, and Heng Xiao. Enforcing Statistical Constraints in Generative Adversarial Networks for Modeling Chaotic Dynamical Systems. *arXiv e-prints*, May 2019.

[245] You Xie, Erik Franz, Mengyu Chu, and Nils Thuerey. tempogan: A temporally coherent, volumetric gan for super-resolution fluid flow. *ACM Transactions on Graphics (TOG)*, 37(4):95, 2018.

[246] Dan Xu, Wanli Ouyang, Xiaogang Wang, and N. Sebe. Pad-net: Multi-tasks guided prediction-and-distillation network for simultaneous depth estimation and scene parsing. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 675–684, 2018.

[247] Huaxiu Yao, Ying Wei, Junzhou Huang, and Zhenhui Li. Hierarchically structured meta-learning. In *International Conference on Machine Learning*, pages 7045–7054. PMLR, 2019.

[248] Enoch Yeung, Soumya Kundu, and Nathan Hodas. Learning deep neural network representations for koopman operators of nonlinear dynamical systems. In *2019 American Control Conference (ACC)*, pages 4832–4839. IEEE, 2019.

[249] Yuan Yin, Vincent LE GUEN, Jérémie DONA, Emmanuel de Bezenac, Ibrahim Ayed, Nicolas THOME, and patrick gallinari. Augmenting physical models with deep networks for complex dynamics forecasting. In *International Conference on Learning Representations*, 2021.

[250] Jaesik Yoon, Taesup Kim, Ousmane Dia, Sungwoong Kim, Yoshua Bengio, and Sungjin Ahn. Bayesian model-agnostic meta-learning. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 7343–7353, 2018.

[251] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations*, 2017.

[252] Wei Zhang. Shift-invariant pattern recognition neural network and its optical architecture. In *Proceedings of annual conference of the Japan Society of Applied Physics*, 1988.

[253] Wei Zhang, Kazuyoshi Itoh, Jun Tanida, and Yoshiki Ichioka. Parallel distributed processing model with local space-invariant interconnections and its optical architecture. *Applied optics*, 29(32):4790–4797, 1990.

[254] Rong Zheng, Sicun Gao, and Rose Yu. Lyapunov regularized forecaster. *Machine Learning and the Physical Sciences workshop, NeurIPS 2022.*, 2022.

[255] Allan Zhou, Tom Knowles, and Chelsea Finn. Meta-learning symmetries by reparameterization. *arXiv preprint arXiv:2007.02933*, 2020.

[256] Allan Zhou, Tom Knowles, and Chelsea Finn. Meta-learning symmetries by reparameterization. In *International Conference on Learning Representations*, 2021.

[257] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 11106–11115, 2021.

[258] Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. *arXiv preprint arXiv:2201.12740*, 2022.