Physics-Aware Deep Learning and Dynamical Systems : Hybrid Modeling and Generalization

Yuan Yin

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS
DEPUIS 1257

Sorbonne Université

Doctoral School **École Doctorale Informatique, Télécommunications et Electronique (ED130)**

Joint Research Unit **Institut des Systèmes Intelligents et de Robotique**

Thesis defended by **Yuan Yin**

Defended on **June 28, 2023**

In order to obtain the degree of Doctor of Sorbonne Université

Academic Field **Information and Communication Sciences and Technologies**

Physics-Aware Deep Learning and Dynamical Systems: Hybrid Modeling and Generalization

Thesis supervised by	Patrick Gallinari	Supervisor
	Nicolas Baskiotis	Co-Monitor

Committee members

Referees	Alexandros Kalousis	Professor at Haute École spécialisée de Suisse occidentale	
	Christian Wolf	Principal Scientist at Naver Labs Europe	
Examiners	Paola Cinnella	Professor at Sorbonne Université	Committee President
	Gustau Camps-Valls	Professor at Universitat de València	
	Corentin Lapeyre	Senior Researcher at Centre européen de recherche et de formation avancée en calcul scientifique	
Supervisors	Patrick Gallinari	Professor at Sorbonne Université	
	Nicolas Baskiotis	Associate Professor at Sorbonne Université	

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

Sorbonne Université

École doctorale **École Doctorale Informatique, Télécommunications et Electronique (ED130)**

Unité mixte de recherche **Institut des Systèmes Intelligents et de Robotique**

Thèse présentée par **Yuan Yin**

Soutenue le **28 juin 2023**

En vue de l'obtention du grade de Docteur de Sorbonne Université

Discipline **Sciences et technologies de l'information et de la communication**

Apprentissage profond pour la physique et les systèmes dynamiques : Modélisation hybride et généralisation

Thèse dirigée par Patrick Gallinari directeur
Nicolas Baskiotis co-encadrant

Composition du jury

Rapporteurs	Alexandros Kalousis	professeur à la Haute École spécialisée de Suisse occidentale	
	Christian Wolf	scientifique principal chez Naver Labs Europe	
Examinateur·rice·s	Paola Cinnella	professeure à Sorbonne Université	présidente du jury
	Gustau Camps-Valls	professeur à l'Universitat de València	
	Corentin Lapeyre	chercheur sénior au Centre européen de recherche et de formation avancée en calcul scientifique	
Directeurs de thèse	Patrick Gallinari	professeur à Sorbonne Université	
	Nicolas Baskiotis	MCF à Sorbonne Université	

Keywords: deep learning, dynamical system, physical phenomenon, neural network, hybrid modeling, generalization, adaptation, out-of-distribution, continuous dynamics modeling

Mots clés : apprentissage profond, système dynamique, phénomène physique, réseau de neurones, modélisation hybride, généralisation, adaptation, hors distribution, modélisation dynamique continue

This thesis has been prepared at

Institut des Systèmes Intelligents et de Robotique
Campus Pierre et Marie Curie
Pyramide, Tour 55
4, place Jussieu
75005 Paris
France

☎ +33 (0)1 44 27 51 41
🖷 +33 (0)1 44 27 51 45
✉ contact@isir.upmc.fr
Web Site https://www.isir.upmc.fr/

Dédié à ma mère et à ma famille
献给我的母亲和我的家人

Physics-Aware Deep Learning and Dynamical Systems: Hybrid Modeling and Generalization

Abstract

Deep learning has made significant progress in various fields and has emerged as a promising tool for modeling physical dynamical phenomena that exhibit highly nonlinear relationships. However, existing approaches are limited in their ability to make physically sound predictions due to the lack of prior knowledge and to handle real-world scenarios where data comes from multiple dynamics or is irregularly distributed in time and space. This thesis aims to overcome these limitations in the following directions: improving neural network-based dynamics modeling by leveraging physical models through hybrid modeling; extending the generalization power of dynamics models by learning commonalities from data of different dynamics to extrapolate to unseen systems; and handling free-form data and continuously predicting phenomena in time and space through continuous modeling. We highlight the versatility of deep learning techniques, and the proposed directions show promise for improving their accuracy and generalization power, paving the way for future research in new applications.

Keywords: deep learning, dynamical system, physical phenomenon, neural network, hybrid modeling, generalization, adaptation, out-of-distribution, continuous dynamics modeling

---

Apprentissage profond pour la physique et les systèmes dynamiques : Modélisation hybride et généralisation

Résumé

L'apprentissage profond a fait des progrès dans divers domaines et est devenu un outil prometteur pour modéliser les phénomènes dynamiques physiques présentant des relations hautement non linéaires. Cependant, les approches existantes sont limitées dans leur capacité à faire des prédictions physiquement fiables en raison du manque de connaissances préalables et à gérer les scénarios du monde réel où les données proviennent de dynamiques multiples ou sont irrégulièrement distribuées dans le temps et l'espace. Cette thèse vise à surmonter ces limitations dans les directions suivantes : améliorer la modélisation de la dynamique basée sur les réseaux neuronaux en exploitant des modèles physiques grâce à la modélisation hybride ; étendre le pouvoir de généralisation des modèles de dynamique en apprenant les similitudes à partir de données de différentes dynamiques pour extrapoler vers des systèmes invisibles ; et gérer les données de forme libre et prédire continuellement les phénomènes dans le temps et l'espace grâce à la modélisation continue. Nous soulignons la polyvalence des techniques d'apprentissage profond, et les directions proposées montrent des promesses pour améliorer leur précision et leur puissance de généralisation, ouvrant la voie à des recherches futures dans de nouvelles applications.

Mots clés : apprentissage profond, système dynamique, phénomène physique, réseau de neurones, modélisation hybride, généralisation, adaptation, hors distribution, modélisation dynamique continue

---

Institut des Systèmes Intelligents et de Robotique
Campus Pierre et Marie Curie – Pyramide, Tour 55 – 4, place Jussieu – 75005 Paris – France

Remerciements

Acknowledgments – 致谢

Profondément marquée par la pandémie de Covid, cette thèse a dû se dérouler dans des circonstances particulières, entre le télé-travail, l'enseignement à distance et l'impossibilité d'assister aux conférences en personne. Sans le formidable soutien de celles et ceux qui m'entourent, je n'aurais pas pu mener à bien cette belle et acharnée aventure.

Tout d'abord, je dois absolument remercier Patrick, mon directeur de thèse, qui a été d'une aide précieuse tout au long de mes recherches. Sa confiance en moi, ses précieux conseils, son encouragement à collaborer avec les autres doctorants et sa patience en cas de besoin ont été d'une grande aide. Il a su nous aider à sortir de l'impasse à de nombreuses occasions clés dans la conception de nos projets. Je tiens également à le remercier pour les discussions stimulantes que nous avons eues sur divers sujets. Je voudrais exprimer mes remerciements à Nicolas, mon co-encadrant de thèse, pour son suivi constant tout au long de cette expérience. Ses conseils et son soutien ont été précieux pour ma recherche.

Je remercie ensuite l'ensemble des membres du jury, qui m'ont fait l'honneur de bien vouloir étudier avec attention mon travail.

I would like to thank all the committee members who honored me by carefully reviewing my work. First and foremost, I extend my gratitude to Alexandros Kalousis and Christian Wolf who agreed to serve as referees for my thesis. Their constructive feedback and support were invaluable to me. I am also grateful to Paola Cinnela, Gustau Camps-Valls, and Corentin Lapeyre for kindly agreeing to be part of the jury and for their participation in the evaluation of my thesis.

Je tiens à exprimer ma sincère gratitude envers tous mes collègues avec lesquels j'ai eu la chance de collaborer dans différents projets de recherche. Arthur, Emmanuel, Ibrahim, Jean-Yves, Jérémie, Matthieu, ainsi que toutes les personnes qui ont contribué indirectement à mes projets, ont joué un rôle essentiel dans la réussite de mes recherches. Leurs idées, leur expertise, leur énergie et leur passion ont été une source d'inspiration pour moi

tout au long de ces projets. Leur présence m'a permis de grandir professionnellement et personnellement, et a rendu mon expérience de recherche unique. Je n'oublierai jamais les moments forts et les difficultés que nous avons traversés ensemble. Merci à vous tous, mes chers collègues, pour votre collaboration et votre amitié. Vos contributions ont été inestimables et ont permis la réussite de mes projets de recherche.

Je suis également de tout cœur avec tous les membres, anciens et actuels, de l'équipe MLIA en leur disant combien je suis reconnaissant pour leur accueil des plus chaleureux depuis mon arrivée. Vos verres de bière et de pastis, vos fêtes et vos parties de baby-foot ont contribué à rendre mon intégration au sein de cette équipe sympathique, accueillante et agréable, encore plus facile. Je me considère chanceux de travailler et de vivre en votre compagnie, et j'apprécie grandement les liens d'amitié que nous avons tissés.

Enfin et surtout, je voudrais remercier ma mère et toute ma famille.

我要感谢我母亲一直以来的关心和心理支撑。她无论相隔万里，都一直支持和关心我。在我远离家乡，在法国生活的时候，她一直是我最可靠的后盾。她信任我在法国的经历和生活，理解和尊重我的选择，并给予我无尽的鼓励和支持。我感到非常幸运拥有这样一位伟大的母亲，我会永远珍惜她的爱和关心。除了我的母亲，我还要感谢我的家人对我的关注和爱护。每次和家人通话，他们都会关心我的近况，询问我的生活情况。我知道，无论在哪里，我的家人都会一直支持我，关心我，给予我无私的爱。他们的关心和鼓励给了我前进的力量和信心，我非常感激他们。

Mes années d'études en France depuis l'automne 2016 ont été une expérience inoubliable et éclairante. Les différentes opportunités académiques, culturelles et sociales que j'ai eues dans ce pays ont été déterminantes dans la personne que je suis devenue aujourd'hui.

Mes études en France m'ont permis d'acquérir une connaissance approfondie de mon domaine, mais également de la langue, la culture, l'histoire et les traditions françaises. J'ai également eu l'opportunité de rencontrer des personnes provenant de différents horizons et de différentes cultures, ce qui a enrichi mon expérience personnelle et m'a permis de développer une ouverture d'esprit envers les différences culturelles. En outre, mes études en France ont été une source d'inspiration pour mon avenir professionnel, m'aidant à définir mes objectifs de carrière et à déterminer les compétences que je souhaite acquérir pour atteindre mes objectifs.

En somme, toutes ces expériences uniques ont eu un impact profond sur ma vie et définiront certainement mon avenir.

Symbols

Symbol	Name	Description
d_θ, d_α	dimension of a vector	
e	environment	
$\mathcal{E}$	set of environments	$e \in \mathcal{E}$
Ψ	flow	
f, g	function	
$\mathcal{F}, \mathcal{G}$	set of functions	$f \in \mathcal{F}, g \in \mathcal{G}$
z	function input	
$\mathcal{Z}$	set of function inputs	$z \in \mathcal{Z}$
y	function output	
$\mathcal{Y}$	set of function outputs	$y \in \mathcal{Y}$
$\mathcal{L}$	loss function	
ℓ	sample-wise loss function	
θ	model parameter	
Θ	set of parameters	$\theta \in \Theta$
$\mathcal{R}$	regularization function	
x	spatial coordinates	
p	dimension of the spatial domain	$\Omega \subset \mathbb{R}^p$
Ω	spatial domain	$x \in \Omega$
$\mathcal{X}$	discretized spatial domain	$\mathcal{X} \subset \Omega$
t	temporal coordinate	
$\mathcal{I}$	temporal domain	$t \in \mathcal{I} \subset \mathbb{R}, \exists t_0 \in \mathcal{I}$
$\mathcal{T}$	discretized temporal domain	$\mathcal{T} \subset \mathcal{I}$
u	continuous trajectory	
$u\|_{\mathcal{T}}$	discretized trajectory	
$\rho_{\mathcal{D}}$	trajectory of distribution	
Γ	set of continuous trajectories	$u \in \Gamma$

Symbol	Name	Description
$\mathcal{D}$	dataset of discretized trajectories	$u\|_{\mathcal{T}} \in \mathcal{D}$
K	number of frames per sequence	
N	number of sequences in a dataset	
u_t	state at time t	
d	number of variables in a state	$\#(u_t(x)) = d$
$\mathcal{U}$	set of states	$u_t \in \mathcal{U}$

Table of Contents

List of Figures

List of Tables

Part I

Foundation

Chapter 1

Introduction

1.1 Motivating Dynamics Modeling with Data

Dynamics modeling is a fundamental area of study in science, spanning centuries of research across multiple disciplines. At its core, dynamics modeling seeks to explain and predict the movement or changes of objects through observation and analysis. The insights gained from this understanding have led to the development of effective models that can predict how a phenomenon will evolve over time, and how it may respond to changes in the environment or other conditions. Today, efforts to model dynamical systems cover a wide spectrum of methods, involving varying degrees of first principles: from sophisticated

numerical models, relying on a full knowledge of physics, to purely data-driven models without any prior physical knowledge. We illustrate the relationship between different modeling modes in Figure 1.1.

In this thesis, we contribute to the recent trend of applying deep learning (DL) to dynamics modeling. Following the breakthroughs made by deep learning in various fields (He et al., 2016; Goodfellow et al., 2020; Devlin et al., 2019; Brown et al., 2020), the versatility of deep learning has started being exploited for various aspects of dynamics modeling, as shown in Figure 1.1. However, this research domain is still at an early stage of development.

Our goal is to enhance the generality and adaptability of data-driven DL models by considering physical or distributional priors in their underlying assumptions. We aim to exploit DL approaches to obtain models that can efficiently extract information from the data close to real-world scenarios. This introductory chapter emphasizes the importance of modeling with data and motivates the different problems we want to address, to match model capabilities with numerical methods.

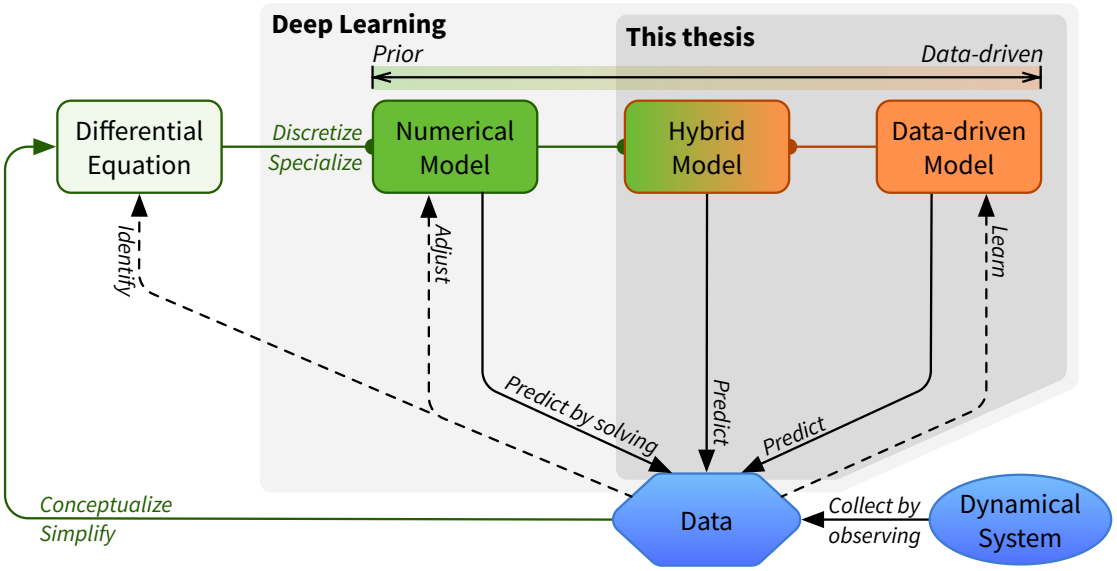
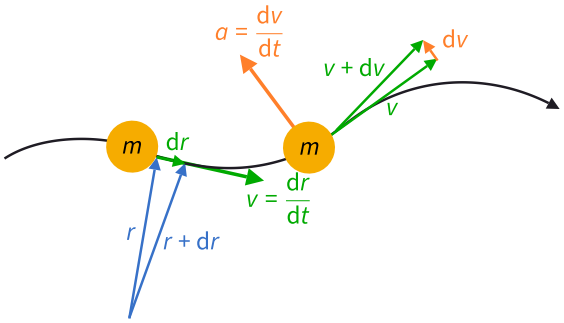


Figure 1.1: Three paradigms of dynamics models: (a) *numerical models*, based on mathematical differential equations formulated from first-principles laws, are then solved by numerical solvers. They must first be identified from the data and then solved by numerical solvers to be able to predict; (b) *data-driven models*, which learn dynamics from data automatically and can predict after being trained; (c) *hybrid models*, which combine the two previous types of model.

1.1.1 Understanding Nature with Differential Equations

The study of dynamics began with the development of mathematical models, aiming to understand the movements of objects by modeling physical laws. From Galileo to the present day, scientists have attempted to explain the observed phenomena by constructing dynamics models that express, e.g., laws of motion, with object properties, e.g., mass, length, and density. These models have been created with concepts abstracted from the observations and based on the human perception of both reality and physical laws.

For example, as illustrated in Figure 1.2, when modeling the behavior of a moving object, instead of exhaustively describing every evolution trajectory of the object's position, one studies the velocity, i.e., the *rate of change* of position, and the acceleration, i.e., the *rate of change* of the velocity. As a result, the rules concerning these rates of change, i.e., *differential equations*, provide a clear, concise, and conceivable ideal description of the temporal evolution of the phenomena and are considered as laws of physics.



Credit: Maschen, CC0, via Wikimedia Commons

Figure 1.2: A classical particle of mass m moves through space along the trajectory ⟶. The quantities describing the kinetic state: position r ⟶, velocity v ⟶, acceleration a ⟶.

To date, a large number of dynamics models formulated as differential equations have been proposed in various disciplines. For example, in physics, Navier-Stokes equations (Stokes, 1851) describe the motion of viscous fluids, e.g., water, air; reaction-diffusion equations (Pearson, 1993) describe how one or more chemical substances react locally and spread in space (Feinberg, 2019); epidemiological dynamics studying how infectious diseases spread in a population (Martcheva, 2015). They rely on a profound understanding of the underlying dynamical phenomena.

If an analytical solution to the differential equation is available, we can use it to predict what will happen. However, since most dynamical systems are nonlinear, their behavior

cannot be fully described by analytical methods alone. As a result, we usually have to resort to numerical methods to be able to make use of the differential equations, which has led to the development of numerical models that can provide an approximation to the intractable analytical solution.

1.1.2 Numerical Models

A *numerical model* is a discretized or simplified version of the differential equation. These models are then solved through a *numerical solver*. The approximate solutions provided by the numerical models and their solvers retain the physical properties of the phenomenon and are used for a wide range of applications.

The numerical models have the following robustness and flexibility properties, which will greatly influence the capability requirements when proposing other types of models, including DL approaches:

- *They are often robust to state changes.* The close link to the differential equation makes the numerical models applicable to a large range of state values. This allows the same model to be applied under different conditions.
- *They describe different systems, obeying the same physical laws under different contexts, often parametrized by a few parameters.* Switching from one system to another can be done directly by modifying parameters. For example, when moving a pendulum for the Earth to the planet Mars, it is enough to change the gravity in the equation to predict the pendulum's movements in the new environment.
- *The approximations are close to physical reality and reflect important properties of the dynamics.* When predicting physical quantities, the solutions should also respect basic physical laws. For example, symmetries in time and space for state changes; conservation of volume and energy; incompressibility for fluid, etc.

However, confronting reality is never easy. Despite many efforts to build realistic differential equations, numerical models, and solvers, many bottlenecks still hinder their application to real-world forecasting:

- *Rare consensus on modeling and solving the same phenomenon.* For many phenomena, it is almost impossible to obtain equations that are unanimously accepted by scientists. For example, for shallow water equations, a.k.a. Saint-Venant equations, there is no consensus on the realistic modeling of the friction (Delestre, 2010,

Chapter 3). A collection of models is proposed to describe the epidemic dynamics of the same phenomenon (Schneckenreither et al., 2008). Sometimes the model and the solver should be adapted for each parametrization or even tailored for different conditions. For example, to solve the Navier-Stokes equation, different numerical models should be selected depending on the physical parameters of the model such as the kinetic viscosity, the domain shape/geometry, the initial flow velocity, etc.

- *A numerical model can only describe a part of reality.* Differential equations and numerical dynamics models are constructed with varying degrees of simplification of the reality. Due to practical constraints, the numerical models are often over-simplified w.r.t. the observed dynamics, making them unsuitable for long-term prediction in the real world.

- *High-quality predictions require significant computational power.* When dealing with complex phenomena that require consideration of small-scale details, achieving high-quality predictions often demands significant computational power. One such instance is solving the Navier-Stokes equations, which entails densely discretizing the domain to ensure accurate predictions. In the experiments of direct numerical simulation, it may require a discretized domain with over 10^{11} degrees of freedom (Lee et al., 2013), and handling the resulting computational cost and memory load demands the use of extremely powerful computing resources, even at medium Reynolds numbers (the higher the Reynolds number, the more chaotic the fluid dynamics).

These bottlenecks limit the real-world applications of numerical models and drive scientists to seek assistance to better describe reality, reduce the cost of running these models, and search for complementary counterparts or even alternatives to numerical models.

1.1.3 Modeling with Data: Toward Broader Applications

In the past decades, the increasing availability of data, generated either by observing the real phenomenon of dynamical systems or by solving equations with numerical models, fostered the development of data-driven machine learning (ML) approaches. Depending on the data source and the goals of the community, different paths have been proposed.

The first type of data consists of information collected by observing real phenomena, such as satellite images of the ocean or the Earth's atmosphere. The data is used to guide numerical models to make plausible predictions consistent with new observations.

The process is called data assimilation (DA) (Kalman, 1960; Courtier et al., 1994), and is a sequential time-stepping procedure. The common strategy involves two steps that are repeated at each update: the numerical model provides a short-term forecast that is compared with newly received observations, and the model state is then updated to reflect the observations. DA has been successfully used in large-scale weather forecasting systems such as the European Centre for Medium-Range Weather Forecasts (ECMWF; Bonavita and Lean, 2021) and operational ocean monitoring and forecasting systems such as MERCATOR Océan (Ferry et al., 2007). DA can effectively find the best estimate of the reality from the numerical model forecast w.r.t. the data. However, the information from data plays the role of correction to the numerical model, not as a part of it.

The second type of data concerns numerical model solutions. Simulations may be used to build emulator models that are less complex and less computationally expensive than numerical models while maintaining a reasonable quality of the solutions. Compared to the original numerical models, namely full-order models (FOMs), the data-driven reduced order models (ROMs) extract a model of lower complexity from a set of solutions given by FOMs. ROMs are mainly related to dimension reduction techniques accompanied by physical assumptions. For dynamical systems, there are techniques like proper orthogonal decomposition (POD) to find the best low-dimensional approximation of the FOM (Choi et al., 2021), and dynamic mode decomposition (DMD) for the best linear approximation of nonlinear dynamics (Kutz et al., 2016). They provide mathematical representations for real-time analysis, but the most effective ones are often limited to linear reduction and linear modeling, which limits their application scenarios and requires extensions to more flexible modeling.

1.1.4 Deep Learning and Dynamics Modeling

Although neural networks were proposed over half a century ago, it is only since the 2010s that they have been widely popularized and that this branch of study, under the name of deep learning, has become the most influential machine learning research topic so far. Thanks to flexible algorithms, enormous amounts of available data, advanced versatile software, and powerful hardware for large-scale model deployment, etc., researchers have achieved multiple breakthroughs in various tasks, e.g., image recognition (He et al., 2016), generation (Goodfellow et al., 2020), text representation (Devlin et al., 2019), generation (Brown et al., 2020). Deep learning highlights the versatile and flexible neural network algorithms for modeling any nonlinear mapping between the given data and the desired

output, achieving operational ML models that were not possible before. This provides a path to effectively extract highly nonlinear complex dynamics directly from data.

In the pre-deep learning era, there were already attempts to propose neural network-based predictive models for dynamical systems. Combining neural networks with numerical integration schemes was also considered. However, learning dynamical systems driven by linear or nonlinear ordinary differential equation (ODE) or partial differential equation (PDE), e.g., Rico-Martínez and Kevrekidis (1993), remained relatively confidential at that time.

Thanks to the advances in the general deep learning industry, recent efforts have rejuvenated this research, namely *deep learning for dynamical systems*. Pioneering work has been done on the integration of physical prior information into the DL/ML models for dynamical systems (Long et al., 2018b; de Bézenac et al., 2018; Raissi et al., 2019; Brunton and Kutz, 2022) by focusing on the use of DL techniques.

From the deep learning side, the connection between dynamical systems and modern neural network architectures has been highlighted by the residual neural networks (ResNets; He et al., 2016), popular in computer vision (CV). Indeed, residual connections implement a forward Euler time step method. Authors, notably E (2017), use this relationship to motivate the use of neural network as solvers for ODEs:

$$z_{t+\delta t} = z_t + f_\theta(t, z_t) \qquad \text{vs.} \qquad \frac{\mathrm{d}z}{\mathrm{d}t}(t) = f_\theta(t, z_t)$$

Following this path, architectures were proposed exploiting ODE theories that guarantee certain physical properties, such as system stability or energy conservation (Haber and Ruthotto, 2017; Ruthotto and Haber, 2020).

The work that first attracted the attention of both the deep learning and dynamics modeling communities is neural ordinary differential equation (Neural ODE; Chen et al., 2018). By replacing traditional ResNets with a numerical solver, it sparked the interest of both communities and reignited the idea of integrating neural networks into differentiable solvers. As a result, the field of DL modeling for dynamical systems and differential equations has experienced a significant growth spurt and continues to make impressive advancements to this day.

Today, deep learning intervenes in almost every type of dynamics modeling shown in Figure 1.1:

- *Act as a purely data-driven forecasting model.* This is the basic setting in deep learning: discovering laws from data. Models are trained from observed spatiotemporal trajectories to forecast future observations (Chen et al., 2018; Pfaff et al., 2021; Li et al., 2021b; Brandstetter et al., 2022);
- *Act as a numerical model.* This objective here is to find alternatives to numerical models and their solvers. One uses the fully known differential equation and conditions to find the unknown solution without data, such as Sirignano and Spiliopoulos (2018); Raissi et al. (2019) and similar work;
- *Act as part of a hybrid model.* To enhance low-cost numerical models to approximate high-precision numerical simulation (Belbute-Peres et al., 2020; Kochkov et al., 2021) or to complement a numerical model by providing some components of this model (de Bézenac et al., 2018; Ayed et al., 2022).

1.2 Real-World Challenges

Despite recent breakthroughs in deep learning for physical modeling, several problems related to the capabilities of learned dynamics models and training data remain under-addressed.

1.2.1 Challenges in Model Requirements

ML dynamics models have become increasingly important in various scientific and engineering fields. However, they face several challenges in terms of accuracy and adaptability, which are critical for their successful application:

Model should be applicable to unseen situations. A fundamental aspect of a successful ML model is its ability to generalize well. This entails using a trained model to make accurate predictions even in situations it has not seen before, which is essential for real-world applications. In the case of a dynamical system, this means that the model should be able to make accurate predictions even when presented with new initial states that it has not encountered during training. In other words, the model should, in particular, be able robust to changes in the distribution of the initial states over time.

Model should be easily adaptable to unseen dynamics. In addition, a good dynamics model should also be able to adapt to changes in the underlying dynamics of the system. If

the dynamics slightly change in the future, or under other circumstances, the model should be able to be adapted quickly with few observed data, without complete retraining of the model.

Model should output physically sound predictions. The estimated states should not only visually resemble reality, but they should also comply with the properties indicated by the laws of physics and real measurements of quantities of interest. For instance, accurate predictions of key physical quantities at the boundary layer of an airfoil are critical and necessitate precise forecasts in specific regions. Guaranteeing that the model produces physically sound predictions is essential to ensure its practical applicability, especially in contexts where accuracy and reliability are critical.

1.2.2 Challenges in Data

ML methods learn dynamics from data. However, the data in the real world is complex and we face also multiple challenges associated with it, in terms of their quantity, variety, and irregularity:

Data is not always abundant in certain scenarios. Given the complexity of modern neural network architectures and their nonlinear nature, most DL methods are prone to generalization issues, especially when training data is scarce. This is still the case of interest to us because, in many scenarios, it is still difficult and costly to obtain data for realistic dynamical systems due to the computational infrastructure, e.g., supercomputers for large-scale simulations. In addition, even if the data exists, it might be owned by governmental and industrial entities, and its availability is limited by confidentiality policies, a.k.a. the closed source data problem. This situation is very different from the problems in other subfields, such as computer vision and natural language processing (NLP), which have a large amount of openly available data thanks to the Internet, forcing us, at least for the next decade, to consider less data-intensive models containing more regularities that favor the application to new data.

Data is from heterogeneous sources. In most cases, the retrieved data need not be uniformly distributed in the space of input-output pairs. This means we should come across data from different sources, of different quality/resolution, and even from different dynamics. Especially in the latter case, we often retrieve data of the same physical phenomenon from different environments, where each environment has its own instantiated

dynamics. For example, we observe epidemics spreading various infectious diseases in countries. They all follow the same general law, but the observed values depend on some factors, such as the infectiousness of the disease and the population structure. This means that even with the same initial condition, the resulting evolution trajectories will vary according to system instances, which may depend on the parameterization of the system:

- *They may differ in evolution laws.* This means that the underlying dynamics are fundamentally different. For example, ocean dynamics at different locations have the corresponding Coriolis force depending on the latitude on Earth. This will be one of the main subjects of this thesis.

- *They may differ in conditions determining the trajectory.* Even with the same dynamics, varying certain conditions causes changes in the trajectory space. For example, changing the temperatures on the domain boundary of the heat equations will lead to different solutions.

- *They may differ in shape and geometry of the system state.* Modeling spatiotemporal dynamics can be challenging, particularly when dealing with complex geometries. For example, the fluid dynamics around an airfoil, e.g., an airplane wing or a boat sail, depends not only on the properties of the fluid but also on the shape of the object, which defines the spatial domain of the trajectories.

The challenge then lies in efficiently leveraging heterogeneous data sources to discover commonalities across the data, particularly in cases where the data exhibit different dynamics. A key objective is then to develop models that generalize well to new data with similar dynamics, once the model is trained on existing data.

Data is observed irregularly in space and time. Real-world data is rarely retrieved on a predefined grid at specific locations. The phenomenon of interest is rarely fully observed and is subject to discretized measurements. Given the unpredictable availability of the sensors, models must be able to handle the data retrieved from them and be robust to any change that may occur. This then requires models to impose fewer restrictions on the input/output format, in both space and time.

1.3 Contributions of the Thesis

In this thesis, we have addressed some of these challenges through the following problems: (a) developing a sound, formal framework for hybrid modeling (APHYNITY); (b) dealing with the generalization problems (LEADS and CoDA); and (c) mesh-free modeling (DINo).

We summarize below our contributions, which will be discussed successively in Part II:

APHYNITY, Yin et al. (2021b) Due to the generalization issues, purely data-driven approaches are arguably insufficient. We focus on making the numerical and data-driven models work together to forecast complex dynamical phenomena where only partial knowledge of their dynamics is available. In this work, we introduce the APHYNITY framework, which consists of decomposing the dynamics into two components: a physical component formulated from partially known first principles, and a data-driven one that complements the previous physical component by describing the dynamics that cannot be captured by the physical model, no more, no less. APHYNITY enhances the capabilities of both components: the hybrid model predicts well under new conditions, achieves better generalization than either method alone, and helps to identify the appropriate incomplete numerical model instance from a large set of candidates. This not only provides the existence and uniqueness of this decomposition but also ensures interpretability and benefits generalization. Experiments on various phenomena show that APHYNITY can efficiently use incomplete physical models to accurately forecast the evolution of the system and correctly identify relevant physical parameters.

Yuan Yin[*1], Vincent Le Guen[*], Jérémie Donà[*], Emmanuel de Bézenac[*], Ibrahim Ayed[*], Nicolas Thome, and Patrick Gallinari. Augmenting physical models with deep networks for complex dynamics forecasting.

Oral at the 9th International Conference on Learning Representations, ICLR 2021.
Also published in Journal of Statistical Mechanics: Theory and Experiment, JSTAT.
➤ See Chapter 4.

LEADS, Yin et al. (2021a) When modeling dynamical systems from real-world data samples, the distribution of the data often changes according to the environment in which it is captured, and the dynamics of the system itself vary from one environment to another, suggesting that under the same condition, trajectories

[1]The equality of contribution between the authors is indicated by an asterisk (*).

change in different environments. In this case, generalizing across environments thus challenges the conventional ML frameworks. The classical settings suggest either considering data as i.i.d. and learning a single model to cover all situations or learning environment-specific models. Both are sub-optimal: the former disregards the discrepancies between environments leading to biased solutions, while the latter does not exploit their potential commonalities and is prone to scarcity problems. In this work, we propose LEADS, a novel framework that leverages the commonalities and discrepancies among known environments to improve model generalization in the phenomenon. This is achieved with a tailored training formulation aiming at capturing common dynamics within a shared model while additional terms capture environment-specific dynamics. We ground our approach in theory, exhibiting a decrease in sample complexity w.r.t. classical alternatives. We show how theory and practice coincide in the simplified case of linear dynamics. Moreover, we instantiate this framework for neural networks and evaluate it experimentally on representative families of nonlinear dynamics. This new setting can exploit knowledge extracted from environment-dependent data and it improves generalization for both known and novel environments, allowing us to take a first step toward efficient adaptation to new dynamics.

Yuan Yin, Ibrahim Ayed, Emmanuel de Bézenac, Nicolas Baskiotis, Patrick Gallinari. LEADS: Learning dynamical systems that generalize across environments.

The 35th Conference on Neural Information Processing Systems, NeurIPS 2021.
➤ See Chapter 5.

CoDA, Kirchmeyer et al. (2022) Following the previous contribution, we move toward the adaptation to a new dynamical system based on the data of a set of known dynamics. Data-driven approaches to modeling physical systems fail to generalize to unseen systems that share the same general dynamics as the training data but correspond to different physical contexts, e.g., parameters of the dynamics. In this work, we propose a new framework to address this key problem, context-informed dynamics adaptation (CoDA), which takes into account the distributional shift across systems for fast and efficient adaptation to new dynamics. CoDA leverages multiple environments, each associated with a different dynamic, and learns to condition the dynamics model on contextual parameters, specific to each environment. The conditioning is performed via a hyper-network, learned jointly with a context vector from observed data. The proposed formulation constrains

the search hypothesis space for fast adaptation and better generalization across environments with few samples. We theoretically motivate our approach and show state-of-the-art generalization results on a set of nonlinear dynamics, representative of a variety of application domains. We also show, on these systems, that new system parameters can be inferred from context vectors with minimal supervision.

Matthieu Kirchmeyer*, **Yuan Yin***, Jérémie Donà, Nicolas Baskiotis, Alain Rakotomamonjy, and Patrick Gallinari. Generalizing to new physical systems via context-informed dynamics model.

Spotlight at the 39th International Conference on Machine Learning, ICML 2022.
➤ See Chapter 6.

DINo, Yin et al. (2023) We tackle the problem of learning dynamics with irregularly sampled data in space and in time. Effective data-driven PDE forecasting methods often rely on fixed spatial and/or temporal discretization. This raises limitations in real-world applications like weather prediction where flexible extrapolation at arbitrary spatiotemporal locations is required. We address this problem by introducing a new data-driven approach, DINo, that models a PDE's flow with continuous-time dynamics of spatially continuous functions. This is achieved by embedding spatial observations independently of their discretization via Implicit Neural Representations in a small latent space temporally driven by a learned ODE. This separate and flexible treatment of time and space makes DINo the first data-driven model to combine the following advantages. It extrapolates at arbitrary spatial and temporal locations; it can learn from sparse irregular grids or manifolds; at test time, it generalizes to new grids or resolutions. DINo outperforms alternative neural PDE forecasters in a variety of challenging generalization scenarios on representative PDE systems.

Yuan Yin*, Matthieu Kirchmeyer*, Jean-Yves Franceschi*, Alain Rakotomamonjy, and Patrick Gallinari. Continuous PDE dynamics forecasting with implicit neural representations.

Spotlight (notable-top-25%) at the 11th International Conference on Learning Representations, ICLR 2023.
➤ See Chapter 7.

Other Contributions

Yin et al. (2019) **Yuan Yin**, Arthur Pajot, Emmanuel de Bézenac, and Patrick Gallinari. Unsupervised inpainting for occluded sea surface temperature sequences.

The 9th International Workshop on Climate Informatics, CI 2019.

Yin et al. (2020) **Yuan Yin**, Arthur Pajot, Emmanuel de Bézenac, and Patrick Gallinari. Unsupervised spatiotemporal data inpainting.

Preprint.

Migus et al. (2022) Léon Migus, **Yuan Yin**, Jocelyn Ahmed Mazari, and Patrick Gallinari. Multi-scale physical representations for approximating PDE solutions with graph neural operators.

ICLR 2022 Workshop on Geometrical and Topological Representation Learning

1.4 Structure of the Thesis

The thesis is organized as follows. In Chapter 2, we provide an overview of dynamical systems and deep learning, and discuss basic deep learning models for dynamics learning and their limitations, alongside corresponding research topics. In Chapter 3, we dive into each topic and present their related work: physics-data-driven hybrid modeling, multiple dynamics modeling, and continuous modeling. In Part II, we present our main contributions in separate chapters: APHYNITY (Chapter 4), LEADS (Chapter 5), CoDA (Chapter 6), and DINo (Chapter 7). In Part III, we provide a conclusion (Chapter 8) and sketch a perspective on promising research directions for DL dynamics models (Chapter 9).

Chapter 2

Background

We first introduce basic concepts of dynamical systems and the fundamentals of deep learning for dynamical system modeling. We then motivate our research questions, starting from the limitations of current mainstream approaches.

2.1 Dynamical Systems

A dynamical physical phenomenon is typically described with a state that evolves over time, and the description of such an evolution is referred to as a *dynamical system*. To better understand this concept, let us consider an example system.

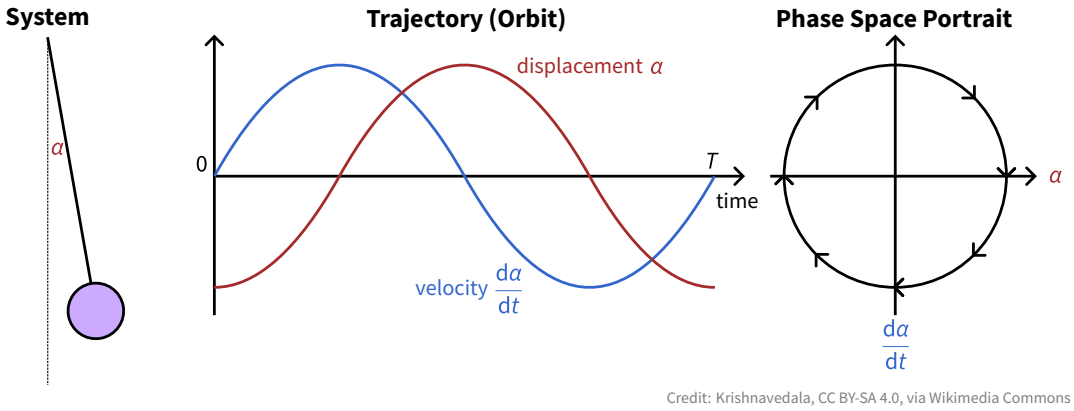


Figure 2.1: Illustration of an ideal pendulum.

An ideal pendulum is shown in Figure 2.1. On the left, we have the pendulum system observed in the physical world: in our example, it is a ball suspended from a fixed point. If we release the ball at the initial position in the figure with zero initial velocity, it will oscillate between two end positions under the influence of gravity. This phenomenon *evolves according to time*. The *state* of the pendulum system, an element in a *phase space* or *state space*, is described by the displacement/position and velocity of the ball at each time stamp. The beginning of the phenomenon is called the *initial state*. It evolves according to an *evolution function*, i.e., the dynamics. During this evolution, we can collect a series of states at each time up to some horizon, namely a *trajectory*, as shown in the middle of Figure 2.1. On the right, this trajectory is represented in the form of a *phase space portrait* (only possible for state vectors of 2 or 3 dimensions), which shows the characteristics of the system's behavior.

This conceptualization allows us to introduce the formal definition of a dynamical system. In this thesis, we are interested in continuous-time dynamical systems.

Definition 2.1 (Continuous-time dynamical system)**.** Following the example above, we say that a dynamical system evolves according to:

- $\mathcal{I} \subset \mathbb{R}, t_0 \in \mathcal{I}$, an interval in the set of real numbers and each $t \in \mathcal{I}$ is a timestamp. The time interval include an initial time t_0, often fixed at $t_0 = 0$ by convention,
- $\mathcal{U}$, a non-empty set called *phase space* or *state space*, and the system state at any

time t is an element of this space $u(t) \in \mathcal{U}$,

- $\Upsilon : \mathcal{I} \times \mathcal{U} \to \mathcal{U}$, a function that describes the *evolution of the state*, s.t. $u(t) = \Upsilon(t, u(0))$. For any t and t', $u(t + t') = \Upsilon(t', \Upsilon(t, u(0)))$.

A *trajectory* is a sequence of states, $u = \{\forall t \in \mathcal{I}, u(t) = \Upsilon(t, u(0))\}$, starting from an initial state $u(0) \in \mathcal{U}$. Typically, the trajectory is viewed as a continuous function on the interval $\mathcal{I}$ and belongs to a space of trajectories denoted as $\Gamma : \mathcal{I} \to \mathcal{U}$.

For the sake of simplicity, we often denote $u(t)$ as u_t when we interpret it as a state at t.

Below we present two families of dynamical systems which are the main objects dealt with in this thesis: ordinary differential equation (ODE), for an initial value problem (IVP), and time-dependent partial differential equation (PDE), for an initial boundary value problem (IBVP).

ODE and IVP. If the state space $\mathcal{U}$ is a subset of the d-dimensional real or complex vector space $\mathbb{R}^d$ or $\mathbb{C}^d$, the flow of the system can be written in the form of an IVP for an ODE:

$$\text{(ODE)} \quad \frac{\mathrm{d}u}{\mathrm{d}t} = f(t, u) \quad \text{on } \mathcal{I}, \qquad \text{(IC)} \quad u(t = 0) = u_0 \in \mathcal{U}, \tag{2.1}$$

where u_0 is an initial condition (IC), f is a differential operator describing the rate of change of the value of the solution function u at every time t. The solution is a function $u : \mathcal{I} \to \mathbb{R}^d$ for the IVP. Throughout the thesis, we interpret f as a *vector field*, which is a mapping from the state space to its tangent bundle, i.e., $f : \mathcal{I} \times \mathcal{U} \to \mathrm{T}\mathcal{U}$ mapping each state $u(t) \in \mathcal{U}$ to the corresponding the rate of change of the state $f(t, u(t)) \in \mathrm{T}\mathcal{U}$. The dynamics represented by f can either change with time, denoted by $\frac{\mathrm{d}u}{\mathrm{d}t} = f(t, u)$, or remain constant, denoted by $\frac{\mathrm{d}u}{\mathrm{d}t} = f(u)$. It is important to note that this should not be confused with the changes in states. The latter case is referred to as an *autonomous system* and is the main case considered in this thesis.

Example 2.1 (Ideal pendulum). For example, in the IVP of an ideal pendulum, the state is a vector with two values: angular position w.r.t. a rest position α and velocity $\frac{\mathrm{d}\alpha}{\mathrm{d}t}$, s.t. $\left(\alpha(t), \frac{\mathrm{d}\alpha}{\mathrm{d}t}(t)\right) \in \mathbb{R}^2$. The dynamics write as:

$$\frac{\mathrm{d}^2\alpha}{\mathrm{d}t^2} = -\omega^2 \sin\alpha, \qquad \alpha(t = 0) = \alpha_0, \qquad \frac{\mathrm{d}\alpha}{\mathrm{d}t}(t = 0) = \dot{\alpha}_0 \tag{2.2}$$

It describes the flow with the initial angle of the pendulum as $\alpha(t = 0)$ when released

with the initial velocity $\frac{\mathrm{d}\alpha}{\mathrm{d}t}(t=0)$. The flow can be written as:

$$\begin{bmatrix} \alpha(t) \\ \frac{\mathrm{d}\alpha}{\mathrm{d}t}(t) \end{bmatrix} = \Psi\left(t, \begin{bmatrix} \alpha_0 \\ \dot{\alpha}_0 \end{bmatrix}\right) = \begin{bmatrix} \alpha_0 \\ \dot{\alpha}_0 \end{bmatrix} + \int_{s=0}^{s=t} \begin{bmatrix} \frac{\mathrm{d}\alpha}{\mathrm{d}t}(s) \\ -\omega^2 \alpha(s) \end{bmatrix} \mathrm{d}s \tag{2.3}$$

If $\alpha_0, \dot{\alpha}_0$ are small, the equation is linearized and the analytical solution of the flow is:

$$\begin{aligned} \alpha(t) &= \alpha_0 \cos(\omega t) + \frac{\dot{\alpha}_0}{\omega} \sin(\omega t) \\ \frac{\mathrm{d}\alpha}{\mathrm{d}t}(t) &= -\omega \alpha_0 \sin(\omega t) + \dot{\alpha}_0 \cos(\omega t) \end{aligned} \tag{2.4}$$

The system is considered autonomous because the operations in the equation remain constant over time, regardless of any changes in the system's state.

If we add a driving forcing term to the system, for example,

$$\frac{\mathrm{d}^2\alpha}{\mathrm{d}t^2} = -\omega^2 \sin\alpha + \cos(\omega_{\text{ext}} t) \tag{2.5}$$

The system then becomes a driven pendulum and is no longer autonomous.

Time-dependent PDE and IBVP. If $\mathcal{U}$ is a subset of a function space, defined on a *spatial domain* Ω (which is commonly a closed subset of $\mathbb{R}^p$) and outputs a vector in $\mathbb{R}^d$ or $\mathbb{C}^d$, then the flow of the system writes as an IBVP for a time-dependent PDE:

$$\text{(PDE)} \quad \frac{\partial u}{\partial t} = f(t,u), \text{ on } \mathcal{I} \times \Omega \qquad \text{(IC)} \quad u(t=0,\cdot) = u_0 \in \mathcal{U} \tag{2.6}$$

Similar to the IVP, the IBVP requires an initial condition u_0, which is now a function defined over the spatial domain Ω. The PDE governs the evolution of the solution at every position $x \in \Omega$. The solution $u\colon \mathcal{I} \times \Omega \to \mathbb{R}^d$ is a function defined in both space and time.

Example 2.2 (Wave equation)**.** For example, the scalar wave equation in $\Omega \subset \mathbb{R}^2$ is defined as follows:

$$\frac{\partial^2 u}{\partial t^2} = c^2 \left(\frac{\partial^2 u}{\partial x_1^2} + \frac{\partial^2 u}{\partial x_2^2} \right) = c^2 \Delta u \tag{2.7}$$

where $u\colon \mathcal{I} \times \Omega \to \mathbb{R}$ is a scalar field which is a wave solution for scalar physical quantities, such as pressure in the air when a sound wave passes through, or the

displacement of a string or a surface from its resting position. c^2 is the speed of propagation of the wave. The Laplace operator $\frac{\partial^2}{\partial x_1^2} + \frac{\partial^2}{\partial x_2^2}$ or in its vector notation Δ, describes the diffusion behavior of the system.

In addition to these components, boundary conditions (BCs) should be also considered:

$$\text{(BC)} \quad B(u) = 0 \text{ on } \mathcal{I} \times \partial\Omega \tag{2.8}$$

They add constraints on the spatial boundary of the solutions. We describe here the most common BCs considered in dynamical systems.

Example 2.3 (Common types of BC)**.**

- *Dirichlet BC* specifies the value at the boundary with a function f_b

$$u_t(x) = f_b(x) \tag{2.9}$$

  For example, for the 1D wave equations on a string, the Dirichlet BC specifies the positions of the ends of the string.

- *Neumann BC* specifies the value of the normal derivative, which is the directional derivative taken in the direction orthogonal to $\partial\Omega$, of the solution function:

$$\frac{\partial u_t}{\partial n}(x) = f_b(x) \tag{2.10}$$

  where n denotes the normal to the boundary. For example, the Neumann BC describes the forces on the string at the endpoints for the 1D wave equations.

- *Periodic BC* supposes that the phenomenon lives in a domain repeated infinitely in the entire space. For 2D domain $\mathbb{R}^2$ with period (r_1, r_2) in two directions,

$$u_t(x_1, x_2) = u_t(x_1 - r_1, x_2) = u_t(x_1, x_2 - r_2) \tag{2.11}$$

  It projects the phenomenon on a torus $\mathbb{T}^2$.

Boundary conditions can have a profound impact on the behavior of a system, particularly when dealing with complex interactions. In many cases, the subdomain around the boundary requires dense discretization to accurately capture the nuances of the dynamics, which can be influenced by various factors such as viscosity. For instance, in the context of viscous fluid dynamics, fluid flow near a boundary is significantly slower due to the viscosity, leading to zero velocity at the boundary. This phenomenon is known as the

boundary layer, which is often characterized by the emergence of turbulence.

2.2 Basic Setting of Dynamics Modeling with Data

We present the basic setting of data-driven dynamics modeling. To show the data with the intuitions on the discretization, we consider the trajectory space Γ as in Definition 2.1 that contains trajectories continuous in time and space. We generate the training data by sampling points on these continuous trajectories. If they are only temporal, i.e., each state $u_t \in \mathbb{R}^d$ is a vector, we sample the reference solution on a discrete set of time $\mathcal{T} \subset \mathcal{I}$. The resulting dataset $\mathcal{D}$ is organized as follows,

$$\mathcal{D} = \left\{ u^{(i)}|_{\mathcal{T}} \,\middle|\, u_0^{(i)} \sim \rho_0(\mathcal{U}), u^{(i)} \in \Gamma \right\}_{i \in [\![1,N]\!]} \tag{2.12}$$

which is a set of discretized trajectories on $\mathcal{T}$ with the initial condition provided. We sample the trajectories by sampling initial conditions according to a distribution ρ_0.

For spatiotemporal data, the state $u_t \colon \Omega \to \mathbb{R}^d$ is a spatial function, so the trajectory is sampled both on a discrete set of timestamps $\mathcal{T} \subset \mathcal{I}$ and spatial coordinates $\mathcal{X} \subset \Omega$.

$$\mathcal{D} = \left\{ u^{(i)}|_{\mathcal{T}\times\mathcal{X}} \,\middle|\, u_0^{(i)} \sim \rho_0(\mathcal{U}), u^{(i)} \in \Gamma \right\}_{i \in [\![1,N]\!]} \tag{2.13}$$

The aim is to map the initial condition u_0 to the corresponding trajectory $u|_{t\geq 0}$, which may be continuous. The ideal learned dynamics model takes the form of:

$$\begin{aligned} \text{DynamicsModel}_\theta \colon \mathcal{U} &\to (\mathcal{I} \to \mathcal{U}) \\ u_0 &\mapsto (t \mapsto u_t) \end{aligned} \tag{2.14}$$

The modeling assumption described above will be a key factor in the generalization problems investigated in this thesis. Specifically, we will examine two aspects of the model's performance: robustness to changes in initial conditions and adaptation of the mapping toward the output trajectory.

In practice, the predicted state space $\mathcal{U}$ is often discretized. The output can be a sequence of states evaluated on a discrete timestamp grid. Additionally, it is important to note that the data used for learning and evaluation may have different time and/or space grids, which further complicates the modeling process. Nonetheless, we will carefully analyze these issues and propose effective solutions throughout this thesis.

2.3 Deep Learning for Dynamics Modeling

This section outlines the current approach to modeling dynamics using deep learning (DL). We first introduce the fundamental deep learning algorithms in Section 2.3.1. Subsequently, we describe the neural spatial and temporal architectures that are commonly used in deep dynamics models in Sections 2.3.2 and 2.3.3, respectively.

2.3.1 Deep Learning in a Nutshell

We provide here a brief overview of the essentials of deep learning from a supervised learning perspective.

Supervised learning. As we presented in Chapter 1, we are interested in acquiring the knowledge given by an objective through some learning process, which is *supervised learning*. Here, we focus on learning with parametrized models. During this learning process, a.k.a. training, the model is adjusted based on the information given by a finite set of data, consisting of pairs of input $z \in \mathcal{Z}$ and output $y \in \mathcal{Y}$, known as a training set $\mathcal{D}_{\text{tr}} = \{(z^{(i)}, y^{(i)})\}_{i \in [\![1, N_{\text{tr}}]\!]}$. A parametrized model $f : \mathcal{Z} \times \Theta \to \mathcal{Y}$ provides an output $\tilde{y} \in \mathcal{Y}$ in the output space of the training set. For convenience, we write $\tilde{y} = f(x, \theta) = f_\theta(x)$. This allows us to compare the model output with the desired output given by the *training set* $\mathcal{D}_{\text{tr}}$. The model can be evaluated by a sample-wise *loss function* $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$ on this training set.

Suppose that the model is a single function f_θ parametrized by θ that estimates $\tilde{y} = f_\theta(z^{(i)})$ and that our primary objective is to fit the data $\mathcal{D}_{\text{tr}}$. In this case, we can construct an aggregated empirical loss function $\mathcal{L}$ for the entire dataset distribution:

$$\mathcal{L}(f_\theta, \mathcal{D}_{\text{tr}}) = \sum_{i=1}^{N_{\text{tr}}} \ell(f_\theta(z^{(i)}), y^{(i)}) \tag{2.15}$$

where ℓ is a loss function measuring the discrepancy between y and $\tilde{y}$. For instance, mean square error (MSE) is defined as $\ell_{\text{MSE}}(f_\theta(z^{(i)}), y^{(i)}) = \|f_\theta(z^{(i)}) - y^{(i)}\|_2^2$. Other alternative loss functions can be used under different modeling hypotheses.

For a spatiotemporal dynamics model defined in Eq. (2.14), the total loss takes the form of

$$\mathcal{L}(\text{DynamicsModel}_\theta, \mathcal{D}_{\text{tr}}) = \sum_{i=1}^{N_{\text{tr}}} \sum_{t\in\mathcal{T}} \sum_{x\in\mathcal{X}} \ell(\text{DynamicsModel}_\theta(u_0^{(i)})(t,x), u(t,x)). \quad (2.16)$$

with $\text{DynamicsModel}_\theta \colon (\mathcal{X} \to \mathbb{R}^d) \to (\mathcal{T}\times\mathcal{X} \to \mathbb{R}^d)$. Once the model has been evaluated on the training data, we can define an optimization objective to search for a candidate model within an eligible function space of $f_\theta \in \mathcal{F}$ that meets this objective. In the case of a parametrized model, we search for the best set of parameters θ^* in the parameter space Θ:

$$\theta^* = \arg\min_{\theta\in\Theta} \mathcal{L}(f_\theta, \mathcal{D}_{\text{tr}}) \quad (2.17)$$

After obtaining the optimal parameters, we evaluate the quality of the learned model by testing it on unseen data, which is typically organized as a separate *test set* $\mathcal{D}_{\text{ts}}$.

However, it is worth noting that the *best* model according to the optimization problem above is not necessarily unique in deep learning and may perform poorly on the test set. This is known as the generalization problem, which can be attributed to several factors, such as an insufficient quantity of data, a.k.a. the regime of data, an imbalanced distribution of data, an over-parametrized model, or inadequate modeling assumptions.

Feedforward neural networks. Generally speaking, feedforward neural networks are differentiable functions, composed of an alternating cascade between linear transformations and nonlinear activation functions from input to output. The repetitive structure in the network is called a *layer*. The following example is the basic structure of a sequential feedforward neural network:

$$y = f(z) = \underbrace{\sigma^{(L)} \circ T^{(L)}}_{\text{layer } L} \circ \cdots \circ \sigma^{(1)} \circ T^{(1)}(z) \quad (2.18)$$

In this L-layer network, each layer l is composed of a linear transformation $T^{(l)} : \mathcal{Z}^{(l-1)} \to \mathcal{Z}^{(l)}$, and a local element-wise activation function $\sigma^{(l)} : \mathcal{Z}^{(l)} \to \mathcal{Z}^{(l)}$. The input and output spaces are denoted by $\mathcal{Z} = \mathcal{Z}^{(0)}$ and $\mathcal{Y} = \mathcal{Z}^{(L)}$ respectively. Linear maps $T^{(l)}$ are parameterized, while activation functions $\sigma^{(l)}$ are generally nonlinear and not parameterized (see some exceptions in Example 2.4). More complex structures have been derived from the sequential feedforward neural networks, such as those used in ResNet and U-Nets (cf. Examples 2.5 and 2.6). When evaluating the function f_θ at the input z, a

forward pass is performed, resulting in a prediction at that point.

Linear maps are chosen according to the format of the input, and several examples are presented in Section 2.3.2. Activation functions, on the other hand, are typically considered as hyperparameters. Here are some commonly considered activation functions:

Example 2.4 (Common activation functions)**.** For simplicity, z is a real number.

$$\sigma(z) = z \qquad \text{linear} \tag{2.19}$$

$$\sigma(z) = \frac{1}{1+e^{-z}} \qquad \text{logistic, sigmoid} \tag{2.20}$$

$$\sigma(z) = \tanh(z) \qquad \text{hyperbolic tangent} \tag{2.21}$$

$$\sigma(z) = \max(0, z) \qquad \text{rectified linear unit (ReLU)} \tag{2.22}$$

$$\sigma(z) = \frac{z}{1+e^{-\beta z}} \qquad \text{Swish } (\beta \text{ learnable}) \tag{2.23}$$

If z has multiple values, i.e., vector, matrix, tensor, the activation function is applied element-wise.

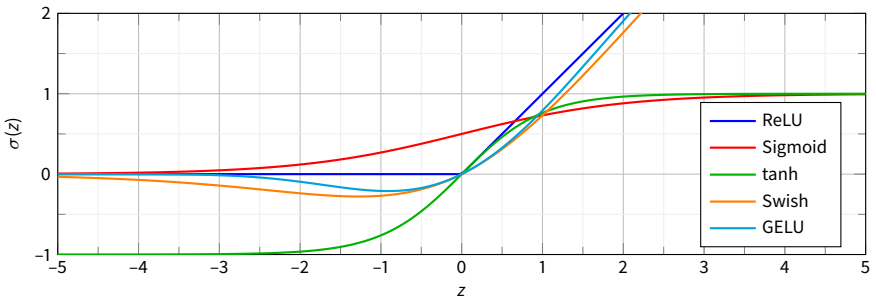


Figure 2.2: Activation functions.

Common activation functions used in hidden layers ($l < L$) include sigmoid-like functions, or ReLU and its variants (e.g., GELU, Hendrycks and Gimpel, 2016, Swish, Ramachandran et al., 2017). For the final layer ($l = L$), activation functions are chosen based on the desired output value range.

Optimization. Given the nature of the model, the landscape of the loss w.r.t. parameter is very high dimensional and highly non-convex (or non-concave for the maximization problem), making it challenging to find a global minimum. Therefore, one may use local information to find a local minimum that performs reasonably well instead. Starting from

an initialization θ_0, the parameter is iteratively updated using gradient descent (GD) to reach the nearest local minimum:

$$\theta_{t+\delta t} = \theta_t - \eta \nabla_\theta \mathcal{L}(f_\theta, \mathcal{D}_{\text{tr}}) \quad \text{or} \quad \frac{\mathrm{d}\theta}{\mathrm{d}t} = -\nabla_\theta \mathcal{L}(f_\theta, \mathcal{D}_{\text{tr}}) \tag{2.24}$$

When the dataset is much too large, stochastic gradient descent (SGD) can be used instead by randomly sampling a subset $\mathcal{D}_t \subset \mathcal{D}_{\text{tr}}$, a.k.a. a batch, from the complete dataset. Methods such as Adam (Kingma and Ba, 2015) propose per parameter adaptative learning rate using historic gradient and momentum. To compute the gradient, *backpropagation*, a.k.a. the reverse mode of automatic differentiation (AUTODIFF), is used, which efficiently calculates the derivatives w.r.t. an evaluation point by constructing a computational graph.

2.3.2 Learning Mappings in Dynamics Models with Neural Networks

In dynamics models, it is necessary to learn a nonlinear mapping between two vector spaces, such as a transition between states $u_t \mapsto u_{t+\delta t}$. In this context, we briefly present the neural network architectures often used in DL dynamics modeling. Specially, we focus on learning functions from discrete spatiotemporal data sampled from continuous signals, as our data is expected to be in this form.

Recall that in a feedforward neural network, we learn a mapping $f_\theta \colon z \in \mathcal{Z} \mapsto y \in \mathcal{Y}$. If the input signal is a vector $z \in \mathbb{R}^{d_z}$ without any specific structure, we can apply matrix linear maps to transform the input and formulate the network as follows:

Multi-layer perceptron (MLP). The neural network uses matrices to perform linear mapping. The transformation at layer l writes as follows:

$$z^{(l)} = \sigma \circ T^{(l)}(z^{(l-1)}) = \sigma\left(W^{(l)} z^{(l-1)} + b^{(l)}\right) \tag{2.25}$$

where $z^{(l-1)} \in \mathbb{R}^{d_{z^{(l-1)}}}$ is hidden layer feature coming from the previous layer, $W^{(l)} \in \mathbb{R}^{d_{z^{(l)}} \times d_{z^{(l-1)}}}$ is the matrix linear transformation between $l-1^{\text{th}}$ and l^{th} layers, $b^{(l)}$ is the bias. See Figure 2.3 for the illustration of an MLP.

Although MLPs can handle any high-dimensional input vector in theory, they often fail to perform well on signals like digital images. This is because MLPs lack prior knowledge about the structure of the input signal and do not account for invariances under common

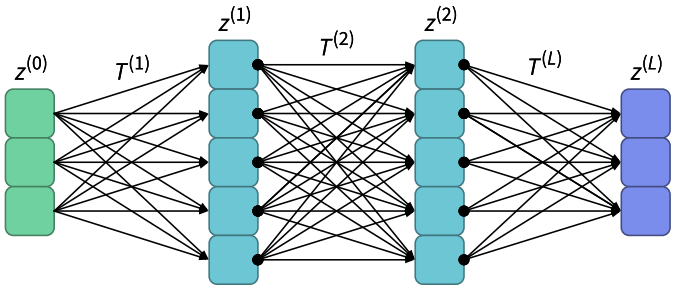


Figure 2.3: Illustration of a 3-layer MLP.

transformations like translation.

* * *

To address these limitations, researchers have developed regularized versions of this general architecture that make assumptions about the form or structure of the input signal. For example, images can be represented as collections of local vectors sampled at different spatial points. In this case, the input vector image can be viewed as a discretized version of the continuous vector-valued function $z\colon \Omega \to \mathbb{R}^d$ (d is often called the number of channels), where it is only sampled at positions $x \in \mathcal{X} \subset \Omega$. We will explore some of the architecture developed for this type of data.

Convolutional neural network (ConvNet). ConvNets are MLPs regularized for multi-channel signals regularly sampled on a rectangular domain Ω, e.g., digital color images. Here, the sampled position $\mathcal{X}$ is a regular grid with sample interval δx_i for each dimension i. The size $\#\mathcal{X}$ is the sampling resolution.

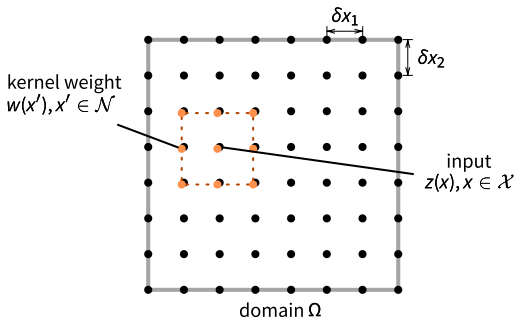


Figure 2.4: Illustration of the conventional convolution in ConvNet.

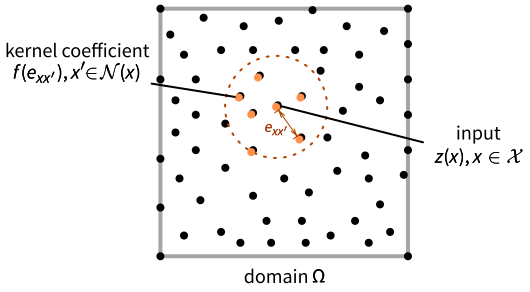


Figure 2.5: Illustration of convolution with MPNN.

After evaluating z on the grid $\mathcal{X}$, the input $z|_{\mathcal{X}} \in \mathbb{R}^{\#\mathcal{X}\times d}$ is under the form of a tensor, e.g., for a 64-by-64 RGB image $z|_{\mathcal{X}} \in \mathbb{R}^{64\times 64\times 3}$. Hidden layer features are also in the same form and potentially discretized on a specific grid $\mathcal{X}^{(l)}$ of the l^{th} layer, i.e., $z_j^{(l)}|_{\mathcal{X}^{(l)}}$. One common method of transitioning between grids involves the use of downsampling (pooling) or upsampling techniques.

To perform the linear transformation in a ConvNet, the discretized input feature $z_j^{(l)}|_{\mathcal{X}^{(l)}}$ is convolved with a *kernel*, i.e., a local parameterized function $w_{jk}^{(l)} : \Omega \to \mathbb{R}$ from the input channel j to the output channel k. In ConvNets, the kernel has non-zero values only in a closed subset, often a rectangular region, of the input domain. This kernel is discretized on a local grid $\mathcal{N}^{(l)}$ with the same sample interval δx_i as $\mathcal{X}^{(l)}$, and evaluated on $\mathcal{N}^{(l)}$ to obtain the kernel matrix $w_{jk}^{(l)}|_{\mathcal{N}^{(l)}} \in \mathbb{R}^{\#\mathcal{N}}$. For example, a 3-by-3 two-dimensional kernel results in $w_{jk}^{(l)}|_{\mathcal{N}^{(l)}} \in \mathbb{R}^{3\times 3}$. See Figure 2.4 for an illustration.

Then at each point $x \in \mathcal{X}^{(l)}$, the transformation from channel j to k at the l^{th} layer writes as follows:

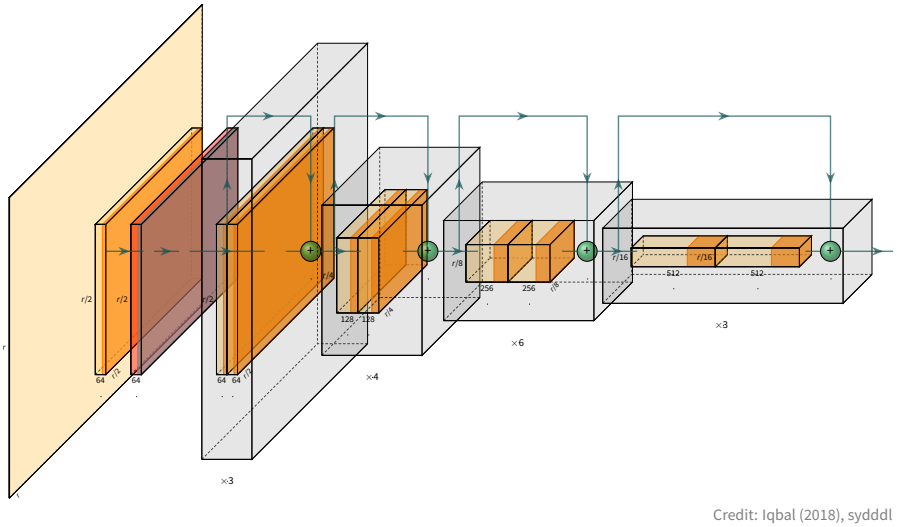
$$z_k^{(l)}(x) = \sigma \circ (T_{jk}^{(l)}(z_j^{(l-1)}|_{\mathcal{X}^{(l-1)}})(x) = \sigma\left(\sum_{x'\in\mathcal{N}^{(l)}} z_j^{(l-1)}(x - x')w_{jk}^{(l)}(x') + b_{jk}^{(l)}\right) \tag{2.26}$$

which is a weighted mean of $z_j^{(l-1)}|_{\mathcal{X}^{(l-1)}}$ in the neighborhood defined by $\mathcal{N}^{(l)}$. Because the kernel is always evaluated on the same grid template, the kernel matrix $w|_{\mathcal{N}}$ contains only a limited number of weights and does not change with the spatial discretization of the input.

The resolution of the convolved signal is normally very high w.r.t. the number of points in the kernel, then the kernel operates in a very local zone, e.g., the conventional kernel size 3-by-3 compared to the 1024-by-1024 images. If one needs to perform transformations at larger scales, the feature $z_j^{(l)}$ should be subsampled on a coarser grid, e.g., by average pooling or maximum pooling.

Most modern DL architectures for images are constructed upon this convolutional backbone. We here present residual neural network (ResNet; He et al., 2016) and U-Net (Ronneberger et al., 2015), two remarkable examples often used for dynamics prediction.

> **Example 2.5** (ResNet; He et al., 2016)**.** Originally motivated to mitigate the problem of vanishing gradient encountered when training deeper convolutional neural networks,



Credit: Iqbal (2018), sydddl

Figure 2.6: ResNet architecture. The gray boxes delineate the convolution blocks, which are typically sequential ConvNets. For each block, a skip connection is added between the input and the output, indicated by a green arrow. The gray boxes are repeated sequentially by the number of times indicated below.

ResNet uses skip connections every few layers to allow for direct gradient flow, enabling successful training of very deep networks. Figure 2.6 illustrates a typical ResNet for an image classification problem.

ResNet's convolution layers are grouped into residual blocks, as illustrated in Figure 2.6 with the gray boxes. A residual block consists of multiple convolutional layers, forming a sub-network. The input of the block is added to its output as follows:

$$z^{(l+\Delta l)} = z^{(l)} + f_\theta^{(l)}(z^{(l)}) \tag{2.27}$$

where $f_\theta^{(l)}$ is the sequential sub-network the residual block, Δl is the number of layers in the sub-network. The skip connection bears a strong resemblance to the forward Euler numerical scheme and has been subsequently associated with ODEs, as demonstrated in Example 2.12.

Example 2.6 (U-Net; Ronneberger et al., 2015)**.** Initially designed for medical image segmentation, the U-Net architecture (shown in Figure 2.7) has evolved into a widely used multi-scale approach for image-to-image mapping.

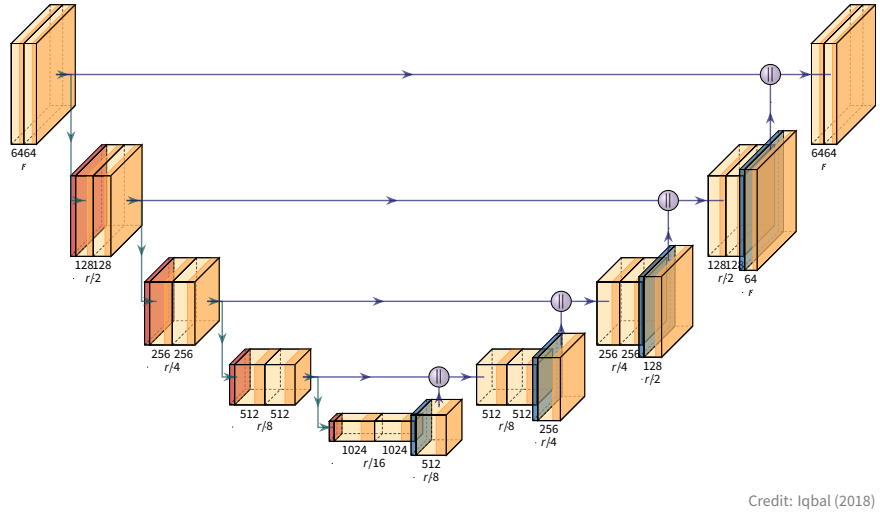


Figure 2.7: U-Net architecture. Each block represents the feature maps after transformations. "||" signifies the concatenation of feature maps along the channel axis.

> The network follows an encoder-decoder structure, where the encoder is composed of a cascade of upscaling convolutional blocks that reduce the resolution from r^2 to $(r/16)^2$. This allows the network to extract feature maps at different predefined scales. The decoder, on the other hand, performs the opposite operation, upsampling the feature maps and reusing the features at the same scale through residual connections. This approach enables the network to effectively leverage multi-scale information and generate images with fine details.

It is worth noting the significant architectural bias inherent in ConvNets. The resolution of the input and the convolutional layers in a ConvNet are structurally linked due to the kernel weights possessing the same resolution (δx_i) as the input. Consequently, modifying the input resolution can have an impact on the performance and accuracy of the network.

Graph neural network (GNN). GNNs are a family of architectures that operate on inputs represented by a graph. They are closely related to the geometry of the input since the graph is one of its representations (Bronstein et al., 2021). In this context, we consider the input to be a Euclidean graph with geometrical nodes that represent points in a domain Ω, and edges that have lengths equal to the Euclidean distance between the corresponding points. This type of graph is often used to represent unstructured mesh grids used in solving PDEs.

In general, the graph nodes represent a finite set of positions $\mathcal{X}$ given by the input grid. This neighborhood relation can be established using a predefined connectivity matrix, also known as an *adjacency matrix*. Alternatively, the neighborhood relation can be defined based on other criteria. For example, in the case of distance-based neighborhoods, the neighbors of a node x can be identified by including all nodes within a (hyper)sphere of radius r centered at x. This yields the set of neighbors $\mathcal{N}_r(x) = \{x' \in \mathcal{X} \mid d(x, x') \leq r\}$, where d is a distance function.

Here, we present the concept of GNNs, starting with the graph convolutional network (GCN). Given a grid $\mathcal{X}$ and node connectivity defined for each $x \in \mathcal{X}$ by $\mathcal{N}(x)$, a GCN computes at each layer a weighted sum of features in the neighborhood $\mathcal{N}(x)$ to update the feature at x, similar to a fixed convolution kernel in a ConvNet.

Example 2.7 (GCN; Kipf and Welling, 2017)**.** The GCN was introduced to perform node classification in a graph, where labels are available only for a small subset of nodes. The l^{th} layer writes as:

$$z^{(l)}(x) = \sigma\left(W^{(l)}\left(\sum_{x' \in \mathcal{N}(x)} c_{xx'} z^{(l-1)}(x')\right)\right) \tag{2.28}$$

where $c_{xx'}$ is a specific weight determined by the graph's adjacency matrix, which is not learnable and depends on the connectivity of the graph, and $W^{(l)}$ is a trainable weight matrix that transforms the aggregated information from the neighborhood. σ is the activation function resulting in the output feature map $z^{(l)}(x)$.

We observe that the convolutional coefficients in the GCN only consider connectivity. In dynamics modeling, it is also important to incorporate the distance between two nodes to differentiate interactions of various ranges. In this case, the architectures should also be able to incorporate edge properties between graph nodes. One of the most commonly used architectures for this purpose is the message passing neural network (MPNN).

Example 2.8 (MPNN)**.** We present the general framework of an MPNN. At the l^{th} layer, the transformation of node features at the position x in a neighborhood $\mathcal{N}$ is:

$$z^{(l)}(x) = g^{(l)}\left(z^{(l)}(x), \sum_{x' \in \mathcal{N}(x)} f^{(l)}\left(z^{(l-1)}(x), z^{(l-1)}(x'), e_{xx'}\right)\right) \tag{2.29}$$

where $f^{(l)}$ is a neural network that transforms node features at previous layer $z^{(l-1)}(x), z^{(l-1)}(x')$ and edge features $e_{xx'}$ into a message from the node at position x' to x. $g^{(l)}$ transforms the node features by conditioning the network with aggregated messages from all neighbors.

In the original MPNN (Gilmer et al., 2017), Eq. (2.29) is implemented as follows:

$$z^{(l)}(x) = \sigma\left(W^{(l)} z^{(l-1)}(x) + \sum_{x' \in \mathcal{N}(x)} z^{(l-1)}(x') \cdot \mathrm{MLP}_\theta^{(l)}(e_{xx'})\right) \tag{2.30}$$

The edge feature $e_{xx'}$ can contain any input involving the relation between two nodes, e.g., the difference between two node positions $x - x'$, or between two input node features $u(x) - u(x')$, or values conditioning the relation such as priorly known equation parameters. This more flexible modeling is adopted by most recent graph-based neural PDE dynamics models, such as (Li et al., 2020; Iakovlev et al., 2021; Brandstetter et al., 2022).

However, incorporating these edge features introduces a sampling bias into the convolution kernels. If the network is trained on a dense sampling with a small neighborhood, it may not perform well when tested on a sparse sampling with a large neighborhood.

Operator learning. As previously mentioned, when modeling PDE dynamics, the learned mapping between input and output states is typically discretized on a spatial grid $\mathcal{X}$. The model's parameterization is inherently biased toward — even tied to — the structure of the input grid. However, in many applications, it is desirable to generalize the model beyond the grid and have free-form input and output. This objective is akin to learning function-to-function mapping, which is what an operator does analytically. In this idealized model, it is assumed that both the input and the output of the mapping are continuous objects. This assumption has led to recent neural methods that partially fulfill these objectives.

Here we present two representative families of methods that address the aforementioned objectives. The first approach emphasizes achieving a free-form output.

Example 2.9 (DeepONet; Lu et al., 2021)**.** DeepONet is built upon an architecture that has been proven to be a universal operator approximator (Chen and Chen, 1995). It takes an input function $z\colon \Omega \to \mathbb{R}$ observed on a fixed irregular grid $\mathcal{X}$, where each

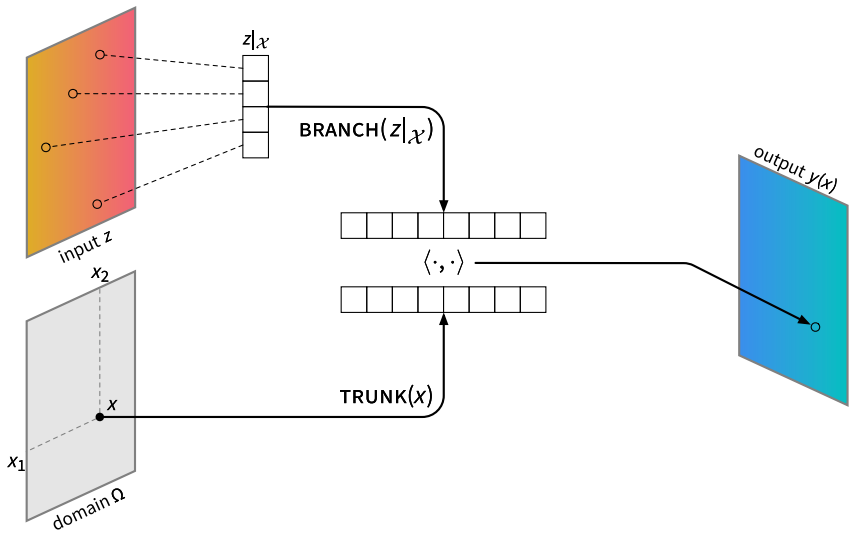


Figure 2.8: Illustration of DeepONet.

$x \in \mathcal{X}$ is a sensor, and maps it to an output function $y\colon \Omega \to \mathbb{R}$. The output function f_θ is a continuous function that can be evaluated at any position $x \in \Omega$.

$$y(x) = f_\theta(x, z|_{\mathcal{X}}) \tag{2.31}$$

f_θ is constructed by the following two subnets:

- *Branch net*: $v^z_{\text{BRANCH}} = \text{BRANCH}(z|_{\mathcal{X}})$, a neural network that transforms the input $z|_{\mathcal{X}}$ into a vector $v_{\text{BRANCH}} \in \mathbb{R}^q$;
- *Trunk net*: $v^x_{\text{TRUNK}} = \text{TRUNK}(x)$, a neural network that takes coordinates $x \in \Omega$ as input and outputs a vector $v_{\text{TRUNK}} \in \mathbb{R}^q$ at that position, aligned with the dimension of the branch net output.

The branch and the trunk net are usually implemented with MLPs. The branch net encodes the input function into a q-dimensional vector, the trunk net provides the evaluation of q basis functions at a point. The output is obtained by a dot product between the two vectors above:

$$y(x) = \langle \text{TRUNK}(x), \text{BRANCH}(z|_{\mathcal{X}}) \rangle = \langle v^x_{\text{TRUNK}}, v^z_{\text{BRANCH}} \rangle \tag{2.32}$$

which means that the output function is a linear combination of the basis functions with

coefficients conditioned by $z|_{\mathcal{X}}$.

As a result, the output function can be evaluated at any point in the domain. The input encoder is bound to the input grid if the encoder is a simple MLP.

The second family, called neural operator (NO; Kovachki et al., 2021), proposes alternatives to conventional convolution kernels from a functional viewpoint. We present here Fourier neural operator (FNO; Li et al., 2021b), an effective and also the most influential instance of the family. It addresses the need for large kernels in certain phenomena, such as incompressible Navier-Stokes, where the time derivative at each position depends on information across the entire domain.

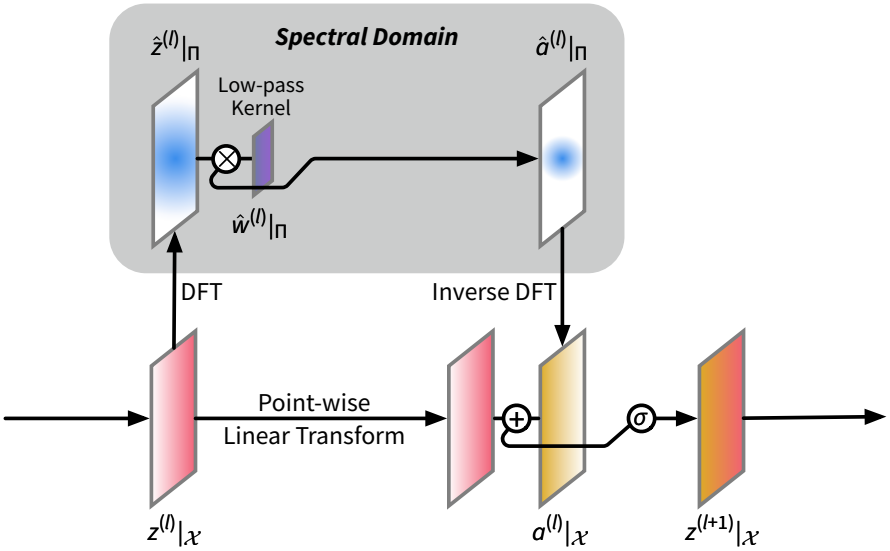


Figure 2.9: Illustration of an FNO layer.

Example 2.10 (FNO; Li et al., 2021b)**.** The FNO replaces the conventional convolution kernels of a classical ResNet with ones discretized in Fourier space. The signals to be convolved are transformed with discrete Fourier transform (DFT), allowing for convolution to be performed directly in Fourier space. In FNO, the spatial grids $\mathcal{X}$ can be aligned consistently during inference and can be adjusted to different resolutions as needed. The spectral space kernel is defined given a finite set of frequencies $\omega \in \Pi \subset \mathbb{R}^p$ for the j^{th} input channel to the k^{th} output channel. An FNO layer writes as:

$$\hat{z}_j^{(l-1)}(\omega) = \sum_{x \in \mathcal{X}} z_j^{(l-1)}(x) e^{-i\langle \omega, x \rangle} \ \forall \omega \in \Pi \qquad \text{(DFT)} \tag{2.33}$$

$$\hat{a}_k^{(l)}(\omega) = \hat{w}_{jk}^{(l)}(\omega) \cdot \hat{z}_j^{(l-1)}(\omega) \ \forall \omega \in \Pi_{|\omega| \leq c} \quad \text{(Conv. with low-pass filtered kernel)} \tag{2.34}$$

$$a_k^{(l)}(x) = \frac{1}{\#\Pi} \sum_{\omega \in \Pi} \hat{a}_k^{(l)}(\omega) e^{i\langle \omega, x \rangle} \ \forall x \in \mathcal{X} \quad \text{(Inverse DFT)} \tag{2.35}$$

$$z^{(l)}(x) = \sigma(W^{(l)} z^{(l-1)}(x) + a^{(l)}(x)) \quad \text{(Skip connection)} \tag{2.36}$$

where $\hat{z}_j$ is the DFT-transformed input z_j, $\hat{w}_{jk}$ is the kernel from the input channel j to the output channel k in spectral space, filtered with a maximum frequency c. The pre-activation feature $\hat{a}_k^{(l)}$ is brought back to the spatial domain $a_k^{(l)}$ with the inverse DFT. To perform the skip connection, the layer input is point-wise linearly transformed as in Eq. (2.36). The layer is illustrated in Figure 2.9. The low-pass filtered kernel allows learning large kernels in space. In practice, to ensure computational efficiency, the DFT is implemented using the fast Fourier transform (FFT) with a regular grid in both the physical domain and the spectral domain.

Variants have been proposed around the FNO (cf. Kovachki et al., 2021 for a thorough review).

Although these methods assume continuous modeling, they may still necessitate inflexible spatial discretization in certain regions of the model. DeepONet, for instance, only accepts input on a fixed grid, rendering it unusable when the input is sampled at other locations. Likewise, while FNO employs the FFT to circumvent the quadratic time complexity of the brute force DFT, this choice restricts the input format to regular grids, constraining the model's flexibility. Additionally, the skip connections in NOs link the input grid to the output grid, limiting the freedom to evaluate the output.

2.3.3 Neural Dynamics Models

We introduce two major families of neural dynamics models: discrete-time models, such as classical neural autoregressive models, and continuous-time models, which include neural differential equations. These models originate from statistical machine learning and numerical analysis, respectively, and have gained significant attention in recent years. We denote the time by the following rule: t is an arbitrary time; τ is an arbitrary time interval; δt is a fixed, predetermined time interval (as a model hyperparameter).

Discrete-time models. Dynamical systems have temporal dependencies, meaning that the current state depends on the previous state(s), propagating information in time. This brings to mind the similarity to autoregressive models, which estimate a state according to its history, which is a factorization of the entire temporal process:

$$p(u) = \prod_{t \in \mathcal{T}} p(u_t \mid u|_{\mathcal{T}_{<t}}) \tag{2.37}$$

If the dynamical system is autonomous, the autoregressive model does not need to account for direct long-term dependencies. In this case, the current state only depends on the previous state:

$$p(u_t \mid u_{t-\delta t}) \tag{2.38}$$

This can be materialized with the following iteration implemented with a neural network mapping $u_t = f_\theta(u_{t-\delta t})$. Recent approaches to PDEs implement this with MLP, ConvNet, or GNN, cf. Long et al. (2018b); de Bézenac et al. (2018); Pfaff et al. (2021); Li et al. (2021a); Brandstetter et al. (2022).

Many other approaches extend the dependency to a small window of past c frames,

$$p(u_t \mid u|_{\mathcal{T}_{t-c\delta t \leq \tau < t}}) \tag{2.39}$$

Similar to the previous model, it can be implemented as $u_t = f_\theta(u_{t-\delta t}, u_{t-2\delta t}, \ldots, u_{t-c\delta t})$. This structure has been adopted by, e.g., Li et al. (2021b).

Recurrent neural network (RNN) can also implement this process given its temporal nature. They depend on a (hidden) state s_t that evolves in time, with a temporal input z_t.

> **Example 2.11** (Elman RNN; Elman, 1990). This fully-connected RNN writes as:
>
> $$s_t = \sigma_s(W_s s_{t-\delta t} + U_z z_t + b_s) \tag{2.40}$$
> $$y_t = \sigma_y(W_y s_t + b_y) \tag{2.41}$$
>
> Eq. (2.40) describes the hidden discretized dynamics s_t perturbed by the input z_t. The decoding in Eq. (2.41) can take place at any time t to project the state into the output space y_t.

Variants have been proposed to mitigate the problem of vanishing and exploding gradients,

e.g., long short-term memory (LSTM; Hochreiter and Schmidhuber, 1997) and gated recurrent unit (GRU; Cho et al., 2014). Convolutional versions of the previous networks are proposed for spatiotemporal prediction (Shi et al., 2015), known as ConvLSTM.

When RNN is used as an autoregressive model for sequential prediction, at each timestamp t, given the input $z_t := u_t$, the corresponding output is the state at the next time step $y_t := u_{t+\delta t}$ and it will serve as the next input $z_{t+\delta t} := u_{t+\delta t}$. PredRNNs (Wang et al., 2017, 2018) are such models built upon ConvLSTM. Note that RNNs is also one of the first neural models considered for controlled dynamical systems (Pearlmutter, 1989; Funahashi and Nakamura, 1993).

Continuous-time models. Directly inspired by solvers proposed in numerical analysis, this family of approaches replaces the dynamics with neural networks and solves them to learn the dynamics. Originally, the numerical methods solve a given differential equation and output the unknown solution under initial or boundary conditions. In neural differential equations, the objective is finding the differential equation with a learnable differentiable function by fitting observed data. We demonstrate this idea with neural ordinary differential equation (Neural ODE; Chen et al., 2018).

Example 2.12 (Neural ODE; Chen et al., 2018). Popularized by Chen et al. (2018) but also mentioned in Rico-Martínez and Kevrekidis (1993); Raissi et al. (2018), Neural ODE suppose that the data can be modeled with an ODE $\frac{\mathrm{d}u}{\mathrm{d}t} = f_\theta(t, u)$. It is interpreted as a vector field, represented by a differentiable function $f_\theta \colon \mathcal{I} \times \mathcal{U} \to \mathrm{T}\mathcal{U}$.

Given an initial condition $u_0 \in \mathcal{U}$, the solution at $t \in \mathcal{I}$ is given by iterating the numerical integration with a numerical IVP scheme S

$$u_{t+\tau} = u_t + \int_{s=t}^{s=t+\tau} f_\theta(t, u_s)\mathrm{d}s \approx u_t + S(t, t+\tau, f, u_t, u_{t+\tau}) \tag{2.42}$$

The simplest scheme is the explicit forward Euler:

$$u_{t+\tau} \approx u_t + \tau \cdot f_\theta(t, u_t) \tag{2.43}$$

This scheme is strongly linked to the ResNet ☞ Example 2.5. Expanding the numerical schema S with finer time steps leads to more advanced methods, such as the explicit

4th-order Runge-Kutta:

$$u_{t+\tau} \approx u_t + \frac{\tau}{6}(k_1 + 2k_2 + 2k_3 + k_4), \tag{2.44}$$

where $k_1 = f_\theta(t, u_t), k_2 = f_\theta(t + \tau/2, u_t + \tau \cdot k_1/2), k_3 = f_\theta(t + \tau/2, u_t + \tau \cdot k_2/2), k_4 = f_\theta(t + \tau, u_t + \tau \cdot k_3)$. The gradient w.r.t. the parameter $\nabla_\theta \mathcal{L}(f_\theta, \mathcal{D})$ can be computed with the reverse mode AUTODIFF directly through the operations in the numerical scheme S or indirectly with the adjoint state method by solving an augmented system in reverse time as shown in Chen et al. (2018).

This approach can also be extended to predict time-dependent PDEs by employing a neural network that achieves an image-to-image mapping for f_θ, such as a ConvNet or an FNO. By using a numerical scheme, it is possible to compute the trajectory's states at any time t.

Numerous neural differential equation methods have been proposed for systems that have a numerical solver (Rackauckas et al., 2020). In addition to dynamical systems, they have been applied to other temporal prediction tasks, such as time series prediction, as demonstrated in Dupont et al. (2019); Kidger et al. (2020); Norcliffe et al. (2021).

2.3.4 Neural Solvers

Here we briefly mention a related but distinct type of neural method, which is not within the scope of this thesis but is related to the contribution in Chapter 7 (cf. Section 3.3 for discussions). They aim to directly solve a given differential equation with known conditions, similar to conventional solvers, without the need for data. They have gained popularity in scientific communities because of their accessible concepts and easy-to-use community implementations. The dominant method is known under the name of physics-informed neural networks (PINNs; Raissi et al., 2019) and has been proposed in parallel as deep Galerkin method (DGM; Sirignano and Spiliopoulos, 2018).

Example 2.13 (Sirignano and Spiliopoulos, 2018; Raissi et al., 2019)**.** The approach finds a solution to a boundary value problem (BVP) or IBVP. By way of example, let us consider the following IBVP:

$$\begin{gathered} \text{(PDE)} \quad F(u) = \frac{\partial u}{\partial t} - f(t,u) = 0, \text{ on } \mathcal{I} \times \Omega \qquad \text{(IC)} \quad u(0) = u_0 \in \mathcal{U} \\ \text{(BC)} \quad u_t(x) = f_b(x) \text{ on } x \in \partial\Omega \end{gathered} \tag{2.45}$$

The unknown solution u is represented by a parametrized function $u_\theta(t, x)$ that is continuously differentiable and typically implemented as an MLP with coordinate input for $(t, x) \in \mathcal{I} \times \Omega$. This solution is then optimized with an objective loss function reformulated from Eq. (2.45). By sampling x and t resp. on Ω and $\mathcal{I}$, the loss $\mathcal{L}$ is composed of, the PDE, the IC and the BC losses based on resp. F, u_0, f_b of Eq. (2.45):

$$\mathcal{L}(u_\theta, [F, u_0, f_b]) = \mathcal{L}_{\text{PDE}}(u_\theta, F) + \lambda_{\text{IC}}\mathcal{L}_{\text{IC}}(u_\theta, u_0) + \lambda_{\text{BC}}\mathcal{L}_{\text{BC}}(u_\theta, f_b) \tag{2.46}$$

where

$$\mathcal{L}_{\text{PDE}}(u_\theta, F) = \mathbb{E}_{t,x\sim\rho(\mathcal{I}\times\Omega)} \| F(u_\theta)|_{(t,x)} \|_2^2 \tag{2.47}$$

$$\mathcal{L}_{\text{IC}}(u_\theta, u_0) = \mathbb{E}_{x\sim\rho(\Omega)} \|u_\theta(0, x) - u_0(x)\|_2^2 \tag{2.48}$$

$$\mathcal{L}_{\text{BC}}(u_\theta, f_b) = \mathbb{E}_{t,x\sim\rho(\mathcal{I}\times\partial\Omega)} \|u_\theta(t, x) - f_b(x)\|_2^2 \tag{2.49}$$

In brief, the ideal solution minimizes the loss $\mathcal{L}$ to 0 to match the PDE and the conditions. The core technique of the approach is the way to calculate $F(u_\theta)$ in $\mathcal{L}_{\text{PDE}}$. For example, if F contains $\frac{\partial}{\partial x}$ and $\frac{\partial}{\partial t}$, one can firstly calculate $u_\theta(t, x)$ with a forward pass of the neural network, then retrieve the derivative at t, x with AUTODIFF w.r.t. t, x, i.e., $\frac{\partial u_\theta}{\partial x}(t, x)$ and $\frac{\partial u_\theta}{\partial t}(t, x)$. For higher-order derivatives, it suffices to do a second AUTODIFF with previous first-order derivatives.

With the support of sampling in coordinate space, the differential equation and conditions intervene in the form of a loss function in PINNs. The idea has since been adopted by a range of physical methods, e.g., Wandel et al. (2021); Wang and Perdikaris (2021); Li et al. (2021c); Goswami et al. (2022). Later data-driven approaches, such as implicit neural representations (INRs) (cf. Section 3.3.3), share a similar idea, but the difference lies in the loss function, which is based on data rather than the differential equation and conditions.

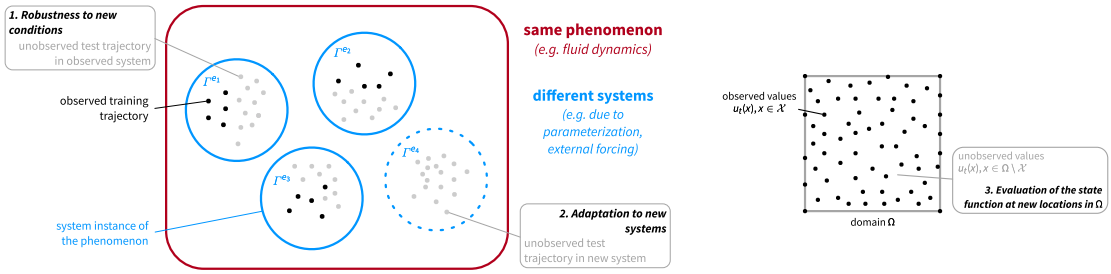
2.4 Limitations

Despite the efforts, the current mainstream of data-driven DL approaches still faces several under-addressed issues, which hinder the application to real-world scenarios and limit their generalization capabilities compared to numerical methods. In this section, we highlight these issues in detail.

What if the conditions change in the same dynamics? Here, we discuss the generalization w.r.t. to the input of the dynamics model, i.e., initial conditions, for a single dynamics. We expect our dynamics model to be robust to the change in the initial input. This is the most straightforward requirement when applying a DL dynamics model because we want the model to make accurate predictions on unseen conditions after training. For example, when the underlying PDE dynamics remain unchanged, the dynamics model should be able to correctly predict the future, even when the initial conditions change. This challenge is directly related to the concept of generalization error in machine learning (ML). However, like any data-driven ML method, purely data-driven neural methods are inherently biased toward the observed data. The problem here may arise due to the lack of sufficient inductive bias in current models, which requires the development of new approaches that allow the injection of other prior information into the model and its solutions.

What if the dynamics change? The current data-driven dynamics models focus on data from a single system of dynamics, assuming that the underlying dynamics do not change. With the current dominating learning scheme, empirical risk minimization (ERM), the modeling assumption is limited to the data from a single system of dynamics. The learned model can only predict the trajectories of the system similar to the ones seen during training. However, if the data represents multiple dynamics, the learned dynamics model would be invalid and unusable for any other dynamics.

What if the spatial sampling changes? In most cases, we need to discretize the continuous spatial states in PDE dynamics into vectors. However, this poses a challenge due to the inherent bias present in existing neural network architectures. As discussed in Section 2.3.2, at the end of the presentation of each architecture, these architectures typically have a strong structural bias toward a specific spatial sampling type and strict input-output grid coupling. While such models can extrapolate toward the future, they lack the flexibility of true continuous models which do not impose such constraints but cannot predict the future. This presents a need for new alternatives that can predict with more flexibility in space and time, as discussed in Section 3.3.



(a) Generalization in trajectory space Γ.

(b) Continuous modeling in spatial domain Ω.

Figure 2.10: Problems considered in this thesis.

2.5 Research Questions

We frame our exploration of the above questions through the following research questions. We are positioned in a low-data regime where we always have non-abundant data for training, corresponding to most real-world situations.

How to make a neural network cooperate with a numerical model? It's important to recognize the limitations of purely data-driven DL approaches, especially when working with limited data. Solutions produced by such models may lack regularity, requiring the model to be constrained to provide physically meaningful outcomes. Conversely, when a first-principles parametrized numerical model is incomplete, it cannot be used to predict the future or to identify its parameters. To address these limitations, we propose investigating how can achieve the efficient cooperation of these two models to enhance the capabilities of both.

In this case, we assume that we have also access to a neural network and a set of incomplete numerical models parameterized with optimizable variables. This setup requires not only learning the neural network but also determining and identifying the numerical model. Therefore, our research question is twofold: first, how can these two models cooperate to produce a more accurate solution, and second, how can we define the specific roles of each model to achieve this cooperation?

How can we effectively use data from different dynamics to help data-driven models generalize better? We focus on modeling dynamics with data from different sources that reflect changing dynamics. Among the limitations mentioned in Section 2.4,

we highlight the case where the dynamics in the same phenomenon differ in the parameterization of its differential equation. In such cases, we are faced with scarce and insufficient data in each system to train a high-performing neural network-based model. In this regard, our research question is how to effectively share knowledge across systems while the model is still capable of predicting trajectories in those systems.

How to adapt learned models to new dynamics? To ensure the practicality and applicability of the learned model, we need to consider its adaptability to new dynamics within the same phenomenon. Ideally, the model should be able to quickly and efficiently adapt to new systems with only a limited amount of data. Therefore, the question we ask is how to make the learned model transferable to new dynamics, while minimizing the amount of data needed for adaptation.

How to make the model structure less dependent on spatial and temporal sampling? To overcome the architectural bias limitations in current neural network architectures, we aim to develop a type of continuous modeling that only introduces bias related to data sampling and is introduced solely during the training process, rather than being dictated by the choice of features or architecture design. By doing so, we aim to increase the flexibility of the model while retaining the same predictive ability and enabling it to generalize and predict beyond the time range it has been trained on.

* * *

In the next chapter, we will provide context to these questions by surveying existing methods both in dynamics modeling and general machine learning, thus setting the stage for our main contributions in Part II.

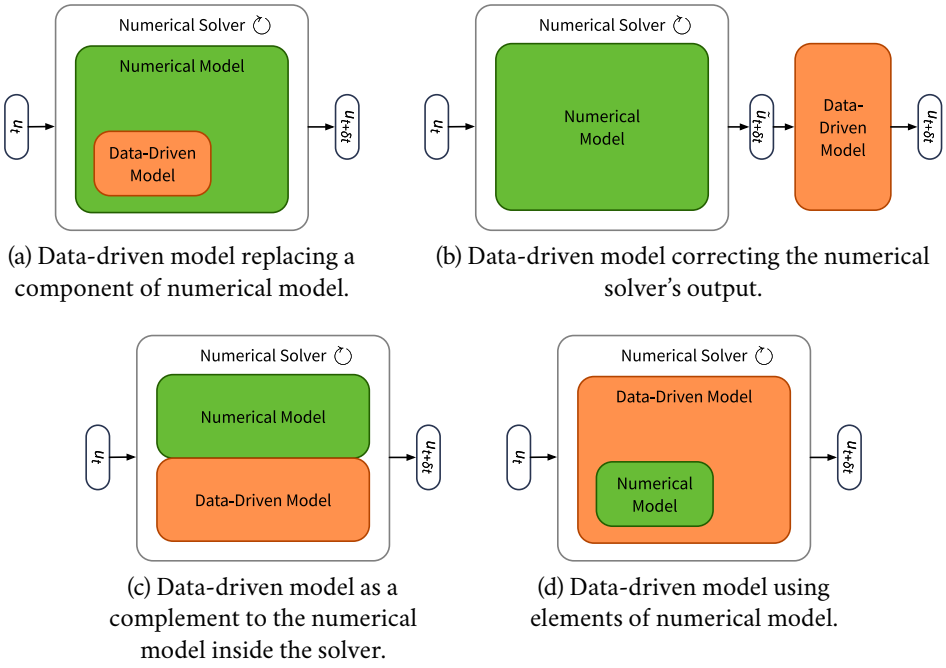
Chapter 3

Related Work

In this chapter, we provide an overview of the existing approaches, in dynamics learning and general machine learning, as they relate to the research problems we address. Section 3.1 deals with hybrid modeling using machine learning techniques to support numerical models. Section 3.2 reviews the approaches to learning models from different data sources and links them to invariance learning. Section 3.3 summarizes current approaches to modeling spatiotemporal dynamics, intending to highlight their limitations.

3.1 Physics-Enriched Deep Dynamics Models

In this section, we explore the current research avenues for integrating numerical and machine learning (ML) models. We categorize these schemes as shown in Figure 3.1, based on the level of involvement of the data-driven model components. The schemes range from the least to the most involvement:



(a) Data-driven model replacing a component of numerical model.

(b) Data-driven model correcting the numerical solver's output.

(c) Data-driven model as a complement to the numerical model inside the solver.

(d) Data-driven model using elements of numerical model.

Figure 3.1: Major hybrid modeling schemes. A model can be called several times in the solver (marked with ↻).

(a) *The data-driven model replaces components of a numerical model.* For example, Kochkov et al. (2021) use a neural network to replace an underperforming component in a low-fidelity numerical model.

(b) *The data-driven model corrects the solution of a numerical model.* It takes the output given by a numerical model through a solver and refines it by correcting errors, as shown in Belbute-Peres et al. (2020); Um et al. (2020).

(c) *The data-driven model plays a complementary role to the numerical model within the solver.* Both models are part of the overall dynamics model and predict the future together, for example, in de Bézenac et al. (2018), the neural network estimates

an unobserved variable of the dynamics, in Yin et al. (2021b), both models use the previous state to predict corresponding part of the time derivative for the solver.

(d) *The data-driven model uses elements of a numerical model.* It may consider using differential operators to calculate physically meaningful features without knowing the equation of the underlying phenomenon, as in Long et al. (2019).

Note that these schemes can be applied together within the same framework. It is also worth noting that in (b), the data-driven model rectifies the results of the numerical solver for a numerical model without interfering with the numerical model, whereas in (c), the data-driven model is part of the model to be solved. In the following, we will elaborate on the example methods belonging to these categories.

3.1.1 Correction and Acceleration of Numerical Models

Online correction: Data assimilation (DA). In DA, the expert prior model is used for calibrating the prediction according to incoming data. It has been tackled by traditional statistical calibration techniques, which often rely on Bayesian methods (Pernot and Cailliez, 2017). For data assimilation techniques, e.g., the Kalman filter (Kalman, 1960; Becker et al., 2019), 4D-Var (Courtier et al., 1994), prediction errors are modeled probabilistically, and a correction using observed data is applied after each prediction step to update the estimate of system states.

Example 3.1 (4D-Var; Courtier et al., 1994)**.** 4D-Var supposes that the dynamical system is under the form of an initial value problem (IVP): $\begin{cases} \frac{\mathrm{d}u}{\mathrm{d}t} = f(u) \\ u(0) = u_0 \end{cases}$ where *only* the initial condition is unknown, f is known and implemented through a numerical model. We find an estimate of this initial condition based on the observations $\{o_t\}_t$ at different times $t \in \mathcal{T}$ in an *assimilation window* and on the corresponding observation operators $\{H_t\}_t$ by minimizing the following loss function:

$$\mathcal{L}(u_0) = \underbrace{(u_0 - u_0^b)^\top \mathbf{B}^{-1}(u_0 - u_0^b)}_{\text{initial state loss}} + \underbrace{\sum_{t\in\mathcal{T}} (o_t - H_t(u_t))^\top \mathbf{R}_t^{-1}(o_t - H_t(u_t))}_{\text{observation loss}}$$

where $\mathbf{B}$, $\{\mathbf{R}_t\}_t$ are error covariance matrices. u_0^b, which is called *background* state, can

be the previous prediction of the model until $t = 0$. u_t is the prediction with a numerical model that implements f, until time t. Operators $\{H_t\}_t$ are supposed to be known.

There is no training in this two-stage process, and the data component is not part of the dynamics model but rather used only to calibrate the numerical model.

Learn to correct toward learn to accelerate. In a different scenario, instead of correcting the simulation based on observed data, one branch of study considers closing the gap between a low-accurate and low-computationally-expensive model and a highly accurate but computationally expensive numerical model with neural networks (NNs).

In these approaches, we suppose that two numerical models exist to solve the same differential equation, but we have only access to: (a) The low fidelity model S^{LOW}, (b) A dataset $\mathcal{D} = \{u^{\text{HIGH},(i)}\}_i$ from a reference model, of higher fidelity. The goal is to approximate higher-fidelity solutions by correcting the numerical errors when solving the low-fidelity model with a neural network. It is important to note that high-fidelity results do not necessarily mean high resolution. The fidelity of a numerical model refers to how accurately it represents the true solutions it is intended to model. These methods then learn a mapping f_θ that intervenes at each time step when solving the lower-fidelity S^{LOW}.

Example 3.2 (CFD-GCN; Belbute-Peres et al., 2020)**.** CFD-GCN proposes an approach where the neural network is conditioned by the results of a differentiable numerical solver. Given some physical parameters θ_{P}, a graph neural network (GNN) f_θ takes a dense mesh $\mathcal{X}^{\text{HIGH}}$ as input and predicts steady-state fluid velocity and pressure fields u^{HIGH}. A low-fidelity solution is given by the solver on a coarse mesh $\mathcal{X}^{\text{HIGH}}$. It is then upsampled and concatenated to the intermediate features of the neural network.

$$u^{\text{LOW}} = S^{\text{LOW}}(\mathcal{X}^{\text{LOW}}, \theta_{\text{P}})$$
$$u^{\text{HIGH}} = f_\theta(\mathcal{X}^{\text{HIGH}}, u^{\text{LOW}}, \theta_{\text{P}})$$

A differentiable solver enables the gradient calculation w.r.t. the input mesh via automatic differentiation (AUTODIFF) for eventually adjusting it to optimize the solver and neural network solutions.

Example 3.3 (Solver-in-the-Loop; Um et al., 2020)**.** Solver-in-the-Loop proposes to correct the results of a low-fidelity solver by alternating between the solver and the

neural network at every time step as follows:

$$u_{t+1}^{\text{HIGH}} \approx u_{t+1} = f_\theta \circ S^{\text{LOW}}(u_t)$$

At each time step, the solver provides a low-fidelity solution that is then corrected by the neural network f_θ. The refined prediction u_{t+1} should be close to the high-fidelity data u_{t+1}^{HIGH}.

Example 3.4 (Kochkov et al., 2021). This approach combines two techniques to improve the performance of a low-fidelity numerical model: (a) correcting the solver by a similar procedure as described in Example 3.3, and (b) replacing the underperforming unbiased components in the numerical model with the data-biased ones. The latter is achieved by replacing the unbiased interpolation module, such as cubic interpolation, with a neural network that predicts local interpolation coefficients based on previous states. These coefficients are biased toward the data to favor better prediction of high-fidelity states.

These approaches are applicable when the numerical model is fully determined, and only spatial discretization affects the quality or fidelity of the solution to real physics. ML plays the complementary role by producing the high-fidelity solution and reducing the high computational cost associated with numerical simulations.

3.1.2 Regularizing Neural Networks with Prior Knowledge

There have been several attempts in deep learning (DL) to design architectures for predicting partial differential equation (PDE) dynamics with regularized conventional neural architectures. The main goal is to reinforce certain properties that are present in target PDEs, either in the solutions or in the architectures.

Network parameter regularization. Some approaches choose to directly approximate differential operators by imposing hard constraints on convolution kernels. For instance:

Example 3.5 (PDE-Net; Long et al., 2018b). PDE-Net predicts a time-dependent PDE using a forward Euler schema $u_{t+\delta t} = u_t + \delta t \cdot f_\theta(u_t)$. Long et al. (2018b) propose a way to learn the state transition of a 2D PDE by imposing constraints on convolution kernels in a convolutional neural network (ConvNet) to exploit spatial partial derivatives. For

example, to approximate the 1st-order spatial derivatives we apply two 3-by-3 convolution kernels w_{10}, w_{01}, s.t., $\frac{\partial u_t}{\partial x_1} \approx w_{10} * u_t$, $\frac{\partial u_t}{\partial x_2} \approx w_{01} * u_t$, $w_{10} = \begin{bmatrix} 0 & 0 & \star \\ 1 & \star & \star \\ \star & \star & \star \end{bmatrix}$, $w_{01} = \begin{bmatrix} 0 & 1 & \star \\ 0 & \star & \star \\ \star & \star & \star \end{bmatrix}$ where $\star$ designates unconstrained value. From u_t, with special kernels corresponding to spatial derivatives $\{\frac{\partial u_t}{\partial x_1}, \frac{\partial u_t}{\partial x_2}, \frac{\partial^2 u_t}{\partial x_1^2}, \frac{\partial^2 u_t}{\partial x_1 x_2}, \frac{\partial^2 u_t}{\partial x_2^2}, \dots\}$ are calculated. These derivatives are used as building blocks to output the temporal change with a forward Euler schema at every x: $u_{t+\delta t}(x) = u_t(x) + \delta t \cdot f\left(x, u_t, \frac{\partial u_t}{\partial x_1}, \frac{\partial u_t}{\partial x_2}, \frac{\partial^2 u_t}{\partial x_1^2}, \frac{\partial^2 u_t}{\partial x_1 x_2}, \frac{\partial^2 u_t}{\partial x_2^2}, \dots\right)$, where f can be an multi-layer perceptron (MLP) applied point by point or an equation on the spatial grid.

This way of extracting basic PDE differential operator features has been adopted by several methods aiming to regularize dynamics when forecasting partially physical spatiotemporal sequences. For example, in Le Guen and Thome (2020) this module is combined with a recurrent neural network (RNN).

Symmetry and equivariance. Other approaches are to inject symmetry and invariance. In this regard, Wang et al. (2021c) proposed a method where the physical phenomenon's symmetry groups are identified, and architecture choices are made accordingly to implement each group's properties. The following are some examples: (a) If the equation describing the dynamics does not change with time, the predictor should give the same prediction after the same given state, regardless of when it is applied. This time translation equivariance can be implemented with autoregressive models; (b) For the neural network implementing the state transition function in autoregressive models, the output should be translated by the same offset when the input state is translated in space. This spatial translation equivariance can be implemented with any convolutional neural network; (c) If the equation contains operators describing non-directional changes in space, the output should be rotated by the same amount when the input state is rotated by a certain amount. This rotation translation equivariance can be implemented with ConvNet (Weiler and Cesa, 2019); (d) If the state changes are to be equivariant w.r.t. the spatial scale, then the convolution kernels should be adjusted according to the scale and the value. The input and the output should also be scaled at the same rate. Similar approaches have been extended to GNN by applying equivariant architectures, such as in (Horie and Mitsume, 2022).

Prior on the structure of the dynamics. This branch of predictive modeling aims to incorporate structural prior knowledge of the dynamics and causality of the dynamics into the model construction. These structures often reflect the interactions and dependencies between different components of the dynamics.

To achieve this, researchers have designed deep learning architectures that mimic the form

of the desired first-principles models. For example, Long et al. (2018a) and Saha et al. (2021) separate stochastic perturbations or source terms from the main dynamics in their architectures. For conservative systems, Greydanus et al. (2019); Chen et al. (2020b) have implemented the Hamiltonian as an NN to ensure the conservation of energy throughout the dynamics.

3.1.3 Unobserved System Parameter or State Estimation

In the pre-DL era, researchers proposed various ways to estimate the physical parameters in ordinary differential equation (ODE) systems using neural networks. For example, Psichogios and Ungar (1992) predicts the time-evolving parameters in a known numerical physical model, where the parameters depend on the current state of the system, while Thompson and Kramer (1994) directly estimate the unknown term of the incomplete ODE numerical model with time-evolving parameters. In Rico-Martinez et al. (1994), an NN was trained for systems with known parameters, and used to identify new systems with unknown parameters.

One of the pioneering DL methods, de Bézenac et al. (2018), explicitly incorporates differential operators into an end-to-end deep learning pipeline. The NN estimates a vector field over a powerful ConvNet backbone.

Example 3.6 (De Bézenac et al., 2018). We suppose that the observed sea surface temperature (SST) sequences u are governed by an advection-diffusion:

$$\frac{\partial u}{\partial t} = -(v \cdot \nabla)u + D\Delta u,$$

where v_t is an unknown vector field defined over the spatial domain Ω. One must then estimate the unknown v_t to use this previously known dynamical system.

At each time t, de Bézenac et al. (2018) use a U-Net ☞ Example 2.6 to output an estimate $\tilde{v}_t$ from the current and history SST states $u_{\leq t}$. Then, the advection-diffusion is applied with the pseudo-spectral method, at every $x \in \Omega$:

$$u_{t+\delta t}(x) = \sum_{x' \in \Omega} k(x - \tilde{v}_t(x), x')u_t(x').$$

The interpretation is that to obtain the prediction at $t + \delta t$ we backtrack the position at

> the previous time t (advection) according to the velocity field, i.e., $x - \tilde{v}_t(x)$, and apply to u_t a radial basis function (RBF) kernel k around that point (diffusion).

In this example, the NN estimates the vector field v_t, an unknown physical quantity, given the history of observed states. In a larger context, since the vector field v_t is itself driven by some unobserved dynamics, e.g., Navier-Stokes in the case above, the problem can also be seen as predicting a partially observed system, as done in Ayed et al. (2022); Donà et al. (2022).

3.2 Learning Dynamics with Data from Multiple Systems

We review the approaches considering the case where the data distribution does not follow the i.i.d. hypothesis and undergoes significant changes depending on the *environment* from which it is retrieved. To enhance the generalization ability of purely data-driven ML methods, we search for invariants across data from different environments by preserving information about the environment. This prior information is often weak, and the only knowledge about the environment is its identity. Note that while these concepts exist in the general ML community, they have primarily been applied to classification and regression problems and require further exploration for dynamics models.

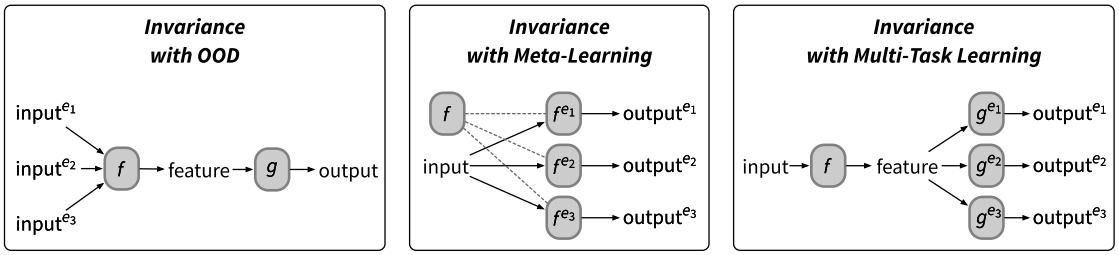


Figure 3.2: Difference between the invariance found with OoD, meta-learning, and multi-task learning. For OoD (left), the invariance is obtained by forcing the regressor $g \circ f$ to ignore the environment-related features which should not influence the output. For gradient-based meta-learning, the invariance is stored in a pre-trained initialization f. For multi-task learning, the invariance is learned by sharing a part of the information f in the model.

3.2.1 Learning Invariance for Out-of-Distribution Generalization

We first examine the existing out-of-distribution (OoD) approaches for multi-source data. Unlike empirical risk minimization (ERM), they assume that, in addition to the data itself, the *environment* $e \in \mathcal{E}$ from which it is sampled is also known. There is only a label to distinguish data from different sources. These approaches aim to find a single function (in our case a single dynamics model) that predicts well invariantly across environments $\mathcal{E}$ with the power to extrapolate outside the known distributions. Their abstract objective is to minimize the following OoD loss:

$$\mathcal{L}^{\text{OoD}}(f_\theta, \mathcal{D}) = \max_{e \in \mathcal{E}} \mathcal{L}(f_\theta, \mathcal{D}^e) \tag{3.1}$$

Here, we illustrate one approach to achieve this goal using the idea of invariant risk minimization (IRM).

Example 3.7 (IRM; Arjovsky et al., 2019)**.** To achieve the goal of finding the features that achieve consistent performance in any environment, Arjovsky et al. (2019) define a function comprising of a representation function f_{rep} and a classifier f_{cls}. The role of f_{rep} is to generate a common representation when the input from different environments should have the same output. The IRM's optimization problem is formulated as follows,

$$\begin{aligned} &\min_{f_{\text{rep}}, f_{\text{cls}}} \sum_{e \in \mathcal{E}_{\text{tr}}} \mathcal{L}(f_{\text{cls}} \circ f_{\text{rep}}, \mathcal{D}^e) \\ &\text{subject to } f_{\text{cls}} \in \arg\min_{f'_{\text{cls}}} \mathcal{L}(f'_{\text{cls}} \circ f_{\text{rep}}, \mathcal{D}^e) \quad \text{for all } e \in \mathcal{E}_{\text{tr}} \end{aligned} \tag{3.2}$$

which means that at convergence the classifier f_{cls} should perform equally well in every single training environment given a representation function f_{rep}. The entire prediction function $f_{\text{cls}} \circ f_{\text{rep}}$ remains the same across all environments.

Following the previous idea, Krueger et al. (2021) propose to attribute a larger weight to the environment with the highest training loss than other environments or minimize the variance of losses across environments. Teney et al. (2021) instead minimize the variance of the last layer parameters of the classifiers across environments. These approaches assume that there is a single underlying process independent of the environment that generates data across all environments, and aim to identify and discard spurious information that is specific to each environment, as shown in Figure 3.2.

However, in the context of dynamics modeling, the observations are often environment-dependent, and therefore, the models must be conditioned on the environment. This means that the differences between environments are not necessarily spurious and should be explicitly taken into account to build environment-aware models.

3.2.2 Learning Invariance with Meta-Learning and Multi-Task Learning

As illustrated in Figure 3.2, modeling multiple dynamics is more closely related to approaches that address multiple tasks. Although the definition of tasks is often unclear in the proposed methods, our problem falls within this scope, where each of our tasks corresponds to a dynamical system. We adopt the same notation for environments $e \in \mathcal{E}$ to represent the tasks. As always, the environment is still used only to distinguish the data sources.

Meta-learning. We focus on the optimization-based meta-learning approaches, a.k.a. gradient-based meta learning (GBML), which are primarily concerned with NNs. These approaches learn a model parameter initialization by bi-level optimization for a parameterized model f_θ. The resulting initialization is then biased toward the distribution of the target tasks, and it can be adapted to perform a new task with a few parameter updates. The standard GBML method is model-agnostic meta learning (MAML; Finn et al., 2017).

> **Example 3.8** (MAML; Finn et al., 2017)**.** Finn et al. (2017) suppose there exists a distribution of environments $e \sim \rho(\mathcal{E})$, where each environment is presented as a *task* in the original paper. The goal is to find an initialization according to the gradient direction indicated by each of the tasks. This optimization problem can be formulated as follows:
>
> $$\theta^* = \underbrace{\arg\min_{\theta} \sum_{e \in \mathcal{E}_{\text{tr}}} \mathcal{L}(f_{\theta^e}, \mathcal{D}^e_{\text{tr}})}_{\text{outer loop}}, \text{ where } \theta^e = \underbrace{\theta - \eta_{\text{inner}} \nabla_\theta \mathcal{L}(f_\theta, \mathcal{D}^e_{\text{tr}})}_{\text{inner loop}} \tag{3.3}$$
>
> In practice, MAML involves two levels of optimization: the *inner loop* and the *outer loop*. The inner loop involves taking a gradient step from the original point θ to each task with the corresponding learning rate η_{inner}, i.e., θ^e. Then, the outer loop aggregates the gradients at these new positions w.r.t. the original parameter. The trained initialization

> can then be adapted to the new task by standard parameter optimization.

This idea has been extended in various directions, including but not limited to:

- *Optimization.* The single-step inner loop can be replaced by multiple gradient updates, such as stochastic gradient descent (SGD). For outer loop optimization, there are different choices for computing the gradient w.r.t. the starting point of the inner loop. One approach involves computing higher-order derivatives, as proposed in Finn et al. (2017). Another approach involves considering only the last inner loop update, which focuses only on first-order information. This approach was explored in Nichol et al. (2018).

- *Inner loop gradient preconditioning.* Instead of directly using the gradient directions in the inner loop with a learning rate shared for each parameter, preconditioning methods aim to apply per-dimension modifications to the gradient to improve adaptation efficiency. For example, Meta-SGD (Li et al., 2017; Park et al., 2019) meta-learns dimension-wise inner-loop learning rates. Lee and Choi (2018) constrain the inner-loop update in a learned subspace. Flennerhag et al. (2020) propose a general wrapping scheme instead of the linear ones in previous examples.

- *Architectural constraints.* Depending on the task at hand, it is possible to choose which model parameters should be enabled for meta-learning. For example, ANIL (Raghu et al., 2020) the scope of MAML to the last layer of a classifier; Chen et al. (2020a) study how the neural network adapts to tasks in different problems (e.g., classification, text-to-speech conversion) and restrict meta-learning to the most frequently updated layers.

- *Contextual meta-learning.* Directly related to the previous point, some methods concentrate the adaptation in a context vector, while keeping the NN parameter invariant across tasks. For instance, Zintgraf et al. (2019); Garnelo et al. (2018) partition the parameters into two groups: the context parameters, which are adapted to each task, and the meta-trained parameters, which are shared across tasks.

In general, these methods do not focus on exploiting the commonalities and discrepancies in the data, as the discrepancies are discarded with each update. Adaptation from the learned initialization point can also suffer from overfitting on training tasks, called meta-overfitting (Mishra et al., 2018).

Multi-task learning (MTL). Another related but distinct approach to learning from multiple tasks is MTL. It involves training multiple models for different tasks simultaneously. The models are designed to share common information across tasks, One of the most common approaches is to share parameters of a certain part of the neural network, a.k.a. hard parameter sharing. For example, Caruana (1998) suggests sharing the first layers of the neural network.

> **Example 3.9** (Caruana, 1998; Baxter, 2000)**.** Hard parameter sharing was first introduced for multi-class classification problems with application to neural networks. A MLP for binary classification illustrated in Section 2.3.2 can be separated into two parts: a feature map $f_{\text{feat}}\colon \mathbb{R}^{d_{\text{in}}} \to \mathbb{R}^{d_{\text{feat}}}$ and a binary classifier $f_{\text{cls}}\colon \mathbb{R}^{d_{\text{feat}}} \to [0,1]$, i.e., $f = f_{\text{cls}} \circ f_{\text{feat}}$.
>
> To predict all C classes, the prediction for each class is output with a common f_{feat} and a set of binary classifiers $\{f_{\text{cls}}^c\}_{c\in[\![1,C]\!]}$ for an input z:
>
> $$[f_{\text{cls}}^1 \circ f_{\text{feat}}(z), f_{\text{cls}}^2 \circ f_{\text{feat}}(z), \cdots, f_{\text{cls}}^C \circ f_{\text{feat}}(z)]^\top \tag{3.4}$$

This hard-sharing technique becomes a part of the standard DL modeling for classification problems. The same inspiration was then revived in the DL era. In recent years, researchers have proposed extensions of this approach to more efficient learning from sets of related tasks, such as those presented in (Rebuffi et al., 2017, 2018). These extensions have primarily focused on classification problems, as seen in (Requeima et al., 2019; Wang et al., 2021a). Soft parameter sharing, on the other hand, preserves individual parameters while regularizing the parameters that are intended to be shared (Yang and Hospedales, 2017).

However, MTL does not address adaptation to new tasks in new environments. Recently, there has been some work showing similarities between GBML and MTL. For example, Wang et al. (2021a) use the neural tangent kernel theory framework to analyze this relationship.

Context-aware neural networks. When learning dynamics models from multiple environments, our goal is to capture the underlying function that describes the dynamics in each environment. When these environments share a large amount of information and differ only in some characteristics, it is beneficial to represent the functions in a finite-dimensional latent space to facilitate the transfer of dynamics from one environment to another.

One way of achieving this is by using a *context vector*. This is a common intuition when modeling different dynamics of the same natural phenomenon in physics. For example, a common physical model, i.e., the incompressible Navier-Stokes equation, conditioned on a few parameters (such as fluid viscosity), can describe the incompressible fluid dynamics of different materials, such as water, honey, and molten metal.

To discover the latent space where the context vectors for the observed dynamics reside, two operations should be modeled: (a) from the dynamics to the context vector, a.k.a. *encoding*, and (b) from the context vector to the dynamics, a.k.a. *decoding*, making the context useful in practice. However, two important challenges arise from these processes: discovering the context vector space and integrating context vectors with neural networks.

The first problem is how to find the context vectors for a set of observed functions $\{f\}$ through encoding. With NNs, the context vectors α can be found via *auto-encoding* or *auto-decoding*, as shown in Figure 3.3. The difference between these two options is how the codes are learned. In auto-decoding, the code is obtained directly by optimizing through a decoder $\textsc{decoder}_\phi$:

$$\underset{\{\alpha^{(i)}\}_{[\![1,N_{\text{tr}}]\!]},\phi}{\arg\min} \sum_{i}^{N_{\text{tr}}} \|\textsc{decoder}_\phi(\alpha^{(i)}) - f^{(i)}\|^2 \tag{3.5}$$

Once the decoder is trained, the auto-decoder uses gradient descent to find a code for a new function. In auto-encoding, the codes are adapted indirectly by the encoder $\textsc{encoder}_\varphi$, in cooperation with $\textsc{decoder}_\phi$:

$$\underset{\phi,\varphi}{\arg\min} \sum_{i}^{N_{\text{tr}}} \|\textsc{decoder}_\phi \circ \textsc{encoder}_\varphi(f^{(i)}) - f^{(i)}\|^2 \tag{3.6}$$

The encoder can directly output a code α for a new incoming f.

For auto-encoding, all the codes can be updated simultaneously with the decoder by updating the encoder, but it often suffers from the underfitting problem, and it is not easy to implement an encoder that has similar flexibility to the decoder; for auto-decoding, we do not need to define an encoder, but the codes must be stored in memory and the updates should be well synchronized with the update of the decoder.

The second problem relates to how we can condition a neural network relates to a context vector. One common method is to concatenate the context vector with the input directly,

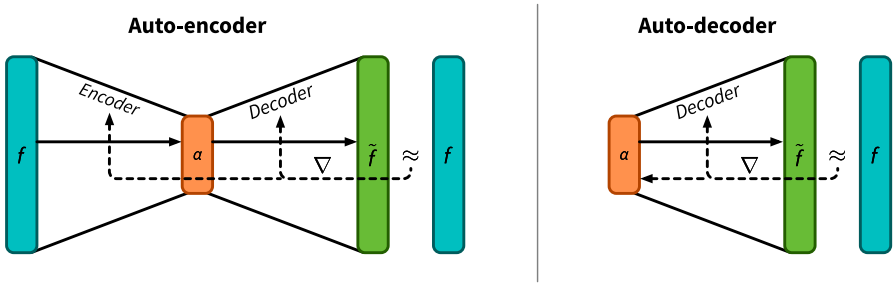


Figure 3.3: Auto-encoding vs. auto-decoding. f is the high-dimensional vector or a function to be encoded, α is the corresponding context code, and $\tilde{f}$ is the decoder's prediction. $\longrightarrow$ is the forward pass, $\dashrightarrow$ calculates gradient w.r.t. adjustable parameters.

as seen in Isola et al. (2017); Zintgraf et al. (2019); Huang et al. (2021). This simple approach modifies the first layer's bias with a linear transformation of the context. Alternatives have been discovered, e.g., for classification tasks adding the context to the last layer is preferred. These approaches condition the neural network at some hand-picked specific layers which may be suboptimal and not universally suitable for every problem.

A more universal approach is presented in Ha et al. (2017), where the parameter of a neural network is generated directly from the codes through a meta neural network. This method is more flexible and does not require the context to be added at specific layers.

Application to DL dynamics modeling. Meta-learning methods have recently been applied to dynamical systems. The objective of these mothers is to train a single model that can be quickly adapted to new dynamics with a few data points with a limited number of training steps. Lee et al. (2021) apply MAML directly to Hamiltonian neural networks (Greydanus et al., 2019) for conservative systems.

One of the rare context-aware MTL approaches for the physical environment is FT-RNN (Spieckermann et al., 2015), where they propose an RNN conditioned by a hard one-hot encoding of a set of environments $\mathcal{E}$. This fixed encoding for known environments limits the ability to adapt to new environments. Another work, DyAd (Wang et al., 2022), adapts the dynamics model by decoding a time-invariant context obtained by encoding observed states. The context is weakly supervised on some hand-picked physical quantities that may be correlated with parameters in the underlying dynamics.

3.3 Dynamics Modeling with Regularly and Irregularly Sampled Data

In this section, we focus on the limitations of current neural dynamics models regarding their ability to take sample data in space and time. We will revisit the architectures introduced in Section 2.3.2 and the types of dynamics models they support. As a reminder, in our notation, t represents an arbitrary point in time, τ represents an arbitrary time interval, and δt represents a fixed, predetermined time interval.

3.3.1 Sequential Spatially Discretized Models

Most sequential dynamics models are trained on a fixed observed grid $\mathcal{X}$ at training time and use spatially discretized models, e.g., ConvNet or GNN, to process the observations. However, these models fail experimentally when making predictions on new grids $\mathcal{X}' \neq \mathcal{X}$ because they are biased towards the training grid $\mathcal{X}$.

Learning mapping with ConvNet. One of the limitations of ConvNets (cf. p. 26) is that they are designed to operate on observations arranged on a regular grid. Once the dynamics are learned, the weights of the convolutional layers are fixed to the resolution seen during training, as the grid on which the convolution kernel is discretized cannot be changed later. This poses a challenge when dealing with data on irregular grids, as the only way to apply ConvNet-based models is to interpolate the observations to the regular grid and run the model on these interpolated values, as shown in Chae et al. (2021). However, the interpolation step introduces additional error, which can adversely affect the overall performance of the model.

Learning mapping with GNN. GNNs, especially message passing neural network (MPNN) ☞ Example 2.8 are slightly more flexible because they can handle irregular grids, albeit at a higher computational and memory cost. It is worth noting that GNNs are still biased towards the structure of the graphs, which can limit their generalization capability. For example, in the MPNN used in Brandstetter et al. (2022), messages are passed based on the difference between the positions of neighbors, which can lead to limited generalization to modified graphs.

Temporal modeling with discretized mapping. Most temporal models are limited in their spatial flexibility by previous backbone architectures for spatial signals but can extrapolate beyond the training horizon due to their sequential nature. Either using autoregressive models $u_0|_{\mathcal{X}} \mapsto (u_t|_{\mathcal{X}} \mapsto u_{t+\delta t}|_{\mathcal{X}})^{\infty}$ as in Long et al. (2018b); de Bézenac et al. (2018); Pfaff et al. (2021); Brandstetter et al. (2022). These models predict the sequence from t only with uniform time steps of δt and not for intermediate time steps. Other time-continuous extensions using numerical solvers, such as neural ordinary differential equation (Neural ODE; Chen et al., 2018) ☞ Example 2.12, remove this rigid grid in time by predicting the entire trajectory $u_0|_{\mathcal{X}} \mapsto (t \in [0, \infty) \mapsto u_t|_{\mathcal{X}})$, e.g., in Yin et al. (2021b); Iakovlev et al. (2021).

3.3.2 Operator Learning

As discussed in Chapter 2, the research on operator learning aims to propose continuous alternatives to the existing deep architectures for continuous signals by finding a parameterized mapping between functions. However, despite the continuous modeling assumption, the concrete implementations often include architectural choices and tricks that remove the theoretically desired continuous properties, such as free-from input and free-form output evaluation.

DeepONet (Lu et al., 2021). DeepONet ☞ Example 2.9 uses a coordinate-based neural network to output a prediction at arbitrary locations in time and space. This implementation allows complete freedom in evaluating the output. However, the input function is still observed on a (randomly sampled) grid in space and/or in time that cannot be changed later during the test. Recent work has attempted to overcome this limitation using attention mechanisms, as seen in Prasthofer et al. (2022).

Neural operators (NOs; Kovachki et al., 2021). NOs (see p. 34) are proposed as alternatives to the conventional convolutions in residual neural network (ResNet; He et al., 2016) (see p. 28) with a more flexible discretization of the convolution kernel. However, despite their flexibility in terms of discretization, NOs still rely on residual connections that align the input grid to the output grid, limiting their ability to freely evaluate the output function as in DeepONet.

Additionally, NOs are still subject to architectural biases that further limit their flexibility. For example, Graph neural operator (GNO; Li et al., 2020) suggests a convolution kernel

that is always discretized in space, implemented via MPNN, which inherits the same bias as discussed in Section 3.3.1. Fourier neural operator (FNO; Li et al., 2021b) ☞ Example 2.10, on the other hand, applies a convolution kernel discretized in the spectral domain via the fast Fourier transform (FFT), which limits the architecture to uniform Cartesian observation grids on a rectangular domain. Replacing the efficient FFT with the less efficient discrete Fourier transform (DFT) can address this limitation (as shown in Li et al., 2022), but it leads to an increase in computational complexity.

Temporal modeling in operator learning. Three types of temporal models have been considered for operator learning, each with its own set of limitations.

- A model derived directly from the formulation of NOs learns a mapping from an initial state to some final state at a given time $t \in [0, T]$, i.e., $u_0 \mapsto u_t$ within the training horizon (Li et al., 2020);
- A time-continuous version $u_0 \mapsto (x \in \Omega, t \in [0, T] \mapsto u_t(x))$ can be derived from DeepONet. It allows for the prediction of solutions at arbitrary time and space locations. The input of the DeepONet's branch net (see Example 2.9) is the initial condition u_0, and the output is given by a neural network taking space and time coordinates as input.
- Sequential autoregressive prediction model $u_0 \mapsto (u_t \mapsto u_{t+\delta t})^\infty$ as proposed in Li et al. (2021a). It maps an initial state u_0 to a sequence of solutions u_t at time steps $\delta t, 2\delta t, \ldots$ by recursively predicting each next state based on the previous one.

The first two approaches mentioned above are limited to the trajectory horizon T and cannot generalize beyond it. The last model, on the other hand, overcomes this limitation but is restricted to predicting solutions at fixed time steps of δt and is unable to predict solutions in-between those intervals.

3.3.3 Spatiotemporal Implicit Neural Representations

Another class of models makes also use of coordinate-based NNs, which is called implicit neural representations (INRs). The idea originates from the need of representing a 3D object with a neural network.

> **Example 3.10** (SIREN; Sitzmann et al., 2020). The idea is to encode a signal by using a neural network. A signal I sampled on a coordinate grid $\mathcal{X} \subset \Omega$ can be represented as a

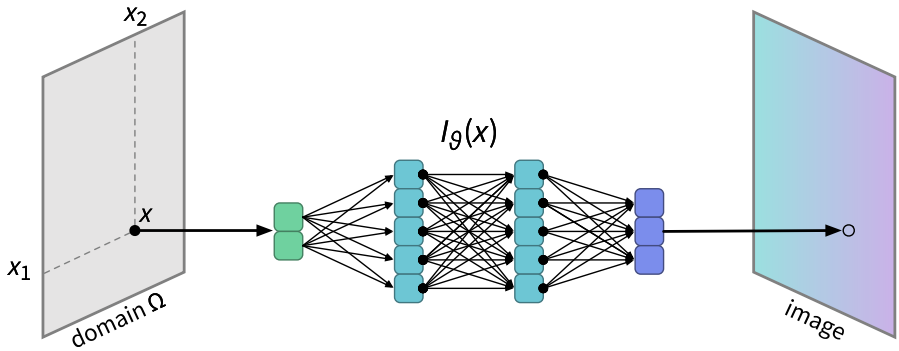


Figure 3.4: Illustration of an INR.

set of coordinate-value pairs, where $I|_{\mathcal{X}} = \{(x, I(x)) \mid I \colon \Omega \to \mathbb{R}^d, x \in \mathcal{X}\}$. As shown in Figure 3.4, to approximate this signal, an NN I_θ takes the coordinates on the input, outputs the value at the corresponding coordinates and optimizes the parameter s.t.

$$\arg\min_\theta \sum_{x \in \mathcal{X}} \ell(I_\theta(x), I(x)).$$

SIREN is an example of an INR, which is implemented as an MLP with sine activation:

$$I_\theta(x) = T^{(L)} \circ \sin \circ T^{(L-1)} \cdots \sin \circ T^{(2)} \circ \sin \circ T^{(1)}(x), \text{ where } T^{(l)}(z^{(l)}) = W^{(l)} z^{(l)} + b^{(l)}$$

There are various ways to implement an INR. Tancik et al. (2020) embed the input with sine and cosine functions to generate *Fourier features*. These features are defined as $I_\theta(\gamma(x)) = I(x)$, where $\gamma(x) = [a \odot \sin(2\pi Bx), a \odot \cos(2\pi Bx)]^\top$, $\odot$ is the element-wise product, and the resulting embedding is fed into an MLP with ReLU activation. Another approach, presented in Fathony et al. (2021), replaces the activation function by multiplications with sinusoidal functions. Although the representation of space-continuous functions is a distinct area of research, it shares a similar goal to operator learning: generating a continuous function. Additionally, it is possible to apply INRs to spatiotemporal data by incorporating time as an input parameter alongside the spatial coordinates, i.e., $I_\theta(x|t)$.

Note that this idea is also used by the physics-informed neural networks (PINNs; Raissi et al., 2019) ☞ Example 2.13. The only difference is that PINNs calculate a physical loss, allowing the neural network to solve an unknown physical function. On the other hand, INRs represents a function that is known from data. However, both types of models are restricted to representing a single solution.

Recent extensions for multi-sequence learning have been proposed, particularly with

context-aware neural networks, e.g., for video generation (Yu et al., 2022; Skorokhodov et al., 2022) or compression (Chen et al., 2021). These models learn a latent vector obtained from an initial condition u_0 and let it condition the prediction until the training horizon, i.e., $u_0 \mapsto (t \in [0, T] \mapsto u_t)$. In the field of physics, Fresca et al. (2020) propose INR approaches to building reduced order models (ROMs). Notably, these models can predict at an arbitrary time t in the training horizon without unrolling a sequential model up to t. However, since they only learn mappings from an initial condition u_0 to a function of time u_t within the training domain $[0, T]$, they fail to predict beyond the training conditions.

Part II

Contributions

Chapter 4

Hybrid Modeling with Neural Networks

Augmenting Physical Models with Deep Networks for Complex Dynamics Forecasting

In this chapter, we present our first contribution to making a neural-network-based model cooperate with an undetermined, incomplete numerical model. The objective is to correctly predict the evolution of the dynamics and identify the incomplete numerical model. This chapter led to a conference paper at ICLR 2021.

> Yuan Yin*, Vincent Le Guen*, Jérémie Donà*, Emmanuel de Bézenac*, Ibrahim Ayed*, Nicolas Thome, and Patrick Gallinari. Augmenting physical models with deep networks for complex dynamics forecasting. ICLR 2021.

4.1 Introduction

Modeling and forecasting complex dynamical systems is a major challenge in domains such as environment and climate (Rolnick et al., 2019), health science (Choi et al., 2016), and in many industrial applications (Toubeau et al., 2018). model-based (MB) approaches typically rely on partial differential equation (PDE) or ordinary differential equation (ODE) and stem from a deep understanding of the underlying physical phenomena. machine learning (ML) and deep learning (DL) methods are more prior agnostic yet have become state-of-the-art for several spatiotemporal prediction tasks (Shi et al., 2015; Wang et al., 2018; Oreshkin et al., 2020; Donà et al., 2021), and connections have been drawn between deep architectures and numerical ODE solvers, e.g., neural ODEs (Chen et al., 2018; Ayed et al., 2022). However, modeling complex physical dynamics is still beyond the scope of pure ML methods, which often cannot properly extrapolate to new conditions as MB approaches do.

Combining the MB and ML paradigms is an emerging trend to develop the interplay between the two paradigms. For example, Brunton et al. (2016) and Long et al. (2018b) learn the explicit form of PDEs directly from data, Raissi et al. (2019) and Sirignano and Spiliopoulos (2018) use neural networks (NNs) as implicit methods for solving PDEs, Seo et al. (2020) learn spatial differences with a graph network, Ummenhofer et al. (2020) introduce continuous convolutions for fluid simulations, de Bézenac et al. (2018) learn the velocity field of an advection-diffusion system, Greydanus et al. (2019) and Chen et al. (2020b) enforce conservation laws in the network architecture or in the loss function. The large majority of aforementioned MB/ML hybrid approaches assume that the physical model adequately describes the observed dynamics. This assumption is, however, commonly violated in practice. This may be due to various factors, e.g., idealized assumptions and difficulty to explain processes from first principles (Gentine et al., 2018), computational constraints prescribing a fine grain modeling of the system (Ayed et al., 2019), unknown external factors, forces and sources which are present (Large and Yeager, 2004). In this paper, we aim at leveraging prior dynamical ODE/PDE knowledge in situations where this physical model is incomplete, i.e., unable to represent the whole complexity of observed data. To handle this case, we introduce a principled learning framework to Augment incomplete PHYsical models for ideNtIfying and forecasTing

complex dYnamics (APHYNITY). The rationale of APHYNITY, illustrated in Figure 4.1 on the pendulum problem, is to *augment* the physical model when—and only when—it falls short.

Designing a general method for combining MB and ML approaches is still a wide-open problem, and a clear problem formulation for the latter is lacking (Reichstein et al., 2019). Our contributions towards these goals are the following:

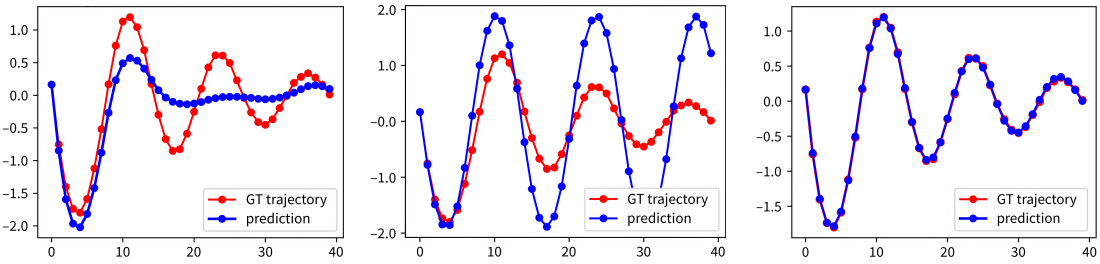
- We introduce a simple yet principled framework for combining both approaches. We decompose the data into a physical and a data-driven term such that the data-driven component only models information that cannot be captured by the physical model. We provide existence and uniqueness guarantees (Section 4.2.2) for the decomposition given mild conditions and show that this formulation ensures interpretability and benefits generalization.
- We propose a trajectory-based training formulation (Section 4.2.3) along with an adaptive optimization scheme (Section 4.2.4) enabling end-to-end learning for both physical and deep learning components. This allows APHYNITY to *automatically* adjust the complexity of the neural network to different approximation levels of the physical model, paving the way to flexible learned hybrid models.
- We demonstrate the generality of the approach on three use cases (reaction-diffusion, wave equations, and the pendulum) representative of different PDE families (parabolic, hyperbolic), having a wide spectrum of application domains, e.g., acoustics, electromagnetism, chemistry, biology, physics (Section 4.3). We show that APHYNITY is able to achieve performances close to complete physical models by augmenting incomplete ones, both in terms of forecasting accuracy and physical parameter identification. Moreover, APHYNITY can also be successfully extended to the partially observable setting (see discussion in Section 4.4).

4.2 Model

4.2.1 Problem Setting

In the following, we study dynamics driven by an equation of the form:

$$\frac{\mathrm{d}u}{\mathrm{d}t} = f(u) \tag{4.1}$$



(a) Data-driven Neural ODE
MSE = 1.5×10^{-1}

(b) Simple physical model
MSE = 7.6×10^{-1}, Error
T_0 = 12.9%

(c) Simple physical model
MSE = 1.9×10^{-4}, Error
T_0 = 0.3%

Figure 4.1: Predicted dynamics for the damped pendulum vs. ground truth (GT) trajectories $\frac{d^2\alpha}{dt^2} + \omega_0^2 \sin\alpha + \gamma\frac{d\alpha}{dt} = 0$. We show that in (a) the data-driven approach (Chen et al., 2018) fails to properly learn the dynamics due to the lack of training data, while in (b) an ideal pendulum cannot take friction into account. The proposed APHYNITY shown in (c) augments the over-simplified physical model in (b) with a data-driven component. APHYNITY improves both forecasting (MSE) and parameter identification (Error T_0) compared to (b).

defined over a finite time interval $[0, T]$, where the state u_t is either vector-valued, i.e., we have $u_t \in \mathbb{R}^d$ for every t (pendulum equations in Section 4.3), or u_t is a d-dimensional vector field over a spatial domain $\Omega \subset \mathbb{R}^p$ with $p \in \{2, 3\}$, discretized on a grid $\mathcal{X} \subset \Omega$, i.e., $u_t(x) \in \mathbb{R}^d$ for every $(t, x) \in [0, T] \times \Omega$ (reaction-diffusion and wave equations in Section 4.3). We suppose that we have access to a set of observed trajectories $\mathcal{D} = \{u^{(i)}|_{\mathcal{T}} \mid u_0^{(i)} \in \mathcal{U}, u^{(i)} \in \Gamma\}$, where $\mathcal{U}$ is the set of u_t values (either $\mathbb{R}^d$ or discretized vector field $\mathbb{R}^{d\times\#\mathcal{X}}$). In our case, the unknown f has $\mathcal{U}$ as domain and we only assume that $f \in \mathcal{F}$, with $(\mathcal{F}, \|\cdot\|)$ a normed vector space.

4.2.2 Decomposing Dynamics into Physical and Augmented Terms

As introduced in Section 4.1, we consider the common situation where incomplete information is available on the dynamics, under the form of a family of ODEs or PDEs characterized by their temporal evolution $f_{\text{P}} \in \mathcal{F}_{\text{P}} \subset \mathcal{F}$. The APHYNITY framework leverages the knowledge of $\mathcal{F}_{\text{P}}$ while mitigating the approximations induced by this simplified model through the combination of physical and data-driven components. $\mathcal{F}$ being a vector space, we can write:

$$f = f_{\text{P}} + f_{\text{A}}$$

where $f_P \in \mathcal{F}_P$ encodes the incomplete physical knowledge and $f_A \in \mathcal{F}$ is the data-driven augmentation term complementing f_P. The incomplete physical prior is supposed to belong to a known family, but the physical parameters, e.g., propagation speed for the wave equation, are unknown and need to be estimated from data. Both f_P and f_A parameters are estimated by fitting the trajectories from $\mathcal{D}$.

The decomposition $f = f_P + f_A$ is in general not unique. For example, all the dynamics could be captured by the f_A component. This decomposition is thus ill-defined, which hampers the interpretability and the extrapolation abilities of the model. In other words, one wants the estimated parameters of f_P to be as close as possible to the true parameter values of the physical model and f_A to play only a complementary role w.r.t. f_P, so *as to model only the information that cannot be captured by the physical prior*. For example, when $f \in \mathcal{F}_P$, the data can be fully described by the physical model, and in this case it is sensible to desire f_A to be nullified; this is of central importance in a setting where one wishes to identify physical quantities, and for the model to generalize and extrapolate to new conditions. In a more general setting where the physical model is incomplete, the action of f_A on the dynamics, as measured through its norm, should be as small as possible.

This general idea is embedded in the following optimization problem:

$$\min_{f_P \in \mathcal{F}_P, f_A \in \mathcal{F}} \|f_A\| \text{ subject to } \forall u \in \mathcal{D}, \forall t \in \mathcal{T}, \frac{\mathrm{d}u}{\mathrm{d}t} = (f_P + f_A)(u) \tag{4.2}$$

The originality of APHYNITY is to leverage model-based prior knowledge by augmenting it with neurally parametrized dynamics. It does so while ensuring optimal cooperation between the prior model and the augmentation.

The first key question is whether the minimum in Eq. (4.2) is indeed well-defined, in other words whether there exists indeed a decomposition with a minimal norm f_A. The answer actually depends on the geometry of $\mathcal{F}_P$, and is formulated in the following proposition:

Proposition 4.1 (Existence of a minimizing pair)**.** *If $\mathcal{F}_P$ is a proximinal set*[1]*, there exists a decomposition minimizing Eq.* (4.2)*.*

☞ Proof in Appendix A.2, p. 180

[1] A proximinal set is one from which every point of the space has at least one nearest point. A Chebyshev set is one from which every point of the space has a unique nearest point. More details in Appendix A.1.

Proximinality is a mild condition that, as shown through the proof of the proposition, cannot be weakened. It is a property verified by any boundedly compact set. In particular, it is true for closed subsets of finite dimensional spaces. However, if only existence is guaranteed, while forecasts would be expected to be accurate, non-uniqueness of the decomposition would hamper the interpretability of f_{P} and this would mean that the identified physical parameters are not uniquely determined.

It is then natural to ask under which conditions solving problem Eq. (4.2) leads to a unique decomposition into a physical and a data-driven component. The following result provides guarantees on the existence and uniqueness of the decomposition under mild conditions.

Proposition 4.2 (Uniqueness of the minimizing pair)**.** *If $\mathcal{F}_{\text{P}}$ is a Chebyshev set*[1]*, Eq. (4.2) admits a unique minimizer. The f_{P} in this minimizer pair is the metric projection of the unknown f onto $\mathcal{F}_{\text{P}}$.*

☞ Proof in Appendix A.2, p. 180

The Chebyshev assumption condition is strictly stronger than proximinality but is still quite mild and necessary. Indeed, in practice, many sets of interest are Chebyshev, including all closed convex spaces in strict normed spaces, and, if $\mathcal{F} = L^2$, $\mathcal{F}_{\text{P}}$ can be any closed convex set, including all finite-dimensional subspaces. In particular, all examples considered in the experiments are Chebyshev sets.

Propositions 4.1 and 4.2 provide, under mild conditions, the theoretical guarantees for the APHYNITY formulation to infer the correct MB/ML decomposition, thus enabling both recovering the proper physical parameters and accurate forecasting.

4.2.3 Solving APHYNITY with Deep Neural Networks

In the following, both terms of the decomposition are parametrized and are denoted as $f_{\text{P}}^{\theta_{\text{P}}}$ and $f_{\text{A}}^{\theta_{\text{A}}}$. Solving APHYNITY then consists in estimating the parameters θ_{P} and θ_{A}. θ_{P} are the physical parameters and are typically low-dimensional, e.g., 2 or 3 in our experiments for the considered physical models. For f_{A}, we need sufficiently expressive models able to optimize over all $\mathcal{F}$: we thus use deep neural networks, which have shown promising performances for the approximation of differential equations (Raissi et al., 2019; Ayed et al., 2022).

When learning the parameters of $f_{\text{P}}^{\theta_{\text{P}}}$ and $f_{\text{A}}^{\theta_{\text{A}}}$, we have access to a finite dataset of trajectories

discretized with a given temporal resolution δt: $\mathcal{D}_{\text{train}} = \{u^{(i)}|_{\hat{\mathcal{T}}}\}_{i\in[\![1,N]\!]}$, where $\mathcal{T} = \{k\delta t\}_{k\in[\![0,T/\delta t]\!]}$. Solving Eq. (4.2) requires estimating the state derivative $\frac{\mathrm{d}u(t)}{\mathrm{d}t}$ appearing in the constraint term. One solution is to approximate this derivative using, e.g., finite differences as in Brunton et al. (2016); Greydanus et al. (2019); Cranmer et al. (2020). This numerical scheme requires high space and time resolutions in the observation space in order to get reliable gradient estimates. Furthermore, it is often unstable, leading to explosive numerical errors as discussed in Appendix A.4. We propose instead to solve Eq. (4.2) using an integral trajectory-based approach: we compute $\tilde{u}^{(i)}|_{\mathcal{T}}$ from an initial state $u_0^{(i)}$ using the current $f_{\text{P}}^{\theta_{\text{P}}} + f_{\text{A}}^{\theta_{\text{A}}}$ dynamics, then enforce the constraint $\tilde{u}^{(i)}|_{\mathcal{T}} = u^{(i)}|_{\mathcal{T}}$. This leads to our final objective function on $(\theta_{\text{P}}, \theta_{\text{A}})$:

$$\min_{\theta_{\text{P}},\theta_{\text{A}}} \left\|f_{\text{A}}^{\theta_{\text{A}}}\right\| \quad \text{subject to} \quad \forall i \in [\![1, N]\!], t \in \mathcal{T}, \tilde{u}_t^{(i)} = u_t^{(i)} \tag{4.3}$$

where $\tilde{u}_t^{(i)}$ is the approximate solution of the integral $u_0^{(i)} + \int_{s=0}^{s=t} (f_{\text{P}}^{\theta_{\text{P}}} + f_{\text{A}}^{\theta_{\text{A}}})(u(s))\mathrm{d}s$ obtained by a differentiable ODE solver.

In our setting, where we consider situations for which $f_{\text{P}}^{\theta_{\text{P}}}$ only partially describes the physical phenomenon, this coupled MB + ML formulation leads to different parameter estimates than using the MB formulation alone, as analyzed more thoroughly in Appendix A.3. Interestingly, our experiments show that using this formulation also leads to better identification of the physical parameters θ_{P} than when fitting the simplified physical model $f_{\text{P}}^{\theta_{\text{P}}}$ alone (Section 4.3). With only an incomplete knowledge of physics, the θ_{P} estimator will be biased by the additional dynamics which need to be fitted in the data. Appendix A.6 also confirms that the integral formulation gives better forecasting results and a more stable behavior than supervising over finite difference approximations of the derivatives.

4.2.4 Adaptively Constrained Optimization

Algorithme 1 : APHYNITY

Initialization: $\lambda \geq 0, \eta_1 > 0, \eta_2 > 0$
for *j in* $[\![1, N_{epochs}]\!]$ **do**
  for *iter in* $[\![1, N_{iter}]\!]$ **do**
    $\theta \leftarrow \theta - \eta_1 \nabla \left[\|f_{\text{A}}\| + \lambda \mathcal{L}_{\text{traj}}(f_{\text{P}}^{\theta_p}, f_{\text{A}}^{\theta_{\text{A}}}) \right]$ `// update` $\theta = (\theta_{\text{P}}, \theta_{\text{A}})$
  $\lambda \leftarrow \lambda + \eta_2 \mathcal{L}_{\text{traj}}(f_{\text{P}}^{\theta_p}, f_{\text{A}}^{\theta_{\text{A}}})$ `// update` λ

The formulation in Eq. (4.3) involves constraints that are difficult to enforce exactly in practice. We considered a variant of the method of multipliers (Bertsekas, 1996) which uses a sequence of Lagrangian relaxations $\mathcal{L}_{\lambda_j}(f_{\mathrm{P}}^{\theta_{\mathrm{P}}}, f_{\mathrm{A}}^{\theta_{\mathrm{A}}})$:

$$\mathcal{L}_{\lambda_j}(f_{\mathrm{P}}^{\theta_{\mathrm{P}}}, f_{\mathrm{A}}^{\theta_{\mathrm{A}}}) = \|f_{\mathrm{A}}^{\theta_{\mathrm{A}}}\| + \lambda_j \cdot \mathcal{L}_{\mathrm{traj}}(f_{\mathrm{P}}^{\theta_{\mathrm{P}}}, f_{\mathrm{A}}^{\theta_{\mathrm{A}}}) \tag{4.4}$$

where $\mathcal{L}_{\mathrm{traj}}(f_{\mathrm{P}}^{\theta_{\mathrm{P}}}, f_{\mathrm{A}}^{\theta_{\mathrm{A}}}) = \sum_{u\in\mathcal{D}} \sum_{t\in\mathcal{T}} \|u_t^{(i)} - \tilde{u}_t^{(i)}\|$.

This method needs an increasing sequence $(\lambda_j)_j$ such that the successive minima of $\mathcal{L}_{\lambda_j}$ converge to a solution (at least a local one) of the constrained problem Eq. (4.3). We select $(\lambda_j)_j$ by using an iterative strategy: starting from a value λ_0, we iterate, minimizing $\mathcal{L}_{\lambda_j}$ by gradient descent[2], then update λ_j with: $\lambda_{j+1} = \lambda_j + \eta_2 \mathcal{L}_{\mathrm{traj}}(\theta_{j+1})$, where η_2 is a chosen hyper-parameter and $\theta = (\theta_{\mathrm{P}}, \theta_{\mathrm{A}})$. This procedure is summarized in Algorithm 1. This adaptive iterative procedure allows us to obtain stable and robust results, in a reproducible fashion, as shown in the experiments.

4.3 Experimental Validation

We validate our approach on 3 classes of challenging physical dynamics: reaction-diffusion, wave propagation, and the damped pendulum, representative of various application domains such as chemistry, biology or ecology (for reaction-diffusion) and earth physic, acoustic, electromagnetism, or even neurobiology (for waves equations). The two first dynamics are described by PDEs and thus in practice should be learned from very high-dimensional vectors, discretized from the original compact domain. This makes the learning much more difficult than from the one-dimensional pendulum case. For each problem, we investigate the cooperation between physical models of increasing complexity encoding incomplete knowledge of the dynamics (denoted *Incomplete physics* in the following) and data-driven models. We show the relevance of APHYNITY (denoted *APHYNITY models*) both in terms of forecasting accuracy and physical parameter identification.

4.3.1 Experimental Setting

We describe the three families of equations studied in the experiments. In all experiments, $\mathcal{F} = L^2(\mathcal{U})$ where $\mathcal{U}$ is the set of all admissible states for each problem, and the L^2 norm is

[2] Convergence to a local minimum isn't necessary, a few steps are often sufficient for successful optimization.

computed on $\mathcal{D}_{\text{tr}}$ by $\|f\|^2 \approx \sum_{i,t} \|f(u_t^{(i)})\|^2$. All considered sets of physical functionals $\mathcal{F}_{\text{P}}$ are closed and convex in $\mathcal{F}$ and thus are Chebyshev. In order to enable the evaluation of both prediction and parameter identification, all our experiments are conducted on simulated datasets with known model parameters. Each dataset has been simulated using an appropriate high-precision integration scheme for the corresponding equation. All solver-based models take the first state u_0 as input and predict the remaining time-steps by integrating f through the same differentiable generic and common ODE solver (4th order Runge-Kutta)[3]. Implementation details and architectures are given in Appendix A.5.

Reaction-diffusion equations. We consider a 2D FitzHugh-Nagumo type model (Klaasen and Troy, 1984). The system is driven by the PDE

$$\frac{\partial v}{\partial t} = a\Delta v + v - v^3 - k - w$$
$$\frac{\partial w}{\partial t} = b\Delta w + v - w$$

where a and b are respectively the diffusion coefficients of v and w, Δ is the Laplace operator. The local reaction terms are $(v - v^3 - k - w)$ for v, and $(v - w)$ for w. The state is $u_t = (v_t, w_t)$ and is defined over a compact rectangular domain Ω with periodic boundary conditions. The considered physical models are:

- *Param PDE (a, b)*, with unknown (a, b) diffusion terms and without reaction terms:

$$\mathcal{F}_{\text{P}} = \{f_{\text{P}}^{a,b} \colon (v_t, w_t) \mapsto (a\Delta v_t, b\Delta w_t) \mid a, b \in [\epsilon, +\infty)\};$$

- *Param PDE (a, b, k)*, the full PDE with unknown parameters:

$$\mathcal{F}_{\text{P}} = \{f_{\text{P}}^{a,b,k} \colon (v_t, w_t) \mapsto (a\Delta v_t + v_t - v_t^3 - k - w_t, b\Delta w_t + v_t - w_t \mid a, b, k \in [\epsilon, +\infty)\}.$$

Damped wave equations. We investigate the damped-wave PDE:

$$\frac{\partial^2 w}{\partial t^2} - c^2\Delta w + \gamma\frac{\partial w}{\partial t} = 0$$

[3]This integration scheme is then different from the one used for data generation, the rationale for this choice being that when training a model one does not know how exactly the data has been generated.

where γ is the damping coefficient. The state is $u_t = \left(w_t, \frac{\partial w_t}{\partial t}\right)$ and we consider a compact spatial domain Ω with Neumann homogeneous boundary conditions. Note that this damping differs from the pendulum, as its effect is global. Our physical models are:

- *Param PDE (c)*, without damping term:

$$\mathcal{F}_{\text{P}} = \left\{ f_{\text{P}}^{c} : \left(w_t, \frac{\partial w_t}{\partial t}\right) \mapsto \frac{\partial w_t}{\partial t}, c^2 \Delta w_t \mid c \in [\epsilon, +\infty) \text{ with } \epsilon > 0 \right\};$$

- *Param PDE (c, k)*:

$$\mathcal{F}_{\text{P}} = \left\{ f_{\text{P}}^{c,k} : \left(w_t, \frac{\partial w_t}{\partial t}\right) \mapsto \left(\frac{\partial w_t}{\partial t}, c^2 \Delta w_t - \gamma \frac{\partial w_t}{\partial t}\right) \mid c, k \in [\epsilon, +\infty) \text{ with } \epsilon > 0 \right\}.$$

Damped pendulum. The evolution follows the ODE

$$\frac{\mathrm{d}^2 \alpha}{\mathrm{d}t^2} + \omega_0^2 \sin \alpha + \gamma \frac{\mathrm{d}\alpha}{\mathrm{d}t} = 0,$$

where α_t is the angle, ω_0 the proper pulsation (T_0 the period) and γ the damping coefficient. The state $u_t = \left(\alpha_t, \frac{\mathrm{d}\alpha_t}{\mathrm{d}t}\right)$. Our physical models are:

- *Hamiltonian* (Greydanus et al., 2019), a conservative approximation, with

$$\mathcal{F}_{\text{P}} = \left\{ f_{\text{P}}^{\mathcal{H}} : \left(\alpha_t, \frac{\mathrm{d}\alpha_t}{\mathrm{d}t}\right) \mapsto \left(\partial_{\dot{\alpha}_t} \mathcal{H}\left(\alpha_t, \frac{\mathrm{d}\alpha_t}{\mathrm{d}t}\right), -\partial_{\alpha_t} \mathcal{H}\left(\alpha_t, \frac{\mathrm{d}\alpha_t}{\mathrm{d}t}\right)\right) \mid \mathcal{H} \in H^1(\mathbb{R}^2) \right\},$$

  where $H^1(\mathbb{R}^2)$ is the first order Sobolev space.

- *Param ODE (ω_0)*, the ideal, frictionless pendulum:

$$\mathcal{F}_{\text{P}} = \left\{ f_{\text{P}}^{\omega_0} : \left(\alpha_t, \frac{\mathrm{d}\alpha_t}{\mathrm{d}t}\right) \mapsto \left(\frac{\mathrm{d}\alpha_t}{\mathrm{d}t}, -\omega_0^2 \sin \alpha_t\right) \mid \omega_0 \in [\epsilon, +\infty) \text{ with } \epsilon > 0 \right\};$$

- *Param ODE (ω_0, γ)*, the full pendulum equation:

$$\mathcal{F}_{\text{P}} = \left\{ f_{\text{P}}^{\omega_0, \alpha} : \left(\alpha_t, \frac{\mathrm{d}\alpha_t}{\mathrm{d}t}\right) \mapsto \left(\frac{\mathrm{d}\alpha_t}{\mathrm{d}t}, -\omega_0^2 \sin \alpha_t - \gamma \frac{\mathrm{d}\alpha_t}{\mathrm{d}t}\right) \mid \omega_0, \alpha \in [\epsilon, +\infty) \text{ with } \epsilon > 0 \right\}.$$

Baselines. As purely data-driven baselines, we use neural ordinary differential equation (Neural ODE; Chen et al., 2018) for the three problems and PredRNN++ (Wang et al., 2018, for reaction-diffusion only) which are competitive models for datasets generated by differential equations and for spatio-temporal data. As MB/ML methods, in the ablations studies (see Appendix A.6), we compare all problems, to the vanilla MB/ML cooperation scheme found in Wang et al. (2019); Mehta et al. (2021). We also show results for *True PDE/ODE*, which corresponds to the equation for data simulation. This does not lead to zero error due to the difference between simulation and training integration schemes. For the pendulum, we compare to Hamiltonian neural networks (Greydanus et al., 2019; Toth et al., 2020) and to the deep Galerkin method (DGM; Sirignano and Spiliopoulos, 2018). See additional details in Appendix A.5.

4.3.2 Results

We analyze and discuss below the results obtained for the three kind of dynamics. We successively examine different evaluation or quality criteria. The conclusions are consistent for the three problems, which allows us to highlight clear trends for all of them.

Forecasting accuracy. The data-driven models do not perform well compared to *True PDE/ODE* (all values are test errors expressed as log MSE): −4.60 for PredRNN++ vs. −9.17 for reaction-diffusion, −2.51 vs. −5.24 for wave equation, and −2.84 vs. −8.44 for the pendulum in Table 4.1. The Deep Galerkin method for the pendulum in complete physics *DGM* (ω_0, γ), being constrained by the equation, outperforms Neural ODE but is far inferior to APHYNITY models. In the incomplete physics case, *DGM* (ω_0) fails to compensate for the missing information. The *incomplete physical models*, *Param PDE* (a, b) for the reaction-diffusion, *Param PDE* (c) for the wave equation, and *Param ODE* (ω_0) and *Hamiltonian models* for the damped pendulum, have even poorer performances than purely data-driven ones, as can be expected since they ignore important dynamical components, e.g., friction in the pendulum case. Using APHYNITY with these imperfect physical models greatly improves forecasting accuracy in all cases, significantly outperforming purely data-driven models, and reaching results often close to the accuracy of the true ODE, when APHYNITY and the true ODE models are integrated with the same numerical scheme (which is different from the one used for data generation, hence the non-null errors even for the true equations), e.g., −5.92 vs. −5.24 for the wave equation in Table 4.1. This clearly highlights the capacity of our approach to augment incomplete physical models with a learned data-driven component.

Table 4.1: Forecasting and identification results on the (a) reaction-diffusion, (b) wave equation, and (c) damped pendulum datasets. We set for (a) $a = 1\times10^{-3}, b = 5\times10^{-3}, k = 5\times10^{-3}$, for (b) $c = 330$, $k = 50$ and for (c) $T_0 = 6$, $\gamma = 0.2$ as true parameters. log MSEs are computed respectively over 25, 25, and 40 predicted timesteps. %Err param. averages the results when several physical parameters are present. For each level of incorporated physical knowledge, equivalent best results according to a Student t-test are shown in bold. — corresponds to non-applicable cases.

Dataset		Method	log MSE	%Err param.	$\|\|f_A\|\|^2$
(a) Reaction-diffusion	Data-driven	Neural ODE	−3.76±0.02	—	—
		PredRNN++	−4.60±0.01	—	—
	Incomplete physics	Param PDE (a, b)	−1.26±0.02	67.6	—
		APHYNITY Param PDE (a, b)	**−5.10±0.21**	**2.3**	67
	Complete physics	Param PDE (a, b, k)	**−9.34±0.20**	0.17	—
		APHYNITY Param PDE (a, b, k)	**−9.35±0.02**	**0.096**	1.5×10^{-6}
		True PDE	−8.81±0.05	—	—
		APHYNITY True PDE	**−9.17±0.02**	—	1.4×10^{-7}
(b) Wave equation	Data-driven	Neural ODE	−2.51±0.29	—	—
	Incomplete physics	Param PDE (c)	0.51±0.07	10.4	—
		APHYNITY Param PDE (c)	**−4.64±0.25**	**0.31**	71
	Complete physics	Param PDE (c, k)	−4.68±0.55	1.38	—
		APHYNITY Param PDE (c, k)	**−6.09±0.28**	**0.70**	4.54
		True PDE	−4.66±0.30	—	—
		APHYNITY True PDE	**−5.24±0.45**	—	0.14

→ *Continued on next page*

Table 4.1: *Continued*

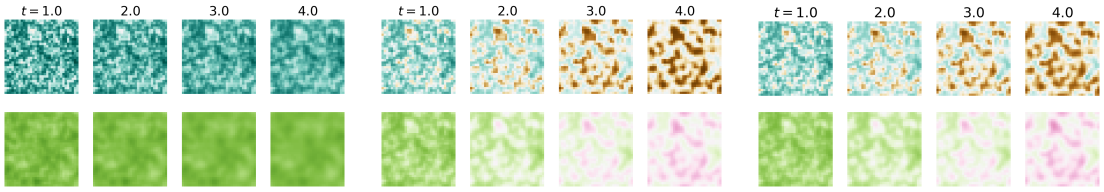
Dataset		Method	log MSE	%Err param.	$\|\|f_A\|\|^2$
(c) Damped pendulum	Data-driven	Neural ODE	−2.84±0.70	—	—
	Incomplete physics	Hamiltonian	−0.35±0.10	—	—
		APHYNITY Hamiltonian	**−3.97±1.20**	—	623
		Param ODE (ω_0)	−0.14±0.10	13.2	—
		Deep Galerkin Method (ω_0)	−3.10±0.40	22.1	—
		APHYNITY Param ODE (ω_0)	**−7.86±0.60**	**4.0**	132
	Complete physics	Param ODE (ω_0, γ)	**−8.28±0.40**	**0.45**	—
		Deep Galerkin Method (ω_0, γ)	−3.14±0.40	7.1	—
		APHYNITY Param ODE (ω_0, γ)	**−8.31±0.30**	**0.39**	8.5
		True ODE	**−8.58±0.20**	—	—
		APHYNITY True ODE	**−8.44±0.20**	—	2.3

Physical parameter estimation. Confirming the phenomenon mentioned in the introduction and detailed in Appendix A.3, incomplete physical models can lead to bad estimates for the relevant physical parameters: an error respectively up to 67.6% and 10.4% for parameters in the reaction-diffusion and wave equations, and an error of more than 13% for parameters for the pendulum in Table 4.1. APHYNITY is able to significantly improve physical parameters identification: 2.3% error for the reaction-diffusion, 0.3% for the wave equation, and 4% for the pendulum. This validates the fact that augmenting a simple physical model to compensate for its approximations is not only beneficial for prediction but also helps to limit errors for parameter identification when dynamical models do not fit data well. This is crucial for the interpretability and explainability of the estimates.

Ablation study. We conduct ablation studies to validate the importance of the APHYNITY augmentation compared to a naive strategy consisting in learning $f = f_P + f_A$ without taking care of the quality of the decomposition, as done in Wang et al. (2019) and Mehta et al. (2021). Results shown in Table A.4 of Appendix A.6 show a consistent gain of APHYNITY for the three use cases and for all physical models: for instance for *Param ODE (a, b)* in reaction-diffusion, both forecasting performances (log MSE = −5.10 vs. −4.56) and identification parameter (Error = 2.33 % vs. 6.39 %) improve. Other ablation results are provided in Appendix A.6 showing the relevance of the trajectory-based approach described in Section 4.2.3 (vs. supervising over finite difference approximations of the derivative f).

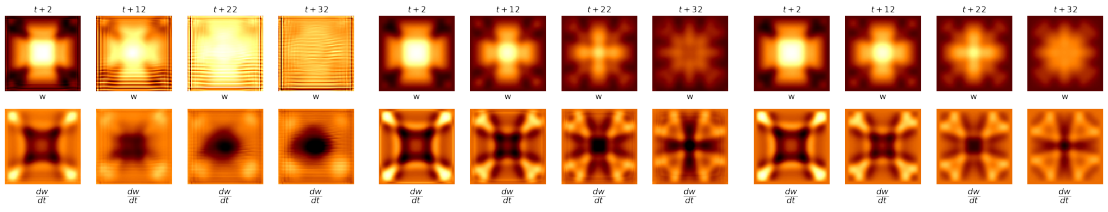
Flexibility. When applied to complete physical models, APHYNITY does not degrade accuracy, contrary to a vanilla cooperation scheme (see ablations in Appendix A.6). This is due to the least action principle of our approach: when the physical knowledge is sufficient for properly predicting the observed dynamics, the model learns to ignore the data-driven augmentation. This is shown by the norm of the trained neural net component f_A, which is reported in Table 4.1 last column: as expected, $\|f_A\|^2$ diminishes as the complexity of the corresponding physical model increases, and, relative to incomplete models, the norm becomes very small for complete physical models (for example in the pendulum experiments, we have $\|f_A\| = 8.5$ for the APHYNITY model to be compared with 132 and 623 for the incomplete models). Thus, we see that the norm of f_A is a good indication of how imperfect the physical models $\mathcal{F}_P$ are. It highlights the flexibility of APHYNITY to successfully adapt to very different levels of prior knowledge. Note also that APHYNITY sometimes slightly improves over the true ODE, as it compensates for

the error introduced by different numerical integration methods for data simulation and training (see Appendix A.5).



(a) Param PDE (a, b), diffusion-only (b) APHYNITY Param PDE (a, b) (c) Ground truth simulation

Figure 4.2: Comparison of predictions of two components u (top) and v (bottom) of the reaction-diffusion system. Note that $t = 4$ is largely beyond the dataset horizon ($t = 2.5$).



(a) Neural ODE (b) APHYNITY Param PDE (c) (c) Ground truth simulation

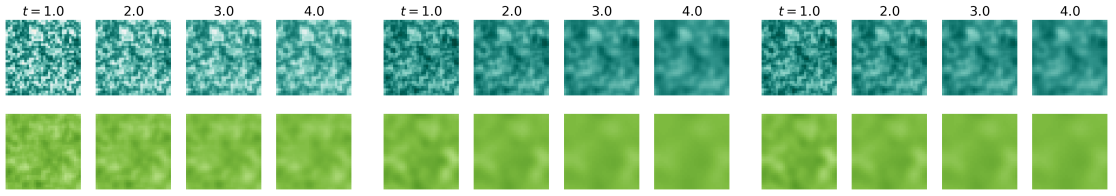
Figure 4.3: Comparison between the prediction of APHYNITY when c is estimated and Neural ODE for the damped wave equation. Note that $t = 32$ is already beyond the training time horizon $T = 25$.

Qualitative visualizations. Results in Figure 4.2 for reaction-diffusion show that the incomplete diffusion parametric PDE in Figure 4.2a is unable to properly match ground truth simulations: the behavior of the two components in Figure 4.2a is reduced to simple independent diffusions due to the lack of interaction terms between v and w. By using APHYNITY in Figure 4.2b, the correlation between the two components appears together with the formation of Turing patterns, which is very similar to the ground truth. This confirms that f_θ can learn the reaction terms and improve prediction quality. In Figure 4.3, we see for the wave equation that the data-driven Neural ODE model fails at approximating $\frac{\mathrm{d}w}{\mathrm{d}t}$ as the forecast horizon increases: it misses crucial details for the second component $\frac{\mathrm{d}w}{\mathrm{d}t}$ which makes the forecast diverge from the ground truth. APHYNITY incorporates a Laplacian term as well as the data-driven f_θ thus capturing the damping phenomenon and

succeeding in maintaining physically sound results for long-term forecasts, unlike Neural ODE.

Extension to non-stationary dynamics. We provide additional results in Appendix A.7 to tackle datasets where the physical parameters of the equations vary in each sequence. To this end, we design an encoder able to perform parameter estimation for each sequence. Results show that APHYNITY accommodates well to this setting, with similar trends as those reported in this section.

Additional illustrations We give further visual illustrations to demonstrate how the estimation of parameters in incomplete physical models is improved with APHYNITY. For the reaction-diffusion equation, we show that the incomplete parametric PDE underestimates both diffusion coefficients. The difference is visually recognizable between the poorly estimated diffusion (Figure 4.4a) and the true one (Figure 4.4c) while APHYNITY gives a fairly good estimation of those diffusion parameters as shown in Figure 4.4b.



(a) $a = 0.33\times10^{-3}, b = 0.94\times10^{-3}$, diffusion estimated with Param PDE (a, b)

(b) $a = 0.97\times10^{-3}, b = 4.75\times10^{-3}$, diffusion estimated with APHYNITY Param PDE (a, b)

(c) $a = 1.00\times10^{-3}, b = 5.00\times10^{-3}$, true diffusion

Figure 4.4: Diffusion predictions using coefficient for reaction-diffusion, either learned with (a) incomplete physical model Param PDE (a, b), or (b) APHYNITY-augmented Param PDE (a, b), compared with the (c) true diffusion.

4.4 Conclusion

In this work, we introduce the APHYNITY framework that can efficiently augment approximate physical models with deep data-driven networks, performing similarly to models for which the underlying dynamics are entirely known. We exhibit the superiority of APHYNITY over data-driven, incomplete physics, and state-of-the-art approaches combining ML and MB methods, both in terms of forecasting and parameter identification

on three various classes of physical systems. Besides, APHYNITY is flexible enough to adapt to different approximation levels of prior physical knowledge.

An appealing perspective is the applicability of APHYNITY in partially-observable settings, such as video prediction. Besides, we hope that the APHYNITY framework will open up the way to the design of a wide range of more flexible MB/ML models, e.g., in climate science, robotics, or reinforcement learning. In particular, analyzing the theoretical decomposition properties in a partially-observed setting is an important direction for future work.

Acknowledgements

Part of this work has been supported by project DL4CLIM, ANR-19-CHIA-0018-01.

Chapter 5

Learning to Generalize in Known Systems

Learning Dynamical Systems that Generalize across Environments

In this chapter, we present our contribution on leveraging the common knowledge in data retrieved from different dynamics. The goal is to show how the commonalities in the multi-source data can be exploited to improve the generalization in both known and novel dynamics. This work resulted in a conference paper at NeurIPS 2021 and paved the way for our next contribution in Chapter 6.

Yuan Yin, Ibrahim Ayed, Emmanuel de Bézenac, Nicolas Baskiotis, Patrick Gallinari. LEADS: Learning dynamical systems that generalize across environments. NeurIPS 2021.

5.1 Introduction

Data-driven approaches offer an interesting alternative and complement to physical-based methods for modeling the dynamics of complex systems and are particularly promising in a wide range of settings: for example, if the underlying dynamics are partially known or understood, if the physical model is incomplete, inaccurate, or fails to adapt to different contexts, or if external perturbation sources and forces are not modeled. The idea of deploying machine learning (ML) to model complex dynamical systems picked momentum a few years ago, relying on recent deep learning progresses and on the development of new methods targeting the evolution of temporal and spatiotemporal systems (Brunton et al., 2016; de Bézenac et al., 2018; Chen et al., 2018; Long et al., 2018b; Raissi et al., 2019; Ayed et al., 2022; Yin et al., 2021b). It is already being applied in different scientific disciplines (see Willard et al. (2023) for a recent survey) and could help accelerate scientific discovery to address challenging domains such as climate (Reichstein et al., 2019) or health (Fresca et al., 2020).

However, despite promising results, current developments are limited and usually postulate an idealized setting where data is *abundant* and *the environment does not change*, the so-called "i.i.d. hypothesis". In practice, real-world data may be expensive or difficult to acquire. Moreover, changes in the environment may be caused by many different factors. For example, in climate modeling, there are external forces, e.g., Coriolis, which depend on the spatial location (Gurvan et al., 2022); or, in health science, parameters need to be personalized for each patient as for cardiac computational models (Neic et al., 2017). More generally, data acquisition and modeling are affected by different factors such as geographical position, sensor variability, measuring circumstances, etc. The classical paradigm either considers all the data as i.i.d. and looks for a global model, or proposes specific models for each environment. The former disregards discrepancies between the environments, thus leading to a biased solution with an averaged model which will

usually perform poorly. The latter ignores the similarities between environments, thus affecting generalization performance, particularly in settings where per-environment data is limited. This is particularly problematic in dynamical settings, as small changes in initial conditions lead to trajectories not covered by the training data.

In this work, we consider a setting where it is explicitly assumed that the trajectories are collected from different environments. Note that in this setting, the i.i.d. hypothesis is removed twice: by considering the temporality of the data and by the existence of multiple environments. In many useful contexts, the dynamics in each environment share similarities, while being distinct which translates into changes in the data distributions. Our objective is to leverage the similarities between environments in order to improve the modeling capacity and generalization performance, while still carefully dealing with the discrepancies across environments. This brings us to consider two research questions:

RQ1 Does modeling the differences between environments improve generalization error w.r.t. classical settings: **One-For-All**, where a unique function is trained for all environments; and **One-Per-Env.**, where a specific function is fitted for each environment? (cf. Section 5.4 for more details)

RQ2 Is it possible to extrapolate to a novel environment that has not been seen during training?

We propose LEarning Across Dynamical Systems (LEADS), a novel learning methodology decomposing the learned dynamics into *shared* and *environment-specific* components. The learning problem is formulated such that the *shared* component captures the dynamics common across environments and exploits all the available data, while the *environment-specific* component only models the remaining dynamics, i.e., those that cannot be expressed by the former, based on environment-specific data. We show, under mild conditions, that the learning problem is well-posed, as the resulting decomposition exists and is unique (Section 5.2.2). We then analyze the properties of this decomposition from a sample complexity perspective. While, in general, the bounds might be too loose to be practical, a more precise study is conducted in the case of linear dynamics for which theory and practice are closer. We then instantiate this framework for more general hypothesis spaces and dynamics, leading to a heuristic for the control of generalization that will be validated experimentally. Overall, we show that this framework provides better generalization properties than One-Per-Env., requiring less training data to reach the same performance level (RQ1). The shared information is also useful to extrapolate to unknown environments: the new function for this environment can be learned from very

little data (RQ2). We experiment with these ideas on three representative cases (Section 5.4) where the dynamics are provided by differential equations: ODEs with the Lotka-Volterra predator-prey model, and PDEs with the Gray-Scott reaction-diffusion and the more challenging incompressible Navier-Stokes equations. Experimental evidence confirms the intuition and the theoretical findings: with a similar amount of data, the approach drastically outperforms One-For-All and One-Per-Env. settings, especially in low data regimes. To our knowledge, it is the first time that generalization in multiple dynamical systems is addressed from an ML perspective.

5.2 Approach

5.2.1 Problem Setting

We consider the problem of learning models of dynamical physical processes with data acquired from a set of environments $\mathcal{E}$. Throughout the paper, we will assume that the dynamics in an environment $e \in \mathcal{E}$ are defined through the evolution of differential equations. This will provide in particular a clear setup for the experiments and the validation. For a given problem, we consider that the dynamics of the different environments share common factors while each environment has its own specificity, resulting in a distinct model per environment. Both the general form of the differential equations and the specific terms of each environment are assumed to be completely unknown. u_t^e denotes the state of the equation for environment e, taking its values from a bounded set $\mathcal{U}$, with evolution term $f_e : \mathcal{U} \longrightarrow \mathrm{T}\mathcal{U}$, $\mathrm{T}\mathcal{U}$ being the tangent bundle of $\mathcal{U}$. In other words, over a fixed time interval $\mathcal{I} = [0, T]$, we have:

$$\frac{\mathrm{d}u^e}{\mathrm{d}t} = f_e(u^e) \tag{5.1}$$

We assume that, for any e, f_e lies in a functional vector space $\mathcal{F}$. In the experiments, we will consider one ordinary differential equation (ODE), in which case $\mathcal{U} \subset \mathbb{R}^d$, and two partial differential equations (PDEs), in which case $\mathcal{U}$ is a p-dimensional vector field over a bounded spatial domain $\Omega \subset \mathbb{R}^p$ discretized on a regular grid $\mathcal{X} \subset \Omega$. The term of the data-generating dynamical system in Eq. (5.1) is sampled from a distribution for each e, i.e. $f_e \sim Q$. From f_e, we define Γ_e, the distribution of time-continuous trajectories u^e verifying Eq. (5.1), induced by a distribution of initial states $u_0^e \sim \rho_0(\mathcal{U})$. The data for this environment is then composed of N trajectories sampled from Γ_e, they are also discretized

on a set of timestamps $\mathcal{T} \subset \mathcal{I} = [0, T]$, and is denoted as $\mathcal{D}_e = \{u^{e,(i)}|_{\mathcal{T}} \mid u_0^e \sim \rho_0(\mathcal{U}), u^e \in \Gamma^e\}_{i \in [\![1,N]\!]}$. We will denote the full dataset by $\mathcal{D} = \bigcup_{e \in \mathcal{E}} \mathcal{D}_e$.

The classical empirical risk minimization (ERM) framework suggests modeling the data dynamics either at the global level (One-For-All), taking trajectories indiscriminately from $\mathcal{D}$, or at the specific environment level (One-Per-Env.), training one model for each $\mathcal{D}_e$. Our aim is to formulate a new learning framework with the objective of explicitly considering the existence of different environments to improve the modeling strategy w.r.t. the classical ERM settings.

5.2.2 LEADS Framework

We decompose the dynamics into two components where $f \in \mathcal{F}$ is shared across environments and $g_e \in \mathcal{F}$ is specific to the environment e, so that

$$\forall e \in \mathcal{E}, \quad f_e = f + g_e \tag{5.2}$$

Since we consider functional vector spaces, this additive hypothesis is not restrictive and such a decomposition always exists. It is also quite natural as a sum of evolution terms can be seen as the sum of the forces acting on the system. Note that the sum of two evolution terms can lead to behaviors very different from those induced by each of those terms. However, learning this decomposition from data defines an ill-posed problem: for any choice of f, there is a $\{g_e\}_{e \in \mathcal{E}}$ such that Eq. (5.2) is verified. A trivial example would be $f = 0$ leading to a solution where each environment is fitted separately.

Our core idea is that f should capture as much of the shared dynamics as is possible, while g_e should focus only on the environment characteristics not captured by f. To formalize this intuition, we introduce $\mathcal{R}(g_e)$, a penalization on g_e, which precise definition will depend on the considered setting. We reformulate the learning objective as the following constrained optimization problem:

$$\begin{aligned} &\min_{f, \{g_e\}_{e \in E}} \quad \sum_{e \in \mathcal{E}} \mathcal{R}(g_e) \\ &\text{subject to} \quad \forall u^{e,(i)} \in \mathcal{D}, \forall t \in \mathcal{T}, \frac{\mathrm{d}u_t^{e,(i)}}{\mathrm{d}t} = (f + g_e)(u_t^{e,(i)}) \end{aligned} \tag{5.3}$$

Minimizing $\mathcal{R}$ aims to reduce g_es' complexity while correctly fitting the dynamics of each environment. This argument will be made formal in the next section. Note that f will

be trained on the data from all environments contrary to g_es. A key question is then to determine under which conditions the minimum in Eq. (5.3) is well-defined. The following proposition provides an answer:

Proposition 5.1 (Existence and Uniqueness). *Assume $\mathcal{R}$ is convex, then the existence of a minimal decomposition $f^\star, \{g_e^\star\}_{e\in\mathcal{E}} \in \mathcal{F}$ of Eq. (5.3) is guaranteed. Furthermore, if $\mathcal{R}$ is strictly convex, this decomposition is unique.*

☞ Proof in Appendix B.1, p. 197

In practice, we consider the following relaxed formulation of Eq. (5.3):

$$\min_{f,\{g_e\}_{e\in\mathcal{E}}} \sum_{e\in\mathcal{E}} \left(\frac{1}{\lambda}\mathcal{R}(g_e) + \sum_{i=1}^{N} \sum_{t\in\mathcal{T}} \left\| \frac{\mathrm{d}u_\tau^{e,(i)}}{\mathrm{d}t} - (f+g_e)(u_t^{e,(i)}) \right\|^2 \right) \tag{5.4}$$

where f, g_e are taken from a hypothesis space $\hat{\mathcal{F}}$ approximating $\mathcal{F}$. λ is a regularization weight and the integral term constrains the learned $f+g_e$ to follow the observed dynamics. The form of this objective and its effective calculation will be detailed in Section 5.4.4.

5.3 Improving Generalization with LEADS

Defining an appropriate $\mathcal{R}$ is crucial for our method. In this section, we show that the generalization error should decrease with the number of environments. While the bounds might be too loose for neural networks (NNs), our analysis is shown to adequately model the decreasing trend in the linear case, linking both our intuition and our theoretical analysis with empirical evidence. This then allows us to construct an appropriate $\mathcal{R}$ for NNs.

5.3.1 General Case

After introducing preliminary notations and definitions, we define the hypothesis spaces associated with our multiple environment framework. Considering a first setting where all environments of interest are present at training time, we prove an upper bound of their effective size based on the covering numbers of the approximation spaces. This allows us to quantitatively control the sample complexity of our model, depending on the number of environments m and other quantities that can be considered and optimized in practice. We then consider an extension for learning in a new and unseen environment. The bounds

here are inspired by ideas initially introduced in Baxter (2000). They consider multi-task classification in vector spaces, where the task-specific classifiers share a common feature extractor. Our extension considers sequences corresponding to dynamical trajectories, and a model with additive components instead of function composition in their case.

Definitions. Sample complexity theory is usually defined for supervised contexts, where for a given input z we want to predict some target y. In our setting, we want to learn trajectories $u^e\big|_{\mathcal{T}}$ starting from an initial condition u_0. We reformulate this problem and cast it as a standard supervised learning problem: Γ_e being the data distribution of trajectories for environment e, as defined in Section 5.2.1, let us consider a trajectory $u^e \sim \Gamma_e$, and time $t \in \mathcal{T} \sim \mathrm{Unif}([0,T])$; we define system states $z = u^e_t \in \mathcal{U}$ as input, and the corresponding values of derivatives $y = f_e(u^e_t) \in \mathrm{T}\mathcal{U}$ as the associated target. We will denote $\mathcal{P}_e$ the underlying distribution of (z,y), and $\hat{\mathcal{P}}_e$ the associated dataset of size n.

We are searching for $f, g_e : \mathcal{U} \to \mathrm{T}\mathcal{U}$ in an approximation function space $\hat{\mathcal{F}}$ of the original space $\mathcal{F}$. Let us define $\hat{\mathcal{G}} \subseteq \hat{\mathcal{F}}$ the effective function space from which the g_es are sampled. Let $f + \hat{\mathcal{G}} := \{f + g : g \in \hat{\mathcal{G}}\}$ be the hypothesis space generated by function pairs (f,g), with a fixed $f \in \hat{\mathcal{F}}$. For any $h : \mathcal{U} \to \mathrm{T}\mathcal{U}$, the error on some test distribution $\mathcal{P}_e$ is given by $\mathrm{er}_{\mathcal{P}_e}(h) = \int_{\mathcal{U}\times\mathrm{T}\mathcal{U}} \|h(z) - y\|^2 \mathrm{d}\mathcal{P}_e(z,y)$ and the error on the training set by $\hat{\mathrm{er}}_{\hat{\mathcal{P}}_e}(h) = \frac{1}{n}\sum_{(z,y)\in\hat{\mathcal{P}}_e} \|h(z) - y\|^2$.

LEADS sample complexity. Let $\mathcal{C}_{\hat{\mathcal{G}}}(\varepsilon, \hat{\mathcal{F}})$ and $\mathcal{C}_{\hat{\mathcal{F}}}(\varepsilon, \hat{\mathcal{G}})$ denote the capacity of $\hat{\mathcal{F}}$ and $\hat{\mathcal{G}}$ at a certain scale $\varepsilon > 0$. Such capacity describes the approximation ability of the space. The capacity of a class of functions is defined based on covering numbers, and the precise definition is provided in Appendix B.2.2, Table B.1. The following result is general and applies for *any* decomposition of the form $f + g_e$. It states that to guarantee a given average test error, the minimal number of samples required is a function of both capacities and the number of environments m, and it provides a step towards RQ1:

Proposition 5.2. *Given m environments, let $\varepsilon_1, \varepsilon_2, \delta > 0, \varepsilon = \varepsilon_1 + \varepsilon_2$. Assume the number of examples n per environment satisfies*

$$n \geq \max\left\{\frac{64}{\varepsilon^2}\left(\frac{1}{m}\left(\log\frac{4}{\delta} + \log\mathcal{C}_{\hat{\mathcal{G}}}\left(\frac{\varepsilon_1}{16}, \hat{\mathcal{F}}\right)\right) + \log\mathcal{C}_{\hat{\mathcal{F}}}\left(\frac{\varepsilon_2}{16}, \hat{\mathcal{G}}\right)\right), \frac{16}{\varepsilon^2}\right\} \tag{5.5}$$

Then with probability at least $1 - \delta$ (over the choice of training sets $\{\hat{\mathcal{P}}_e\}$), any learner

$(f + g_1, \ldots, f + g_m)$ will satisfy $\frac{1}{m}\sum_{e\in\mathcal{E}} \mathrm{er}_{\mathcal{P}_e}(f + g_e) \leq \frac{1}{m}\sum_{e\in\mathcal{E}} \hat{\mathrm{er}}_{\hat{\mathcal{P}}_e}(f + g_e) + \varepsilon$.

☞ Proof in Appendix B.2.2, p. 200

The contribution of $\hat{\mathcal{F}}$ to the sample complexity decreases as m increases, while that of $\hat{\mathcal{G}}$ remains the same: this is due to the fact that shared functions f have access to the data from all environments, which is not the case for g_e. From this finding, one infers the basis of LEADS: when learning from several environments, to control the generalization error through the decomposition $f_e = f + g_e$, *f should account for most of the complexity of f_e while the complexity of g_e should be controlled and minimized*. We then establish an explicit link to our learning problem formulation in Eq. (5.3). Further in this section, we will show for linear ODEs that the optimization of $\mathcal{R}(g_e)$ in Eq. (5.4) controls the capacity of the effective set $\hat{\mathcal{G}}$ by selecting g_es that are as "simple" as possible.

As a corollary, we show that for a fixed total number of samples in $\hat{\mathcal{D}}$, the sample complexity will decrease as the number of environments increases. To see this, suppose that we have two situations corresponding to data generated respectively from m and m/b environments. The total sample complexity for each case will be respectively bounded by $O(\log \mathcal{C}_{\hat{\mathcal{G}}}(\frac{\varepsilon_1}{16}, \hat{\mathcal{F}}) + m \log \mathcal{C}_{\hat{\mathcal{F}}}(\frac{\varepsilon_2}{16}, \hat{\mathcal{G}}))$ and $O(b \log \mathcal{C}_{\hat{\mathcal{G}}}(\frac{\varepsilon_1}{16}, \hat{\mathcal{F}}) + m \log \mathcal{C}_{\hat{\mathcal{F}}}(\frac{\varepsilon_2}{16}, \hat{\mathcal{G}}))$. The latter being larger than the former, a situation with more environments presents a clear advantage. Figure 5.4 in Section 5.4 confirms this result with empirical evidence.

LEADS sample complexity for novel environments. Suppose that problem Eq. (5.3) has been solved for a set of environments $\mathcal{E}$, can we use the learned model for a new environment not present in the initial training set (RQ2)? Let e' be such a new environment, $\mathcal{P}_{e'}$ the trajectory distribution of e', generated from dynamics $f_{e'} \sim Q$, and $\hat{\mathcal{P}}_{e'}$ an associated training set of size n'. The following results show that the number of required examples for reaching a given performance is much lower when training $f + g_{e'}$ with f fixed on this new environment than training another $f' + g_{e'}$ from scratch.

Proposition 5.3. *For all ε, δ with $0 < \varepsilon, \delta < 1$ if the number of samples n' satisfies*

$$n' \geq \max\left\{\frac{64}{\varepsilon^2}\log\frac{4\mathcal{C}(\frac{\varepsilon}{16}, f + \hat{\mathcal{G}})}{\delta}, \frac{16}{\varepsilon^2}\right\}, \tag{5.6}$$

then with probability at least $1 - \delta$ (over the choice of novel training set $\hat{\mathcal{P}}_{e'}$), any learner $f + g_{e'} \in f + \hat{\mathcal{G}}$ will satisfy $\mathrm{er}_{\mathcal{P}_{e'}}(f + g_{e'}) \leq \hat{\mathrm{er}}_{\hat{\mathcal{P}}_{e'}}(f + g_{e'}) + \varepsilon$.

☞ Proof in Appendix B.2.2, p. 200

In Proposition 5.3 as the capacity of $\hat{\mathcal{F}}$ no longer appears, the number of required samples now depends only on the capacity of $f + \hat{\mathcal{G}}$. This sample complexity is then smaller than learning from scratch $f_{e'} = f + g_{e'}$ as can be seen by comparing with Proposition 5.2 at $m = 1$.

From the previous propositions, it is clear that the environment-specific functions g_e need to be explicitly controlled. We now introduce a practical way to do that. Let $\omega(r, \varepsilon)$ be a strictly increasing function w.r.t. r such that

$$\log \mathcal{C}_{\hat{\mathcal{F}}}(\varepsilon, \hat{\mathcal{G}}) \leq \omega(r, \varepsilon), \quad r = \sup_{g \in \hat{\mathcal{G}}} \mathcal{R}(g) \tag{5.7}$$

Minimizing $\mathcal{R}$ would reduce r and then the sample complexity of our model by constraining $\hat{\mathcal{G}}$. Our goal is thus to construct such a pair $(\omega, \mathcal{R})$. In the following, we will first show in Section 5.3.2, how one can construct a penalization term $\mathcal{R}$ based on the covering number bound for linear approximators and linear ODEs. We show with a simple use case that the generalization error obtained in practice follows the same trend as the theoretical error bound when the number of environments varies. Inspired by this result, we then propose in Section 5.3.3 an effective $\mathcal{R}$ to penalize the complexity of the neural networks g_e.

5.3.2 Linear Case: Theoretical Bounds Correctly Predict the Trend of Test Error

Results in Section 5.3.1 provide general guidelines for our approach. We now apply them to a linear system to see how the empirical results meet the tendency predicted by the theoretical bound.

Let us consider a linear ODE $\frac{\mathrm{d}u^e}{\mathrm{d}t} = \mathrm{L}_{\boldsymbol{F}_e}(u^e)$ where $\mathrm{L}_{\boldsymbol{F}_e} : u_t \mapsto \boldsymbol{F}_e u_t$ is a linear transformation associated to the square real-valued matrix $\boldsymbol{F}_e \in M_{d,d}(\mathbb{R})$. We choose as hypothesis space the space of linear functions $\hat{\mathcal{F}} \subset \mathcal{L}(\mathbb{R}^d, \mathbb{R}^d)$ and instantiate a linear LEADS $\frac{\mathrm{d}u^e}{\mathrm{d}t} = (\mathrm{L}_{\boldsymbol{F}} + \mathrm{L}_{\boldsymbol{G}_e})(u^e)$, $\mathrm{L}_{\boldsymbol{F}} \in \hat{\mathcal{F}}, \mathrm{L}_{\boldsymbol{G}_e} \in \hat{\mathcal{G}} \subseteq \hat{\mathcal{F}}$. As suggested in Bartlett et al. (2017), we have that:

Proposition 5.4. *If for all linear maps* $\mathrm{L}_{\boldsymbol{G}_e} \in \hat{\mathcal{G}}$, $\|\boldsymbol{G}\|_F^2 \leq r$, *if the input space is bounded*

s.t. $\|u\|_2 \leq b$, *and the mean square error (MSE) loss function is bounded by* c, *then*

$$\log \mathcal{C}_{\hat{\mathcal{F}}}(\varepsilon, \hat{\mathcal{G}}) \leq \left\lceil \frac{rcd(2b)^2}{\varepsilon^2} \right\rceil \log 2d^2 =: \omega(r, \varepsilon) \tag{5.8}$$

☞ Proof in Appendix B.2.3, p. 203

$\omega(r, \varepsilon)$ is a strictly increasing function w.r.t. r. This indicates that we can choose $\mathcal{R}(\mathrm{L}_{\boldsymbol{G}}) = \|\boldsymbol{G}\|_F$ as our optimization objective in Eq. (5.3). The sample complexity in Eq. (5.5) will decrease with the size the largest possible $r = \sup_{\mathrm{L}_{\boldsymbol{G}} \in \hat{\mathcal{G}}} \mathcal{R}(\mathrm{L}_{\boldsymbol{G}})$. The optimization process will reduce $\mathcal{R}(\mathrm{L}_{\boldsymbol{G}})$ until a minimum is reached. The maximum size of the effective hypothesis space is then bounded and decreases throughout training thanks to the penalty. Then in linear case Proposition 5.2 becomes:

Proposition 5.5. *If for linear maps* $\mathrm{L}_{\boldsymbol{F}} \in \hat{\mathcal{F}}$, $\|\boldsymbol{F}\|_F^2 \leq r'$, $\mathrm{L}_{\boldsymbol{G}} \in \hat{\mathcal{G}}$, $\|\boldsymbol{G}\|_F^2 \leq r$, $\|u\|_2 \leq b$, *and if the MSE loss function is bounded by* c, *given* m *environments and* n *samples per environment, with the probability* $1 - \delta$, *the generalization error upper bound is*

$$\varepsilon = \max\left\{ \sqrt{\frac{(p + \sqrt{p^2 + 4q})}{2}}, \sqrt{\frac{16}{n}} \right\} \tag{5.9}$$

where $p = \frac{64}{mn} \log \frac{4}{\delta}$ *and* $q = \frac{64}{n} \left\lceil \left(\frac{r'}{ml^2} + \frac{r}{(1-l)^2} \right) cd(32b)^2 \right\rceil \log 2d^2$ *for any* $0 < l < 1$.

☞ Proof in Appendix B.2.3, p. 203

In Figure 5.1, we take an instance of linear ODE defined by $\boldsymbol{F}_e = \boldsymbol{Q}\boldsymbol{\Lambda}_e\boldsymbol{Q}^\top$ with the diagonal $\boldsymbol{\Lambda}_e$ specific to each environment. After solving Eq. (5.3) we have at the optimum that $\boldsymbol{G}_e = \boldsymbol{F}_e - \boldsymbol{F}^\star = \boldsymbol{F}_e - \frac{1}{m} \sum_{e' \in \mathcal{E}} \boldsymbol{F}_{e'}$. Then we can take $r = \max_{\{\mathrm{L}_{\boldsymbol{G}_e}\}} \mathcal{R}(\mathrm{L}_{\boldsymbol{G}_e})$ as the norm bound of $\hat{\mathcal{G}}$ when $\mathcal{R}(g_e)$ is optimized. Figure 5.1 shows on the left the test error with and without penalty and the corresponding theoretical bound on the right. We observe that, after applying the penalty $\mathcal{R}$, the test error is reduced as well as the theoretical generalization bound, as indicated by the arrows from the dashed line to the concrete one. See Appendix B.2.3 for more details on this experiment.

5.3.3 Nonlinear Case: Instantiation for Neural Networks

The above linear case validates the ideas introduced in Proposition 5.2 and provides an instantiation guide and an intuition on the more complex nonlinear case. This motivates

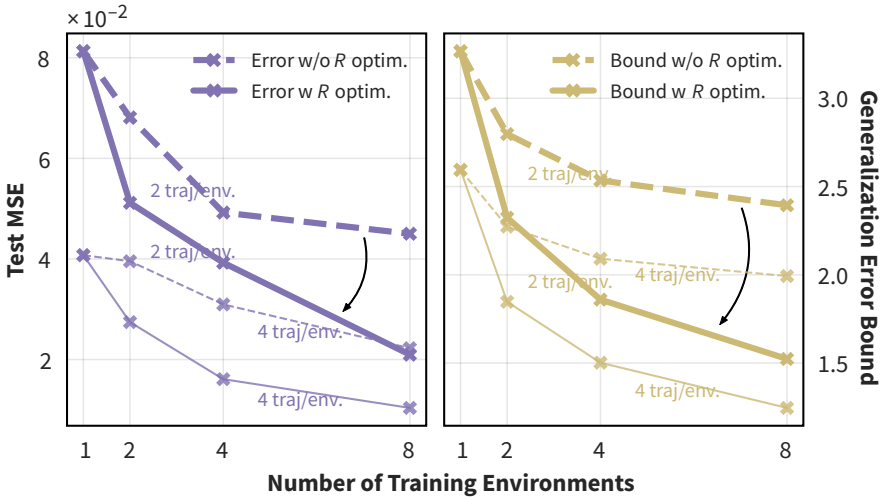


Figure 5.1: Test error compared with the corresponding theoretical bound. The arrows indicate the changes after applying $\mathcal{R}(g_e)$ penalty.

us to instantiate the general case by choosing an appropriate approximating space $\hat{\mathcal{F}}$ and a penalization function $\mathcal{R}$ from the generalization bounds for the corresponding space. Appendix B.2.4 of the Appendix contains additional details justifying those choices. For $\hat{\mathcal{F}}$, we select the space of feed-forward neural networks with a fixed architecture. We choose the following penalty function:

$$\mathcal{R}(g_e) = \|g_e\|_\infty^2 + \alpha \|g_e\|_{\text{Lip}}^2 \tag{5.10}$$

where $\|g\|_\infty = \operatorname{ess\,sup} |g|$ and $\|\cdot\|_{\text{Lip}}$ is the Lipschitz semi-norm, α is a hyperparameter. This is inspired by the existing capacity bound for NNs (Haussler, 1992) (see Appendix B.2.4 for details). Note that constructing tight generalization bounds for neural networks is still an open research problem (Nagarajan and Kolter, 2019); however, it may still yield valuable intuitions and guide algorithm design. This heuristic is tested successfully on three different datasets with different architectures in the experiments (Section 5.4).

5.4 Experiments

Our experiments are conducted on three families of dynamical systems described by three broad classes of differential equations. All exhibit complex and nonlinear dynamics. The

first one is an ODE-driven system used for biological system modeling. The second one is a PDE-driven reaction-diffusion model, well-known in chemistry for its variety of spatiotemporal patterns. The third one is the more physically complex Navier-Stokes equation, expressing the physical laws of incompressible Newtonian fluids. To show the general validity of our framework, we will use 3 different NN architectures: multi-layer perceptron (MLP), convolutional neural network (ConvNet), and Fourier neural operator (FNO; Li et al., 2021b). Each architecture is well-adapted to the corresponding dynamics. This also shows that the framework is valid for a variety of approximating functions.

5.4.1 Dynamics, Environments, and Datasets

Lotka-Volterra. This classical model (Lotka, 1925) is used for describing the dynamics of interaction between a predator and a prey. The dynamics follow the ODE:

$$\begin{aligned} \frac{\mathrm{d}x}{\mathrm{d}t} &= \alpha x - \beta x y, \\ \frac{\mathrm{d}y}{\mathrm{d}t} &= \delta x y - \gamma y \end{aligned} \tag{5.11}$$

with $x(t), y(t)$ the number of prey and predator, $\alpha, \beta, \gamma, \delta > 0$ defining how the two species interact. The system state is $u^e(t) = (x^e(t), y^e(t)) \in \mathbb{R}^2_+$. The initial conditions u^i_0, v^i_0 are sampled from a uniform distribution $\rho_0(\mathcal{U})$. We characterize the dynamics by $\theta_{\text{P}} = (\alpha/\beta, \gamma/\delta) \in \Theta_{\text{P}}$. An environment e is then defined by parameters θ^e_{P} sampled from a uniform distribution over a parameter set Θ_{P}. We then sample two sets of environment parameters: one used as training environments for RQ1, the other treated as novel environments for RQ2.

Gray-Scott. This reaction-diffusion model is famous for its complex spatiotemporal behavior given its simple equation formulation (Pearson, 1993). The governing PDE is:

$$\begin{aligned} \frac{\partial v}{\partial t} &= D_v \Delta v - v w^2 + F_r(1 - v) \\ \frac{\partial w}{\partial t} &= D_w \Delta w + v w^2 - (F_r + k_r) w \end{aligned} \tag{5.12}$$

where the v, w represent the concentrations of two chemical components in the spatial domain Ω with periodic boundary conditions, the state at time t is $u^e_t = (v^e_t, w^e_t) \in \mathbb{R}^{2\times 32\times 32}_+$, spatially discretized on a 32-by-32 regular grid. D_v, D_w denote the diffusion coeffi-

cients respectively for v, w, and are held constant, and F_r, k_r are the reaction parameters determining the spatiotemporal patterns of the dynamics (Pearson, 1993). As for the initial conditions $(v_0, w_0) \sim \rho_0(\mathcal{U})$, we consider uniform concentrations, with three 2-by-2 squares fixed at other concentration values and positioned at uniformly sampled positions in Ω to trigger the reactions. An environment e is defined by its parameters $\theta_e = (F_{r,e}, k_{r,e}) \in \Theta_{\text{P}}$. We consider a set of θ_e parameters uniformly sampled from the environment distribution Q on Θ_{P}.

Navier-Stokes. We consider the Navier-Stokes PDE for incompressible flows:

$$\frac{\partial w}{\partial t} = -v \cdot \nabla w + \nu \Delta w + h \qquad \nabla \cdot v = 0 \tag{5.13}$$

where v is the velocity field, $w = \nabla \times v$ is the vorticity, both v, w lie in a spatial domain Ω with periodic boundary conditions, ν is the viscosity and h is the constant forcing term in the domain Ω. The discretized state on a regular grid $\mathcal{X}$ at time t is the vorticity $u_t^e = w_t^e\big|_{\mathcal{X}} \in \mathbb{R}^{32\times 32}$. Note that v is already contained in w and can be recovered from it. We fix $\nu = 10^{-3}$ across the environments. We sample the initial conditions $w_0^e \sim \rho_0(\mathcal{U})$ as in Li et al. (2021b). An environment e is defined by its forcing term $h_e \in \Theta_{\text{P}}$. We uniformly sampled a set of forcing terms from Q on Θ_{P}.

Datasets. For training, we create two datasets for *Lotka-Volterra* by simulating trajectories of $K = 20$ successive points with temporal resolution $\delta t = 0.5$. We use the first one as a set of training dynamics to validate the LEADS framework. We choose 10 environments and simulate $N_{\text{tr}} = 8$ trajectories, i.e., $n = 8 \cdot K$ states, per environment for training. We can then easily control the number of data points and environments in experiments by taking different subsets. The second one is used to validate the improvement with LEADS while training in novel environments. We simulate $N_{\text{tr}} = 1$ trajectory ($n = 1 \cdot K$ states) for training. We create two datasets for further validation of LEADS with *Gray-Scott* and *Navier-Stokes*. For *Gray-Scott*, we simulate trajectories of $K = 10$ steps with $\delta t = 40$. We choose 3 parameters and simulate $N_{\text{tr}} = 1$ trajectory ($n = 1 \cdot K$ states) for training. For *Navier-Stokes*, we simulate trajectories of $K = 10$ steps with $\delta t = 1$. We choose 4 forcing terms and simulate $N_{\text{tr}} = 8$ trajectories ($n = 8 \cdot K$ states) for training. For test-time evaluation, we create for each equation in each environment a test set of $N_{\text{ts}} = 32$ trajectories ($32 \cdot K$) data points. Note that every environment dataset has the same number of trajectories and the initial conditions are fixed to equal values across the environments to ensure that the data variations only come from the dynamics themselves, i.e., for the i-th

trajectory in $\hat{\mathcal{P}}_e$, $\forall e, u_0^{e,(i)} = u_0^{(i)}$. *Lotka-Volterra* and *Gray-Scott* data are simulated with the DOPRI5 solver in NumPy (Dormand and Prince, 1980; Harris et al., 2020). *Navier-Stokes* data is simulated with the pseudo-spectral method as in Li et al. (2021b).

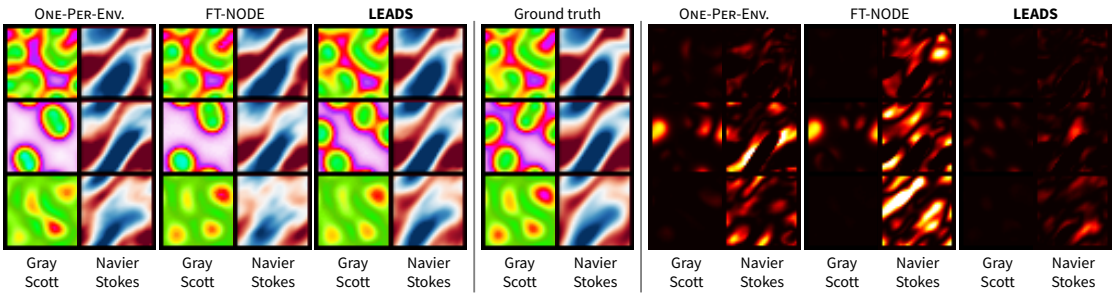


Figure 5.2: Left: final states for *Gray-Scott* and *Navier-Stokes* predicted by the two best baselines (One-Per-Env. and FT-NODE) and LEADS compared with ground truth. Different environments are arranged by row (3 in total). Right: the corresponding MAE error maps, the scale of the error map is $[0, 0.6]$ for *Gray-Scott*, and $[0, 0.2]$ for *Navier-Stokes*; darker is smaller. (See Appendix B.4 for full sequences)

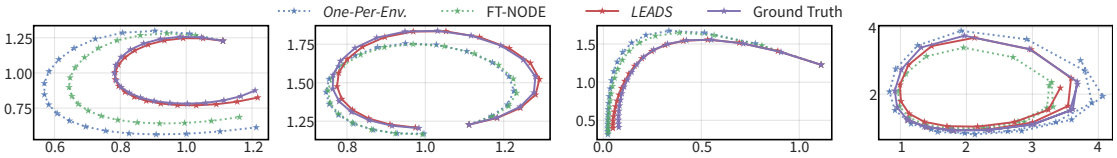


Figure 5.3: Test predicted *Lotka-Volterra* trajectories in phase space with two baselines (One-Per-Env. and FT-NODE) and LEADS compared with ground truth for 4 environments, one per figure from left to right. Quantity of the prey v and the predator w respectively on the horizontal and the vertical axis. The initial state is the rightmost endpoint of the figures and it is common to all the trajectories.

5.4.2 Experimental Settings and Baselines

We validate LEADS in two settings: in the first one all the environments in $\mathcal{E}$ are available at once and then f and all the g_es are all trained on $\mathcal{E}$. In the second one, training has been performed on $\mathcal{E}$ as before, and we consider a novel environment $e' \notin \mathcal{E}$: the shared term f being kept fixed, the approximating function $f_{e'} = f + g_{e'}$ is trained on the data from e' (i.e., only $g_{e'}$ is modified).

All environments available at once. We introduce five baselines used for comparing with LEADS:

(a) **One-For-All**: learning on the entire dataset $\hat{\mathcal{P}}$ over all environments with the sum of a pair of NNs $f + g$, with the standard ERM principle, as in Ayed et al. (2022). Although this is equivalent to using only one function f, we use this formulation to indicate that the number of parameters is the same for this experiment and for the LEADS ones.

(b) **One-Per-Env.**: learning a specific function for each dataset $\hat{\mathcal{P}}_e$. For the same reason as above, we keep the sum formulation $(f + g)_e$.

(c) Factored Tensor RNN or **FT-RNN** (Spieckermann et al., 2015): it modifies the recurrent neural network to integrate a one-hot environment code into each linear transformation of the network. Instead of being encoded in a separate function g_e like in LEADS, the environment appears here as an extra one-hot input for the RNN linear transformations. This can be implemented for representative SOTA (spatio-)temporal predictors such as GRU (Cho et al., 2014) or PredRNN (Wang et al., 2017).

(d) **FT-NODE**: a baseline for which the same environment encoding as FT-RNN is incorporated in a neural ordinary differential equation (Neural ODE; Chen et al., 2018).

(e) Gradient-based meta learning (GBML) like method: we propose a GBML-like baseline which can directly compare to our framework. It follows the principle of model-agnostic meta learning (MAML; Finn et al., 2017), by training One-For-All at first which provides an initialization near the given environments like GBML does, then fitting it individually for each training environment.

(f) **LEADS no min.**: ablation baseline, our proposal without the $\mathcal{R}(g_e)$ penalization.

A comparison with the different baselines is proposed in Table 5.1 for the three dynamics. For concision, we provide a selection of results corresponding to $N_{\text{tr}} = 1$ training trajectory per environment for *Lotka-Volterra* and *Gray-Scott* and $N_{\text{tr}} = 8$ for *Navier-Stokes*. This is the minimal training set size for each dataset. Further experimental results when varying the number of environments from $N_{\text{tr}} = 1$ to $N_{\text{tr}} = 8$ are provided in Figure 5.4 and Table B.3 for *Lotka-Volterra*.

Learning on novel environments. We consider the following training schemes with a pre-trained, fixed f:

(a) **Pre-trained-f-Only**: only the pre-trained f is used for prediction; a sanity

check to ensure that f cannot predict in any novel environment without further adaptation.

(b) **One-Per-Env.**: training from scratch on $\{\hat{\mathcal{P}}_{e'}\}$ as One-Per-Env. in the previous section.

(c) **Pre-trained-f-Plus-Trained-g_e**: we train g on each dataset $\hat{\mathcal{P}}_{e'}$ based on pre-trained f, i.e., $f + g_{e'}$, leaving only $g_{e'}$s adjustable.

We compare the test error evolution during training for the three schemes above for a comparison of convergence speed and performance. Results are given in Figure 5.5.

5.4.3 Experimental Results

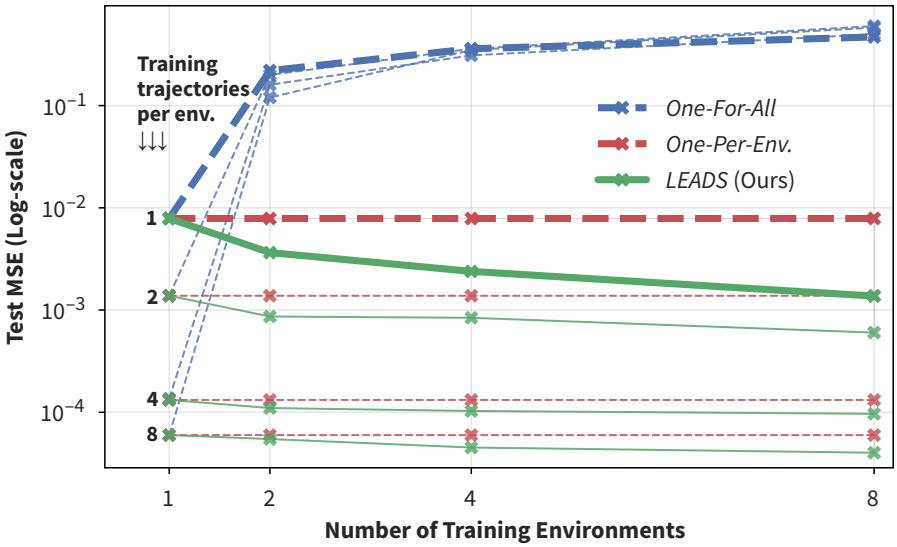


Figure 5.4: Test error for *Lotka-Volterra* w.r.t. the number of environments. We apply the models in 1 to 8 environments. 4 groups of curves correspond to models trained with 1 to 8 trajectories per environment. All groups highlight the same tendencies: increasing One-For-All, stable One-Per-Env., and decreasing LEADS. More results of baselines methods in Appendix B.4.

All environments available at once. We show the results in Table 5.1. For *Lotka-Volterra* systems, we confirm first that the entire dataset cannot be learned properly with a single model (One-For-All) when the number of environments increases. Compared with other baselines, our method LEADS reduces the test MSE over 85% w.r.t. One-Per-Env. and over 60% w.r.t. LEADS no min., we also cut 50%-75% of error w.r.t. other baselines.

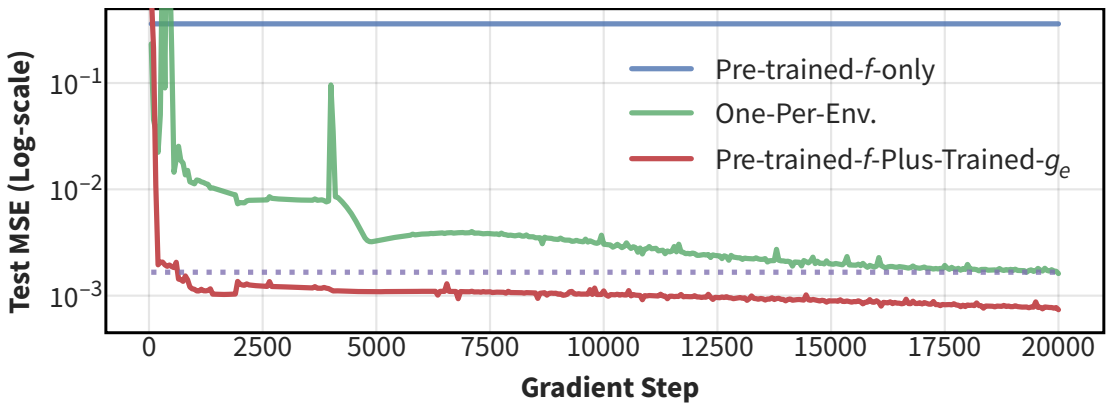


Figure 5.5: Test error evolution during training on 2 novel environments for *Lotka-Volterra*.

Table 5.1: Results for *Lotka-Volterra*, *Gray-Scott*, and *Navier-Stokes* datasets, trained on m environments with n data points per environment.

Method	Lotka-Volterra ($m = 10, n = 1 \cdot K$)		Gray-Scott ($m = 3, n = 1 \cdot K$)		Navier-Stokes ($m = 4, n = 8 \cdot K$)	
	MSE train	MSE test	MSE train	MSE test	MSE train	MSE test
One-For-All	4.57e−1	5.08±0.56e−1	1.55e−2	1.43±0.15e−2	5.17e−2	7.31±5.29e−2
One-Per-Env.	2.15e−5	7.95±6.96e−3	8.48e−5	6.43±3.42e−3	5.60e−6	1.10±0.72e−2
FT-RNN	5.29e−5	6.40±5.69e−3	8.44e−6	8.19±3.09e−3	7.40e−4	5.92±4.00e−2
FT-NODE	7.74e−5	3.40±2.64e−3	3.51e−5	3.86±3.36e−3	1.80e−4	2.96±1.99e−2
GBML-like	3.84e−6	5.87±5.65e−3	1.07e−4	6.01±3.62e−3	1.39e−4	7.37±4.80e−3
LEADS no min.	3.28e−6	3.07±2.58e−3	7.65e−5	5.53±3.43e−3	3.20e−4	7.10±4.24e−3
LEADS (Ours)	5.74e−6	**1.16±0.99e−3**	5.75e−5	**2.08±2.88e−3**	1.03e−4	**5.95±3.65e−3**

Figure 5.3 shows samples of predicted trajectories in test, LEADS follows very closely the ground truth trajectory, while One-Per-Env. under-performs in most environments. We observe the same tendency for the *Gray-Scott* and *Navier-Stokes* systems. The error is reduced by: around ⅔ (*Gray-Scott*) and 45% (*Navier-Stokes*) w.r.t. One-Per-Env.; over 60% (*Gray-Scott*) and 15% (*Navier-Stokes*) w.r.t. LEADS no min.; 45-75% (*Gray-Scott*) and 15-90% (*Navier-Stokes*) w.r.t. other baselines. In Figure 5.2, the final states obtained with LEADS are qualitatively closer to the ground truth. Looking at the error maps on the right, we see that the errors are systematically reduced across all environments compared to the baselines. This shows that LEADS accumulates fewer errors through the integration, which suggests that LEADS alleviates overfitting.

We have also conducted a larger scale experiment on *Lotka-Volterra* (Figure 5.4) to analyze the behavior of the different training approaches as the number of environments increases. We consider three models One-For-All, One-Per-Env. and LEADS, 1, 2, 4 and 8 environments, and for each such case, we have 4 groups of curves, corresponding to $N_{\text{tr}} = 1, 2, 4, 8$ training trajectories per environment. We summarize the main observations. With One-For-All (blue), the error increases as the number of environments increases: the dynamics for each environment being indeed different, this introduces an increasingly large bias, and thus the data cannot be fitted with one single model. The performance of One-Per-Env. (in red), for which models are trained independently for each environment, is constant as expected when the number of environments changes. LEADS (green) circumvents these issues and shows that the shared characteristics among the environments can be leveraged so as to improve generalization: it is particularly effective when the number of samples per environment is small. (See Appendix B.4 for more details on the experiments and on the results).

Learning on novel environments. We demonstrate how the pre-trained dynamics can help to fit a model for novel environments. We took an f pre-trained by LEADS on a set of *Lotka-Volterra* environments. Figure 5.5 shows the evolution of the test loss during training for three systems: a f function pre-trained by LEADS on a set of *Lotka-Volterra* training environments, a g_e function trained from scratch on the new environment and LEADS that uses a pre-trained f and learns a g_e residue on this new environment. Pre-trained-f-Only alone cannot predict in any novel environment. Very fast in the training stages, Pre-trained-f-Plus-Trained-g_e already surpasses the best error of the model trained from scratch (indicated with a dotted line). Similar results are also observed with the *Gray-Scott* and *Navier-Stokes* datasets (cf. Appendix B.4, Table B.5). These empirical

results clearly show that the learned shared dynamics accelerate and improve learning in novel environments.

5.4.4 Training and Implementation Details

Discussion on trajectory-based optimization. Solving the learning problem Eq. (5.2) in our setting, involves computing a trajectory loss (integral term in Eq. (5.4)). However, in practice, we do not have access to the continuous trajectories at every instant t but only to a finite number of snapshots for the state values $u|_{\mathcal{T}}$ on the time grid $\mathcal{T} = \{k\delta t\}_{k\in[\![0,K=T/\delta t]\!]}$ with the fixed time step δ_t. From these observed discrete trajectories, it is still possible to recover an approximate derivative $d_t^{\Lambda} \simeq \frac{\mathrm{d}u}{\mathrm{d}t}(t)$ using a numerical scheme Λ. The integral term for a given sample in the objective Eq. (5.4) would then be estimated as $\sum_{t\in\mathcal{T}} \left\| d_t^S - (f + g_e)(u_t) \right\|^2$. This is not the best solution and we have observed much better prediction performance for all models, including the baselines, when computing the error directly on the states, using an integral formulation $\sum_{t\in\mathcal{T}} \|u_t - \tilde{u}_t\|^2$, where $\tilde{u}_t$ is the solution given by a numerical solver approximating the integral $u_{t-\delta t} + \int_{s=t-\delta t}^{s=t} (f + g_e)(\tilde{u}_s)\mathrm{d}s$ starting from $u_{t-\delta t}$. Comparing directly in the state space yields more accurate results for prediction as the learned network tends to correct the solver's numerical errors, as first highlighted in Yin et al. (2021b).

Calculating $\mathcal{R}$. Given finite data and time, the exact infinity norm and Lipschitz norm are both intractable. We opt for more practical forms in the experiments. For the infinity norm, we chose to minimize the empirical norm of the output vectors on known data points, this choice is motivated in Appendix B.3. In practice, we found out that dividing the output norm by its input norm works better: $\frac{1}{n}\sum_{i,t} \|g_e(u_t^{e,(i)})\|^2 / \|u_t^{e,(i)}\|^2$, where the $u_t^{e,i}$ are known states in the training set. For the Lipschitz norm, as suggested in Bietti et al. (2019), we optimize the sum of the spectral norms of the weight at each layer $\sum_{l=1}^{D} \|W_l^{g_e}\|^2$. We use the power iteration method in Miyato et al. (2018) for fast spectral norm approximation.

Implementation. We used 4-layer MLPs for *Lotka-Volterra*, 4-layer ConvNets for *Gray-Scott* and FNO for *Navier-Stokes*. For FT-RNN baseline, we adapted gated recurrent unit (GRU; Cho et al., 2014) for *Lotka-Volterra* and PredRNN (Wang et al., 2017) for *Gray-Scott* and *Navier-Stokes*. We apply the Swish function (Ramachandran et al., 2017) as the default activation function. Networks are integrated in time with RK4 (*Lotka-Volterra*, *Gray-Scott*) or Euler (*Navier-Stokes*), using the basic back-propagation through the internals of the solver. We apply an exponential Scheduled Sampling (Bengio et al., 2015) with an exponent

of 0.99 to stabilize the training. We use the Adam optimizer (Kingma and Ba, 2015) with the same learning rate 1×10^{-3} and $(\beta_1, \beta_2) = (0.9, 0.999)$ across the experiments. For the hyperparamters in Eq. (5.10), we chose respectively $\lambda = 5\times10^{3}, 1\times10^{2}, 1\times10^{5}$ and $\alpha = 1\times10^{-3}, 1\times10^{-2}, 1\times10^{-5}$ for *Lotka-Volterra*, *Gray-Scott* and *Navier-Stokes*. All experiments are performed with a single NVIDIA Titan Xp GPU.

5.5 Discussions

Limitations Our framework is generic and could be used in many different contexts. On the theoretical side, the existence and uniqueness properties (Proposition 5.1) rely on relatively mild conditions covering a large number of situations. The complexity analysis, on the other side, is only practically relevant for simple hypothesis spaces (here linear), and then serves for developing the intuition on more complex spaces (NNs here) where bounds are too loose to be informative. Another limitation is that the theory and experiments consider deterministic systems only: the experimental validation is performed on simulated deterministic data. Note however that this is the case in the vast majority of the ML literature on ODE/PDE spatiotemporal modeling (Raissi et al., 2019; Long et al., 2019; Li et al., 2021b; Yin et al., 2021b). In addition, modeling complex dynamics from real-world data is a problem in itself.

Conclusion We introduce LEADS, a data-driven framework to learn dynamics from data collected from a set of distinct dynamical systems with commonalities. Experimentally validated with three families of equations, our framework can significantly improve the test performance in every environment w.r.t. classical training, especially when the number of available trajectories is limited. We further show that the dynamics extracted by LEADS can boost learning in similar new environments, which gives us a flexible framework for generalization in novel environments. More generally, we believe that this method is a promising step towards addressing the generalization problem for learning dynamical systems and has the potential to be applied to a large variety of problems.

Acknowledgements

We acknowledge financial support from the ANR AI Chairs program DL4CLIM ANR-19-CHIA-0018-01.

Chapter 6

Learning to Adapt to Unknown Systems

Generalizing to New Physical Systems via Context-Informed Dynamics Model

In this chapter, following the previous promising results on adaptation, we turn our focus on how to make the adaptation simpler, more efficient, and potentially more interpretable. We propose a model that achieves these goals through a contextual decoder-only model. This work resulted in a conference paper at ICML 2022.

> Matthieu Kirchmeyer*, **Yuan Yin***, Jérémie Donà, Nicolas Baskiotis, Alain Rakotomamonjy, and Patrick Gallinari. Generalizing to new physical systems via context-informed dynamics model. ICML 2022.

6.1 Introduction

Neural network (NN) approaches to modeling dynamical systems have recently raised the interest of several communities leading to an increasing number of contributions. This topic was explored in several domains, ranging from simple dynamics, e.g., Hamiltonian systems (Greydanus et al., 2019; Chen et al., 2020b) to more complex settings, e.g., fluid dynamics (Kochkov et al., 2021; Li et al., 2021b; Wandel et al., 2021), earth system science and climate (Reichstein et al., 2019), or health (Fresca et al., 2020). NN emulators are attractive as they may for example provide fast and low cost approximations to complex numerical simulations (Duraisamy et al., 2019; Kochkov et al., 2021), complement existing simulation models when the physical law is partially known (Yin et al., 2021b) or even offer solutions when classical solvers fail, e.g., with very high number of variables (Sirignano and Spiliopoulos, 2018).

A model of a real-world dynamical system should account for a wide range of contexts resulting from different external forces, spatiotemporal conditions, boundary conditions, sensors characteristics, or system parameters. These contexts characterize the dynamics phenomenon. For instance, in cardiac electrophysiology (Neic et al., 2017; Fresca et al., 2020), each patient has its own specificities and represents a particular context. In the study of epidemics' diffusion (Shaier et al., 2022), computational models should handle a variety of spatial, temporal or even sociological contexts. The same holds for most physical problems, e.g., forecasting of spatial-location-dependent dynamics in climate (de Bézenac et al., 2018), fluid dynamics prediction under distinct external forces (Li et al., 2021b), etc.

The physics approach for modeling dynamical systems relies on a strong prior knowledge about the underlying phenomenon. This provides a causal mechanism which is embedded in a physical dynamics model, usually a system of differential equations, and allows the physical model to handle a whole set of contexts. Moreover, it is often possible to adapt the model to new or evolving situations, e.g., via data assimilation (Kalman, 1960; Courtier et al., 1994).

In contrast, empirical risk minimization (ERM) based machine learning (ML) fails to generalize to unseen dynamics. Indeed, it requires i.i.d. data for training and inference while dynamical observations are non-i.i.d. as the distributions change with initial conditions or physical contexts.

Thus any ML framework that handles this question should consider other assumptions. A common one used, e.g., in domain generalization (Wang et al., 2021b), states that data come from several environments a.k.a. domains, each with a different distribution. Training is performed on a sample of the environments and test corresponds to new ones. Domain generalization methods attempt to capture problem invariants via a unique model, assuming that there exists a representation space suitable for all the environments. This might be appropriate for classification, but not for dynamical systems where the underlying dynamics differs for each environment. For this problem, we need to learn a function that adapts to each environment, based on a few observations, instead of learning a single domain-invariant function. This is the objective of meta-learning (Thrun and Pratt, 1998), a general framework for fast adaptation to unknown contexts. The standard gradient-based methods (e.g. Finn et al., 2017) are unsuitable for complex dynamics due to their bi-level optimization and are known to overfit when little data is available for adaptation, as in the few-shot learning setting explored in this paper (Mishra et al., 2018). Like invariant methods, meta-learning usually handles basic tasks e.g. classification; regression on static data or simple sequences and not challenging dynamical systems.

Generalization for modeling real-world dynamical systems is a recent topic. Simple simulated dynamics were considered in Reinforcement Learning (Lee et al., 2020; Nagabandi et al., 2019) while physical dynamics were modeled in recent works (Yin et al., 2021a; Wang et al., 2022). These approaches consider either simplified settings or additional hypotheses e.g. prior knowledge and do not offer general solutions to our adaptation problem (details in Section 3.2).

We propose a new ML framework for generalization in dynamical systems, called **Co**ntext-Informed **D**ynamics **A**daptation (CoDA). Like in domain generalization, we assume

availability of several environments, each with its own specificity, yet sharing some physical properties. Training is performed on a sample of the environments. At test time, we assume access to example data from a new environment, here a trajectory. Our goal is to adapt to the new environment distribution with this trajectory. More precisely, CoDA assumes that the underlying system is described by a parametrized differential equation, either an ordinary differential equation (ODE) or a partial differential equation (PDE). The environments share the parametrized form of the equation but differ by the values of the parameters or initial conditions. CoDA conditions the dynamics model on learned environment characteristics a.k.a. contexts and generalizes to new environments and trajectories with few data. Our main contributions are the following:

- We introduce a multi-environment formulation of the generalization problem for dynamical systems.
- We propose a novel context-informed framework, CoDA, to this problem. It conditions the dynamics model on context vectors via a hypernetwork. CoDA introduces a locality and a low-rank constraint, which enable fast and efficient adaptation with few data.
- We analyze theoretically the validity of our low-rank adaptation setting for modeling dynamical systems.
- We evaluate two variations of CoDA on several ODEs/PDEs representative of a variety of application domains, e.g. chemistry, biology, physics. CoDA achieves state-of-the-art generalization results on in-domain and one-shot adaptation scenarios. We also illustrate how, with minimal supervision, CoDA infers accurately new system parameters from learned contexts.

The paper is organized as follows. In Section 6.2 we present our multi-environment problem. In Section 6.3 we introduce the CoDA framework. In Section 6.4 we detail how to implement our framework. In Section 6.5 we present our experimental results.

6.2 Generalization for Dynamical Systems

We present our generalization problem for dynamical systems, then introduce our multi-environment formalization.

6.2.1 Problem Setting

We consider dynamical systems that are driven by unknown temporal differential equations of the form:

$$\frac{\mathrm{d}u}{\mathrm{d}t} = f(u), \tag{6.1}$$

where $t \in \mathbb{R}$ is a time index, $u(t)$ is a time-dependent state in a space $\mathcal{U}$ and $f : \mathcal{U} \to \mathrm{T}\mathcal{U}$ a function that maps $u(t) \in \mathcal{U}$ to its temporal derivatives in the tangent space $\mathrm{T}\mathcal{U}$. f belongs to a class of vector fields $\mathcal{F}$. $\mathcal{U} \subseteq \mathbb{R}^d$ ($d \in \mathbb{N}^\star$) for ODEs or $\mathcal{U}$ is a function space defined over a spatial domain $\Omega \subset \mathbb{R}^p$, e.g., 2D or 3D Euclidean space, for PDEs.

Functions $f \in \mathcal{F}$ define a space Γ^f of state trajectories $u : \mathcal{I} \to \mathcal{U}$, mapping t in an interval $\mathcal{I}$ including 0, to the state $u(t) \in \mathcal{U}$. Trajectories are defined by the initial condition $u(0) := u_0 \sim \rho_0(\mathcal{U})$ and take the form:

$$\forall t \in \mathcal{I}, u(t) \in \mathcal{U} = u_0 + \int_{\tau=0}^{\tau=t} f(u(\tau))\mathrm{d}\tau \tag{6.2}$$

They are often observed on a fixed time grid $\mathcal{T} = \{0, K = {}^{T}/_{\delta t}\} \subset \mathcal{I}$ of resolution δt. In the following, we assume that $f \in \mathcal{F}$ is parametrized by some unknown attributes, e.g., physical parameters, and external forcing terms which affect the trajectories.

6.2.2 Multi-Environment Learning Problem

We propose to learn the class of functions $\mathcal{F}$ with a data-driven *dynamics model* g_θ parametri-zed by $\theta \in \mathbb{R}^{d_\theta}$. Given $f \in \mathcal{F}$, we observe N trajectories in $\mathcal{D}^f$ (cf. Eq. (6.2)).

The standard ERM objective considers that all trajectories are i.i.d. Here, we propose a multi-environment learning formulation where observed trajectories of f form an environment $e \in \mathcal{E}$. We denote f^e and $\mathcal{D}^e$ the corresponding function and set of N trajectories. We assume that we observe training environments $\mathcal{E}_{\text{tr}}$, consisting of several trajectories from a set of known functions $\{f^e\}_{e\in\mathcal{E}_{\text{tr}}}$. The full dataset $\mathcal{D} = \bigcup_{e\in\mathcal{E}} \mathcal{D}^e$

The goal is to learn g_θ that adapts easily and efficiently to new environments $\mathcal{E}_{\text{ad}}$, corresponding to unseen functions $\{f^e\}_{e\in\mathcal{E}_{\text{ad}}}$ ("ad" stands for adaptation). We define $\forall e \in \mathcal{E}$ the mean square error (MSE) loss, over $\mathcal{D}^e$ as

$$\mathcal{L}(g_\theta, \mathcal{D}^e) := \sum_{u\in\mathcal{D}} \sum_{t\in\mathcal{T}} \|f^e(u^{e,(i)}(t)) - g_\theta(u^{e,(i)}(t))\|_2^2 \tag{6.3}$$

In practice, f^e is unavailable and we can only approximate it from discretized trajectories. We detail later in Eq. (6.11) our approximation method based on an integral formulation. It fits observed trajectories directly in state space.

6.3 The CoDA Learning Framework

We introduce CoDA, a new context-informed framework for learning dynamics in multiple environments. It relies on a general adaptation rule (Section 6.3.1) and introduces two key properties: locality, enforced in the objective (Section 6.3.2) and low-rank adaptation, enforced in the proposed model via hypernetwork-decoding (Section 6.3.3). The validity of this framework for dynamical systems is analyzed in Section 6.3.4 and its benefits are discussed in Section 6.3.5.

6.3.1 Adaptation Rule

The dynamics model g_θ should adapt to new environments. Hence, we propose to condition g_θ on observed trajectories $\mathcal{D}^e, \forall e \in \mathcal{E}$. Conditioning is performed via an *adaptation network* A_π, parametrized by π, which adapts the weights of g_θ to an environment $e \in \mathcal{E}$ according to

$$\theta^e := A_\pi(\mathcal{D}^e) := \theta^c + \delta\theta^e, \quad \pi := \{\theta^c, \{\delta\theta^e\}_{e\in\mathcal{E}}\} \tag{6.4}$$

$\theta^c \in \mathbb{R}^{d_\theta}$ are shared parameters, used as an initial value for fast adaptation to new environments. $\delta\theta^e \in \mathbb{R}^{d_\theta}$ are environment-specific parameters conditioned on $\mathcal{D}^e$.

6.3.2 Constrained Optimization Problem

Given the adaptation rule in Eq. (6.4), we introduce a constrained optimization problem which learns parameters π such that $\forall e \in \mathcal{E}, \delta\theta^e$ is small and g fits observed trajectories. It introduces a locality constraint with a norm $\|\cdot\|$:

$$\min_\pi \sum_{e\in\mathcal{E}} \|\delta\theta^e\|^2 \text{ s.t. } \forall u^e \in \mathcal{D}, \forall t \in \mathcal{T}, \frac{\mathrm{d}u^e(t)}{\mathrm{d}t} = g_{\theta^c+\delta\theta^e}(u^e(t)) \tag{6.5}$$

We consider an approximation of this problem which relaxes the equality constraint with the MSE loss $\mathcal{L}$ in Eq. (6.3).

$$\min_\pi \sum_{e\in\mathcal{E}} \left(\mathcal{L}(\theta^c + \delta\theta^e, \mathcal{D}^e) + \lambda\|\delta\theta^e\|^2\right) \tag{6.6}$$

λ is a hyperparameter. For training, we minimize Eq. (6.6) w.r.t. π over training environments $\mathcal{E}_{\text{tr}}$. After training, θ^c is freezed. For adaptation, we minimize Eq. (6.6) over new environments $\mathcal{E}_{\text{ad}}$ w.r.t. $\{\delta\theta^e\}_{e\in\mathcal{E}_{\text{ad}}}$.

The locality constraint in the training objective Eq. (6.6) enforces $\delta\theta^e$ to remain close to the shared θ^c solutions. It plays several roles. First, it fosters fast adaptation by acting as a constraint over $\theta^c \in \mathbb{R}^{d_\theta}$ during training, s.t., minimas $\{\theta^{e\star}\}_{e\in\mathcal{E}}$ are in a neighborhood of θ^c, i.e., can be reached from θ^c with few update steps. Second, it constrains the hypothesis space at fixed θ^c. Under some assumptions, it can simplify the resolution of the optimization problem w.r.t. $\delta\theta^e$ by turning optimization to a quadratic convex problem with an unique solution. We show this property for our solution in Proposition 6.1. The positive effects of this constraint will be illustrated on an ODE system in Section 6.3.3.

6.3.3 Context-Informed Hypernetwork

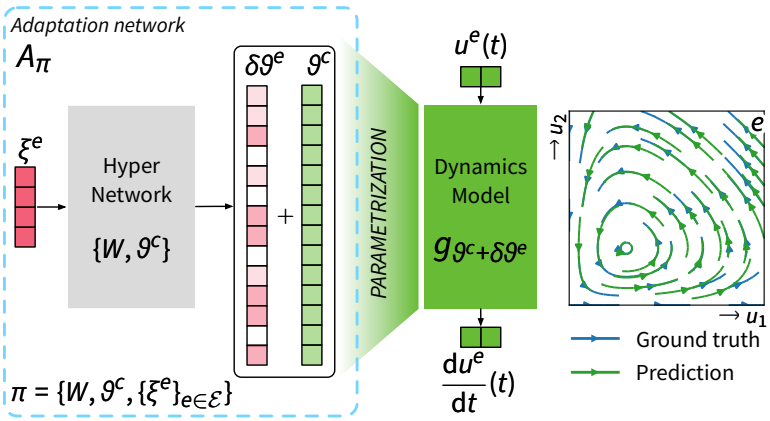


Figure 6.1: Illustration of Context-Informed Dynamics Adaptation (CoDA).

Eq. (6.6) involves learning $\delta\theta^e$ for each environment. For adaptation, $\delta\theta^e$ should be inferred from few observations of the new environment. Learning such high-dimensional parameters is prone to over-fitting, especially in low data regimes. We propose a hypernetwork-based solution (Figure 6.1) to solve efficiently this problem. It operates on a low-dimensional space, yields fixed-cost adaptation and shares efficiently information across environments.

Formulation We estimate $\delta\theta^e$ through a linear mapping of conditioning information, called context, learned from $\mathcal{D}^e$ and denoted $\xi^e \in \mathbb{R}^{d_\xi}$. $W = (W_1, \cdots, W_{d_\xi}) \in \mathbb{R}^{d_\theta \times d_\xi}$ is

the weight matrix of the linear decoder s.t.

$$A_\pi(\mathcal{D}^e) \coloneqq \theta^c + W\xi^e, \quad \pi \coloneqq \{W, \theta^c, \{\xi^e\}_{e\in\mathcal{E}}\} \tag{6.7}$$

The weight matrix W is shared across environments and defines a low-dimensional subspace $\mathcal{W} \coloneqq \text{Span}(W_1, \cdots, W_{d_\xi})$, of dimension at most d_ξ, to which the search space of $\delta\theta^e$ is restricted. ξ^e is specific to each environment and can be interpreted as learning rates along the rows of W. In our experiments, $d_\xi \ll d_\theta$ is small, at most 2. Thus, *adaptation to new environments only requires learning very few parameters, which define a completely new dynamics model g.*

A_π corresponds to an affine mapping of ξ^e parametrized by $\{W, \theta^c\}$, a.k.a. a linear hypernetwork. Note that hypernetworks (Ha et al., 2017) have been designed to handle single-environment problems and learn a separate context per layer. Our formalism involves multiple environments and defines a context per environment for all layers of g.

Linearity of the hypernetwork is not restrictive as contexts are directly learned through an inverse problem detailed in eqs. (6.8) and (6.9), s.t. expressivity is similar to a nonlinear hypernetwork with a final linear activation.

Objectives We derive the training and adaptation objectives by inserting Eq. (6.7) into Eq. (6.6). For training, both contexts and hypernetwork are learned with Eq. (6.8):

$$\min_{\theta^c, W, \{\xi^e\}_{e\in\mathcal{E}_{\text{tr}}}} \sum_{e\in\mathcal{E}_{\text{tr}}} \left(\mathcal{L}(\theta^c + W\xi^e, \mathcal{D}^e) + \lambda\|W\xi^e\|^2\right) \tag{6.8}$$

After training, θ^c is kept fixed and for adaptation to a new environment, only the context vector ξ^e is learned with:

$$\min_{\{\xi^e\}_{e\in\mathcal{E}_{\text{ad}}}} \sum_{e\in\mathcal{E}_{\text{ad}}} \left(\mathcal{L}(\theta^c + W\xi^e, \mathcal{D}^e) + \lambda\|W\xi^e\|^2\right) \tag{6.9}$$

Implementation of eqs. (6.8) and (6.9) is detailed in Section 6.4. We apply gradient descent. In Proposition 6.1, we show for $\|\cdot\| = \ell_2$, that Eq. (6.9) admits an unique solution, recovered from initialization at $\mathbf{0}$ with a single preconditioned gradient step, projected onto subspace $\mathcal{W}$ defined by W.

Proposition 6.1. *Given $\{\theta^c, W\}$ fixed, if $\|\cdot\| = \ell_2$, then Eq. (6.9) is quadratic. If $\lambda' W^\top W$ or $\bar{H}^e(\theta^c) = W^\top \nabla^2_\theta \mathcal{L}(\theta^c, \mathcal{D}^e) W$ are invertible then $\bar{H}^e(\theta^c) + \lambda' W^\top W$ is invertible except for a finite number of λ' values. The problem in Eq. (6.9) is then also convex and admits an unique solution, $\{\xi^{e\star}\}_{e \in \mathcal{E}_{ad}}$. With $\lambda' := 2\lambda$,*

$$\xi^{e*} = -\left(\bar{H}^e(\theta^c) + \lambda' W^\top W\right)^{-1} W^\top \nabla_\theta \mathcal{L}(\theta^c, \mathcal{D}^e) \tag{6.10}$$

☞ Proof in Appendix C.2, p. 224

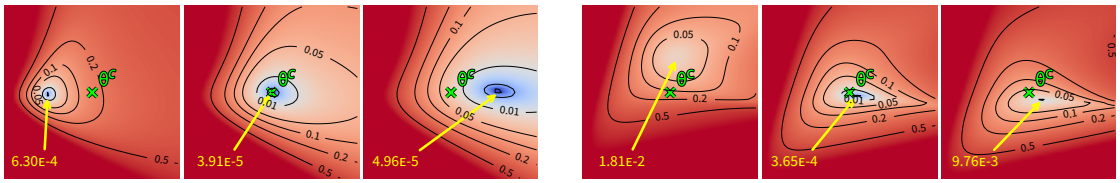
Interpretation We now interpret CoDA by visualizing its loss landscape in Figure 6.2a and comparing it to ERM's loss landscape in Figure 6.2b. We use the package in Li et al. (2018b) to plot loss landscapes around θ^c and consider the *Lotka-Volterra* system, described in Section 6.5.1.

In Figure 6.2a, loss values of CoDA are projected onto subspace $\mathcal{W}$, where $d_\xi = 2$. We make three observations. First, across environments, the loss is smooth and has a single minimum around θ^c. Second, the local optimum of the loss is close to θ^c across environments. Finally, the minimal loss value on $\mathcal{W}$ around θ^c is low across environments. The two first properties were discussed in Section 6.3.2 and are a direct consequence of the locality constraint on $\mathcal{W}$. When $\|\cdot\| = \ell_2$, it makes the optimization problem in Eq. (6.8) quadratic w.r.t. ξ^e and convex under invertibility of $\bar{H}^e(\theta^c) + \lambda' W^\top W$ as detailed in Proposition 6.1. We provided in Eq. (6.10) the closed-form expression of the solution. It also imposes small $\|\xi^e\|$ s.t. when minimizing the loss in Eq. (6.8), θ^c remains close to the local optima of all training environments. The final observation illustrates that CoDA finds a subspace $\mathcal{W}$ with environment-specific parameters of low loss values, i.e., low-rank adaptation performs well.

In Figure 6.2b, loss values of ERM are projected onto the span of the two principal gradient directions. We observe that, unlike CoDA, ERM does not find low loss values. Indeed, it aims at finding θ^c with good performance across environments, thus cannot model several dynamics.

6.3.4 Validity for Dynamical Systems

We further motivate low-rank decoding in our context-informed hypernetwork approach by providing some evidence that gradients at θ^c across environments define a lower-dimensional subspace. We consider the loss $\mathcal{L}$ in Eq. (6.3) and define the gradient subspace



(a) DINo's loss projected onto $\mathcal{W} = \text{Span}(W_1, W_2)$.

(b) ERM's loss projected onto the span of the two principal gradient directions computed with SVD.

Figure 6.2: Loss landscapes centered in θ^c, marked with ×, for 3 *Lotka-Volterra* environments. $\forall e$, → points to the local optimum $\theta^{e\star}$ with loss value reported in yellow.

in Definition 6.1.

Definition 6.1 (Gradient directions)**.** With $\mathcal{L}$ in Eq. (6.3), $\forall \theta^c \in \mathbb{R}^{d_\theta}$ parametrizing a dynamics model g_{θ^c}, the subspace generated by gradient directions at θ^c across environments $\mathcal{E}$ is denoted $\mathcal{G}_{\theta^c} \coloneqq \text{Span}(\{\nabla_\theta \mathcal{L}(\theta^c, \mathcal{D}^e)\}_{e \in \mathcal{E}})$.

We show, in Proposition 6.2, low-dimensionality of $\mathcal{G}_{\theta^c}$ for linearly parametrized systems.

Proposition 6.2 (Low-rank under linearity)**.** *Given a class of linearly parametrized dynamics $\mathcal{F}$ with d_{P} varying parameters, $\forall \theta^c \in \mathbb{R}^{d_\theta}$, subspace $\mathcal{G}_{\theta^c}$ in Definition 6.1 is low-dimensional and* $\dim(\mathcal{G}_{\theta^c}) \leq d_{\text{P}} \ll d_\theta$.

☞ Proof in Appendix C.2, p. 224

The linearity assumption is not restrictive as it is present in a wide variety of real-world systems e.g. Burger or Korteweg–De Vries PDE (Raissi et al., 2019), convection-diffusion (Long et al., 2018b), wave and reaction-diffusion equations (Yin et al., 2021b) etc.

Under nonlinearity, we do not have the same theoretical guarantee, yet, we show empirically in Appendix C.4 that the low-dimensionality of parameters of the dynamics model still holds for several systems. This property is comforted by recent work that highlighted that gradients are low-rank throughout optimization in single-domain settings, i.e., that the solution space is low-dimensional (Gur-Ari et al., 2018; Li et al., 2018a,b). In the same spirit as CoDA, this property was leveraged to design efficient solutions to the learning problems (Frankle and Carbin, 2019; Vogels et al., 2019).

6.3.5 Benefits of CoDA

We highlight the benefits of CoDA. CoDA is a general time-continuous framework that can be used with any approximator g_θ of the derivative Eq. (6.3). It can be trained with a given temporal resolution and tested on another; it handles irregularly-sampled sequences. The choice of the approximator g_θ defines the ability to handle different spatial resolutions for PDEs, as further detailed in Section 6.5.3.

Compared to related adaptation methods, CoDA presents several advantages. First, as detailed in Appendix C.1.1, the adaptation rule in Eq. (6.4) is similar to the one used in gradient-based meta-learning; yet, our first order joint optimization problem in Eq. (6.6) simplifies the complex bi-level optimization problem (Antoniou et al., 2019). Second, CoDA introduces the two key properties of locality constraint and low-rank adaptation which guarantee efficient adaptation to new environments as discussed in Section 6.3.3. Third, it generalizes contextual meta-learning methods (Garnelo et al., 2018; Zintgraf et al., 2019), which also perform low-rank adaptation, via the hypernetwork decoder (details in Appendix C.1.2). Our decoder learns complex environment-conditional dynamics models while controlling their complexity. Finally, CoDA learns context vectors through an inverse problem as Zintgraf et al. (2019). This decoder-only strategy is particularly efficient and flexible in our setting. An alternative is to infer them via a learned encoder of $\mathcal{D}^e$ as Garnelo et al. (2018). Yet, the latter was observed to underfit (Kim et al., 2019), requiring extensive tuning of the encoder and decoder architecture. Overall, CoDA is easy to implement and maintains expressivity with a linear decoder.

6.4 Framework Implementation

We detail how to perform trajectory-based learning with our framework and describe two instantiations of the locality constraint. We detail the corresponding pseudo-code.

Trajectory-Based Formulation As derivatives in Eq. (6.3) are not directly observed, we use in practice for training a trajectory-based formulation of Eq. (6.3). We consider a set of N trajectories, $\mathcal{D}^e$. Each trajectory is discretized over a uniform temporal and spatial grid $\mathcal{T}, \mathcal{X}$.

Our loss is written as:

$$\mathcal{L}(g_\theta, \mathcal{D}^e) = \sum_{u \in \mathcal{D}^e} \sum_{x \in \mathcal{X}} \sum_{t \in \mathcal{T}} \left\| u^{e,(i)}(t,x) - \tilde{u}^{e,(i)}(t,x) \right\|_2^2$$
$$\text{where } \tilde{u}^{e,(i)}(t) = u_0^{e,(i)} + \int_{s=0}^{s=t} g_\theta\left(\tilde{u}^{e,(i)}(s)\right) \mathrm{d}s \tag{6.11}$$

$u^{e,i}(t,x)$ is the value in the i^{th} trajectory from environment e at the spatial coordinate x and time t. $u^{e,i}(t)\big|_{\mathcal{X}}$ is the state vector in the i^{th} trajectory from environment e over the spatial domain at time t and $u_0^{e,i}$ is the corresponding initial condition. To compute $\tilde{u}^{e,i}(t)$, we apply for integration a numerical solver (Hairer et al., 2000) as detailed later.

Locality Constraint Instead of penalizing $\lambda \|W\xi^e\|^2$ in Eq. (6.8), we found it more efficient to penalize separately W and ξ^e. We thus introduce the following regularization:

$$\mathcal{R}(W, \xi^e) \coloneqq \lambda_\xi \mathcal{R}_\xi(\xi^e) + \lambda_W \mathcal{R}_W(W) \tag{6.12}$$

It involves hyperparameters λ_ξ, λ_W and resp. two norms $\mathcal{R}_\xi(\xi^e), \mathcal{R}_W(W)$ which depend on the choice of $\|\cdot\|$ in Eq. (6.6). Minimizing $\mathcal{R}(W, \xi^e)$ minimizes an upper-bound to $\|\cdot\|$, derived in Appendix C.5 for the two considered variations of $\|\cdot\|$:

- CoDA-ℓ_2 sets $\|\cdot\| = \ell_2(\cdot)$ and $\mathcal{R}_\xi = \mathcal{R}_W = \ell_2^2$, constraining $W\xi^e$ to a sphere.
- CoDA-ℓ_1 sets $\|\cdot\| = \ell_1(\cdot)$ and $\mathcal{R}_\xi = \ell_2^2, \mathcal{R}_W(W) = \ell_{1,2}$ over rows, i.e., $\mathcal{R}_W(W) = \sum_{i=1}^{d_\theta} \|W_{i,:}\|_2$ to induce sparsity and find most important parameters for adaptation. $\ell_{1,2}$ constrains $\mathcal{W}$ to be axis-aligned; then the number of solutions is finite as $\dim(\mathcal{W})$ is finite.

Pseudo-Code We solve Eq. (6.8) for training and Eq. (6.9) for adaptation using eqs. (6.11) and (6.12) and Algorithm 2. We back-propagate through the solver with `torchdiffeq` (Chen, 2021) and apply exponential Scheduled Sampling (Bengio et al., 2015) to stabilize training.

6.5 Experiments

We validate our approach on four classes of challenging nonlinear temporal and spatiotemporal physical dynamics, representative of various fields, e.g., chemistry, biology, and fluid

Algorithme 2 : CoDA Pseudo-code

Training

Initialization: $\mathcal{E}_{\text{tr}} \subset \mathcal{E}$, $\{\mathcal{D}^{e_{\text{tr}}}\}_{e_{\text{tr}} \in \mathcal{E}_{\text{tr}}}$ with
$\forall e_{\text{tr}} \in \mathcal{E}_{\text{tr}}, \#\mathcal{D}^{e_{\text{tr}}} = N_{\text{tr}}, \pi = \{W, \theta^c, \{\xi^{e_{\text{tr}}}\}_{e_{\text{tr}} \in \mathcal{E}_{\text{tr}}}\}$ where $W \in \mathbb{R}^{d_\theta \times d_\xi}, \theta^c \in \mathbb{R}^{d_\theta}$
randomly initialized and $\forall e_{\text{tr}} \in \mathcal{E}_{\text{tr}}, \xi^{e_{\text{tr}}} = \mathbf{0} \in \mathbb{R}^{d_\xi}$
while *true* **do**
$\quad \pi \leftarrow \pi - \eta \nabla_\pi \Big(\sum_{e_{\text{tr}} \in \mathcal{E}_{\text{tr}}} \mathcal{L}(\theta^c + W\xi^{e_{\text{tr}}}, \mathcal{D}^{e_{\text{tr}}}) + \mathcal{R}(W, \xi^{e_{\text{tr}}}) \Big)$

Adaptation

Initialization: $e_{\text{ad}} \in \mathcal{E}_{\text{ad}}$, $\mathcal{D}^{e_{\text{ad}}}$ with $\#\mathcal{D}^{e_{\text{ad}}} = N_{\text{ad}}$
while *true* **do**
$\quad \xi^{e_{\text{ad}}} \leftarrow \xi^{e_{\text{ad}}} - \eta \nabla_{\xi^{e_{\text{ad}}}} \Big(\mathcal{L}(\theta^c + W\xi^{e_{\text{ad}}}, \mathcal{D}^{e_{\text{ad}}}) + \mathcal{R}(W, \xi^{e_{\text{ad}}}) \Big)$

dynamics. We evaluate in-domain and adaptation prediction performance and compare them to related baselines. We also investigate how learned context vectors can be used for system parameter estimation. We consider a few-shot adaptation setting where only a few trajectories (N_{ad}) are available at adaptation time in new environments.

6.5.1 Dynamical Systems

We consider four ODEs and PDEs described in Appendix C.6.1. ODEs include *Lotka-Volterra* (Lotka, 1925) and *Glycolitic-Oscillator* (Daniels and Nemenman, 2015), modeling respectively predator-prey interactions and the dynamics of yeast glycolysis. PDEs are defined over a 2D spatial domain and include *Gray-Scott* (Pearson, 1993), a reaction-diffusion system with complex spatiotemporal patterns and the challenging *Navier-Stokes* system (*Navier-Stokes*, Stokes, 1851) for incompressible flows. All systems are nonlinear w.r.t. system states and all but *Glycolitic-Oscillator* are linearly parametrized. The analysis in Section 6.3.4 covers all systems but *Glycolitic-Oscillator*. Experiments on the latter show that CoDA also extends to nonlinearly parametrized systems.

6.5.2 Experimental Setting

We consider forecasting: only the initial condition is used for prediction. We perform two types of evaluation: in-domain generalization on $\mathcal{E}_{\text{tr}}$ *(In-domain)* and out-of-domain adaptation to new environments $\mathcal{E}_{\text{ad}}$ (*Adaptation*). Each environment $e \in \mathcal{E}$ is defined by system parameters and $\theta_{\text{p}}^e \in \mathbb{R}^{d_{\text{p}}}$ denotes those that vary across $\mathcal{E}$. d_{p} represents the

degrees of variations in $\mathcal{F}$; $d_{\text{P}} = 2$ for *Lotka-Volterra*, *Glycolitic-Oscillator*, *Gray-Scott* and $d_{\text{P}} = 1$ for *Navier-Stokes*. Appendix C.6.1 defines for each system the number of training and adaptation environments ($\#\mathcal{E}_{\text{tr}}$ and $\#\mathcal{E}_{\text{ad}}$) and the corresponding parameters. Appendix C.6.1 also reports the number of trajectories N_{tr} per training environment in $\mathcal{E}_{\text{tr}}$ and the distribution $\rho_0(\mathcal{U})$ from which are sampled all initial conditions (including adaptation and evaluation initial conditions). For *Adaptation*, we consider $N_{\text{ad}} = 1$ trajectory per new environment in $\mathcal{E}_{\text{ad}}$ to infer the context vector with Eq. (6.9). We consider more trajectories per adaptation environment in Section 6.5.7.

Evaluation is performed on 32 new test trajectories per environment. We report, in our tables, the mean and standard deviation of MSE across test trajectories (Eq. (6.11)) over four different seeds. We report, in our figures, mean absolute percentage error (MAPE) in % over trajectories, as it allows us to better compare performance across environments and systems. We define $\text{MAPE}(z, y)$ between a d-dimensional input z and target y as $\frac{1}{d}\sum_{j=1\dots d: y_j \neq 0} \frac{|z_j - y_j|}{|y_j|}$. Over a trajectory, it extends into $\sum_{t\in\mathcal{T}} \text{MAPE}(\tilde{u}(t), u(t))\text{d}t$, with $\tilde{u}$ defined in Eq. (6.11).

6.5.3 Implementation of CoDA

We used for g_θ multi-layer perceptrons (MLPs) for ODEs, a resolution-dependent convolutional neural network (ConvNet) for *Gray-Scott* and a resolution-agnostic Fourier neural operator (FNO; Li et al., 2021b) for *Navier-Stokes* that can be used on new resolutions. Architecture details are provided in Appendix C.6.2. We tuned d_ξ and observed that $d_\xi = d_{\text{P}}$, the number of system parameters that vary across environments, performed best (cf. Section 6.5.6). We use Adam optimizer Kingma and Ba (2015) for all datasets; RK4 solver for *Lotka-Volterra*, *Gray-Scott*, *Glycolitic-Oscillator* and Euler solver for *Navier-Stokes*. Optimization and regularization hyperparameters are detailed in Appendix C.6.2.

6.5.4 Baselines

We consider three families of baselines, compared in Appendix Figure C.1 and detailed in Section 3.2. First, gradient-based meta learning (GBML) methods MAML (Finn et al., 2017), ANIL (Rusu et al., 2019) and Meta-SGD (Li et al., 2017). Second, the Multi-Task Learning method LEADS (Yin et al., 2021a). Finally, the contextual meta-learning method CAVIA (Zintgraf et al., 2019), with conditioning via concatenation (Concat) or linear modulation of final hidden features (FiLM, Perez et al., 2018). All baselines are adapted

Table 6.1: Test MSE (↓) in training environments $\mathcal{E}_{\text{tr}}$ (*In-Domain*), new environments $\mathcal{E}_{\text{ad}}$ (*Adaptation*). Best in **bold**; second underlined.

	Lotka-Volterra		*Glycolitic-Oscillator*		*Gray-Scott*		*Navier-Stokes*	
	In-domain	Adaptation	In-domain	Adaptation	In-domain	Adaptation	In-domain	Adaptation
MAML	6.03±0.13E−4	3.15±0.94E−2	5.73±0.21E−3	1.08±0.06E−2	3.67±0.53E−3	2.25±0.39E−3	6.80±0.80E−3	5.11±0.40E−3
ANIL	3.81±0.76E−3	4.57±2.39E−2	7.45±1.15E−3	1.69±0.23E−2	5.01±0.80E−3	3.95±0.11E−3	6.17±0.43E−3	4.86±0.32E−3
Meta-SGD	3.27±1.26E−4	7.22±4.58E−2	4.23±0.69E−3	1.57±0.41E−2	2.85±0.54E−3	2.68±0.20E−3	5.39±2.81E−3	4.43±2.71E−3
LEADS	3.70±0.27E−5	4.76±1.25E−4	3.14±0.33E−3	1.14±0.42E−2	2.90±0.76E−3	1.36±0.43E−3	1.40±0.16E−3	2.86±0.72E−3
CAVIA-FiLM	4.38±1.15E−5	8.41±3.20E−5	4.44±1.46E−4	3.87±1.28E−4	2.81±1.15E−3	1.43±1.07E−3	2.32±1.21E−3	2.26±0.99E−3
CAVIA-Concat	2.43±0.66E−5	6.26±0.77E−5	5.09±0.35E−4	2.37±0.23E−4	2.67±0.48E−3	1.62±0.85E−3	2.55±0.63E−3	2.60±0.82E−3
DINo-ℓ_2	1.52±0.08E−5	1.82±0.24E−5	2.45±0.38E−4	1.98±0.06E−4	1.01±0.15E−3	7.70±0.10E−4	9.40±1.13E−4	1.03±0.15E−3
DINo-ℓ_1	**1.35±0.22E−5**	**1.24±0.20E−5**	**2.20±0.26E−4**	**1.86±0.29E−4**	**9.00±0.57E−4**	**7.40±0.10E−4**	**8.35±1.71E−4**	**9.65±1.37E−4**

to be dynamics-aware i.e. time-continuous: they consider the loss in Eq. (6.11), as CoDA. Moreover, they share the same architecture for g_θ as CoDA.

6.5.5 Generalization Results

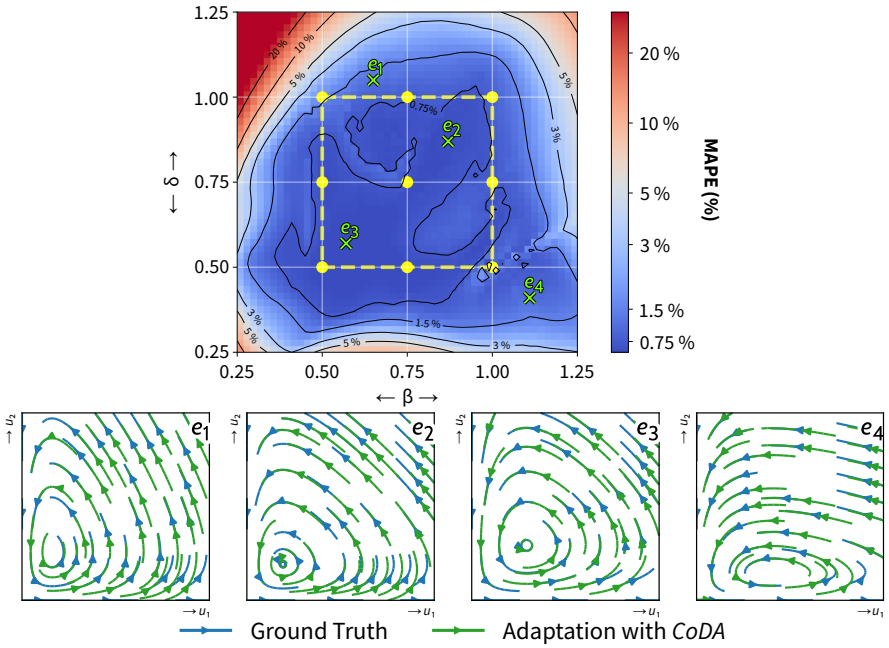


Figure 6.3: *Adaptation* results with CoDA-ℓ_1 on *Lotka-Volterra*. System parameters $\theta_P = (\beta, \delta)$ are sampled in $[0.25, 1.25]^2$ on a 51×51 uniform grid, leading to 2601 adaptation environments $\mathcal{E}_{\text{ad}}$. • are training environments $\mathcal{E}_{\text{tr}}$. We report MAPE (↓) across $\mathcal{E}_{\text{ad}}$ (top). On the bottom, we choose four of them (×, e_1–e_4), to show the ground-truth (blue) and predicted (green) phase space portraits. x, y are respectively the quantity of prey and predator in the system in Eq. (C.12).

In Table 6.1, we observe that CoDA improves significantly test MSE w.r.t. our baselines for both *In-Domain* and *Adaptation* settings. For PDE systems and a given test trajectory, we visualize in Figures C.3 and C.4 in Appendix C.7 the predicted MSE by these models along the ground truth. We also notice improvements for CoDA over our baselines. Across datasets, all baselines are subject to a drop in performance between *In-Domain* and *Adaptation* while CoDA maintains remarkably the same level of performance in both cases. In more detail, GBML methods (MAML, ANIL, Meta-SGD) overfit on training *In-Domain* data especially when data is scarce. This is the case for ODEs which include fewer system states for training than PDEs. LEADS performs better than GBML but overfits

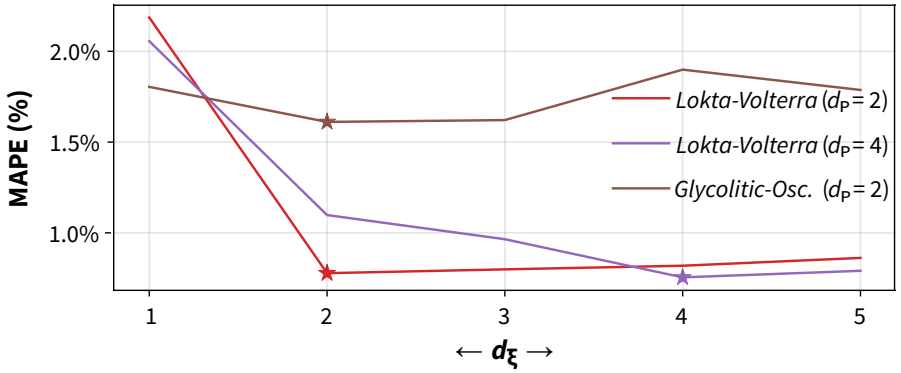


Figure 6.4: Dimension of the context vectors (d_ξ) and test *In-Domain* MAPE (↓) with DINo-ℓ_1. "★" is the smallest MAPE.

Table 6.2: Locality and *In-Domain* test MSE (↓). Best in **bold**, the second underlined.

	Lotka-Volterra		*Glycolitic-Oscillator*	
CoDA	W/o ℓ_2	With ℓ_2	W/o ℓ_2	With ℓ_2
Full	2.28±0.29E−5	1.52±0.08E−5	2.98±0.71E−4	2.45±0.38E−4
First-Layer	2.25±0.29E−5	2.41±0.23E−5	2.38±0.71E−4	**2.12±0.55E−4**
Last-Layer	1.86±0.24E−5	**1.27±0.03E−5**	2.84±0.06E−3	2.84±0.06E−3

for *Adaptation* as it does not adapt efficiently. CAVIA-Concat/FiLM perform better than GBML and LEADS, as they leverage a context, but are less expressive than CoDA. Both variations of CoDA perform best as they combine the benefits of low-rank adaptation and locality constraint. CoDA-ℓ_1 is better than CoDA-ℓ_2 as it induces sparsity, further constraining the hypothesis space.

We evaluate in Figure 6.3 CoDA-ℓ_1 on *Lotka-Volterra* for *Adaptation* over a wider range of adaptation environments ($\#\mathcal{E}_{\text{ad}} = 51 \times 51 = 2601$). We report mean MAPE over $\mathcal{E}_{\text{ad}}$ (top). We observe three regimes: inside the convex hull of training environments $\mathcal{E}_{\text{tr}}$, MAPE is very low; outside the convex hull, MAPE remains low in a neighborhood of $\mathcal{E}_{\text{tr}}$; beyond this neighborhood, MAPE increases. CoDA thus generalizes efficiently in the neighborhood of training environments and degrades outside this neighborhood. We plot reconstructed phase space portraits (bottom) on four selected environments and observe that the learned solution (green) closely follows the target trajectories (blue).

6.5.6 Ablation Studies

We perform two studies on *Lotka-Volterra* and *Glycolitic-Oscillator*. In a first study in Table 6.2, we evaluate the gains due to using ℓ_2 locality constraint on *In-Domain* evaluation. On line 1 (Full), we observe that CoDA-ℓ_2 performs better than CoDA without locality constraint. Prior work perform adaptation only on the final layer with some performance improvements on classification or Hamiltonian system modeling (Raghu et al., 2020; Chen et al., 2020a). In order to evaluate this strategy, we manually restrict hypernetwork-decoding to only one layer in the dynamics model g_θ, either the first layer (line 2) or the last layer (line 3). We observe that the importance of the layer depends on the parametrization of the system: for *Lotka-Volterra*, linearly parametrized, the last layer is better while for *Glycolitic-Oscillator*, nonlinearly parametrized, the first layer is better. CoDA-ℓ_1 generalizes this idea by automatically selecting the useful adaptation subspace via $\ell_{1,2}$ regularization, offering a more flexible approach to induce sparsity.

In a second study in Section 6.5.5, we analyze the impact on MAPE of the dimension of context vectors d_ξ for CoDA-ℓ_1. We recall that d_ξ upper-bounds the dimension of the adaptation subspace $\mathcal{W}$ and was cross-validated in Table 6.1. In the following, d_{P} is the number of parameters that vary across environments. We illustrate the effect of the cross-validation on MAPE for $d_{\text{P}} = 2$ on *Lotka-Volterra* and *Glycolitic-Oscillator* as in Section 6.5.5 and additionally for $d_{\text{P}} = 4$ on *Lotka-Volterra*. We observe in Section 6.5.5 that the minimum of MAPE is reached for $d_\xi = d_{\text{P}}$ with two regimes: when $d_\xi < d_{\text{P}}$, performance decreases as some system dimensions cannot be learned; when $d_\xi > d_{\text{P}}$, performance degrades slightly as unnecessary directions of variations are added, increasing the hypothesis search space. This study shows the validity of the low-rank assumption and illustrates how the unknown d_{P} can be recovered through cross-validation.

6.5.7 Sample Efficiency

We handled originally one-shot adaptation ($N_{\text{ad}} = 1$), the most challenging setting. We vary the number of adaptation trajectories N_{ad} on *Lotka-Volterra* in Table 6.3. With more trajectories, performance improves significantly for MAML; moderately for LEADS; while it remains flat for CoDA. This highlights CoDA's sample-efficiency and meta-overfitting for GBML (Mishra et al., 2018).

Table 6.3: Test MSE (↓) in new environments $\mathcal{E}_{\text{ad}}$ (*Adaptation*) on *Lotka-Volterra* according to the number of adaptation trajectoires. Best for each setting in **bold**.

	Number of adaptation trajectories N_{ad}		
	1	5	10
MAML	3.15±0.94E−3	2.39±0.16E−3	1.73±0.10E−3
LEADS	4.76±1.25E−4	1.99±0.72E−4	1.94±0.35E−4
CoDA-ℓ_1	**1.24±0.20E−5**	**1.21±0.18E−5**	**1.20±0.17E−5**

Table 6.4: Parameter estimation MAPE (↓) for DINo-ℓ_1 on *Lotka-Volterra* ($\#\mathcal{E}_{\text{tr}} = 9$), *Gray-Scott* ($\#\mathcal{E}_{\text{tr}} = 4$) and *Navier-Stokes* ($\#\mathcal{E}_{\text{tr}} = 5$).

	In-convex-hull		Out-of-convex-hull		Overall
	MAPE (%)	$\#\mathcal{E}_{\text{ad}}$	MAPE (%)	$\#\mathcal{E}_{\text{ad}}$	MAPE (%)
Lotka-Volterra	0.15±0.11	625	0.73±1.33	1976	0.59±1.33
Gray-Scott	0.37±0.25	625	0.74±0.67	1976	0.65±0.62
Navier-Stokes	0.10±0.08	40	0.51±0.35	41	0.30±0.33

6.5.8 Parameter Estimation

We use CoDA to perform parameter estimation, leveraging the links between learned context and system parameters.

Empirical observations

In Figure 6.5a (left), we visualize on *Lotka-Volterra* the learned context vectors ξ^e (red) and the system parameters θ_{p}^e (black), $\forall e \in \mathcal{E}_{\text{tr}} \cup \mathcal{E}_{\text{ad}}$. We observe empirically a linear bijection between these two sets of vectors. Such correspondence is being learned in the training environments, we can use the correspondence to verify if it still applies to new adaptation environments. Said otherwise, we can check if our model is able to infer the true parameters for new environments.

We evaluate in Table 6.4 the parameter estimation MAPE over *Lotka-Volterra*, *Gray-Scott* and *Navier-Stokes*. Figure 6.5 displays estimated parameters along estimation MAPE. Experimentally, we observe low MAPE inside and even outside the convex hull of training environments. Thus, CoDA identifies accurately the unknown system parameters with little supervision.

Theoretical motivation

We justify these empirical observations theoretically in Proposition 6.3 under the following conditions:

Assumption 6.1. The dynamics in $\mathcal{F}$ are linear w.r.t. inputs and system parameters.

Assumption 6.2. Dynamics model g, hypernet A are linear.

Assumption 6.3. $\forall e \in \mathcal{E}$, parameters $\theta_{\text{P}}^e \in \mathbb{R}^{d_{\text{P}}}$ are unique.

Assumption 6.4. Context vectors have dimension $d_\xi = d_{\text{P}}$.

Assumption 6.5. The system parameters θ_{P} of all dynamics f in a basis $\mathcal{B}$ of $\mathcal{F}$ are known.

Proposition 6.3 (Identification under linearity)**.** *Under Assumptions 6.1 to 6.5, system parameters are perfectly identified on new environments if the dynamics model g and hypernetwork A satisfy $\forall f \in \mathcal{B}$ with system parameter θ_{P}, $g_{A(\theta_{\text{P}})} = f$.*

☞ Proof in Appendix C.3, p. 226

Intuitively, Proposition 6.3 says that given some observations representative of the degrees of variation of the data (a basis of $\mathcal{F}$) and given the system parameters for these observations (Assumption 6.5), we are guaranteed to recover the parameters of new environments for a family systems. This strong guarantee requires strong conditions. Assumptions 6.1 and 6.2 state that the systems should be linear w.r.t. inputs and that the dynamics model should be linear too. Linearity of the hypernetwork is not an issue as detailed in Section 6.3.3. Assumption 6.3 applies to several real-world systems used in our experiments (cf. Appendix C.3, lemmas C.1 and C.2). Assumption 6.4 is not restrictive as we showed that d_{P} is recovered through cross-validation (Section 6.5.5).

We propose an extension of Proposition 6.3 in Proposition 6.4 to nonlinear systems w.r.t. inputs and nonlinear dynamics model g. This alleviates the linearity assumption in Assumptions 6.1 and 6.2 and better fits our experimental setting.

Proposition 6.4 (Local identification under nonlinearity)**.** *For linearly parametrized systems, nonlinear w.r.t. inputs and nonlinear dynamics model g_θ with parameters output*

by a linear hypernetwork A, $\exists \alpha > 0$ s.t. system parameters are perfectly identified $\forall e \in \mathcal{E}$ where $\|\xi^e\| \leq \alpha$ if $\forall f \in \mathcal{B}$ with parameter θ_{P}, $g_{A(\alpha \frac{\theta_{\mathrm{P}}}{\|\theta_{\mathrm{P}}\|})} = f$.

☞ Proof in Appendix C.3, p. 226

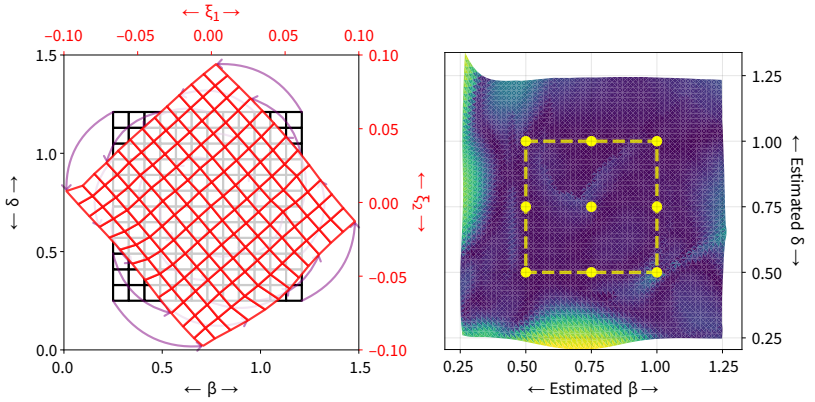
Proposition 6.4 states that system parameters are recovered for environments with context vectors of small norm, under a rescaling condition on true system parameters. Proposition 6.4 explains why estimation error increases when system parameters differ greatly from training ones, as these systems are more likely to violate the norm condition.

6.6 Conclusion

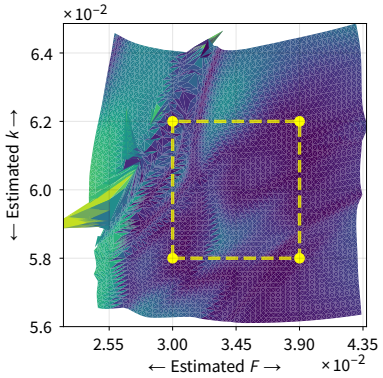
We introduced CoDA, a new framework to learn context-informed data-driven dynamics models in multiple environments. CoDA generalizes with little retraining and few data to new related physical systems and outperforms prior methods on several real-world nonlinear dynamics. Many promising applications of CoDA are possible, notably for spatiotemporal problems, e.g., partially observed systems, reinforcement learning, or NN-based simulation.

Acknowledgements

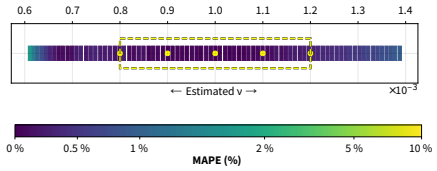
We acknowledge the financial support from DL4CLIM ANR-19-CHIA-0018-01, DEEPNUM ANR-21-CE23-0017-02, OATMIL ANR-17-CE23-0012, RAIMO ANR-20-CHIA-0021-01 and LEAUDS ANR-18-CE23-0020.



(a) *Lotka-Voterra*: $(\beta, \delta) \in [0.25, 1.25]^2$



(b) *Gray-Scott*:
$F_r \in [2.25, 4.35] \cdot 10^{-2}$,
$k_r \in [5.6, 6.4] \cdot 10^{-2}$



(c) *Navier-Stokes*: $\nu \in [0.6, 1.4] \cdot 10^{-3}$

Figure 6.5: Parameter estimation with CoDA-ℓ_1 in new adaptation environments on (a) *Lotka-Voterra*, (b) *Gray-Scott* and (c) *Navier-Stokes*. In (a), we visualize: on the left, context vectors ξ (red); on the right, true parameters (β, δ) (black). In (b) and (c), we visualize estimated parameters with corresponding estimation MAPE (↓). ● are training environments $\mathcal{E}_{\text{tr}}$ with known parameters. - - delimits the convex hull of $\mathcal{E}_{\text{tr}}$.

Chapter 7

Modeling Continuous Dynamics

Continuous PDE Dynamics Forecasting with Implicit Neural Representations

In this final chapter of contributions, thanks to recent advances in representing continuous signals, we search for discovering a path toward completely continuous modeling of the spatiotemporal dynamics. We focus on orchestrating the time-continuous dynamics predictor and the space-continuous function representing the state signal via contextual techniques, inspired by previous efforts to link dynamics learning and contextual neural networks. This work resulted in a conference paper at ICLR 2023.

Yuan Yin[*], Matthieu Kirchmeyer[*], Jean-Yves Franceschi[*], Alain Rakotomamonjy, and Patrick Gallinari. Continuous PDE dynamics forecasting with implicit neural representations. ICLR 2023.

7.1 Introduction

Modeling the dynamics and predicting the temporal evolution of physical phenomena is paramount in many fields, e.g., climate modeling, biology, fluid mechanics, and energy (Willard et al., 2023). Classical solutions rely on a well-established physical paradigm: the evolution is described by differential equations derived from physical first principles, and then solved using numerical analysis tools, e.g., finite elements, finite volumes, or spectral methods (Olver, 2014). The availability of large amounts of data from observations or simulations has motivated data-driven approaches to this problem (Brunton and Kutz, 2022), leading to the rapid development of the field with deep learning methods. The main motivations for this research track include developing surrogate or reduced order models that can approximate high-fidelity full order models at reduced computational costs (Kochkov et al., 2021), complementing classical solvers, e.g., to account for additional components of the dynamics (Yin et al., 2021b), or improving low fidelity models (Belbute-Peres et al., 2020).

Most of these attempts rely on workhorses of deep learning like convolutional neural networks (ConvNets) (Ayed et al., 2022) or graph neural networks (GNNs) (Li et al., 2020; Pfaff et al., 2021; Brandstetter et al., 2022). They all require prior space discretization either on regular or irregular grids, such that they only capture the dynamics on the training grid and cannot generalize outside it. Neural operators, a recent trend, learn mappings between function spaces (Li et al., 2021b; Lu et al., 2021) and thus alleviate some limitations of prior discretization approaches. Yet, they still rely on fixed grid discretization for training and inference: e.g., regular grids for Li et al. (2021b) or a free-form but predetermined grid for Lu et al. (2021). Hence, the number and/or location of sensors have to be fixed across train and test which is restrictive in many situations (Prasthofer et al., 2022). Mesh-agnostic approaches for solving canonical partial differential equations (PDEs) are another trend (Raissi et al., 2019; Sirignano and Spiliopoulos, 2018). In contrast to physics-agnostic grid-based approaches, they aim at solving a known PDE as usual solvers do, and cannot cope

with unknown dynamics. This idea was concurrently developed for computer graphics, e.g., for learning 3D shapes (Sitzmann et al., 2019; Mildenhall et al., 2020; Tancik et al., 2020) and coined as implicit neural representations (INRs).

When used as solvers, these methods can only tackle a single initial value problem and are not designed for long-term forecasting outside the training horizon. Because of these limitations, none of the above approaches can handle situations encountered in many practical applications such as different geometries, e.g., phenomena lying on a Euclidean plane or an Earth-like sphere; variable sampling, e.g., irregular observation grids that may evolve at train and test time as in adaptive meshing (Berger and Oliger, 1984); scarce training data, e.g., when observations are only available at a few spatiotemporal locations; multi-scale phenomena, e.g., in large scale-dynamics systems as climate modeling, where integrating intertwined subgrid scales, a.k.a. the closure problem, is ubiquitous (Zanna and Bolton, 2021). These considerations motivate the development of new machine learning models that improve existing approaches on several of these aspects.

In our work, we aim at forecasting PDE-based spatiotemporal physical processes with a versatile model tackling the aforementioned limitations. We adopt an agnostic approach, i.e., not assuming any prior knowledge on the physics. We introduce DINo (Dynamics-aware Implicit Neural representations), a model operating continuously in space and time, with the following contributions.

Continuous flow learning. DINo aims at learning the PDE's flow to forecast its solutions, in a continuous manner so that it can be trained on any spatial and temporal discretization and applied to another. To this end, DINo embeds spatial observations into a small latent space via INRs; then it models continuous-time evolution by a learned latent ordinary differential equation (ODE).

Space-time separation. To efficiently encode different sequences, we propose a novel INR parameterization, amplitude modulation, implementing a space-time separation of variables. This simplifies the learned dynamics, reduces the number of parameters and greatly improves performance.

Spatiotemporal versatility. DINo combines the benefits of prior models (cf. Table 7.1). It tackles new sequences via its amplitude modulation. Sequential modeling with an ODE makes it extrapolate to unseen times within or beyond the training horizon. Thanks to INRs' spatial flexibility, it generalizes to new grids or resolutions, predicts at arbitrary positions, and handles sparse irregular grids or manifolds.

Table 7.1: Comparison of data-driven approaches to spatiotemporal PDE forecasting.

	Model	Reference	1. PDE-agnostic prediction on new initial conditions	2. Train / test space grid independence	3. Evaluation at unobserved spatial locations	4. Free-form spatial domain (manifold, irregular mesh)	5. Time continuous	6. Time extrapolation
Discrete	NODE	Chen et al. (2018)	✓	✗	✗	✗	✓	✓
	MP-PDE	Brandstetter et al. (2022)	✓	✗	✗	✓	✗	✓
Operator	Markov neural operator (MNO)	Li et al. (2021a)	✓	✓	✗	✗	✗	✓
	DeepONet	Lu et al. (2021)	✓	✗	✓	✓	✓	✗
INRs	PINNs	Raissi et al. (2019)	✗	✓	✓	✓	✓	✗
	DINo	Ours	✓	✓	✓	✓	✓	✓

Empirical validation. We demonstrate DINo's versatility and state-of-the-art performance vs. prior neural PDE forecasters, representative of grid-based, operator, and INR-based methods, via thorough experiments on challenging multi-dimensional PDEs in various spatiotemporal generalization settings.

7.2 Problem Description

Problem Setting. We aim at modeling, via a data-driven approach, the temporal evolution of a continuous fully-observed deterministic spatiotemporal phenomenon. It is described by trajectories $u\colon \mathbb{R} \to \mathcal{U}$ in a set Γ; we use $u_t := u(t) \in \mathcal{U}$. We focus on Initial Value Problems, where only u_t at any time t is required to infer $u_{t'}$ for $t' > t$. Hence, trajectories share the same dynamics but differ by their initial condition $u_0 \in \mathcal{U}$. $\mathbb{R}$ is the temporal domain and $\mathcal{U}$ is the functional space of the form $\Omega \to \mathbb{R}^d$, where $\Omega \subset \mathbb{R}^p$ is a compact spatial domain and d the number of observed values. In other words, u_t is a spatial function of $x \in \Omega$, with vectorial output $u_t(x) \in \mathbb{R}^d$; cf. examples of Section 7.4.1.

Table 7.2: Notation for the observation grids in space and time.

	Training	Test	Comments
Spatial grid	$\mathcal{X}_{\text{tr}} = \mathcal{X}$	$\mathcal{X}_{\text{ts}} \in \{\mathcal{X}, \mathcal{X}'\}$	$\mathcal{X}, \mathcal{X}' \subset \Omega, \mathcal{X} \neq \mathcal{X}'$
Temporal grid	$\mathcal{T}_{\text{tr}} = \mathcal{T}$	$\mathcal{T}_{\text{ts}} \in \{\mathcal{T}, \mathcal{T}'\}$	$\mathcal{T} \in \mathcal{I}, \mathcal{T}' \in \mathcal{I}', \mathcal{T} \neq \mathcal{T}'$

To this end, we consider the setting illustrated in Figure 7.1. We observe a finite training set of trajectories $\mathcal{D}$, with a free-form spatial observation grid $\mathcal{X}_{\text{tr}} \subset \Omega$ and on discrete times $t \in \mathcal{T} \subset \mathcal{I} = [0, T]$. At test time, we are only given a new initial condition u_0, with observed values $u_0|_{\mathcal{X}_{\text{ts}}}$ on a new observation grid $\mathcal{X}_{\text{ts}}$, potentially different from $\mathcal{X}_{\text{tr}}$. Inference is performed on both train and test trajectories given only the initial condition, on a new free-form grid $\mathcal{X}' \subset \Omega$ and times $t \in \mathcal{T}' \subset \mathcal{I}' = [0, T']$. Inference grid $\mathcal{X}'$ comprises observed positions (respectively $\mathcal{X}_{\text{tr}}$ and $\mathcal{X}_{\text{ts}}$ for train and test trajectories) and unobserved positions corresponding to spatial interpolation. Note that the inference temporal horizon is larger than the train one: $T < T'$.

For simplicity, *In-s* refers to data in $\mathcal{X}'$ on the observation grid ($\mathcal{X}_{\text{tr}}$ for *train* / $\mathcal{X}_{\text{ts}}$ for *test*), *Out-s* to data in $\mathcal{X}'$ outside the observation grid; *In-t* refers to times within the train horizon $\mathcal{T} \subset [0, T]$, and *Out-t* to times in $\mathcal{T}' \setminus \mathcal{T} \subset (T, T']$, beyond T, up to inference horizon T'.

Evaluation scenarios. The desired properties in Section 7.1 call for spatiotemporally continuous forecasting models. We select six criteria that our approach should meet; cf. column titles of Table 7.1. First, the model should be robust to the change of initial condition u_0, i.e., generalize to *test* trajectories (col. 1). Second, it should extrapolate beyond the train conditions: in space, on a test observation grid that differs from the train one, i.e., $\mathcal{X}' = \mathcal{X}_{\text{ts}} \neq \mathcal{X}_{\text{tr}}$ (*In-s*, col. 2), and outside the observed train and test grid, i.e., on $\mathcal{X}' \setminus \mathcal{X}_{\text{ts}}, \mathcal{X}' \setminus \mathcal{X}_{\text{tr}}$ (*Out-s*, col. 3); in time, between train snapshots (col. 5) and beyond the observed train horizon T (*Out-t*, col. 6). Finally, it should adapt to free-form spatial domains, i.e., to various geometries (e.g., manifolds) or irregular grids (col. 4). See also Figure 7.1.

Objective. To satisfy these requirements, we learn the flow Ψ of the physical system:

$$\begin{aligned} \Psi \colon (\mathcal{U} \times \mathbb{R}) &\to \mathcal{U}, \\ u_t, \tau &\mapsto \Psi_\tau(u_t) = u_{t+\tau} \quad \forall u \in \Gamma, t, \tau \in \mathbb{R}_+. \end{aligned} \tag{7.1}$$

Learning the flow is a common strategy in sequential models to better generalize beyond the train time horizon. Yet, so far, it has always been learned with discretized models, which poses generalization issues violating our requirements.

7.3 Model

We present DINo, the first space/time-continuous model that tackles all prediction tasks of Section 7.2, without the above limitations. We specify DINo's inference procedure (Section 7.3.1), illustrated in Figure 7.2 (left), then introduce each of its components (Section 7.3.2) and how they are trained (Section 7.3.3, Figure 7.2 (right)). Finally, we detail our implementation based on amplitude modulation, a novel INR parameterization for spatiotemporal data which performs the separation of variables (Section 7.3.4).

7.3.1 Inference Model

As explained in Section 7.2, we aim at estimating the flow Ψ in Eq. (7.1), so that our model can be trained on an observed grid $\mathcal{X}_{\text{tr}}$ and perform inference given a new one $\mathcal{X}_{\text{ts}}$, both possibly irregular. To this end, we leverage a space- and time-continuous formulation, independent of a given data discretization.

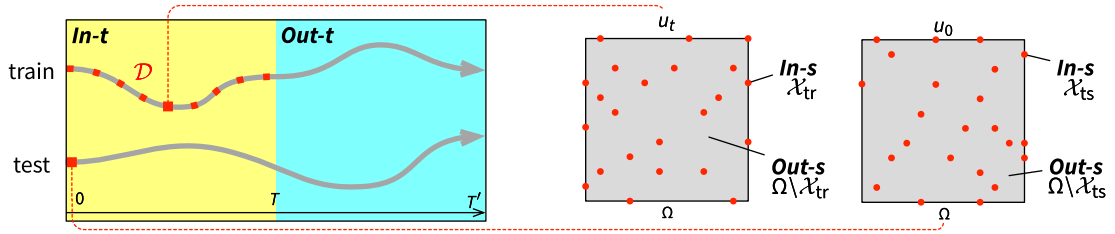


Figure 7.1: (Left) We represent time contexts. The *train* trajectory consists of training snapshots (■), observed in a train interval $[0, T]$ denoted *In-t*. The line (—) in continuation is a forecasting of this trajectory beyond *In-t*, in $(T, T']$ denoted *Out-t*. The line below (—, *test*) is a forecasting from a new initial condition u_0 (■) on *In-t* and *Out-t*. (Middle and right) We illustrate spatial contexts. (Middle) Dots (●) correspond to the train observation grid $\mathcal{X}_{tr}$, denoted *In-s*. *Out-s* denotes the complementary domain $\Omega \setminus \mathcal{X}_{tr}$. (Right) New test observation grid $\mathcal{X}_{ts}$, used as an initial point for forecasting (left).

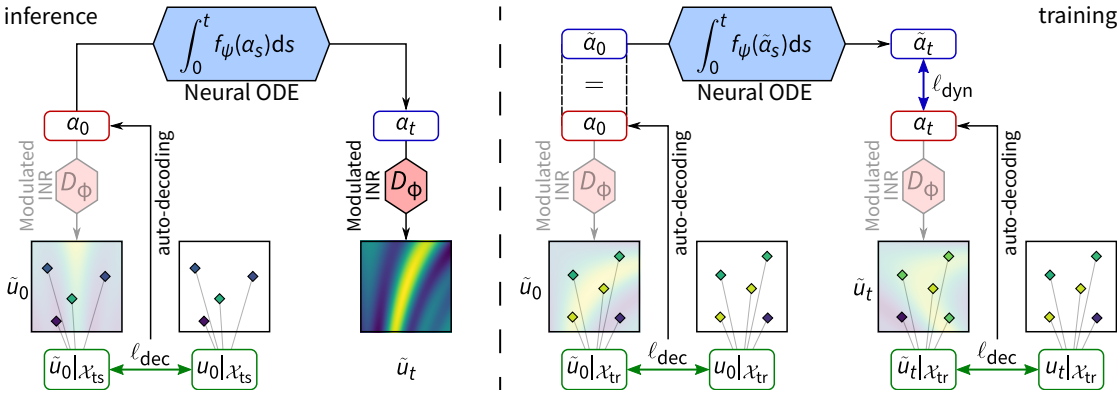


Figure 7.2: Proposed DINo model. Inference (left): given a new initial condition observed on a grid $\mathcal{X}_{ts}$, $u_0|_{\mathcal{X}_{ts}}$, forecasting amounts at decoding α_t to $\tilde{u}_t$, by unrolling α_0 with a time-continuous ODE dynamics model f_ψ. Train (right): given an observation grid $\mathcal{X}_{tr}$ and a space-continuous decoder D_ϕ, α_t is learned by auto-decoding s.t. $D_\phi(\alpha_t)|_{\mathcal{X}_{tr}} = u_t|_{\mathcal{X}_{tr}}$. Its evolution is then modeled with f_ψ.

At inference, DINo starts from a single initial condition $u_0 \in \mathcal{U}$ and uses a flow to forecast its dynamics. DINo first embeds spatial observations from u_0 into a latent vector α_0 of small dimension d_α via an encoder of spatial functions $E_\varphi : \mathcal{U} \to \mathbb{R}^{d_\alpha}$ (ENC). Then, it unrolls a latent time-continuous dynamics model $f_\psi : \mathbb{R}^{d_\alpha} \to \mathbb{R}^{d_\alpha}$ given this initial condition (DYN). Finally, it decodes latent vectors via a decoder $D_\phi : \mathbb{R}^{d_\alpha} \to \mathcal{U}$ into a function of space (DEC). At any time t, D_ϕ takes as input α_t and outputs a function $\tilde{u}_t : \Omega \to \mathbb{R}^n$. This results in the following model, illustrated in Figure 7.2 (left):

$$\text{(ENC)}\ \alpha_0 = E_\varphi(u_0), \qquad \text{(DYN)}\ \frac{\mathrm{d}\alpha}{\mathrm{d}t} = f_\psi(\alpha), \qquad \text{(DEC)}\ \forall t, \tilde{u}_t = D_\phi(\alpha_t). \tag{7.2}$$

7.3.2 Components

We now further detail each component involved at inference from Eq. (7.2).

Encoder: $\alpha_t = E_\varphi(u_t)$. The encoder computes a latent vector α_t given observation u_t at any time t. It is used in two different contexts, respectively for train and test. At train time, given an observed trajectory $u|_\mathcal{T}$, it will encode any u_t into α_t (see Section 7.3.3). At inference time, only u_0 is available, and then only α_0 is computed to be used as the initial value for the dynamics. Given the decoder D_ϕ, α_t is a solution to the inverse problem $D_\phi(\alpha_t) = u_t$. We solve this inverse problem with auto-decoding (Park et al., 2019). Denoting $\ell_{\text{dec}}(\phi, \alpha_t; u_t) = \|D_\phi(\alpha_t) - u_t\|_2^2$ the decoding loss where $\|\cdot\|_2$ is the euclidean norm of a function and S the number of update steps, auto-decoding defines E_φ as:

$$E_\varphi(u_t) = \alpha_t^S, \quad \text{where} \quad \forall i > 1, \alpha_t^i = \alpha_t^{i-1} - \eta \nabla_{\alpha_t} \ell_{\text{dec}}(\phi, \alpha_t^{i-1}; u_t) \quad \text{and} \quad \varphi = \phi. \tag{7.3}$$

In practice, we observe a discretization $(\mathcal{X}_{\text{tr}}, \mathcal{X}_{\text{ts}})$ and accordingly approximate the norm in ℓ_{dec} as in Eq. (7.6). Compared to auto-encoding, auto-decoding underfits less (Kim et al., 2019) and is more flexible: without requiring specialized encoder architecture, it handles free-formed (irregular or on a manifold) observation grids as long as the decoder shares the same property.

Decoder: $\tilde{v}_t = D_\phi(\alpha_t)$. We define a flexible decoder using a coordinate-based INR network with parameters conditioned on α_t. An INR $I_\theta : \Omega \to \mathbb{R}^n$ is a space-continuous model parameterized by $\theta \in \mathbb{R}^{d_\theta}$ defined on domain Ω. It approximates functions independently of the observation grid, e.g., it handles irregular grids and changing observation positions, unlike FNO and DeepONet. Thus, it constitutes a flexible alternative

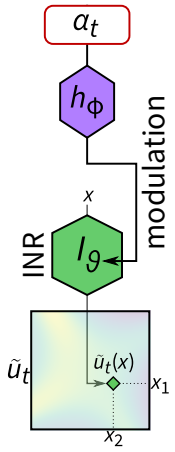


Figure 7.3: Decoding via INR – Eq. (7.4)

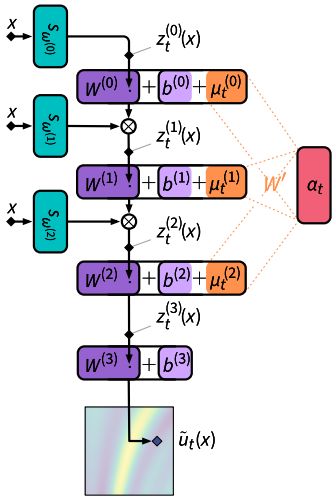


Figure 7.4: Amplitude modulation – Eq. (7.9). $z_t^{(l-1)}$ is input to the l^{th} linear layer and combined with the spatial basis $s_{\omega^{(l)}}$ via Hadamard product.

to operators suitable for auto-decoding. To implement the conditioning of the INR's parameters, we use a hypernetwork (Ha et al., 2017) $h_\phi \colon \mathbb{R}^{d_\alpha} \to \mathbb{R}^{d_\theta}$, as illustrated in Figure 7.3. It generates high-dimensional parameters $\theta_t \in \mathbb{R}^{d_\theta}$ of the INR given the low-dimensional latent vector $\alpha_t \in \mathbb{R}^{d_\alpha}$. In summary, the decoder D_ϕ, parameterized by ϕ, is defined as:

$$\forall x \in \Omega, \quad \tilde{u}_t(x) = D_\phi(\alpha_t)(x) \coloneqq I_{h_\phi(\alpha_t)}(x). \tag{7.4}$$

The decoder's predictions at all spatial locations $x \in \Omega$ thus all depend on α_t. We provide further details on the precise implementation in Section 7.3.4.

Dynamics model: $\frac{\mathrm{d}\alpha_t}{\mathrm{d}t} = f_\psi(\alpha_t)$**.** Finally, the dynamics model $f_\psi \colon \mathbb{R}^{d_\alpha} \to \mathbb{R}^{d_\alpha}$ defines a flow via an ODE in the latent space. The initial condition can be defined at any time t by encoding with E_φ the corresponding input function u_t.

Overall flow. Combined together, our components define the following flow in the input space that can approximate the data flow Ψ in Eq. (7.1):

$$\forall t, \tau \in \mathbb{R}_+, \qquad (u_t, \tau) \mapsto D_\phi\Big(E_\varphi(u_t) + \int_t^{t+\tau} f_\psi(\alpha_s)\mathrm{d}s\Big) \quad \text{where } \alpha_t = E_\varphi(u_t). \tag{7.5}$$

To summarize, DINo defines a time-continuous latent temporal model with a space-continuous emission function D_ϕ, combining the flexibility of space and time continuity. This is fully novel to our knowledge, as prior latent approaches are discretized (cf. Fraccaro (2018) for state-space models).

7.3.3 Training

Given components (ENC), (DEC), (DYN), we present their training procedure, illustrated in Figure 7.2 (right). We use a simple two-stage optimization process, close to recent works in video prediction (Yan et al., 2021). Given the train sequences $\mathcal{D}$, we first apply auto-decoding to obtain the latent vectors $\alpha_\mathcal{T} = \{\alpha_t^u\}_{t\in\mathcal{T},u\in\mathcal{D}}$ and the decoder parameters ϕ. We then learn the parameters of the dynamics ψ by modeling the latent flow over $\alpha_t^u, \forall u \in \mathcal{D}$. We detail this procedure in Appendix D.3.1, which can be formalized as a two-stage optimization problem that we solve in parallel without inducing training instability (cf. Appendix D.3.2):

$$\begin{aligned} \min_{\psi} \quad & \ell_{\text{dyn}}(f_\psi, \alpha_\mathcal{T}) \coloneqq \sum_{u\in\mathcal{D}} \sum_{t\in\mathcal{T}} \left\| \alpha_t^u - \left(\alpha_0^u + \int_0^t f_\psi(\alpha_s^u)\mathrm{d}s\right)\right\|_2^2 \\ \text{s.t. } \alpha_\mathcal{T}, \phi = \arg\min_{\alpha_\mathcal{T},\phi} \quad & \ell_{\text{dec}}(D_\phi, \alpha_\mathcal{T}) \coloneqq \sum_{u\in\mathcal{D}} \sum_{x\in\mathcal{X}_{\text{tr}}} \sum_{t\in\mathcal{T}} \left\| u_t(x) - D_\phi(\alpha_t^u)(x)\right\|_2^2. \end{aligned} \tag{7.6}$$

7.3.4 Decoder Implementation via Amplitude-Modulated INRs

We now specify our implementation of decoder D_ϕ in Eq. (7.4). This includes the definition of the INR architecture I_θ and of the hypernetwork h_ϕ. We introduce for the latter a new method called amplitude modulation, which implements a space-time separation of variables.

I_θ as FourierNet. We implement I_θ as a FourierNet, a state-of-the-art INR architecture, which instantiates a multiplicative filter network (MFN). A FourierNet relies on the recursion in Eq. (7.7), where $x \in \Omega$ is an input spatial location, $z^{(l)}(x)$ is the hidden feature vector at layer l for x and $s_{\omega^{(l)}}(x) = [\cos(\omega^{(l)}x), \sin(\omega^{(l)}x)]$ is a Fourier basis:

$$\begin{cases} z^{(0)}(x) = s_{\omega^{(0)}}(x), \qquad z^{(L)}(x) = W^{(L-1)}z^{(L-1)}(x) + b^{(L-1)}, \\ z^{(l)}(x) = \left(W^{(l-1)}z^{(l-1)}(x) + b^{(l-1)}\right) \odot s_{\omega^{(l)}}(x) \quad \text{for } l \in [\![1, L-1]\!], \end{cases} \tag{7.7}$$

where we fix $W^{(0)} = 0$, $b^{(0)} = 1$, $s_{\omega^{(0)}}(x) = x$. Denoting $W = [W^{(l)}]_{l=1}^{L-1}, b = [b^{(l)}]_{l=1}^{L-1}, \omega = [\omega^{(l)}]_{l=1}^{L-1}$, we fit a FourierNet to an input function v observed on a grid $\mathcal{X}$ by learning $\{W, b, \omega\}$ s.t. $\forall x \in \mathcal{X}, z^{(L)}(x) = v(x)$. In practice, we observe that fixing ω uniformly sampled performs similarly to learning them, so we exclude them from training parameters.

FourierNets are interpretable, a property we leverage to separate time and space via amplitude modulation. Fathony et al. (2021) show that $\exists M \gg L \in \mathbb{N}, \exists \{c_j^{(m)}\}_{m=1}^{M}$ a set of coefficients that depend individually on $\{W, b\}$ and $\exists \{\gamma^{(m)}\}_{m=1}^{M}$ a set of parameters that depend individually on those of the filters ω s.t. the j^{th} dimension of $z^{(L)}(x)$ can be expressed as:

$$z_j^{(L)}(x) = \sum_{m=1}^{M} c_j^{(m)} s_{\gamma^{(m)}}(x) + \text{bias} \tag{7.8}$$

Eq. (7.8) involves a basis of spatial functions $\{s_{\gamma^{(m)}}\}_{m=1}^{M}$ evaluated on x and the amplitudes of this basis $\{c_j^{(m)}\}_{m=1}^{M}$. Note that Eq. (7.8) can be extended to other choices of $s_{\omega^{(l)}}$ (Fathony et al., 2021).

***h* as amplitude modulation.** h generates the INR's parameters θ_t given α_t to model a target input function u_t. We implement h as element-wise shift and scale transformations (FiLM; Perez et al., 2018) of the linear layers parameters W, b (excluding those of the filters ω). Then, in Eq. (7.8), amplitudes $c_j^{(m)}$ only depend on time while the basis functions $s_{\gamma^{(m)}}$ only depend on space: this corresponds to a modeling assumption of separation of variables (Le Dret and Lucquin, 2016) in u. We call our technique amplitude modulation. In practice, as Dupont et al. (2022), we consider latent shift transformations (Figure 7.4), detailed in Eq. (7.9). Eq. (7.9) extends Eq. (7.7) by introducing a shift term $\mu_t^{(l-1)}$ at each layer l, defined as $\mu_t^{(l-1)} = W'^{(l-1)}\alpha_t$, where $W' = [W'^{(l-1)}]_{l=1}^{L-1}$ is another weight matrix:

$$z_t^{(l)}(x) = \left(W^{(l-1)} z_t^{(l-1)}(x) + b^{(l-1)} + \mu_t^{(l-1)}\right) \odot s_{\omega^{(l)}}(x). \tag{7.9}$$

The INR's parameters are defined as $h_\phi(\alpha_t) = \{W; b + W'\alpha_t; \omega\}$ where $\phi = \{W, b, W'\}$ are h's parameters. Thus, amplitude modulation separates time and space. We show in Table D.1 that it significantly improves performance, particularly time extrapolation.

7.4 Experiments

We assess the spatiotemporal versatility of DINo, following Section 7.2. We introduce our experimental setting (Section 7.4.1), which includes a variety of challenging PDE datasets, state-of-the-art baselines, and forecasting tasks. Then, we present and comment on the experimental results (Section 7.4.2).

7.4.1 Experimental Setting

Datasets. We consider the following PDEs defined over a spatial domain Ω, with further details in Appendix D.3.

- **2D Wave equation** (*Wave*) is a second-order PDE $\frac{\partial^2 v}{\partial t^2} = c^2 \Delta v$. v is the displacement w.r.t. the rest position and c is the wave traveling speed. We consider its first-order form, so that $u_t = (v_t, \frac{\partial v_t}{\partial t})$ has a two-dimensional output ($d = 2$).
- **2D Navier Stokes** (*Navier-Stokes*, Stokes, 1851) corresponds to an incompressible fluid dynamics $\frac{\partial w}{\partial t} = -v\nabla w + \nu \Delta w + f, w = \nabla \times v, \nabla v = 0$, where v is the velocity field and w the vorticity. ν is the viscosity and f is a constant forcing term. The input has only one channel, i.e., $d = 1$.
- **3D Spherical shallow water** (*Shallow-Water*, Galewsky et al., 2004): it involves the vorticity w, tangent to the sphere's surface, and the thickness of the fluid h. The input is $u_t = (w_t, h_t)$ ($d = 2$).

Baselines. We reimplement representative models from Section 3.3 and table 7.1 and adapt them to our multi-dimensional datasets.

- **CNODE** (Ayed et al., 2022) combines a ConvNet and an ODE solver to handle regular grids.
- **MP-PDE** (Brandstetter et al., 2022) uses a GNN to handle free-formed grids, yet is unable to predict outside the observation grid. We developed an interpolative extension, **I-MP-PDE**, to handle this limitation; it performs bicubic interpolation on the observed grid, and training is done on the resulting interpolation.
- **MNO** (Li et al., 2021a): an autoregressive version of Fourier neural operator (FNO; Li et al., 2021b) for regular grids; MNO can be evaluated on new uniform grids.

- **DeepONet** (Lu et al., 2021), considered autoregressively (Wang and Perdikaris, 2021) where we remove time from the trunk net's input. DeepONet can be evaluated on new spatial locations without interpolation.

- **SIREN** (Sitzmann et al., 2019) and **MFN** (Fathony et al., 2021) are two INR methods that we extend to fit our setting. We consider an agnostic setting, i.e., without the knowledge of the differential equation, and perform sequence conditioning to generalize to more than a trajectory. This is achieved by learning a latent vector with auto-decoding; it is then concatenated to the spatial coordinates.

Tasks. We evaluate models on various forecasting tasks which combine the evaluation scenarios of Section 7.2. Performance is measured by the prediction mean square error (MSE) given only an initial condition.

- **Space and time generalization.** We consider a uniform grid $\mathcal{X}'$ for inference. Training is performed on different observations grids $\mathcal{X}_{\text{tr}}$ subsampled from $\mathcal{X}'$ with different ratios, $s \in \{5\%, 25\%, 50\%, 100\%\}$ where $s = 100\%$ corresponds to the full inference grid, i.e., $\mathcal{X}_{\text{tr}} = \mathcal{X}'$. In this setting, we consider that all trajectories (*train* and *test*) share the same observation grid $\mathcal{X}_{\text{tr}} = \mathcal{X}_{\text{ts}}$. We evaluate MSE error on $\mathcal{X}'$ over the train time interval (*In-t*) and beyond (*Out-t*) at each subsampling ratio.

- **Flexibility w.r.t. input grid.** We vary the test observation grid, i.e., $\mathcal{X}_{\text{ts}} \neq \mathcal{X}_{\text{tr}}$ and perform inference on $\mathcal{X}' = \mathcal{X}_{\text{ts}}$, i.e., on the test observation grid (*In-s*) under two settings:

  - **Generalizing across grids:** $\mathcal{X}_{\text{tr}}, \mathcal{X}_{\text{ts}}$ are subsampled differently from the same uniform grid; s_{tr} (resp. s_{ts}) is the train (resp. test) subsampling ratio.

  - **Generalizing across resolutions:** $\mathcal{X}_{\text{tr}}, \mathcal{X}_{\text{ts}}$ are subsampled with the same ratio s from two uniform grids with different resolutions; the train resolution is fixed to $r_{\text{tr}} = 64$ while we vary the test resolution $r_{\text{ts}} \in \{32, 64, 256\}$.

- **Data on manifold.** We consider a PDE on a sphere and combine several evaluation scenarios, as described later.

- **Finer time resolution.** We consider an inference time grid $\mathcal{T}'$ with a finer resolution than the train one $\mathcal{T}$.

Table 7.3: Space and time generalization. Train and test observation grids are equal and subsampled from an uniform 64×64 grid, used for inference. We report MSE (↓) on the inference time interval $\mathcal{T}'$, divided within training horizon (*In-t*, $\mathcal{T}$) and beyond (*Out-t*, outside $\mathcal{T}$) across subsampling ratios.

Model		Navier-Stokes				Wave			
		Train		Test		Train		Test	
		In-t	Out-t	In-t	Out-t	In-t	Out-t	In-t	Out-t
s = 5% subsampling ratio									
Discrete	I-MP-PDE	8.154E−3	8.166E−3	7.926E−3	8.225E−3	7.055E−4	7.097E−4	1.138E−3	1.116E−3
Operator	DeepONet	3.330E−3	7.370E−3	1.346E−2	1.408E−2	8.331E−4	9.295E−3	1.692E−2	3.256E−2
INR	SIREN	8.741E−3	1.767E−1	4.303E−2	2.126E−1	2.738E−3	1.818E−2	3.339E−2	6.964E−2
	DINo	**1.029E−3**	**1.655E−3**	**1.326E−3**	**1.813E−3**	**4.088E−5**	**4.121E−5**	**6.415E−5**	**7.392E−5**
s = 25% subsampling ratio									
Discrete	I-MP-PDE	3.135E−4	7.245E−4	3.476E−4	7.658E−4	3.293E−5	1.108E−4	5.142E−5	1.545E−4
Operator	DeepONet	9.016E−4	5.936E−3	9.376E−3	1.328E−2	5.722E−4	1.061E−2	1.757E−2	3.221E−2
INR	SIREN	5.180E−3	2.175E−1	2.436E−1	3.861E−1	8.995E−4	1.292E−2	1.783E−2	5.143E−2
	DINo	**1.020E−4**	**4.504E−4**	**2.646E−4**	**5.951E−4**	**3.949E−6**	**4.436E−6**	**1.089E−5**	**1.174E−5**
s = 100% subsampling ratio									
Discrete	CNODE	2.319E−2	9.652E−2	2.305E−2	1.143E−1	2.337E−5	5.280E−4	3.057E−5	7.288E−4
	MP-PDE	1.140E−4	5.500E−4	**1.785E−4**	5.856E−4	**1.718E−7**	1.993E−5	**9.256E−7**	4.261E−5
Operator	MNO	**3.190E−5**	8.678E−4	2.763E−4	8.946E−4	9.381E−6	4.890E−3	1.993E−4	6.128E−3
	DeepONet	1.375E−3	6.573E−3	9.704E−3	1.244E−2	6.431E−4	1.293E−2	1.847E−2	3.317E−2
INR	SIREN	1.066E−3	4.336E−1	3.874E−1	1.037E0	3.674E−4	9.956E−3	3.013E−2	7.842E−2
	MFN	1.651E−3	1.037E0	2.106E−1	1.059E0	1.408E−4	1.763E−1	4.735E−3	2.274E−1
	DINo (no sep.)	3.235E−4	1.593E−3	7.850E−4	1.889E−3	2.641E−6	4.081E−5	5.977E−5	2.979E−4
	DINo	8.339E−5	**3.115E−4**	2.092E−4	**4.311E−4**	3.309E−6	**3.506E−6**	9.495E−6	**9.946E−6**

Table 7.4: Flexibility w.r.t. input grid. Observed test / train grid differ ($\mathcal{X}_{ts} \neq \mathcal{X}_{tr}$). We report *test* MSE (↓) for *Navier-Stokes* on $\mathcal{X}' = \mathcal{X}_{ts}$ (*In-s*). **Green** **Yellow** **Red** mean excellent, good, poor MSE.

(a) Generalization across grids: $\mathcal{X}_{tr}, \mathcal{X}_{ts}$ are subsampled with different ratios $s_{tr} \neq s_{ts}$ among $\{5, 50, 100\}\%$ from the same uniform 64×64 grid.

Subsampling	Test→	$s_{ts} = 5\%$		$s_{ts} = 50\%$		$s_{ts} = 100\%$	
Train ↓		In-t	Out-t	In-t	Out-t	In-t	Out-t
$s_{tr} = 5\%$	MP-PDE	1.330E−1	3.852E−1	1.859E−1	6.680E−1	2.105E−1	7.120E−1
	DINo	**1.494E−3**	**2.291E−3**	**1.257E−3**	**1.883E−3**	**1.287E−3**	**1.947E−3**
$s_{tr} = 50\%$	MP-PDE	4.494E−2	9.403E−2	4.793E−3	1.997E−2	6.330E−3	3.712E−2
	DINo	**2.470E−4**	**4.697E−4**	**2.073E−4**	**4.284E−4**	**2.058E−4**	**4.361E−4**
$s_{tr} = 100\%$	MP-PDE	1.358E−1	3.355E−1	1.182E−2	2.664E−2	**1.785E−4**	5.856E−4
	DINo	**2.495E−4**	**4.805E−4**	**2.109E−4**	**4.325E−4**	2.092E−4	**4.311E−4**

(b) **Generalization across resolutions**: $\mathcal{X}_{ts}$ (resp. $\mathcal{X}_{tr}$) are subsampled at the same ratio $s \in \{5, 100\}\%$ from different uniform grids with resolution $r_{ts} \in \{32, 64, 256\}$ (resp. $r_{tr} = 64$).

Test resolution →		$r_{ts} = 32$ - $\mathcal{X}_{ts} \neq \mathcal{X}_{tr}$		$r_{ts} = 64$ - $\mathcal{X}_{ts} = \mathcal{X}_{tr}$		$r_{ts} = 256$ - $\mathcal{X}_{ts} \neq \mathcal{X}_{tr}$	
Subsampling ↓		In-t	Out-t	In-t	Out-t	In-t	Out-t
$s = 5\%$	MP-PDE	3.209E−1	6.472E−1	**2.465E−4**	1.105E−3	2.239E−1	8.253E−1
	DINo	**5.308E−3**	**9.544E−3**	2.533E−4	**8.832E−4**	**1.991E−3**	**2.942E−3**
$s = 100\%$	MNO	4.547E−3	9.281E−3	**1.277E−4**	8.525E−4	2.174E−3	4.975E−3
	MP-PDE	4.194E−2	9.109E−2	1.597E−4	6.483E−4	4.648E−2	1.381E−1
	DINo	**2.321E−4**	**6.386E−4**	2.320E−4	**6.385E−4**	**2.322E−4**	**6.385E−4**

7.4.2 Results

Space and time generalization. We report prediction MSE in Table 7.3 for varying subsampling ratios $s \in \{5\%, 25\%, 100\%\}$ on *Navier-Stokes* and *Wave*. Appendix D.1 provides a fine-grained evaluation inside the train observation grid (*In-s*) or outside (*Out-s*) and reports additionally the results for $s = 50\%$. We visualize some predictions in Appendix D.2. DINo is compared to all baselines when $s = 100\%$, i.e., $\mathcal{X}' = \mathcal{X}_{\text{tr}} = \mathcal{X}_{\text{ts}}$, and otherwise it is compared only to models which handle irregular grids and prediction at arbitrary spatial locations (DeepONet, SIREN, MFN, I-MP-PDE).

- **General analysis.** We observe that all models degrade when the subsampling ratio s decreases. DINo performs competitively overall: it achieves the best *Out-t* performance on all subsampling settings, outperforms all the baselines on low subsampling ratios, and performs comparably to the competitive discretized alternatives (MP-PDE, CNODE) and operator (MNO) when $s = 100\%$, i.e., when observation and inference grids are equal. Note that this fully observed setting is favorable for CNODE, MP-PDE, and MNO, designed to perform inference on the observation grid. This can be seen in Table 7.3, where DINo is slightly outperformed only for a few settings. MP-PDE is significantly better only on *Wave* for *In-t*. Overall, ConvNets and GNNs exhibit good performance for spatially local dynamics like *Wave*, while INRs (like DINo) and MNO are more adapted to global dynamics like *Navier-Stokes*.

- **Analysis per model.** MP-PDE is the most competitive baseline across datasets as it combines a strong and flexible encoder (GNNs) to a good dynamics model; however, it cannot predict outside the observation grid (*Out-s*). To keep a strong competitor, we extend this baseline into its interpolative version I-MP-PDE on subsampled settings. I-MP-PDE is competitive for high subsampling ratios, e.g., $s \in \{50\%, 100\%\}$ but underperforms w.r.t. DINo at lower subsampling ratios due to the accumulated interpolation error. MNO is a competitive baseline on *Navier-Stokes*, performing on par with MP-PDE and DINo inside the training horizon (*In-t*); its performance on *Out-t* degrades more significantly compared to other models, especially DINo. DeepONet is more flexible than MP-PDE as it can predict at arbitrary locations. As no interpolation error is introduced, it outperforms I-MP-PDE for $s = 5\%$ on *train* data. Yet, we observe that it underperforms especially on *Out-t* w.r.t. its alternatives. Finally, we observe that SIREN and MFN fit correctly the training horizon *In-t* on *train*, yet generalize poorly outside this horizon *Out-t* or

on new initial conditions (*test*). This is in accordance with our analysis of Section 3.3; we highlight that this is not the case for DINo which extrapolates temporally and generalizes to new initial conditions thanks to its sequential modeling of the flow. Thus, *DINo is currently the state-of-the-art INR model for spatiotemporal data*.

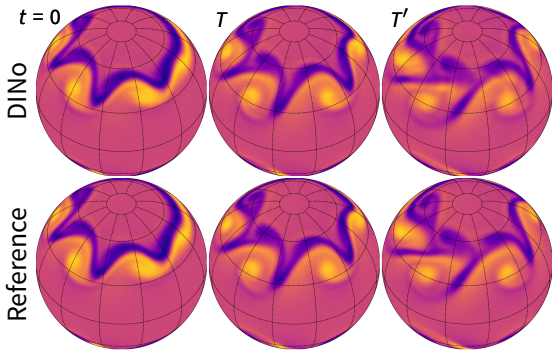
- **Modulation.** We observe that modulating both amplitudes and frequencies (row DINo (no sep.) in Table 7.3) degrades performance w.r.t. DINo (row DINo in Table 7.3) that only modulates amplitudes. Amplitude modulation enables long temporal extrapolation and reduces the number of parameters. Hence, as opposed to DINo (no sep.) which is outperformed by some baselines, time-space variable separation in DINo is an essential ingredient of the model to reach state-of-the-art levels.

Flexibility w.r.t. input grid. We consider in Table 7.4 *Navier-Stokes* and compare DINo to the most competitive baselines, MP-PDE and MNO (with $s = 100\%$ subsampling ratio).

- **Generalizing across grids.** In Table 7.4a, we consider that the test observation grid $\mathcal{X}_{\text{ts}}$ is different from the train one $\mathcal{X}_{\text{tr}}$. This occurs when sensors differ between two observed trajectories. We vary the subsampling ratio for the train observation grid s_{tr} and the test one s_{ts}. We report *test* MSE on new grids $\mathcal{X}' = \mathcal{X}_{\text{ts}}$. We observe that DINo is very robust to changing grids between *train* and *test*, while MP-PDE's performance degrades, especially for low subsampling ratios, e.g., 5%. For reference, we report in Table D.2 Appendix D.1 (col. 3) the performance when $\mathcal{X}' = \mathcal{X}_{\text{tr}}$, where MP-PDE is substantially better.

- **Generalizing across spatial resolutions.** In Table 7.4b we vary the test resolution r_{ts}. We train at a resolution $r_{\text{tr}} = 64$ and perform inference at resolutions $r_{\text{ts}} \in \{32, 64, 256\}$. For that, we build a high-fidelity 256×256 simulation dataset and downscale it to obtain the other resolutions. We observe that DINo's performance is the stablest across resolutions in the uniform or irregular setting. MNO is also relatively stable but is only applicable to uniform grids while MP-PDE is particularly brittle, especially for a 5% ratio.

Data on manifold. We consider in Figure 7.5 *Shallow-Water* in a super-resolution setting: test resolution is twice the train one, close to weather prediction applications. We observe an irregular 3D Euclidean coordinate grid $\mathcal{X}_{\text{tr}} = \mathcal{X}_{\text{ts}} \subset \mathbb{R}^3$ shared for *train* and *test*. It samples uniformly Euclidean positions on the sphere, via the quasi-uniform skipped



Model	In-t	Out-t
I-MP-PDE	1.908E−3	7.240E−3
DINo	**1.063E−4**	**6.466E−4**

Figure 7.5: Data on manifold. DINo's *Shallow-Water* super-resolution *test* prediction against the reference (left); *test* MSE comparison (↓) (right).

latitude-longitude grid (Weller et al., 2012). We predict the PDE on *test* trajectories with a conventional latitude-longitude inference grid $\mathcal{X}'$. At Earth scale, $\mathcal{X}_{\text{tr}}$ corresponds to a resolution of about 300 km, and $\mathcal{X}'$ to 150 km. DINo significantly outperforms I-MP-PDE, making it a viable candidate for this complex setting.

Table 7.5: Finer time resolution. *Test* MSE (↓) under $\mathcal{T}'$ for *Navier-Stokes*.

Model	In-t	Out-t
I-DINo (linear)	3.459E−4	5.598E−4
I-DINo (quadratic)	2.165E−4	4.473E−4
DINo (ODE solve)	**2.151E−4**	**4.388E−4**

Finer time resolution. We consider in Table 7.5 a longer and 10 times finer test time grid $\mathcal{T}'$ than the training grid $\mathcal{T}$ on *Navier-Stokes*. We observe the same spatial uniform grid across *train* and *test* and perform inference on this grid. We compare DINo that performs prediction with an ODE solver, to interpolating coarser predictions obtained at the train resolution (I-DINo). We report the corresponding *test* MSE. We observe that the ODE solver accurately extrapolates outside the train temporal grid, outperforming interpolation. This confirms that DINo benefits from its continuous-time modeling of the flow, providing consistency and stability across temporal resolutions.

7.5 Conclusion

We propose DINo, a novel space- and time-continuous data-driven forecasting model for PDEs. DINo handles settings encountered in many applications, where existing methods fail. We assess its extensive spatiotemporal generalization abilities on a variety of PDEs and its generalization to unseen sparse irregular meshes and resolutions. Its competitive results over recent PDE forecasters make it a strong alternative for real-world settings with free-formed spatiotemporal conditions. There are many promising extensions, e.g., improving generalization to new parameters (Kirchmeyer et al., 2022).

Acknowledgements

We thank Emmanuel de Bézenac and Jérémie Donà for helpful insights and discussions on this project. We also acknowledge financial support from DL4CLIM (ANR-19-CHIA-0018-01) and DEEPNUM (ANR-21-CE23-0017-02) ANR projects. This study has been conducted using E. U. Copernicus Marine Service Information.

Part III

Epilogue

Chapter 8

Conclusion

In this thesis, we have addressed several problems that arise when applying data-driven neural dynamics models in real-world scenarios. We have analyzed and clarified these long-ignored issues and have paved the way for extending the capabilities of current models, notably in their generality and adaptability.

We first focused on the scenario where we have a single source of data from single dynamics. Following the trend of incorporating physical priors into deep learning pipelines, we discussed how to make the neural network cooperate with a numerical model yet to be identified. We proposed a way to automatically calibrate the contribution of both parts, which maximizes the advantages of two counterpart models. The hybrid model benefits from the flexibility of neural networks and the regularity of first-principles numerical models, making it more performant than learning from data with neural networks alone. In practice, this calibration also helps to better identify the undetermined numerical model.

Then we dealt with multi-environment data that reveal different dynamics. We exploited the data from two different perspectives: generalization in known dynamical systems and adaptation to novel systems. For the first, we were inspired by the intuition from differential equations: if different systems have only marginal changes w.r.t. each other, it is always possible to find a common neural network. This common term is trained using all data. The ability to predict in each system is preserved by a constrained environment-

dependent neural network.

By migrating the reflections from the functional space to the parameter space, we were then able to propose an efficient, minimal adaptation through a low-dimensional learnable context vector. The context-based approach was inspired by the way the numerical method is parameterized. This not only reduces the cost of adaptation to new systems but also simplifies adaptation by convexifying the loss surface for similar systems. By analyzing the resulting contexts, we also found that they are directly related to the physical parameter of the phenomenon and can be used to identify the parameter of a new system.

With such adaptation techniques, we turned our focus to continuous modeling with free-form spatially discretized data for a single system. By representing the system states in a latent space through a continuous spatial decoder and learning a continuous dynamics model only in this space, we obtained a predictive model that is continuous in space and time and capable of predicting beyond the horizon of the training data.

In conclusion, this thesis has made contributions to improving the performance and adaptability of data-driven neural dynamics models for real-world applications. By addressing various challenges and proposing novel techniques, we have expanded the capabilities of current models and paved the way for future developments in this field. The proposed approaches for incorporating physical priors and adapting to new systems have the potential to significantly improve the accuracy and efficiency of these models, while the continuous modeling technique offers a powerful tool for predicting complex dynamic systems.

This thesis contributes to the field of neural dynamics modeling, presenting new opportunities for research and applications. Drawing upon established literature on neural dynamics models, machine learning, and deep learning techniques such as model-based/machine-learning hybrid physics modeling, meta-learning, multi-task learning, contextual deep learning, etc., the thesis demonstrates the potential of general machine learning frameworks for dynamics modeling. The positive results and potential impact of this work make it an exciting direction for further exploration. In the upcoming chapter, we will highlight several under-explored paths with potential for future investigation.

Chapter 9

Perspective

This chapter discusses several challenges in the field of dynamical systems and proposes potential research directions to address them. The challenges include handling various boundary conditions, learning dynamics on non-rectangular domains, designing efficient encoders for dynamical systems, handling partially-observed systems, and finding alternatives to neural solvers for spatiotemporal dynamics.

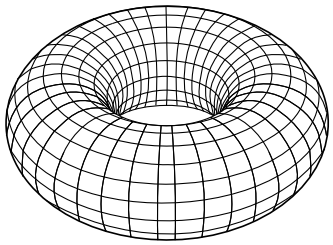
Handling various boundary conditions. The boundary condition of a system plays a crucial role in determining the possible trajectories and dynamics of the system. In some cases, it can even change the nature of the system's behavior. This challenge is akin to the problem of learning from multiple dynamics, as discussed in this thesis (refer to Figure 2.10a). However, existing approaches have limitations in adapting to new boundary conditions since they tightly integrate the boundary condition with the dynamics model. For example, circular padding is often used in ConvNet to handle periodic boundary conditions. Only a few explorations have been made in recent literature like Wang et al. (2021b); Karlbauer et al. (2022). To overcome these implementation biases, more flexible dynamics models are needed that can handle a wide range of boundary conditions.

Learning dynamics on geometries. Learning dynamics on non-rectangular domains is a challenging problem that arises from the need to generalize across boundary conditions. In addition to the function that determines the boundary condition, the shape of the

spatial domain Ω itself also poses a challenge.

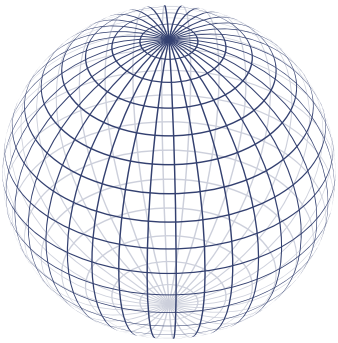
Most neural dynamics models are based on established architectures such as ConvNet and learn dynamics in a rectangular domain (de Bézenac et al., 2018; Li et al., 2021b; Kochkov et al., 2021), which is only compatible with image-like data. However, in many scenarios, the dynamics should be learned in the domain with a different shape. In many applications, the spatial domain can be a non-rectangular subset of $\mathbb{R}^p$, and sometimes even harder, a non-convex subset, e.g., the airflow around an airfoil or an aircraft should be solved in a spatial domain with a hole inside which correspond to the shape of the object (See Figure 9.1c for an example); the simulation of the phenomena on Earth surface.

Most approaches attempting to solve this problem are based on graphs, e.g., Iakovlev et al. (2021); Pfaff et al. (2021); Grattarola and Vandergheynst (2022), and are therefore limited by the structure of the graph to connect the values, not by the geometry itself. The learned model is biased towards the graph as presented in Section 3.3. Others, by projecting the non-rectangular domain onto a rectangular on which the existing models can be applied, e.g., Li et al. (2022), without an explicit encoding of the geometry. Although the INRs have the potential to address the topic, they are currently in an underdeveloped state.



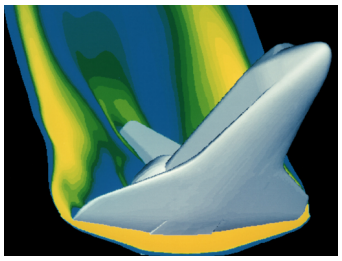
GYassineMrabetTalk, CC BY-SA 3.0, via Wikimedia Commons

(a) A torus. The periodic boundary conditions on the Euclidean plane generate such a topology in space.



Geek3, CC BY-SA 3.0, via Wikimedia Commons

(b) A sphere. Phenomena at the surface of a planet consider the spherical geometry.



Credit: NASA, Public domain, via Wikimedia Commons

(c) A simulation of the fluid dynamics around a model of the space shuttle. The airflow is simulated around the solid aircraft.

Figure 9.1: Common types of geometry for dynamics modeling.

Efficient encoding for a dynamical system. The importance of context learned from data through encoding is emphasized in Part II. However, existing encoders may not always

be suitable for our data and can act as bottlenecks in the overall pipeline. To address this issue, we employ auto-decoders, which offer advantages such as the ability to combine a neural network with a differential equation solver or to be evaluated at any position in the spatial domain without structural bias in weights, as demonstrated in CoDA and DINo, respectively.

Despite these advantages, the training of auto-decoders is more difficult and the code generation for new incoming functions requires optimization, which is impractical for broader applications. Auto-encoders with explicit encoders have the advantage of simplifying the optimization problem during training procedures and adapting to new situations using new inputs without the need for retraining. Therefore, designing new encoders that match the capabilities of decoders is an exciting research direction. Graph-based approaches are often used for free-form input encoding, but they still face generalization issues. A recent study by Prasthofer et al. (2022) sheds light on this problem.

Hybrid modeling for partially-observed dynamical systems. The study of partially-observed dynamical systems poses a significant challenge in the field of dynamical systems, as much of the current research is focused on completely observed systems. In many real-world scenarios, such as weather prediction, the dynamics are only partially observed, making it difficult to accurately predict their behavior. This is because only certain observable quantities are available, while others remain hidden. For example, in weather prediction, while changes in temperature, pressure, and other variables at different altitudes can be directly measured, the full velocity field that drives the weather phenomena remains unobserved.

Purely data-driven methods can struggle to accurately model partially-observed systems due to their lack of physical bias. Hybrid modeling, which combines data-driven approaches with known parts of the dynamics, offers a promising solution to this problem. Recent attempts to apply this approach to real-world applications, such as the work of Ayed et al. (2022); Donà et al. (2022), have yielded promising results. However, there are still several challenges to be addressed in bridging the gap between current research and real-world deployment. For instance, to enable more flexible modeling with neural networks, the physical solvers may need to be reimplemented using differentiable programming tools, which will require significant domain-specific scientific and engineering expertise. Nonetheless, overcoming these challenges is crucial to realize the full potential of hybrid modeling for addressing the complexities of partially-observed dynamical systems in a

range of real-world applications.

Alternatives to neural solvers. In recent years, neural solvers have gained popularity as a promising approach to solving differential equations. The current dominant approach, known as physics-informed neural networks (PINNs; Raissi et al., 2019), integrates the physical signal with a loss function to produce an unknown solution for a differential equation. The physical signal is encoded in the neural solution through gradient-based optimization.

While PINNs have been applied to many differential equations, they still face challenges in optimization due to the highly nonconvex nature of the physical loss. Furthermore, this approach has been underexplored for spatiotemporal dynamics, and currently, there are no efficient neural alternatives for spatiotemporal dynamics solvers. This highlights the need for further research in this area to develop new and innovative solutions that can efficiently handle the complexities of spatiotemporal dynamics. Overcoming these challenges will require a combination of advanced optimization techniques, innovative architectures, and novel loss functions. Despite these challenges, the potential benefits of neural solvers for spatiotemporal dynamics are immense, making this an exciting area of research with significant potential for real-world impact.

* * *

With the rapid development of machine learning techniques and the growing interest in physical modeling, the field is poised for exciting new developments in the coming years. The future of dynamical systems modeling with machine learning looks bright, with numerous opportunities for interdisciplinary collaborations and the potential for significant advancements.

Bibliography

Antreas Antoniou, Harrison Edwards, and Amos J. Storkey. How to train your MAML. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=HJGven05Y7. (p. 113)

Martín Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization. *CoRR*, abs/1907.02893, 2019. URL http://arxiv.org/abs/1907.02893. (p. 51)

Ibrahim Ayed, Nicolas Cedilnik, Patrick Gallinari, and Maxime Sermesant. EP-Net: Learning cardiac electrophysiology models for physiology-based constraints in data-driven predictions. In Yves Coudière, Valéry Ozenne, Edward J. Vigmond, and Nejib Zemzemi, editors, *Functional Imaging and Modeling of the Heart - 10th International Conference, FIMH 2019, Bordeaux, France, June 6-8, 2019, Proceedings*, volume 11504 of *Lecture Notes in Computer Science*, pages 55–63. Springer, 2019. URL https://doi.org/10.1007/978-3-030-21949-9_7. (p. 66)

Ibrahim Ayed, Emmanuel de Bézenac, Arthur Pajot, and Patrick Gallinari. Modelling spatiotemporal dynamics from earth observation data with neural differential equations. volume 111, pages 2349–2380, 2022. doi: 10.1007/s10994-022-06139-2. URL https://doi.org/10.1007/s10994-022-06139-2. (pp. 10, 50, 66, 70, 84, 97, 126, 136, 151, and 261)

Peter L. Bartlett, Dylan J. Foster, and Matus J. Telgarsky. Spectrally-normalized margin bounds for neural networks. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30, pages 6240–6249. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/b22b257ad0519d4500539da3c8bcf4dd-Paper.pdf. (pp. 91 and 204)

Jonathan Baxter. A model of inductive bias learning. *Journal of Artificial Intelligence Research*, 12:149–198, Feb 2000. doi: 10.1613/jair.731. URL https://doi.org/10.1613/jair.731. (pp. 54, 89, 200, 201, and 202)

Philipp Becker, Harit Pandya, Gregor H. W. Gebhardt, Cheng Zhao, C. James Taylor, and Gerhard Neumann. Recurrent kalman networks: Factorized inference in high-dimensional deep feature spaces. 97:544–552, 2019. URL http://proceedings.mlr.press/v97/becker19a.html. (p. 45)

Filipe de Avila Belbute-Peres, Thomas D. Economon, and J. Zico Kolter. Combining differentiable PDE solvers and graph neural networks for fluid flow prediction. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 2402–2411. PMLR, 2020. URL http://proceedings.mlr.press/v119/de-avila-belbute-peres20a.html. (pp. 10, 44, 46, 126, and 261)

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 1171–1179, 2015. URL https://proceedings.neurips.cc/paper/2015/hash/e995f98d56967d946471af29d7bf99f1-Abstract.html. (pp. 101, 114, and 245)

Marsha J. Berger and Joseph Oliger. Adaptive mesh refinement for hyperbolic partial differential equations. *Journal of Computational Physics*, 53(3):484–512, March 1984. (p. 127)

Luca Bertinetto, João F. Henriques, Philip H. S. Torr, and Andrea Vedaldi. Meta-learning with differentiable closed-form solvers. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=HyxnZh0ct7. (p. 222)

Dimitri P. Bertsekas. *Constrained Optimization and Lagrange Multiplier Methods*. Athena Scientific, 1996. (p. 72)

Alberto Bietti, Grégoire Mialon, Dexiong Chen, and Julien Mairal. A kernel perspective for regularizing deep neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 664–674. PMLR, 2019. URL http://proceedings.mlr.press/v97/bietti19a.html. (p. 101)

Massimo Bonavita and Peter Lean. 4D-Var for numerical weather prediction. *Weather*, 76 (2):65–66, 2021. (pp. 8 and 259)

Johannes Brandstetter, Daniel E. Worrall, and Max Welling. Message passing neural PDE solvers. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=vSix3HPYKSU. (pp. 10, 32, 36, 57, 58, 126, 128, 136, 245, and 261)

Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Velickovic. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *CoRR*, abs/2104.13478, 2021. URL https://arxiv.org/abs/2104.13478. (p. 30)

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html. (pp. 4, 8, 256, and 259)

Steven L. Brunton and J. Nathan Kutz. *Data-driven science and engineering: Machine learning, dynamical systems, and control*. Cambridge University Press, 2022. (pp. 9, 126, and 260)

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. doi: 10.1073/pnas.1517384113. URL https://www.pnas.org/doi/abs/10.1073/pnas.1517384113. (pp. 66, 71, and 84)

Keaton J. Burns, Geoffrey M. Vasil, Jeffrey S. Oishi, Daniel Lecoanet, and Benjamin P. Brown. Dedalus: A flexible framework for numerical simulations with spectral methods. *Physical Review Research*, 2(2), April 2020. (p. 243)

Rich Caruana. Multitask learning. pages 95–133, 1998. doi: 10.1007/978-1-4615-5529-2_5. URL https://doi.org/10.1007/978-1-4615-5529-2_5. (pp. 54 and 221)

Sangwon Chae, Joonhyeok Shin, Sungjun Kwon, Sangmok Lee, Sungwon Kang, and Donghyun Lee. PM10 and PM2.5 real-time prediction models using an interpolated convolutional neural network. *Scientific Reports*, 11, 2021. (p. 57)

Hao Chen, Bo He, Hanyu Wang, Yixuan Ren, Ser Nam Lim, and Abhinav Shrivastava. NeRV: Neural representations for videos. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann Dauphin, Percy Liang, and Jenn Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 21557–21568. Curran Associates, Inc., 2021. (p. 61)

Ricky T. Q. Chen. `torchdiffeq`, 2021. URL https://github.com/rtqichen/torchdiffeq. (p. 114)

Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 6572–6583. 2018. URL https://proceedings.neurips.cc/paper/2018/hash/69386f6bb1dfed68692a24c8686939b9-Abstract.html. (pp. 9, 10, 37, 38, 58, 66, 68, 75, 84, 97, 128, 186, 193, 195, 196, 245, 261, and 268)

Tianping Chen and Hong Chen. Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. *IEEE Trans. Neural Networks*, 6(4):911–917, 1995. doi: 10.1109/72.392253. URL https://doi.org/10.1109/72.392253. (p. 32)

Yutian Chen, Abram L. Friesen, Feryal M. P. Behbahani, Arnaud Doucet, David Budden, Matthew Hoffman, and Nando de Freitas. Modular meta-learning with shrinkage. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020a. URL https://proceedings.neurips.cc/paper/2020/hash/1e04b969bf040acd252e1faafb51f829-Abstract.html. (pp. 53 and 120)

Zhengdao Chen, Jianyu Zhang, Martín Arjovsky, and Léon Bottou. Symplectic recurrent neural networks. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020b. URL https://openreview.net/forum?id=BkgYPREtPr. (pp. 49, 66, and 104)

Lénaïc Chizat and Francis R. Bach. On the global convergence of gradient descent for over-parameterized models using optimal transport. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett,

editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 3040–3050, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/a1afc58c6ca9540d057299ec3016d726-Abstract.html. (p. 205)

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1724–1734, 2014. doi: 10.3115/v1/d14-1179. URL https://doi.org/10.3115/v1/d14-1179. (pp. 37, 97, 101, and 267)

Edward Choi, Mohammad Taha Bahadori, Jimeng Sun, Joshua Kulas, Andy Schuetz, and Walter F. Stewart. RETAIN: an interpretable predictive model for healthcare using reverse time attention mechanism. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 3504–3512, 2016. URL https://proceedings.neurips.cc/paper/2016/hash/231141b34c82aa95e48810a9d1b33a79-Abstract.html. (p. 66)

Youngsoo Choi, Peter Brown, William Arrighi, Robert Anderson, and Kevin Huynh. Space–time reduced order model for large-scale linear dynamical systems with application to boltzmann transport problems. *Journal of Computational Physics*, 424:109845, 2021. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2020.109845. URL https://www.sciencedirect.com/science/article/pii/S0021999120306197. (pp. 8 and 259)

Philippe Courtier, Jean-Noël Thépaut, and Anthony Hollingsworth. A strategy for operational implementation of 4D-Var, using an incremental approach. *Quarterly Journal of the Royal Meteorological Society*, 120(519):1367–1387, 1994. (pp. 8, 45, 105, and 259)

Miles D. Cranmer, Sam Greydanus, Stephan Hoyer, Peter W. Battaglia, David N. Spergel, and Shirley Ho. Lagrangian neural networks. *CoRR*, abs/2003.04630, 2020. URL https://arxiv.org/abs/2003.04630. (pp. 71 and 189)

Bryan C. Daniels and Ilya Nemenman. Efficient inference of parsimonious phenomenological models of cellular dynamics using s-systems and alternating regression. *PLOS ONE*, 10(3):1–14, 03 2015. (pp. 115 and 230)

Emmanuel de Bézenac, Arthur Pajot, and Patrick Gallinari. Deep learning for physical processes: Incorporating prior scientific knowledge. In *6th International Conference on*

Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings. OpenReview.net, 2018. URL https://openreview.net/forum?id=By4HsfWAZ. (pp. 9, 10, 36, 44, 49, 58, 66, 84, 104, 150, 247, 260, and 261)

Olivier Delestre. *Simulation du ruissellement d'eau de pluie sur des surfaces agricoles*. Theses, Université d'Orléans, July 2010. URL https://theses.hal.science/tel-00531377. (pp. 6 and 258)

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics, 2019. doi: 10.18653/v1/n19-1423. URL https://doi.org/10.18653/v1/n19-1423. (pp. 4, 8, 256, and 259)

Jérémie Donà, Jean-Yves Franceschi, Sylvain Lamprier, and Patrick Gallinari. PDE-driven spatiotemporal disentanglement. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=vLaHRtHvfFp. (pp. 66, 247, 248, and 249)

Jérémie Donà, Marie Déchelle, Patrick Gallinari, and Marina Levy. Constrained physical-statistics models for dynamical system identification and prediction. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*, 2022. URL https://openreview.net/forum?id=gbe1zHyA73. (pp. 50 and 151)

John R. Dormand and Peter J. Prince. A family of embedded runge-kutta formulae. *Journal of Computational and Applied Mathematics*, 6(1):19 – 26, 1980. ISSN 0377-0427. doi: https://doi.org/10.1016/0771-050X(80)90013-3. URL http://www.sciencedirect.com/science/article/pii/0771050X80900133. (pp. 96 and 187)

Emilien Dupont, Arnaud Doucet, and Yee Whye Teh. Augmented neural ODEs. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 3134–3144, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/21be9a4bd4f81549a9d1d241981cec3c-Abstract.html. (p. 38)

Emilien Dupont, Hyunjik Kim, S. M. Ali Eslami, Danilo Jimenez Rezende, and Dan Rosenbaum. From data to functa: Your data point is a function and you can treat it like

one. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 5694–5725. PMLR, July 2022. (p. 135)

Karthik Duraisamy, Gianluca Iaccarino, and Heng Xiao. Turbulence modeling in the age of data. *Annual Review of Fluid Mechanics*, 51:357–377, 2019. (p. 104)

Weinan E. A proposal on machine learning via dynamical systems. *Communications in Mathematics and Statistics*, 5(1):1–11, 2017. doi: 10.1007/s40304-017-0103-z. URL https://doi.org/10.1007/s40304-017-0103-z. (pp. 9 and 260)

Jeffrey L. Elman. Finding structure in time. *Cogn. Sci.*, 14(2):179–211, 1990. doi: 10.1207/s15516709cog1402_1. URL https://doi.org/10.1207/s15516709cog1402_1. (p. 36)

Rizal Fathony, Anit Kumar Sahu, Devin Willmott, and J. Zico Kolter. Multiplicative filter networks. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, 2021. URL https://openreview.net/forum?id=OmtmcPkkhT. (pp. 60, 135, 137, and 245)

Martin Feinberg. *Foundations of Chemical Reaction Network Theory*. Applied Mathematical Sciences. Springer Cham, 1 edition, 2019. (pp. 5 and 256)

Nicolas Ferry, Elisabeth Rémy, Pierre Brasseur, and Christophe Maes. The mercator global ocean operational analysis system: Assessment and validation of an 11-year reanalysis. *Journal of Marine Systems*, 65(1):540–560, 2007. ISSN 0924-7963. doi: https://doi.org/10.1016/j.jmarsys.2005.08.004. URL https://www.sciencedirect.com/science/article/pii/S0924796306003113. Marine Environmental Monitoring and Prediction. (pp. 8 and 259)

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135. PMLR, 2017. URL http://proceedings.mlr.press/v70/finn17a.html. (pp. 52, 53, 97, 105, 116, 221, 222, and 268)

Sebastian Flennerhag, Andrei A. Rusu, Razvan Pascanu, Francesco Visin, Hujun Yin, and Raia Hadsell. Meta-learning with warped gradient descent. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=rkeiQlBFPB. (p. 53)

James Fletcher and Warren Moors. Chebyshev sets. *Journal of the Australian Mathematical Society*, 98:161–231, 04 2014. doi: 10.1017/S1446788714000561. (p. 179)

Marco Fraccaro. *Deep Latent Variable Models for Sequential Data*. PhD thesis, Danmarks Tekniske Universitet, 2018. (p. 134)

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=rJl-b3RcF7. (p. 112)

Stefania Fresca, Andrea Manzoni, Luca Dedè, and Alfio Quarteroni. Deep learning-based reduced order models in cardiac electrophysiology. *PloS one*, 15(10):e0239416–e0239416, 10 2020. doi: 10.1371/journal.pone.0239416. URL https://pubmed.ncbi.nlm.nih.gov/33002014. (pp. 61, 84, and 104)

Ken-ichi Funahashi and Yuichi Nakamura. Approximation of dynamical systems by continuous time recurrent neural networks. *Neural Networks*, 6(6):801–806, 1993. ISSN 0893-6080. doi: https://doi.org/10.1016/S0893-6080(05)80125-X. URL https://www.sciencedirect.com/science/article/pii/S089360800580125X. (p. 37)

Joseph Galewsky, Richard K. Scott, and Lorenzo M. Polvani. An initial-value problem for testing numerical models of the global shallow-water equations. *Tellus A: Dynamic Meteorology and Oceanography*, 56(5):429–440, 2004. (pp. 136, 242, and 243)

Marta Garnelo, Dan Rosenbaum, Christopher Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo Jimenez Rezende, and S. M. Ali Eslami. Conditional neural processes. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 1690–1699. PMLR, 2018. URL http://proceedings.mlr.press/v80/garnelo18a.html. (pp. 53 and 113)

Pierre Gentine, Michael Pritchard, Stephan Rasp, Gael Reinaudi, and Galen Yacalis. Could machine learning break the convection parameterization deadlock? *Geophysical Research Letters*, 45(11):5742–5751, 2018. doi: https://doi.org/10.1029/2018GL078202. URL https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2018GL078202. (p. 66)

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning*

Research, pages 1263–1272. PMLR, 2017. URL http://proceedings.mlr.press/v70/gilmer17a.html. (p. 32)

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial networks. *Commun. ACM*, 63(11):139–144, 2020. doi: 10.1145/3422622. URL https://doi.org/10.1145/3422622. (pp. 4, 8, 256, and 259)

Somdatta Goswami, Aniruddha Bora, Yue Yu, and George Em Karniadakis. Physics-informed deep neural operator networks. *CoRR*, abs/2207.05748, 2022. doi: 10.48550/arXiv.2207.05748. URL https://doi.org/10.48550/arXiv.2207.05748. (p. 39)

Daniele Grattarola and Pierre Vandergheynst. Generalised implicit neural representations. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=2fD1Ux9InIW. (p. 150)

Samuel Greydanus, Misko Dzamba, and Jason Yosinski. Hamiltonian neural networks. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 15353–15363, 2019. URL https://proceedings.neurips.cc/paper/2019/file/26cd8ecadce0d4efd6cc8a8725cbd1f8-Paper.pdf. (pp. 49, 56, 66, 71, 74, 75, 104, 187, 188, and 196)

Guy Gur-Ari, Daniel A. Roberts, and Ethan Dyer. Gradient descent happens in a tiny subspace. *CoRR*, abs/1812.04754, 2018. URL http://arxiv.org/abs/1812.04754. (p. 112)

Madec Gurvan, Romain Bourdallé-Badie, Jérôme Chanut, Emanuela Clementi, Andrew Coward, Christian Ethé, Doroteaciro Iovino, Dan Lea, Claire Lévy, Tomas Lovato, Nicolas Martin, Sébastien Masson, Silvia Mocavero, Clément Rousset, Dave Storkey, Simon Müeller, George Nurser, Mike Bell, Guillaume Samson, Pierre Mathiot, Francesca Mele, and Aimie Moulin. NEMO ocean engine, March 2022. URL https://doi.org/10.5281/zenodo.6334656. (pp. 84 and 247)

David Ha, Andrew M. Dai, and Quoc V. Le. Hypernetworks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=rkpACe1lx. (pp. 56, 110, and 133)

Eldad Haber and Lars Ruthotto. Stable architectures for deep neural networks. *CoRR*, abs/1705.03341, 2017. URL http://arxiv.org/abs/1705.03341. (pp. 9 and 260)

Ernst Hairer, Syvert P. Nørsett, and Gerhard Wanner. *Solving Ordinary Differential Equations I: Nonstiff problems*. Springer, Berlin, second edition, 2000. (p. 114)

Charles R. Harris, K. Jarrod Millman, Stéfan van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585:357–362, 2020. doi: 10.1038/s41586-020-2649-2. URL https://doi.org/10.1038/s41586-020-2649-2. (p. 96)

David Haussler. Decision theoretic generalizations of the pac model for neural net and other learning applications. *Information and Computation*, 100(1):78 – 150, 1992. ISSN 0890-5401. doi: https://doi.org/10.1016/0890-5401(92)90010-D. URL http://www.sciencedirect.com/science/article/pii/089054019290010D. (pp. 93 and 206)

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. pages 770–778, 2016. doi: 10.1109/CVPR.2016.90. URL https://doi.org/10.1109/CVPR.2016.90. (pp. 4, 8, 9, 28, 58, 256, 259, 260, and 268)

Dan Hendrycks and Kevin Gimpel. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *CoRR*, abs/1606.08415, 2016. URL http://arxiv.org/abs/1606.08415. (p. 25)

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9 (8):1735–1780, 1997. doi: 10.1162/neco.1997.9.8.1735. URL https://doi.org/10.1162/neco.1997.9.8.1735. (pp. 37 and 268)

Masanobu Horie and Naoto Mitsume. Physics-embedded neural networks: E(n)-equivariant graph neural PDE solvers. *CoRR*, abs/2205.11912, 2022. doi: 10.48550/arXiv.2205.11912. URL https://doi.org/10.48550/arXiv.2205.11912. (p. 48)

Xiang Huang, Zhanhong Ye, Hongsheng Liu, Beiji Shi, Zidong Wang, Kang Yang, Yang Li, Bingya Weng, Min Wang, Haotian Chu, Jing Zhou, Fan Yu, Bei Hua, Lei Chen, and Bin Dong. Meta-auto-decoder for solving parametric partial differential equations. *CoRR*, abs/2111.08823, 2021. URL https://arxiv.org/abs/2111.08823. (p. 56)

Valerii Iakovlev, Markus Heinonen, and Harri Lähdesmäki. Learning continuous-time pdes from sparse data with graph neural networks. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=aUX5Plaq7Oy. (pp. 32, 58, and 150)

Haris Iqbal. Harisiqbal88/plotneuralnet v1.0.0, December 2018. URL https://doi.org/10.5281/zenodo.2526396. (pp. 29 and 30)

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 5967–5976. IEEE Computer Society, 2017. doi: 10.1109/CVPR.2017.632. URL https://doi.org/10.1109/CVPR.2017.632. (p. 56)

Gordon G. Johnson. A nonconvex set which has the unique nearest point property. *Journal of Approximation Theory*, 51(4):289–332, 1987. (p. 180)

Rudolf Emil Kalman. A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*, 82(1):35–45, 03 1960. ISSN 0021-9223. doi: 10.1115/1.3662552. URL https://doi.org/10.1115/1.3662552. (pp. 8, 45, 105, and 259)

Matthias Karlbauer, Timothy Praditia, Sebastian Otte, Sergey Oladyshkin, Wolfgang Nowak, and Martin V. Butz. Composing partial differential equations with physics-aware neural networks. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato, editors, *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 10773–10801. PMLR, 2022. URL https://proceedings.mlr.press/v162/karlbauer22a.html. (p. 149)

Patrick Kidger and Terry J. Lyons. Universal approximation with deep narrow networks. In Jacob D. Abernethy and Shivani Agarwal, editors, *Conference on Learning Theory, COLT 2020, 9-12 July 2020, Virtual Event [Graz, Austria]*, volume 125 of *Proceedings of Machine Learning Research*, pages 2306–2327. PMLR, 2020. URL http://proceedings.mlr.press/v125/kidger20a.html. (p. 205)

Patrick Kidger, James Morrill, James Foster, and Terry J. Lyons. Neural controlled differential equations for irregular time series. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/4a5876b450b45371f6cfe5047ac8cd45-Abstract.html. (p. 38)

Hyunjik Kim, Andriy Mnih, Jonathan Schwarz, Marta Garnelo, S. M. Ali Eslami, Dan Rosenbaum, Oriol Vinyals, and Yee Whye Teh. Attentive neural processes. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=SkE6PjC9KX. (pp. 113 and 132)

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980. (pp. 26, 102, 116, 232, and 245)

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=SJU4ayYgl. (p. 31)

Matthieu Kirchmeyer, Yuan Yin, Jérémie Donà, Nicolas Baskiotis, Alain Rakotomamonjy, and Patrick Gallinari. Generalizing to new physical systems via context-informed dynamics model. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato, editors, *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 11283–11301. PMLR, 2022. URL https://proceedings.mlr.press/v162/kirchmeyer22a.html. (pp. 14, 143, and 265)

Gene A. Klaasen and William C. Troy. Stationary wave solutions of a system of reaction-diffusion equations derived from the fitzhugh–nagumo equations. *SIAM Journal on Applied Mathematics*, 44(1):96–110, 1984. doi: 10.1137/0144008. (pp. 73 and 185)

Dmitrii Kochkov, Jamie A. Smith, Ayya Alieva, Qing Wang, Michael P. Brenner, and Stephan Hoyer. Machine learning–accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences*, 118(21):e2101784118, 2021. doi: 10.1073/pnas.2101784118. URL https://www.pnas.org/doi/abs/10.1073/pnas.2101784118. (pp. 10, 44, 47, 104, 126, 150, and 261)

Nikola B. Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew M. Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces. *CoRR*, abs/2108.08481, 2021. URL https://arxiv.org/abs/2108.08481. (pp. 34, 35, 58, and 268)

David Krueger, Ethan Caballero, Jörn-Henrik Jacobsen, Amy Zhang, Jonathan Binas, Dinghuai Zhang, Rémi Le Priol, and Aaron C. Courville. Out-of-distribution generalization via risk extrapolation (rex). In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 5815–5826. PMLR, 2021. URL http://proceedings.mlr.press/v139/krueger21a.html. (p. 51)

J. Nathan Kutz, Steven L. Brunton, Bingni W. Brunton, and Joshua L. Proctor. *Dynamic Mode Decomposition: Data-Driven Modeling of Complex Systems*. SIAM-Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2016. ISBN 1611974496. (pp. 8 and 259)

William Large and Stephen Yeager. Diurnal to decadal global forcing for ocean and sea-ice models: The data sets and flux climatologies, 05 2004. (p. 66)

Hervé Le Dret and Brigitte Lucquin. *Partial Differential Equations: Modeling, Analysis and Numerical Approximation*, chapter The Heat Equation, pages 219–251. International Series of Numerical Mathematics. Springer International Publishing, Cham, Switzerland, 2016. (p. 135)

Vincent Le Guen and Nicolas Thome. Disentangling physical dynamics from unknown factors for unsupervised video prediction. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 11471–11481. Computer Vision Foundation / IEEE, 2020. doi: 10.1109/CVPR42600.2020.01149. URL https://openaccess.thecvf.com/content_CVPR_2020/html/Le_Guen_Disentangling_Physical_Dynamics_From_Unknown_Factors_for_Unsupervised_Video_Prediction_CVPR_2020_paper.html. (pp. 48 and 188)

Kimin Lee, Younggyo Seo, Seunghyun Lee, Honglak Lee, and Jinwoo Shin. Context-aware dynamics model for generalization in model-based reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 5757–5766. PMLR, 2020. URL http://proceedings.mlr.press/v119/lee20g.html. (p. 105)

Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. Meta-learning with differentiable convex optimization. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 10657–10665. Computer Vision Foundation / IEEE, 2019. doi: 10.1109/CVPR.2019.01091. URL http://openaccess.thecvf.com/content_CVPR_2019/html/Lee_Meta-Learning_With_Differentiable_Convex_Optimization_CVPR_2019_paper.html. (p. 222)

Myoungkyu Lee, Nicholas Malaya, and Robert D. Moser. Petascale direct numerical simulation of turbulent channel flow on up to 786k cores. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, SC '13, New York, NY, USA, 2013. Association for Computing Machinery. ISBN 9781450323789. doi: 10.1145/2503210.2503298. URL https://doi.org/10.1145/2503210.2503298. (pp. 7 and 258)

Seungjun Lee, Haesang Yang, and Woojae Seong. Identifying physical law of hamiltonian systems via meta-learning. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=45NZvF1UHam. (p. 56)

Yoonho Lee and Seungjin Choi. Gradient-based meta-learning with learned layerwise metric and subspace. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 2933–2942. PMLR, 2018. URL http://proceedings.mlr.press/v80/lee18a.html. (p. 53)

Chunyuan Li, Heerad Farkhoor, Rosanne Liu, and Jason Yosinski. Measuring the intrinsic dimension of objective landscapes. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018a. URL https://openreview.net/forum?id=ryup8-WCW. (p. 112)

Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018b. (pp. 111 and 112)

Zhenguo Li, Fengwei Zhou, Fei Chen, and Hang Li. Meta-SGD: Learning to learn quickly for few shot learning. *CoRR*, abs/1707.09835, 2017. URL http://arxiv.org/abs/1707.09835. (pp. 53 and 116)

Zongyi Li, Nikola B. Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Andrew M. Stuart, Kaushik Bhattacharya, and Anima Anandkumar. Multipole graph neural operator for parametric partial differential equations. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/4b21cf96d4cf612f239a6c322b10c8fe-Abstract.html. (pp. 32, 58, 59, 126, and 267)

Zongyi Li, Nikola B. Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew M. Stuart, and Anima Anandkumar. Markov neural operators for learning chaotic systems. *CoRR*, abs/2106.06898, 2021a. URL https://arxiv.org/abs/2106.06898. (pp. 36, 59, 128, and 136)

Zongyi Li, Nikola Borislavov Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew M. Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In *9th International Conference on Learning*

Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021. OpenReview.net, 2021b. URL https://openreview.net/forum?id=c8P9NQVtmnO. (pp. 10, 34, 36, 59, 94, 95, 96, 102, 104, 116, 126, 136, 150, 211, 231, 242, 245, 261, and 267)

Zongyi Li, Hongkai Zheng, Nikola B. Kovachki, David Jin, Haoxuan Chen, Burigede Liu, Kamyar Azizzadenesheli, and Anima Anandkumar. Physics-informed neural operator for learning partial differential equations. *CoRR*, abs/2111.03794, 2021c. URL https://arxiv.org/abs/2111.03794. (p. 39)

Zongyi Li, Daniel Zhengyu Huang, Burigede Liu, and Anima Anandkumar. Fourier neural operator with learned deformations for pdes on general geometries. *CoRR*, abs/2207.05209, 2022. doi: 10.48550/arXiv.2207.05209. URL https://doi.org/10.48550/arXiv.2207.05209. (pp. 59 and 150)

Yun Long, Xueyuan She, and Saibal Mukhopadhyay. HybridNet: Integrating model-based and data-driven learning to predict evolution of dynamical systems. In *2nd Annual Conference on Robot Learning, CoRL 2018, Zürich, Switzerland, 29-31 October 2018, Proceedings*, volume 87 of *Proceedings of Machine Learning Research*, pages 551–560. PMLR, 2018a. URL http://proceedings.mlr.press/v87/long18a.html. (p. 49)

Zichao Long, Yiping Lu, Xianzhong Ma, and Bin Dong. PDE-Net: Learning PDEs from data. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 3214–3222. PMLR, 2018b. URL http://proceedings.mlr.press/v80/long18a.html. (pp. 9, 36, 47, 58, 66, 84, 112, and 260)

Zichao Long, Yiping Lu, and Bin Dong. PDE-Net 2.0: Learning PDEs from data with a numeric-symbolic hybrid deep network. *J. Comput. Phys.*, 399, 2019. doi: 10.1016/j.jcp.2019.108925. URL https://doi.org/10.1016/j.jcp.2019.108925. (pp. 45 and 102)

Alfred James Lotka. Elements of physical biology. *Nature*, 116(2917):461–461, 1925. (pp. 94, 115, and 229)

Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nat. Mach. Intell.*, 3(3):218–229, 2021. doi: 10.1038/s42256-021-00302-5. URL https://doi.org/10.1038/s42256-021-00302-5. (pp. 32, 58, 126, 128, and 137)

Maia Martcheva. *An Introduction to Mathematical Epidemiology*. Texts in Applied Mathematics. Springer New York, NY, 1 edition, october 2015. (pp. 5 and 256)

Viraj Mehta, Ian Char, Willie Neiswanger, Youngseog Chung, Andrew Oakleigh Nelson, Mark D. Boyer, Egemen Kolemen, and Jeff G. Schneider. Neural dynamical systems: Balancing structure and flexibility in physical prediction. In *2021 60th IEEE Conference on Decision and Control (CDC), Austin, TX, USA, December 14-17, 2021*, pages 3735–3742. IEEE, 2021. doi: 10.1109/CDC45484.2021.9682807. URL https://doi.org/10.1109/CDC45484.2021.9682807. (pp. 75, 78, 188, and 190)

Léon Migus, Yuan Yin, Jocelyn Ahmed Mazari, and Patrick Gallinari. Multi-scale physical representations for approximating pde solutions with graph neural operators. In Alexander Cloninger, Timothy Doster, Tegan Emerson, Manohar Kaul, Ira Ktena, Henry Kvinge, Nina Miolane, Bastian Rieck, Sarah Tymochko, and Guy Wolf, editors, *Proceedings of Topological, Algebraic, and Geometric Learning Workshops 2022*, volume 196 of *Proceedings of Machine Learning Research*, pages 332–339. PMLR, 2022. URL https://proceedings.mlr.press/v196/migus22a.html. (p. 16)

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part I*, pages 405–421, 2020. doi: 10.1007/978-3-030-58452-8_24. URL https://doi.org/10.1007/978-3-030-58452-8_24. (p. 127)

Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=B1DmUzWAW. (pp. 53, 105, and 120)

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. 2018. URL https://openreview.net/forum?id=B1QRgziT-. (p. 101)

Anusha Nagabandi, Ignasi Clavera, Simin Liu, Ronald S. Fearing, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=HyztsoC5Y7. (p. 105)

Vaishnavh Nagarajan and J. Zico Kolter. Uniform convergence may be unable to explain generalization in deep learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/05e97c207235d63ceb1db43c60db7bbb-Paper.pdf. (p. 93)

Aurel Neic, Fernando Otaviano Campos, Anton J. Prassl, Steven A. Niederer, Martin J. Bishop, Edward J. Vigmond, and Gernot Plank. Efficient computation of electrograms and ecgs in human whole heart simulations using a reaction-eikonal model. *Journal of Computational Physics*, 346:191–211, 2017. doi: 10.1016/j.jcp.2017.06.020. URL https://doi.org/10.1016/j.jcp.2017.06.020. (pp. 84 and 104)

Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *CoRR*, abs/1803.02999, 2018. URL http://arxiv.org/abs/1803.02999. (p. 53)

Alexander Norcliffe, Cristian Bodnar, Ben Day, Jacob Moss, and Pietro Liò. Neural ODE processes. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=27acGyyI1BY. (p. 38)

Peter J. Olver. *Introduction to partial differential equations*. Undergraduate Texts in Mathematics. Springer Cham, 2014. (p. 126)

Boris N. Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=r1ecqn4YwB. (pp. 66, 195, and 196)

Jeong Joon Park, Peter Florence, Julian Straub, Richard A. Newcombe, and Steven Lovegrove. DeepSDF: Learning continuous signed distance functions for shape representation. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 165–174. Computer Vision Foundation / IEEE, 2019. doi: 10.1109/CVPR.2019.00025. URL http://openaccess.thecvf.com/content_CVPR_2019/html/Park_DeepSDF_Learning_Continuous_Signed_Distance_Functions_for_Shape_Representation_CVPR_2019_paper.html. (pp. 53 and 132)

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Z. Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 8024–8035. 2019. URL https://proceedings.neurips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html. (pp. 185 and 244)

Barak A. Pearlmutter. Learning state space trajectories in recurrent neural networks. *Neural Computation*, 1(2):263–269, 1989. doi: 10.1162/neco.1989.1.2.263. (p. 37)

John E. Pearson. Complex patterns in a simple system. *Science*, 261(5118):189–192, 1993. doi: 10.1126/science.261.5118.189. URL https://www.science.org/doi/abs/10.1126/science.261.5118.189. (pp. 5, 94, 95, 115, 230, and 256)

Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron C. Courville. FiLM: Visual reasoning with a general conditioning layer. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 3942–3951, 2018. URL https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16528. (pp. 116, 135, and 223)

Pascal Pernot and Fabien Cailliez. A critical review of statistical calibration/prediction models handling data inconsistency and model inadequacy. *AIChE Journal*, 63(10):4642–4665, 2017. (p. 45)

Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W. Battaglia. Learning mesh-based simulation with graph networks. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, 2021. URL https://openreview.net/forum?id=roNqYL0_XP. (pp. 10, 36, 58, 126, 150, and 261)

Michael Prasthofer, Tim De Ryck, and Siddhartha Mishra. Variable-input deep operator networks. *arXiv preprint arXiv:2205.11404*, 2022. (pp. 58, 126, and 151)

Dimitris C. Psichogios and Lyle H. Ungar. A hybrid neural network-first principles approach to process modeling. *AIChE Journal*, 38(10):1499–1511, 1992. (p. 49)

Christopher Rackauckas, Yingbo Ma, Julius Martensen, Collin Warner, Kirill Zubov, Rohit Supekar, Dominic Skinner, and Ali Jasim Ramadhan. Universal differential equations for scientific machine learning. *CoRR*, abs/2001.04385, 2020. URL https://arxiv.org/abs/2001.04385. (p. 38)

Aniruddh Raghu, Maithra Raghu, Samy Bengio, and Oriol Vinyals. Rapid learning or feature reuse? towards understanding the effectiveness of MAML. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=rkgMkCEtPB. (pp. 53, 120, and 221)

Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Multistep Neural Networks for Data-driven Discovery of Nonlinear Dynamical Systems. *arXiv e-prints*, art. arXiv:1801.01236, January 2018. (p. 37)

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. doi: 10.1016/j.jcp.2018.10.045. URL https://doi.org/10.1016/j.jcp.2018.10.045. (pp. 9, 10, 38, 60, 66, 70, 84, 102, 112, 126, 128, 152, 260, 261, and 268)

Prajit Ramachandran, Barret Zoph, and Quoc V. Le. Searching for activation functions. *CoRR*, abs/1710.05941, 2017. URL http://arxiv.org/abs/1710.05941. (pp. 25, 101, and 232)

Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Learning multiple visual domains with residual adapters. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 506–516, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/e7b24b112a44fdd9ee93bdf998c6ca0e-Abstract.html. (p. 54)

Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Efficient parametrization of multi-domain deep neural networks. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 8119–8127. Computer Vision Foundation / IEEE Computer Society, 2018. doi: 10.1109/CVPR.2018.00847. URL http://openaccess.thecvf.com/content_cvpr_2018/html/Rebuffi_Efficient_Parametrization_of_CVPR_2018_paper.html. (p. 54)

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. doi: 10.1038/s41586-019-0912-1. URL https://doi.org/10.1038/s41586-019-0912-1. (pp. 67, 84, and 104)

James Requeima, Jonathan Gordon, John Bronskill, Sebastian Nowozin, and Richard E. Turner. Fast and flexible multi-task classification using conditional neural adaptive processes. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 7957–7968. 2019. URL https://proceedings.neurips.cc/paper/2019/file/1138d90ef0a0848a542e57d1595f58ea-Paper.pdf. (p. 54)

Ramiro Rico-Martínez and Ioannis G. Kevrekidis. Continuous time modeling of nonlinear systems: a neural network-based approach. In *Proceedings of International Conference on Neural Networks (ICNN'88), San Francisco, CA, USA, March 28 - April 1, 1993*, pages

1522–1525. IEEE, 1993. doi: 10.1109/ICNN.1993.298782. URL https://doi.org/10.1109/ICNN.1993.298782. (pp. 9, 37, and 260)

Ramiro Rico-Martinez, J. S. Anderson, and Ioannis G. Kevrekidis. Continuous-time nonlinear signal processing: a neural network based approach for gray box identification. In *Proceedings of IEEE Workshop on Neural Networks for Signal Processing*, pages 596–605. IEEE, 1994. doi: 10.1109/NNSP.1994.366006. (p. 49)

David Rolnick, Priya L. Donti, Lynn H. Kaack, Kelly Kochanski, Alexandre Lacoste, Kris Sankaran, Andrew Slavin Ross, Nikola Milojevic-Dupont, Natasha Jaques, Anna Waldman-Brown, et al. Tackling climate change with machine learning. In *NeurIPS 2019 workshop on Climate Change with Machine Learning*, 2019. (p. 66)

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells III, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2015 - 18th International Conference Munich, Germany, October 5 - 9, 2015, Proceedings, Part III*, volume 9351 of *Lecture Notes in Computer Science*, pages 234–241. Springer, 2015. doi: 10.1007/978-3-319-24574-4_28. URL https://doi.org/10.1007/978-3-319-24574-4_28. (pp. 28 and 29)

Sebastian Ruder. An overview of multi-task learning in deep neural networks. *CoRR*, abs/1706.05098, 2017. URL http://arxiv.org/abs/1706.05098. (p. 221)

Andrei A. Rusu, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, and Raia Hadsell. Meta-learning with latent embedding optimization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=BJgklhAcK7. (p. 116)

Lars Ruthotto and Eldad Haber. Deep neural networks motivated by partial differential equations. *J. Math. Imaging Vis.*, 62(3):352–364, 2020. doi: 10.1007/s10851-019-00903-1. URL https://doi.org/10.1007/s10851-019-00903-1. (pp. 9 and 260)

Priyabrata Saha, Saurabh Dash, and Saibal Mukhopadhyay. Physics-incorporated convolutional recurrent neural networks for source identification and forecasting of dynamical systems. *Neural Networks*, 144:359–371, 2021. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2021.08.033. URL https://www.sciencedirect.com/science/article/pii/S0893608021003464. (p. 49)

Günter Schneckenreither, Niki Popper, Günther Zauner, and Felix Breitenecker. Modelling SIR-type epidemics by ODEs, PDEs, difference equations and cellular automata – A comparative study. *Simulation Modelling Practice and Theory*, 16(8):1014–1023, 2008.

ISSN 1569-190X. doi: https://doi.org/10.1016/j.simpat.2008.05.015. URL https://www.sciencedirect.com/science/article/pii/S1569190X08001160. EUROSIM 2007. (pp. 7 and 258)

Sungyong Seo, Chuizheng Meng, and Yan Liu. Physics-aware difference graph networks for sparsely-observed dynamics. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=r1gelyrtwH. (p. 66)

Sagi Shaier, Maziar Raissi, and SeshaiyerPadmanabhan. Data-driven approaches for predicting spread of infectious diseases through dinns: Disease informed neural networks. *Letters in Biomathematics*, 9(1):71–105, Aug. 2022. URL https://lettersinbiomath.journals.publicknowledgeproject.org/index.php/lib/article/view/513. (p. 104)

Shai Shalev-Shwartz and Shai Ben-David. *Covering Numbers*, pages 337–340. Cambridge University Press, 2014. doi: 10.1017/CBO9781107298019.028. (p. 200)

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. In Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 802–810, 2015. URL https://proceedings.neurips.cc/paper/2015/hash/07563a3fe3bbe7e3ba84431ad9d055af-Abstract.html. (pp. 37, 66, and 267)

Justin A. Sirignano and Konstantinos Spiliopoulos. DGM: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics*, 375:1339–1364, 2018. doi: 10.1016/j.jcp.2018.08.029. URL https://doi.org/10.1016/j.jcp.2018.08.029. (pp. 10, 38, 66, 75, 104, 126, 261, and 267)

Vincent Sitzmann, Michael Zollhoefer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/b5dc4e5d9b495d0196f61d45b26ef33e-Paper.pdf. (pp. 127, 137, 245, and 246)

Vincent Sitzmann, Julien N. P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. 2020. URL https://proceedings.neurips.cc/paper/2020/hash/53c04118df112c13a8c34b38343b9c10-Abstract.html. (p. 59)

Ivan Skorokhodov, Sergey Tulyakov, and Mohamed Elhoseiny. StyleGAN-V: A continuous video generator with the price, image quality and perks of StyleGAN2. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3626–3636, June 2022. (p. 61)

Sigurd Spieckermann, Siegmund Düll, Steffen Udluft, Alexander Hentschel, and Thomas Runkler. Exploiting similarity in system identification tasks with recurrent neural networks. *Neurocomputing*, 169:343 – 349, 2015. ISSN 0925-2312. Learning for Visual Semantic Understanding in Big Data ESANN 2014 Industrial Data Processing and Analysis. (pp. 56 and 97)

George Gabriel Stokes. On the effect of the internal friction of fluids on the motion of pendulums. *Transactions of the Cambridge Philosophical Society*, 9(2):8–106, 1851. (pp. 5, 115, 136, 231, 241, and 256)

Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 7537–7547. Curran Associates, Inc., 2020. (pp. 60 and 127)

Damien Teney, Ehsan Abbasnejad, and Anton van den Hengel. Unshuffling data for improved generalization in visual question answering. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, pages 1397–1407. IEEE, 2021. doi: 10.1109/ICCV48922.2021.00145. URL https://doi.org/10.1109/ICCV48922.2021.00145. (p. 51)

Michael L. Thompson and Mark A. Kramer. Modeling chemical processes using prior knowledge and neural networks. *AIChE Journal*, 40(8):1328–1340, 1994. (p. 49)

Sebastian Thrun and Lorien Y. Pratt. Learning to learn: Introduction and overview. In Sebastian Thrun and Lorien Y. Pratt, editors, *Learning to Learn*, pages 3–17. Springer, 1998. ISBN 978-1-4613-7527-2. doi: 10.1007/978-1-4615-5529-2_1. URL https://doi.org/10.1007/978-1-4615-5529-2_1. (p. 105)

Peter Toth, Danilo J. Rezende, Andrew Jaegle, Sébastien Racanière, Aleksandar Botev, and Irina Higgins. Hamiltonian generative networks. 2020. URL https://openreview.net/forum?id=HJenn6VFvB. (pp. 75 and 187)

Jean-François Toubeau, Jérémie Bottieau, François Vallée, and Zacharie De Grève. Deep learning-based multivariate probabilistic forecasting for short-term scheduling in power markets. *IEEE Transactions on Power Systems*, 34(2):1203–1215, 2018. (p. 66)

Kiwon Um, Robert Brand, Yun (Raymond) Fei, Philipp Holl, and Nils Thuerey. Solver-in-the-Loop: Learning from differentiable physics to interact with iterative PDE-solvers. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/43e4e6a6f341e00671e123714de019a8-Abstract.html. (pp. 44 and 46)

Benjamin Ummenhofer, Lukas Prantl, Nils Thuerey, and Vladlen Koltun. Lagrangian fluid simulation with continuous convolutions. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=B1lDoJSYDH. (p. 66)

Thijs Vogels, Sai Praneeth Karimireddy, and Martin Jaggi. Powersgd: Practical low-rank gradient compression for distributed optimization. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/d9fbed9da256e344c1fa46bb46c34c5f-Paper.pdf. (p. 112)

Nils Wandel, Michael Weinmann, and Reinhard Klein. Learning incompressible fluid dynamics from scratch - towards fast, differentiable fluid models that generalize. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, 2021. URL https://openreview.net/forum?id=KUDUoRsEphu. (pp. 39 and 104)

Haoxiang Wang, Han Zhao, and Bo Li. Bridging multi-task learning and meta-learning: Towards efficient training and effective adaptation. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 10991–11002. PMLR, 2021a. URL http://proceedings.mlr.press/v139/wang21ad.html. (p. 54)

Jindong Wang, Cuiling Lan, Chang Liu, Yidong Ouyang, and Tao Qin. Generalizing to unseen domains: A survey on domain generalization. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*, pages 4627–4635, 2021b. doi: 10.24963/ijcai.2021/628. URL https://doi.org/10.24963/ijcai.2021/628. (pp. 105 and 149)

Qi Wang, Feng Li, Yi Tang, and Yan Xu. Integrating model-driven and data-driven methods for power system frequency stability assessment and control. *IEEE Transactions on Power Systems*, 34(6):4557–4568, 2019. (pp. 75, 78, 188, and 190)

Rui Wang, Robin Walters, and Rose Yu. Incorporating symmetry into deep dynamics models for improved generalization. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021c. URL https://openreview.net/forum?id=wta_8Hx2KD. (p. 48)

Rui Wang, Robin Walters, and Rose Yu. Meta-learning dynamics forecasting using task inference. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*. OpenReview.net, 2022. URL https://openreview.net/forum?id=BsSP7pZGFQO. (pp. 56 and 105)

Sifan Wang and Paris Perdikaris. Long-time integration of parametric evolution equations with physics-informed DeepONets. *CoRR*, abs/2106.05384, 2021. URL https://arxiv.org/abs/2106.05384. (pp. 39 and 137)

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 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 879–888, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/e5f6ad6ce374177eef023bf5d0c018b6-Abstract.html. (pp. 37, 97, and 101)

Yunbo Wang, Zhifeng Gao, Mingsheng Long, Jianmin Wang, and Philip S. Yu. PredRNN++: Towards A resolution of the deep-in-time dilemma in spatiotemporal predictive learning. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 5110–5119. PMLR, 2018. URL http://proceedings.mlr.press/v80/wang18b.html. (pp. 37, 66, and 75)

Maurice Weiler and Gabriele Cesa. General e(2)-equivariant steerable cnns. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 14334–14345, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/45d6637b718d0f24a237069fe41b0db4-Abstract.html. (p. 48)

Hilary Weller, John Thuburn, and Colin J. Cotter. Computational modes and grid imprinting on five quasi-uniform spherical C grids. *Monthly Weather Review*, 140(8): 2734–2755, 2012. (p. 142)

Jared Willard, Xiaowei Jia, Shaoming Xu, Michael S. Steinbach, and Vipin Kumar. Integrating scientific knowledge with machine learning for engineering and environmental

systems. *ACM Computing Surveys*, 55(4):66:1–66:37, 2023. doi: 10.1145/3514228. URL https://doi.org/10.1145/3514228. (pp. 84 and 126)

Wilson Yan, Yunzhi Zhang, Pieter Abbeel, and Aravind Srinivas. VideoGPT: Video generation using VQ-VAE and transformers. *arXiv preprint arXiv:2104.10157*, 2021. (p. 134)

Yongxin Yang and Timothy M. Hospedales. Trace norm regularised deep multi-task learning. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=rknkNR7Ke. (p. 54)

Yuan Yin, Arthur Pajot, Emmanuel de Bézenac, and Patrick Gallinari. Unsupervised inpainting for occluded sea surface temperature sequences. In Julien Brajard, Anastase Charantonis, Chen Chen, and Jakob Runge, editors, *Proceedings of the 9th International Workshop on Climate Informatics: CI 2019*, Dec. 2019. doi: 10.5065/y82j-f154. (p. 16)

Yuan Yin, Arthur Pajot, Emmanuel de Bézenac, and Patrick Gallinari. Unsupervised spatiotemporal data inpainting, 2020. URL https://openreview.net/forum?id=rylqmxBKvH. (p. 16)

Yuan Yin, Ibrahim Ayed, Emmanuel de Bézenac, Nicolas Baskiotis, and Patrick Gallinari. LEADS: learning dynamical systems that generalize across environments. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 7561–7573, 2021a. URL https://proceedings.neurips.cc/paper/2021/hash/3df1d4b96d8976ff5986393e8767f5b2-Abstract.html. (pp. 13, 105, 116, 221, 232, and 264)

Yuan Yin, Vincent Le Guen, Jérémie Donà, Emmanuel de Bézenac, Ibrahim Ayed, Nicolas Thome, and Patrick Gallinari. Augmenting physical models with deep networks for complex dynamics forecasting. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021b. URL https://openreview.net/forum?id=kmG8vRXTFv. (pp. 13, 45, 58, 84, 101, 102, 104, 112, 126, and 264)

Yuan Yin, Matthieu Kirchmeyer, Jean-Yves Franceschi, Alain Rakotomamonjy, and Patrick Gallinari. Continuous PDE dynamics forecasting with implicit neural representations. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*, 2023. URL https://openreview.net/forum?id=B73niNjbPs. (pp. 15 and 265)

Sihyun Yu, Jihoon Tack, Sangwoo Mo, Hyunsu Kim, Junho Kim, Jung-Woo Ha, and Jinwoo Shin. Generating videos with dynamics-aware implicit generative adversarial networks. In *International Conference on Learning Representations*, 2022. (p. 61)

Çağatay Yıldız, Markus Heinonen, and Harri Lähdesmäki. ODE^{2}VAE: deep generative second order ODEs with bayesian neural networks, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/99a401435dcb65c4008d3ad22c8cdad0-Abstract.html. (p. 248)

Laure Zanna and Thomas Bolton. *Deep Learning of Unresolved Turbulent Ocean Processes in Climate Models*, chapter 20, pages 298–306. John Wiley & Sons, Ltd, 2021. (p. 127)

Luisa M. Zintgraf, Kyriacos Shiarlis, Vitaly Kurin, Katja Hofmann, and Shimon Whiteson. Fast context adaptation via meta-learning. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 7693–7702. PMLR, 2019. URL http://proceedings.mlr.press/v97/zintgraf19a.html. (pp. 53, 56, 113, 116, and 221)

Appendix A

Appendix of Chapter 4

A.1 Reminder on Proximinal and Chebyshev Sets

We begin by giving a definition of proximinal and Chebyshev sets, taken from Fletcher and Moors (2014):

Definition A.1. A *proximinal set* of a normed space $(E, \|\cdot\|)$ is a subset $\mathcal{C} \subset E$ such that every $x \in E$ admits at least a nearest point in $\mathcal{C}$.

Definition A.2. A *Chebyshev set* of a normed space $(E, \|\cdot\|)$ is a subset $\mathcal{C} \subset E$ such that every $x \in E$ admits a unique nearest point in $\mathcal{C}$.

Proximinality reduces to a compacity condition in finite-dimensional spaces. In general, it is a weaker one: boundedly compact sets verify this property for example.

In Euclidean spaces, Chebyshev sets are simply closed convex subsets. The question of knowing whether it is the case that all Chebyshev sets are closed convex sets in infinite dimensional Hilbert spaces is still an open question. In general, there exist examples of non-convex Chebyshev sets, a famous one being presented in Johnson (1987) for a non-complete inner-product space.

Given the importance of this topic in approximation theory, finding the necessary conditions for a set to be Chebyshev and studying the properties of those sets have been the subject of many efforts. Some of those properties are summarized below:

- The metric projection on a boundedly compact Chebyshev set is continuous.
- If the norm is strict, every closed convex space, in particular any finite-dimensional subspace is Chebyshev.
- In a Hilbert space, every closed convex set is Chebyshev.

A.2 Proof of Propositions 4.1 and 4.2

We prove the following result which implies propositions 4.1 and 4.2 in the article:

Proposition A.1. *The optimization problem:*

$$\min_{f_P \in \mathcal{F}_P, f_A \in \mathcal{F}} \|f_A\| \text{ subject to } \forall u \in \mathcal{D}, t \in \mathcal{T}, \frac{\mathrm{d}u_t}{\mathrm{d}t} = (f_P + f_A)(u_t) \tag{A.1}$$

is equivalent a metric projection onto $\mathcal{F}_P$.

If $\mathcal{F}_P$ is proximinal, Eq. (A.1) admits a minimizing pair.

If $\mathcal{F}_P$ is Chebyshev, Eq. (A.1) admits a unique minimizing pair which f_P is the metric projection.

Proof. The idea is to reconstruct the full functional from the trajectories of $\mathcal{D}$. By

definition, $\mathcal{U}^{\mathcal{D}}$ is the set of points reached by trajectories in $\mathcal{D}$ so that:

$$\mathcal{U}^{\mathcal{D}} = \{u(t) \in \mathbb{R}^d \mid \exists u \in \mathcal{D}, \exists t \in \mathcal{T}\}$$

Then let us define a function $f^{\mathcal{D}}$ in the following way: For $u_t \in \mathcal{U}^{\mathcal{D}}$, we can find $u \in \mathcal{D}$ and t. Differentiating u at t, which is possible by definition of $\mathcal{D}$, we take:

$$f^{\mathcal{D}}(u(t)) = \frac{\mathrm{d}u}{\mathrm{d}t}(t)$$

For any $(f_{\mathrm{P}}, f_{\mathrm{A}})$ satisfying the constraint in Eq. (A.1), we then have that $(f_{\mathrm{P}} + f_{\mathrm{A}})(u(t)) = \frac{\mathrm{d}u}{\mathrm{d}t}(t) = f^{\mathcal{D}}(u(t))$ for all $u(t) \in \mathcal{U}$. Conversely, any pair such that $(f_{\mathrm{P}}, f_{\mathrm{A}}) \in \mathcal{F}_{\mathrm{P}} \times \mathcal{F}$ and $f_{\mathrm{P}} + f_{\mathrm{A}} = f^{\mathcal{D}}$, verifies the constraint.

Thus we have the equivalence between Eq. (A.1) and the metric projection formulated as:

$$\min_{f_{\mathrm{P}} \in \mathcal{F}_{\mathrm{P}}} \left\| f^{\mathcal{D}} - f_{\mathrm{P}} \right\| \tag{A.2}$$

If $\mathcal{F}_{\mathrm{P}}$ is proximinal, the projection problem admits a solution which we denote $f_{\mathrm{P}}^{\star}$. Taking $f_{\mathrm{A}}^{\star} = f^{\mathcal{D}} - f_{\mathrm{P}}^{\star}$, we have that $f_{\mathrm{P}}^{\star} + f_{\mathrm{A}}^{\star} = f^{\mathcal{D}}$ so that $(f_{\mathrm{P}}^{\star}, f_{\mathrm{A}}^{\star})$ verifies the constraint of Eq. (4.2). Moreover, if there is $(f_{\mathrm{P}}, f_{\mathrm{A}})$ satisfying the constraint of Eq. (4.2), we have that $f_{\mathrm{P}} + f_{\mathrm{A}} = f^{\mathcal{D}}$ by what was shown above and $\|f_{\mathrm{A}}\| = \|f^{\mathcal{D}} - f_{\mathrm{P}}\| \geq \|f^{\mathcal{D}} - f_{\mathrm{P}}^{\star}\|$ by definition of $f_{\mathrm{P}}^{\star}$. This shows that $(f_{\mathrm{P}}^{\star}, f_{\mathrm{A}}^{\star})$ is minimal.

Moreover, if $\mathcal{F}_{\mathrm{P}}$ is a Chebyshev set, by uniqueness of the projection, if $f_{\mathrm{P}} \neq f_{\mathrm{P}}^{\star}$ then $\|f_{\mathrm{A}}\| > \|f_{\mathrm{A}}^{\star}\|$. Thus the minimal pair is unique.

□

A.3 Parameter Estimation in Incomplete Physical Models

Classically, when a set $\mathcal{F}_{\mathrm{P}} \subset \mathcal{F}$ summarizing the most important properties of a system is available, this gives a *simplified model* of the true dynamics and the adopted problem is then to fit the trajectories using this model as well as possible, solving:

$$\begin{gathered} \min_{f_{\mathrm{P}} \in \mathcal{F}_{\mathrm{P}}} \mathbb{E}_{u \sim \mathcal{D}} l(\tilde{u}, u) \\ \text{subject to} \quad \forall u_0 \in \mathcal{I} \subset \mathcal{U}, \text{ and } \forall t \in \mathcal{T}, \ \frac{\mathrm{d}\tilde{u}_t}{\mathrm{d}t} = f_{\mathrm{P}}(\tilde{u}_t) \end{gathered} \tag{A.3}$$

where l is a discrepancy measure between trajectories. Recall that u is the resulting trajectory of an ODE solver taking u_0 as initial condition. In other words, we try to find

a function f_{P} which gives trajectories as close as possible to the ones from the dataset. While the estimation of the function becomes easier, there is then a residual part that is left unexplained and this can be a non-negligible issue in at least two ways:

- When $f \notin \mathcal{F}_{\text{P}}$, the loss is strictly positive at the minimum. This means that reducing the space of functions $\mathcal{F}_{\text{P}}$ makes us lose in terms of accuracy.[1]

- The obtained function f_{P} might not even be the most meaningful function from $\mathcal{F}_{\text{P}}$ as it would try to capture phenomena that are not explainable with functions in $\mathcal{F}_{\text{P}}$, thus giving the wrong bias to the calculated function. For example, if one is considering a dampened periodic trajectory where only the period can be learned in $\mathcal{F}_{\text{P}}$ but not the dampening, the estimated period will account for the dampening and will thus be biased.

This is confirmed in the paper in Section 4.3: the incomplete physical models augmented with APHYNITY get different and experimentally better physical identification results than the physical models alone.

Let us compare our approach with this one on the linearized damped pendulum to show how estimates of physical parameters can differ. The equation is the following:

$$\frac{\mathrm{d}^2\alpha}{\mathrm{d}t^2} + \omega_0^2 \sin\alpha + \gamma\frac{\mathrm{d}\alpha}{\mathrm{d}t} = 0$$

For simplicity, let us take $\alpha_0 \ll 1$, the equation is then reduced to

$$\frac{\mathrm{d}^2\alpha}{\mathrm{d}t^2} + \omega_0^2\alpha + \gamma\frac{\mathrm{d}\alpha}{\mathrm{d}t} = 0$$

We take the same notations as in the article and parametrize the simplified physical models as:

$$f_{\text{P}}^{\omega_0} : \left(\alpha_t, \frac{\mathrm{d}\alpha_t}{\mathrm{d}t}\right) \mapsto \left(\frac{\mathrm{d}\alpha_t}{\mathrm{d}t}, -\omega_0^2\alpha_t\right)$$

The corresponding prediction for an initial state $u_0 = (\alpha_0, \frac{\mathrm{d}\alpha_0}{\mathrm{d}t}) = (\alpha_0, 0)$ can then written explicitly as:

$$\hat{\alpha}_t = \alpha_0 \cos\omega_0 t$$

Let us consider damped pendulum solutions u written as:

$$\alpha_t = \alpha_0 e^{-t}\cos t$$

[1] This is true in theory, although not necessarily in practice when f overfits a small dataset.

which corresponds to the system where $\omega_0^2 = \gamma = 2$:

$$f : \left(\alpha_t, \frac{\mathrm{d}\alpha_t}{\mathrm{d}t}\right) \mapsto \left(\frac{\mathrm{d}\alpha_t}{\mathrm{d}t}, -2\alpha_t - 2\frac{\mathrm{d}\alpha_t}{\mathrm{d}t}\right)$$

It is then easy to see that the estimate of a with the physical model alone can be obtained by minimizing:

$$\int_0^T |\hat{\alpha}_t - \alpha_t|^2 = \int_0^T |\cos \omega_0 t - e^{-t} \cos t|^2$$

This expression depends on T and thus, depending on the chosen time interval and the way the integral is discretized will almost always give biased estimates. In other words, the estimated value of a will not give us the desired solution $t \mapsto \cos t$.

On the other hand, for a given ω_0, in the APHYNITY framework, the residual must be equal to:

$$f_{\mathrm{A}} : \left(\alpha_t, \frac{\mathrm{d}\alpha_t}{\mathrm{d}t}\right) \mapsto \left(0, -(2 - \omega_0^2)\alpha_t - 2\frac{\mathrm{d}\alpha_t}{\mathrm{d}t}\right)$$

in order to satisfy the fitting constraint. Minimizing its norm, we obtain $\omega_0^2 = 2$ which gives us $f = f_{\mathrm{A}} + f_{\mathrm{P}}^{\omega_0=\sqrt{2}}$ and the desired solution for α_0:

$$\alpha_t = \alpha_0 e^{-t} \cos t$$

with the right period.

A.4 Discussion on Supervision over Derivatives

To find the appropriate decomposition $(f_{\mathrm{P}}, f_{\mathrm{A}})$, we use a trajectory-based error by solving:

$$\begin{aligned} &\min_{f_{\mathrm{P}} \in \mathcal{F}_{\mathrm{P}}, f_{\mathrm{A}} \in \mathcal{F}} \|f_{\mathrm{A}}\| \\ \text{subject to} \quad &\forall u_0 \in \mathcal{I} \subset \mathcal{U}, \forall t \in \mathcal{T}, \ \frac{\mathrm{d}\tilde{u}_t}{\mathrm{d}t} = (f_{\mathrm{P}} + f_{\mathrm{A}})(\tilde{u}_t) \\ &\forall u \in \mathcal{D}, \ \mathcal{L}(u, \tilde{u}) = 0 \end{aligned} \tag{A.4}$$

In the continuous setting where the data is available at all times t, this problem is in fact equivalent to the following one:

$$\min_{f_{\mathrm{P}} \in \mathcal{F}_{\mathrm{P}}} \mathbb{E}_{u \sim \mathcal{D}} \int_0^T \left\| \frac{\mathrm{d}u_\tau}{\mathrm{d}t} - f_{\mathrm{P}}(u_\tau) \right\| \mathrm{d}\tau \tag{A.5}$$

where the supervision is done directly over derivatives, obtained through finite-difference

schemes. This echoes the proof in Appendix A.2 where f can be reconstructed from the continuous data.

However, in practice, data is only available at discrete times with a certain time resolution. While Eq. (A.5) is indeed equivalent to Eq. (A.4) in the continuous setting, in the practical discrete one, the way error propagates is not anymore: For Eq. (A.4) it is controlled over integrated trajectories while for Eq. (A.5) the supervision is over the approximate derivatives of the trajectories from the dataset. We argue that the trajectory-based approach is more flexible and more robust for the following reasons:

- In Eq. (A.4), if f_{A} is appropriately parameterized, it is possible to perfectly fit the data trajectories at the sampled points.
- The use of finite differences schemes to estimate f as is done in Eq. (A.5) necessarily induces a non-zero discretization error.
- This discretization error is explosive in terms of divergence from the true trajectories.

This last point is quite important, especially when time sampling is sparse (even though we do observe this adverse effect empirically in our experiments with relatively finely time-sampled trajectories). The following gives a heuristical reasoning as to why this is the case. Let $\tilde{f} = f + \epsilon$ be the function estimated from the sampled points with an error ϵ such that $\|\epsilon\|_\infty \leq c$. Denoting $\tilde{u}$ the corresponding trajectory generated by $\tilde{f}$, we then have, for all $u \in \mathcal{D}$:

$$\forall t \in \mathcal{T},\ \frac{\mathrm{d}(u-\tilde{u})_t}{\mathrm{d}t} = f(u_t) - \hat{f}(\tilde{u}_t) = f(u_t) - f(\tilde{u}_t) - \epsilon(\tilde{u}_t)$$

Integrating over $\mathcal{T} = [0, T]$ and using the triangular inequality as well as the mean value inequality, supposing that f has uniformly bounded spatial derivatives:

$$\forall t \in [0,T],\ \|(u-\tilde{u})_t\| \leq \|\nabla f\|_\infty \int_0^t \|u_\tau - \tilde{u}_\tau\| \mathrm{d}\tau + ct$$

which, using a variant of the Grönwall lemma, gives us the inequality:

$$\forall t \in [0,T],\ \|u_t - \tilde{u}_t\| \leq \frac{c}{\|\nabla f\|_\infty}(\exp(\|\nabla f\|_\infty t) - 1)$$

When $c \to 0$, we recover the true trajectories u. However, as c is bounded away from 0 by the available temporal resolution, this inequality gives a rough estimate of the way $\tilde{u}$ diverges from them, and it can be an equality in many cases. This exponential behavior explains our choice of a trajectory-based optimization.

A.5 Implementation Details

We describe here the three use cases studied in the paper for validating APHYNITY. All experiments are implemented with PyTorch (Paszke et al., 2019) and the differentiable ODE solvers with the adjoint method implemented in `torchdiffeq`.[2]

A.5.1 Reaction-Diffusion Equations

The system is driven by a FitzHugh-Nagumo type PDE (Klaasen and Troy, 1984)

$$\frac{\partial v}{\partial t} = a\Delta v + v - v^3 - k - w, \frac{\partial w}{\partial t} = b\Delta w + v - w$$

where a and b are respectively the diffusion coefficients of chemical components v and w, Δ is the Laplace operator. The local reaction terms are $(v - v^3 - k - w)$ and $(v - w)$.

The state $u_t = (v_t, w_t)$ is defined over a compact rectangular domain $\Omega = [-1, 1]^2$ with periodic boundary conditions. Ω is spatially discretized with a 32×32 2D uniform square mesh grid. The periodic boundary condition is implemented with circular padding around the borders. Δ is systematically estimated with a 3×3 discrete Laplace operator.

Dataset Starting from a randomly sampled initial state $u_0 \in [0, 1]^{2\times32\times32}$, we generate states by integrating the true PDE with fixed a, b, and k in a dataset ($a = 1 \times 10^{-3}, b = 5 \times 10^{-3}, k = 5 \times 10^{-3}$). We firstly simulate high time-resolution ($\delta t_{\text{sim}} = 0.001$) sequences with explicit finite difference method. We then extract states every $\delta t_{\text{data}} = 0.1$ to construct our low time-resolution datasets.

We set the time of random initial state to $t = -0.5$ and the time horizon to $t = 2.5$. 1920 sequences are generated, with 1600 for training/validation and 320 for test. We take the state at $t = 0$ as u_0 and predict the sequence until the horizon (equivalent to 25 time steps) in all reaction-diffusion experiments. Note that the sub-sequence with $t < 0$ are reserved for the extensive experiments in Appendix A.7.1.

Neural network architectures. Our f_{A} here is a 3-layer convolution network (ConvNet). The two input channels are (v_t, w_t) and two output ones are $(\frac{\partial v_t}{\partial t}, \frac{\partial w_t}{\partial t})$. The purely data-driven Neural ODE uses such ConvNet as its f. The detailed architecture is provided in Table Table A.1. The estimated physical parameters θ_{P} in f_{P} are simply a trainable vector $(a, b) \in \mathbb{R}^2_+$ or $(a, b, k) \in \mathbb{R}^3_+$.

Optimization hyperparameters. We choose to apply the same hyperparameters for all the reaction-diffusion experiments: $N_{\text{iter}} = 1, \lambda_0 = 1, \eta_1 = 1 \times 10^{-3}, \eta_2 = 1 \times 10^3$.

[2] https://github.com/rtqichen/torchdiffeq

Table A.1: ConvNet architecture in reaction-diffusion and wave equation experiments, used as data-driven derivative operator in APHYNITY and Neural ODE (Chen et al., 2018).

Module	Specification
2D Conv.	3×3 kernel, 2 input channels, 16 output channels, 1 pixel zero padding
2D Batch Norm.	No average tracking
ReLU activation	—
2D Conv.	3×3 kernel, 16 input channels, 16 output channels, 1 pixel zero padding
2D Batch Norm.	No average tracking
ReLU activation	—
2D Conv.	3×3 kernel, 16 input channels, 2 output channels, 1 pixel zero padding

A.5.2 Wave Equations

The damped wave equation is defined by

$$\frac{\partial^2 w}{\partial t^2} - c^2 \Delta w + \gamma \frac{\partial w}{\partial t} = 0$$

where c is the wave speed and γ is the damping coefficient. The state is $u_t = (w_t, \frac{\partial w_t}{\partial t})$.

We consider a compact spatial domain Ω represented as a 64×64 grid and discretize the Laplacian operator similarly. Δ is implemented using a 5×5 discrete Laplace operator in simulation whereas in the experiment is a 3×3 Laplace operator. Null Neumann boundary conditions are imposed for generation.

Dataset. δt was set to 0.001 to respect Courant number and provide stable integration. The simulation was integrated using a 4th order finite difference Runge-Kutta scheme for 300 steps from a Gaussian field initial state, i.e. for every sequence at $t = 0$, we have:

$$w(x, y, t = 0) = C \exp\left(\frac{(x - x_0)^2 + (y - y_0)^2}{\sigma^2}\right) \tag{A.6}$$

The amplitude C is fixed to 1, and $(x_0, y_0) = (32, 32)$ to make the Gaussian curve centered for all sequences. However, σ is different for each sequence and uniformly sampled in $[10, 100]$. The same δt was used for train and test. All initial conditions are Gaussian with varying amplitudes. 250 sequences are generated, 200 are used for training and 50 are reserved as a test set. In the main paper setting, $c = 330$ and $k = 50$. As with the reaction-diffusion case, the algorithm takes as input a state $u_0 = (w_0, \frac{\mathrm{d}w}{\mathrm{d}t}(0))$ and predicts all states from $t_0 + \delta t$ up to $t_0 + 25\delta t$.

Neural network architectures. The neural network for f_{A} is a 3-layer convolution neural network with the same architecture as in Table A.1. For f_{P}, the parameter(s) to be estimated is either a scalar $c \in \mathbb{R}_+$ or a vector $(c, k) \in \mathbb{R}_+^2$. Similarly, Neural ODE networks are built as presented in Table Table A.1.

Optimization hyperparameters. We use the same hyperparameters for the experiments: $N_{\text{iter}} = 3, \lambda_0 = 1, \eta_1 = 1 \times 10^{-4}, \eta_2 = 1 \times 10^2$.

A.5.3 Damped Pendulum

We consider the non-linear damped pendulum problem, governed by the ODE

$$\frac{\mathrm{d}^2\alpha}{\mathrm{d}t^2} + \omega_0^2 \sin\alpha + \gamma\frac{\mathrm{d}\alpha}{\mathrm{d}t} = 0$$

where $\theta(t)$ is the angle, $\omega_0 = \frac{2\pi}{T_0}$ is the proper pulsation (T_0 being the period) and γ is the damping coefficient. With the state $u = (\alpha_t, \frac{\mathrm{d}\alpha}{\mathrm{d}t}(0))$, the ODE can be written as $\frac{\mathrm{d}u}{\mathrm{d}t} = f(u)$ with $f : u_t \mapsto (\frac{\mathrm{d}\alpha}{\mathrm{d}t}, -\omega_0^2 \sin\alpha - \gamma\frac{\mathrm{d}\alpha}{\mathrm{d}t})$.

Dataset. For each train/validation/test split, we simulate a dataset with 25 trajectories of 40 timesteps (time interval $[0, 20]$, timestep $\delta t = 0.5$) with fixed ODE coefficients ($T_0 = 12, \gamma = 0.2$) and varying initial conditions. The simulation integrator is the Dormand-Prince Runge-Kutta method of order (4)5 (DOPRI5; Dormand and Prince, 1980). We also add a small amount of white Gaussian noise ($\sigma = 0.01$) to the state. Note that our pendulum dataset is much more challenging than the ideal frictionless pendulum considered in (Greydanus et al., 2019).

Neural network architectures. We detail in Table Table A.2 the neural architectures used for the damped pendulum experiments. All data-driven augmentations for approximating the mapping $u_t \mapsto f(u_t)$ are implemented by multi-layer perceptron (MLP) with 3 layers of 200 neurons and ReLU activation functions (except at the last layer: linear activation). The Hamiltonian (Greydanus et al., 2019; Toth et al., 2020) is implemented by an MLP that takes the state u_t and outputs a scalar estimation of the Hamiltonian $\mathcal{H}$ of the system: the derivative is then computed by an in-graph gradient of $\mathcal{H}$ w.r.t. the input: $f(u_t) = \left(\frac{\partial\mathcal{H}}{\partial(\mathrm{d}\alpha/\mathrm{d}t)}, -\frac{\partial\mathcal{H}}{\mathrm{d}\alpha}\right)$.

Optimization hyperparameters. The hyperparameters of the APHYNITY optimization algorithm ($Niter, \lambda_0, \eta_1, \eta_2$) were cross-validated on the validation set and are shown in Table A.3. All models were trained with a maximum number of 5000 steps with early stopping.

Table A.2: Neural network architectures for the damped pendulum experiments. — corresponds to non-applicable cases.

Method	Physical model	Data-driven model
Neural ODE	—	MLP(in=2, units=200, layers=3, out=2)
Hamiltonian	MLP(in=2, units=200, layers=3, out=1)	—
APHYNITY Hamiltonian	MLP(in=2, units=200, layers=3, out=1)	MLP(in=2, units=200, layers=3, out=2)
Param ODE (ω_0)	1 trainable parameter ω_0	—
APHYNITY Param ODE (ω_0)	1 trainable parameter ω_0	MLP(in=2, units=200, layers=3, out=2)
Param ODE (ω_0, γ)	2 trainable parameters ω_0, λ	—
APHYNITY Param ODE (ω_0, γ)	2 trainable parameters ω_0, γ	MLP(in=2, units=200, layers=3, out=2)

Table A.3: Hyperparameters of the damped pendulum experiments.

Method	N_{iter}	λ_0	η_1	η_2
APHYNITY Hamiltonian	5	1	1	0.1
APHYNITY Param ODE (ω_0)	5	1	1	10
APHYNITY Param ODE (ω_0, γ)	5	1000	1	100

A.6 Ablation Study

We conduct ablation studies to show the effectiveness of APHYNITY's adaptive optimization and trajectory-based learning scheme.

A.6.1 Ablation to Vanilla MB/ML Cooperation

In Table A.4, we consider the ablation case with the vanilla augmentation scheme found in Le Guen and Thome (2020); Wang et al. (2019); Mehta et al. (2021), which does not present any proper decomposition guarantee. We observe that the APHYNITY cooperation scheme outperforms this vanilla scheme in all cases, both in terms of forecasting performances, e.g., log MSE = -0.35 vs.-3.97 for the Hamiltonian in the pendulum case, and parameter identification, e.g., Err Param=8.4% vs. 2.3% for Param PDE (a, b) for reaction-diffusion. It confirms the crucial benefits of APHYNITY's principled decomposition scheme.

A.6.2 Detailed Ablation Study

We conduct also two other ablations in Table A.5:

- *Derivative supervision*: in which $f_{\text{P}}+f_{\text{A}}$ is trained with supervision over approximated derivatives on ground truth trajectory, as performed in Greydanus et al. (2019) and

Cranmer et al. (2020). More precisely, APHYNITY's $\mathcal{L}_{\text{traj}}$ is here replaced with $\mathcal{L}_{\text{deriv}} = \|\frac{\mathrm{d}u_t}{\mathrm{d}t} - f(u_t)\|$ as in Eq. (A.5), where $\frac{\mathrm{d}u_t}{\mathrm{d}t}$ is approximated by finite differences on u_t.

- *Non-adaptive optim.*: in which we train APHYNITY by minimizing $\|f_{\text{A}}\|$ without the adaptive optimization of λ shown in Algorithm 1. This case is equivalent to $\lambda = 1, \eta_2 = 0$.

We highlight the importance to use a principled adaptive optimization algorithm (APHYNITY algorithm described in the paper) compared to a non-adaptive optimization: for example in the reaction-diffusion case, log MSE= -4.55 vs.-5.10 for Param PDE (a, b). Finally, when the supervision occurs on the derivative, both forecasting and parameter identification results are systematically lower than with APHYNITY's trajectory-based approach: for example, log MSE=-1.16 vs.-4.64 for Param PDE (c) in the wave equation. It confirms the good properties of the APHYNITY training scheme.

A.7 Additional Experiments

A.7.1 Reaction-Diffusion Systems with Varying Diffusion Parameters

We conduct an extensive evaluation on a setting with varying diffusion parameters for reaction-diffusion equations. The only varying parameters are diffusion coefficients, i.e. individual a and b for each sequence. We randomly sample $a \in [1 \times 10^{-3}, 2 \times 10^{-3}]$ and $b \in [3 \times 10^{-3}, 7 \times 10^{-3}]$. k is still fixed to 5×10^{-3} across the dataset.

To estimate a and b for each sequence, we use here a ConvNet encoder E to estimate parameters from 5 reserved frames ($t < 0$). The architecture of the encoder E is similar to the one in Table A.1 except that E takes 5 frames (10 channels) as input and E outputs a vector of estimated $(\tilde{a}, \tilde{b})$ after applying a sigmoid activation scaled by 1×10^{-2} (to avoid possible divergence). For the baseline Neural ODE, we concatenate a and b to each sequence as two channels.

In Table A.6, we observe that combining data-driven and physical components outperforms the pure data-driven one. When applying APHYNITY to Param PDE (a, b), the prediction precision is significantly improved (log MSE: -1.32 vs.-4.32) with a and b respectively reduced from 55.6% and 54.1% to 11.8% and 18.7%. For complete physics cases, the parameter estimations are also improved for Param PDE (a, b, k) by reducing over 60% of the error of b (3.10 vs. 1.23) and 10% to 20% of the errors of a and k (resp. 1.55/0.59 vs. 1.29/0.39).

The extensive results reflect the same conclusion as shown in the main article: APHYNITY

Table A.4: Ablation study comparing APHYNITY to the vanilla augmentation scheme (Wang et al., 2019; Mehta et al., 2021) for the reaction-diffusion equation, wave equation, and damped pendulum.

Dataset	Method	log MSE	%Err Param.	$\|f_A\|^2$
Reaction-diffusion	Param. PDE (a, b) with vanilla aug.	−4.56±0.52	8.4	7.5E1
	APHYNITY Param. PDE (a, b)	**−5.10±0.21**	**2.3**	6.7E1
	Param. PDE (a, b, k) with vanilla aug.	−8.04±0.03	25.4	1.5E−2
	APHYNITY Param. PDE (a, b, k)	**−9.35±0.02**	**0.1**	1.5E−6
	True PDE with vanilla aug.	−8.12±0.05	—	6.1E−4
	APHYNITY True PDE	**−9.17±0.02**	—	1.4E−7
Wave equation	Param PDE (c) with vanilla aug.	−3.90±0.27	0.5	88.66
	APHYNITY Param PDE (c)	**−4.64±0.25**	**0.3**	71.0
	Param PDE (c, k) with vanilla aug.	−5.96±0.10	0.7	25.1
	APHYNITY Param PDE (c, k)	**−6.09±0.28**	**0.7**	4.54
Damped pendulum	Hamiltonian with vanilla aug.	−0.35±0.10	—	837
	APHYNITY Hamiltonian	**−3.97±1.20**	—	623
	Param ODE (ω_0) with vanilla aug.	−7.02±1.70	4.5	148
	APHYNITY Param ODE (ω_0)	**−7.86±0.60**	**4.0**	132
	Param ODE (ω_0, α) with vanilla aug.	−7.60±0.60	4.7	35.5
	APHYNITY Param ODE (ω_0, α)	**−8.31±0.30**	**0.4**	8.5
	Augmented True ODE with vanilla aug.	**−8.40±0.20**	—	3.4
	APHYNITY True ODE	**−8.44±0.20**	—	2.3

Table A.5: Detailed ablation study on supervision and optimization for the reaction-diffusion equation, wave equation, and damped pendulum.

Dataset	Method	log MSE	%Err Param.	$\|f_A\|^2$
Reaction-diffusion	Augmented Param. PDE (a, b) derivative supervision	−4.42±0.25	12.6	68
	Augmented Param. PDE (a, b) non-adaptive optim.	−4.55±0.11	7.5	76
	APHYNITY Param. PDE (a, b)	**−5.10±0.21**	**2.3**	67
	Augmented Param. PDE (a, b, k) derivative supervision	−4.90±0.06	11.7	0.19
	Augmented Param. PDE (a, b, k) non-adaptive optim.	−9.10±0.02	0.21	5.5E−7
	APHYNITY Param. PDE (a, b, k)	**−9.35±0.02**	**0.096**	1.5E−6
	Augmented True PDE derivative supervision	−6.03±0.01	—	3.1E−3
	Augmented True PDE non-adaptive optim.	−9.01±0.01	—	1.5E−6
	APHYNITY True PDE	**−9.17±0.02**	—	1.4E−7
Wave equation	Augmented Param PDE (c) derivative supervision	−1.16±0.48	12.1	2.4E−4
	Augmented Param PDE (c) non-adaptive optim.	−2.57±0.21	3.1	43.6
	APHYNITY Param PDE (c)	**−4.64±0.25**	**0.31**	71.0
	Augmented Param PDE (c, k) derivative supervision	−4.19±0.36	7.2	1.2E−4
	Augmented Param PDE (c, k) non-adaptive optim.	−4.93±0.51	1.32	5.4E−2
	APHYNITY Param PDE (c, k)	**−6.09±0.28**	**0.70**	4.54
	Augmented True PDE derivative supervision	−4.42±0.33	—	6.02E−5
	Augmented True PDE non-adaptive optim.	−4.97±0.49	—	0.23
	APHYNITY True PDE	**−5.24±0.45**	—	0.14
Damped pendulum	Augmented Hamiltonian derivative supervision	−0.83±0.30	—	642
	Augmented Hamiltonian non-adaptive optim.	−0.49±0.58	—	165
	APHYNITY Hamiltonian	**−3.97±1.20**	—	623

(*Continued on next page* →)

Table A.5: *(Continued)*

Dataset	Method	log MSE	%Err Param.	$\|\|f_A\|\|^2$
	Augmented Param ODE (ω_0) derivative supervision	−1.02±0.04	5.8	136
	Augmented Param ODE (ω_0) non-adaptive optim.	−4.30±1.30	4.4	90.4
	APHYNITY Param ODE (ω_0)	**−7.86±0.60**	**4.0**	132
	Augmented Param ODE (ω_0, α) derivative supervision	−2.61±0.20	5.0	3.2
	Augmented Param ODE (ω_0, α) non-adaptive optim.	−7.69±1.30	1.65	4.8
	APHYNITY Param ODE (ω_0, α)	**−8.31±0.30**	**0.39**	8.5
	Augmented True ODE derivative supervision	−2.14±0.30	—	4.1
	Augmented True ODE non-adaptive optim.	**−8.34±0.40**	—	1.4
	APHYNITY True ODE	**−8.44±0.20**	—	2.3

Table A.6: Results of reaction-diffusion with varying (a, b). $k = 5\text{E}{-3}$ is shared across the dataset.

	Method	log MSE	%Err a	%Err b	%Err k	$\|f_\text{A}\|^2$
Data-driven	Neural ODE (Chen et al., 2018)	−3.61±0.07	—	—	—	—
Incomplete physics	Param PDE (a, b)	−1.32±0.02	55.6	54.1	—	—
	APHYNITY Param PDE (a, b)	**−4.32±0.32**	**11.8**	**18.7**	—	4.3E1
Complete physics	Param PDE (a, b, k)	**−5.54±0.38**	1.55	3.10	0.59	—
	APHYNITY Param PDE (a, b, k)	**−5.72±0.25**	**1.29**	**1.23**	**0.39**	5.9E−1
	True PDE	**−8.86±0.02**	—	—	—	—
	APHYNITY True PDE	**−8.82±0.15**	—	—	—	1.8E−5

Table A.7: Results for the damped wave equation when considering multiple c, sampled uniformly in $[300, 400]$ in the dataset. k is shared across all sequences and $k = 50$.

	Method	log MSE	%Error c	%Error k	$\|f_\text{A}\|^2$
Data-driven	Neural ODE	0.056±0.340	—	—	—
Incomplete physics	Param PDE (c)	−1.32±0.27	23.9	—	—
	APHYNITY Param PDE (c)	**−4.51±0.38**	3.2	—	171
Complete physics	Param PDE (c, k)	−4.25±0.28	3.54	1.43	—
	APHYNITY Param PDE (c, k)	**−4.84±0.57**	2.41	0.064	3.64
	True PDE (c, k)	**−4.51±0.29**	—	—	—
	APHYNITY True PDE (c, k)	**−4.49±0.22**	—	—	5E−4

improves prediction precision and parameter estimation. The same decreasing tendency of $\|f_A\|$ is also confirmed.

A.7.2 Additional Results for the Wave Equation

We conduct an experiment where each sequence is generated with a different wave celerity. This dataset is challenging because both c and the initial conditions vary across the sequences. For each simulated sequence, an initial condition is sampled as described previously, along with a wave celerity c also sampled uniformly in $[300, 400]$. Finally, our initial state is integrated with the same Runge-Kutta scheme. 200 of such sequences are generated for training while 50 are kept for testing.

For this experiment, we also use a ConvNet encoder to estimate the wave speed c from 5 consecutive reserved states $(w, \frac{\partial w}{\partial t})$. The architecture of the encoder E is the same as in Table A.1 but with 10 input channels. Here also, k is fixed for all sequences and $k = 50$. The hyper-parameters used in these experiments are the same than described in Appendix A.5.2.

The results when multiple wave speeds c are in the dataset are consistent with the one present when only one is considered. Indeed, while prediction performances are slightly hindered, the parameter estimation remains consistent for both c and k. This extension provides elements attesting to the robustness and adaptability of our method to more complex settings. Finally, the purely data-driven Neural-ODE fails to cope with the increasing difficulty.

A.7.3 Damped Pendulum with Varying Parameters

To extend the experiments conducted in the paper (section Section 4.3) with fixed parameters ($T_0 = 6, \alpha = 0.2$) and varying initial conditions, we evaluate APHYNITY on a much more challenging dataset where we vary both the parameters (T_0, α) and the initial conditions between trajectories.

We simulate 500/50/50 trajectories for the train/valid/test sets integrated with DOPRI5. For each trajectory, the period T_0 (resp. the damping coefficient α) are sampled uniformly in the range $[3, 10]$ (resp. $[0, 0.5]$).

We train models that take the first 20 steps as input and predict the next 20 steps. To account for the varying ODE parameters between sequences, we use an encoder that estimates the parameters based on the first 20 timesteps. In practice, we use a recurrent encoder composed of 1 layer of 128 GRU units. The output of the encoder is fed as additional input to the data-driven augmentation models and an MLP with final softplus activations to estimate the physical parameters when necessary ($\omega_0 \in \mathbb{R}_+$ for Param ODE (ω_0), $(\omega_0, \alpha) \in \mathbb{R}_+^2$ for Param ODE (ω_0, α)).

In this varying ODE context, we also compare to the state-of-the-art univariate time series forecasting method N-Beats (Oreshkin et al., 2020).

Results shown in Table Table A.8 are consistent with those presented in the paper. Pure data-driven models Neural ODE (Chen et al., 2018) and N-Beats (Oreshkin et al., 2020) fail to properly extrapolate the pendulum dynamics. Incomplete physical models (Hamiltonian and ParamODE (ω_0)) are even worse since they do not account for friction. Augmenting them with APHYNITY significantly and consistently improves forecasting results and parameter identification.

Table A.8: Forecasting and identification results on the damped pendulum dataset with different parameters for each sequence. log MSEs are computed over 20 predicted time-steps. For each level of incorporated physical knowledge, equivalent best results according to a Student t-test are shown in bold. — corresponds to non-applicable cases.

	Method	log MSE	%Error T_0	%Error γ	$\|\|f_A\|\|^2$
Data-driven	Neural ODE (Chen et al., 2018)	−4.35±0.90	—	—	—
	N-Beats (Oreshkin et al., 2020)	−4.57±0.50	—	—	—
Incomplete physics	Hamiltonian (Greydanus et al., 2019)	−1.31±0.40	—	—	—
	APHYNITY Hamiltonian	**−4.70±0.40**	—	—	5.6
	Param ODE (ω_0)	−2.66±0.90	22 ± 19	—	—
	APHYNITY Param ODE (ω_0)	**−5.94±0.70**	**5.0±1.8**	—	0.49
Complete physics	Param ODE (ω_0, α)	**−5.71±0.40**	4.1±0.8	152±129	—
	APHYNITY Param ODE (ω_0, α)	**−6.22±0.70**	**3.3±0.6**	**62±27**	5.39E−10
	True ODE	**−8.58±0.10**	—	—	—
	APHYNITY True ODE	**−8.58±0.10**	—	—	2.15E−4

Appendix B

Appendix of Chapter 5

B.1 Proof of Proposition 5.1

Proposition 5.1 (Existence and Uniqueness). *Assume $\mathcal{R}$ is convex, then the existence of a minimal decomposition $f^\star, \{g_e^\star\}_{e\in\mathcal{E}} \in \mathcal{F}$ of Eq. (5.3) is guaranteed. Furthermore, if $\mathcal{R}$ is strictly convex, this decomposition is unique.*

Proof. The optimization problem is:

$$\min_{f, g_e \in \mathcal{F}} \sum_{e \in \mathcal{E}} \mathcal{R}(g_e) \quad \text{subject to} \quad \forall u^{e,(i)} \in \mathcal{D}, \forall t \in \mathcal{I}, \frac{\mathrm{d}u_t^{e,(i)}}{\mathrm{d}t} = (f + g_e)(u_t^{e,(i)}) \tag{5.1}$$

The idea is to first reconstruct the full functional from the trajectories of $\mathcal{D}$. By definition, $\hat{\mathcal{U}}^e \subset \mathcal{U}$ is the set of points reached by trajectories in $\mathcal{D}$ from environment e so that:

$$\hat{\mathcal{U}}^e = \{z \in \mathbb{R}^d \mid \exists u^e\big|_{\mathcal{T}} \in \mathcal{D}, \exists t \in \mathcal{T},\ z = u_t^e\}$$

Then let us define a function f_e^{data} in the following way, $\forall e \in \mathcal{E}$, take $z \in \hat{\mathcal{U}}^e$, we can find $u^e\big|_{\mathcal{T}} \in \mathcal{D}$ and τ such that $u_\tau^e = z$. Differentiating u^e at t_0, which is possible by definition of $\hat{\mathcal{T}}$, we take:

$$f_e^{\text{data}}(z) = \left.\frac{\mathrm{d}u^e}{\mathrm{d}t}\right|_{t=t_0}$$

For any (f, g_e) satisfying the constraint in Eq. (5.1), we then have $(f + g_e)(z) = \left.\frac{\mathrm{d}u_t}{\mathrm{d}t}\right|_{t_0} = f_e^{\text{data}}(z)$ for all $z \in \hat{\mathcal{U}}^e$. Conversely, any pair such that $(f, g_e) \in \mathcal{F} \times \mathcal{F}$ and $f + g_e = f_e^{\text{data}}$, verifies the constraint.

Thus we have the equivalence between Eq. (5.1) and the following objective:

$$\min_{f \in \mathcal{F}} \sum_e \mathcal{R}(f_e^{\text{data}} - f) \tag{B.1}$$

The result directly follows from the fact that the objective is a sum of (strictly) convex functions in f and is thus (strictly) convex in f. □

B.2 Further Details on the Generalization with LEADS

In this section, we will give more details on the link between our framework and its generalization performance. After introducing the necessary definitions in Appendix B.2.1, we show the proofs of the results for the general case in Section 5.3. Then in Appendix B.2.3 we provide the instantiation for linear approximators. Finally, we show how we derived our heuristic instantiation for neural networks in Eq. (5.10) in Section 5.3.3 from the existing capacity bound for neural networks.

Table B.1: Capacity definitions of different sets by covering number with associated metric or pseudo-metric.

Capacity	Metric or pseudo-metric	Mentioned in
$\mathcal{C}(\varepsilon, \mathbb{H}^m) := \sup_{\boldsymbol{\mathcal{P}}} \mathcal{N}(\varepsilon, \mathbb{H}^m, d_{\boldsymbol{\mathcal{P}}})$	$d_{\boldsymbol{\mathcal{P}}}((f+g_1,\dots,f+g_m),(f'+g'_1,\dots,f'+g'_m)) = \int_{(\mathcal{U}\times\mathrm{T}\mathcal{U})^m} \frac{1}{m} \lvert \sum_{e\in\mathcal{E}} \lVert (f+g_e)(z^e) - y^e\rVert^2 - \sum_{e\in\mathcal{E}} \lVert (f'+g'_e)(z^e) - y^e\rVert^2 \rvert \mathrm{d}\boldsymbol{\mathcal{P}}(\mathbf{z},\mathbf{y})$	Theorem B.1 and proposition B.1
$\mathcal{C}_{\hat{\mathcal{G}}}(\varepsilon, \hat{\mathcal{F}}) := \sup_{\mathcal{P}} \mathcal{N}(\varepsilon, \hat{\mathcal{F}}, d_{[\mathcal{P},\hat{\mathcal{G}}]})$	$d_{[\mathcal{P},\hat{\mathcal{G}}]}(f,f') = \int_{\mathcal{U}\times\mathrm{T}\mathcal{U}} \sup_{g\in\hat{\mathcal{G}}} \lvert \lVert (f+g)(x) - y\rVert^2 - \lVert (f'+g)(x) - y\rVert^2 \rvert \mathrm{d}\mathcal{P}(x,y)$	Propositions 5.2, B.1 and B.4 and corollary B.5
$\mathcal{C}_{\hat{\mathcal{F}}}(\varepsilon, \hat{\mathcal{G}}) := \sup_{\mathcal{P}} \mathcal{N}(\varepsilon, \hat{\mathcal{G}}, d_{[\mathcal{P},\hat{\mathcal{F}}]})$	$d_{[\mathcal{P},\hat{\mathcal{F}}]}(g,g') = \int_{\mathcal{U}\times\mathrm{T}\mathcal{U}} \sup_{f\in\hat{\mathcal{F}}} \lvert \lVert (f+g)(z) - y\rVert^2 - \lVert (f+g')(z) - y\rVert^2 \rvert \mathrm{d}\mathcal{P}(z,y)$	Propositions 5.2, B.1 and B.3
$\mathcal{C}(\varepsilon, f+\hat{\mathcal{G}}) := \sup_{\mathcal{P}} \mathcal{N}(\varepsilon, f+\hat{\mathcal{G}}, d_{\mathcal{P}})$	$d_{\mathcal{P}}(f+g, f+g') = \int_{\mathcal{U}\times\mathrm{T}\mathcal{U}} \lvert \lVert (f+g)(z) - y\rVert^2 - \lVert (f+g')(z) - y\rVert^2 \rvert \mathrm{d}\mathcal{P}(z,y)$	Proposition 5.3
$\mathcal{C}(\varepsilon, \hat{\mathcal{G}}, L^1) := \sup_{\mathcal{P}} \mathcal{N}(\varepsilon, \hat{\mathcal{G}}, d_{L^1(\mathcal{P})})$	$d_{L^1(\mathcal{P})}(g,g') = \int_{\mathbb{R}^d} \lVert (g-g')(z)\rVert_1 \mathrm{d}\mathcal{P}(z)$	Proposition B.3 and theorem B.3
$\mathcal{C}(\varepsilon, \hat{\mathcal{G}}, L^2) := \sup_{\mathcal{P}} \mathcal{N}(\varepsilon, \hat{\mathcal{G}}, d_{L^2(\mathcal{P})})$	$d_{L^2(\mathcal{P})}(g,g') = \sqrt{\int_{\mathbb{R}^d} \lVert (g-g')(z)\rVert_2^2 \mathrm{d}\mathcal{P}(z)}$	Proposition 5.4 and lemma B.2

B.2.1 Preliminaries

Table B.1 gives the definition of the different capacity instances considered in the paper for each hypothesis space, and the associated distances. We say that a space $\mathcal{H}$ is ε-covered by a set H, with respect to a metric or pseudo-metric $d(\cdot,\cdot)$, if for all $h \in \mathcal{H}$ there exists $h' \in H$ with $d(h,h') \leq \varepsilon$. We define by $\mathcal{N}(\varepsilon,\mathcal{H},d)$ the cardinality of the smallest H that ε-covers $\mathcal{H}$, also called covering number Shalev-Shwartz and Ben-David (2014). The capacity of each hypothesis space is then defined by the maximum covering number over all distributions. Note that the loss function is involved in every metric in Table B.1. For simplicity, we therefore omit the notation of loss function for the hypothesis spaces.

As in Baxter (2000), covering numbers are based on pseudo-metrics. We can verify that all distances in Table B.1 are pseudo-metrics:

Proof. This is trivially verified. For example, for the distance $d_{\mathcal{P}}(f+g, f+g')$ given in Table B.1, which is the distance between $f+g, f+g' \in f+\hat{\mathcal{G}}$, it is easy to check that the following properties do hold:

- $d_{\mathcal{P}}(f+g, f+g') = 0$ (subtraction of same functions evaluated on same x and y)
- $d_{\mathcal{P}}(f+g, f+g') = d_{\mathcal{P}}(f+g', f+g)$ (evenness of absolute value)
- $d_{\mathcal{P}}(f+g, f+g') \leq d_{\mathcal{P}}(f+g, f+g'') + d_{\mathcal{P}}(f+g'', f+g')$ (triangular inequality of absolute value)

Other distances in Table B.1 can be proven to be pseudo-metrics in the same way. □

B.2.2 General Case

Proof of Proposition 5.2

Proposition 5.2. *Given m environments, let $\varepsilon_1, \varepsilon_2, \delta > 0, \varepsilon = \varepsilon_1 + \varepsilon_2$. Assume the number of examples n per environment satisfies*

$$n \geq \max\left\{\frac{64}{\varepsilon^2}\left(\frac{1}{m}\left(\log\frac{4}{\delta} + \log \mathcal{C}_{\hat{\mathcal{G}}}\left(\frac{\varepsilon_1}{16}, \hat{\mathcal{F}}\right)\right) + \log \mathcal{C}_{\hat{\mathcal{F}}}\left(\frac{\varepsilon_2}{16}, \hat{\mathcal{G}}\right)\right), \frac{16}{\varepsilon^2}\right\} \tag{5.5}$$

Then with probability at least $1-\delta$ (over the choice of training sets $\{\hat{\mathcal{P}}_e\}$), any learner $(f+g_1, \ldots, f+g_m)$ will satisfy $\frac{1}{m}\sum_{e\in\mathcal{E}} \mathrm{er}_{\mathcal{P}_e}(f+g_e) \leq \frac{1}{m}\sum_{e\in\mathcal{E}} \hat{\mathrm{er}}_{\hat{\mathcal{P}}_e}(f+g_e) + \varepsilon$.

Proof. We introduce some extra definitions that are necessary for proving the proposition. Let $\mathcal{H} = f + \hat{\mathcal{G}}$ defined for each $f \in \hat{\mathcal{F}}$, and let us define the product space $\mathcal{H}^m = \{(f+g_1, \ldots, f+g_m) : f+g_e \in \mathcal{H}\}$. Functions in this hypothesis space all have the same f, but not necessarily the same g_e. Let $\mathbb{H}$ be the collection of all hypothesis spaces $\mathcal{H} = f +$

$\hat{\mathcal{G}}, \forall f \in \hat{\mathcal{F}}$. The hypothesis space associated to multiple environments is then defined as $\mathbb{H}^m := \bigcup_{\mathcal{H}\in\mathbb{H}} \mathcal{H}^m$.

Our proof makes use of two intermediary results addressed in Theorem B.1 and Proposition B.1.

Theorem B.1 (Baxter, 2000, Theorem 4, adapted to our setting)**.** *Assuming $\mathbb{H}$ is a permissible hypothesis space family. For all $\varepsilon > 0$, if the number of examples n of each environment satisfies:*

$$n \geq \max\left\{\frac{64}{m\varepsilon^2}\log\frac{4\mathcal{C}(\frac{\varepsilon}{16},\mathbb{H}^m)}{\delta}, \frac{16}{\varepsilon^2}\right\}$$

Then with probability at least $1-\delta$ (over the choice of $\{\hat{\mathcal{P}}_e\}$), any $(f+g_1,\ldots,f+g_m)$ will satisfy

$$\frac{1}{m}\sum_{e\in\mathcal{E}} \mathrm{er}_{\mathcal{P}_e}(f+g_e) \leq \frac{1}{m}\sum_{e\in\mathcal{E}} \hat{\mathrm{er}}_{\hat{\mathcal{P}}_e}(f+g_e) + \varepsilon$$

Note that permissibility (as defined in Baxter (2000)) is a weak measure-theoretic condition satisfied by many real world hypothesis space families Baxter (2000). We will now express the capacity of $\mathbb{H}^m$ in terms of the capacities of its two constituent component-spaces $\hat{\mathcal{F}}$ and $\hat{\mathcal{G}}$, thus leading to the main result.

Proposition B.1. *For all $\varepsilon, \varepsilon_1, \varepsilon_2 > 0$ such that $\varepsilon = \varepsilon_1 + \varepsilon_2$,*

$$\log \mathcal{C}(\varepsilon, \mathbb{H}^m) \leq \log \mathcal{C}_{\hat{\mathcal{G}}}(\varepsilon_1, \hat{\mathcal{F}}) + m \log \mathcal{C}_{\hat{\mathcal{F}}}(\varepsilon_2, \hat{\mathcal{G}}) \tag{B.2}$$

Proof of Proposition B.1. To prove the proposition it is sufficient to show the property of covering sets for any joint distribution defined on all environments $\boldsymbol{\mathcal{P}}$ on the space $(\mathcal{U}\times\mathrm{T}\mathcal{U})^m$. Let us then fix such a distribution $\boldsymbol{\mathcal{P}}$. and let $\bar{\mathcal{P}} = \frac{1}{m}\sum_{e\in\mathcal{E}}\mathcal{P}_e$ be the average distribution.

Suppose that F is an ε_1-cover of $(\hat{\mathcal{F}}, d_{[\bar{\mathcal{P}},\hat{\mathcal{G}}]})$ and $\{G_e\}_{e\in\mathcal{E}}$ are ε_2-covers of $(\hat{\mathcal{G}}, d_{[\mathcal{P}_e,\hat{\mathcal{F}}]})$. Let $H = \{(x_1,\ldots,x_m) \mapsto ((f+g_1)(x_1),\ldots,(f+g_m)(x_m)),\ f\in F,\ g_e \in G_e\}$, be a set built from the covering sets aforementioned. Note that by definition $|H| = |F|\cdot\prod_{e\in\mathcal{E}}|G_e| \leq \mathcal{C}_{\hat{\mathcal{G}}}(\varepsilon_1,\hat{\mathcal{F}})\cdot\mathcal{C}_{\hat{\mathcal{F}}}(\varepsilon_2,\hat{\mathcal{G}})^m$ as we take some distribution instances.

For each learner $(f+g_1,\ldots,f+g_m)\in\mathbb{H}^m$ in the hypothesis space, we take any $f'\in F$ such that $d_{[\bar{\mathcal{P}},\hat{\mathcal{G}}]}(f,f') \leq \varepsilon_1$ and $g'_e \in G_e$ for all e such that $d_{[\mathcal{P}_e,\hat{\mathcal{F}}]}(g_e,g'_e) \leq \varepsilon_2$, and we build $(f'+g'_1,\ldots,f'+g'_m)$. The distance is then:

$$\begin{aligned} &d_{\boldsymbol{\mathcal{P}}}((f+g_1,\ldots,f+g_m),(f'+g'_1,\ldots,f'+g'_m)) \\ \leq\ &d_{\boldsymbol{\mathcal{P}}}((f+g_1,\ldots,f+g_m),(f'+g_1,\ldots,f'+g_m)) \end{aligned}$$

$$+ d_{\boldsymbol{\mathcal{P}}}((f' + g_1, \dots, f' + g_m), (f' + g'_1, \dots, f' + g'_m))$$

(triangular inequality of pseudo-metric)

$$\leq \frac{1}{m}\left[\sum_{e\in\mathcal{E}} d_{\mathcal{P}_e}(f + g_e, f' + g_e) + \sum_{e\in\mathcal{E}} d_{\mathcal{P}_e}(f' + g_e, f' + g'_e)\right]$$

(triangular inequality of absolute value)

$$\leq \frac{1}{m}\sum_{e\in\mathcal{E}} d_{[\mathcal{P}_e,\hat{\mathcal{G}}]}(f, f') + \frac{1}{m}\sum_{e\in\mathcal{E}} d_{[\mathcal{P}_e,\hat{\mathcal{F}}]}(g_e, g'_e) \qquad \text{(by definition of } d_{[\mathcal{P}_e,\hat{\mathcal{G}}]} \text{ and } d_{[\mathcal{P}_e,\hat{\mathcal{F}}]})$$

$$= d_{[\bar{\mathcal{P}},\hat{\mathcal{G}}]}(f, f') + \frac{1}{m}\sum_{e\in\mathcal{E}} d_{[\mathcal{P}_e,\hat{\mathcal{F}}]}(g_e, g'_e) \leq \varepsilon_1 + \varepsilon_2$$

(mean of the distance on different $\mathcal{P}_e$ is the distance on $\bar{\mathcal{P}}$)

To conclude, for any distribution $\boldsymbol{\mathcal{P}}$, when F is an ε_1-cover of $\hat{\mathcal{F}}$ and $\{G_e\}$ are ε_2-covers of $\hat{\mathcal{G}}$, the set H built upon them is an $(\varepsilon_1 + \varepsilon_2)$-cover of $\mathbb{H}^m$. Then if we take the maximum over all distributions we conclude that $\mathcal{C}(\varepsilon_1 + \varepsilon_2, \mathbb{H}^m) \leq \mathcal{C}_{\hat{\mathcal{G}}}(\varepsilon_1, \hat{\mathcal{F}})\mathcal{C}_{\hat{\mathcal{F}}}(\varepsilon_2, \hat{\mathcal{G}})^m$ and we have Eq. (B.2). ■

We can now use the bound developed in Proposition B.1 and use it together with Theorem B.1, therefore concluding the proof of Proposition 5.2. □

Proof of Proposition 5.3

Proposition 5.3. *For all ε, δ with $0 < \varepsilon, \delta < 1$ if the number of samples n' satisfies*

$$n' \geq \max\left\{\frac{64}{\varepsilon^2}\log\frac{4\mathcal{C}(\frac{\varepsilon}{16}, f + \hat{\mathcal{G}})}{\delta}, \frac{16}{\varepsilon^2}\right\}, \tag{5.6}$$

then with probability at least $1 - \delta$ (over the choice of novel training set $\hat{\mathcal{P}}_{e'}$), any learner $f + g_{e'} \in f + \hat{\mathcal{G}}$ will satisfy $\mathrm{er}_{\mathcal{P}_{e'}}(f + g_{e'}) \leq \hat{\mathrm{er}}_{\hat{\mathcal{P}}_{e'}}(f + g_{e'}) + \varepsilon$.

Proof. The proof is derived from the following theorem which can be easily adapted to our context:

Theorem B.2 (Baxter, 2000, Theorem 3)**.** *Let $\mathcal{H}$ a permissible hypothesis space. For all $0 < \varepsilon, \delta < 1$, if the number of examples n of each environment satisfies:*

$$n \geq \max\left\{\frac{64}{m\varepsilon^2}\log\frac{4\mathcal{C}(\frac{\varepsilon}{16}, \mathcal{H})}{\delta}, \frac{16}{\varepsilon^2}\right\}$$

Then with probability at least $1 - \delta$ (over the choice of dataset $\hat{\mathcal{P}}$ sampled from $\mathcal{P}$), any

$h \in \mathcal{H}$ will satisfy

$$\mathrm{er}_{\mathcal{P}}(h) \leq \hat{\mathrm{er}}_{\hat{\mathcal{P}}}(h) + \varepsilon$$

Given that $\hat{\mathcal{P}}_{e'}$ is sampled from the same environment distribution Q, then by fixing the pre-trained f, we fix the space of hypothesis to $f + \hat{\mathcal{G}}$, and we apply Theorem B.2 to obtain the proposition. □

B.2.3 Linear Case

We provide here the proofs of theoretical bounds given in Section 5.3.2. See the description in Appendix B.4 for detailed information on the example linear ODE dataset and the training with a varying number of environments.

Proof of Proposition 4

Proposition 5.4. *If for all linear maps $\mathrm{L}_{\boldsymbol{G}_e} \in \hat{\mathcal{G}}$, $\|\boldsymbol{G}\|_F^2 \leq r$, if the input space is bounded s.t. $\|u\|_2 \leq b$, and the MSE loss function is bounded by c, then*

$$\log \mathcal{C}_{\hat{\mathcal{F}}}(\varepsilon, \hat{\mathcal{G}}) \leq \left\lceil \frac{rcd(2b)^2}{\varepsilon^2} \right\rceil \log 2d^2 =: \omega(r, \varepsilon) \tag{5.8}$$

Proof. Let us take G an $\frac{\varepsilon}{2\sqrt{c}}$-cover of $\hat{\mathcal{G}}$ with L^2-distance: $d_{L^2(\mathcal{P})}$ (see definition in Table B.1). Therefore, for each $\mathrm{L}_{\boldsymbol{G}} \in \hat{\mathcal{G}}$ take $g' \in G$ such that $d_{L^2}(\mathrm{L}_{\boldsymbol{G}}, \mathrm{L}_{\boldsymbol{G}'}) \leq \frac{\varepsilon}{2\sqrt{c}}$, then

$$\begin{aligned}
&d_{[\mathcal{P},\hat{\mathcal{F}}]}(\mathrm{L}_{\boldsymbol{G}}, \mathrm{L}_{\boldsymbol{G}'}) \\
=&\int_{\mathcal{U}\times \mathrm{T}\mathcal{U}} \sup_{\mathrm{L}_{\boldsymbol{F}}\in\hat{\mathcal{F}}} |\|(\boldsymbol{F}+\boldsymbol{G})z - y\|_2^2 - \|(\boldsymbol{F}+\boldsymbol{G}')z - y\|_2^2| \mathrm{d}\mathcal{P}(z, y) \\
\leq&\int_{\mathcal{U}\times \mathrm{T}\mathcal{U}} \sup_{\mathrm{L}_{\boldsymbol{F}}\in\hat{\mathcal{F}}} \|(\boldsymbol{G}-\boldsymbol{G}')z\|_2 (\|(\boldsymbol{F}+\boldsymbol{G})z - y\|_2 + \|(\boldsymbol{F}+\boldsymbol{G}')(z) - y\|_2) \mathrm{d}\mathcal{P}(z, y) \\
\leq&\sqrt{\int_{\mathcal{U}} \|(\boldsymbol{G}-\boldsymbol{G}')z\|_2 \mathrm{d}\mathcal{P}(z)} \sqrt{\int_{\mathcal{U}\times \mathrm{T}\mathcal{U}} \sup_{\mathrm{L}_{\boldsymbol{F}}\in\hat{\mathcal{F}}} (\|(\boldsymbol{F}+\boldsymbol{G})z - y\|_2 + \|(\boldsymbol{F}+\boldsymbol{G}')z - y\|_2)^2 \mathrm{d}\mathcal{P}(z, y)} \\
\leq&2\sqrt{c}\sqrt{\int_{\mathbb{R}^d} \|(\boldsymbol{G}-\boldsymbol{G}')x\|_2 \mathrm{d}\mathcal{P}(x)} \leq \varepsilon
\end{aligned}$$

We have the $\mathcal{C}_{\mathcal{F}}(\varepsilon, \hat{\mathcal{G}}) \leq \mathcal{C}(\frac{\varepsilon}{2\sqrt{c}}, \hat{\mathcal{G}}, L^2)$. According to the following lemma:

Lemma B.2 (Bartlett et al., 2017, Lemma 3.2, Adapted). *Given positive reals (a, b, ε) and positive integer d. Let vector $z \in \mathbb{R}^d$ be given with $\|z\|_p \leq b$, $\hat{\mathcal{G}} = \{\mathrm{L}_{\boldsymbol{G}} : \boldsymbol{G} \in \mathbb{R}^{d\times d}, \|\boldsymbol{G}\|_F^2 \leq r\}$ where $\|\cdot\|_F$ is the Frobenius norm. Then*

$$\log \mathcal{C}(\varepsilon, \hat{\mathcal{G}}, L^2) \leq \left\lceil \frac{rdb^2}{\varepsilon^2} \right\rceil \log 2d^2$$

And we obtain that

$$\log \mathcal{C}_{\hat{\mathcal{F}}}(\varepsilon, \hat{\mathcal{G}}) \leq \left\lceil \frac{rcd(2b)^2}{\varepsilon^2} \right\rceil \log 2d^2 =: \omega(r, \varepsilon)$$

where $\omega(r, \varepsilon)$ is a strictly increasing function w.r.t. r. □

Proof of Proposition 5

Proposition 5.5. *If for linear maps $\mathrm{L}_{\boldsymbol{F}} \in \hat{\mathcal{F}}$, $\|\boldsymbol{F}\|_F^2 \leq r'$, $\mathrm{L}_{\boldsymbol{G}} \in \hat{\mathcal{G}}$, $\|\boldsymbol{G}\|_F^2 \leq r$, $\|u\|_2 \leq b$, and if the MSE loss function is bounded by c, given m environments and n samples per environment, with the probability $1-\delta$, the generalization error upper bound is*

$$\varepsilon = \max\left\{ \sqrt{\frac{(p + \sqrt{p^2 + 4q})}{2}}, \sqrt{\frac{16}{n}} \right\} \tag{5.9}$$

where $p = \frac{64}{mn} \log \frac{4}{\delta}$ and $q = \frac{64}{n}\left\lceil \left(\frac{r'}{ml^2} + \frac{r}{(1-l)^2}\right) cd(32b)^2 \right\rceil \log 2d^2$ for any $0 < l < 1$.

Proof. This can be derived from Proposition 5.2 with the help of Proposition 5.4 for linear maps. If we take the lower bounds of two capacities $\log \mathcal{C}_{\hat{\mathcal{F}}}(\frac{\varepsilon_1}{16}, \hat{\mathcal{G}})$ and $\log \mathcal{C}_{\hat{\mathcal{G}}}(\frac{\varepsilon_2}{16}, \hat{\mathcal{F}})$ for the linear maps hypothesis spaces $\hat{\mathcal{F}}, \hat{\mathcal{G}}$, then the number of required samples per environment N now can be expressed as follows:

$$N = \max\left\{ \frac{64}{\varepsilon^2}\left(\frac{1}{m} \log \frac{4}{\delta} + \frac{1}{m} \left\lceil \frac{r'cd(32b)^2}{\varepsilon_1^2} \right\rceil \log 2d^2 + \left\lceil \frac{rcd(32b)^2}{\varepsilon_2^2} \right\rceil \log 2d^2 \right), \frac{16}{\varepsilon^2} \right\}$$

To simplify the resolution of the equation above, we take $\varepsilon_1 = z\varepsilon$ for any $0 < z < 1$, then $\varepsilon_2 = \varepsilon - \varepsilon_1 = (1-z)\varepsilon$. Then by resolving the equation, the generalization margin is then upper bounded by ε with:

$$\varepsilon = \max\left\{ \sqrt{\frac{p + \sqrt{p^2 + 4q}}{2}}, \sqrt{\frac{16}{n}} \right\}$$

where $p = \frac{64}{mn} \log \frac{4}{\delta}$ and $q = \frac{64}{n} \left\lceil \left(\frac{r}{mz^2} + \frac{r'}{(1-z)^2} \right) cd(32b)^2 \right\rceil \log 2d^2$. □

B.2.4 Nonlinear Case: Instantiation for Neural Networks

We show in this section how we design a concrete model for nonlinear dynamics following the general guidelines given in Section 5.3.1. This is mainly composed of the following two parts: (a) choosing an appropriate approximation space and (b) choosing a penalization function $\mathcal{R}$ for this space. It is important to note that, even if the bounds given in the following sections may be loose in general, it could provide useful intuitions on the design of the algorithms which can be validated by experiments in our case.

Choosing approximation space $\hat{\mathcal{F}}$

We choose the space of feed-forward neural networks with a fixed architecture. Given the universal approximation properties of neural networks Kidger and Lyons (2020), and the existence of efficient optimization algorithms Chizat and Bach (2018), this is a reasonable choice, but other families of approximating functions could be used as well.

We then consider the function space of neural networks with D-layers with inputs and outputs in $\mathbb{R}^d$: $\hat{\mathcal{F}}_{\text{NN}} = \{\nu : x \mapsto \sigma_D(W_D \cdots \sigma_1(W_1 x))) : x, \nu(x) \in \mathbb{R}^d\}$, D is the depth of the network, σ_j is a Lipschitz activation function at layer j, and W_j weight matrix from layer $j-1$ to j. The number of adjustable parameters is fixed to W for the architecture. This definition covers fully connected neural networks (NNs) and convolutional NNs. Note that the FNO used in the experiments for *Navier-Stokes* can be also covered by the definition above, as it performs alternatively the convolution in the Fourier space.

Choosing penalization $\mathcal{R}$

Now we choose an $\mathcal{R}$ for the space above. Let us first introduce a practical way to bound the capacity of $\hat{\mathcal{G}} \in \hat{\mathcal{F}}_{\text{NN}}$. Proposition B.3 tells us that for a fixed NN architecture (implying constant parameter number W and depth D), we can control the capacity through the maximum output norm R and Lipschitz norm L defined in the proposition.

Proposition B.3. *If for all neural network $g \in \hat{\mathcal{G}}$, $\|g\|_\infty = \operatorname{ess\,sup} |g| \leq R$ and $\|g\|_{\text{Lip}} \leq L$, with $\|\cdot\|_{\text{Lip}}$ the Lipschitz semi-norm, then:*

$$\log \mathcal{C}_{\hat{\mathcal{F}}}(\varepsilon, \hat{\mathcal{G}}) \leq \omega(R, L, \varepsilon) \tag{B.3}$$

where $\omega(R, L, \varepsilon) = c_1 \log \frac{RL}{\varepsilon} + c_2$ for $c_1 = 2W$ and $c_2 = 2W \log 8e\sqrt{c}D$, with c the bound of MSE loss. $\omega(R, L, \varepsilon)$ is a strictly increasing function w.r.t. R and L.

Proof. To link the capacity to some quantity that can be optimized for neural networks, we need to apply the following theorem:

Theorem B.3 (Haussler, 1992, Theorem 11, Adapted)**.** *With the neural network function space $\hat{\mathcal{F}}_{NN}$, let W be the total number of adjustable parameters, D the depth of the architecture. Let $\hat{\mathcal{G}} \subseteq \hat{\mathcal{F}}_{NN}$ be all functions into $[-R, R]^d$ representable on the architecture, and all these functions are at most L-Lipschitz. Then for all $0 < \varepsilon < 2R$,*

$$\mathcal{C}(\varepsilon, \hat{\mathcal{G}}, L^1) \leq \left(\frac{2e \cdot 2R \cdot DL}{\varepsilon}\right)^{2W}$$

Here, we need to prove firstly that the $\hat{\mathcal{F}}$-dependent capacity of $\hat{\mathcal{G}}$ is bounded by a scaled independent capacity on L^1 of itself. We suppose that the MSE loss function (used in the definitions in Table B.1) is bounded by some constant c. This is a reasonable assumption given that the input and output of neural networks are bounded in a compact set. Let us take G an $\frac{\varepsilon}{2\sqrt{c}}$-cover of $\hat{\mathcal{G}}$ with L^1-distance: $d_{L^1(\mathcal{P})}$ (see definition in Table B.1). Therefore, for each $g \in \hat{\mathcal{G}}$ take $g' \in G$ such that $d_{L^1}(g, g') \leq \frac{\varepsilon}{2\sqrt{c}}$, then

$$\begin{aligned}
d_{[\mathcal{P},\hat{\mathcal{F}}]}(g, g') &= \int_{\mathcal{U}\times\mathcal{U}'} \sup_{f\in\hat{\mathcal{F}}} |\|(f+g)(z) - y\|_2^2 - \|(f+g')(z) - y\|_2^2| \mathrm{d}\mathcal{P}(z, y) \\
&\leq \int_{\mathcal{U}\times\mathrm{T}\mathcal{U}} \sup_{f\in\hat{\mathcal{F}}} \|(g-g')(z)\|_2 (\|(f+g)(z) - y\|_2 + \|(f+g')(z) - y\|_2) \mathrm{d}\mathcal{P}(z, y) \\
&\leq 2\sqrt{c} \int_{\mathbb{R}^d} \|(g-g')(x)\|_1 \mathrm{d}\mathcal{P}(z) \leq \varepsilon
\end{aligned}$$

Then we have the first inequality $\mathcal{C}_{\mathcal{F}}(\varepsilon, \hat{\mathcal{G}}) \leq \mathcal{C}(\frac{\varepsilon}{2c}, \hat{\mathcal{G}}, L^1)$. As we suppose that $\|g\|_\infty \leq R$ for all $g \in \hat{\mathcal{G}}$, then for all $g \in \hat{\mathcal{G}}$, we have $g(x) \in [-R, R]^d$. We now apply Theorem B.3 on $\hat{\mathcal{G}}$, we then have the following inequality

$$\log \mathcal{C}\left(\frac{\varepsilon}{2\sqrt{c}}, \hat{\mathcal{G}}, L^1\right) \leq 2W \log \frac{8e\sqrt{c}DRL}{\varepsilon} \tag{B.4}$$

where e is the base of the natural logarithm, W is the number of parameters of the architecture, D is the depth of the architecture. Then if we consider W, c, D as constants, the bound becomes:

$$\log \mathcal{C}\left(\frac{\varepsilon}{2\sqrt{c}}, \hat{\mathcal{G}}, L^1\right) \leq c_1 \log \frac{RL}{\varepsilon} + c_2 = \omega(R, L, \varepsilon) \tag{B.5}$$

for $c_1 = 2W$ and $c_2 = 2W \log 8e\sqrt{c}D$. □

This leads us to choose for $\mathcal{R}$ a strictly increasing function that bounds $\omega(R, L, \varepsilon)$. Given the inequality (Eq. (B.3)), this choice for $\mathcal{R}$ will allow us to bound practically the capacity of $\hat{\mathcal{G}}$.

Minimizing $\mathcal{R}$ will then reduce the effective capacity of the parametric set used to learn g_e. Concretely, we choose for $\mathcal{R}$:

$$\mathcal{R}(g_e) = \|g_e\|_\infty^2 + \alpha \|g_e\|_{\text{Lip}}^2 \tag{7}$$

where $\alpha > 0$ is a hyper-parameter. This function is strictly convex and attains its unique minimum at the null function.

With this choice, let us instantiate Proposition 5.2 for our familly of NNs. Let $r = \sup_{g \in \hat{\mathcal{G}}} \mathcal{R}(g)$, and $\omega(r, \varepsilon) = c_1 \log \frac{r}{\varepsilon\sqrt{\alpha}} + c_2$ (strictly increasing w.r.t. the r) for given parameters $c_1, c_2 > 0$. We have:

Proposition B.4. *If $r = \sup_{g \in \hat{\mathcal{G}}} \mathcal{R}(g)$ is finite, the number of samples n in Eq. (5.5), required to satisfy the error bound in Proposition 5.2 with the same δ, ε, ε_1 and ε_2 becomes:*

$$n \geq \max\left\{ \frac{64}{\varepsilon^2}\left(\frac{1}{m} \log \frac{4\mathcal{C}_{\hat{\mathcal{G}}}(\frac{\varepsilon_1}{16}, \hat{\mathcal{F}})}{\delta} + \omega\left(r, \frac{\varepsilon_2}{16}\right)\right), \frac{16}{\varepsilon^2} \right\} \tag{B.6}$$

Proof. If $\mathcal{R}(g_e) \leq r$, we have $2 \log R \leq \log r$ and $2 \log L + \log \alpha \leq \log r$, then

$$\log RL \leq \log \frac{r}{\sqrt{\alpha}}$$

We can therefore bound $\omega(R, L, \varepsilon)$ by

$$\omega(R, L, \varepsilon) = c_1 \log \frac{RL}{\varepsilon} + c_2 \leq c_1 \log \frac{r}{\varepsilon\sqrt{\alpha}} + c_2 = \omega(r, \varepsilon)$$

The result follows from Proposition B.3. □

This means that the number of required samples will decrease with the size the largest possible $\mathcal{R}(g) = r$. The optimization process will reduce $\mathcal{R}(g_e)$ until a minimum is reached. The maximum size of the effective hypothesis space is then bounded and decreases throughout training. In particular, the following result follows:

Corollary B.5. *Optimizing Eq. (5.4) for a given λ, we have that the number of samples n in Eq. (5.5) required to satisfy the error bound in Proposition 5.2 with the same δ, ε, ε_1 and*

ε_2 is:

$$n \geq \max\left\{\frac{64}{\varepsilon^2}\left(\frac{1}{m}\log\frac{4\mathcal{C}_{\hat{\mathcal{G}}}(\frac{\varepsilon_1}{16},\hat{\mathcal{F}})}{\delta} + \omega\left(\lambda\varkappa, \frac{\varepsilon_2}{16}\right)\right), \frac{16}{\varepsilon^2}\right\} \tag{B.7}$$

where $\varkappa = \sum_{e\in\mathcal{E}}\sum_{i=1}^{N}\sum_{t\in\mathcal{T}}\left\|\frac{\mathrm{d}u_t^{e,(i)}}{\mathrm{d}t}\right\|^2$.

Proof. Denote $\mathcal{L}_\lambda(f,\{g_e\})$ the loss function defining Eq. (5.4). Consider a minimizer $(f^\star,\{g_e^\star\})$ of $\mathcal{L}_\lambda$. Then:

$$\mathcal{L}_\lambda(f^\star,\{g_e^\star\}) \leq \mathcal{L}_\lambda(0,\{0\}) = \varkappa$$

which gives:

$$\forall e,\ \mathcal{R}(g_e^\star) \leq \sum_e \mathcal{R}(g_e^\star) \leq \lambda\varkappa$$

Defining $\hat{\mathcal{G}} = \{g \in \hat{\mathcal{F}} \mid \mathcal{R}(g) \leq \lambda\varkappa\}$, we then have that Eq. (5.4) is equivalent to:

$$\min_{f\in\hat{\mathcal{F}},\{g_e\in\hat{\mathcal{G}}\}_{e\in\mathcal{E}}} \sum_{e\in\mathcal{E}}\left(\frac{\mathcal{R}(g_e)}{\lambda} + \sum_{i=1}^{N}\sum_{t\in\mathcal{T}}\left\|\frac{\mathrm{d}u_t^{e,(i)}}{\mathrm{d}t} - (f+g_e)(u^{e,(i)})\right\|^2 \mathrm{d}\tau\right) \tag{B.8}$$

and the result follows from Proposition B.4. □

We can then decrease the sample complexity in the chosen NN family by: (a) increasing the number of training environments engaged in the framework, and (b) decreasing $\mathcal{R}(g_e)$ for all g_e, with $\mathcal{R}(g_e)$ instantiated as in Section 5.3.1. $\mathcal{R}$ provides a bound based on the largest output norm and the Lipschitz constant for a family of NNs. The experiments (Section 5.4) confirm that this is indeed an effective way to control the capacity of the approximating function family. Note that in our experiments, the number of samples needed in practice is much smaller than suggested by the theoretical bound.

B.3 Optimizing $\mathcal{R}$ in Practice

In Section 5.3.3, we developed an instantiation of the LEADS framework for neural networks. We proposed to control the capacity of the g_es components through a penalization function $\mathcal{R}$ defined as $\mathcal{R}(g_e) = \|g_e\|_\infty^2 + \alpha\|g_e\|_{\mathrm{Lip}}^2$. This definition ensures the properties required to control the sample complexity.

However, in practice, both terms in $\mathcal{R}(g_e)$ are difficult to compute as they do not yield an analytical form for neural networks. For a fixed activation function, the Lipschitz-norm of a trained model only depends on the model parameters and, for our class of neural

Table B.2: Details for the results of evaluation error in test on linear systems in Figure 5.1.

Samples per env.	Method	$m = 1$	$m = 2$	$m = 4$	$m = 8$
$n = 2 \cdot K$	LEADS NO MIN.	8.13±5.56E−2	6.81±4.44E−2	4.92±4.26E−2	4.50±3.10E−2
	LEADS (Ours)	8.13±5.56E−2	**5.11±3.20E−2**	**3.93±2.88E−2**	**2.10±0.96E−2**
$n = 4 \cdot K$	LEADS NO MIN.	4.08±2.57E−2	3.96±2.56E−2	3.10±2.08E−2	2.23±1.44E−2
	LEADS (Ours)	4.08±2.57E−2	**2.74±1.96E−2**	**1.61±1.24E−2**	**1.02±0.74E−2**

networks, can be bounded by the spectral norms of the weight matrices, as described in Section 5.4.4. This allows for a practical implementation.

The infinity norm on its side depends on the domain definition of the function and practical implementations require an empirical estimate. Since there is no trivial estimator for the infinity norm of a function, we performed tests with different proxies such as the *empirical* L^q and L^∞ norms, respectively defined as $\|g_e\|_{L^q(\hat{\mathcal{P}}_e)} = \left(\frac{1}{n}\sum_{z\in\hat{\mathcal{P}}_e} |g_e(z)|^q\right)^{1/q}$ for $1 \leq q < \infty$ and $\|g_e\|_{L^\infty(\hat{\mathcal{P}}_e)} = \max_{z\in\hat{\mathcal{P}}_e} |g_e(z)|$. Here $|\cdot|$ is an ℓ^2 vector norm. Note that on a finite set of points, these norms reduce to vector norms $\|(|g_e(z_1)|, \ldots, |g_e(z_d)|)^\top\|_q$. They are then all equivalent on the space defined by the training set. Table B.4 shows the results of experiments performed on LV equation with different $1 \leq q \leq \infty$. Overall we found that L^q for small values of q worked better and chose in our experiments set $q = 2$.

Moreover, using both minimized quantities $\|g_e\|^2_{L^2(\hat{\mathcal{P}}_e)}$ and the spectral norm of the product of weight matrices, denoted $L(g_e)$ and $\Pi(g_e)$ respectively, we can give a bound on $\mathcal{R}(g_e)$. First, for any z in the compact support of $\mathcal{P}_e$, we have that, fixing some $z_0 \in \hat{\mathcal{P}}_e$:

$$|g_e(z)| \leq |g_e(x) - g_e(x_0)| + |g_e(z_0)|$$

For the first term:

$$|g_e(z) - g_e(z_0)| \leq \|g_e\|_{\text{Lip}}|z - z_0| \leq \Pi(g_e)|z - z_0|$$

and the support of $\mathcal{P}_e$ being compact by hypothesis, denoting by δ its diameter:

$$|g_e(z) - g_e(z_0)| \leq \delta\Pi(g_e)$$

Moreover, for the second term:

$$|g_e(z_0)| = \sqrt{|g_e(z_0)|^2} \leq \sqrt{L(g_e)}$$

and summing both contributions gives us the bound:

$$\|g_e\|_\infty \leq \delta \Pi(g_e) + \sqrt{L(g_e)}$$

so that:

$$\mathcal{R}(g_e) \leq (\delta + \alpha)\Pi(g_e) + \sqrt{L(g_e)}$$

Note that this estimation is a crude one and improvements can be made by considering the closest z_0 from z and taking δ to be the maximal distance between points not from the support of $\mathcal{P}_e$ and $\hat{\mathcal{P}}_e$.

Finally, we noticed that minimizing $\|\frac{g_e}{id}\|^2_{L^2(\hat{\mathcal{P}}_e)}$ in domains bounded away from zero gave better results as normalizing by the norm of the output allowed to adaptively rescale the computed norm. Formally, minimizing this quantity does not fundamentally change the optimization as we have that:

$$\forall u, \frac{1}{M^2}|g_e(z)|^2 \leq \left|\frac{g_e(z)}{z}\right|^2 \leq \frac{1}{m^2}|g_e(z)|^2$$

meaning that:

$$\frac{1}{M^2}L(g_e) \leq \left\|\frac{g_e}{id}\right\|^2_{L^2(\hat{\mathcal{P}}_e)} \leq \frac{1}{m^2}L(g_e)$$

where m, M are the lower and upper bound of $|z|$ on the support of $\mathcal{P}_e$ with $m > 0$ by hypothesis (the quantity we minimize is still higher than $L(g_e)$ even if this is not the case).

B.4 Additional Experimental Details

B.4.1 Details on the Environment Dynamics

Lotka-Volterra. The model dynamics follow the ODE:

$$\frac{\mathrm{d}x}{\mathrm{d}t} = \alpha x - \beta x y$$
$$\frac{\mathrm{d}y}{\mathrm{d}t} = \delta x y - \gamma y$$

with v, w the number of prey and predator, $\alpha, \beta, \gamma, \delta > 0$ defining how the two species interact. The initial conditions v_0^i, w_0^i are sampled from a uniform distribution $P_0 = \mathrm{Unif}([1,2]^2)$. We characterize the dynamics by $\theta = (\alpha/\beta, \gamma/\delta) \in \Theta = \{0.5, 1, 1.44, 1.5, 1.86, 2\}^2$. An environment e is then defined by parameters θ_e sampled from a uniform distribution over the parameter set Θ.

Gray-Scott. The governing PDE is:

$$\frac{\partial v}{\partial t} = D_v \Delta v - v w^2 + F_r(1 - v)$$
$$\frac{\partial w}{\partial t} = D_w \Delta w + v w^2 - (F_r + k_r) w$$

where the v, w represent the concentrations of two chemical components in the spatial domain Ω with periodic boundary conditions. D_v, D_w denote the diffusion coefficients respectively for v, w, and are held constant to $D_v = 0.2097, D_w = 0.105$, and F_r, k_r are the reaction parameters depending on the environment. As for the initial conditions $(v_0, w_0) \sim P_0$, we place 3 2-by-2 squares at uniformly sampled positions in Ω to trigger the reactions. The values of (v_0, w_0) are fixed to $(0, 1)$ outside the squares and to $(1 - \epsilon, \epsilon)$ with a small $\epsilon > 0$ inside. An environment e is defined by its parameters $\theta_e = (F_{r,e}, k_{r,e}) \in \Theta = \{(0.037, 0.060), (0.030, 0.062), (0.039, 0.058)\}$. We consider a set of θ_e parameters uniformly sampled from the environment distribution Q on Θ.

Navier-Stokes. We consider the Navier-Stokes PDE for incompressible flows:

$$\frac{\partial w}{\partial t} = -v \cdot \nabla w + \nu \Delta w + f^{\text{fc.}} \qquad \nabla \cdot v = 0$$

where v is the velocity field, $w = \nabla \times v$ is the vorticity, both v, w lie in a spatial domain Ω with periodic boundary conditions, ν is the viscosity and h is the constant forcing term in the domain Ω. We fix $\nu = 10^{-3}$ across the environments. We sample the initial conditions $w_0^e \sim P_0$ as in Li et al. (2021b). An environment e is defined by its forcing term $f_e^{\text{fc.}} \in \Theta_{f^{\text{fc.}}} = \{f_1^{\text{fc.}}, f_2^{\text{fc.}}, f_3^{\text{fc.}}, f_4^{\text{fc.}}\}$ with

$$f_1^{\text{fc.}}(x_1, x_2) = 0.1(\sin(2\pi(x_1 + x_2)) + \cos(2\pi(x_1 + x_2)))$$
$$f_2^{\text{fc.}}(x_1, x_2) = 0.1(\sin(2\pi(x_1 + x_2)) + \cos(2\pi(x_1 + 2x_2)))$$
$$f_3^{\text{fc.}}(x_1, x_2) = 0.1(\sin(2\pi(x_1 + x_2)) + \cos(2\pi(2x_1 + x_2)))$$
$$f_4^{\text{fc.}}(x_1, x_2) = 0.1(\sin(2\pi(2x_1 + x_2)) + \cos(2\pi(2x_1 + x_2)))$$

where $x = (x_1, x_2) \in \Omega$ is the position in the domain Ω. We uniformly sampled a set of forcing terms from Q on Θ_h.

Linear ODE. We take an example of linear ODE expressed by the following formula:

$$\frac{\mathrm{d}u}{\mathrm{d}t} = \mathrm{L}_{Q\Lambda Q^\top}(u) = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^\top u$$

where $u_t \in \mathbb{R}^8$ is the system state, $\boldsymbol{Q} \in M_{8,8}(\mathbb{R})$ is an orthogonal matrix such that $\boldsymbol{Q}\boldsymbol{Q}^\top = 1$, and $\boldsymbol{\Lambda} \in M_{8,8}(\mathbb{R})$ is a diagonal matrix containing eigenvalues. We sample $\boldsymbol{\Lambda}_e$ from a uniform distribution on $\Theta_\Lambda = \{\boldsymbol{\Lambda}_1, \ldots, \boldsymbol{\Lambda}_8\}$, defined for each $\boldsymbol{\Lambda}_i$ by:

$$[\boldsymbol{\Lambda}_i]_{jj} = \begin{cases} 0, & \text{if } i = j \text{ for } i, j \in \{1, \ldots, 8\}, \\ -0.5, & \text{otherwise.} \end{cases}$$

which means that the i-th eigenvalue is set to 0, while others are set to a common value -0.5.

B.4.2 Choosing Hyperparameters

As usual, the hyperparameters need to be tuned for each considered set of systems. We therefore chose the hyperparameters using standard cross-validation techniques. We did not conduct a systematic sensitivity analysis. In practice, we found that: (a) if the regularization term is too large w.r.t. the trajectory loss, the model cannot fit the trajectories, and (b) if the regularization term is too small, the performance is similar to LEADS NO MIN. The candidate hyperparameters are defined on a very sparse grid, for example, for neural nets, $(10^3, 10^4, 10^5, 10^6)$ for λ and $(10^{-2}, 10^{-3}, 10^{-4}, 10^{-5})$ for α.

B.4.3 Details on the Experiments with a Varying Number of Environments

We conducted large-scale experiments respectively for linear ODEs (Section 5.3.2, Figure 5.1) and LV (Section 5.4, Figure 5.4) to compare the tendency of LEADS w.r.t. the theoretical bound and the baselines by varying the number of environments available for the instantiated model.

To guarantee the comparability of the test-time results, we need to use the same test set when varying the number of environments. We therefore propose to firstly generate a global set of environments, separate it into subgroups for training, then we test these separately trained models on the global test set.

We performed the experiments as follows:

- In the training phase, we consider $M = 8$ environments in total in the environment set $\mathcal{E}_{\text{total}}$. We denote here the cardinality of an environment set $\mathcal{E}$ by $\text{card}(\mathcal{E})$, the environments are then arranged into $b = 1, 2, 4$ or 8 disjoint groups of the same size, i.e. $\{\mathcal{E}_1, \ldots, \mathcal{E}_b\}$ such that $\bigcup_{i=1}^{b} \mathcal{E}_i = \mathcal{E}_{\text{total}}$, $\text{card}(\mathcal{E}_1) = \cdots = \text{card}(\mathcal{E}_b) = \lfloor M/b \rfloor =: m$, where m is the number of environments per group, and $\mathcal{E}_i \cap \mathcal{E}_j = \emptyset$ whenever $i \neq j$. For example, for $m = 1$, all the original environments are gathered into one global environment, when for $m = 8$ we keep all the original environments. The

methods are then instantiated respectively for each $\mathcal{E}_i$. For example, for LEADS with b environment groups, we instantiate $\text{LEADS}_1, \cdots, \text{LEADS}_b$ respectively on $\mathcal{E}_1, \cdots, \mathcal{E}_b$. Other frameworks are applied in the same way.

Note that when $m = 1$, having $b = 8$ environment groups of one single environment, One-For-All, One-Per-Env. and LEADS are reduced to One-Per-Env. applied on all M environments. We can see in Figure 5.4 that each group of plots starts from the same point.

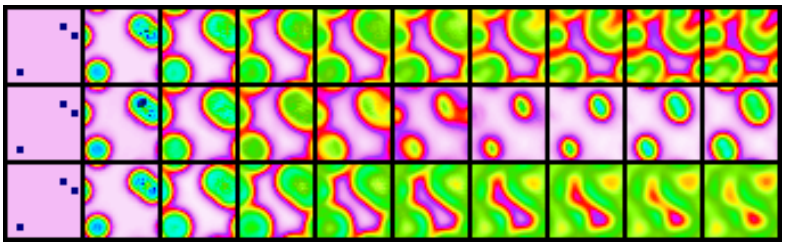
- In the test phase, the performance of the model trained with the group $\mathcal{E}_i$ is tested with the test samples of the corresponding group. Then we take the mean error over all b groups to obtain the results on all M environments. Note that the result at each point in figs. 5.1 and 5.4 is calculated on the same total test set, which guarantees the comparability between results.

B.4.4 Additional Experimental Results

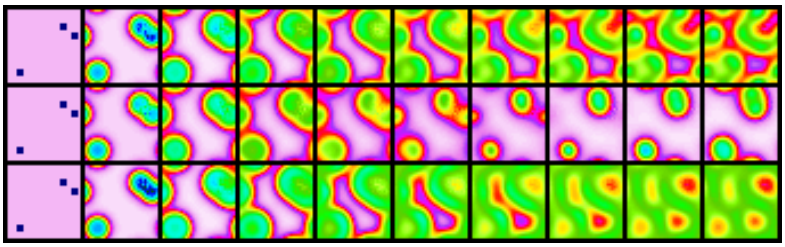
Experiments with a varying number of environments We show in Tables B.2 and B.3 the detailed results used for the plots in Figures 5.1 and 5.4, compared to baseline methods.

Learning in novel environments We conducted same experiments as in Section 5.4.3 to learn in unseen environments for *Gray-Scott* and *Navier-Stokes* datasets. The test MSE at different training steps is shown in Table B.5.

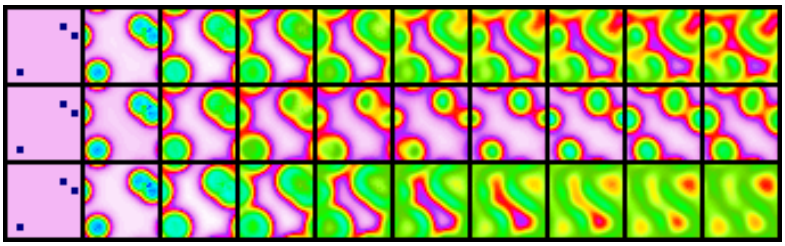
Full-length trajectories We provide in figures S1-S4 the full-length sample trajectories for *Gray-Scott* and *Navier-Stokes* of Figure 5.2.



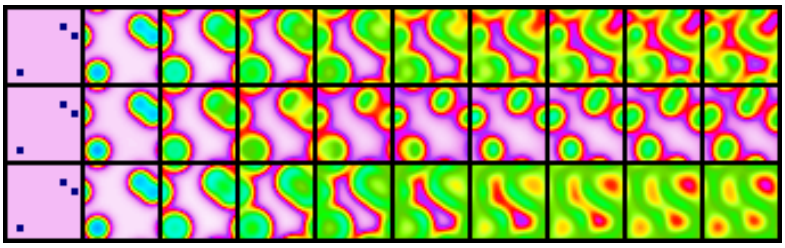
(a) One-Per-Env.



(b) FT-NODE

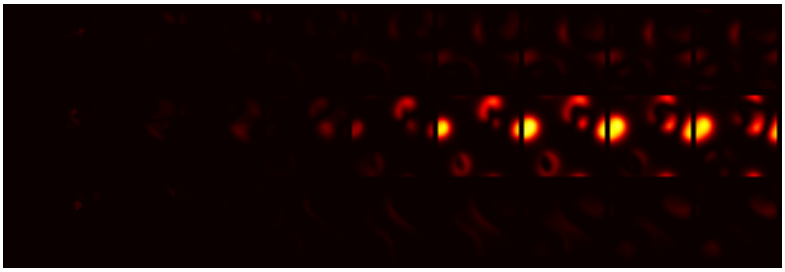


(c) LEADS

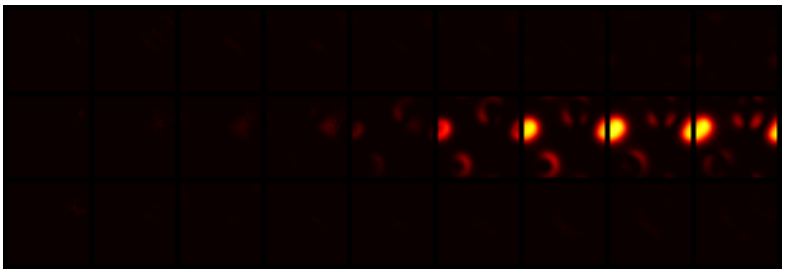


(d) Ground truth

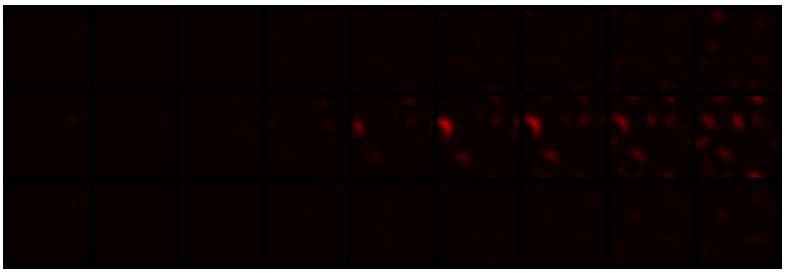
Figure B.1: Full-length prediction comparison of Figure 5.2 for *Gray-Scott*. In each figure, from top to bottom, the trajectory snapshots are output respectively from 3 training environments. The temporal resolution of each sequence is $\delta t = 40$.



(a) Difference between One-Per-Env. and Ground truth

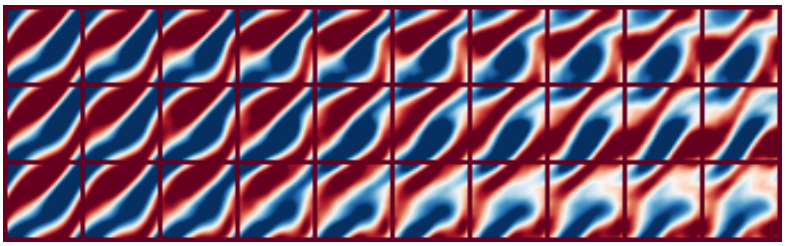


(b) Difference between FT-NODE and Ground truth

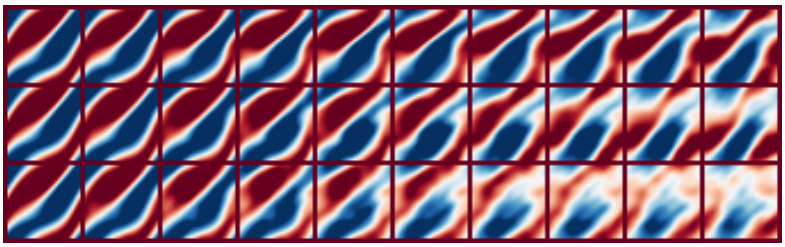


(c) Difference between LEADS and Ground truth

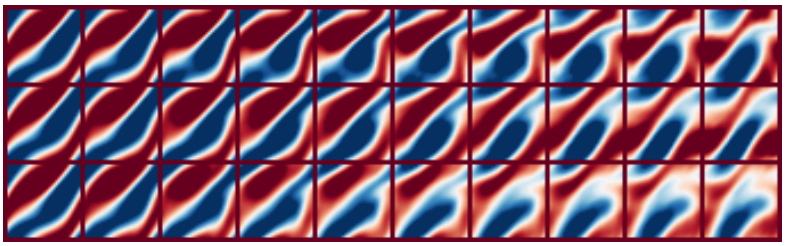
Figure B.2: Full-length error maps of Figure 5.2 for *Gray-Scott*. In each figure, from top to bottom, the trajectory snapshots correspond to 3 training environments, one per row. The temporal resolution of each sequence is $\delta t = 40$.



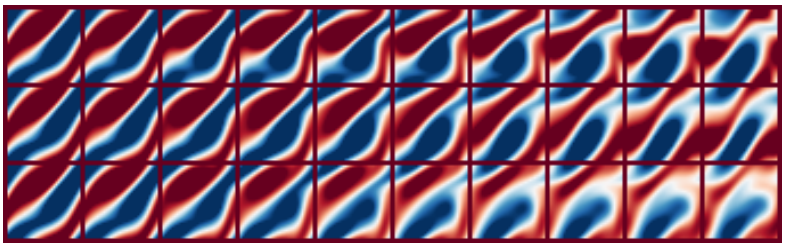
(a) One-Per-Env.



(b) FT-NODE

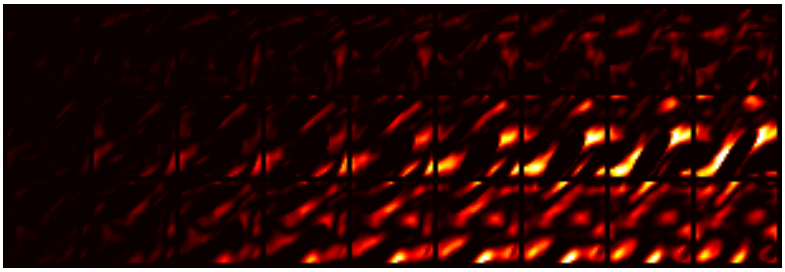


(c) LEADS

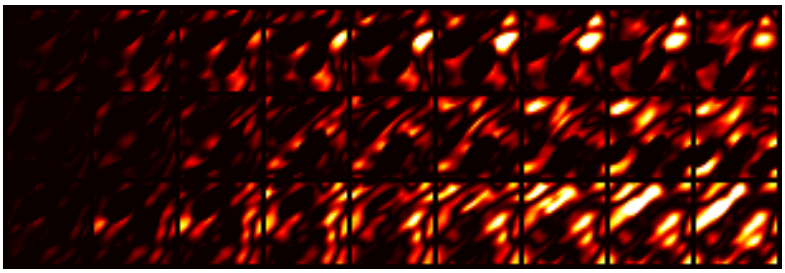


(d) Ground truth

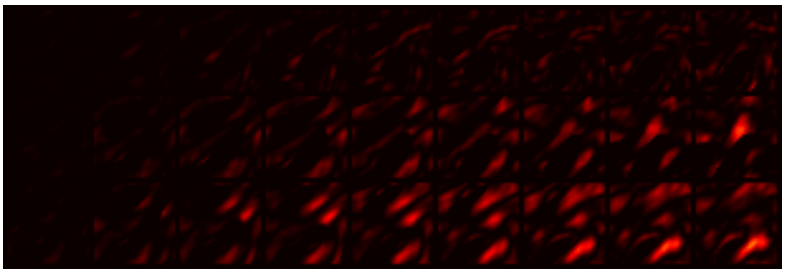
Figure B.3: Full-length prediction comparison of Figure 5.2 for *Navier-Stokes*. In each figure, from top to bottom, the trajectory snapshots correspond to 3 training environments. The temporal resolution of each sequence is $\delta t = 1$.



(a) Difference between One-Per-Env. and Ground truth



(b) Difference between FT-NODE and Ground truth



(c) Difference between LEADS and Ground truth

Figure B.4: Full-length error maps of Figure 5.2 for *Navier-Stokes*. In each figure, from top to bottom, the trajectory snapshots correspond to from 3 training environments. The temporal resolution of each sequence is $\delta t = 1$.

Table B.3: Detailed results of evaluation error in test on *Lotka-Volterra* systems for Figure 5.4. For the case of $m = 1$, all baselines except FT-RNN are equivalent to One-Per-Env.. The arrows indicate that the table cells share the same value.

Samples per env.	Method	$m = 1$	$m = 2$	$m = 4$	$m = 8$
$n = 1 \cdot K$	One-For-All	7.87±7.54e−3	0.22±0.06	0.33±0.06	0.47±0.04
	One-Per-Env.	7.87±7.54e−3	⟶	⟶	⟶
	FT-RNN	4.02±3.17e−2	1.62±1.14e−2	1.62±1.40e−2	1.08±1.03e−2
	FT-NODE	7.87±7.54e−3	7.63±5.84e−3	4.18±3.77e−3	4.92±4.19e−3
	GBML-like	7.87±7.54e−3	6.32±5.72e−2	1.44±0.66e−1	9.85±8.84e−3
	LEADS (Ours)	7.87±7.54e−3	**3.65±2.99e−3**	**2.39±1.83e−3**	**1.37±1.14e−3**
$n = 2 \cdot K$	One-For-All	1.38±1.61e−3	0.22±0.04	0.36±0.07	0.60±0.11
	One-Per-Env.	1.38±1.61e−3	⟶	⟶	⟶
	FT-RNN	7.20±7.12e−2	2.72±4.00e−2	1.69±1.57e−2	1.38±1.25e−2
	FT-NODE	1.38±1.61e−3	9.02±8.81e−3	1.11±1.05e−3	1.00±0.95e−3
	GBML-like	1.38±1.61e−3	9.26±8.27e−3	1.17±1.09e−2	1.96±1.95e−2
	LEADS (Ours)	1.38±1.61e−3	**8.65±9.61e−4**	**8.40±9.76e−4**	**6.02±6.12e−4**
$n = 4 \cdot K$	One-For-All	1.36±1.25e−4	0.19±0.02	0.31±0.04	0.50±0.04
	One-Per-Env.	1.36±1.25e−4	⟶	⟶	⟶
	FT-RNN	8.69±8.36e−4	3.39±3.38e−4	3.02±1.50e−4	2.26±1.45e−4
	FT-NODE	1.36±1.25e−4	1.74±1.65e−4	1.78±1.71e−4	1.39±1.20e−4
	GBML-like	1.36±1.25e−4	2.57±7.18e−3	2.65±3.26e−3	2.36±3.58e−3
	LEADS (Ours)	1.36±1.25e−4	**1.10±0.92e−4**	**1.03±0.98e−4**	**9.66±9.79e−5**
$n = 8 \cdot K$	One-For-All	5.98±5.13e−5	0.16±0.03	0.35±0.06	0.52±0.06
	One-Per-Env.	5.98±5.13e−5	⟶	⟶	⟶
	FT-RNN	2.09±1.73e−4	1.18±1.16e−4	1.13±1.13e−4	9.13±8.31e−5
	FT-NODE	5.98±5.13e−5	6.91±4.46e−5	7.82±6.95e−5	6.88±6.39e−5
	GBML-like	5.98±5.13e−5	1.02±1.68e−4	1.41±2.68e−4	0.99±1.53e−4
	LEADS (Ours)	5.98±5.13e−5	**5.47±4.63e−5**	**4.52±3.98e−5**	**3.94±3.49e−5**

Table B.4: Test MSE of experiments on *Lotka-Volterra* ($m = 4, n = 1 \cdot K$) with different empirical norms.

Empirical Norm L^q	$q = 1$	$q = 2$	$q = 3$	$q = 10$	$q = \infty$
Test MSE	2.30e−3	2.36e−3	2.34e−3	3.41e−3	6.12e−3

Table B.5: Results on 2 novel environments for *Lotka-Volterra*, *Gray-Scott*, and *Navier-Stokes* at different training steps with n data points per env. The arrows indicate that the table cells share the same value.

Dataset	Training Schema	Test MSE at training step		
		50	2500	10000
Lotka-Volterra	Pre-trained-f-Only	0.360	⟶	⟶
($n = 1 \cdot K$)	One-Per-Env. from scratch	0.230	8.85e−3	3.05e−3
	Pre-trained-f-Plus-Trained-g_e	0.730	**1.36e−3**	**1.11e−3**
Gray-Scott	Pre-trained-f-Only	5.44e−3	⟶	⟶
($n = 1 \cdot K$)	One-Per-Env. from scratch	4.20e−2	5.53e−3	3.05e−3
	Pre-trained-f-Plus-Trained-g_e	2.29e−3	**1.45e−3**	**1.27e−3**
Navier-Stokes	Pre-trained-f-Only	1.75e−1	⟶	⟶
($n = 8 \cdot K$)	One-Per-Env. from scratch	6.76e−2	1.70e−2	1.18e−2
	Pre-trained-f-Plus-Trained-g_e	1.37e−2	**8.07e−3**	**7.14e−3**

Appendix C

Appendix of Chapter 6

C.1 Discussion

We discuss in more detail the originality and differences of CoDA w.r.t. several multi-task learning (MTL) and gradient-based meta learning (GBML) or contextual meta-learning methods illustrated in Figure C.1. We consider CAVIA (Zintgraf et al., 2019), MAML (Finn et al., 2017), ANIL (Raghu et al., 2020), hard-parameter sharing MTL (Caruana, 1998; Ruder, 2017), LEADS (Yin et al., 2021a).

C.1.1 Adaptation Rule

We compare the adaptation rule in Eq. (6.4) w.r.t. these work.

GBML Given k gradient steps, Model-agnostic meta learning (MAML; Finn et al., 2017) defines

$$\theta^e = \theta^c + (-\eta \sum_{i=0}^{k} \nabla_\theta \mathcal{L}(\theta_i^e, \mathcal{D}^e)) \tag{C.1}$$

$$\text{where} \begin{cases} \theta_{i+1}^e = \theta_i^e - \eta \nabla_\theta \mathcal{L}(\theta_i^e, \mathcal{D}^e) & i > 0 \\ \theta_0^e = \theta^c & i = 0 \end{cases}$$

With $\delta\theta^e := -\eta \sum_{i=0}^{k} \nabla_\theta \mathcal{L}(\theta_i^e, \mathcal{D}^e)$, Eq. (6.4) thus includes MAML. ANIL and related GBML methods (Lee et al., 2019; Bertinetto et al., 2019) restrict Eq. (C.1) to parameters of the final layer while the remaining parameters are shared.

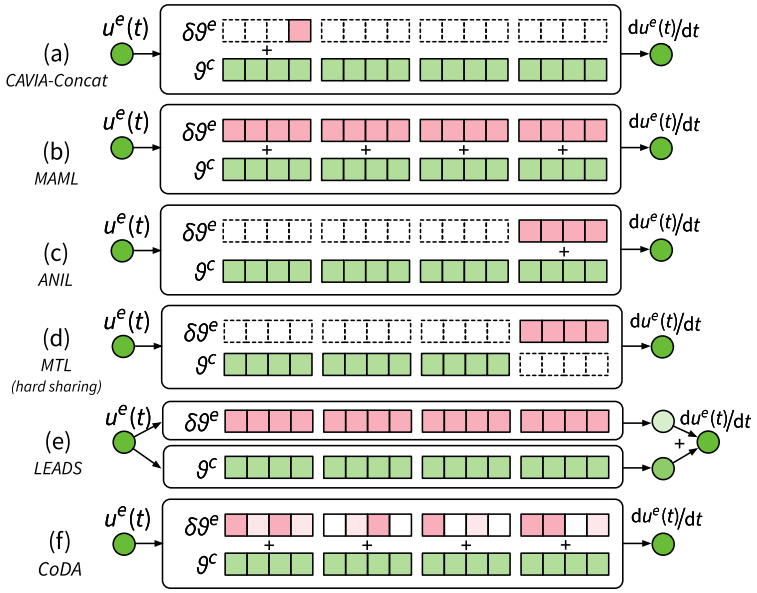


Figure C.1: Illustration of representative baselines for multi-environment learning. Shared parameters are blue, environment-specific parameters are red. (a) CAVIA-Concat acts upon the bias of the first layer with conditioning via concatenation. (b) MAML acts upon all parameters without penalization nor prior structure information. (c) ANIL restricts meta-learning to the final layer. (d) Hard-sharing MTL train the final layer from scratch, while the remaining is a *hard-shared*. (e) LEADS sums the output of a common and a environment-specific network. (f) CoDA acts upon a subspace of the parameter space with a locality constraint.

MTL MTL models can be identified to Eq. (C.1). They fix $\theta^c = 0$, removing the ability to perform fast adaptation as parameters are retrained from scratch instead of being initialized to θ^c. Hard-parameter sharing MTL restricts the sum in Eq. (C.1) to the final layer, as ANIL. LEADS sums the outputs of a shared and an environment-specific network, thus splitting parameters into two independent blocks that do not share connections.

C.1.2 Decoding for Context-Informed Adaptation

We show that conditioning strategies in contextual meta-learning for decoding context vectors ξ^e into $\delta\theta^e$ are a special case of hypernetwork-decoding. The two main approaches are conditioning via concatenation and conditioning via feature modulation a.k.a. FiLM (Perez et al., 2018).

Conditioning via Concatenation

We show that conditioning via concatenation is equivalent to a linear hypernetwork $A_\phi : \xi^e \mapsto W\xi^e + \theta^c$ with $\phi = \{\theta^c, W\}$ that only predicts the bias of the first layer of g_θ.

We assume that g_θ has L layers and analyze the output of the first layer of g_θ, omitting the nonlinearity, when the input $u_t \in \mathbb{R}^d$ in an environment $e \in \mathcal{E}$ is concatenated to a context vector $\xi^e \in \mathbb{R}^{d_\xi}$. We denote $u_t \| \xi^e$ the concatenated vector, d_h the number of hidden units of the first layer, $W^{(1)} \in \mathbb{R}^{d_h \times (d_x + d_\xi)}$ and $b^{(1)} \in \mathbb{R}^{d_h}$ the weight matrix and bias term of the first layer, $W^{(2)}, \cdots, W^{(L)}$ and $b^{(2)}, \cdots, b^{(L)}$ those of the following layers. The output of the first layer is

$$y^{(1)} = W^{(1)}(u_t \| \xi^e) + b^{(1)} \tag{C.2}$$

We split $W^{(1)}$ along rows into two weight matrices, $W_u^{(1)} \in \mathbb{R}^{d_h \times d}$ and $W_\xi^{(1)} \in \mathbb{R}^{d_h \times d_\xi}$ s.t.

$$y^{(1)} = W_u^{(1)} u_t + W_\xi^{(1)} \xi^e + b^{(1)} \tag{C.3}$$

$b_\xi^{(1)} := W_\xi^{(1)} \xi^e + b^{(1)}$ does not depend on x and corresponds to an environment-specific bias. Thus, concatenation is included in Eq. (6.4) when

$$\begin{aligned} \theta^c &:= \{W_x^{(1)}, b^{(1)}, W^{(2)}, b^{(2)}, \cdots, W^{(L)}, b^{(L)}\} \\ \delta\theta^e &:= \{\ 0\ , b_\xi^{(1)},\ 0\ , 0, \cdots,\ 0\ , 0\} \end{aligned} \tag{C.4}$$

where $\delta\theta^e$ is decoded via a hypernetwork with parameters $\{\theta^c, W := (0, W_\xi^{(1)}, 0, \cdots, 0)\}$.

Conditioning via Feature Modulation

We show that conditioning via FiLM is equivalent to a linear hypernetwork $A_\phi : \xi^e \mapsto W\xi^e + \theta^c$ with $\phi = \{\theta^c, W\}$ that only predicts the batch norm (BN) statistics of g_θ.

For simplicity, we focus on a single BN layer and denote $\{h_i\}_{i=1}^M$, M feature maps output by preceding convolutional layers. These feature maps are first normalized then rescaled with an affine transformation. Rescaling is similar to a FiLM layer that transforms linearly $\{h_i\}_{i=1}^M$ with:

$$\forall i \in \{1, \cdots, M\}, \text{FiLM}(h_i) = \gamma_i \odot h_i + \beta \tag{C.5}$$

where $\gamma, \beta \in \mathbb{R}^M$ are output by a NN f_ψ conditioned on the context vectors ξ^e i.e. $[\gamma, \beta] = f_\psi(\xi^e)$. In general, f_ψ is linear s.t. $f_\psi(\xi^e) := W_\xi \xi^e + b_\xi$, with $\psi = \{W_\xi, b_\xi\}$. Then $\gamma = W_\xi^\gamma \xi^e + b_\xi^\gamma, \beta = W_\xi^\beta \xi^e + b_\xi^\beta$.

Thus, for this layer, modulation is included in Eq. (6.4) when

$$\delta\theta^e := W\xi^e = \{W_\xi^\gamma \xi^e, W_\xi^\beta \xi^e\}$$
$$\theta^c := b_\xi = \{b_\xi^\gamma, b_\xi^\beta\}$$

where $\delta\theta^e$ is decoded via hypernetwork $f_\psi := A_\phi$ with parameters $\phi = \{\theta^c := b_\xi, W := W_\xi\}$.

C.2 Proofs

Proposition 6.1. *Given $\{\theta^c, W\}$ fixed, if $\|\cdot\| = \ell_2$, then Eq. (6.9) is quadratic. If $\lambda' W^\top W$ or $\bar{H}^e(\theta^c) = W^\top \nabla_\theta^2 \mathcal{L}(\theta^c, \mathcal{D}^e) W$ are invertible then $\bar{H}^e(\theta^c) + \lambda' W^\top W$ is invertible except for a finite number of λ' values. The problem in Eq. (6.9) is then also convex and admits an unique solution, $\{\xi^{e\star}\}_{e \in \mathcal{E}_{ad}}$. With $\lambda' := 2\lambda$,*

$$\xi^{e*} = -\left(\bar{H}^e(\theta^c) + \lambda' W^\top W\right)^{-1} W^\top \nabla_\theta \mathcal{L}(\theta^c, \mathcal{D}^e) \tag{6.10}$$

Proof. When $\|\cdot\| = \ell_2$, we consider the following second order Taylor expansion of $\mathcal{L}_r(\theta, \mathcal{D}^e) := \mathcal{L}(\theta, \mathcal{D}^e) + \lambda \|\theta - \theta^c\|_2^2$ at θ^c, where $\delta\theta^e = \theta - \theta^c = W\xi^e$.

$$\mathcal{L}_r(\theta^c + \delta\theta^e, \mathcal{D}^e) = \mathcal{L}(\theta^c, \mathcal{D}^e) + \nabla_\theta \mathcal{L}(\theta^c, \mathcal{D}^e)^\top \delta\theta^e + \frac{1}{2} \delta\theta^{e\top} \left(\nabla_\theta^2 \mathcal{L}(\theta^c, \mathcal{D}^e) + 2\lambda \text{Id}\right) \delta\theta^e + o(\|\delta\theta^e\|_2^3) \tag{C.6}$$

With $\delta\theta^e = W\xi^e$, we expand Eq. (C.6) into

$$\mathcal{L}_{\mathrm{r}}(\theta^c + W\xi^e, \mathcal{D}^e) = \mathcal{L}(\theta^c, \mathcal{D}^e) + (W^\top \nabla_\theta \mathcal{L}(\theta^c, \mathcal{D}^e))^\top \xi^e + \frac{1}{2}\xi^{e\top}(W^\top \nabla^2_\theta \mathcal{L}(\theta^c, \mathcal{D}^e)W + 2\lambda W^\top W)\xi^e + o(\|\delta\theta^e\|_2^3) \quad \text{(C.7)}$$

i.e. with $\bar{H}^e(\theta^c) = W^\top \nabla^2_\theta \mathcal{L}(\theta^c, \mathcal{D}^e)W$ and $\lambda' = 2\lambda$

$$\mathcal{L}_{\mathrm{r}}(\theta^c + W\xi^e, \mathcal{D}^e) = \mathcal{L}(\theta^c, \mathcal{D}^e) + (W^\top \nabla_\theta \mathcal{L}(\theta^c, \mathcal{D}^e))^\top \xi^e + \frac{1}{2}\xi^{e\top}\left(\bar{H}^e(\theta^c) + \lambda' W^\top W\right)\xi^e + o(\|\delta\theta^e\|_2^3) \quad \text{(C.8)}$$

Eq. (C.8) is quadratic. If $\bar{H}^e(\theta^c) + \lambda' W^\top W$ is invertible, then the problem is also convex with a unique solution

$$\xi^{e*} = -\left(\bar{H}^e(\theta^c) + \lambda' W^\top W\right)^{-1} W^\top \nabla_\theta \mathcal{L}(\theta^c, \mathcal{D}^e) \quad \text{(C.9)}$$

$\bar{H}^e(\theta^c)$ and $\lambda' W^\top W$ are two square matrices. The application $p : \lambda' \mapsto \det(\bar{H}^e(\theta^c) + \lambda' W^\top W)$ is well-defined and forms a continuous polynomial. Thus either it equals zero or it has a finite number of roots. If $\bar{H}^e(\theta^c)$ or $\lambda' W^\top W$ is invertible, then $p(0) = \det(\bar{H}^e(\theta^c)) \neq 0$ or $p(\infty) \sim \det(\lambda' W^\top W) \neq 0$. Thus $p \neq 0$ has a finite number of roots, i.e., $\bar{H}^e(\theta^c) + \lambda' W^\top W$ is invertible $\forall \lambda'$ except for a finite number of values corresponding to the roots of p. □

Proposition 6.2 (Low-rank under linearity). *Given a class of linearly parametrized dynamics $\mathcal{F}$ with d_{P} varying parameters, $\forall \theta^c \in \mathbb{R}^{d_\theta}$, subspace $\mathcal{G}_{\theta^c}$ in Definition 6.1 is low-dimensional and* $\dim(\mathcal{G}_{\theta^c}) \leq d_{\mathrm{P}} \ll d_\theta$.

Proof. We define the linear mapping $\psi\colon \theta_{\mathrm{P}} \in \mathbb{R}^{d_{\mathrm{P}}} \to f \in \mathcal{F}$ from parameters to dynamics s.t. $\psi(\mathbb{R}^{d_{\mathrm{P}}}) = \mathcal{F}$. Given this linear mapping, we first prove the following lemma: $\dim(\mathcal{F}) \leq d_{\mathrm{P}}$. The proof is based on the surjectivity of ψ onto $\mathcal{F}$, given by definition. We define $\{b_i\}_{i=1}^{d_{\mathrm{P}}}$ a basis of $\mathbb{R}^{d_{\mathrm{P}}}$. Given $f \in \mathcal{F}, \exists \theta_{\mathrm{P}} \in \mathbb{R}^{d_{\mathrm{P}}}, \psi(\theta_{\mathrm{P}}) = f$. We note $\theta_{\mathrm{P}} = \sum_{i=1}^{d_{\mathrm{P}}} \lambda_i b_i$ where $\forall i, \lambda_i \in \mathbb{R}$. Then $\psi(\theta_{\mathrm{P}}) = \sum_{i=1}^{d_{\mathrm{P}}} \lambda_i \psi(b_i)$. We extract a basis from $\{\psi(b_i)\}_{i=1}^{d_{\mathrm{P}}}$ and denote $d_f \leq d_{\mathrm{P}}$ the number of elements in this basis. This basis forms a basis of $\mathcal{F}$, i.e., $d_f = \dim(\mathcal{F}) \leq d_{\mathrm{P}}$.

Now, given $\theta \in \mathbb{R}^{d_\theta}$ and $f^e \in \mathcal{F}$. We precise that given a (probability) measure $\rho_{\mathcal{U}}$ on $\mathcal{U} \subset \mathbb{R}^d$, the function space $\mathcal{F} \subset L^2(\rho_x, \mathbb{R}^d)$, then

$$\mathcal{L}(\theta, \mathcal{D}^e) := \int_{\mathcal{X}} \|(f^e - g_\theta)(u_t)\|_2^2 \mathrm{d}\rho_{\mathcal{U}}(u_t) = \|f^e - g_\theta\|_2^2 \quad \text{(C.10)}$$

The gradient of $\mathcal{L}(\theta, \mathcal{D}^e)$ is then

$$\begin{aligned}
\nabla_\theta \mathcal{L}(\theta^c, \mathcal{D}^e) &= \nabla_\theta \int_{\mathcal{U}} \|f^e(u_t) - g_{\theta^c}(u_t)\|_2^2 \mathrm{d}\rho_{\mathcal{U}}(u_t) \\
&\qquad\qquad \text{(By definition of the derivative loss function)} \\
&= \int_{\mathcal{U}} \nabla_\theta \|f^e(u_t) - g_{\theta^c}(u_t)\|_2^2 \mathrm{d}\rho_{\mathcal{U}}(u_t) \quad \text{(Linearity of the gradient operator)} \\
&= -2 \int_{\mathcal{U}} \mathbf{J}_\theta g_{\theta^c}(u_t)^\top (f^e(u_t) - g_{\theta^c}(u_t)) \mathrm{d}\rho_{\mathcal{U}}(u_t) \quad \text{(Chain rule)} \\
&= -2 D_\theta g_{\theta^c}^\top (f^e - g_{\theta^c}) \quad \text{(Rewritten)}
\end{aligned}$$

where $\mathbf{J}_\theta g_\theta(u)$ is the Jacobian matrix of g_{θ^c} w.r.t. θ at point u. $\theta \mapsto D_\theta g_{\theta^c}$ is the differential of g_θ. Note that $D_\theta g_{\theta^c} : \mathbb{R}^{d_\theta} \to \mathcal{F}$ is a linear map (analog of the Jacobian matrix). $D_\theta g_{\theta^c}^\top : \mathcal{F}^\star \to \mathbb{R}^{d_\theta}$ denotes its adjoint (analog of the transposed matrix), which is also a linear map.

As $\mathcal{G}_{\theta^c} \subseteq \mathrm{Im}(D_\theta g_\theta^\top)$, then according to Rank-nullity theorem,

$$\dim(\mathcal{G}_{\theta^c}) \leq \dim(\mathrm{Im}(D_\theta g_{\theta^c}^\top)) = \dim(\mathcal{F}) - \dim(\mathrm{Ker}(D_\theta g_{\theta^c}^\top)) \leq \dim(\mathcal{F}) \leq d_{\mathrm{P}}. \quad \text{(C.11)}$$

□

C.3 System Parameter Estimation

Proposition 6.3 (Identification under linearity). *Under Assumptions 6.1 to 6.5, system parameters are perfectly identified on new environments if the dynamics model g and hypernetwork A satisfy $\forall f \in \mathcal{B}$ with system parameter θ_{P}, $g_{A(\theta_{\mathrm{P}})} = f$.*

Proof. We define the linear mapping $\psi : \theta_{\mathrm{P}} \in \mathbb{R}^{d_{\mathrm{P}}} \to f \in \mathcal{F}$ from parameters to dynamics s.t. $\psi(\mathbb{R}^{d_{\mathrm{P}}}) = \mathcal{F}$ (Assumption 6.1). Unicity of parameters (Assumption 6.3) implies that ψ is bijective with inverse ψ^{-1}, thus $\dim(\mathcal{F}) = \dim(\mathbb{R}^{d_{\mathrm{P}}}) = d_{\mathrm{P}}$. Given a basis $\mathcal{B} = \{f_i\}_{i=1}^{d_{\mathrm{P}}}$ of $\mathcal{F}$, we denote $\theta_{\mathrm{P},i} = \psi^{-1}(f_i)$. We fix g, A s.t. $\forall i \in \{1, ..., d_{\mathrm{P}}\}, g_{A(p_i)} = f_i = \psi(\theta_{\mathrm{P},i})$. This is possible as f_i and g are linear w.r.t. inputs (Assumptions 6.1 and 6.2) and $\theta_{\mathrm{P},i}$ are known (Assumption 6.5).

g and A are both linear (Assumption 6.2), thus $g_{A(\cdot)}$ is linear with inputs in $\mathbb{R}^{d_\xi}$. Then, $\dim(\mathrm{Im}(g_{A(\cdot)})) \leq d_\xi$. Moreover, $\forall i \in \{1, \cdots, d_{\mathrm{P}}\}, f_i \in \mathrm{Im}(g_{A(\cdot)})$, thus $\mathcal{F} \subset \mathrm{Im}(g_{A(\cdot)})$ i.e. $d_{\mathrm{P}} \leq \dim(\mathrm{Im}(g_{A(\cdot)}))$. Thus, $d_{\mathrm{P}} \leq \dim(\mathrm{Im}(g_{A(\cdot)})) \leq d_\xi$. Assumption 6.4 states that $d_\xi = d_{\mathrm{P}}$, s.t. $\dim(\mathrm{Im}(g_{A(\cdot)})) = d_{\mathrm{P}}$. As $\mathcal{F} \subset \mathrm{Im}(g_{A(\cdot)})$ and $\dim(\mathcal{F}) = \dim(\mathrm{Im}(g_{A(\cdot)}))$, $\mathcal{F} = \mathrm{Im}(g_{A(\cdot)})$ i.e. $g_{A(\cdot)}$ is surjective onto $\mathcal{F}$. As $\dim(\mathcal{F}) = d_\xi$, the dimension of the input space, $g_{A(\cdot)}$ is bijective.

By bijectivity of ψ, $\{\theta_{\mathrm{P},i}\}_{i=1}^{d_\mathrm{P}}$ forms a basis of $\mathbb{R}^{d_\mathrm{P}}$. $g_{A(\cdot)}$ and ψ map this basis to the same basis $\{f_i\}_{i=1}^{d_\mathrm{P}}$ of $\mathcal{F}$. As both mappings are bijective, this implies that $g_{A(\cdot)} = \psi(\cdot)$. This means that $\forall e \in \mathcal{E}, {g_A}^{-1}(f^e) = \psi^{-1}(f^e)$ i.e. system parameters p^e are recovered. □

Proposition 6.4 (Local identification under nonlinearity)**.** *For linearly parametrized systems, nonlinear w.r.t. inputs and nonlinear dynamics model g_θ with parameters output by a linear hypernetwork A, $\exists \alpha > 0$ s.t. system parameters are perfectly identified $\forall e \in \mathcal{E}$ where $\|\xi^e\| \leq \alpha$ if $\forall f \in \mathcal{B}$ with parameter θ_P, $g_{A(\alpha \frac{\theta_\mathrm{P}}{\|\theta_\mathrm{P}\|})} = f$.*

Proof. On environment $e \in \mathcal{E}$, g_{θ^e} is differentiable w.r.t. $\theta^e = A(\xi^e) = \theta^c + W\xi^e \in \mathbb{R}^{d_\theta}$. We perform a first order Taylor expansion of $g_{A(\cdot)}$ around $\mathbf{0}$. We note $\alpha > 0$, s.t. $\forall \xi^e \in \mathbb{R}^{d_\xi}$ that satisfy $\|\xi^e\| < \alpha$, we have $g_{\theta^e} = g_{\theta^c} + \nabla_\theta g_{\theta^c} W\xi^e$. $g_{A(\cdot)}$ is then linear in the neighborhood of 0 defined by α. $\forall i \in [\![1, d_\mathrm{P}]\!], \alpha \frac{\theta_{\mathrm{P},i}}{\|\theta_{\mathrm{P},i}\|}$ belongs to this neighborhood s.t. the proof of Proposition 6.3 applies to this neighborhood if $\forall i \in [\![1, d_\mathrm{P}]\!], g_{A(\alpha \frac{\theta_{\mathrm{P},i}}{\|\theta_{\mathrm{P},i}\|})} = f_i$, where $\mathcal{B} = \{f_i\}_{i=1}^{d_\mathrm{P}}$ is a basis of $\mathcal{F}$. □

We now show the validity of the unicity condition (Assumption 6.3) for two linearly parametrized systems.

Lemma C.1. *There is a unique set of parameters in $\mathbb{R}^4$ for a Lotka-Volterra (Lotka-Volterra) system.*

Proof. With $\psi : c := (\alpha, \beta, \delta, \gamma) \mapsto \left[\begin{pmatrix} x_t \\ y_t \end{pmatrix} \mapsto \begin{pmatrix} \alpha x_t - \beta x_t y_t \\ \delta x_t y_t - \gamma y_t \end{pmatrix} \right]$ a surjective linear mapping from $\mathbb{R}^4$ to $\mathcal{F}$ (all *Lotka-Volterra* systems are parametrized). Injectivity of ψ i.e. $\psi(c_1) = \psi(c_2) \iff c_1 = c_2$ will imply bijectivity i.e. unicity of parameters for a *Lotka-Volterra* system. As ψ is linear, injectivity is equivalent to $\psi(c) = 0 \iff c = 0$, shown below:

$$
\begin{aligned}
\psi(c) = 0 &\iff \forall \begin{pmatrix} x_t \\ y_t \end{pmatrix}, \begin{pmatrix} x_t(\alpha - \beta y_t) \\ (\delta x_t - \gamma) y_t \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \\
&\iff \forall \begin{pmatrix} x_t \\ y_t \end{pmatrix}, \begin{pmatrix} \alpha - \beta y_t \\ \delta x_t - \gamma \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \\
&\iff c = (\alpha, \beta, \delta, \gamma) = (0, 0, 0, 0)
\end{aligned}
$$

□

Lemma C.2. *There is a unique set of parameters in $\mathbb{R}^{d+1}$, where d is the grid size, for a*

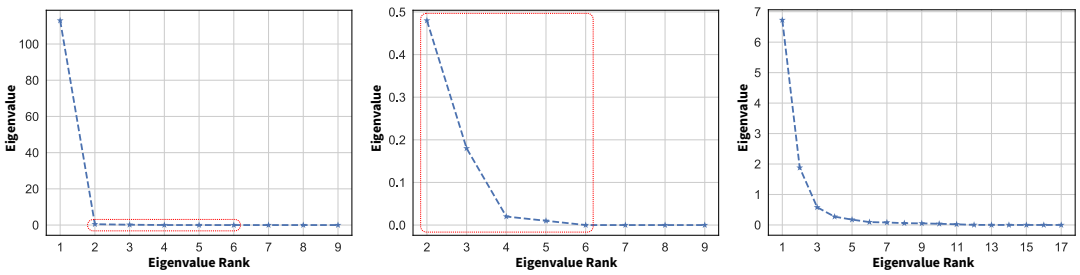
Navier-Stokes (Navier-Stokes) system.

Proof. With $\psi\colon c \coloneqq (\nu, f) \mapsto \Big[w \mapsto -v\nabla w + \nu\Delta w + h\Big]$, a surjective linear mapping from $\mathbb{R}^{d+1}$ to $\mathcal{F}$ (all *Navier-Stokes* systems are parametrized), bijectivity of ψ is induced by injectivity i.e. $\psi(c_1) = \psi(c_2) \iff c_1 = c_2$, shown below:

$$\begin{aligned}
&\psi(c_1) = \psi(c_2)\\
\iff &\forall w, -v\nabla w + \nu_1\Delta w + h_1 = -v\nabla w + \nu_2\Delta w + h_2\\
\iff &\forall w, (\nu_1 - \nu_2)\Delta w = -(h_1 - h_2)\\
\iff &(\nu_1, f_1) = (\nu_2, f_2) \iff c_1 = c_2
\end{aligned}$$

□

C.4 Low-Rank Assumption



(a) *Glycolitic-Oscillator*: k_1 and K_1 vary across $\mathcal{E}$. (b) *Sinusoidal*.

Figure C.2: Ranked singular values of the gradients across environments $\mathcal{E}_{\text{tr}}, \mathcal{G}_{\theta^c}$ for CoDA-ℓ_1.

When the systems are nonlinearly parametrized, we show empirically with Figure C.2 that the low-rank assumption is still reasonable for two different systems.

Glycolitic-Oscillator We first consider the Glycolitic oscillator system described in Appendix C.6.1, which is nonlinear w.r.t. K_1. We vary parameters k_1, K_1 in Eq. (C.13) across environments. We observe in Figure C.2a that there are three main gradient directions with SVD. The first is the most significant one while the second and third ones are orders of magnitude smaller.

Sinusoidal We consider a sinusoidal family of functions $S(N) = \{f : \mathbb{R} \to \mathbb{R} \mid f(x) = \sum_{i=1}^{N} \lambda_i \sin(\omega_i x + \phi_i)\}$. We sample 20 environments, each corresponding to different amplitudes $\lambda_i \sim \text{Unif}([0, 1])$, frequencies $\omega_i \sim \text{Unif}([0, 10])$ and phases $\phi_i \sim \text{Unif}([0, \pi])$. We depict in Figure C.2b the evaluation of the singular values at initialization. Figure C.2b shows that the number of directions to consider for convergence is small and that a single direction accounts for a significant amount of the variance in the gradients. This corroborates the low-rank assumption.

C.5 Locality Constraint

We derive the upper bounds to $\|\cdot\|$ for two variations.

$\|\cdot\| = \ell_2$: we apply triangle inequality to obtain $\mathcal{R}_W = \ell_2^2$

$$\|W\xi^e\|_2^2 \leq \|W\|_2^2 \cdot \|\xi^e\|_2^2$$

$\|\cdot\| = \ell_1$: we apply Cauchy-Schwartz inequality to obtain $\mathcal{R}_W(W) = \ell_{1,2}(W) = \sum_{i=1}^{d_\theta} \|W_{i,:}\|_2$

$$\|W\xi^e\|_1 = \sum_{i=1}^{d_\theta} |W_{i,:}\xi^e| \leq \|\xi^e\|_2 \sum_{i=1}^{d_\theta} \|W_{i,:}\|_2$$

Eq. (6.12) minimizes the log of the above upper-bounds.

C.6 Experimental Settings

We present in Appendix C.6.1 the equations and the data generation specificities for all considered dynamical systems.

C.6.1 Dynamical Systems

Lotka-Volterra (Lotka, 1925) The system describes the interaction between a prey-predator pair in an ecosystem, formalized into the following ODE:

$$\begin{aligned} \frac{\mathrm{d}x}{\mathrm{d}t} &= \alpha x - \beta x y \\ \frac{\mathrm{d}y}{\mathrm{d}t} &= \delta x y - \gamma y \end{aligned} \tag{C.12}$$

where x, y are respectively the quantity of the prey and the predator, $\alpha, \beta, \delta, \gamma$ define how two species interact.

We generate trajectories on a temporal grid with $\delta t = 0.5$ and temporal horizon $T = 10$. We sample on each training environment $N_{\text{tr}} = 4$ initial conditions for training from a uniform distribution $\rho_0(\mathcal{U}) = \text{Unif}([1,3]^2)$. We sample for evaluation 32 initial conditions from $\rho_0(\mathcal{U})$. Across environments, $\alpha = 0.5, \gamma = 0.5$. For training, we consider $\#\mathcal{E}_{\text{tr}} = 9$ environments with parameters $\beta, \delta \in \{0.5, 0.75, 1.0\}^2$. For adaptation, we consider $\#\mathcal{E}_{\text{ad}} = 4$ environments with parameters $\beta, \delta \in \{0.625, 1.125\}^2$.

Glycolytic-Oscillator (Daniels and Nemenman, 2015) *Glycolitic-Oscillator* describes yeast glycolysis dynamics with the ODE:

$$\begin{aligned}
\frac{dS_1}{dt} &= J_0 - \frac{k_1 S_1 S_6}{1 + (1/K_1^q) S_6^q} \\
\frac{dS_2}{dt} &= 2\frac{k_1 S_1 S_6}{1 + (1/K_1^q) S_6^q} - k_2 S_2 (N - S_5) - k_6 S_2 S_5 \\
\frac{dS_3}{dt} &= k_2 S_2 (N - S_5) - k_3 S_3 (A - S_6) \\
\frac{dS_4}{dt} &= k_3 S_3 (A - S_6) - k_4 S_4 S_5 - \kappa (S_4 - S_7) \\
\frac{dS_5}{dt} &= k_2 S_2 (N - S_5) - k_4 S_4 S_5 - k_6 S_2 S_5 \\
\frac{dS_6}{dt} &= -2\frac{k_1 S_1 S_6}{1 + (1/K_1^q) S_6^q} + 2k_3 S_3 (A - S_6) - k_5 S_6 \\
\frac{dS_7}{dt} &= \psi \kappa (S_4 - S_7) - k S_7
\end{aligned} \tag{C.13}$$

where $S_1, S_2, S_3, S_4, S_5, S_6, S_7$ represent the concentrations of 7 biochemical species. We generate trajectories on a temporal grid with $\delta t = 0.05$ and temporal horizon $T = 1$. We sample on each training environment $N_{\text{tr}} = 32$ initial conditions for training from a uniform distribution $p_0(\mathcal{U})$ defined in Table 2 in (Daniels and Nemenman, 2015). Across environments, $J_0 = 2.5, k_2 = 6, k_3 = 16, k_4 = 100, k_5 = 1.28, k_6 = 12, q = 4, N = 1, A = 4, \kappa = 13, \psi = 0.1, k = 1.8$. For training, we consider $\#\mathcal{E}_{\text{tr}} = 9$ environments with parameters $k_1 \in \{100, 90, 80\}, K_1 \in \{1, 0.75, 0.5\}$. For adaptation, we consider $\#\mathcal{E}_{\text{ad}} = 4$ environments with parameters $k_1 \in \{85, 95\}, K_1 \in \{0.625, 0.875\}$.

Gray-Scott (Pearson, 1993) The PDE descibes a reaction-diffusion system with complex spatiotemporal patterns through the following 2D PDE:

$$\begin{aligned}
\frac{\partial v}{\partial t} &= D_v \Delta v - v w^2 + F(1 - v) \\
\frac{\partial w}{\partial t} &= D_w \Delta w + v w^2 - (F_r + k_r) w
\end{aligned} \tag{C.14}$$

where v, w represent the concentrations of two chemical components in the spatial domain Ω with periodic boundary conditions. D_v, D_w denote the diffusion coefficients respectively for v, w and F_r, k_r are the reaction parameters.

We generate trajectories on a temporal grid with $\delta t = 40$ and temporal horizon $T = 400$. Ω is a 2D space discretized on a regular grid of dimension 32×32 with spatial resolution of $\delta x_i = 2$. We define initial conditions $(v_0, w_0) \sim \rho_0(\mathcal{U})$ by uniformly sampling three two-by-two squares in Ω. These squares trigger the reactions. $(v_0, w_0) = (1 - \epsilon, \epsilon)$ with $\epsilon = 0.05$ inside the squares and $(v_0, w_0) = (0, 1)$ outside the squares. We sample on each training environment $N_{\text{tr}} = 1$ initial conditions for training. Across environments, $D_v = 0.2097, D_w = 0.105$. For training, we consider $\#\mathcal{E}_{\text{tr}} = 4$ environments with parameters $F_r \in \{0.30, 0.39\}, k_r \in \{0.058, 0.062\}$. For adaptation, we consider $\#\mathcal{E}_{\text{ad}} = 4$ environments with parameters $F_r \in \{0.33, 0.36\}, k_r \in \{0.59, 0.61\}$.

Navier-Stokes (Stokes, 1851) *Navier-Stokes* describes the dynamics of incompressible flows with the 2D PDE:

$$\begin{aligned} &\frac{\partial w}{\partial t} = -v\nabla w + \nu\Delta w + f^{\text{fc.}} \text{ where } w = \nabla \times v \\ &\nabla v = 0 \end{aligned} \tag{C.15}$$

where v is the velocity field, $w = \nabla \times v$ is the vorticity. Both v, w lie in a spatial domain Ω with periodic boundary conditions, ν is the viscosity and f is the constant forcing term in the domain Ω. We generate trajectories on a temporal grid with $\delta t = 1$ and temporal horizon $T = 10$. Ω is a 2D space of dimension 32×32 with spatial resolution of $\delta x = 1$. We sample on each training environment $N_{\text{tr}} = 16$ initial conditions for training from $\rho_0(\mathcal{U})$ as in Li et al. (2021b). Across environments, $f^{\text{fc.}}(x_1, x_2) = 0.1(\sin(2\pi(x_1 + x_2)) + \cos(2\pi(x_1 + x_2)))$. For training, we consider $\#\mathcal{E}_{\text{tr}} = 5$ environments with parameters $\nu \in \{8\times10^{-4}, 9\times10^{-4}, 1\times10^{-3}, 1.1\times10^{-3}, 1.2\times10^{-3}\}$. For adaptation, we consider $\#\mathcal{E}_{\text{ad}} = 4$ environments with parameters $\nu \in \{8.5\times10^{-4}, 9.5\times10^{-4}, 1.05\times10^{-3}, 1.15\times10^{-3}\}$.

C.6.2 Implementation and Hyperparameters

Architecture We implement the dynamics model g_θ with the following architectures:

- *Lotka-Volterra, Glycolitic-Oscillator*: 4-layer MLPs with hidden layers of width 64.
- *Gray-Scott*: 4-layer ConvNets with 64-channel hidden layers, and 3×3 convolution kernels
- *Navier-Stokes*: FNOs with 4 spectral convolution layers. 12 frequency modes and hidden layers with width 10.

We apply Swish activation (Ramachandran et al., 2017). The hypernet A is a single affine layer NN.

Optimizer We use the Adam optimizer (Kingma and Ba, 2015) with learning rate 10^{-3} and $(\beta_1, \beta_2) = (0.9, 0.999)$. We apply early stopping. All experiments are performed with a single NVIDIA Titan Xp GPU on an internal cluster. We distribute training by batching together predictions across trajectories to reduce running time. States across batch elements are concatenated.

Hyperparameters We define the hyperparameters for the following models: (a) CoDA: • *Lotka-Volterra*: $\lambda_\xi = 10^{-4}$, $\lambda_{\ell_1} = 10^{-6}$, $\lambda_{\ell_2} = 10^{-5}$ • *Glycolitic-Oscillator*: $\lambda_\xi = 10^{-3}$, $\lambda_{\ell_1} = 10^{-7}$, $\lambda_{\ell_2} = 10^{-7}$ • *Gray-Scott*: $\lambda_\xi = 10^{-2}$, $\lambda_{\ell_1} = 10^{-5}$, $\lambda_{\ell_2} = 10^{-5}$ • *Navier-Stokes*: $\lambda_\xi = 110^{-3}$, $\lambda_{\ell_1} = 2{\times}10^{-3}$, $\lambda_{\ell_2} = 2{\times}10^{-3}$ (b) LEADS: we use the same parameters as Yin et al. (2021a). (c) GBML: the outer-loop learning rate is 10^{-3}, we apply a 1-step inner-loop update for training and adaptation to maintain low running times. The inner-loop learning rate for each system is: • *Lotka-Volterra*: 10^{-1} • *Glycolitic-Oscillator*: 10^{-2} • *Gray-Scott*: 10^{-3} • *Navier-Stokes*: 10^{-3}. These values are also used to initialize the per-parameter inner-loop learning rate in Meta-SGD.

C.7 Trajectory Prediction Visualization

We visualize in Figures C.3 and C.4 the prediction MSE by MAML, LEADS, CAVIA-Concat and CoDA-ℓ_1 along ground truth trajectories on the PDE systems *Navier-Stokes* and *Gray-Scott*. We consider a new test trajectory on an *Adaptation* environment $e \in \mathcal{E}_{\text{ad}}$ with parameters defined in the caption.

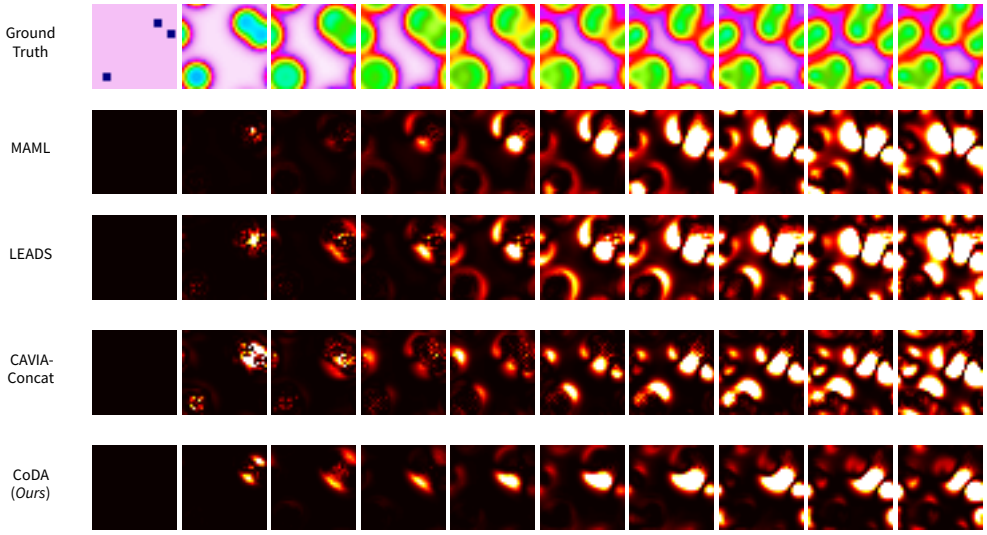


Figure C.3: Adaptation to new *Gray-Scott* system - $(F_r, k_r, D_v, D_w) = (0.033, 0.061, 0.2097, 0.105)$. Ground-truth trajectory and prediction MSE per frame for MAML, LEADS, CAVIA-Concat and CoDA.

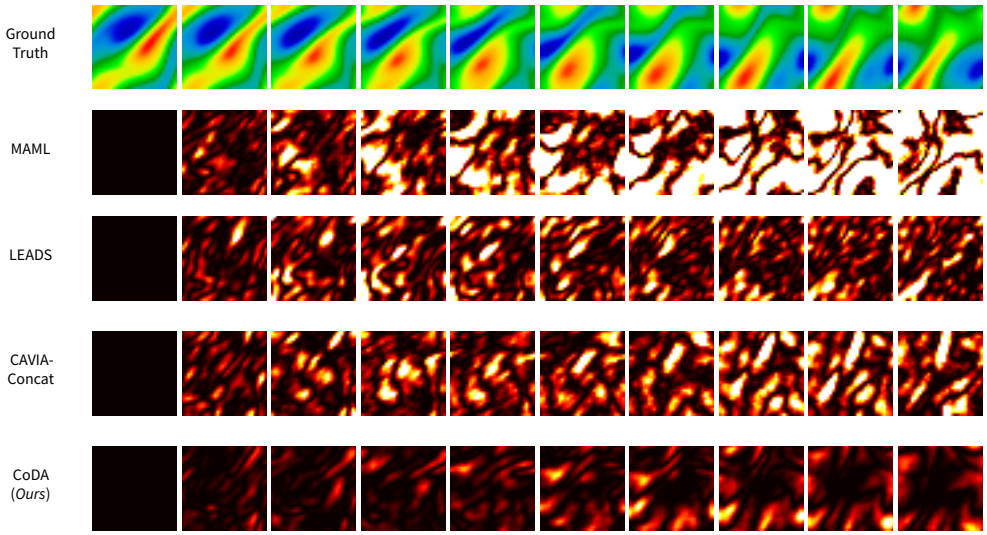


Figure C.4: Adaptation to new *Navier-Stokes* system - $\nu = 1.15\times10^{-3}$. Ground-truth trajectory and prediction MSE per frame for MAML, LEADS, CAVIA-Concat, and CoDA.

Appendix D

Appendix of Chapter 7

D.1 Full Results

We provide in Table D.1 a more detailed version of Table 7.3 for the space-time extrapolation problem where we report the performance *In-s* (on the observation grid) and *Out-s* (outside). We add $s = 50\%$.

Then, we report in Table D.2, a more detailed version of Table 7.4a, which includes the results of $\mathcal{X}_{\text{ts}} = \mathcal{X}_{\text{tr}}$. This corresponds to our generalization across grids problem.

Table D.1: Space and time generalization. The train and test observation grids are equal; they are subsampled with a ratio s from a uniform 64×64 grid fixed here to be the inference grid $\mathcal{X}'$. We report MSE (↓) on $\mathcal{X}'$ (on the observation grid *In-s*, outside *Out-s* or on both *Full*) and the inference time interval $\mathcal{T}'$, divided within training horizon (*In-t*, $\mathcal{T}$) and beyond (*Out-t*, outside $\mathcal{T}$) across subsampling ratios $s \in \{5\%, 25\%, 50\%, 100\%\}$. Best in **bold** and second best underlined.

	Model	Navier-Stokes				Wave			
		Train		Test		Train		Test	
		In-t	Out-t	In-t	Out-t	In-t	Out-t	In-t	Out-t
		$s = 5\%$ subsampling							
In-s	I-MP-PDE	**3.525E−5**	1.295E−3	4.554E−4	1.414E−3	**1.824E−6**	8.672E−5	1.113E−5	1.987E−4
	DeepONet	4.778E−4	4.517E−3	1.060E−2	1.059E−2	2.546E−4	8.831E−3	1.501E−2	3.196E−2
	SIREN	5.966E−3	1.769E−1	4.082E−2	2.150E−1	1.690E−3	1.707E−2	2.951E−2	6.955E−2
	DINo	1.016E−4	**6.945E−4**	**3.623E−4**	**8.306E−4**	2.250E−6	**5.283E−6**	**7.530E−6**	**2.146E−5**
Out-s	I-MP-PDE	8.550E−3	8.515E−3	8.306E−3	8.571E−3	7.412E−4	7.414E−4	1.195E−3	1.163E−3
	DeepONet	3.475E−3	7.515E−3	1.361E−2	1.426E−2	8.624E−4	9.318E−3	1.702E−2	3.259E−2
	SIREN	8.882E−3	1.767E−1	4.314E−2	2.124E−1	2.791E−3	1.823E−2	3.359E−2	6.965E−2
	DINo	**1.076E−3**	**1.704E−3**	**1.375E−3**	**1.863E−3**	**4.285E−5**	**4.304E−5**	**6.703E−5**	**7.659E−5**
Full	I-MP-PDE	8.154E−3	8.166E−3	7.926E−3	8.225E−3	7.055E−4	7.097E−4	1.138E−3	1.116E−3
	DeepONet	3.330E−3	7.370E−3	1.346E−2	1.408E−2	8.331E−4	9.295E−3	1.692E−2	3.256E−2
	SIREN	8.741E−3	1.767E−1	4.303E−2	2.126E−1	2.738E−3	1.818E−2	3.339E−2	6.964E−2
	DINo	**1.029E−3**	**1.655E−3**	**1.326E−3**	**1.813E−3**	**4.088E−5**	**4.121E−5**	**6.415E−5**	**7.392E−5**
		$s = 25\%$ subsampling							

(*Continued on next page* →)

Table D.1: *(Continued)*

	Model	Navier-Stokes				Wave			
		Train		Test		Train		Test	
		In-t	Out-t	In-t	Out-t	In-t	Out-t	In-t	Out-t
In-s	I-MP-PDE	1.447E−4	5.677E−4	**1.763E−4**	6.147E−4	**6.754E−7**	8.251E−5	**9.253E−7**	1.227E−4
	DeepONet	7.500E−4	5.779E−3	9.227E−3	1.300E−2	5.196E−4	1.058E−2	1.743E−2	3.246E−2
	SIREN	4.786E−3	2.178E−1	2.461E−1	3.884E−1	8.478E−4	1.282E−2	1.733E−2	5.104E−2
	DINo	**8.295E−5**	**4.273E−4**	2.444E−4	**5.735E−4**	3.194E−6	**3.747E−6**	8.907E−6	**1.029E−5**
Out-s	I-MP-PDE	3.678E−4	7.748E−4	4.026E−4	8.143E−4	4.330E−5	1.200E−4	6.764E−5	1.648E−4
	DeepONet	9.503E−4	5.987E−3	9.423E−3	1.337E−2	5.891E−4	1.062E−2	1.762E−2	3.213E−2
	SIREN	5.305E−3	2.173E−1	2.428E−1	3.853E−1	9.159E−4	1.295E−2	1.798E−2	5.156E−2
	DINo	**1.081E−4**	**4.578E−4**	**2.711E−4**	**6.021E−4**	**4.192E−6**	**4.657E−6**	**1.153E−5**	**1.220E−5**
Full	I-MP-PDE	3.135E−4	7.245E−4	3.476E−4	7.658E−4	3.293E−5	1.108E−4	5.142E−5	1.545E−4
	DeepONet	9.016E−4	5.936E−3	9.376E−3	1.328E−2	5.722E−4	1.061E−2	1.757E−2	3.221E−2
	SIREN	5.180E−3	2.175E−1	2.436E−1	3.861E−1	8.995E−4	1.292E−2	1.783E−2	5.143E−2
	DINo	**1.020E−4**	**4.504E−4**	**2.646E−4**	**5.951E−4**	**3.949E−6**	**4.436E−6**	**1.089E−5**	**1.174E−5**
		$s = 50\%$ subsampling							
In-s	I-MP-PDE	1.153E−4	5.016E−4	**1.594E−4**	6.043E−4	**2.200E−7**	3.179E−5	**8.843E−7**	5.854E−5
	DeepONet	6.214E−4	4.277E−3	5.699E−3	1.082E−2	7.581E−4	1.187E−2	1.649E−2	3.378E−2
	SIREN	4.911E−3	6.815E−1	1.607E−1	6.889E−1	5.134E−4	1.481E−2	3.086E−2	8.196E−2
	DINo	**8.151E−5**	**2.920E−4**	2.004E−4	**4.283E−4**	3.277E−6	**3.659E−6**	8.978E−6	**9.572E−6**
Out-s	I-MP-PDE	1.186E−4	5.010E−4	**1.626E−4**	6.132E−4	**9.638E−7**	3.153E−5	**2.367E−6**	5.574E−5
	DeepONet	6.851E−4	4.343E−3	5.740E−3	1.099E−2	7.842E−4	1.185E−2	1.679E−2	3.391E−2
	SIREN	5.067E−3	6.867E−1	1.599E−1	6.845E−1	5.354E−4	1.492E−2	3.113E−2	8.333E−2

(*Continued on next page* →)

Table D.1: *(Continued)*

	Model	Navier-Stokes				Wave			
		Train		Test		Train		Test	
		In-t	Out-t	In-t	Out-t	In-t	Out-t	In-t	Out-t
	DINo	**9.175E−5**	**3.041E−4**	2.116E−4	**4.409E−4**	3.277E−6	**3.659E−6**	8.978E−6	**9.572E−6**
Full	I-MP-PDE	1.170E−4	5.013E−4	**1.611E−4**	6.088E−4	**6.021E−7**	3.166E−5	**1.646E−6**	5.710E−5
	DeepONet	6.541E−4	4.311E−3	5.720E−3	1.091E−2	7.715E−4	1.186E−2	1.665E−2	3.385E−2
	SIREN	4.995E−3	6.841E−1	1.603E−1	6.867E−1	5.246E−4	1.486E−2	3.100E−2	8.265E−2
	DINo	**8.677E−5**	**2.982E−4**	2.062E−4	**4.348E−4**	3.380E−6	**3.751E−6**	9.251E−6	**9.710E−6**
		s = 100% subsampling							
Full	CNODE	2.319E−2	9.652E−2	2.305E−2	1.143E−1	2.337E−5	5.280E−4	3.057E−5	7.288E−4
	MP-PDE	1.140E−4	5.500E−4	**1.785E−4**	5.856E−4	**1.718E−7**	1.993E−5	**9.256E−7**	4.261E−5
	MNO	**3.190E−5**	8.678E−4	2.763E−4	8.946E−4	9.381E−6	4.890E−3	1.993E−4	6.128E−3
	DeepONet	1.375E−3	6.573E−3	9.704E−3	1.244E−2	6.431E−4	1.293E−2	1.847E−2	3.317E−2
	SIREN	1.066E−3	4.336E−1	3.874E−1	1.037E0	3.674E−4	9.956E−3	3.013E−2	7.842E−2
	MFN	1.651E−3	1.037E0	2.106E−1	1.059E0	1.408E−4	1.763E−1	4.735E−3	2.274E−1
	DINo (no sep.)	3.235E−4	1.593E−3	7.850E−4	1.889E−3	2.641E−6	4.081E−5	5.977E−5	2.979E−4
	DINo	8.339E−5	**3.115E−4**	2.092E−4	**4.311E−4**	3.309E−6	**3.506E−6**	9.495E−6	**9.946E−6**

Table D.2: Generalization across grids. $\mathcal{X}_{\text{tr}}, \mathcal{X}_{\text{ts}}$ are subsampled with different ratios $s_{\text{tr}} \neq s_{\text{ts}} \in \{5, 50, 100\}\%$ from the same uniform 64×64 grid. We report *test* MSE within $\mathcal{X}_{ts}$ (*In-s*). **Best** in bold.

Subsampling	Test →	$\mathcal{X}_{\text{ts}} = \mathcal{X}_{\text{tr}}$		$\mathcal{X}_{\text{ts}} \neq \mathcal{X}_{\text{tr}}$					
		$s_{\text{ts}} = s_{\text{tr}}$		$s_{\text{ts}} = 5\%$		$s_{\text{ts}} = 50\%$		$s_{\text{ts}} = 100\%$	
Train ↓		In-t	Out-t	In-t	Out-t	In-t	Out-t	In-t	Out-t
$s_{\text{tr}} = 5\%$	MP-PDE	**1.967E−4**	**6.631E−4**	1.330E−1	3.852E−1	1.859E−1	6.680E−1	2.105E−1	7.120E−1
	DINo	3.623E−4	8.306E−4	**1.494E−3**	**2.291E−3**	**1.257E−3**	**1.883E−3**	**1.287E−3**	**1.947E−3**
$s_{\text{tr}} = 50\%$	MP-PDE	**1.346E−4**	5.110E−4	4.494E−2	9.403E−2	4.793E−3	1.997E−2	6.330E−3	3.712E−2
	DINo	2.004E−4	**4.283E−4**	**2.470E−4**	**4.697E−4**	**2.073E−4**	**4.284E−4**	**2.058E−4**	**4.361E−4**
$s_{\text{tr}} = 100\%$	MP-PDE	**1.785E−4**	5.856E−4	1.358E−1	3.355E−1	1.182E−2	2.664E−2	**1.785E−4**	5.856E−4
	DINo	2.092E−4	**4.311E−4**	**2.495E−4**	**4.805E−4**	**2.109E−4**	**4.325E−4**	2.092E−4	**4.311E−4**

D.2 Prediction

We display the test prediction of DINo (Figure D.1) and I-MP-PDE (Figure D.2) for various subsampling levels when $\mathcal{X} = \mathcal{X}_{\text{tr}} = \mathcal{X}_{\text{ts}}$. Predictions are performed on a 64×64 uniform grid which defines the observation grid $\mathcal{X}$ via different subsampling rates. Yellow points correspond to the observation grid $\mathcal{X}$ (*In-s*) while purple points indicate off-grid points (*Out-s*). The prediction for I-MP-PDE at $t = 0$ is the interpolated initial condition.

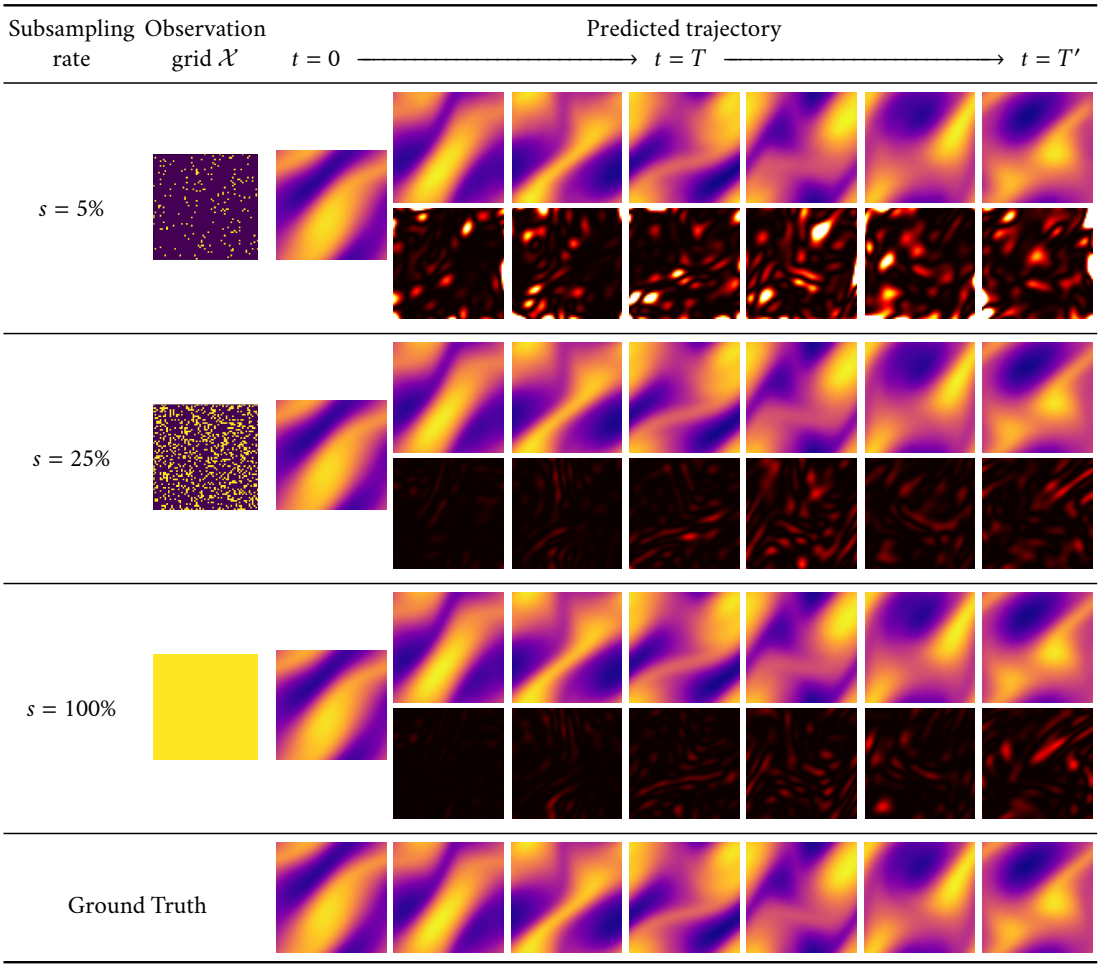


Figure D.1: Prediction MSE per frame for **DINo** on *Navier-Stokes* with its corresponding observed grid $\mathcal{X}$. For each model, the first row contains the predicted trajectory from 0 to T', the second row is the corresponding error maps w.r.t. the reference data (the darker the pixel, the lower the error).

D.3 Detailed Description of Datasets

We choose $\mathcal{T}$ (resp. $\mathcal{T}'$) on a regular grid in $[0, T]$ (resp. $[0, T']$) with a given temporal resolution and fix $T' = 2T$. The range of T depends on the nature of the dataset; however, we always consider 10 consecutive frames for *In-t* and 10 more for *Out-t*. We provide further details on the choice of these parameters and other experimental parameters, such as the number of observed trajectories.

2D Wave equation (*Wave*)**.** It is a second-order PDE:

$$\frac{\partial^2 u}{\partial t^2} = c^2 \Delta u, \tag{D.1}$$

where u is a function of the displacement at each point in space w.r.t. the rest position, $c \in \mathbb{R}_+^*$ is the speed of wave travel. We transform the equation to a first-order form, considering the input $v_t = \left(u_t, \frac{\partial u_t}{\partial t}\right)$, so that the dimension of $v_t(x)$ at each point $x \in \Omega$ is $n = 2$.

We generate our dataset for speed $c = 2$ with periodic boundary conditions. The domain is $\Omega = [-1, 1]^2$. For initial conditions $v_0 = \left(u_0, \frac{\partial u_t}{\partial t}\Big|_{t=0}\right)$, the initial displacement u_0 is a Gaussian function:

$$u_0(x; a, b, r) = a \exp\left(-\frac{(x-b)^2}{2r^2}\right), \tag{D.2}$$

where the height of the peak displacement is $a \sim \mathcal{U}(2, 4)$, the location of the peak displacement is $(b_1, b_2) \sim \mathcal{U}(-1, 1)$, and the standard deviation is $r \sim \mathcal{U}(0.25, 0.3)$. The initial time derivative is $\frac{\partial u_t}{\partial t}\Big|_{t=0} = 0$. Each snapshot is generated on a uniform grid of 64×64. Each sequence is generated with fixed interval $\delta t = 0.25$. We set the training horizon $T = 2.25$ and the inference horizon $T = 4.75$. We generated 512 training trajectories and 32 test trajectories.

2D Navier Stokes (*Navier-Stokes*, Stokes, 1851)**.** This dataset corresponds to incompressible fluid dynamics described by:

$$\frac{\partial w}{\partial t} = -u \nabla w + \nu \Delta w + f^{\text{fc.}}, \quad w = \nabla \times u, \quad \nabla u = 0, \tag{D.3}$$

where u is the velocity field and w the vorticity. u, w lie on a spatial domain with periodic boundary conditions, ν is the viscosity and f is a constant forcing term. The input v_t is w_t ($n = 1$). ν is the viscosity and f is the constant forcing term in the domain Ω.

The spatial domain is $\Omega = [-1, 1]^2$, the viscosity is $\nu = 1\times10^{-3}$, the forcing term is set as:

$$\forall x \in \Omega, f^{\text{fc.}}(x_1, x_2) = 0.1(\sin(2\pi(x_1 + x_2)) + \cos(2\pi(x_1 + x_2))). \tag{D.4}$$

The full spatial grid is of dimension 64×64 or 256×256 according to experiments in Section 7.4. We sample initial conditions as in Li et al. (2021b) to create different trajectories. The first 20 steps of the trajectories are cut off as they are too noisy and not informative in terms of dynamics. Trajectories are collected with $\delta t = 1$. We set the training horizon $T = 19$ and the inference horizon $T' = 39$. We generated 512 training trajectories and 32 test trajectories.

3D spherical shallow water (*Shallow-Water*, Galewsky et al., 2004)**.** The following problem is originally presented for testing numerical models of global shallow-water equations. The shallow water equations are written as:

$$\begin{aligned} \frac{\mathrm{d}u}{\mathrm{d}t} &= -fk \times u - g\nabla h + \nu\Delta u, \\ \frac{\mathrm{d}h}{\mathrm{d}t} &= -h\nabla \cdot u + \nu\Delta h. \end{aligned} \tag{D.5}$$

where $\frac{\mathrm{d}}{\mathrm{d}t}$ is the material derivative, k is the unit vector orthogonal to the spherical surface, u is the velocity field tangent to the surface of the sphere, which can be transformed into the vorticity $w = \nabla \times u$, h is the thickness of the sphere. Note that the data we observe at each time t is $v_t = (w_t, h_t)$. f, g, ν, Ω are parameters of the Earth (cf. Galewsky et al., 2004 for details).

The initial conditions are slightly modified from Galewsky et al. (2004), detailed below, to create symmetric phenomena in the northern and southern hemisphere. The initial zonal velocity u_0 contains two non-null symmetric bands in both hemispheres, which are parallel to the circles of latitude. At each latitude and longitude $\phi, \theta \in [-\pi/2, \pi/2] \times [-\pi, \pi]$:

$$u_0(\phi, \theta) = \begin{cases} \left(\frac{u_{\max}}{e_{\mathrm{n}}} \exp\left(\frac{1}{(\phi - \phi_0)(\phi - \phi_1)}\right), 0\right) & \text{if } \phi \in (\phi_0, \phi_1), \\ \left(\frac{u_{\max}}{e_{\mathrm{n}}} \exp\left(\frac{1}{(\phi + \phi_0)(\phi + \phi_1)}\right), 0\right) & \text{if } \phi \in (-\phi_1, -\phi_0), \\ (0, 0) & \text{otherwise.} \end{cases} \tag{D.6}$$

where $u_{\max}$ is the maximum velocity, $\phi_0 = \pi/7$, $\phi_1 = \pi/2 - \phi_0$, and $e_{\mathrm{n}} = \exp(-4/(\phi_1 - \phi_0)^2)$. The water height h_0 is initialized by solving a boundary value condition problem as in Galewsky

et al. (2004). It is then perturbed by adding the following h'_0 to h_0:

$$h'_0(\phi,\theta) = \hat{h}\cos(\phi)\exp\left(-\left(\frac{\theta}{\alpha}\right)^2\right)\left[\exp\left(-\left(\frac{\phi_2-\phi}{\beta}\right)^2\right)+\exp\left(-\left(\frac{\phi_2+\phi}{\beta}\right)^2\right)\right]. \tag{D.7}$$

where $\phi_2 = \pi/4, \hat{h} = 120\,\text{m}, \alpha = 1/3, \beta = 1/15$ are constants defined in Galewsky et al. (2004).

We simulate this phenomenon with Dedalus (Burns et al., 2020) on a latitude-longitude (lat-lon) grid. The size of the grid is 128 (lat)×256 (lon). We take different initial conditions by sampling $u_{\max} \sim \mathcal{U}(60, 80)$ to generate long trajectories. These long trajectories are then sliced into shorter ones. For simulation, we take one snapshot per hour (of internal simulation time), i.e. $\delta t = 1\,\text{h}$. We stop the simulation at the 320^{th} hour. To construct a dataset rich in dynamical phenomena, we take the snapshots within the last 160 h in a long trajectory and slice them into 8 shorter trajectories. Also, note that the data is scaled into a reasonable range: the height h is scaled by a factor of $3{\times}10^3$, and the vorticity w by a factor 2. In each short trajectory, $T = 9\,\text{h}$ and $T' = 19\,\text{h}$. In total, we generated 16 long trajectories (i.e. 128 short trajectories) for the train, and 2 for the test (i.e. 16 short trajectories).

D.3.1 Algorithm

We detail the algorithm of DINo for training and test via pseudo-code in Algorithm 3. Training consists in solving Eq. (7.6) w.r.t. $\psi, \alpha_{\mathcal{T}}, \phi$. Inference consists in only finding α_0.

Algorithme 3 : DINo pseudo-code

Training

Input : $\mathcal{D} = \{u_{\mathcal{T}}\}, \{\alpha^u_{\mathcal{T}}\}_{u\in\mathcal{D}} \leftarrow \{0\}, \phi \leftarrow \phi_0, \psi \leftarrow \psi_0$;
while *not converged* **do**
  for $u \in \mathcal{D}$ **do** $\alpha^u_{\mathcal{T}} \leftarrow \alpha^u_{\mathcal{T}} - \eta_\alpha \nabla_{\alpha^u_{\mathcal{T}}} \ell_{\text{dec}}(D_\phi, \alpha^u_{\mathcal{T}})$; `/* Modulation */`
  $\phi \leftarrow \phi - \eta_\phi \nabla_\phi \left(\sum_{u\in\mathcal{D}} \ell_{\text{dec}}(D_\phi, \alpha^u_{\mathcal{T}})\right)$; `/* Hypernetwork */`
  $\psi \leftarrow \psi - \eta_\psi \nabla_\psi \left(\sum_{u\in\mathcal{D}} \ell_{\text{dyn}}(f_\psi, \alpha^u_{\mathcal{T}})\right)$; `/* Dynamics */`

Test

Input : $\mathcal{D}'_0 = \{u_0\}, \{\alpha^u_0\}_{u\in\mathcal{D}'} \leftarrow \{0\}, \phi^\star, \psi^\star, \mathcal{T}' \neq \mathcal{T}$;
while *not converged* **do**
  for $u \in \mathcal{D}'$ **do** $\alpha^u_0 \leftarrow \alpha^u_0 - \eta \nabla_{\alpha^u_0} \ell_{\text{dec}}(D^\star_\phi, \alpha^u_0)$; `/* Modulation */`
for $u \in \mathcal{D}', t \in \mathcal{T}'$ **do** $\alpha^u_t \leftarrow \alpha^u_0 + \int_0^t f_{\psi^\star}(\alpha^u_\tau)\mathrm{d}\tau$; `/* Unroll dynamics */`

D.3.2 Convergence

Convergence analysis. In practice, we observe no training instability induced by the two-stage learning process of Eq. (7.6) and Algorithm 3: the objectives are non-conflicting. To assess this, we track the evolution of the auto-decoding loss ℓ_{dec} and the dynamics loss ℓ_{dyn} throughout training on *Navier-Stokes* ($s = 100\%$) in Figure D.4. We observe that both losses smoothly converge until the end of training.

D.3.3 Time Efficiency

Our auto-decoding strategy coupled with a latent neural ODE makes DINo computationally efficient compared to our best competitor MP-PDE.

Inferring α_0 via auto-decoding. Given a decoder and an observation frame v_0, finding α_0 corresponds to solving an inverse problem, cf. Eq. (7.3). At inference, we use 300 steps to infer α_0; using less steps is possible but results in slight under-fitting. This represents 2.76 s for 64 trajectories on a single Tesla V100 Nvidia GPU. Note that, as we unroll dynamics in the latent space, there is no need to relearn α_t when $t > 0$. Moreover, this differs from training, where α_t is continuously optimized for all $t \in [0, T]$ within the train horizon, alternatively with our INR decoder. Overall, we trained MP-PDE and DINo for approximately 7 days such that there is no major additional temporal training cost for DINo.

Latent neural ODE. Unrolling the dynamics with a neural ODE is efficient (0.35 s for 19 time predictions for 64 trajectories on a single Tesla V100 Nvidia GPU). Indeed, the latent space is small (at most 100 dimension) and the dynamics models uses a simple four-layer MLP for f_ψ. With the same latent dynamics model, using an RK4 numerical scheme only incurs four additional function evaluations over a discretized alternative, e.g., standard ResNet. This incurs a minor computational cost but enables DINo to operate at different temporal resolutions, unlike, e.g., MP-PDE.

In comparison, the official code of MP-PDE takes 312 s for inference on the same hardware for the same number of trajectories (vs 3 s for DINo). MP-PDE requires building an adjacency matrix and incurs for this reason a high memory cost, especially as the number of nodes increases. Interpolation also significantly increases inference time. This is not the case for DINo, which is faster.

D.3.4 Additional Implementation details

We use PyTorch (Paszke et al., 2019) to implement DINo and our baselines. Hyperparameters are further defined in Appendix D.3.5. The dynamics model f_ψ is a multilayer

perceptron. Its input and output size are same as the size of latent space d_α. All hidden layers share the same size. DINo's parameters are initialized with the default initialization in PyTorch, defining ϕ_0, ψ_0, ω in Algorithm 3. We recall that ω is fixed throughout training to reduce the number of optimized parameters without loss of performance. As in related work (Sitzmann et al., 2019; Fathony et al., 2021), the frequency parameters ω are scaled by a factor, ω_s, considered as a hyperparameter. For dynamics learning, we use an RK4 integrator via `torchdiffeq` (Chen et al., 2018) and apply exponential Scheduled Sampling (Bengio et al., 2015) to stabilize training. In practice, modulations α_t are learned channel-wise such that $I_\theta \colon \Omega \to \mathbb{R}^{d_c}$ has separate parameters per output dimension to make predictions less correlated across channels. We optimize all parameters ϕ, α, ψ using Adam (Kingma and Ba, 2015) with decay parameters $(\beta_1, \beta_2) = (0.9, 0.999)$.

D.3.5 Hyperparameters

We list the hyperparameters of DINo for each dataset in Table D.3. In practice, we observe it is beneficial to decay the learning rates η_ϕ, η_α when the loss reaches a plateau.

D.3.6 Baselines Implementation

We detail in the following the hyperparameters and architectures used in our experiments for the considered baselines, which we reimplemented for our paper.

- **CNODE** is implemented with four 2D convolutional layers with 64 hidden features, ReLU activations, 3×3 kernel and zero padding. Learning rate is fixed to 10^{-3}. We use an adjoint method for integration like (Chen et al., 2018).
- **MNO.** We use the FNO architecture in Li et al. (2021b) with three FNO blocks, GeLU activations, 12 modes and a width of 32. Learning rate is fixed to 10^{-3}.
- **DeepONet.** We consider an autoregressive formulation of DeepONet. We choose a width of 1000 for hidden features with a depth of 4 for both trunk and branch nets with ReLU activations. Learning rate is fixed to 10^{-5}.
- **MP-PDE.** We adapt the implementation in Brandstetter et al. (2022) to handle 2D and 3D PDEs. We use a time window of 1 with pushforward trick. Batch size and number of neighbors are fixed to 8. Learning rate is fixed to 10^{-3}. We use ReLU activations.
- **SIREN.** To represent data in space and time, SIREN takes space and time coordinates (x, t) as input. To handle multiple trajectories, we concatenate an optimizable per-trajectory context code α to the coordinates like in DINo. We fix the hidden layer size of SIREN to 256. We initialize the parameters and use the default input

Table D.3: DINo's hyperparameters.

Hyperparameter	Navier-Stokes	Wave	Shallow-Water
Decoder $D_\phi = I_{h_\phi}$			
Number of layers	3	3	6
Number hidden channels	64	64	256
Frequency scale factor ω_s	64	64	64
Size of latent space d_α	100	50	300
Dynamics model f_ψ			
Number of layers	4	4	4
Hidden layer size	512	512	800
Activation function	Swish	Swish	Swish
Optimization			
Learning rate η_ϕ	10^{-2}	10^{-2}	10^{-2}
Learning rate η_α	10^{-3}	10^{-3}	10^{-3}
Learning rate η_ψ	10^{-3}	10^{-3}	10^{-3}
Number of epochs	12 000	12 000	12 000
Batch size i.e. sequences per batch	64	64	16

scale as in Sitzmann et al. (2019). The size of the context code is $d_\alpha = 800$. The learning rate is 10^{-3}.

- **MFN.** Similarly to the previous SIREN baseline, we concatenate the per-trajectory context code to space and time coordinates at the first layer. The hidden layer size is fixed to 256 and we use the default parameter initialization with a frequency scale ω_s of 64 higher than DINo. The size of the context code is $d_\alpha = 800$. The learning rate is 10^{-3}.

D.4 Complementary Analyses

We detail in this section additional experiments, allowing us to further analyze and assess the performance of DINo.

D.4.1 Long-Term Temporal Extrapolation

We provide in table D.4 an analysis of error accumulation over time for long-term extrapolation. More precisely, we generate a Navier-Stokes dataset with longer trajectories and report MSE for $T' = T + \Delta T$ where $\Delta T \in \{T, 5T, 10T, 50T\}$. Note that $\Delta T = T$ is the

setting in our initial submission ($T' = 2T$).

We observe that DINo's MSE in long-term forecasting is more than an order of magnitude smaller than for (I-)MP-PDE. This demonstrates the extrapolation abilities of our model.

D.4.2 INRs' Advantage Over Interpolation

We report in Table D.5 the MSE of bicubic interpolation, our FourierNet's MSE (auto-decoding with amplitude modulation but without dynamics model) and DINo's MSE (with dynamics model) on train In-t for both *Navier-Stokes* and *Wave*. This corresponds to MSE averaged over all training frames within the train horizon and not only the initial condition v_0.

We observe that FourierNet is better than interpolation. Indeed, interpolation is poorly adapted to sparse observation grids: the interpolation errors are clearly visible in Figure D.2, first row (5% setting). Interestingly, DINo's MSE is only slightly worse than the FourierNet's MSE, showing that we correctly learned the dynamics of latent modulations α_t. I-MP-PDE, which combines bicubic interpolation with MP-PDE, is then expectedly outperformed by DINo on this challenging 5% setting. This shows the advantage of using INRs instead of standard bicubic interpolation to interpolate between observed spatial locations.

D.4.3 Modeling Real-World Data

SST. We evaluate DINo on real-world data to further assess its applicability. Following de Bézenac et al. (2018); Donà et al. (2021), we model the Sea Surface Temperature (SST) of the Atlantic ocean, derived from the data-assimilation engine NEMO (Nucleus for European Modeling of the Ocean, Gurvan et al., 2022) using E.U. Copernicus Marine Service Information.[1] Accurately modeling SST dynamics is critical in weather forecasting or planning of coastal activities. This problem is particularly challenging as SST dynamics are only partially observed: several unobserved variables affecting the dynamics (e.g., the sea water flow) need to be estimated from data.

For this experiment, we consider trajectories collected from three geographical zones (17 to 20) following the initial train / test split of de Bézenac et al. (2018). Notably, $T = 9$ d, which includes $\tau = 4$ d of conditioning frames, i.e. models are tested to predict $v_{t \in [\![\tau, T]\!]}$ from $v_{t \in [\![0, \tau]\!]}$.

Incorporating consecutive time steps. To model SST which includes non-Markovian data and thus does not correspond to an Initial Value Problem as in Section 7.2, we modify

[1] https://data.marine.copernicus.eu/product/GLOBAL_ANALYSIS_FORECAST_PHY_001_024/description

Table D.4: Long term extrapolation performance of DINo and (I-)MP-PDE in the space and time generalization experiment for test trajectories on Out-t ($]T, T' = T + \Delta T]$); cf. Table 7.3 and section 7.4.1.

Subsampling ratio	Model	$\Delta T = T$	$\Delta T = 5T$	$\Delta T = 10T$	$\Delta T = 50T$
$s = 5\%$	DINo	**2.017E−3**	**4.895E−3**	**1.209E−2**	**1.440E−1**
	I-MP-PDE	8.387E−3	3.580E−2	3.356E−1	4.031E1
$s = 100\%$	DINo	**4.617E−4**	**2.082E−3**	**6.901E−3**	**1.215E−1**
	MP-PDE	5.251E−4	3.524E−2	3.339E−1	9.755E1

Table D.5: MSE reconstruction error (*In-s* and *Out-s*) of train sequences within the train horizon (*In-t*) for three different methods: interpolation of observed points in $\mathcal{X}_{\text{tr}}$, FourierNet learned over individual frames in $\mathcal{X}_{\text{tr}}$, and DINo (FourierNet with a dynamics model).

MSE train In-t	Interpolation	FourierNet	DINo
Navier-Stokes, $s = 5\%$	8.277E−3	**9.673E−4**	1.029E−3
Wave, $s = 5\%$	7.075E−4	**4.085E−5**	4.088E−5

our dynamics model in a similar fashion to Yıldız et al. (2019) to integrate a history of several consecutive observations $v_{t\in[\![0,\tau]\!]}$ instead of only the initial observation v_0. In more details, we define a neural ODE over an augmented state $[\alpha_t, \alpha'_t]$ where α_t is our auto-decoded state and α'_t is an encoding of $\tau = 4$ past auto-decoded observations via a neural network c_ξ. We adjust our inference and training settings as follows:

- inference: we compute $\alpha'_{\tau-1} = c_\xi(\alpha_0, \cdots, \alpha_{\tau-1})$ and then unroll our neural ODE from the initial condition $[\alpha_{\tau-1}, \alpha'_{\tau-1}]$ to obtain $[\alpha_t, \alpha'_t]$ for all $t > \tau - 1$:

$$\forall t \in [\![0, \tau-1]\!], \alpha_t = e_\varphi(v_t), \quad \alpha'_{\tau-1} = c_\xi(\alpha_0, \cdots, \alpha_{\tau-1}), \quad \frac{\mathrm{d}[\alpha_t, \alpha'_t]}{\mathrm{d}t} = f_\psi([\alpha_t, \alpha'_t]);$$

- training: for all t, we infer $\alpha'_{t+\tau-1} = c_\xi(\alpha_t, \cdots, \alpha_{t+\tau-1})$ and fit the above neural ODE on the $[\alpha_t, \alpha'_t]$ obtained for all $t \in [\![0, T-\tau+1]\!]$.

This experiment confirms that our space- and time-continuous framework can easily be extended to incorporate refined temporal models.

Results. We report in Table D.6 test MSE for DINo and VarSep (Donà et al., 2021), the current state-of-the-art on SST, retrained on the same training data. DINo notably outperforms VarSep in prediction performance. This demonstrates DINo's potential to

Table D.6: SST test prediction performance for DINo and VarSep.

Method	MSE
VarSep (Donà et al., 2021)	1.43
DINo	**1.27**

handle complex real-world spatiotemporal dynamics. We also provide some visualizations of DINo's train and test predictions in Figure D.5. We make two observations. First, DINo fits very accurately the train data. Second, on the test, we observe that the dynamics on low frequencies seem to be correctly modeled while the prediction of high frequencies dynamics are less accurate. Larger scale experiments would be required to effectively evaluate the model performance on this challenging dataset. Given the complexity of the data, this is out of the scope of the paper. Yet, these experiments already demonstrate that DINo behaves competitively w.r.t. the previous state-of-the-art.

Implementation choices. We choose a similar INR and dynamics architecture than for our Shallow-water experiment. We use for c_ξ, which takes as input four consecutive α_ts, individual encodings of the α_ts through a four-layer fully connected network which are then fed to a single linear layer.

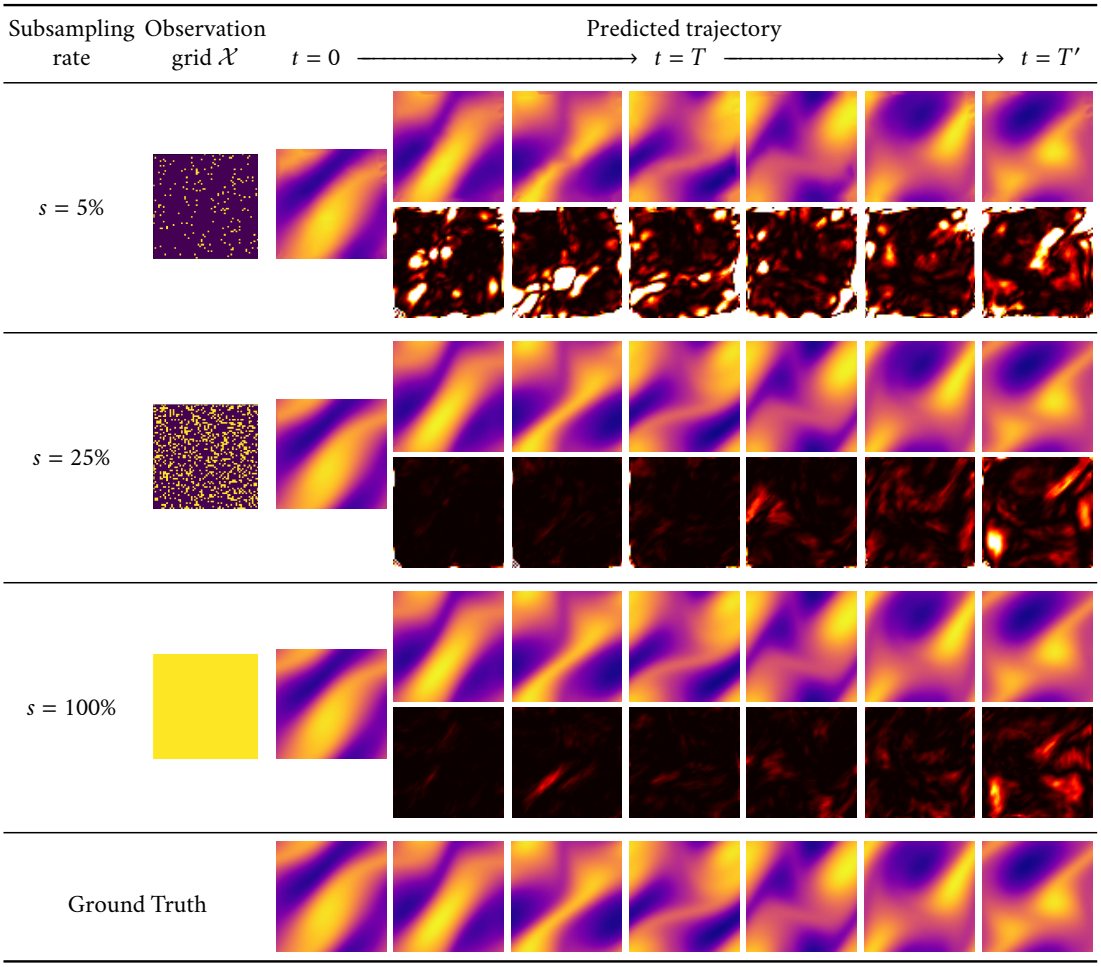


Figure D.2: Prediction MSE per frame for **I-MP-PDE** on *Navier-Stokes* with its corresponding observed grid $\mathcal{X}$. For each model, the first row contains the predicted trajectory from 0 to T', the second row is the corresponding error maps w.r.t. the reference data (the darker the pixel, the lower the error).

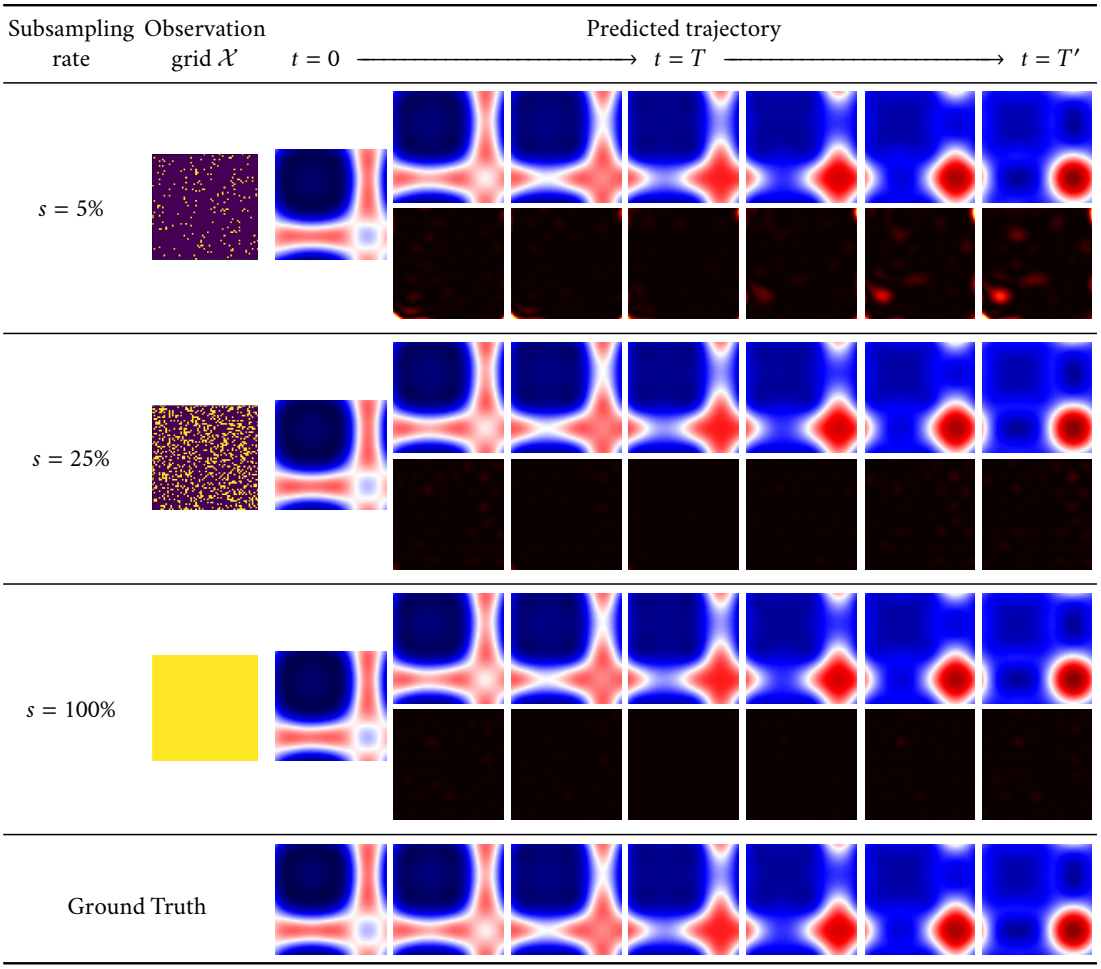


Figure D.3: Prediction MSE per frame for DINo on *Wave* with its corresponding observed grid $\mathcal{X}$. For each model, the first row contains the predicted trajectory from 0 to T', the second row is the corresponding error maps w.r.t. the reference data (the darker the pixel, the lower the error).

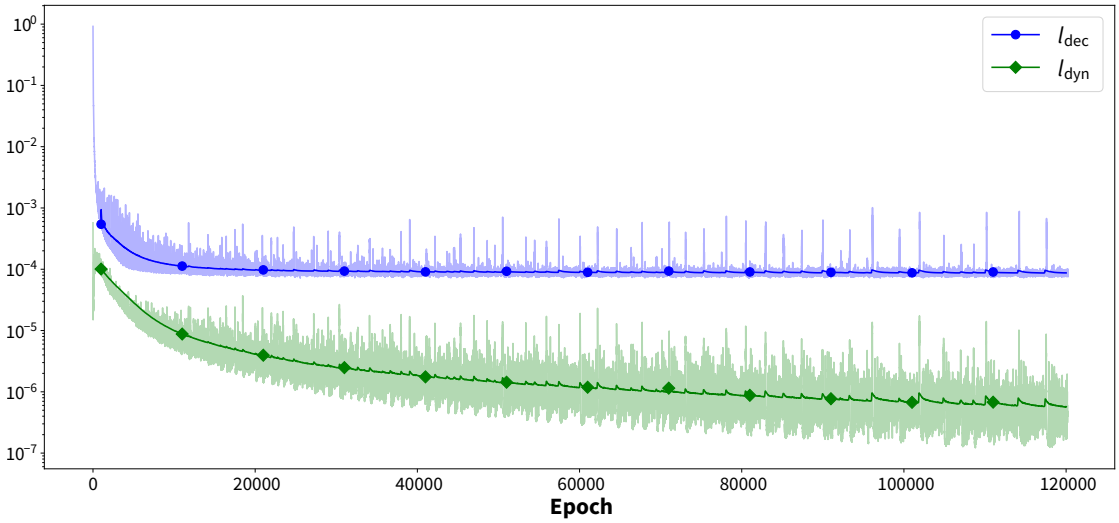
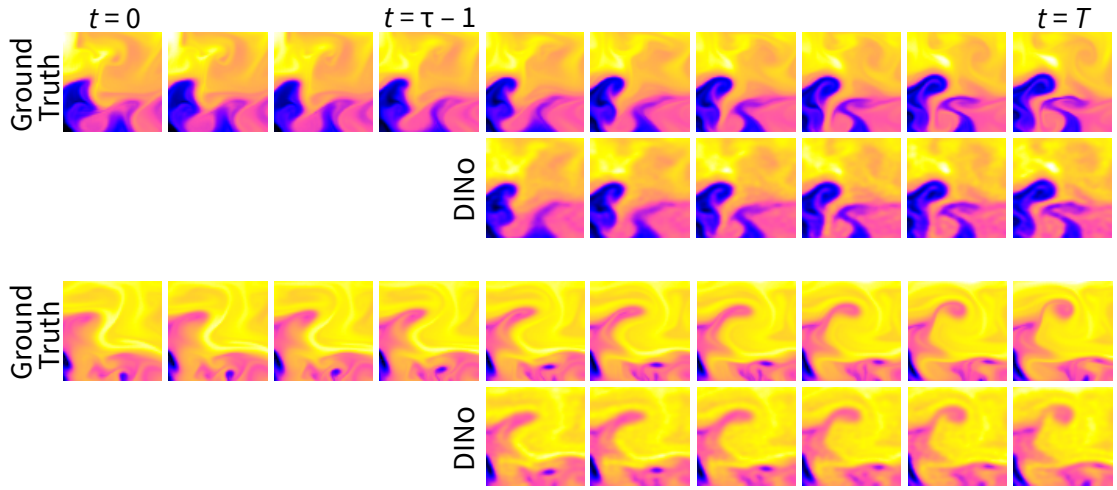
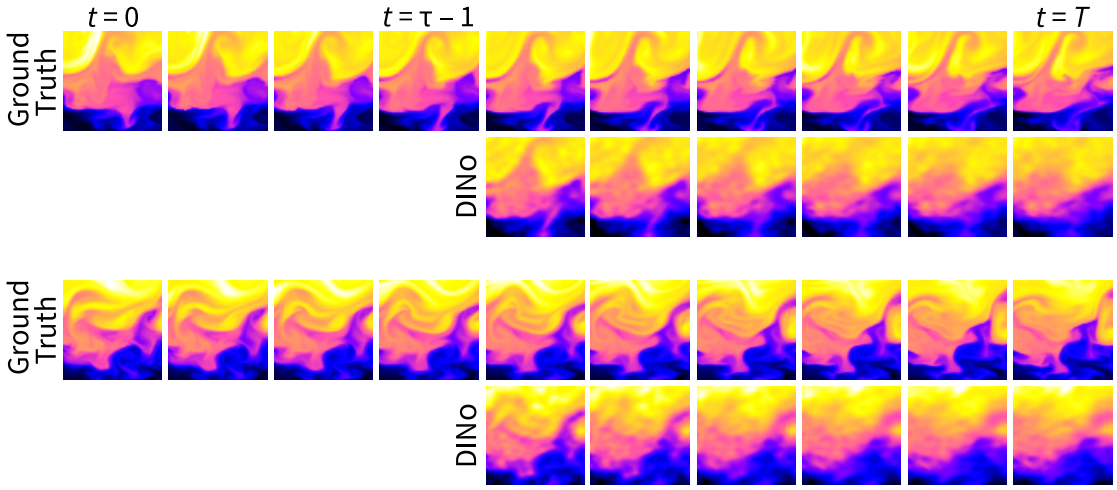


Figure D.4: Learning curves of DINo on *Navier-Stokes* for ℓ_{dec} and ℓ_{dyn} throughout training (pale lines) and corresponding exponential moving averages from epoch 500 with half-life 1000 (opaque lines).



(a) Train predictions.



(b) Test predictions.

Figure D.5: DINo's prediction examples on SST.

Appendix E

Résumé étendu de la thèse en français

Motiver la modélisation dynamique à l'aide de données

La modélisation dynamique est un domaine d'étude fondamental en science, qui s'étend sur des siècles de recherche dans de multiples disciplines. À la base, la modélisation dynamique cherche à expliquer et à prédire le mouvement ou les changements des objets par l'observation et l'analyse. Les connaissances acquises grâce à cette compréhension ont conduit à l'élaboration de modèles efficaces capables de prédire l'évolution d'un phénomène dans le temps et sa réaction à des changements dans l'environnement ou dans d'autres conditions. Aujourd'hui, les efforts de modélisation des systèmes dynamiques couvrent un large éventail de méthodes, impliquant divers degrés de principes premiers : des modèles numériques sophistiqués, reposant sur une connaissance complète de

la physique, aux modèles purement fondés sur des données sans aucune connaissance physique préalable.

Dans cette thèse, nous contribuons à la tendance récente qui consiste à appliquer l'apprentissage profond (*deep learning*, DL) à la modélisation de la dynamique. Suite aux percées réalisées par le DL dans divers domaines (He et al., 2016; Goodfellow et al., 2020; Devlin et al., 2019; Brown et al., 2020), la polyvalence du DL a commencé à être exploitée pour divers aspects de la modélisation de la dynamique. Néanmoins, ce domaine de recherche en est encore à ses débuts.

Notre objectif est d'améliorer la généralité et l'adaptabilité des modèles d'apprentissage profond pilotés par les données en tenant compte des a priori physiques ou distributionnels dans leurs hypothèses sous-jacentes. Nous visons à exploiter les approches DL pour obtenir des modèles qui peuvent extraire efficacement des informations des données proches des scénarios du monde réel. Ce résumé souligne l'importance de la modélisation avec des données et motive les différents problèmes que nous voulons aborder, afin de faire correspondre les capacités des modèles avec les méthodes numériques.

Comprendre la nature avec les équations différentielles

L'étude de la dynamique a commencé avec le développement de modèles mathématiques, visant à comprendre les mouvements des objets en modélisant les lois physiques. De Galilée à nos jours, les scientifiques ont tenté d'expliquer les phénomènes observés en construisant des modèles dynamiques qui expriment, p. ex., les lois du mouvement, avec les propriétés des objets, p. ex., la masse, la longueur et la densité des objets. Ces modèles ont été créés à partir de concepts abstraits des observations et basés sur la perception humaine de la réalité et des lois physiques.

Par exemple, lors de la modélisation du comportement d'un objet en mouvement, au lieu de décrire exhaustivement toutes les trajectoires d'évolution de la position de l'objet, on étudie la vitesse, c.-à-d., le *taux de variation* de la position, et l'accélération, c.-à-d., le *taux de variation* de la vitesse. Par conséquent, les règles concernant ces taux de changement, les équations différentielles, fournissent une description idéale claire, concise et concevable de l'évolution temporelle des phénomènes et sont considérées comme des lois de la physique.

À ce jour, un grand nombre de dynamiques formulées sous forme d'équations différentielles ont été proposées dans diverses disciplines. Par exemple, en physique, les équations de Navier-Stokes (Stokes, 1851) décrivent le mouvement des fluides visqueux, p. ex., l'eau, l'air ; les équations de réaction-diffusion (Pearson, 1993) décrivent comment une ou plusieurs substances chimiques réagissent localement et se propagent dans l'espace,(Feinberg, 2019) ; la dynamique épidémiologique étudie comment les maladies infectieuses se propagent dans une population (Martcheva, 2015). Elles reposent sur une compréhension approfondie des phénomènes dynamiques sous-jacents.

Si une solution analytique à l'équation différentielle est disponible, nous pouvons l'utiliser pour prédire ce qui va se passer. Cependant, comme la plupart des systèmes dynamiques sont non linéaires, leur comportement ne peut pas être entièrement décrit par les seules méthodes analytiques. Par conséquent, nous devons généralement recourir à des méthodes numériques pour pouvoir utiliser les équations différentielles, ce qui a conduit au développement de modèles numériques capables de fournir une approximation de la solution analytique intraitable.

Modèles numériques

Un *modèle numérique* est une version discrétisée ou simplifiée de l'équation différentielle. Ces modèles sont ensuite résolus par un *solveur numérique*. Les solutions approximatives fournies par les modèles numériques et leurs solveurs conservent les propriétés physiques et sont utilisées pour un large éventail d'applications.

Les modèles numériques présentent les caractéristiques de robustesse et de flexibilité suivantes, qui influenceront grandement les exigences en matière de capacité lors de la proposition d'autres types de modèles, y compris les approches DL :

- *Ils sont souvent robustes aux changements d'état.* Le lien étroit avec l'équation différentielle rend les modèles numériques applicables à une large gamme de valeurs d'état. Cela permet d'appliquer le même modèle dans des conditions différentes.
- *Ils décrivent des systèmes différents, obéissant aux mêmes lois physiques dans des contextes différents, souvent paramétrisés par quelques paramètres.* Le passage d'un système à un autre peut se faire directement en modifiant les paramètres. Par exemple, lorsqu'on déplace un pendule de la Terre vers la planète Mars, il suffit de modifier la gravité dans l'équation pour prédire le pendule dans le nouvel environnement.
- *Les approximations sont proches de la réalité physique et reflètent des propriétés importantes de la dynamique.* Lorsqu'elle prédit des quantités physiques, la solution doit également respecter les lois physiques fondamentales. Par exemple, les symétries dans le temps et l'espace pour les changements d'état, la conservation du volume et de l'énergie, l'incompressibilité des fluides, etc.

Cependant, il n'est jamais facile de se confronter à la réalité. Malgré de nombreux efforts pour construire des équations différentielles, des modèles numériques et des solveurs réalistes, il existe encore de nombreux goulets d'étranglement qui entravent leur application à la prévision dans le monde réel.

- *Rare consensus sur la modélisation et la résolution d'un même phénomène.* Pour de nombreux phénomènes, il est presque impossible d'obtenir des équations unanimement acceptées par les scientifiques. Par exemple, pour les équations des eaux peu profondes, alias équations de Saint-Venant, il n'y a pas de consensus sur la

modélisation réaliste du frottement (Delestre, 2010, Chapitre 3). Un ensemble de modèles est proposé pour décrire la dynamique épidémique du même phénomène (Schneckenreither et al., 2008). Parfois, le modèle et le solveur doivent être adaptés à chaque paramétrage, voire à des conditions différentes. Par exemple, pour résoudre l'équation de Navier-Stokes, différents modèles numériques doivent être sélectionnés en fonction des paramètres physiques du modèle, de la forme du domaine, de la vitesse initiale de l'écoulement, etc.

- *Un modèle ne peut décrire qu'une partie de la réalité.* Les modèles d'équations différentielles et de dynamique numérique sont construits avec différents niveaux de simplification de la réalité. Pour des raisons pratiques, la méthode numérique est souvent trop simplifiée par rapport à la dynamique observée.
- *Les prédictions de haute qualité nécessitent une puissance de calcul importante.* Lorsque le phénomène est intrinsèquement complexe, il nécessite une attention particulière aux détails à petite échelle et entraîne souvent une charge de calcul extrêmement élevée en termes de mémoire et de temps. Par exemple, lors de la résolution des équations de Navier-Stokes, l'obtention d'une prédiction de haute qualité nécessite souvent une discrétisation dense du domaine sur lequel l'équation doit être résolue. Dans certaines expériences, comme la simulation numérique directe de l'équation de Navier-Stokes, l'ensemble du domaine comporte plus de $> 10^{11}$ degrés de liberté (Lee et al., 2013). Même avec un nombre de Reynold moyen (plus le nombre est élevé, plus les écoulements de fluides sont complexes), la résolution d'un tel modèle numérique de Navier-Stokes induit des coûts de calcul et une charge de mémoire considérables qui nécessitent une puissance de calcul extrêmement importante.

Ces goulets d'étranglement limitent les applications des modèles numériques dans le monde réel et poussent les scientifiques à chercher de l'aide pour mieux décrire la réalité, réduire le coût de fonctionnement de ces modèles et rechercher des contreparties complémentaires ou même des alternatives aux modèles numériques.

Modélisation avec des données : Vers des applications plus larges

Au cours des dernières décennies, la disponibilité croissante des données, générées soit par l'observation du phénomène réel des systèmes dynamiques, soit par la résolution d'équations à l'aide de modèles numériques, a favorisé le développement d'approches basées sur les données. En fonction de la source de données et des objectifs de la communauté, différentes voies ont été proposées.

Le premier type de données consiste en des informations collectées par l'observation de phénomènes réels, comme des images satellites de l'océan ou de l'atmosphère terrestre. Ces données sont utilisées pour guider les modèles numériques afin qu'ils fassent des prédictions plausibles et cohérentes avec les nouvelles observations. Ce processus est

appelé l'assimilation des données (Kalman, 1960; Courtier et al., 1994), et il s'agit d'une procédure séquentielle à pas de temps. La stratégie courante comporte deux étapes qui se répètent à chaque mise à jour : le modèle numérique fournit une prévision à court terme qui est comparée aux nouvelles observations reçues, et l'état du modèle est ensuite mis à jour pour refléter les observations. L'assimilation des données a été utilisé avec succès dans les systèmes de prévision météorologique à grande échelle tels que le Centre européen pour les prévisions météorologiques à moyen terme (CEPMMT ; Bonavita and Lean, 2021) et les systèmes opérationnels de surveillance et de prévision des océans tels que MERCATOR Océan (Ferry et al., 2007). L'assimilation des données peut effectivement trouver la meilleure estimation de la réalité à partir de la prévision du modèle numérique par rapport aux données. Cependant, les informations provenant des données jouent le rôle de correction du modèle numérique, et non de partie de celui-ci.

Le deuxième type de données concerne les solutions du modèle numérique. Les simulations peuvent être utilisées pour construire des modèles émulateurs moins complexes et moins coûteux en calcul que les modèles numériques, tout en maintenant une qualité raisonnable des solutions. Par rapport aux modèles numériques originaux, à savoir les modèles d'ordre complet (*full-order model*, FOM), les modèles d'ordre réduit (*reduced-order models*, ROM) basé sur les données extraient un modèle de moindre complexité à partir d'un ensemble de solutions données par les FOM. Les ROM sont principalement liées aux techniques de réduction des dimensions accompagnées d'hypothèses physiques. Pour les systèmes dynamiques, il existe des techniques telles que la décomposition orthogonale aux valeurs propres (*proper orthogonal decomposition*, POD) pour trouver la meilleure approximation à faible dimension du FOM (Choi et al., 2021), et la décomposition des modes dynamiques pour la meilleure approximation linéaire de la dynamique non linéaire (Kutz et al., 2016). Ils fournissent des représentations mathématiques pour l'analyse en temps réel, mais les plus efficaces sont souvent limités à la réduction linéaire et à la modélisation linéaire, ce qui limite leurs scénarios d'application et nécessite des extensions pour une modélisation plus flexible.

Modélisation de l'apprentissage en profondeur et de la dynamique

Bien que les réseaux de neurones aient été proposées il y a plus d'un demi-siècle, ce n'est que depuis les années 2010 qu'elles ont été largement popularisées et que cette branche d'étude, sous le nom du DL, est devenue le sujet de recherche le plus influent de l'apprentissage automatique (*machine learning*, ML) à ce jour. Grâce à des algorithmes flexibles, à d'énormes quantités de données disponibles, à des logiciels polyvalents avancés et à du matériel puissant pour le déploiement de modèles à grande échelle, etc., les chercheurs ont réalisé de multiples percées dans diverses tâches, p. ex., la reconnaissance d'images (He et al., 2016), la génération (Goodfellow et al., 2020), la représentation de texte (Devlin et al., 2019), la génération (Brown et al., 2020). Le DL met en évidence les algorithmes de réseau de

neurone polyvalents et flexibles pour modéliser toute correspondance non linéaire entre les données données et la sortie souhaitée, réalisant des modèles ML opérationnels qui n'étaient pas possibles auparavant. Cela permet d'extraire efficacement des dynamiques complexes hautement non linéaires directement à partir des données.

Avant l'ère du DL, des tentatives ont déjà été faites pour proposer des modèles prédictifs basés sur les réseaux de neurones pour les systèmes dynamiques. La combinaison des réseaux de neurones avec des schémas d'intégration numérique a également été envisagée. Cependant, l'apprentissage des systèmes dynamiques pilotés par des équations différentielles ordinaires (EDO) ou les équations différentielles ordinaires (EDP) linéaires ou non linéaires, p. ex., Rico-Martínez and Kevrekidis (1993) est resté relativement confidentiel à cette époque.

Grâce aux progrès de l'industrie générale du DL, les efforts récents ont rajeuni cette recherche, à savoir *apprentissage en profondeur pour les systèmes dynamiques*. Des travaux pionniers ont été réalisés sur l'intégration d'informations préalables physiques dans les modèles DL/ML pour les systèmes dynamiques (Long et al., 2018b; de Bézenac et al., 2018; Raissi et al., 2019; Brunton and Kutz, 2022) en se concentrant sur l'utilisation de techniques DL.

Du côté du DL, le lien entre les systèmes dynamiques et les architectures de réseau de neurones modernes a été mis en évidence par ResNet (He et al., 2016), populaire dans la vision par ordinateur. Les connexions résiduelles mettent en œuvre une méthode de pas de temps d'Euler en avant. Des auteurs, notamment E (2017), utilisent cette relation pour motiver l'utilisation des réseaux de neurones comme solveur pour les EDO :

$$z_{t+\delta t} = z_t + f_\theta(t, z_t) \qquad \text{vs.} \qquad \frac{\mathrm{d}z}{\mathrm{d}t}(t) = f_\theta(t, z_t)$$

En suivant cette voie, des architectures ont été proposées en exploitant la théorie des EDO qui garantit certaines propriétés physiques, telles que la stabilité ou la conservation de l'énergie (Haber and Ruthotto, 2017; Ruthotto and Haber, 2020).

Le travail qui a le premier attiré l'attention des communautés de l'apprentissage profond et de la modélisation dynamique est les *neural ODE*, les EDO neuronales. En remplaçant les ResNet traditionnels par un solveur numérique, il a suscité l'intérêt des deux communautés et relancé l'idée de l'intégration des réseaux neuronaux dans les solveurs différentiables. En conséquence, le domaine de la modélisation avec le DL pour les systèmes dynamiques et les équations différentielles a connu une croissance importante et continue de faire des progrès impressionnants à ce jour.

Aujourd'hui, le DL intervient dans presque tous les types de modélisation dynamique :

- *Agir comme un modèle de prévision purement piloté par les données.* Il s'agit du cadre

de base de l'apprentissage profond : découvrir des lois à partir de données. Les modèles sont formés à partir des trajectoires spatiotemporelles observées dans le but de prévoir les observations futures (Chen et al., 2018; Pfaff et al., 2021; Li et al., 2021b; Brandstetter et al., 2022) ;

- *Agir comme un modèle numérique.* L'objectif est ici de trouver des alternatives aux modèles numériques et à leurs solveurs. On utilise l'équation différentielle entièrement connue et les conditions pour trouver la solution inconnue sans données, comme Sirignano and Spiliopoulos (2018); Raissi et al. (2019) et d'autres travaux similaires ;
- *Agir dans le cadre d'un modèle hybride.* Améliorer les modèles numériques à faible coût pour approcher la simulation numérique de haute précision (Belbute-Peres et al., 2020; Kochkov et al., 2021) ou compléter un modèle numérique en fournissant certains composants de ce modèle (de Bézenac et al., 2018; Ayed et al., 2022).

Défis du monde réel

Malgré les récentes percées dans le domaine de l'apprentissage profond pour la modélisation physique, plusieurs problèmes liés aux capacités des modèles dynamiques appris et aux données d'entraînement ne sont toujours pas résolus.

Défis liés aux exigences en matière de capacité des modèles

Les modèles dynamiques en ML sont devenus de plus en plus importants dans divers domaines scientifiques et techniques. Cependant, ils sont confrontés à plusieurs défis en termes de précision et d'adaptabilité, qui sont essentiels pour leur application réussie :

Le modèle doit être applicable à des situations inédites. Un aspect fondamental d'un modèle d'apprentissage automatique réussi est sa capacité à bien se généraliser. Il s'agit d'utiliser un modèle entraîné pour faire des prédictions précises même dans des situations qu'il n'a jamais vues auparavant, ce qui est essentiel pour les applications du monde réel. Dans le cas d'un système dynamique, cela signifie que le modèle doit être capable de faire des prédictions précises même lorsqu'il est confronté à de nouveaux états initiaux qu'il n'a pas rencontrés au cours de la formation. En d'autres termes, le modèle doit, en particulier, être capable de résister aux changements dans la distribution des états initiaux au fil du temps.

Le modèle doit s'adapter facilement à des dynamiques inédites. En outre, un bon modèle dynamique doit également être capable de s'adapter aux changements dans la dynamique sous-jacente du système. Si la dynamique change légèrement à l'avenir, ou

dans d'autres circonstances, le modèle doit pouvoir être adapté rapidement avec peu de données observées, sans qu'il soit nécessaire de procéder à un réapprentissage complet du modèle.

Le modèle doit produire des prédictions physiquement fiables. Les états estimés ne doivent pas seulement ressembler visuellement à la réalité, mais ils doivent également respecter les propriétés indiquées par les lois de la physique et les mesures réelles des quantités d'intérêt. Par exemple, des prévisions précises de quantités physiques clés au niveau de la couche limite d'un profil aérodynamique sont essentielles et nécessitent des prévisions précises dans des régions spécifiques. Il est essentiel de garantir que le modèle produit des prévisions physiquement fiables pour assurer son applicabilité pratique, en particulier dans les contextes où la précision et la fiabilité sont essentielles.

Défis liés aux données

Les méthodes ML apprennent la dynamique à partir des données. Cependant, dans le monde réel, les données sont complexes et nous sommes confrontés à de multiples défis, en termes de quantité, de variété et d'irrégularité :

Les données ne sont pas toujours abondantes dans certains scénarios. Compte tenu de la complexité des architectures de réseau de neurones modernes et de leur nature non linéaire, la plupart des méthodes DL sont sujettes à des problèmes de généralisation, en particulier lorsque les données d'entraînement sont rares. Ce cas nous intéresse toujours car, dans de nombreux scénarios, il est encore difficile et coûteux d'obtenir des données pour des systèmes dynamiques réalistes en raison de l'infrastructure de calcul, par exemple les superordinateurs pour les simulations à grande échelle. En outre, même si les données existent, elles peuvent être détenues par des entités gouvernementales et industrielles, et leur disponibilité est limitée par des politiques de confidentialité, a.k.a. le problème des données à source fermée. Cette situation est très différente des problèmes rencontrés dans d'autres sous-domaines, tels que la vision par ordinateur et le traitement du langage naturel, qui disposent d'une grande quantité de données librement accessibles grâce à Internet, ce qui nous oblige, au moins pour la prochaine décennie, à envisager des modèles moins gourmands en données et contenant davantage de régularités qui favorisent l'application à de nouvelles données.

Les données proviennent de sources hétérogènes. Dans la plupart des cas, les données récupérées ne doivent pas être uniformément réparties dans l'espace des paires entrée-sortie. Cela signifie que nous devrions rencontrer des données provenant de différentes sources, de différentes qualités/résolutions et même de différentes dynamiques. Dans ce dernier cas en particulier, nous récupérons souvent des données relatives au même

phénomène physique dans différents environnements, chaque environnement ayant sa propre dynamique instanciée. Par exemple, nous observons des épidémies qui propagent diverses maladies infectieuses dans les pays. Elles suivent toutes la même loi générale, mais les valeurs observées dépendent de certains facteurs, tels que la contagiosité de la maladie et la structure de la population. Cela signifie que même avec la même condition initiale, les trajectoires d'évolution résultantes varieront en fonction des instances du système, qui peuvent dépendre de la paramétrisation du système :

- *Ils peuvent différer dans les lois d'évolution.* Cela signifie que les dynamiques sous-jacentes sont fondamentalement différentes. Par exemple, la dynamique des océans à différents endroits a la force de Coriolis correspondante en fonction de la latitude sur Terre. Ce sera l'un des principaux sujets de cette thèse.

- *Ils peuvent différer dans les conditions qui déterminent la trajectoire.* Même avec la même dynamique, la variation de certaines conditions entraîne des changements dans l'espace des trajectoires. Par exemple, la modification des températures sur la frontière du domaine des équations de la chaleur conduira à des solutions différentes.

- *Ils peuvent différer dans la forme et la géométrie de l'état du système.* La modélisation de la dynamique spatio-temporelle peut s'avérer difficile, en particulier lorsqu'il s'agit de géométries complexes. Par exemple, la dynamique des fluides autour d'un profil aérodynamique, d'une aile d'avion ou d'une voile de bateau, dépend non seulement des propriétés du fluide mais aussi de la forme de l'objet, qui définit le domaine spatial des trajectoires.

Le défi consiste à exploiter efficacement des sources de données hétérogènes afin de découvrir des points communs entre les données, en particulier dans les cas où les données présentent des dynamiques différentes. L'un des principaux objectifs est alors de développer des modèles qui se généralisent bien à de nouvelles données présentant une dynamique similaire, une fois que le modèle a été entraîné sur les données existantes.

Les données sont observées de manière irrégulière dans l'espace et le temps. Les données du monde réel sont rarement récupérées sur une grille prédéfinie à des endroits spécifiques. Le phénomène en question n'est presque jamais observé dans son intégralité et fait l'objet de mesures discrétisées. Compte tenu de la disponibilité imprévisible des capteurs, les modèles doivent être capables de traiter les données qui en sont extraites et d'être robustes face à tout changement susceptible de se produire. Les modèles doivent donc imposer moins de restrictions sur le format d'entrée/sortie, à la fois dans l'espace et dans le temps.

Contributions de la thèse

Dans cette thèse, nous avons abordé certains de ces défis à travers les problèmes suivants : (a) développement d'un cadre formel solide pour la modélisation hybride (APHYNITY); (b) traiter les problèmes de généralisation (LEADS et CoDA) ; et (c) la modélisation sans maillage (DINo).

Nous résumons ci-dessous nos contributions, qui seront examinées successivement dans la thèse :

APHYNITY, Yin et al. (2021b) En raison des problèmes de généralisation, les approches purement axées sur les données sont sans doute insuffisantes. Nous nous efforçons de faire fonctionner ensemble les modèles numériques et les modèles guidés par les données afin de prévoir des phénomènes dynamiques complexes pour lesquels on ne dispose que d'une connaissance partielle de leur dynamique. Dans ce travail, nous présentons le cadre APHYNITY, qui consiste à décomposer la dynamique en deux composantes : une composante physique formulée à partir de premiers principes partiellement connus, et une composante pilotée par les données qui complète la composante physique précédente en décrivant la dynamique qui ne peut pas être capturée par le modèle physique, ni plus ni moins. APHYNITY améliore les capacités des deux composantes : le modèle hybride prédit bien dans de nouvelles conditions, atteint une meilleure généralisation que l'une ou l'autre méthode seule et aide à identifier l'instance de modèle numérique incomplète appropriée à partir d'un grand ensemble de candidats. Cela permet non seulement d'assurer l'existence et l'unicité de cette décomposition, mais aussi d'en garantir l'interprétabilité et de favoriser la généralisation. Des expériences sur divers phénomènes montrent qu'APHYNITY peut utiliser efficacement des modèles physiques incomplets pour prévoir avec précision l'évolution du système et identifier correctement les paramètres physiques pertinents.

LEADS, Yin et al. (2021a) Lors de la modélisation de systèmes dynamiques à partir des échantillons de données du monde réel, la distribution des données change souvent en fonction de l'environnement dans lequel elles sont capturées, et la dynamique du système lui-même varie d'un environnement à l'autre, ce qui suggère que dans les mêmes conditions, les trajectoires changent dans des environnements différents. Dans ce cas, la généralisation à travers les environnements défie les cadres conventionnels. Les cadres classiques proposent soit de considérer les données comme identiques et d'apprendre un modèle unique pour couvrir toutes les situations, soit d'apprendre des modèles spécifiques à l'environnement. Ces deux approches sont sous-optimales : la première ne tient pas compte des différences entre les environnements, ce qui conduit à des solutions biaisées, tandis que la seconde n'exploite pas leurs points communs potentiels et est sujette à des problèmes de

pénurie. Dans ce travail, nous proposons LEADS, un nouveau cadre qui exploite les points communs et les différences entre les environnements connus afin d'améliorer la généralisation du modèle dans le phénomène. Nous y parvenons grâce à une formulation d'apprentissage sur mesure visant à capturer les dynamiques communes au sein d'un modèle partagé, tandis que des termes supplémentaires capturent les dynamiques spécifiques à chaque environnement. Nous fondons notre approche sur la théorie, en démontrant une diminution de la complexité de l'échantillonnage par rapport aux alternatives classiques. Nous montrons comment la théorie et la pratique coïncident dans le cas simplifié de la dynamique linéaire. En outre, nous instancions ce cadre pour les réseaux neuronaux et l'évaluons expérimentalement sur des familles représentatives de dynamiques non linéaires. Ce nouveau cadre peut exploiter les connaissances extraites des données dépendantes de l'environnement et améliore la généralisation pour les environnements connus et nouveaux, ce qui nous permet de faire un premier pas vers une adaptation efficace à de nouvelles dynamiques.

CoDA, Kirchmeyer et al. (2022) En suivant la voie tracée par la contribution précédente, nous nous dirigeons vers l'adaptation à un nouveau système dynamique basé sur les données d'un ensemble de dynamiques connues. Les approches de modélisation des systèmes physiques basées sur les données ne parviennent pas à se généraliser à des systèmes inédits qui partagent la même dynamique générale que les données d'apprentissage, mais qui correspondent à des contextes physiques différents, c'est-à-dire à des paramètres différents de la dynamique. Dans ce travail, nous proposons un nouveau cadre pour aborder ce problème clé, l'adaptation dynamique informée par le contexte (CoDA), qui prend en compte le changement de distribution à travers les systèmes pour une adaptation rapide et efficace à de nouvelles dynamiques. CoDA exploite de multiples environnements, chacun associé à une dynamique différente, et apprend à conditionner le modèle de dynamique sur des paramètres contextuels, spécifiques à chaque environnement. Le conditionnement est effectué par l'intermédiaire d'un hyper-réseau, appris conjointement avec un vecteur de contexte à partir de données observées. La formulation proposée contraint l'espace des hypothèses de recherche pour une adaptation rapide et une meilleure généralisation dans des environnements avec peu d'échantillons. Nous motivons théoriquement notre approche et montrons des résultats de généralisation de pointe sur un ensemble de dynamiques non linéaires, représentatives d'une variété de domaines d'application. Nous montrons également, sur ces systèmes, que les nouveaux paramètres du système peuvent être déduits des vecteurs de contexte avec une supervision minimale.

DINo, Yin et al. (2023) Inspiré par l'approche contextuelle précédente, nous abordons le problème de l'apprentissage de la dynamique avec des données échantillonnées

de manière irrégulière dans l'espace et dans le temps. Les méthodes de prédiction efficaces basées sur les données reposent souvent sur une discrétisation spatiale et/ou temporelle fixe. Cela pose des limitations dans les applications du monde réel, comme les prévisions météorologiques, où une extrapolation flexible à des emplacements spatio-temporels arbitraires est nécessaire. Nous abordons ce problème en introduisant une nouvelle approche basée sur les données, DINo, qui modélise le flux d'une EDP avec une dynamique en temps continu de fonctions spatialement continues. Pour ce faire, les observations spatiales sont intégrées indépendamment de leur discrétisation via des représentations neuronales implicites dans un petit espace latent piloté temporellement par une EDO apprise. Ce traitement séparé et flexible du temps et de l'espace fait de DINo le premier modèle piloté par les données à combiner les avantages suivants. Il extrapole à des emplacements spatiaux et temporels arbitraires ; il peut apprendre à partir de grilles irrégulières peu denses ou de manifolds ; au moment du test, il se généralise à de nouvelles grilles ou résolutions. DINo surpasse les prédicteurs neuronaux alternatifs dans une variété de scénarios de généralisation difficiles sur des systèmes EDP représentatifs.

Glossary

A | B | C | D | E | F | G | I | L | M | N | O | P | R | S

A

AUTODIFF automatic differentiation 26, 38, 39, 46

B

BC boundary condition 21, 38, 39
BVP boundary value problem 38

C

ConvLSTM convolutional LSTM (Shi et al., 2015) 37
ConvNet convolutional neural network 27, 28, 30, 31, 36, 38, 47–49, 57, 94, 101, 116, 126, 136, 140, 150, 231
CV computer vision 9, 11

D

DA data assimilation 8, 45
DFT discrete Fourier transform 34, 35, 59
DGM deep Galerkin method (Sirignano and Spiliopoulos, 2018) 38, 75
DL deep learning 4, 6, 9, 11, 16, 23, 26, 28, 39–41, 47, 49, 54, 56, 66
DMD dynamic mode decomposition 8

E

ERM empirical risk minimization 40, 51, 87, 97, 105, 111

F

FFT fast Fourier transform 35, 59
FNO Fourier neural operator (Li et al., 2021b) 34, 35, 38, 59, 94, 101, 116, 136, 205, 231
FOM full-order model 8

G

GBML gradient-based meta learning 52, 54, 97, 116, 118–120, 221, 222
GCN graph convolutional network 31
GD gradient descent 26
GNN graph neural network 30, 31, 36, 46, 48, 57, 126, 136, 140
GNO graph neural operator (Li et al., 2020) 58
GRU gated recurrent unit (Cho et al., 2014) 37, 101

I

IBVP initial boundary value problem 19, 20, 38
IC initial condition 19, 20, 38, 39
INR implicit neural representation 39, 59–61, 127, 129, 130, 132–135, 137, 140, 141, 150, 244, 247, 249
IRM invariant risk minimization 51
IVP initial value problem 19, 20, 37, 45

L

LSTM long short-term memory (Hochreiter and Schmidhuber, 1997) 37

M

MAML model-agnostic meta learning (Finn et al., 2017) 52, 53, 56, 97, 222
MAPE mean absolute percentage error 116, 118–121
MB model-based 66, 67, 70
MFN multiplicative filter network 134, 137
ML machine learning 7, 9–11, 14, 40, 44, 47, 50, 66, 67, 70, 84, 86, 102, 105
MLP multi-layer perceptron 26, 27, 33, 34, 36, 48, 54, 60, 94, 101, 116, 187, 231
MNO Markov neural operator 128, 136, 138–141
MPNN message passing neural network 31, 32, 57, 59
MSE mean square error 23, 68, 75–78, 92, 98, 99, 107, 108, 116, 118, 137–142, 203, 204
MTL multi-task learning 54, 56, 221, 223

N

Neural ODE neural ordinary differential equation (Chen et al., 2018) 9, 37, 58, 75–77, 79, 80, 97
NLP natural language processing 11
NN neural network 46, 49, 50, 52, 53, 55, 59, 60, 66, 88, 93, 94, 97, 102, 104, 205, 232
NO neural operator (Kovachki et al., 2021) 34, 35, 58, 59

O

ODE ordinary differential equation 9, 15, 19, 29, 37, 49, 66, 68, 71, 73–75, 78, 86, 94, 102, 106, 107, 109, 115, 116, 118, 127, 136, 142
OoD out-of-distribution 51

P

PDE partial differential equation 9, 15, 19, 20, 30, 32, 36, 38–40, 47, 48, 66–68, 72, 73, 86, 94, 102, 106, 107, 112, 113, 115, 118, 126, 127, 129, 136, 137, 142, 143
PINN physics-informed neural network (Raissi et al., 2019) 38, 39, 60, 152
POD proper orthogonal decomposition 8

R

RBF radial basis function 50
ReLU rectified linear unit 25
ResNet residual neural network (He et al., 2016) 9, 24, 28, 29, 34, 37, 58
RNN recurrent neural network 36, 37, 48, 56
ROM reduced order model 8, 61

S

SGD stochastic gradient descent 26, 53
SST sea surface temperature 49

Table of Contents

Physics-Aware Deep Learning and Dynamical Systems: Hybrid Modeling and Generalization

Abstract

Deep learning has made significant progress in various fields and has emerged as a promising tool for modeling physical dynamical phenomena that exhibit highly nonlinear relationships. However, existing approaches are limited in their ability to make physically sound predictions due to the lack of prior knowledge and to handle real-world scenarios where data comes from multiple dynamics or is irregularly distributed in time and space. This thesis aims to overcome these limitations in the following directions: improving neural network-based dynamics modeling by leveraging physical models through hybrid modeling; extending the generalization power of dynamics models by learning commonalities from data of different dynamics to extrapolate to unseen systems; and handling free-form data and continuously predicting phenomena in time and space through continuous modeling. We highlight the versatility of deep learning techniques, and the proposed directions show promise for improving their accuracy and generalization power, paving the way for future research in new applications.

Keywords: deep learning, dynamical system, physical phenomenon, neural network, hybrid modeling, generalization, adaptation, out-of-distribution, continuous dynamics modeling

---

Apprentissage profond pour la physique et les systèmes dynamiques : Modélisation hybride et généralisation

Résumé

L'apprentissage profond a fait des progrès dans divers domaines et est devenu un outil prometteur pour modéliser les phénomènes dynamiques physiques présentant des relations hautement non linéaires. Cependant, les approches existantes sont limitées dans leur capacité à faire des prédictions physiquement fiables en raison du manque de connaissances préalables et à gérer les scénarios du monde réel où les données proviennent de dynamiques multiples ou sont irrégulièrement distribuées dans le temps et l'espace. Cette thèse vise à surmonter ces limitations dans les directions suivantes : améliorer la modélisation de la dynamique basée sur les réseaux neuronaux en exploitant des modèles physiques grâce à la modélisation hybride ; étendre le pouvoir de généralisation des modèles de dynamique en apprenant les similitudes à partir de données de différentes dynamiques pour extrapoler vers des systèmes invisibles ; et gérer les données de forme libre et prédire continuellement les phénomènes dans le temps et l'espace grâce à la modélisation continue. Nous soulignons la polyvalence des techniques d'apprentissage profond, et les directions proposées montrent des promesses pour améliorer leur précision et leur puissance de généralisation, ouvrant la voie à des recherches futures dans de nouvelles applications.

Mots clés : apprentissage profond, système dynamique, phénomène physique, réseau de neurones, modélisation hybride, généralisation, adaptation, hors distribution, modélisation dynamique continue

---

Institut des Systèmes Intelligents et de Robotique
Campus Pierre et Marie Curie – Pyramide, Tour 55 – 4, place Jussieu – 75005 Paris – France