Neural Models for Learning Real World Dynamics and the Neural Dynamics of Learning

Ibrahim Ayed

Learning Real World Dynamics with Neural Models and the Neural Dynamics of Learning

par Ibrahim Ayed

Doctoral Dissertation
supervised by Patrick Gallinari

Jury :

Rémi Flamary	Rapporteur	Professeur des universités, Ecole polytechnique
Alexandre Allauzen	Rapporteur	Professeur des universités, ESPCI
Rose Yu	Examinatrice	Professeur des universités, UC San Diego
Lionel Mathelin	Examinateur	Chargé de recherche, Université Paris-Saclay
François Sausset	Examinateur	Industriel, Thales
Lionel Mathelin	Examinateur	Chargé de recherche, Université Paris-Saclay
Julien Salomon	Examinateur	Directeur de recherche, INRIA
Taraneh Sayadi	Examinatrice	Chargée de recherche, Sorbonne Université
Patrick Gallinari	Directeur de thèse	Professeur des universités, Sorbonne Université

Contents

Introduction

Those introductory considerations aim to present and motivate the general approach adopted in our work.

The work presented in this thesis was initially motivated by the discrepancy between the impressive performances of modern neural networks and the lack of applications to scientific problems for which data abounds. Focusing on evolution problems which are classically modelled through ordinary or partial differential equations (O/PDEs) naturally brought us to consider the more general problem of representing and learning such equations from raw data with neural networks. This was the inception of the first part of our work.

The point of view considered in this first part has a natural counterpart: what about the dynamics induced by the trajectories of the NN's weights during training or by the trajectories of data points within them during inference? Can they be usefully modelled? This question was the core of the second part of our work and, while theoretical tools other than O/PDEs happened to be useful in our analysis, our reasoning and intuition were fundamentally driven by considerations stemming from a dynamical viewpoint.

In the following, we start by motivating the general philosophy of our work in studying phenomena through the dynamical lens then provide more details for each of the two parts of this thesis.

0.1 From Static to Dynamic

Let us consider a certain system which is characterized by a certain number of information contained in its state X. Let us also consider the set of all states $\mathcal{S}$. Here, we will be mostly interested in how this state varies over time, meaning that there is an additional temporal variable, which we often consider in a finite time interval $[0, T]$, such that, at a given time $t \in [0, T]$, starting from the state X, the studied system is characterized by its state $X_t = T_t(X) \in \mathcal{S}$.

In other words, the transition function T_t for a $t > 0$ defines a transformation from $\mathcal{S}$ onto itself which associates for a given state the corresponding one after t units of time. In particular, this means that, for $t_1, t_2 > 0$, a state X becomes $T_{t_1}(X)$ after t_1 units of time then, after another t_2 units, becomes $T_{t_2}(T_{t_1}(X))$. Observing that, after $t_1 + t_2$ units, X also becomes $T_{t_1+t_2}(X)$, we have the fundamental property:

$$T_{t_1+t_2} = T_{t_2} \circ T_{t_1} = T_{t_1} \circ T_{t_2}$$

One might endow $\mathcal{S}$ with more structure. For example, we could consider the states belonging to a differentiable manifold, and assume that the trajectories $(X_t)_t = (T_t(X))_t$ are sufficiently regular. This is quite natural as, in most cases, the system shouldn't be able to "teleport" from one state to another. We can then also characterize those trajectories by the tangent vectors $\frac{\mathrm{d}X_t}{\mathrm{d}t}$ which belong to the tangent space of $\mathcal{S}$ at X_t by definition and, for a given trajectory, are only a function of X_t which we denote F which can thus be defined as $F(X_t) := \frac{\mathrm{d}X_t}{\mathrm{d}t}$.

In other words, under mild assumptions, F along with the starting point of the trajectory completely characterizes the trajectory of the system. Therefore, the transition equation

$$X_t = T_t(X_0)$$

can also be replaced by

$$\frac{\mathrm{d}X_t}{\mathrm{d}t} = F(X_t)$$

along with the initial condition X_0. Then T_t can be recovered as the transformation induced by the previous differential equation, which is also called *the flow* of the equation.

In particular, note that this last differential equation becomes a standard *Ordinary Differential Equation* (ODE) whenever $\mathcal{S}$ is a subset of a finite dimensional vector space, in particular a subset of $\mathbb{R}^d$; or, when $\mathcal{S}$ is an appropriate infinite dimensional space, a *Partial Differential Equation* (PDE) instead.

The following chapters will be interested in learning and analyzing various equations of this form. The objectives will be both to model real world phenomena and to study the dynamics induced by or characterizing Deep Neural Networks. This means in practice that the time variable t will designate three distinct types of temporalities:

Data temporality This is the most natural notion of time as it refers to the time of a given "real world" phenomenon, e.g. those studied in Chapters 2 and 3.

DNN inner temporality By definition, a DNN is the composition of a certain number N of successive operations. Thus, for a given input x, we can consider the output after t operations x_t. By normalizing with N and considering the limit of infinitely deep networks, we recover a continuous time in $[0, 1]$. This setting corresponds to that of Chapter 4.

Training temporality When training a DNN characterized by its parameters θ, those parameters take various values θ_t after t training updates. When the number of updates becomes higher, by normalizing with their total number, t belongs to $[0, 1]$ and we can consider the continuous formulation of training. In particular, this makes the training time variable continuous. This setting corresponds to Chapter 5.

After Chapter 1, which introduces the fundamental mathematical and algorithmic tools used in following chapters, this thesis is organized in two parts:

- The first is about the principled construction of neural models for real world phenomena and deals with the first temporality described above.
- The second is about studying the inner dynamics involved in training and using neural networks and thus deals with the second and third temporalities described above.

The two following sections can serve as reading guides to those two parts.

0.2 Building Neural Models for Real World Dynamics

Deep Neural Networks have been used successfully in order to produce giant leaps in performance for many standard learning tasks such as image classification, some speech and natural language related tasks, etc.. However, although there exists a lot of similarities between those standard deep learning domains and scientific modelling, the latter has specificities which make them extremely different from the classical playground of deep learning. Let us present a few which are of particular relevance to the work presented in this thesis.

- Many scientific models are concerned with the modeling of complex phenomena involving spatio-temporal dynamics. This shares similarities with video and motion prediction [180, 73, 94], but the underlying phenomena are usually much more complex, involving time-evolving multidimensional structures and observations at different spatio-temporal resolutions.

- Only raw observations are available and, in most scenarios, labels are not available. Many such tasks suffer from indeterminacy issues and the objectives of the modelling of a given phenomenon might depend on the specific application domain it is aimed at.
- The full state of the system itself is usually not observable so that observations only reflect some partial or indirect knowledge on the true state values [45]: for example when studying ocean's circulation, variables contained in the system's state such as surface temperature, salinity or sea surface height are observable via satellite imaging, while subsurface variables characterizing ocean dynamics are substantially much more difficult to observe.
- Extrapolation is not guaranteed since such problems are typically under-constrained, possibly leading to models with high predictive performance on the training / test sets that do not generalize well to new contexts.
- While there is an important body of existing models for any domain of interest, making them collaborate with modern ML techniques is not always natural and can sometimes be quite challenging.

Our broad goal in conducting this work has been to construct tractable and robust neural models which are usable in realistic settings, close from practice. The general philosophy has been to consider neurally parametrized equations of the form

$$\frac{\mathrm{d}X_t}{\mathrm{d}t} = F_\theta(X_t)$$

where F_θ is a Neural Network which are then appropriately learned, through an instanciation of the general algorithm presented in Section 1.3.1, and constrained in order to solve the defined task. For each, we have tried to develop generic approaches along with theoretical and empirical analyses, showing in particular that the methods we propose can be applied to different domains by evaluating them on datasets generated by diverse dynamical systems.

More specifically, we present our work in two parts:

- In Chapter 2, we present our work in learning partially observable evolution differential equations. For us, this was the start of our work with learning neural models for complex dynamical systems and was also, more generally, one of the first approaches to do such learning from raw high-dimensional data without prior knowledge of the studied system. We study the indeterminacy issue stemming from this context, propose two settings to analyze it which we evaluate with the incompressible Navier-Stokes equations. We finally test the limits of our model in dealing with non stationary more complex systems, 3D data and unseen boundary conditions.
- In Chapter 3, we delve further into constructing well-constrained neural models adapted to some specific contexts and settings. We build two types of models specific to two different settings. The first type of models aim to leverage existing expert domain knowledge on the studied system in order to incorporate it optimally in the final neural model, leading to the APHYNITY framework. For the second family of models, the goal is being able to learn better when given non-i.i.d. data, thus leading to the first generic meta-learning framework for dynamical systems, *LEADS*. For both approaches, we follow the same methodology by building them on principled theoretical guarantees then evaluating them on a diverse set of differential equations, e.g. the wave equation, reaction-diffusion equations, etc..

The work in this first part of the thesis lead to a number of presentations at international conferences such as Climate Informatics (CI 2018 and 2019) and the International Meeting on Applied Mathematics and Evolution (IMAME 2019) and to the following publications:

- *Learning Dynamical Systems from Partial Observations*, I Ayed*, E de Bézenac*, A Pajot, P Gallinari, ICASSP 2020;
- *Modelling Spatiotemporal Dynamics from Earth Observation Data with Neural Differential Equations*, I Ayed*, E de Bézenac*, A Pajot, P Gallinari, Springer's Machine Learning Journal;
- *EP-Net: Learning Cardiac Electrophysiology Models for Physiology-based Constraints in Data-Driven Predictions*, I Ayed, N Cedilnik, P Gallinari, M Sermesant, FIMH 2019, Poster;
- *EP-Net 2.0: Out-of-Domain Generalisation for Deep Learning Models of Cardiac Electrophysiology*, V Kashtanova, I Ayed, N Cedilnik, P Gallinari, M Sermesant, FIMH 2021, Poster;
- *Augmenting Incomplete Physical Models with Deep Networks for Complex Dynamics Forecasting*, Y Yin*, J Dona*, V Le Guen*, E de Bézenac*, I Ayed*, P Gallinari, N Thome, ICLR 2021, Oral presentation;
- *Learning Dynamical Systems across Environments*, Y Yin, I Ayed, E de Bézenac, P Gallinari, NeurIPS 2021, Poster.
- *Deep Learning for Model Correction in Cardiac Electrophysiological Imaging*, V Kashtanova, I Ayed, A Arrieula, M Potse, M Sermesant, MIDL 2022, Poster.

0.3 Analyzing the Dynamics of Deep Neural Networks

In Chapters 4 and 5, instead of using DNNs in order to learn real world dynamics, we are interested in the inner dynamics of DNNs themselves for the more standard learning tasks they are traditionally used for. We start by analysing the trajectories of data points within DNNs viewed as flows of ODEs. We then study the training dynamics of the DNNs used in Generative Adversarial Networks (GANs) in order to construct a theory more consistent with their empirical properties.

0.3.1 The Dynamics of Data Trajectories

Broadly speaking, Deep Neural Networks are infinitely expressive and have been shown, both theoretically and in practice, to be able to learn almost any function, no matter how complex or irregular. However, practice shows that, when used for standard tasks, they are also able to generalize quite well which seems contradictory. Our goal here is, by studying the trajectories those models induce on their input data, to characterize more precisely their hidden biases in order to understand how and why they perform so well.

Let us consider a residual network which typically has updates of the form

$$x_{t+1} = x_t + f^{\theta_t}(x_t)$$

As remarked in [80], we can see it as a first order, explicit Euler discretization with time-step $\Delta t = 1$ of the differential equation

$$\frac{\mathrm{d}x_t}{\mathrm{d}t} = f^{\theta_t}(x_t)$$

starting from an input x_0 as initial condition. By a reasoning similar to the one conducted in Section 0.1, we can generalize this point of view to any neurally parametrized transformation of a data point x_0. By definition, a DNN being a composition of transformations, if we denote $T_i^{\theta_i}$ such that $x_i = T_i^{\theta_i}(x_0)$, we can always see them from the alternative differential viewpoint. The output of a DNN then becomes the endpoint of a trajectory parametrized with a differential equation of the

form above. Note that this is also the point of view proposed in [50] which suggest creating neural architectures by parametrizing ODE solvers.

Starting from this point of view, our aim will be to study the trajectories of data-points and thus the overall transformation induced for the corresponding data distribution under ODEs and PDEs of the form above. The questions we ask are then the following

- How can we characterize, empirically and theoretically, the trajectories induced by DNNs which are successfully trained to accomplish the tasks they were designed for?
- Can we use this characterization to construct better performing, more robust models?

In our work, we show that (successfully) trained DNNs can be characterized as solutions to a variational problem inspired from the Optimal Transport problem. Our objective in the ensuing experiments is then both to show that this characterization does describe the implicit biases of DNNs as trained by practitioners but also that expliciting those biases can help us in constructing more robust and more flexible neural models for various tasks, in particular through the examples of Unsupervised Domain Translation such as solved in [295] and standard image classification tasks.

This part of our work lead to the following publications and preprints:

- *A Principle of Least Action for the Training of Neural Networks*, S Karkar*, I Ayed*, E de Bézenac*, P Gallinari, ECML 2020, Runner-up to Best student paper award;
- *CycleGAN Through the Lens of (Dynamical) Optimal Transport*, E de Bézenac*, I Ayed*, P Gallinari, ECML 2021;
- *Block-wise Training of Residual Networks via the Minimizing Movement Scheme*, Skander Karkar, Ibrahim Ayed, Emmanuel de Bézenac, Patrick Gallinari, preprint.

0.3.2 The Training Dynamics of Neural Networks

While the previous section was interested in characterizing the behavior of DNNs once trained and the class of functions they tend to approximate, here we consider the training part of the process and how it affects the learned functions. Our main goal is still the same: gain a better understanding of the implications of common practices and choices within a sound theoretical and empirical framework.

Our point of entry is similar to the one considered in the previous section. Consider the typical, simplified, gradient update of a DNN:

$$\theta_{t+1} = \theta_t - \eta \nabla \mathcal{L}(f_{\theta_t})$$

where θ_t represents the weights of the DNN after t training steps, η the learning rate and $\mathcal{L}$ is the minimized loss. While this obviously glosses over some aspects of actually used learning methods, such as stochasticity and momentum modification, it is still quite representative of how DNNs are learned and is, for our purpose, a good enough learning procedure approximation. One can then naturally consider its continuous version:

$$\frac{\mathrm{d}\theta}{\mathrm{d}t} = -\nabla \mathcal{L}(f_{\theta_t})$$

The trajectories induced by this ODE on weights $(\theta_t)_t$ translates into trajectories on the parametrized functions $(f_{\theta_t})_t$ which are the main object of our study here.

More specifically, we focus in this part of our work on the training dynamics of DNNs learned with generic GAN losses. Our objectives are the following:

- characterize the shortcomings of standard GAN analysis;

- build a sound theoretical framework which can account for common practices and empirical observations;
- confront those theoretical considerations to practice with practical experiments on standard generation tasks.

The work conducted here was accepted for publication at ICML 2022:

- *A Neural Tangent Kernel Perspective of GANs*, J-Y Franceschi*, E de Bézenac*, I Ayed*, P Gallinari, ICML 2022.

Chapter 1

A general framework for learning differential equations

This chapter presents the formal setting as well as the theoretical tools used throughout this PhD work. We start by describing the general assumptions defining our context then propose a formulation of the adjoint gradient equation which grounds most algorithms presented later in this work, along with a proof as well as corollary robustness and stability results.

1.1 General Setting and Notations

This section gives a general account of the philosophy followed in the next chapters of this work.

The objective of many learning tasks, whether they are about forecasting or classification, is to learn a function transforming its input data in a meaningful, useful way as defined by the task at hand.

The key idea here is the following: we aim to learn those transformations of input data as *operators* defined by the flow of an Ordinary Differential Equation (ODE) in finite or infinite dimensional spaces and parametrized through their differential term.

More precisely, we will study equations of the form:

$$\frac{\mathrm{d}X_t}{\mathrm{d}t} = F(X_t) \tag{1.1}$$

where:

- $X_\cdot$ is a vector-valued function defined over a finite interval $[0, T]$;
- for $t \in [0, T]$, X_t lies:
  - either in an adequate subspace of $\mathbb{R}^d$, which makes eq. 1.1 a standard ODE,
  - or is a vector-valued function of space $X_t(\cdot)$ of adequate regularity, defined over a compact subset of $\mathbb{R}^k$, where k is generally taken to be 2 or 3, thus covering many spatio-temporal PDEs of interest (such as the different variants of the Navier-Stokes equations or spatial reaction-diffusion equations);
- F is the function driving the evolution of the system.

In other words, the main idea here is to parametrize the differential equation with an F_θ, generally a Neural Network architecture, making it of the form:

$$\frac{\mathrm{d}X^\theta_t}{\mathrm{d}t} = F_\theta(X^\theta_t) \tag{1.2}$$

and then to find a θ such that the induced trajectories X_t^θ and / or the final X_T^θ transform the input in a satisfactory manner w.r.t. the task being considered. Note that the dependence on θ is only through the choice of F_θ. This is reminiscent of Optimal Control problems but here the control functions are the parameters of the learned differential term and are not time-varying.

Thus, the algorithms developped in the coming chapters of this thesis will be mainly derived of an optimisation problem of the form:

$$\begin{aligned} \underset{\theta}{minimize} \quad & \mathbb{E}_{(D_0, D_{>0}) \in \text{Dataset}} \left[\mathcal{J}(D_{>0}, X^{\theta, D_0}, F) \right] \\ \text{subject to} \quad & \frac{\mathrm{d}X_t^{\theta, D_0}}{\mathrm{d}t} = F_\theta(X_t^{\theta, D_0}), \\ & X_0^{\theta, D_0} = g_\theta(D_0) \end{aligned} \tag{1.3}$$

where:

- g_θ is an additional function which is useful, in some contexts, to construct adequate initial conditions from the initial data D_0 and will be taken equal to identity whenever such a transformation is not necessary;
- D_0 is the initial, input data in the handled task, which nature depends on the context and can be a sequence of observations or a proxy of the initial state as in Chapter 2 or an appropriate initial state[1];
- $D_{>0}$ is the data which is to be fit, thus defining the task at hand, and can be a sequence of future observations as in Chapter 2, class labels as in Chapter 4, etc.;
- $\mathcal{J}$ is the cost function to be minimized thus quantifying the objective of the learning task and can be a classification loss, a squared error w.r.t. future observations, etc..
- X^{θ, D_0} is the solution induced by parameters θ and initial data D_0.

In subsequent chapters, we will take $\mathcal{J}$ of the form:

$$\mathcal{J}(D_{>0}, X, F) = \alpha \int_0^T j(X_t, (D_{>0})_t) \mathrm{d}t + \beta j_T(X_T, D_{>0}) + \gamma R(F) \tag{1.4}$$

where, generally:

- j ensures the fit of the trajectories to the available supervision data as in Chapters 2 and 3;
- j_T allows to impose a condition over the final state X_T as in Chapter 4;
- R can be used to regularize or constrain F as in Chapter 3.

The following chapters will show that this broad formulation can encapsulate various tasks as diverse as forecasting, classification or parameter identification for physical systems. However, many aspects are still to be considered in order to transform eq. 1.3 into a practical algorithm.

Apart from those specific to each task which will be tackled later on, we will devote our attention, for the remainder of this chapter, to two general questions:

- How to deal with the cases where X_t is a function of space, which induces ODEs in infinite dimensional spaces, e.g. for many PDEs?
- How to conduct gradient learning for eq. 1.3?

Those questions are considered in the next two section: the first in Section 1.2 and the second in Section1.3.

[1] In which case g_θ can be the identity function.

1.2 On Infinite Dimensional ODEs

In our case, the question is about the settings where X_t is a function, generally of a space variable x. Indeed, in this case, even computing $F_\theta(X_t(\cdot))$ can be challenging, let alone computing gradients.

Many possibilities exist to solve this issue.

The first one is to build a parametrized function F_θ which can directly act on the function X_t. This has for example been done in [158] or [173]. It is however important to note that this point of view also involves a discretization at some point, albeit maybe not in the original input space of the X_ts, e.g. if the mapping is represented in the Fourier space the discretization can be done in the latter.

Our choice, given the equations we have considered and the nature of the data we used, is to make use of the discretization of data.

Indeed, in our datasets, X_t is often given as a tensor, e.g. in the 2-dimensional case $\{(X_t(x_{i\Delta x}, y_{j\Delta y}))_l\}_{i,j,l}$ where (i, j) are the indexes of the discretized spatial points and l indexes the vector values of the function.

This means that, when $F(X_t)$ only depends on first or second order spatial derivatives of X_t, a reasonable approximation to those derivatives can be computed with those discrete values by using, for each spatial coordinate (x, y), only neighbouring values of $X_t(x, y)$ via a finite differences scheme. In other words, we can write:

$$F(X_t)(x, y) \approx F_{discrete}(X_t(x, y), \{X_t(x', y')\}_{(x',y')\in\mathcal{N}(x,y)})$$

This means that, by appropriately including convolutions in the parametrization of F_θ, we can ensure its ability to approximate F. More precisely, whenever we have had to deal with such cases, we have chosen to parametrize F as a variant of a Convolutional Neural Network.

Let us also note that another possibility would be to manually construct a dictionary of approximations to the spatial derivatives at different orders. This has been adopted in the works of [230, 44, 168] for example. However, while doing so might usefully constrain the learning problem, as one can focus on learning the relevant dynamics, such approaches have many shortcomings:

- the experiments we show in the following chapters show that such restrictions are not necessary in our data regimes;
- such restrictions can provide the learning network with a less optimal representation of the necessary spatial information, typically worse than what it can unsupervisedly learn;
- being able to decide *a priori* which spatial derivatives are necessary and with which numerical schemes to compute them supposes a greater level of supervision than we want to, as our objective is to consider purely data-driven methods in most settings for the learning of F_θ.

1.3 Calculating the Gradient

This section introduces a general formulation of the adjoint equation. This result is the central theoretical foundation of most algorithms presented later in the thesis as it allows us to use gradient descent methods for[2] eq. 1.3.

The formulation presented here is adapted to our context of neural models. We also discuss the two possible discretizations of the adjoint equation as well as the stability and robustness of the resulting gradient.

[2]Note that several variants of it have been widely known in the literature across many domains.

1.3.1 The Adjoint State Equation

A few additional notations and assumptions In what follows, all considered functions are supposed to be twice continuously differentiable in all variables and we will use the notation $\partial_u f(u_0)$ to denote the differential of f with respect to u at u_0 *i.e.*:

$$f(u_0 + \delta u) = f(u_0) + \partial_u f(u_0) \cdot \delta u + o(\delta u)$$

By hypothesis, we consider this differential operator to be continuous.

Moreover, when X_t is a function of space $X_t(x)$, we will assume it is in the Hilbert space $\left(\mathcal{C}^2(\Omega), \langle \cdot, \cdot \rangle\right)$ where Ω is a compact subspace of $\mathbb{R}^k$ and $\langle \cdot, \cdot \rangle$ is the inner product of $L^2(\Omega)$. When the X_ts are vectors of $\mathbb{R}^d$, $\langle \cdot, \cdot \rangle$ is taken as the standard euclidean inner product of $\mathbb{R}^d$.

Statement and proof eq. 1.3 is clearly intractable in general and we will use gradient descent algorithms to solve it. In order to do so, we need to compute the gradient of the cost functional $\mathcal{J}$ under the given constraints, *i.e.* the differential of $\theta \to \mathbb{E}_D \mathcal{J}(D, X^\theta, F_\theta)$. However, this implies calculating $\dfrac{\partial X^\theta}{\partial \theta}$, which is often computationally demanding, as it implies solving $\dim(\theta)$ forward equations, the latter being high in our case. The adjoint state method avoids those costly computations by considering the Lagrangian formulation of the constrained optimization problem. A standard calculation extended here to our setting gives the expression stated in the following theorem. Here, we omit data-dependance both in the expression of the loss function and by not taking the expectation for simplification purposes: indeed in the practical case of a discrete dataset, this expectation reduces to a discrete weighted sum.

Note that while the technique of the proof is standard as it follows those for other, close, versions of the adjoint state equation [50], the result is still specific to our formulation which is, to the best of our knowledge, unique to this work.

Theorem 1.1 (*Adjoint State Equation*)

Suppose we have a loss function of the form:

$$\mathcal{J}(X, F) = \alpha \int_0^T j(X_t)\mathrm{d}t + \beta j_T(X_T) + \gamma R(F) \tag{1.5}$$

where, for ease of notation, the data dependance is omitted.

Then we have:

$$\partial_\theta \mathcal{J} \cdot \delta\theta = -\int_0^T \left\langle \lambda_t, \partial_\theta F_\theta(X_t^\theta) \cdot \delta\theta \right\rangle \mathrm{d}t - \langle \lambda_0, \partial_\theta g_\theta \cdot \delta\theta \rangle + \gamma \partial_F R(F_\theta) \cdot (\partial_\theta F_\theta \cdot \delta\theta) \tag{1.6}$$

or, written in terms of the gradient vector:

$$\nabla_\theta \mathcal{J} = \left(-\int_0^T \left\langle \lambda_t, \frac{\partial F_\theta(X_t^\theta)}{\partial \theta_i} \right\rangle \mathrm{d}t - \left\langle \lambda_0, \frac{\partial g_\theta}{\partial \theta_i} \right\rangle + \gamma \left\langle \nabla_F R(F_\theta), \frac{\partial F_\theta}{\partial \theta_i} \right\rangle \right)_i \tag{1.7}$$

Here, λ is solution of :

$$\frac{\mathrm{d}\lambda_t}{\mathrm{d}t} = A_t \cdot \lambda_t + B_t \tag{1.8}$$

solved backwards, starting with $\lambda_T = -\beta \nabla_X j_T(X_T^\theta)$, and where :

$$A_t = -(\partial_X F_\theta(X_t^\theta))^\star$$

and

$$B_t = \alpha \nabla_X j(X_t^\theta)$$

where $M^\star$ denotes the adjoint operator of the linear operator M.

Proof:

Step 1 Let us define :

$$\begin{aligned}\mathcal{L}(X,\lambda,\mu,\theta) = \mathcal{J}(X,F_\theta) + \int_0^T \left\langle \lambda_t, \frac{dX_t}{dt} - F_\theta(X_t) \right\rangle \mathrm{dt} \\ + \langle \mu, X_0 - g_\theta \rangle\end{aligned} \tag{1.9}$$

As, for any θ, X^θ satisfies the constraints by definition, we can now write :

$$\forall \theta, \lambda, \mu,\ \mathcal{L}(X^\theta,\lambda,\mu,\theta) = \mathcal{J}(X^\theta, F_\theta)$$

which gives :

$$\forall \lambda, \mu, \theta,\ \partial_\theta \mathcal{L}(X^\theta,\lambda,\mu,\theta) = \partial_\theta \mathcal{J}(X^\theta, F_\theta)$$

Let us fix θ and a variation $\delta\theta$.
Straightforward calculus gives us:

$$\partial_\theta \mathcal{J}(X^\theta, F_\theta)\cdot\delta\theta = \alpha \int_0^T \partial_X j(X_t^\theta)\cdot(\partial_\theta X_t^\theta \cdot \delta\theta)\mathrm{dt} + \beta \partial_X j_T(X_T^\theta)\cdot(\partial_\theta X_T^\theta\cdot\delta\theta) + \gamma \partial_F R(F_\theta)\cdot(\partial_\theta F_\theta \cdot \delta\theta)$$

Moreover, because $\partial_X j(X_t^\theta)$ and $\partial_X j_T(X_T^\theta)$ are continuous linear operators w.r.t. $\langle\cdot,\cdot\rangle$ Riesz representation theorem allows us to represent them with their gradients so that:

$$\partial_X j(X_t^\theta)\cdot(\partial_\theta X_t^\theta\cdot\delta\theta) = \left\langle \nabla_X j(X_t^\theta), \partial_\theta X_t^\theta \cdot \delta\theta \right\rangle$$

and

$$\partial_X j_T(X_T^\theta)\cdot(\partial_\theta X_T^\theta\cdot\delta\theta) = \left\langle \nabla_X j_T(X_T^\theta), \partial_\theta X_T^\theta \cdot \delta\theta \right\rangle$$

Step 2 We also have, by definition:

$$X^{\theta+\delta\theta} = X_t^\theta + \partial_\theta X_t^\theta \cdot \delta\theta + o(\delta\theta)$$

and, for any X and any δX:

$$F_\theta(X+\delta X) = F_\theta(X) + \partial_X F_\theta(X)\cdot\delta X + o(\delta X)$$

and:

$$F_{\theta+\delta\theta}(X) = F_\theta(X) + \partial_\theta F_\theta(X)\cdot\delta\theta + o(\delta\theta)$$

so that:

$$F_{\theta+\delta\theta}(X_t^{\theta+\delta\theta}) = F_\theta(X_t^{\theta+\delta\theta}) + \partial_\theta F_\theta(X_t^{\theta+\delta\theta})\cdot\delta\theta + o(\delta\theta)$$

Then, because F is twice continuously differentiable:

$$\begin{aligned}\partial_\theta F_\theta(X_t^{\theta+\delta\theta}) &= \partial_\theta F_\theta\left(X_t^\theta + \partial_\theta X_t^\theta\cdot\delta\theta + o(\delta\theta)\right) \\ &= \partial_\theta F_\theta(X_t^\theta) + \partial_X\partial_\theta F_\theta(X_t^\theta)\cdot\partial_\theta X_t^\theta\cdot\delta\theta \\ &\quad + o(\delta\theta)\end{aligned}$$

and:

$$\begin{aligned}F_\theta(X_t^{\theta+\delta\theta}) &= F_\theta\left(X_t^\theta + \partial_\theta X_t^\theta\cdot\delta\theta + o(\delta\theta)\right) \\ &= F_\theta(X_t^\theta) + \partial_X F_\theta(X_t^\theta)\cdot\partial_\theta X_t^\theta\cdot\delta\theta + o(\delta\theta)\end{aligned}$$

Moreover, as all differential operators below are continuous by hypothesis, we have that:

$$\|(\partial_X\partial_\theta F_\theta(X_t^\theta)\cdot\partial_\theta X_t^\theta\cdot\delta\theta)\cdot\delta\theta\| \leq \|\partial_X\partial_\theta F_\theta(X_t^\theta)\|\ \|\partial_\theta X_t^\theta\|\ \|\delta\theta\|^2$$

so that:

$$\begin{aligned}&F_{\theta+\delta\theta}(X_t^{\theta+\delta\theta}) \\ &\quad = F_\theta(X_t^\theta) + \left(\partial_X F_\theta(X_t^\theta)\cdot\partial_\theta X_t^\theta + \partial_\theta F_\theta(X_t^\theta)\right)\cdot\delta\theta + o(\delta\theta)\end{aligned}$$

Step 3 We now calculate the derivative of $\mathcal{L}$:

$$\partial_\theta \mathcal{L} \cdot \delta\theta = \partial_\theta \mathcal{J}(X^\theta, F_\theta) \cdot \delta\theta + \int_0^T \left\langle \lambda_t, \partial_\theta \frac{\mathrm{d}X_t^\theta}{\mathrm{d}t} \cdot \delta\theta - \partial_X F_\theta(X_t^\theta) \cdot (\partial_\theta X_t^\theta \cdot \delta\theta) - \partial_\theta F_\theta(X_t^\theta) \cdot \delta\theta \right\rangle \mathrm{d}t + \left\langle \mu, \partial_\theta X_0^\theta - \partial_\theta g_\theta \right\rangle$$

By the Schwarz theorem, as X is twice continuously differentiable in (t, θ), we have that $\frac{\partial}{\partial \theta_i} \frac{\mathrm{d}X_t^\theta}{\mathrm{d}t} = \frac{\mathrm{d}}{\mathrm{d}t} \frac{\partial X_t^\theta}{\partial \theta_i}$ for all i so that $\partial_\theta \frac{\mathrm{d}X_t^\theta}{\mathrm{d}t} \cdot \delta\theta = \frac{\mathrm{d}}{\mathrm{d}t} \partial_\theta X_t^\theta \cdot \delta\theta$

Integrating by parts, we get:

$$\int_0^T \left\langle \lambda_t, \partial_\theta \frac{\mathrm{d}X_t^\theta}{\mathrm{d}t} \cdot \delta\theta \right\rangle \mathrm{d}t = \left\langle \lambda_T, \partial_\theta X_T^\theta \cdot \delta\theta \right\rangle - \left\langle \lambda_0, \partial_\theta X_0^\theta \cdot \delta\theta \right\rangle - \int_0^T \left\langle \frac{\mathrm{d}\lambda_t}{\mathrm{d}t}, \partial_\theta X_t^\theta \cdot \delta\theta \right\rangle \mathrm{d}t$$

Putting all this together and arranging it, we get:

$$\begin{aligned} \partial_\theta \mathcal{L} \cdot \delta\theta = &\int_0^T \left\langle \partial_\theta X_t^\theta \cdot \delta\theta, -\frac{\mathrm{d}\lambda_t}{\mathrm{d}t} - \partial_X F_\theta(X_t^\theta)^\star \lambda_t + \alpha \nabla_X j(X_t^\theta) \right\rangle \mathrm{d}t \\ &+ \left\langle \partial_\theta X_T^\theta \cdot \delta\theta, \lambda_T + \beta \nabla_X j_T(X_T^\theta) \right\rangle + \left\langle \mu - \lambda_0, \partial_\theta X_0^\theta \right\rangle \\ &- \int_0^T \left\langle \lambda_t, \partial_\theta F_\theta(X_t^\theta) \cdot \delta\theta \right\rangle \mathrm{d}t - \langle \mu, \partial_\theta g_\theta \rangle + \gamma \partial_F R(F_\theta) \cdot (\partial_\theta F_\theta \cdot \delta\theta) \end{aligned}$$

Note that the adjoint $(\partial_X F_\theta(X_t^\theta))^\star$ does exist because, by the regularity of F_θ, this is a bounded linear operator.

Step 4 We can now define:

$$A_t = -(\partial_X F_\theta(X_t^\theta))^\star$$

and

$$B_t = \alpha \nabla_X j(X_t^\theta)$$

and, recalling that λ can be freely chosen, impose that λ is solution of:

$$\frac{\mathrm{d}\lambda_t}{\mathrm{d}t} = A_t \lambda_t + B_t$$

with final condition $\lambda_T = -\beta \nabla_X j_T(X_T^\theta)$. We also choose $\mu = \lambda_0$ so that, finally, we have:

$$\partial_\theta \mathcal{L} \cdot \delta\theta = -\int_0^T \left\langle \lambda_t, \partial_\theta F_\theta(X_t^\theta) \cdot \delta\theta \right\rangle \mathrm{d}t - \langle \lambda_0, \partial_\theta g_\theta \cdot \delta\theta \rangle + \gamma \partial_F R(F_\theta) \cdot (\partial_\theta F_\theta \cdot \delta\theta)$$

This means that we can write:

$$\nabla_\theta \mathcal{J} = \left(-\int_0^T \left\langle \lambda_t, \frac{\partial F_\theta(X_t^\theta)}{\partial \theta_i} \right\rangle \mathrm{d}t - \left\langle \lambda_0, \frac{\partial g_\theta}{\partial \theta_i} \right\rangle + \gamma \partial_F R(F_\theta) \cdot \frac{\partial F_\theta}{\partial \theta_i} \right)_i$$

which concludes the proof. □

1.3.2 Properties of the Adjoint Equation

In this section, we derive some properties of the adjoint equation of Theorem 1.1. More specifically, we show that any perturbation of the forward pass solving for X yields an error for the gradient which is uniformly bounded by the magnitude of the perturbation.

Below, we keep the same notations as in the statement and proof of Theorem 1.1.

Let us start by stating a version of the Gronwall lemma:

Lemma 1.1 (*Gronwall*)

Let u be solution to:

$$\frac{\mathrm{d}u_t}{\mathrm{d}t} = \alpha_t u_t + \beta_t$$

with $u_T = 0$. Then:

$$\|u_t\| \leq \int_t^T \|\beta_s\| ds + \int_t^T \int_s^T \|\beta_r\| dr \|\alpha_s\| \exp\left(\int_t^s \|\alpha_r\| dr\right) ds$$

We can then prove a first stability result for the gradient calculated by the adjoint method:

Proposition 1.1 (*Stability of the gradient*)

Under the hypothesis in theorem 1.1, λ is defined and bounded over $[0, T]$. Consequently, $\nabla_\theta \mathcal{J}$ is also well-defined and bounded.

Proof: Using the lemma above, we have, using the same notations as in Theorem 1.1, that:

$$\forall t,\ \|\lambda_t\| \leq \int_t^T \|B_s\| ds + \int_t^T \int_s^T \|B_r\| dr \|A_s\| \exp\left(\int_t^s \|A_r\| dr\right) ds$$

Moreover, by hypothesis, $t \to \partial_X F_\theta(X_t^\theta)$ and $t \to \nabla_X j(X_t^\theta)$ are continuous over $[0, T]$ is compact so that A and B have bounded norms over this closed interval. Combining this fact with the inequality above gives us the boundedness of λ. Finally, g_θ and $\partial_\theta F_\theta$ are continuous as well so that $\nabla_\theta \mathcal{J}$ is also bounded. □

This is a minimal requirement for the descent algorithm we use to be meaningful: The solution of the adjoint equation has to be well-defined and the gradient has to be stable enough.

In the following, we denote $\|f\|_\infty = \sup_{(t,x)\in[0,T]\times\Omega} \|f(t,x)\|$ for any function defined over $[0, T] \times \Omega$. This notation is also used when considering a tuple-valued function by taking the maximum over all scalars of the tuple.

Consider a perturbed solution $\widetilde{X}$ of the equation driven by F_θ. The perturbation can typically be due to numerical approximation errors. While this perturbation necessarily entails a modification of the underlying gradient, the result below shows that this modification is reasonable in our setting.

Proposition 1.2 (*Robustness of the gradient*)

Consider a compact neighbourhood V of X. Then, for a perturbed solution $\widetilde{X}$ in V of the forward equation, the corresponding perturbed adjoint $\widetilde{\lambda}$ as well as the perturbed gradient $\widetilde{\nabla_\theta \mathcal{J}}$ verify:

$$\|\widetilde{\lambda} - \lambda\|_\infty \leq M \|\widetilde{X} - X\|_\infty$$

and

$$\|\nabla_\theta \mathcal{J} - \widetilde{\nabla_\theta \mathcal{J}}\|_\infty \leq M' \|\widetilde{X} - X\|_\infty$$

where M and M' do not depend on $\widetilde{X}$.

Proof:

Notations In the following, we will omit θ in the notations as it is fixed and doesn't play a role here. Define $\widetilde{\lambda}$ which is solution to the perturbed adjoint equation:

$$\frac{\mathrm{d}\lambda_t}{\mathrm{d}t} = \widetilde{A}_t \cdot \lambda_t + \widetilde{B}_t$$

and $\widetilde{\lambda}_T = -\beta \nabla_X j_T(\widetilde{X}_T)$.
Moreover, define $\widetilde{\lambda}^{mod}$ which verifies the same equation as $\widetilde{\lambda}$ with $\widetilde{\lambda}^{mod}_T = -\beta \nabla_X j_T(X_T) = \lambda_T$.
Let us also define:

$$\epsilon^{(1)} = \widetilde{\lambda} - \widetilde{\lambda}^{mod}$$

and

$$\epsilon^{(2)} = \widetilde{\lambda}^{mod} - \lambda$$

and

$$\epsilon = \widetilde{\lambda} - \lambda$$

Perturbation of the adjoint We have:

$$\frac{\mathrm{d}\epsilon^{(1)}_t}{\mathrm{d}t} = \widetilde{A}_t \epsilon^{(1)}_t$$

which means that:

$$\|\epsilon^{(1)}\|_\infty \leq \beta \|\nabla_X j_T(X_T) - \nabla_X j_T(\widetilde{X}_T)\| \exp \|\widetilde{A}\|_\infty T$$

and, because j_T is sufficiently regular by hypothesis, we can put $M_1 = \sup_{H \in V} \|\nabla^2_X j_T(H_T)\|$ so that:

$$\|\epsilon^{(1)}\|_\infty \leq \beta M_1 \exp \|\widetilde{A}\|_\infty T \|X_T - \widetilde{X}_T\| \leq \beta M_1 \exp \|\widetilde{A}\|_\infty T \|X - \widetilde{X}\|_\infty$$

On the other hand, we have:

$$\frac{\mathrm{d}\epsilon^{(2)}_t}{\mathrm{d}t} = \widetilde{A}_t \epsilon^{(2)}_t + (\widetilde{A}_t - A_t)\lambda_t + \widetilde{B}_t - B_t$$

Moreover:

$$\forall t, \|\widetilde{B}_t - B_t\| = \|\nabla_X j(\widetilde{X}_t) - \nabla_X j(X_t)\|$$

As a function of t, the latter is, by hypothesis, continuous over the closed interval $[0, T]$ so that there must be an s such that:

$$\forall t, \|\widetilde{B}_t - B_t\| \leq \|\nabla_X j(\widetilde{X}_s) - \nabla_X j(X_s)\|$$

Then, similarly to before, taking $M_2 = \sup_{H \in V, t \in [0,T]} \|\nabla^2_X j(H_t)\|$, we have:

$$\|\widetilde{B}_t - B_t\| \leq M_2 \|\widetilde{X} - X\|_\infty$$

Similarly, because F is also twice continuously differentiable on X, there is M_3 such that:

$$\|\widetilde{A}_t - A_t\| \leq M_3 \|\widetilde{X} - X\|_\infty$$

Thus, taking $\beta_t = (\widetilde{A}_t - A_t)\lambda_t + \widetilde{B}_t - B_t$ and combining the inequalities above and the fact that λ is bounded, we have:

$$\|\beta_t\| \leq (M_3 \|\lambda\|_\infty + M_2) \|\widetilde{X} - X\|_\infty$$

Moreover, taking $\alpha_t = \widetilde{A}_t$, and recalling that $\epsilon_T^{(2)} = \widetilde{\lambda}_T^{mod} - \lambda_T = 0$, we can then apply the Grönwall lemma and deduce:

$$\|\epsilon^{(2)}\|_\infty \leq T(M_3\|\lambda\|_\infty + M_2)(1 + T\|\widetilde{A}\|_\infty \exp T\|\widetilde{A}\|_\infty)\|\widetilde{X} - X\|_\infty$$

Finally, regrouping all terms:

$$\|\epsilon\|_\infty \leq L\|\widetilde{X} - X\|_\infty$$

where:

$$L = \beta M_1 \exp\|\widetilde{A}\|_\infty T + T\left(M_3\|\lambda\|_\infty + M_2\right)\left(1 + T\|\widetilde{A}\|_\infty \exp T\|\widetilde{A}\|_\infty\right)$$

Perturbation of the gradient We have:

$$\|\nabla_\theta \mathcal{J} - \widetilde{\nabla_\theta \mathcal{J}}\|_\infty \leq \max_i \int_0^T \left|\left\langle \lambda_t, \frac{\partial F_\theta(X_t)}{\partial \theta_i}\right\rangle - \left\langle \widetilde{\lambda}_t, \frac{\partial F_\theta(\widetilde{X}_t)}{\partial \theta_i}\right\rangle\right| \mathrm{d}t + \left|\left\langle \lambda_0, \frac{\partial g_\theta}{\partial \theta_i}\right\rangle - \left\langle \widetilde{\lambda}_0, \frac{\partial g_\theta}{\partial \theta_i}\right\rangle\right|$$

Regarding the first term:

$$\int_0^T \left|\left\langle \lambda_t, \frac{\partial F_\theta(X_t)}{\partial \theta_i}\right\rangle - \left\langle \widetilde{\lambda}_t, \frac{\partial F_\theta(\widetilde{X}_t)}{\partial \theta_i}\right\rangle\right| \mathrm{d}t \leq \int_0^T \left|\left\langle \lambda_t, \frac{\partial F_\theta(X_t)}{\partial \theta_i} - \frac{\partial F_\theta(\widetilde{X}_t)}{\partial \theta_i}\right\rangle + \left\langle \lambda_t - \widetilde{\lambda}_t, \frac{\partial F_\theta(\widetilde{X}_t)}{\partial \theta_i}\right\rangle\right| \mathrm{d}t$$

then, using the regularity of F, the Cauchy-Schwarz inequality, the boundedness of λ and the previous result for the perturbation on the adjoint, there exists M_4 and M_5 such that we have:

$$\int_0^T \left|\left\langle \lambda_t, \frac{\partial F_\theta(X_t)}{\partial \theta_i}\right\rangle - \left\langle \widetilde{\lambda}_t, \frac{\partial F_\theta(\widetilde{X}_t)}{\partial \theta_i}\right\rangle\right| \mathrm{d}t \leq \left(TM_4\|\lambda\|_\infty + M_5 L\right)\|X - \widetilde{X}\|$$

For the second term, we simply use the Cauchy-Schwarz inequality along with the previous result on the adjoint:

$$\left|\left\langle \lambda_0 - \widetilde{\lambda}_0, \frac{\partial g_\theta}{\partial \theta_i}\right\rangle\right| \leq L\left\|\frac{\partial g_\theta}{\partial \theta_i}\right\|\|X - \widetilde{X}\|$$

Combining the two inequalities gives us the desired result. □

In other words, the gradient calculated through the adjoint method is robust in the sense that its perturbation is linearly controlled by the magnitude of the perturbation.

Such perturbations have two generic sources: noise in the data, e.g. when relying on measurements, and numerical errors when solving in time the forward equation.

However, it is also important to note that those results are essentially qualitative and that the constants M, M' which are derived in the proof could have high values, depending essentially on the regularity of the different functions defining the cost and the exact parametric family of F.

1.3.3 Approximate Solutions

Theorem 1.1 gives us a way to calculate, for a given value of θ, the gradient of the constrained problem studied here which allows to construct gradient descent algorithms for it. However, solving the forward and backward equations eq. 1.1 and eq. 1.8 isn't generally straightforward. They do not yield a closed form solution and approximations are necessary at some point in order to obtain a tractable algorithm.

There are essentially two different ways to tackle this problem [107]: the *differentiate-then-discretize* approach, and the *discretize-then-differentiate* approach[3]. Note that both approaches converge to the result of Theorem 1.1 when the discretization step goes to zero.

[3]The differentiate-then-discretize method is also often referred to as the *continuous adjoint method*, and the *discretize-then-differentiate* approach as the *discrete adjoint method* [245].

Differentiate-then-discretize In this approach, both equations, forward and backward, are discretized using numerical schemes. More specifically:

1. forward and backward equations are derived;
2. adequate numerical schemes for both equations are chosen and used to solve them;
3. the gradient is calculated.

There are advantages to this method:

- Numerical schemes can be chosen independently for the two equations, taking their structure into account if there is a need and tailoring numerical precision according to needs.
- Any numerical scheme can be chosen, including non-differentiable ones.
- As advocated in [50], it is even possible to make those choices automatically, using available solvers.
- Existing legacy solvers can be used without any modification to the forward equation solver.

The main problem with this approach is its lack of stability and robustness. Indeed, regarding the methodology adopted in [50], there are many studied such as [279, 90] which show that learning those models is not easy. More generally, this shortcoming is actually well-known and goes beyond neural parametric families. Indeed, it can yield inconsistent gradients of the cost functional $\mathcal{J}$ and the discretization of the adjoint equation depends on the studied problem. It therefore must be carefully selected and tuned [40] which cannot be done with black-box solvers and broad parametric families. Moreover, with parametric families F_θ as expressive as NNs which are used in this work, the corresponding equations can greatly vary which makes it impossible to make principled and sound choices.

Discretize-then-differentiate In this approach, a differentiable solver for the forward equations is used. More specifically:

1. a differentiable solver for the forward equation is chosen;
2. the solution X^θ is computed and thus has a differentiable relation to θ through the solver;
3. the gradient of $\mathcal{J}(X^\theta)$ can then be computed by using automatic differentiation through the (differentiable) solver.

In other words, the optimization problem becomes of the form:

$$\begin{aligned} \underset{\theta}{minimize} \quad & \mathbb{E}_{(D_0, D_{>0}) \in \text{Dataset}} \left[\mathcal{J}(D_{>0}, X^{\theta, D_0}, F) \right] \\ \text{subject to} \quad & \forall t, X_t^{\theta, D_0} = \text{DiffSolve}(F_\theta, X_0^{\theta, D_0}, t), \\ & X_0^{\theta, D_0} = g_\theta(D_0) \end{aligned}$$

The shortcomings of this approach are obvious:

- Some forward equations might not be possible to solve with differentiable schemes.
- It isn't obvious to make a principled choice for the solver, especially when the parametric family of equations is large.

- It isn't always possible to use existing legacy solvers, e.g. when gradients cannot bederived easily from them. This might be a problem for some applications, *e.g.* climate modelling.

However, it also has many strengths:

- It is quite straightforward to implement with existing tools, especially for NNs.
- When F_θ is taken in an expressive enough parametric family, it can compensate for a lacking numerical scheme (a striking example is shown in Chapter 3 with the APHYNITY model when used with exact knowledge of the physics).
- Whenever the forward equation is well-approximated, the computed gradients are stable and consistent as there is no need for further discretization. We have also shown through Proposition 1.2 that those gradients are also robust to small perturbations of the computed forward solution X.

Our choice Let us recall that, while the two methods are consistent and while both converge to the equations derived in Theorem 1.1 when the discretization steps tend to zero, they do not yield the same results in general. In other words, discretization and differentiation are not commutative. Building on preliminary experiments as well as on existing litterature, our choice in the remainder of this work was thus to choose the second approach. For all the systems studied in the following, it proved more stable and the fact that we were limited to differentiable solvers was never limiting.

PART A

Learning Differential Equations with Neural Networks

This part deals with the construction of neural models for the study of real world phenomena. In its first chapter, we describe a general framework for the setting where the state describing the evolution of the system is only partially observable. In the second, we propose two general models which, by adequately constraining the learning problem, are able to handle the case where prior knowledge is available for the first and where the data is heterogeneous for the second.

Our main contributions are thus the following:

- a general framework to learn neurally parametrized differential models in the partially observable setting from raw observations;
- the APHYNITY model which allows to construct adaptive hybrid models merging existing expert knowledge with a minimal data-driven, neurally parametrized augmentation;
- the *LEADS* model which learns jointly from non-i.i.d. data, generated by distinct dynamics;
- theoretical analysis whenever possible of the proposed approaches and related questions and extensive experimentation with various dynamical systems.

Chapter 2

Learning Partially Observable Differential Equations

This chapter considers the problem of learning neurally parametrized differential equations from data representing raw partial observations. Its content covers that of the following publications[1]:

- *Learning Dynamical Systems from Partial Observations*, I Ayed*, E de Bézenac*, A Pajot, P Gallinari, ICASSP 2020;
- *Modelling Spatiotemporal Dynamics from Earth Observation Data with Neural Differential Equations*, I Ayed*, E de Bézenac*, A Pajot, P Gallinari, Springer's Machine Learning Journal;
- *EP-Net: Learning Cardiac Electrophysiology Models for Physiology-based Constraints in Data-Driven Predictions*, I Ayed, N Cedilnik, P Gallinari, M Sermesant, FIMH 2019, Poster;
- *EP-Net 2.0: Out-of-Domain Generalisation for Deep Learning Models of Cardiac Electrophysiology*, V Kashtanova, I Ayed, N Cedilnik, P Gallinari, M Sermesant, FIMH 2021, Poster.

2.1 Introduction

We consider the task of learning spatio-temporal dynamics when observations are supposed to represent partial information of the underlying system state and the dynamics governing the state evolution are unknown. This is the general situation in most scientific modelling problems. We assume that the unknown dynamics obey a set of differential equations with general form:

$$\frac{\mathrm{d}X_t}{\mathrm{d}t} = F(X_t) \tag{2.1}$$

Where X is the system state, considered here as a spatio-temporal vector field. Its value at time t is denoted $X_t(x) \in \mathbb{R}^d$.

Many phenomena studied in physics, computer vision, biology [186], geoscience [85], finance [278], etc. obey a general equation of this form. For this reason, an extensive effort has been put into crafting such equations, solving and overall gaining a better understanding of them. Generally, when F is known, predicting and analyzing the dynamics of the system often amounts to using an adequate numerical solver.

However, for many practical problems, F may not be fully known, e.g. the relations between the components of the state can be difficult to establish from first principles or even by phenomenological modelling heuristics. With the availability of very large amounts of data captured via diverse sensors,

[1] In particular, while the field has known many advancements since, the related work and cited papers as well as baseline models in the experiments reflect its state at the time of their publication.

of substantially improved computational infrastructure and techniques and the recent advances of statistical methods,a data-driven paradigm for modeling dynamical systems where the state dynamics is automatically discovered based on the observations has thus emerged and gained traction those last few years [66, 5]. This is usually performed by considering an adequate class of admissible functions $\{F_\theta\}$ parametrized by θ, and looking for a θ such that the solution X^θ of :

$$\frac{\mathrm{d}X_t}{\mathrm{d}t} = F_\theta(X_t) \tag{2.2}$$

fits the measured data.

However, there is a major drawback to this approach: for most real-world applications, the variables describing the system are not fully visible to external sensors [46], *e.g.* when studying ocean's circulation, variables contained in the system's state such as surface temperature or salinity are observable via satellite imaging, while others subsurface variables are substantially more difficult or costly to observe. In this case, the state is said to be *partially observable*. In other words, for most real-world applications, the X variables describing the system are not fully visible to sensors and this is the setting considered in this Chapter.

Instead, we assume that sequences of partial observations $\{(Y_i)_i\}$ are acquired on a regular spatial grid, thus providing incomplete information about the unknown underlying process represented with (full) state variables $\{(X_i)_i\}$. We make the natural hypothesis that incomplete observations Y_t can be computed from the corresponding unknown state X_t. In order to model the unknown spatio-temporal dynamics, we will consider a class of admissible functions F_θ implemented by deep convolutional neural networks in order to take into account complex spatial dependencies and multi-scale behavior. Our objective is then to learn parameters θ capturing the dynamics of the system's state from raw data and then perform long-term forecasts without prior knowledge of the system.

More precisely, we start by constructing a general optimization program allowing to learn neural differential models of partially observable systems (Section 2.3). We then propose two idealized settings in order to gain a better understanding of the non-observed part of the reconstructed state (Section 2.4). Experiments on the incompressible Navier-Stockes equations allow us to evaluate our proposed framework and dive deeper into understanding its properties (Section 2.5). The last two sections intend to go further by studying the limits of our modelling methodology: by applying it to a more realistic context where the hypothesis underlying its formulation are not verified (Section 2.6), by instanciating it on 3D data and by testing its extrapolation ability w.r.t. new, unseen boundary conditions (Section 2.7).

To summarize, our main contributions are the following:

- We propose a framework for learning complex spatio-temporal dynamics in the challenging partially observable, large size observation spaces context.

- We introduce two settings: the first relies only on observations while the second assumes that a full initial state is available for each trajectory. For both, we analyze the learned state representaions w.r.t. the canonical interpretable physical states.

- We demonstrate its performances and test its limits on three problems: the incompressible Navier Stokes equations, the challenging and realistic Glorys2v4 dataset of Sea Surface Temperatures and data from the Electro-Physiological modeeling of the heart.

Overall, the most promising aspect of our contributions is the fact that, when parametrized and trained correctly, Neural Networks are able to learn realistic models of realistic dynamics with reasonable amounts of data, even in the partially observable setting and without any prior knowledge while exhibiting interesting robustness and extrapolation abilities.

2.2 Related Work

The idea of leveraging ML methods in order to learn data-driven models of dynamical systems is not new: [200] gives a thorough introduction to the closely related field of Nonlinear System Identification while [66, 5] gives an early example of such endeavours. More recently, [230] and [293] used sparse regression on a dictionary of differential terms to recover the underlying PDE. [81] construct a bilinear network and use an architecture similar to finite difference schemes to learn fully observed dynamical systems. [169] also carefully tailor the neural network architecture, based on the discretization of different terms of the underlying PDE. In [219], they propose recovering the coefficients of the differential terms by deriving the kernel of a Gaussian Process from a linearized form of the PDE. Developping the so-called "Physically Informed Neural Networks" (known as PINNs) formalism, [218] propose to learn data-driven solvers where the loss ensures that a known or parametrized differential equations is verified while data representing initial and boundary conditions are fit by the learned solutions, parametrized as Neural Networks. More generally, [276] propose a broader survey of ML in physics-based modeling. Overall, in all of those approaches, we can generally see that either the form of the PDE or the variable dependency are supposed to be known and that the context is often one where the state is fully observed.

Related to our work, there is the field of data assimilation [170, 47], where one is interested in using (partial) observations, in conjunction with the evolution model, supposed known, in order to retrieve the canonical state. Typically, our constrained optimisation problem is similar to the one posed in classical 4D-Var [47], where the constraint is the evolution equation of the state. Although there have been work in data assimilation community where the evolution equation is only partially known and some unknown forcing terms are estimated from the data [35], our work takes a more data-driven approach, where we make no assumptions and use no prior knowledge of the underlying evolution equation.

Recently, other approaches, combining ideas from data-assimilation and machine learning attempt to tackle the problem of learning the system from partial observations. [201] learn an LSTM to forecast Lorenz-63 system when only sparse acquisitions in time of the full state are available. However, these methods evaluate themselves solely on the observed data, and do not consider the hidden states that are predicted by the model. A more hybrid example is [69], corresponding to our PKnI baseline, where they propose to learn a forecasting system in the partially observable case, where part of the differential equation is known, and the other is approximated using data, which allows the network's hidden state to acquir a structure similar to that of the true hidden state.

There are quite a few modelling applications for ML. Recent developments include [134] which perform change detection for satellite image time series using autoencoders, extreme weather event detection considered in [215] and Convolutional LSTMs used in [241] for nowcasting. [136] is one of the first papers constraining neural networks to be consistent with physics and using prior physical knowledge for a prediction task, the application being lake temperature modeling. [265] makes use of a super resolution convolutional neural network with multi-scale input channels for statistical downscaling of climate variables. [208] also tackle the forecasting and assimilation of geophysical fields and consider sea surface temperature as an application.

2.3 Learning the Dynamics of Partially Observable Systems

In this section, we formalize the setting we will be working on and cast it into an optimization problem within the framework developed in the previous chapter.

2.3.1 Partially Observable Systems: Hypothesis, General Setting and Notations

The task considered in this chapter is about learning the dynamics of a given dynamical phenomenon while assuming access only to partial measurements of the system's state.

More formally, our hypothesis are the following:

- We have a dataset $\{Y^{(i)}\}_i = 1...N$ corresponding to N sequences of observations. Here, $Y_l^{(i)}$ denote the available measurement at time l from the $i-$th sample sequence, and $Y_{l:m}^{(i)}$ the sub-sequence of observations from time l to m. For simplicity, the superscript $^{(i)}$ may be omitted.

- There exists a stationary, deterministic and differentiable function $\mathcal{H}$ and a spatial vector field $X_\cdot$ satisfying equation eq. 2.1 such that $\mathcal{H}(X_\cdot) = Y_\cdot$.

Note that the state in this chapter is a spatial vector field which we study over a compact set $[0, T] \times \Omega$, meaning that for $(t, x) \in [0, T] \times \Omega$ we have that $X_t(x) \in \mathbb{R}^d$. Additionally, let us also note that while we assume the existence of such a state function generating the observations, nothing is assumed regarding its uniqueness.

Moreover, $\mathcal{H}$ represents the loss of information between the state X describing the evolution system and observations Y. In particular, in our experiments, $\mathcal{H}$ will be a point-wise projection operator but all general considerations would remain true in other contexts.

A first question is then whether it is always possible to reconstruct a state function X from the observations, for any function $\mathcal{H}$. This is clearly not the case in general: if $\mathcal{H}$ has a constant value for all inputs for example. However, the Takens theorem, see [255] for the original statement and [227] for a more recent version, states[2] that, for a dense set of observation functions $\mathcal{H}$, there exists an integer K such that $Y_{t-K+1:t}$ can be transformed into X_t. In the following, we suppose that $\mathcal{H}$ is such a function. In other words, there exists K and a function g such that $X_t = g(Y_{t-K+1:t})$. In practice, K is treated as a hyper-parameter of our models.

Another question regards the uniqueness of the state X. Indeed, $\mathcal{H}$ represents a loss of information and is not injective which in particular means that it is not invertible. This implies that there could exist many state representations which induce the same observations Y.[3] Our experiments are performed on simulated data, providing access to all the variables of the system, so that we will denote by **canonical state**, the true state of the physical model *i.e.* the one used in the simulation, which is not available for training in our context. Having access to this ground truth state will allow us to measure how much of the ground truth state information has been learned by our model. Of course, this analysis is performed here for the evaluation of the model and to understand its properties. It is mostly not feasible in real situations where we have no access to state variables.

2.3.2 Optimization Problem

We want to learn a state representation and its evolution dynamics from sequences of partial observations. A natural formulation as an optimization problem is then following:

$$\begin{aligned} \underset{\theta}{minimize} \quad & \mathbb{E}_{(Y_{-k+1:0}, Y_{1:T}) \in \text{Dataset}} \left[\mathcal{J}(Y_{1:T}, \mathcal{H}(X^\theta)_{1:T}) \right] \\ \text{subject to} \quad & \frac{\mathrm{d}X_t^\theta}{\mathrm{d}t} = F_\theta(X_t^\theta), \\ & X_0^\theta = g_\theta(Y_{-k+1:0}) \end{aligned} \tag{2.3}$$

[2] We have voluntarily stated the theorem in loose terms as there are many versions of it in many different settings and the involved technicalities are beyond the scope of the heuristical argument we present here.

[3] We give a more rigorous statement regarding the more particular situation of interest in this work in Section 2.4.

where we take:

$$\mathcal{J}((Y_i)_i, (\widetilde{Y}_i)_i) = \sum_{l=1}^{T} \|Y_i - \widetilde{Y}_i\|_{L^2}^2 \tag{2.4}$$

where the L^2 norm is taken over the compact spatial domain $\Omega \subset \mathbb{R}^d$ over which the vector fields are defined here and where the dataset is a set of the form $\{(Y_{-k+1}^{(i)}, ..., Y_0^{(i)}, ..., Y_T^{(i)})\}_i$, where all observations $Y^{(i)}$ are supposed to be generated through the same underlying dynamical system, with different initial conditions. g_θ is a function to be learned and used for predicting an initial state X_0 from past observations $Y_{-k+1:0}$. This last dependency is omitted in the notation for ease of reading but it should be emphasized that it exists nonetheless in general, except in trivial cases.

The difficulty and originality in our context stems from the combination of multiple factors: the incomplete information setting, the complexity of the considered dynamics and the high-dimensional, raw spatial data provided as observations. Classical non-linear system identification do not handle this type of data [268]. Closer to us, neural differential equation solvers, e.g. [243, 217] or [50], all assume **having access to the full states** and not only to incomplete observations as we do here. [243, 217] furthermore assume that the form of the differential equation is known. Solving the problem in Equation eq. 2.3 in this context requires a specific parametrization of the model: we choose F_θ and g_θ to be deep convolutional networks, which allows us to learn complex spatial differential operators from data like advection or diffusion terms present in Navier-Stokes [231] unsupervisedly. The time evolution is then obtained by solving the forward equation parametrized by F_θ.

While this model can (and will) be discretized, it is continuous in nature and can thus be related to the equations used in standard physical models. Here, θ is the variable controlling the learning and contains the parameters for both F and g.

2.3.3 Training and Inference Algorithms

The formulation clearly fits into the framework defined in Chapter 1 where we have taken:

- D_0 as the sequence of past observations $Y_{-k+1:0}$ and $D > 0$ as future observations $Y_{1:T}$;
- $\mathcal{J}$ as in eq. 1.4 with $\alpha = 1$, $= \gamma = 0$ and j as the squared L^2 loss over Ω.

Thus, the optimization problem defined here can be solved using gradient descent methods as discussed previously. In our case, F_θ and g_θ are parameterized as neural networks but other parametric families without any changes to the algorithm, the only constraint being differentiability with respect to θ.

The (general) *training* algorithm adopted here is Algorithm 1 below.

For *inference*, the parameters being learnt and fixed, we simply calculate $X_0 = g_\theta(Y_{-k+1:0})$ then use it as an initial condition to solve the equation parametrized by F_θ which gives us X_t for any t.

2.4 Analyzing the Hidden Dynamics

In this section, we show that the optimization problem defined above is ill-posed and admits non-canonical state representations as optimal solutions. We then outline two settings where we analyze the induced state representation.

2.4.1 Learning An Ill-Posed Problem

For all of the following, we will consider the more specific (but still broad) situation where we take Y and X to be vector-valued spatio-temporal fields with values respectively in $\mathbb{R}^l$ and $\mathbb{R}^d$ where $l \leq d$,

Algorithm 1: Training Procedure in (in the experiments)

Input: Training samples $\{(Y_{-k+1:0}), Y_{1:T}\}$.
Guess initial parameters θ
while not converged **do**
  Randomly select sample sequence $\{Y_{-k+1:0}, Y_{1:T}\}$
  $\widetilde{X}_0 \leftarrow g_\theta(Y_{-k+1:0})$
  for $1 \leq i \leq l$ **do**
    $\widetilde{X}_i \leftarrow Solve(\widetilde{X}_{i-1}, F_\theta)$
    $\widetilde{Y}_i \leftarrow \mathcal{H}(\widetilde{X}_i)$
  end for
  Compute gradient of $\mathcal{J}((Y_i)_{1\leq i\leq l}, (\widetilde{Y}_i)_{1\leq i\leq l})$
  Update θ in the steepest descent direction
end while
Output: Learned parameters θ.

thus reflecting the loss of information through $\mathcal{H}$. This last operator is taken as a linear projection. Without loss of generality, we can thus consider Y to be constituted by the first l components of X.

Given the remarks in Section 2.3.1, the following result shows that there is usually an infinite number of solutions to the optimisation problem.

Proposition 2.1

If $l < d$ and the unobserved part of the state is non trivial, the non-parametric version of the optimization problem eq. 2.3 admits an infinite number of null loss solutions which are distinct from canonical state representations.

Proof: Let us suppose we have an equation F along with a state X perfectly fitting all observations so that the loss is null and that we can write, because all observations are perfectly fit, $X_t = (Y_t, Z_t)$ where Z is an $\mathbb{R}^{d-l}$-valued spatio-temporal field. We also write the $\mathbb{R}^d$ to $\mathbb{R}^d$ function F as $(F^{(1)}, F^{(2)})$ so that we have:

$$\frac{\mathrm{d}X_t}{\mathrm{d}t} = \left(\frac{\mathrm{d}Y_t}{\mathrm{d}t}, \frac{\mathrm{d}Z_t}{dt}\right) = F(X_t) = \left(F^{(1)}(X_t), F^{(2)}(X_t)\right)$$

Let ϕ be a smooth diffeomorphism of $\mathbb{R}^{d-l}$, meaning that it is a smooth invertible function with a smooth inverse[4], and let $\phi_\sharp X$ be defined as:

$$\forall t,\ (\phi_\sharp X)_t = (Y_t, \phi(Z_t))$$

We then have:

$$\frac{\mathrm{d}\phi(Z_t)}{dt} = \partial_Z \phi(Z_t) \cdot \frac{\mathrm{d}Z_t}{\mathrm{d}t} = \partial_Z \phi(Z_t) \cdot F^{(2)}(X_t)$$

so that:

$$\frac{\mathrm{d}(\phi_\sharp X)_t}{\mathrm{d}t} = F^\phi\left((\phi_\sharp X)_t\right)$$

where F^ϕ is defined by:

$$F^\phi(W) = \Big(F^{(1)}\left((\phi^{-1})_\sharp(X)\right),$$
$$\partial_Z \phi\left(P^{(2)}\left((\phi^{-1})_\sharp(W)\right)\right) \cdot F^{(2)}\left((\phi^{-1})_\sharp(W)\right)\Big)$$

[4]Smoothness may depend on the considered system but here we need it at least to be of class $\mathcal{C}^3$ so that F^ϕ can be $\mathcal{C}^2$ as needed for the gradient descent algorithm.

with $P^{(2)}$ being the projection associating to a vector of $\mathbb{R}^d$ the vector of its last $d-l$ components.

Moreover, $\phi_\sharp X$ fits all observations as $\mathcal{H}(\phi_\sharp X) = \mathcal{H}(X)$ by construction. Thus, $\phi_\sharp X$ is also a null loss solution. Finally, whenever ϕ is not the identity over the range of Z, which is for an infinite number of transformations because by assumption the range of Z is non trivial, $\phi_\sharp X \neq X$ which gives us an infinite number of null loss solutions.

By construction, the canonical state X^{can} along with the canonical ODE which has generated the dataset perfectly fits the observations and thus has a null loss. From this, we can thus generate an infinite number of null loss solutions which are distinct from X^{can} by the arguments above. □

Moreover, as a corollary, if the chosen parametric families are universal approximators, which is true in our case, this means that we can obtain state representations which are non-canonical with arbitrary low losses over observations. In other words, this result shows that solving the optimization problem defining our model doesn't necessarily leads to a state space that is physically interpretable, even when observations are accurately forecasted.

However, this doesn't tell us how the states actually learned via gradient descent with our neural parametrization will look like.

In order to investigate this, in the following, we introduce two settings for analyzing the learned hidden states and help understanding what information has been learned. As we show in Section 2.6, the properties of those two settings can be useful when dealing with real world data.

2.4.2 Setting 1: Jointly Trained (JT) States

In this setting we fix the architectures of g_θ and F_θ and train the model. The dataset used is only composed of observations and is of the form $\{(Y_{-k+1}^{(i)}, ..., Y_0^{(i)}, ..., Y_T^{(i)})\}_i$. The states learned in this setting will be referred to as **Jointly Trained** (JT) states.

We can't expect JT states to have any particular structure for its $d-l$ hidden components as we don't prescribe any in the loss nor in the formulation of the problem. However, two questions can still be asked:

- Is this model able to learn dynamics which can generate accurate forecasts for the observations?
- Do the JT states contain the same information as canonical ones? In other words, can we transform JT states into canonical ones?

Intuitively, any method which sucessfully forecasts observations up to arbitrary forecasting horizons and for different initial conditions using some state representation should have stored the relevant information into the learned state representation. The following proposition makes a more precise statement of this intuition:

Proposition 2.2

There exists an invertible function e which transforms jointly learned states into canonical states.

Proof: Let X^{can}, resp. X^{JT}, be the canonical state, resp. the jointly learned state, which dynamics are described by F^{can}, resp. F^{JT}. Let Φ^{can}, resp. Φ^{JT}, denote the flow of the corresponding ODEs so that $\Phi^{\text{can}}_{t,X}$, resp. $\Phi^{\text{JT}}_{t,X}$, is the value of the canonical state, resp. jointly learned state, at time t if it was of value X at time 0. Remember that $\Phi_{t,\cdot}$ is invertible at every t for both states. Finally, let us denote S^{can}, resp. S^{JT}, the space spanned by all canonical states, resp. jointly learned states.

By construction, there is a function e and an integer K such that $X_t^{\text{JT}} = e(Y_{t-K+1}, \cdots, Y_t)$. Then, if we

denote by j the function:

$$j : X \longrightarrow e\left(\mathcal{H}\left((\Phi^{\text{can}}_{K-1,\cdot})^{-1}(X)\right), ..., \mathcal{H}\left((\Phi^{\text{can}}_{1,\cdot})^{-1}(X)\right), \mathcal{H}(X)\right)$$

we have that:

$$\forall t,\ X_t^{\text{JT}} = j(X_t^{\text{can}})$$

Let us now suppose that two different canonical states, X and X' are such that $j(X) = j(X')$. Then, applying j then the flow Φ^{JT} then $\mathcal{H}$, we see that those two states generate the same sequence of observations, as, again by construction, we always have $\mathcal{H}(X^{\text{can}}) = \mathcal{H}(X^{\text{JT}})$. This means that the two states are the same, as those are two canonical states generating the exact same sequences of observations. Thus j is injective.

Moreover, all possible observations can be generated by canonical states by definition and thus for any $X_t^{\text{JT}} \in S^{\text{JT}}$ there is a sequence of observations such that $X_t^{\text{JT}} = e(Y_{t-K\Delta t+1}, ..., Y_t)$ and then taking the corresponding canonical state X_t^{can} we have that $X_t^{\text{JT}} = j(X_t^{\text{can}})$. Thus j is surjective. □

This implies that when a model is trained without any supervision or prior information about the true states, it is still able to capture the information present in the canonical states.

2.4.3 Setting 2: Feeding in a Canonical Initial Condition

In this second setting, we inject some prior information to constrain the learned state space. There are several ways to do that. One may for example add terms to the loss that reflect physical constraints, constrain the parametrization of F to follow some predefined dynamics, etc. However, all those methods would be problem specific.

We chose here to inject prior information by prescribing an initial state with canonical structure instead of using g as above. This comes at a cost: the algorithm now has to take a full state as input for each sequence of observations. Thus, in this setting, the dataset used is of the form $\{(X_0^{(i)}, Y_1^{(i)}, ..., Y_T^{(i)})\}_i$. This is an idealized setting since usually true state information will not be available, but it is used here as a simple and generic way to inject prior information.[5]

There are also two main questions to ask in this setting:

- Are we still able to forecast accurately observations with this additional constraint? How does it compare to the JT setting?
- How is the structure of the initial state transformed through time? Are the dynamics of the hidden components of the state preserved by the prescription of the ground truth initial state?

A first fact is that, for the same reasons outlined in the proof of Proposition 2.1, there still are infinitely many possible state representations which produce accurate forecasts for observations, even when X_0 is fed as an input to the model. The idea here is that if the evolution term F happens to have a bias towards preserving structure, meaning that it would preserve through time the way information is encoded canonically, as it is in X_0, then we could hope to keep the canonical structure throughout state forecasts and thus learn unsupervisedly the hidden canonical dynamics. We will see that this happens to be the case for the Navier-Stokes equations but is not necessarily a general fact.

[5]Note however that the number of needed states is roughly T **times less** than the number of observations. Thus, this setting stays coherent with the spirit of our work.

2.5 Experiments on the Navier-Stokes Equations

In this section, we present experiments conducted on simulations of the two-dimensional incompressible Navier-Stokes equation. The dataset is the result of a simulation, thus giving us a controlled environment for experimentation and giving us a way to test our model and its properties. Moreover, those equations are fundamental for modeling transport phenomena in the atmosphere and in the ocean including the data generated for the more complex Glorys2v4 experiment (Section 2.6).

2.5.1 A Short Reminder About the Navier-Stokes Equations

A modern and thorough presentation of the incompressible Navier-Stokes equations and the underlying mathematical objects can be found in [93] for example.

Those equations are:

$$\begin{aligned} \frac{\partial u}{\partial t} + (u \cdot \nabla)u &= -\frac{\nabla p}{\rho} + g + \nu \nabla^2 u \\ \frac{\partial \rho}{\partial t} + (u \cdot \nabla)\rho &= 0 \\ \nabla \cdot u &= 0 \end{aligned} \tag{2.5}$$

where $\nabla\cdot$ is the divergence operator, u corresponds to the flow velocity vector, p to the pressure, and ρ to the density.

The Navier-Stokes equations are not of the form of eq. 2.1 as we still have the pressure variable p as well as the null divergence constraint. However, the Helmholz-Leray decomposition result [93], states that for any vector field a, there exists b and c such that:

$$a = \nabla b + c$$

and

$$\nabla \cdot c = 0$$

Moreover, this pair is unique up to an additive constant for b. Thus, we can define a linear operator $\mathbb{P}$ by:

$$\mathbb{P}(a) = c$$

This operator is a continuous linear projector which is the identity for divergence-free vector fields and vanishes for those deriving from a potential.

By applying $\mathbb{P}$ on the first line of eq. 2.5, we have, as u is divergence free from the third equation and as g derives from a potential:

$$\frac{\partial u}{\partial t} = -\mathbb{P}[(u \cdot \nabla)u] + \nu \mathbb{P}(\nabla^2 u)$$

where permuting derivation and $\mathbb{P}$ is justified by the continuity of the operator[6].

Thus, if u is solution to eq. 2.5, it is also a solution of :

$$\begin{aligned} \frac{\partial u}{\partial t} &= -\mathbb{P}[(u \cdot \nabla)u] + \nu \mathbb{P}(\nabla^2 u) \\ \frac{\partial \rho}{\partial t} &= -(u \cdot \nabla)\rho \end{aligned}$$

[6]One can use a finite difference approximation to show it for example.

which is of the form of eq. 2.1.

Conversely, the solution of the above system is such that:

$$u_t = \int \frac{\partial u}{\partial t} = \int -\mathbb{P}[(u \cdot \nabla)u] + \nu \mathbb{P}(\nabla^2 u)$$

which gives, by exchanging $\mathbb{P}$ and the integral[7] :

$$u_t = \mathbb{P}\left[\int -(u \cdot \nabla)u + \nu \nabla^2 u\right]$$

so that u is automatically of null divergence by definition of $\mathbb{P}$. The two systems are thus equivalent.

In conclusion, we have:

$$X = \begin{pmatrix} u \\ \rho \end{pmatrix}, \text{ and } Y = \mathcal{H}(X) = \rho$$

Moreover, u is generally a two or three-dimensional spatial field while ρ is a scalar field.

2.5.2 Implementation and Dataset Details

Here, we give additional practical details regarding the Navier-Stokes experiments.

- *The Dataset*

We have produced 600 separate simulations with **independently and randomly generated initial conditions**,[8] with the 2D spatial domain containing 64×64 points. The simulations were conducted with $\Delta t = 0.5s$ then subsampled 5 times. This means that the frames in the figures and tables, both in the training supervision loss and during inference, are separated by $2.5s$. The total length was 50 time-steps per simulation. Regarding turbulence, the fluid has been chosen with relatively low viscosity, close from the Euler equations in the velocity regime we sampled from, with a Reynolds number of 10000.

We have taken 300 from those simulations to construct the training set, 200 for validation and 100 for test. In particular, this means that the sequences used in the test results we present and analyze below are produced by **initial conditions the model has never seen**. For both settings, this gives us a total of 15000 observations for the training set and 10000 for the test set. In setting 1, for the restructuring of JT states experiment, we used 500 additional full states to train the transformation. In setting 2, we use an additional 2500 full states for training and 1666 for testing where each full state is the full initial state for a certain trajectory.[9]

As stated before, one also has to choose a training horizon T, to construct the used dataset of the form $\{(Y^{(i)}_{-k+1}, ..., Y^{(i)}_0, ..., Y^{(i)}_T)\}_i$ for **setting 1** and
$\{(X^{(i)}_0, Y^{(i)}_1, ..., Y^{(i)}_T)\}_i$ for **setting 2**. We have treated T as a hyperparameter of the model and have chosen it to be equal to 6. An important observation is that the higher T, the more memory demanding the training will be and the more carefully the gradient descent has to be done, especially at the first steps (by tuning the learning rate, scheduled sampling,...). However, we have observed that models with higher horizons tend to generalize better and forecast more accurately for farther time horizons, which makes sense as it makes the model take into account long term effects.

Another misconception to avoid is to confuse the training horizon T with the inference horizons at test time: For example, a model which is trained for sequences with $T = 6$ can be very accurate for longer time horizons as we show in the results below.

[7] To prove this, we can take a sum approximation to the integral and use again the linearity then the continuity of $\mathbb{P}$.

[8] For each, we have chosen a random location where we put a concentric density, as well as a random velocity field.

[9] This means in particular that the supervision loss is still only calculated w.r.t. observations.

• *Implementation*

In practice, the cost functional $\mathcal{J}$ is estimated on a minibatch of sequences from the dataset and optimized using stochastic gradient descent. Throughout *all* the experiments, F_θ is a standard residual network [115], with 2 downsampling layers, 6 residual blocks, and bilinear up-convolutions instead of transposed convolutions.

In the experiments for setting 1, we parametrize g_θ as a Unet [229]. More precisely, we have used a modified variant of the FlowNetS architecture of [77] with:

- three double convolution steps, each double step having a two-strided first convolution then a one-strided second one, with all convolutions having kernels of size 3 and batch normalization;
- non linearities are Leaky ReLU with parameter 0.1;
- the last two deconvolution steps are replaced with convolutions so that the desired number of output channels is obtained (in our case, we start with four input channels and output two).

Note that all experiments were conducted with the same architectures, showing the genericity of our approach: while adapting the parametrization can improve quantitative reults, it doesn't fundamentally alter our conclusions and shows that the cost of experimentation when developping a model for a new dataset can be decreased.

To discretize the forward equation eq. 2.2 in time, we use a simple Euler scheme. Note that the discretization step-size may differ from the actual time interval between consecutive observations as defined by the simulations; in our case, we apply 3 Euler steps between two observations, *i.e.* $\delta t = \frac{1}{3} \times 2.5s$. For the spatial discretization, we use the standard grid discretization induced by the dataset. The weights of the residual network θ are initialized using an orthogonal initialization. Our model is trained using a scheduled sampling scheme with exponential decay, along with the Adam optimizer, with a learning rate set to 1×10^{-5}. We use the Pytorch deep learning library [209].

• *Baselines and Metrics*

We compare our models to two different baselines:

PKnI It is a physics-informed deep learning model described in [69], where prior physical knowledge is integrated: it uses an advection-diffusion equation to link the velocity with the observed temperatures, and uses a neural network to estimate the velocities.

PRNN [271] It is a heavy-weight, state of the art model used for video prediction tasks. It is based on a Spatiotemporal Convolutional LSTM that models spatial deformations and temporal variations simultaneously.

We use a renormalized relative squared error as a metric for observations:

$$\frac{1}{T}\frac{1}{|\Omega|}\sum_{k=1}^{T}\sum_{x\in\Omega}\frac{\|\mathcal{H}(X_k(x)) - Y_k(x)\|_2}{\|Y_k(x)\|_2} \tag{2.6}$$

To evaluate the quality of the hidden states, we use cosine similarity between the model's hidden state and the true hidden state of the system[10]:

$$\frac{1}{K}\sum_{k=1}^{K}\frac{1}{|\Omega|}\sum_{x\in\Omega}\frac{\left\langle u^1(x), u^2(x)\right\rangle}{\|u^1(x)\|\,\|u^2(x)\|} \tag{2.7}$$

[10]The cosine similarity is relevant for the comparison with PKnI: the norm of its hidden state may not correspond to the ground truth norm.

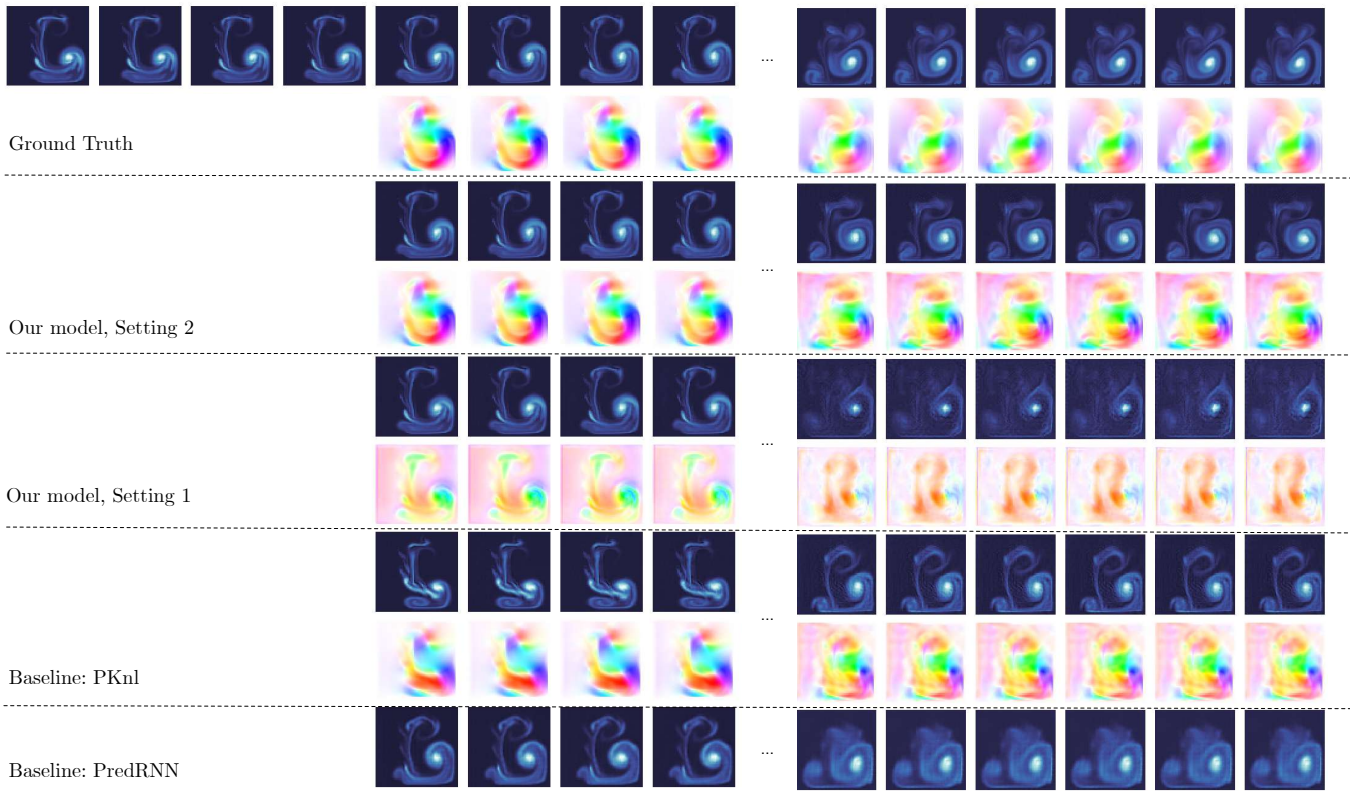


Figure 2.1: Forecasting the Navier Stokes equations 30 time-steps ahead with different models, starting from a given initial condition. In this figure as well as in the following ones, the velocity field is represented using the Middlebury Color Code as implemented in the flowlib library https://github.com/liruoteng/OpticalFlowToolkit/blob/master/lib/flowlib.py.

where the u^i are the horizontal and vertical components of the velocity field u

The cosine similarity is relevant for the comparison with PKnI: the norm of its hidden state may not correspond to the ground truth norm.

For the velocity vector field representation, color represents the angle, and the intensity the magnitude of the associated vectors. More specifically, we represent the velocity fields into an image by using the Middlebury color code from the flowlib library (for which there is a code here: https://github.com/liruoteng/OpticalFlowToolkit/blob/master/lib/flowlib.py).

2.5.3 Forecasting Observations

Table 2.1: Relative MSE $\frac{1}{T}\frac{1}{|\Omega|}\sum_{k=1}^{T}\sum_{x\in\Omega}\frac{\|\mathcal{H}(X_k(x))-Y_k(x)\|_2}{\|Y_k(x)\|_2}$ for our model and different baselines, at different temporal horizons on the Navier Stokes equations. Note that the Setting 2 model uses a full, true initial state as X_0 while the Setting 1 one only relies on observations.

Model	$T=5$	$T=10$	$T=50$
Ours (Setting 1)	0.152	0.243	0.650
Ours (Setting 2)	**0.118**	**0.180**	**0.483**
PKnI [69]	0.194	0.221	0.752
PRNN [271]	0.170	0.227	0.719

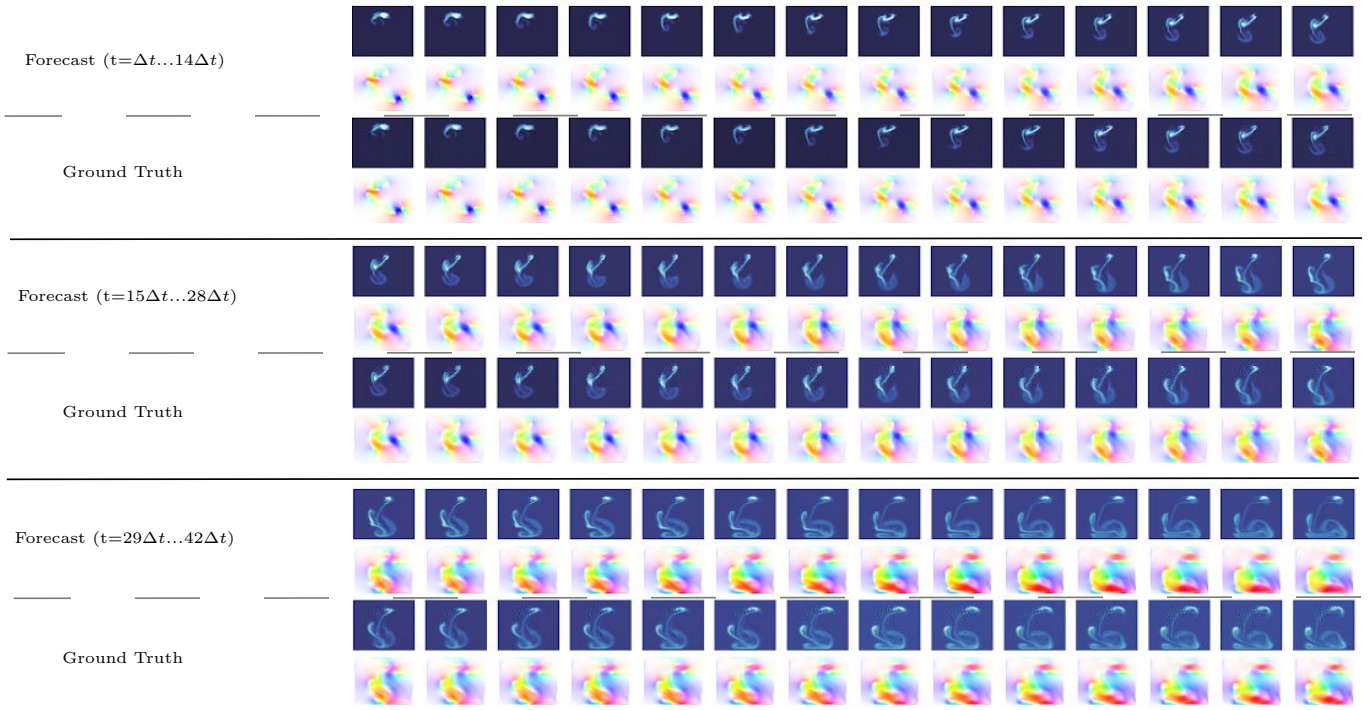


Figure 2.2: Setting 2: Forecasting the Navier Stokes equations, starting from a given initial condition (not shown here). We forecast 42 time-steps ahead and compare results with the ground truth simulation.

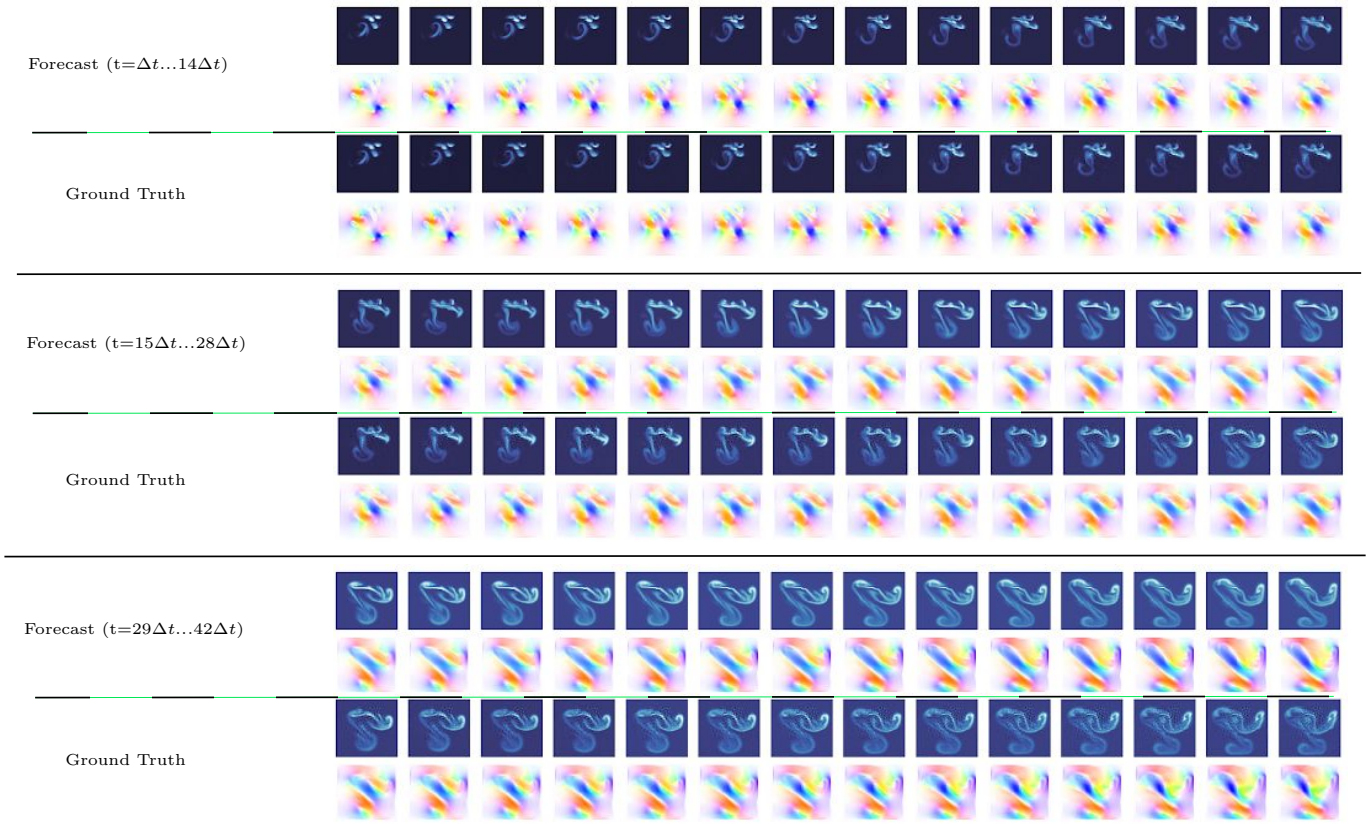


Figure 2.3: Setting 2: Forecasting the Navier Stokes equations, starting from a given initial condition (not shown here). We forecast 42 time-steps ahead and compare results with the ground truth simulation.

Figure 2.1 shows a sample of the predictions of our system over the test set for the Navier Stokes equations for both settings 1 and 2. The good results it shows are confirmed by Table 2.1. Our model is able to predict observations up to a long forecasting horizon (results are shown for up to 30 steps in Figure 2.1 and 50 steps in Table 2.1), which means that it has managed to learn the dynamical system. Note that for setting 2 in Figure 2.1, the initial states used at test time have never been seen at training time which means that the optimization problem was solved correctly without over-fitting. Moreover, the cost function is only calculated using observations which means full states are not used for supervision, in accordance with our setting. An interesting remark is to observe that the jointly trained model (setting 1) is slightly less accurate than the one given X_0 (setting 2), which makes sense as this last algorithm is given a few additional full states when JT isn't given any.

Visually, as can be seen in Figure 2.1 by looking at the small features of the observations, our model manages to capture many details which are important to robust long term forecasts while the PRNN model, which proves to be a strong baseline at the level of observations[11] for the first few steps, produces less sharp predictions which explains its worse performance when evaluated on long term predictions. Additional samples shown in Figure 2.2 and Figure 2.3 confirm this observation.

2.5.4 Restructuring Jointly Trained States

Proposition 2.2 shows that there must exist a way to transform JT states into canonical ones, which would make them more palatable and easier to interpret. In order to confirm this theoretical result empirically, we did the following:

1. We took a small set of full canonical states from the Navier-Stokes dataset, corresponding to 10 sequences (to be compared to 300 sequences of observations used for training) and computed the corresponding JT states.

2. We used it as a training set to learn the invertible transformation between JT states X^{JT} and canonical ones X^{can}, which boils down to a regression problem where we want to predict X^{can} from X^{JT}.

Figure 2.4 shows an example of the output from the transformation this transformation yields: it allows us to transform the non-structured hidden states of the jointly trained model into interpretable states corresponding to the canonical representation. From a quantitative point of view, after 5 predictions, the average cosine similarity over the whole test set goes from 0.192 in the jointly trained representation to 0.582 when transformed. While this result is far from perfect[12], it still shows promise and demonstrates that this approach could be applied in many cases to recover canonical states.

2.5.5 Imposing the Initial Condition Prescribes the Hidden Dynamics

Figure 2.1 shows that in **setting 2**, when we add a full initial state, our model is able to forecast not only observations but also the dynamics of the hidden components of the state. This is a surprising result: even though this model gets additional structured information at the input, there are still an infinite number of ways to transport that information through time-steps and to store it into the state representation. Table 2.2, shows the mean cosine similarity between target and predicted states for our model in setting 2. This similarity is high (around 0.8) for short term prediction (5 steps) and still substantial for long term prediction (around 0.5 for 50 steps). For comparison, we also indicate in Table 2.2 the values obtained with the PKnI model.

[11] It doesn't produce meaningful hidden states.

[12] In particular, the choice of the regression algorithm isn't obvious and the size of the needed dataset will depend on this choice as well as on the desired accuracy, as with any regression problem.

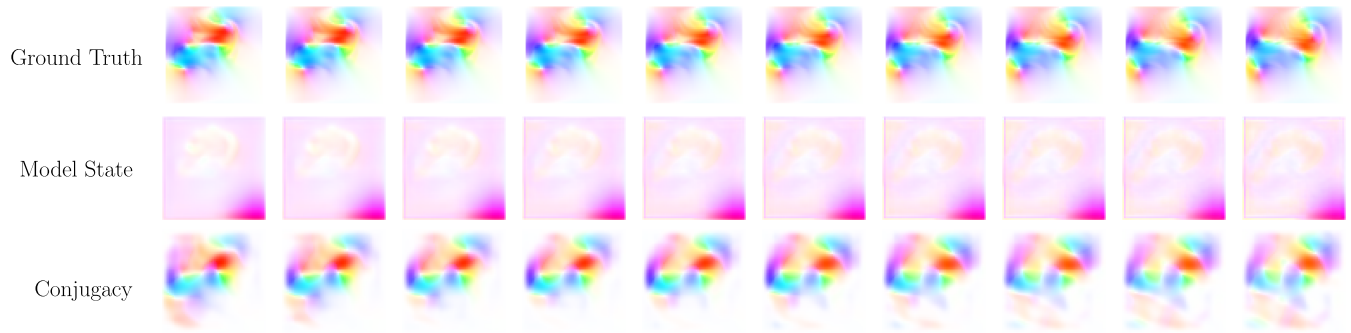


Figure 2.4: Setting 1: Example of a sequence of hidden states transformed by the calculated conjugacy.

Table 2.2: Cosine similarity $\frac{1}{K}\sum_{k=1}^{K}\frac{1}{|\Omega|}\sum_{x\in\Omega}\frac{\langle u(x),v(x)\rangle}{\|u(x)\|\|v(x)\|}$ scores for our models and a baseline, at different temporal horizons on the Navier Stokes equations.

Model	$T=5$	$T=10$	$T=50$
Ours (Setting 2)	**0.798**	**0.679**	**0.483**
PKnI	0.243	0.207	0.098

In order to see if this is a property of the particular architecture used here, we conduct a series of ablation studies where we try to remove different components of the model and see how it behaves (numerical results are shown in Table 2.3):

ResNet. Here we simply use a residual network, with *the exact same architecture* as the one used to parameterize our model. The difference is that here we use it directly, not through an Euler solver. The results are notably less accurate for observations but, more importantly, this model turns out to be completely unable to forecast hidden states corresponding to the true ones. This shows that the way our model is structured around a solver which takes into account the differential structure of the studied problem is a strong regularizer.

ResNet no skip. This last argument may remind us that a residual network closely resembles the non-uniform discretization of an ODE. Thus, this should help it perform well and explains the relatively good results on observations for the ResNet and, by getting rid of the skip connections while keeping all layers untouched, which leads to a CNN, the performance should worsen. This is indeed what happens in our tests.

Unet. We tried using this other classical architecture, which is often used for regression problems, with roughly the same number of parameters as in our parameterization. It proved to be weak against our model for both observations and hidden states.

Ours, Modified $\mathcal{H}$. Here, we seek to check whether our results depend of our particular choice of $\mathcal{H}$: we change it and make it project to the first dimension of the velocity field (instead of the density). We use our model in setting 2 (we give X_0 as input). The results, while slightly less good, are quite robust to this change, considering that we haven't changed the hyper-parameters of the model.

Table 2.3: Ablation study for our model, at different temporal horizons on the Navier Stokes equations

Model	T=5		T=10		T=50	
	MSE	cosine	MSE	cosine	MSE	cosine
Ours (Set. 2)	0.118	0.798	0.180	0.679	0.628	0.483
Ours, (Set. 2, Mod. $\mathcal{H}$)	0.191	0.732	0.288	0.620	0.49	0.534
Resnet	0.288	0.604	0.391	0.333	0.73	0.032
Unet	0.659	0.069	0.692	0.028	0.84	0.023
Resnet No Skip	0.615	0.162	0.71	0.060	0.897	-0.04

- *Discussion of the results*

  Those experiments lead us to the following conclusions:

  - In the case of the Navier-Stokes equations, our model, with a simple solver for an equation parametrized through a residual network, allows us to learn unsupervisedly the dynamics of the hidden dynamics of the state.
  - This result is robust to a change to the dimension $\mathcal{H}$ project onto.
  - The fact that a solver is used, instead of a direct regression model, appears to be very important, as comparisons to other standard powerful architectures show, even when the exact same parametrizations are used.

However, this still doesn't explain why this works for the hidden components, as the problem is ill-posed nonetheless. We hypothesize that the architecture of the network used to parametrize the equation is biased towards the preservation of the input code, which happens to be that of the canonical state because X_0 is fed into it.

We show in Chapter 4 that a similar kind of phenomenon is also empirically observed in other learning tasks and delve further into the details of this phenomenon in Section 4.11.

In those first experiments, we have studied two separate settings with different levels of supervision over the full state: Setting 1 supposes that none is available while Setting 2 allows to initialize with a fully known state. However, in practice, systems of interest may present a hybrid setting, for example with the availability of a proxy for a part of the unobserved part of the state. This is what the following section is about as it will study an example of such a situation for a more complex, more realistic dynamical system. Finally, the last section further studies the generalization of our model with changes in boundary conditions.

2.6 Testing the Limits of the Model: Forecasting Ocean Circulation Dynamics From Satellite Images

In this section, we use our model to study Sea Surface Temperatures dynamics as modeled by the Glorys2v4 simulations. We assume access to a proxy for part of the hidden components of the state for the initial condition which places us in a hybrid setting when compared to the two settings used for Navier-Stokes equations. This allows us to leverage the properties of both while remaining in a realistic context.

We first describe the realistic, state-of-the-art simulation of ocean circulation we used to form our dataset. We then propose two instances of our model and compare them to standard baselines.

2.6.1 The Glorys2v4 Dataset

The Glorys2v4 product is a reanalysis of the global Ocean (and the Sea Ice, not considered in this work). The numerical ocean model is NEMOv3.1 [177] constrained by partial real observations of Temperature, Salinity and Sea Level. Oceanic output variables of this model are daily means of Temperature, Salinity, Currents, Sea Surface Height at a resolution of 1/4 degree horizontal resolution.

The NEMO model describes the ocean by the primitive equations (Navier-Stokes equations together with an equation of states). Let $(\mathbf{i}, \mathbf{j}, \mathbf{k})$ the 3D basis vectors, U the vector velocity, $\mathbf{U} = \mathbf{U}_h + w\mathbf{k}$ (the subscript h denotes the local horizontal vector, *i.e.* over the $(\mathbf{i}, \mathbf{j})$ plane), T the potential temperature, S the salinity, ρ the *in situ* density. The vector invariant form of the primitive equations in the $(\mathbf{i}, \mathbf{j}, \mathbf{k})$ vector system provides the following six equations (namely the momentum balance, the hydrostatic equilibrium, the incompressibility equation, the heat and salt conservation equations and an equation of state):

$$
\begin{aligned}
&\frac{\partial \mathbf{U}_h}{\partial t} = -\Big[(\mathbf{U}.\nabla)\mathbf{U}\Big]_h - f\mathbf{k} \times \mathbf{U}_h - \frac{1}{\rho_0}\nabla_h p + D^{\mathbf{U}} + F^{\mathbf{U}} \\
&\frac{\partial p}{\partial z} = -\rho g \\
&\nabla.\mathbf{U} = 0 \\
&\frac{\partial T}{\partial t} = -\nabla.(T\mathbf{U}) + D^T + F^T \\
&\frac{\partial S}{\partial t} = -\nabla.(S\mathbf{U}) + D^S + F^S \\
&\rho = \rho(T, S, p)
\end{aligned}
$$

where ρ is the *in situ* density, ρ_0 is a reference density, p the pressure, $f = 2\Omega\mathbf{k}$ is the Coriolis acceleration. D^U, D^T and D^S are the parameterizations of small-scale physics for momentum, temperature and salinity, and F^U, F^T and F^S surface forcing terms.

As in Section 2.5, the divergence-free constraint can be enforced through the Leray operator. Moreover, ρ is a function of other state variables so that the state can be written as:

$$
X = \begin{pmatrix} U \\ p \\ S \\ T \end{pmatrix} \text{ and } \mathcal{H}(X) = \overline{T}.
$$

where $\overline{T}$ is the daily mean surface temperature derived from the instantaneous potential temperature T in the model.

The level of supervision for the initial state here is hybrid when compared to the two settings described in the previous sections: in addition to the temperature observations, it is possible to access an estimation of the velocity field $\tilde{w}_0$. We also assume that this access frequency is less than daily, meaning that the situation is akin to Setting 2 described in the previous section.

2.6.2 Models

This dataset is much more challenging and represents a leap from the fully simulated one presented before. One reason is obviously the high dimensionality of the system and the absence of a full state as initial input to our system as we only have a proxy over the velocity field. A second one is the fact that the neural network model is trained over sequences where only a local spatial region is observed (see Figure 2.6) corresponding to fixed size zones of the ocean. The physical model, on the other hand, simulates the dynamics over a larger area on the ocean. This means that informations from the

neighboring areas beyond the fixed size zone is not available to the neural network. This makes the dynamics for the coresponding zone **non-stationary** as boundary conditions are constantly shifting, thus violating an assumption of our method and making it difficult to make long term forecasts with a reasonable number of observations. We can hope for the dynamics to be locally stationary so that the model can work well for a few steps.

In other words, the initial temperatures T_0 (since we observe the temperatures, $Y_0 = T_0$) and the proxy of the velocity field $\tilde{w}_0$ provided as initial input are insufficient to represent the full state. Taking this fact into account, we build on the results obtained in the case of the Navier Stokes equations and propose two variants of our model:

- **Ours**, which is the same as before in Setting 2, taking as initial state:

$$X_0 = \begin{pmatrix} Y_0 \\ \breve{w}_0 \\ 0 \end{pmatrix}$$

- **Ours, with Estimation** where we use past observations $Y_{-K:0}$ in order to infer the unknown part of the initial state, similarly to what is done in the JT model:

$$X_0 = g_\theta = E_\theta(Y_{-K:0}, \tilde{w}_0) + \begin{pmatrix} Y_0 \\ \breve{w}_0 \\ 0 \end{pmatrix} \tag{2.8}$$

Here, E_θ is an encoder neural network. Using it allows us to encode available information from the observations $Y_{-K:0}$ which is not contained in $\breve{w}_0$ nor in T_0. For E_θ, we use the UNet architecture [229].

2.6.3 Results and Conclusions

We have used the same hyper-parameters to build and train our architectures as for the Navier-Stokes simulations (described in Section 2.5.2). We also consider the same baselines. As a reviewer of the paper corresponding to this work suggested, we also compute a persistence score which is produced by simply considering a constant output corresponding to the initial value. This is meaningful as it allows to evaluate the "memory" of the ocean over the timescales considered here.

Regarding the forecasting of observations, we can clearly see, as expected, from Figures 2.5 to 2.7 as well as Table 2.4 that this task is more challenging, with lower performances for all models when compared to those obtained in the case of the Navier Stokes equations, even though we evaluate for shorter time horizons. Nevertheless, the two variants of our model still perform better than the two powerful Deep Learning baselines we test against, as well as against the persistence score which does underperform all other baselines.

We also observe, from the cosine similarity results, that our models are still able to reproduce some coherent dynamics for the hidden components of the state for which the initial condition was given. Using an additional estimation, while lowering the accuracy for observations, also helps with improving the cosine similarity for those dynamics. However, comparing to the persistence baseline shows that our models are not really doing better than simply preserving the structure of the velocity field.

This last fact is actually very interesting as it shows that the phenomenon we observed for the Navier-Stokes equations is indeed specific to those. This will be investigated further in Section 4.11 of Chapter 4.

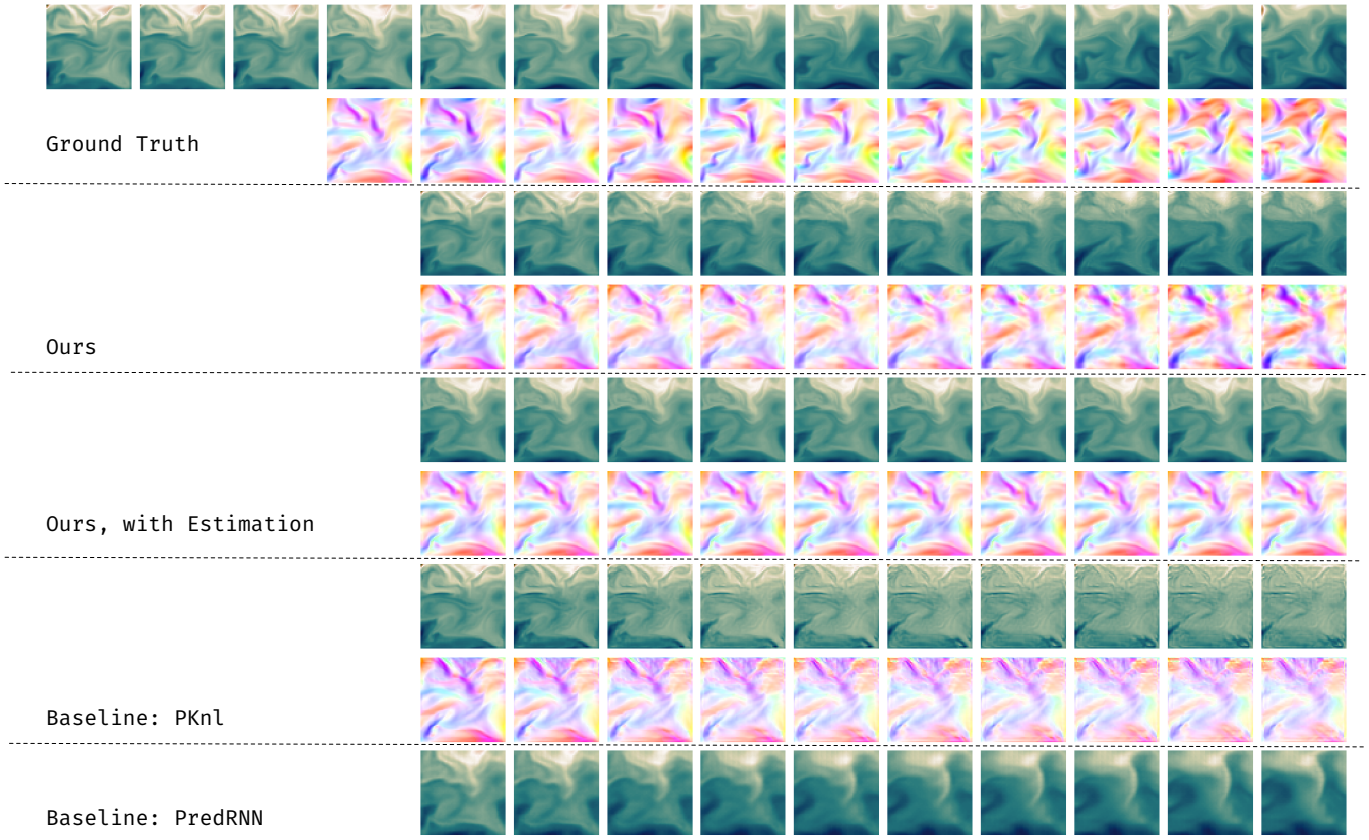


Figure 2.5: Forecasting Sea Surface Temperatures 10 time-steps ahead with different models, starting from a given initial condition.

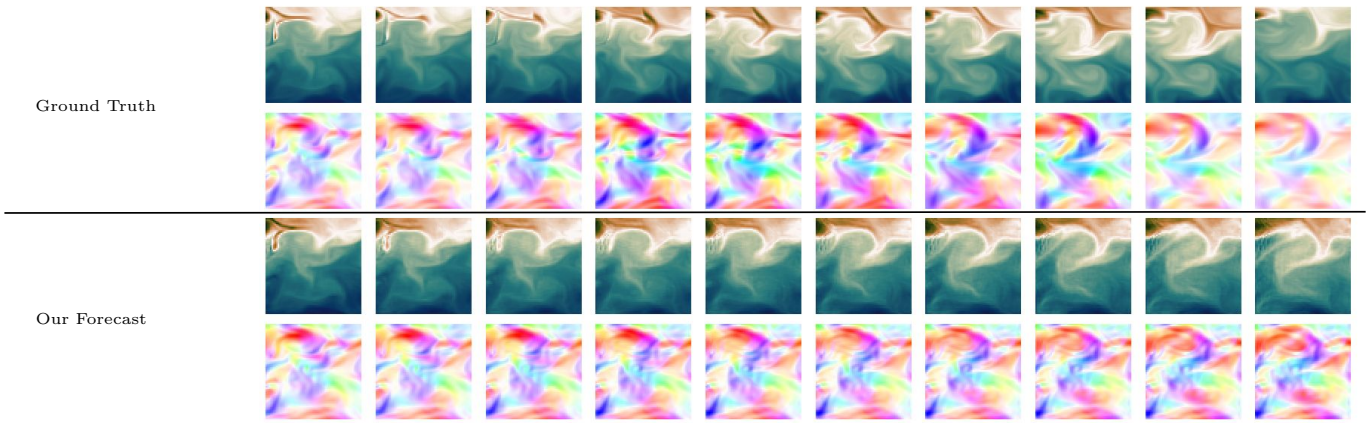


Figure 2.6: Forecasting Glorys2v4 10 time-steps ahead, starting from a given, full state, initial condition (not shown here), wthout the estimation.

Table 2.4: Relative MSE and cosine similarity scores for our models and different baselines, at different temporal horizons on the Glorys2v4 dataset. State estimation is not available for PRNN which is based on a recurrent network, hence the crosses in the "cosine" columns

Model	T=5		T=10	
	MSE	cosine	MSE	cosine
Persistence	0.476	**0.788**	0.842	0.666
Ours	**0.306**	0.671	**0.402**	0.589
Ours, Est.	0.364	0.718	0.490	**0.670**
PKnI	0.411	0.448	0.494	0.368
PRNN	0.423	XX	0.546	XX

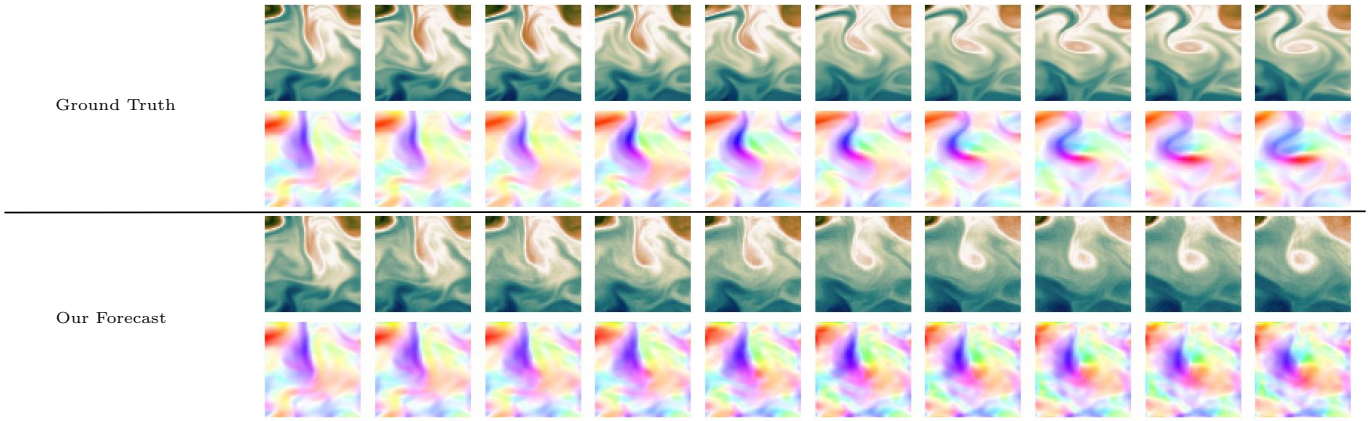


Figure 2.7: Forecasting Glorys2v4 10 time-steps ahead, starting from a given, full state, initial condition (not shown here), without the estimation.

2.7 Testing the Limits of the Model: Learning in 3D and Generalizing to Unseen Boundary Conditions for Cardiac Electro-Physiology Equations

The work presented in this section has been conducted as a collaboration with the Computational Cardiology team of Maxime Sermesant at Inria Epione.

Here, we study reaction-diffusion equations which are often used in the context of Cardiac electrophysiology (EP) models. Those, while achieving good progress in simulating cardiac electrical activity, have limited applicability in clinical contexts because of numerical issues and computational times.

In this section, we show that our model can be applied to this context and, by doing so, explore two new aspects: its usability beyond two-dimensional data and its ability to generalize to unseen boundary conditions (which were considered fixed throughout the previous experiments).

2.7.1 Equations and Datasets: Electro-Physiological Modelling

Mathematical modelling of the cardiac cell has been an active research area for the last decades. Cardiac electrophysiology models can accurately reproduce cardiac cells electrical behaviour. This

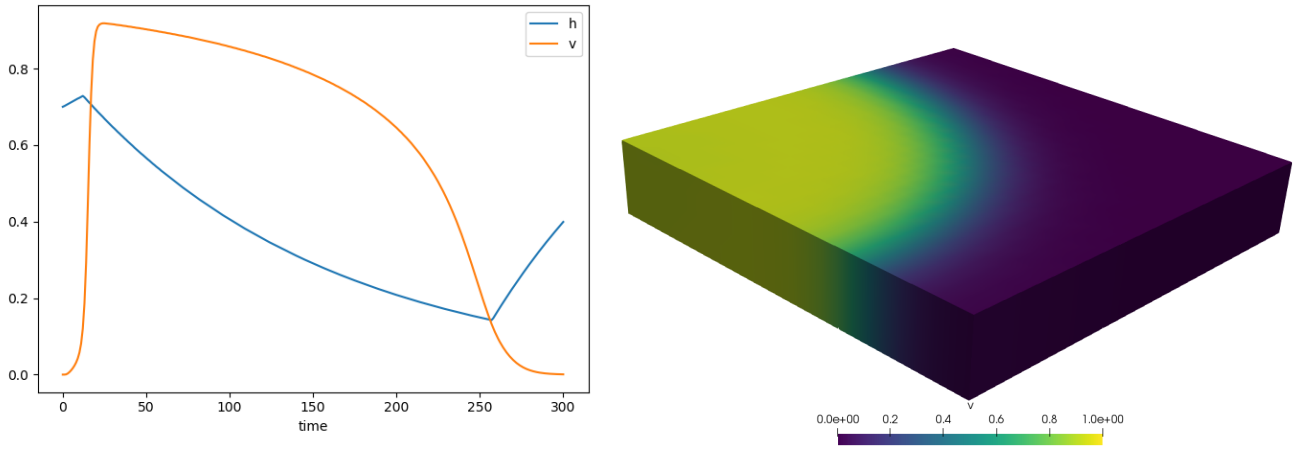


Figure 2.8: Cardiac electrophysiology model simulation. (Left) v and h values along time for a given point of the domain, (Right) spatial propagation of v at a given time point.

provides a mathematical framework to simulate cardiac activity, however it remains challenging to achieve in 3D due to numerical issues and computational time. These difficulties become all the more apparent when trying to personalise such model by matching patient data. This matching can also be hampered by the approximations of the model.

Cardiac electrophysiology modelling offers a large variety in terms of model complexity. Here, we used the Mitchell-Schaeffer model [187] which is a two variables model that has been successfully used in patient-specific modelling [225]. The variable v represents the transmembrane potential while the "gating" variable h controls the repolarisation:

$$\frac{\partial v}{\partial t} = \Delta v + \frac{hv^2(1-v)}{\tau_{in}} - \frac{v}{\tau_{out}} + J_{stim} \tag{2.9}$$

$$\frac{\partial h}{\partial t} = \begin{cases} \frac{1-v}{\tau_{open}} & \text{if } v < v_{gate} \\ \frac{v}{\tau_{close}} & \text{if } v > v_{gate} \end{cases} \tag{2.10}$$

In order to use convolutional neural networks, it is much more efficient to work on a Cartesian grid. To this end, a Lattice Boltzmann method was used to solve the EP model [222] for the data used here.

Combining equations eq. 2.9 and eq. 2.10, the studied dynamical system can be written in the form:

$$\frac{dX_t}{dt} = F(X_t)$$

as usual with X is a spatio-temporal three-dimensional vector field over compact spatial domain $\Omega \subset \mathbb{R}^3$. In this case, we thus have:

$$X = \begin{pmatrix} v \\ h \end{pmatrix}$$

In other words, for any given time t and point of the 3D domain (or 2D depending on the section) $x \in \Omega$, $X_t(x)$ is a two-dimensional vector.

In practice, while the value of v can be measured, h is a hidden variable which is difficult to estimate. Therefore, we put:

$$\mathcal{H}(X) = v$$

which makes our model a partially observed one, as only some parts of the full state can be directly measured.

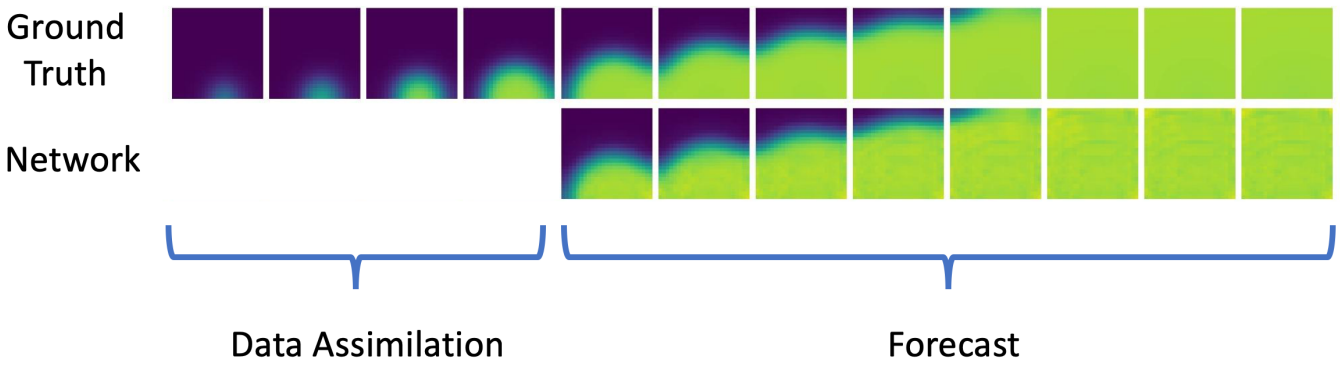


Figure 2.9: Transmembrane potential ground-truth (top) and forecasted (bottom) for one slice of the tissue slab. While only one slice is shown, prediction is conducted over the 3D

2.7.2 Learning with Three-Dimensional Data

We can now present our first experiments with those equations which are the first in a 3D domain.

• *Practical details*

Dataset The domain is a slab of cardiac tissue of size $25 \times 25 \times 5$ mm^3, which we will denote Ω, discretised in voxels of 1 mm^3. To initiate the propagation (J_{stim}), a current of 1 normalised transmembrane potential unit was applied on a single voxel for 10 ms. The Mitchell-Schaeffer model parameters are taken from the original paper, except h_0 which was set to 0.7. The simulation was conducted for 300 ms, and stored every ms with a discrete time step of 0.1 ms. The simulation database was created by stimulating the domain from every voxel. We therefore have 3 125 simulations in the database. Let V and H be the complete solutions in space and time of these equations.

Implementation We conduct our experiments in the JT setting, *i.e.* Setting 1 of Section 2.4.2.

Operator g is implemented as a three-dimensional U-Net inspired from [229] and modified with 25 filters at the initial stage. F is implemented as a 3D Residual Network again inspired from [115] with three-dimensional convolutions, two downsampling initial layers, three intermediary blocks and an inner dimension of 24. In order to solve the forward equation, we use an explicit one-step Euler scheme.

To construct a training dataset, we have randomly selected 500 simulations, temporally downsampled three times where each slab is of size $25 \times 25 \times 5$. A validation set of 300 simulations and a test set of 200 were also randomly selected from the remaining simulations and down-sampled likewise. Again, this means in particular that test and training simulations have different initial conditions so that the model is tested with data it has never seen. Only the v variable was used as supervision and we have taken a sequence of 4 observations as input to g and a training horizon of 8 time-steps.

The optimization over θ uses the ADAM optimizer [140] with a learning rate of 10^{-3}. We also use exponential scheduled sampling [34] with parameter 0.9999 during training and start with a reweighted orthogonal initialization for the parameters of F.

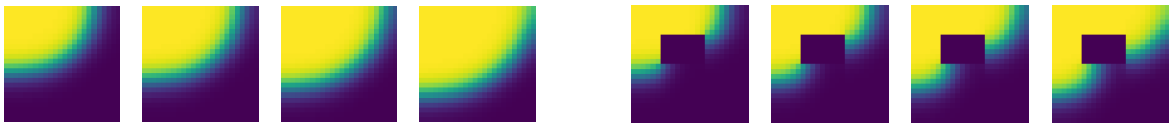
• *Results*

We present here results on the forecast over 8 time frames after assimilating the first 4 frames (see Figure 2.9). We can observe very good agreement with the ground truth on this forecast, which represents an important part of cardiac dynamics within this virtual slab of tissue, from early depolarisation to full depolarisation.

Table 2.5 shows MSE results for our algorithm for different forecasting horizons. We can see that up to a reasonable number of steps, here corresponding to 51 ms, our model does quite well.

Table 2.5: Relative mean-squared error of normalised transmembrane potential per time-step for different forecasting horizons.

Time	2 (6 ms)	5 (15 ms)	8 (24 ms)	11 (33 ms)	14 (42 ms)	17 (51 ms)
MSE	0.005	0.012	0.048	0.060	0.14	0.58



(a) (b)

Figure 2.10: Example of transmembrane potential (yellow) propagation in the cardiac tissue slab in absence (a) and presence (b) of scar tissue, through successive time steps.

Figure 2.9 visually confirms those numerical results.

It has to be noted that for such a cardiac model, predicting the propagation between time 10 and 60 ms on such a slab of tissue represents the whole depolarisation wave, which is the most important part of cardiac electrophysiology dynamics.

Moreover, while this model was slower to train compared to those of previous experiments, the inference with the trained weights is still very quick and does not require any recalibration which paves the way for practical clinical applications of those results were confirmed on more realistic data.

One first step to do so would be to apply this model on hearts with damaged cardiac tissues. In terms of the equations, this can be translated into boundary conditions with more complex geometries which would be unseen on training time. This is the subject of the last part of this section.

2.7.3 Generalizing to Unseen Boundary Conditions

In this section, we study the behaviour of our learning model and its ability to generalize in unseen contexts. More precisely, we want to know whether it is possible for it to generalize with domain geometries and characteristics different from those contained in the training dataset.

In the particular context of Cardio-Electrophysiological modelling, this extrapolation ability is paramount for the model to be applicable to hearts with scarred tissues.

- *Description of the setting*

Going back to the model presented in the previous section, we considered that the domain was a homogeneous rectangular slab. Here, we consider diffusion in tissues with ischaemic (non conductive) regions, denoted scars in the following. In clinical practice it is essential to be able to recognise and to estimate the impact of scars because they are the main cause of cardiac arrhythmias. For example, in Figure 2.10 we can clearly see the changes in the dynamics of the depolarisation wave in the presence of scar tissue (black area). We also introduce in our simulations multiple onsets and colliding fronts, as it is a classical situation in cardiac electrophysiology.

The focus is then to evaluate the ability of our model to generalise to unseen conditions. The model is trained on simulated data corresponding to a relatively simple context (one type of scar, one front and a set of several discrete conduction velocities) and its generalisation ability is evaluated on more challenging contexts like more complex scars, multiple fronts and any real conduction velocity sampled from a given interval. Figure 2.11 presents the general experimental setting used in the manuscript.

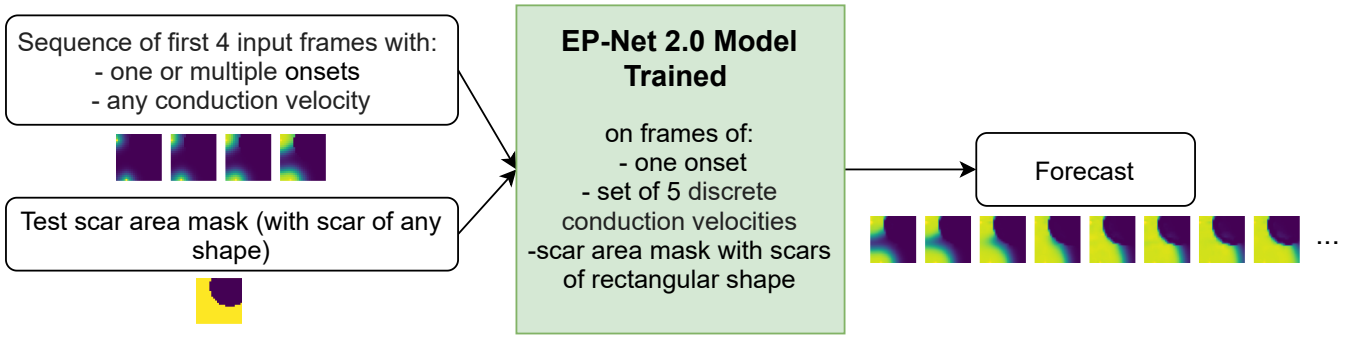


Figure 2.11: General setting used throughout the manuscript. Once trained, the EP-Net 2.0 model takes as input a context consisting of a few (4 here) observations plus an indication of the scar area (left), and forecasts the depolarization wave dynamics (right).

In practice this translates into a constraint corresponding to the presence of the scar:

$$\forall t, \forall x \in \Omega_{scar}, X_t(x) = 0 \tag{2.11}$$

where $\Omega_{scar} \subset \Omega \subset \mathbb{R}^2$and scars are considered here as binary masks for simplification.

This additional constraint thus gives us the following optimization problem

$$\begin{aligned} \underset{\theta}{minimize} \quad & \mathbb{E}_{V \in \text{Dataset}} \mathcal{L}(V, \mathcal{H}(X^\theta)) \\ \text{subject to} \quad & \frac{dX_t}{dt} = F_\theta(X_t), \\ & X_0 = g_\theta(V_{-k}), \\ & (X_t)_{|\Omega_{scar}} \equiv 0 \end{aligned} \tag{2.12}$$

which is then enforced in practice using a modified loss:

$$\mathcal{L}(V, \widetilde{V}) = \mathcal{L}_{obs}(V, \widetilde{V}) + \lambda_{scar} \mathcal{L}_{scar}(\widetilde{V}), \tag{2.13}$$

$$\text{where: } \mathcal{L}_{obs}(V, \widetilde{V}) = \int_0^T \|V_t - \widetilde{V}_t\|^2 \mathrm{dt}, \; \mathcal{L}_{scar}(\widetilde{V}) = \int_0^T \|1_{\Omega_{scar}} \odot \widetilde{V}_t\|^2 \mathrm{d}t$$

with $\odot$ the element-wise product and λ_{scar} a hyper-parameter used to balance the losses.

• *Practical details*

Data collection. We generated 2D data frames using the Lattice Boltzmann method to solve the EP model [223] on a Cartesian grid. The Mitchell-Schaeffer model parameters are taken as in the original paper [188]: $\tau_{in} = 0.3$, $\tau_{out} = 6$, $\tau_{open} = 120$, $\tau_{close} = 150$, $v_{gate} = 0.13$. The computational domain represents a slab of cardiac tissue of size 24×24 mm^2 discretised with 1 mm^2 pixels. A stimulation current was applied for 10 ms to initiate the propagation in selected pixels. We superposed a mask with randomly generated rectangular area (with random size and position) of excluded domain to simulate the cardiac scars. Training was performed with 5 different conduction velocities, each corresponding to a given parameter of conductivity: $\sigma \in \{1, 2, 3, 4, 5\}$. The conductivity was applied uniformly on whole cardiac slab except for the scar area in which there is 0 conductivity by definition with a single value of σ being used per simulation. The simulations were conducted for $30ms$, with a discrete time step of $0.1ms$, and stored every $1ms$. Then random sequences of 10 data frames were extracted at different time points for training / validation data. Overall we have a database of 30000 training and 12000 validation samples.

Training settings. The parameter λ_{scar} in the loss eq. 2.13 was set to 0.1 and a learning rate for ADAM optimiser was set to 10^{-3}. We use ResNet with 64 filters at the initial stage, three downsampling initial layers and three intermediary blocks and start with a reweighted orthogonal initialisation for its parameters. We also use exponential scheduled sampling [34] with parameter 0.9999 during training. We trained our model until full model convergence (about 5000 epochs). In each training (and validation) sequence we used the first 4 frames for initialisation, as input to g_θ, and the rest as supervision with training horizon 6.

- *Results*

  Tests were performed in two situations:

  - the first with a scar and current distributions similar to the training set, which allows to test our model in a context where there is an obstacle which is more difficult than the one in the previous section;
  - the second in a context where the training dataset is the same but with different scar geometries and initial current onsets distributions in the test dataset thus allowing to test the model ability to generalise to new situations.

Testing environment similar to the training one: scars of rectangular shape. Figure 2.12 illustrate the behaviour of our trained EP-Net 2.0 model in test conditions similar to the training ones: rectangular scars with random size and position plus one onset only. The Figure 2.12a shows the forecast over 9 time frames (9 ms) after assimilating the first 4 frames (not presented in Figure 2.12a). We observe very good agreement with the ground truth on this forecast, which represents an important part of cardiac dynamics within this virtual slab of tissue, from early depolarisation to full depolarisation. Figure 2.12b shows that our model has a very good precision on depolarization during more than 50 ms, an equilibrium state for the model, but cannot predict a repolarisation (Figure 2.12c). However, this is to be expected as the training horizon was too short to capture this phenomenon. Quantitative results provided in Table 2.6 for different forecasting horizons T (6, 12 and 24 ms) show excellent performance, while the training time horizon was only 6 ms.

Generalisation ability: scars of various shapes and multiple onsets. The ability to extrapolate in new contexts is paramount since for example different patients will have different characteristics and heart geometries. This lead us to perform two types of tests on our model, one with scars with different shapes while training was conducted only with rectangular ones, and one with multiple onsets while training considered only one.

Therefore, regarding different scar shapes, we evaluated our model with triangular, circular and complex scars (see Figure 2.13). Table 2.6 shows that the model performs well on the different shapes. The errors are slightly larger than for the rectangular scars used for training, but remains satisfyingly low. They however increase for long term predictions (24 ms here). Figure 2.13 illustrates the behaviour of the model for typical test sequences.

As for multiple onsets, the model shows good results for forecasting of multiple depolarisation waves on one cardiac slab tissue (Figure 2.14), which is essential for ventricular tachycardia simulation. As one can see from Table 2.7, relative mean-squared error is larger for multiple onsets than for one onset but still acceptable.

Generalisation ability: various conduction velocities To estimate the ability of the model to extrapolate to new conduction velocities of the cardiac tissue, we performed tests with various cardiac slab conductivities σ. The tests have been performed with sigma values used for training ($\sigma \in \{1, 2, 3, 4, 5\}$), and sigma values sampled outside the training set.

Table 2.6: Relative mean-squared error (MSE) of transmembrane potential forecasting in presence of scars of various forms for different forecasting horizons (cardiac slab conductivity $\sigma = 2$).

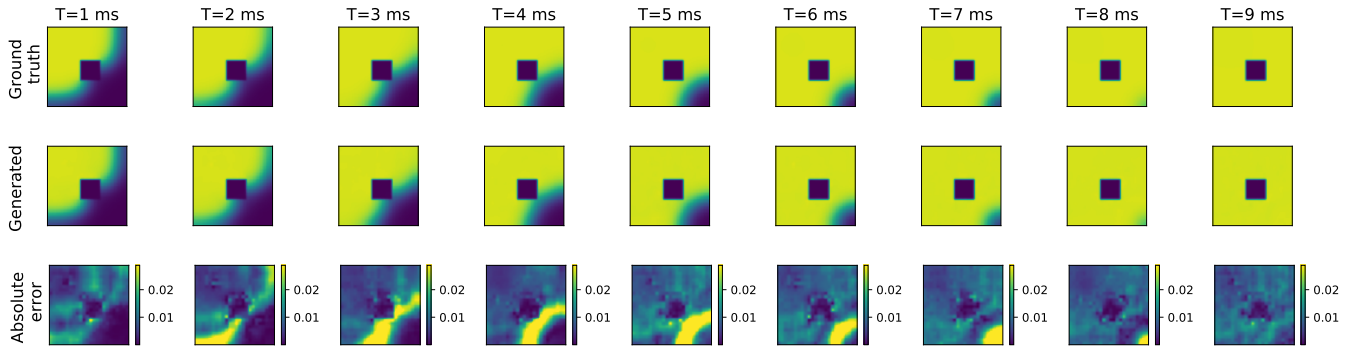
	MSE (6 ms)	MSE (12 ms)	MSE (24 ms)
Rectangular shape	1.8×10^{-4}	4.45×10^{-4}	$6,8 \times 10^{-4}$
Triangular shape	3.1×10^{-4}	8×10^{-4}	1.36×10^{-3}
Circular shape	2.7×10^{-4}	8.2×10^{-4}	3.4×10^{-3}
Complex shape	4.6×10^{-4}	1.9×10^{-3}	6.36×10^{-3}

Table 2.7: Relative mean-squared error (MSE) of potential forecasting in presence of multiple onsets and scar of rectangular form for different forecasting horizons (cardiac slab conductivity $\sigma = 2$).

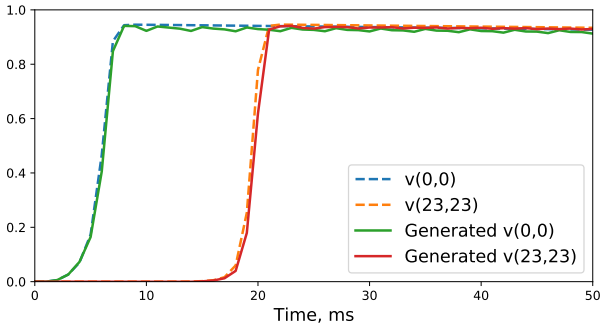
	MSE (6 ms)	MSE (12 ms)	MSE (24 ms)
One Onset	1.8×10^{-4}	4.45×10^{-4}	$6,8 \times 10^{-4}$
Multiple Onsets	4.7×10^{-4}	5.8×10^{-4}	6.9×10^{-4}

Table 2.8: Relative mean-squared error (MSE) of potential forecasting in presence of various conduction velocities of cardiac slab and scar of rectangular form for different forecasting horizons.

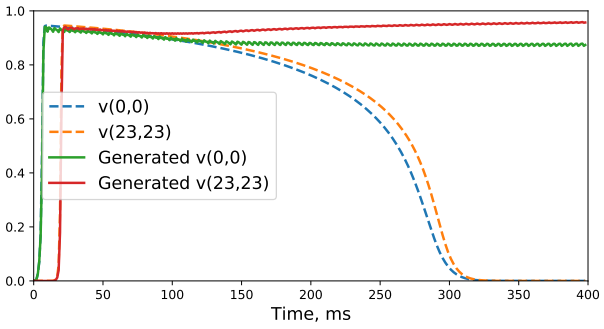
	MSE (6 ms)	MSE (12 ms)	MSE (24 ms)
$\sigma = 0.7$	4.65×10^{-4}	3.95×10^{-3}	$1,9 \times 10^{-2}$
$\sigma = 2$	1.8×10^{-4}	4.45×10^{-4}	$6,8 \times 10^{-4}$
$\sigma = 2.5$	3.5×10^{-4}	1.4×10^{-3}	$1,6 \times 10^{-4}$
$\sigma = 6$	2×10^{-3}	4.7×10^{-3}	3×10^{-3}



(a)



(b)



(c)

Figure 2.12: (a) Results of the trained model (9 ms of forecast, cardiac slab conductivity $\sigma = 2$). (b,c) Transmembrane potential graph at the leftmost upper point (0,0) and the rightmost bottom point (23,23) in the slab with different forecasting horizons.

As shown in Figure 2.15, the model is still able to generalise to unseen conditions, such as scars of various shapes and multiple onsets. Quantitative results provided in Table 2.8 shows that the model achieves a good precision in forecasting depolarisation waves in cardiac tissue slabs for any conductivity even when different from the values of the training dataset.

Limitations Although our approach can achieve compelling results in many cases, there are still limitations. For example, as shown in Figure 2.16, our model does not work properly on thin scars (thickness less than 2 pixels) and diffuses transmembrane potential through the scar.

Moreover, as observed before, the current model has been trained only to model depolarization of the cardiac slab tissue and cannot predict its repolarisation (see Figure 2.12c). This is left for future work.

Finally, it is also important to recall that the dynamics studied here are simplified quite importantly when compared to real data.

2.8 Discussion and Conclusion

In the machine learning community, the forecasting problem is often seen as learning a neural network mapping consecutive states in time. By parametrizing directly the unknown differential equation, this work takes an alternate approach and uses the neural network to express the rate of change of the states instead. Our results, including the ablations, show that this is considerably simpler for the network, and is in fact the natural way to model time varying processes. This would also allow to accommodate irregularly acquired observations and could also allow interpolation between observations.

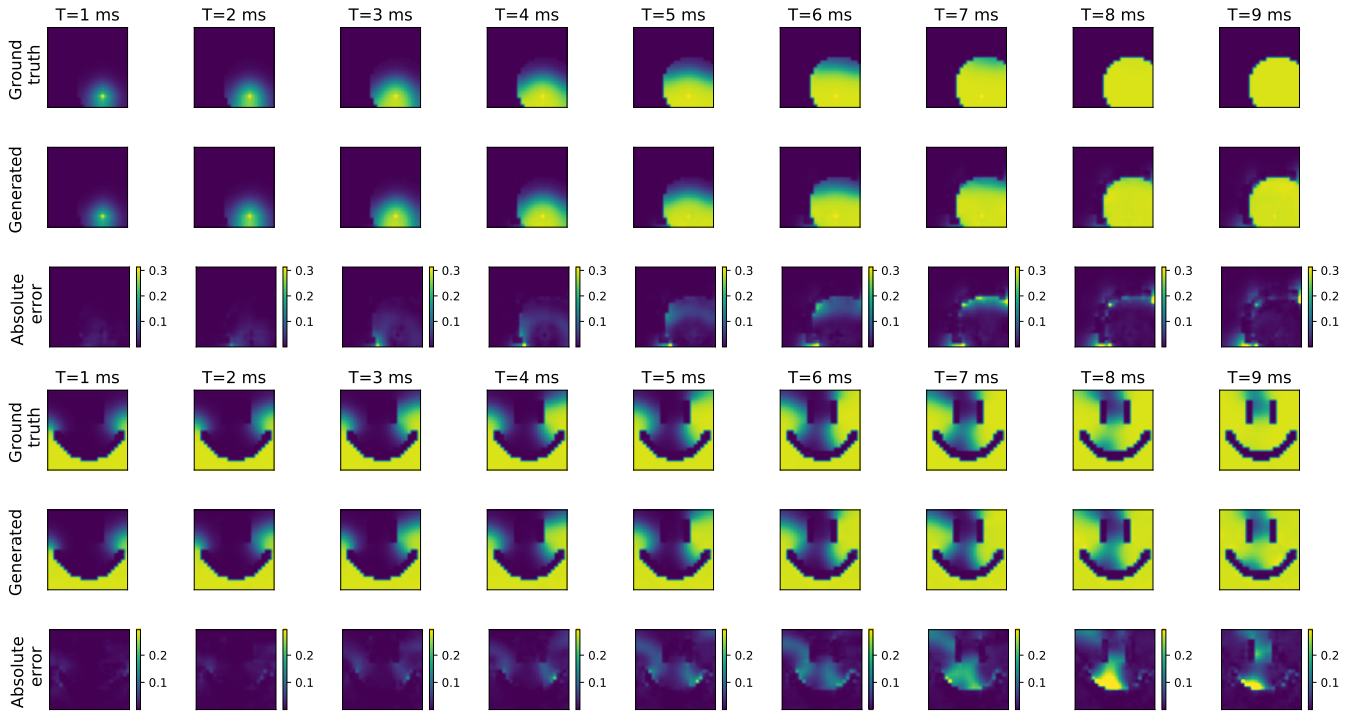


Figure 2.13: Results of the trained model on scar with circular (top three rows) and complex (bottom three rows) shape (9 ms of forecast, cardiac slab conductivity $\sigma = 2$).

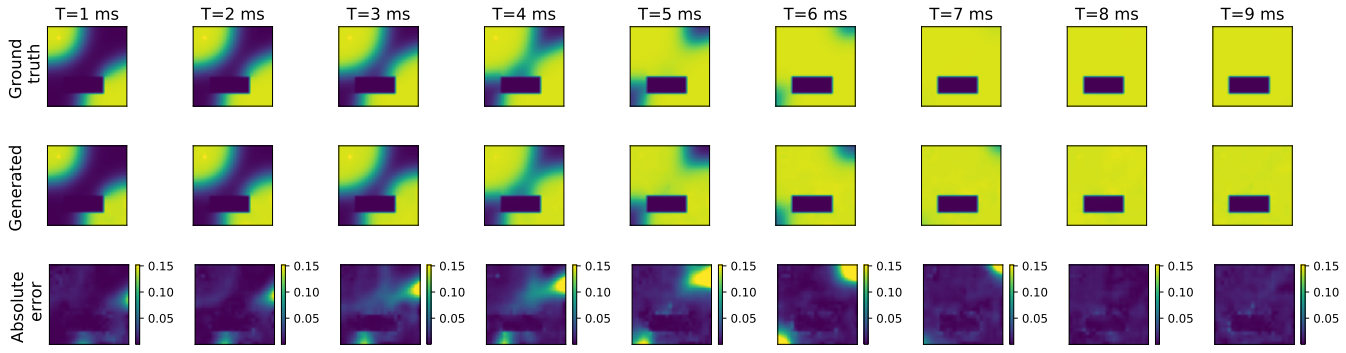


Figure 2.14: Results of the trained model with two stimulation currents applied on different pixels and at different times (9 ms of forecast, cardiac slab conductivity $\sigma = 2$).

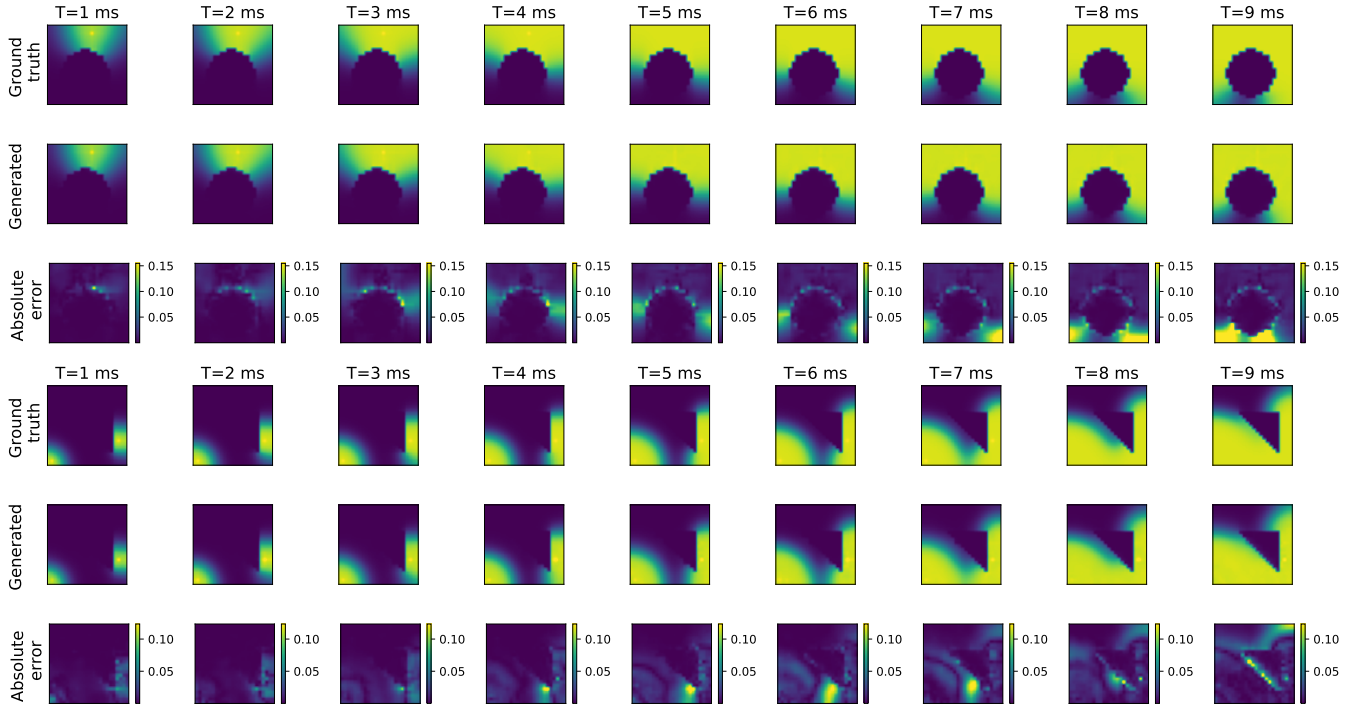


Figure 2.15: Results of the trained model on scar with circular shape and cardiac slab conductivity $\sigma = 3.8$ (top three rows), and on scar with triangular shape, two onsets and cardiac slab conductivity $\sigma = 1.5$ (bottom three rows).

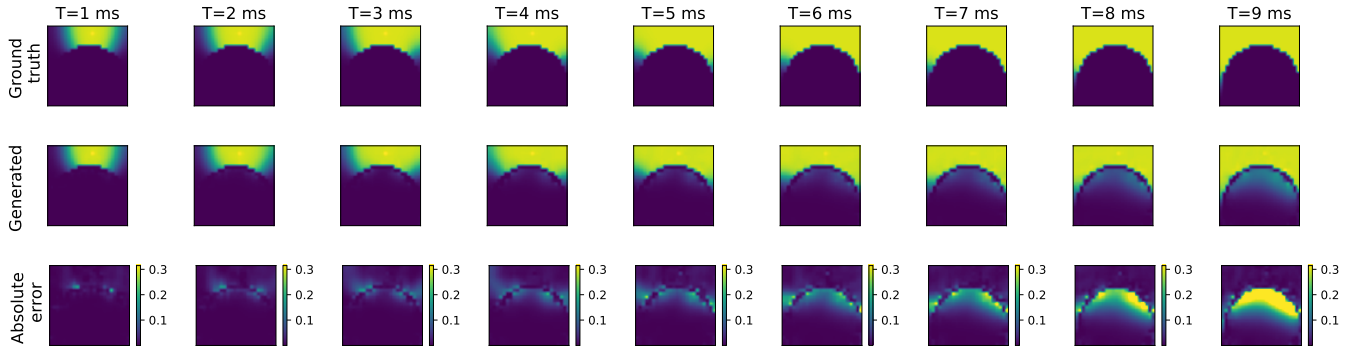


Figure 2.16: Results of the trained model on thin scar with circular shape.

By placing our work in the partially observable context, we have faced the indeterminacy issue of the hidden part of the state. This has led us to explore various ways of constraining the hidden dynamics the extensive study of which then allows us to tackle more realistic settings such as the forecasting of SST dynamics. It also opens up interesting directions for future exploration, including in order to make hidden states more interpretable.

Among the limitations of this work, we have also restricted ourselves to a linear $\mathcal{H}$, and it would be interesting to see how our algorithms work for operators with a more complicated structure, including operators which obstruct the state spatially. Moreover, we have restricted ourselves to the stationary hypothesis while, as we can see through the Glorys2v4 example, real-world processes, when looked at from a local point of view, typically when not all exterior forces are factored into the model, aren't. These are interesting directions for future work.

Overall, the proposed neural models we propose are surprisingly robust and versatile, allowing to build powerful forecasting tools without prior knowledge of the dynamics. This should pave the way for new methods for integrating prior physical knowledge, *e.g.* by imposing constraints directly on the modeled evolution term. The following chapter explores two settings for doing so.

Chapter 3

Improving Generalization and Robustness for Neural Differential Models of Dynamical Systems

In this chapter, we present two models, APHYNITY and *LEADS*, which can be seen as extensions of the work presented in the previous chapter. They allow us to show, for two different settings and corresponding tasks, that it is possible to improve substantially the neural modelling of differential equations by constraining them accordingly.

The content of this chapter essentially covers the content of the corresponding two publications:

- *Augmenting Incomplete Physical Models with Deep Networks for Complex Dynamics Forecasting*, Y Yin*, J Dona*, V Le Guen*, E de Bézenac*, I Ayed*, P Gallinari, N Thome, ICLR 2021, Oral presentation;
- *Learning Dynamical Systems across Environments*, Y Yin, I Ayed, E de Bézenac, P Gallinari, NeurIPS 2021, Poster.

An application of the APHYNITY framework to the Electro-Physiological modelling of heart dynamics has also been conducted and, alhough it is not presented here, it can be found as

- *Deep Learning for Model Correction in Cardiac Electrophysiological Imaging*, V Kashtanova, I Ayed, A Arrieula, M Potse, P Gallinari, M Sermesant, Poster, MIDL 2022.

3.1 Introduction

As shown in Chapter 2, data-driven approaches offer an interesting alternative and complement to physical-based methods for modeling the dynamics of complex systems. They are particularly promising in a wide range of settings: e.g. 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 [44, 69, 50, 169, 217] as well as the work we presented earlier in this manuscript. It is already being applied in different scientific disciplines (see e.g. [276] for a recent survey) and could help accelerate scientific discovery to address challenging domains such as climate [224] or health [95].

This chapter is about extending an approach of the kind we presented in Chapter 2, in slightly different contexts, in order to improve the extrapolation properties of the generic models presented earlier. More specifically, we aim to see whether it is possible:

- to use existing prior knowledge about a system in order to improve learning, both in terms of forecasting accuracy and of parameter identification?

- to improve learning in cases where the data is not i.i.d. and is generated by distinct differential equations with common characteristics?

In both cases, we want our model to go further than simply fitting a neural model to a dataset of trajectories as we want to endow it with stronger properties. This obviously needs a more involved approach and learning framework. After a section discussing general considerations common to both, we present two general models answering those two questions: APHYNITY for the first and LEADS for the second.

In the following, we will thus study dynamics of the general form introduced in Chapter 2, which are driven by an equation of the form:

$$\frac{\mathrm{d}X_t}{\mathrm{d}t} = F(X_t) \tag{3.1}$$

defined over a finite time interval $[0, T]$, where the state X is either vector-valued, i.e. we have $X_t \in \mathbb{R}^d$ for every t (e.g. pendulum and Lotka-Volterra equations), or X_t is a d-dimensional vector field over a spatial domain $\Omega \subset \mathbb{R}^k$, with $k \in \{2, 3\}$, i.e. $X_t(x) \in \mathbb{R}^d$ for every $(t, x) \in [0, T] \times \Omega$.

We assume access to a set of observed trajectories $\mathcal{D} = \{X_\cdot : [0, T] \to \mathcal{A} \mid \forall t \in [0, T], \frac{\mathrm{d}X_t}{\mathrm{d}t} = F(X_t)\}$, where $\mathcal{A}$ is the set of X values (either $\mathbb{R}^d$ or vector field). Notice in particular that the dataset here is assumed to contain fully observable states, contrary to the previous chapter.

3.2 Preliminaries

This section discusses two important shared aspects of the optimization of the APHYNITY and *LEADS* models presented in the following: our choice of supervising with losses on trajectories and the way we solve the neurally parametrized optimization problems under constraints which define both models using the general methodology presented in Section 1.3 of Chapter 1.

3.2.1 Trajectory-based Supervision

Contrary to the setting of Chapter 2 where only partial observations were available, in both of the models presented in this chapter, we assume that full states are available for supervision. It woud therefore be possible to retrieve the unknown Fs driving the considered differential equations by solving the regression problem where the target values are computed as finite difference approximations of the actual derivative.

This is indeed a possibility which we chose not to follow as we opted for optimizing over trajectories instead of derivatives. In other words, instead of computing derivatives over the dataset then regressing over those values to find the function F, we try to find F such that the trajectories it induces as solutions fit correctly to the corresponding ones from the dataset. The following is a heuristic reasoning explaining why this choice was done. It is confirmed by experiments as shown in the Ablation Study conducted in Section 3.3.3.

More formally, let us consider the following optimization problem:

$$\begin{aligned} \underset{F \in F}{min} \quad & \mathbb{E}_{X \sim \mathcal{D}} \left[L(X, \widetilde{X}^g) \right] \\ \text{s.t.} \quad & \forall g \in \mathcal{I},\ \widetilde{X}^g_0 = g \text{ and } \forall t,\ \frac{\mathrm{d}\widetilde{X}^g_t}{\mathrm{d}t} = F(\widetilde{X}^g_t) \end{aligned} \tag{3.2}$$

where L is a loss over trajectories and $\mathcal{I}$ is the domain of initial states. Here, $\widetilde{X}$ represents the solution defined by F which we want to be close from X which is the corresponding actual trajectory sampled from the dataset.

In the continuous, non realistic, setting where the data is available at all times t, for many reasonable losses L, one can find an appropriate loss $\widetilde{L}$ such that this problem is in fact equivalent to the following one:

$$\min_{F \in F} \quad \mathbb{E}_{X \sim \mathcal{D}} \left[\widetilde{L}(\frac{\mathrm{d}X_t}{\mathrm{d}t}, F(X_t)) \right] \tag{3.3}$$

where the supervision is done directly over the derivatives $\frac{\mathrm{d}X_t}{\mathrm{d}t}$, obtained through finite-difference schemes from the dataset $\mathcal{D}$[1].

However, in practice, data is only available at discrete times with a certain time resolution. While eq. 3.3 is indeed equivalent to eq. 3.2 in the continuous setting, in the practical discrete one, error does not propagate in the same way: For eq. 3.2 it is controlled over integrated trajectories while for eq. 3.3 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:

- The use of finite differences schemes to estimate F as is done in eq. 3.3 necessarily induces a non-zero discretization error which is bounded below by the available temporal resolution.

- 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 $\widetilde{F} = F + \epsilon$ be the function estimated from the sampled points with an error ϵ such that $\|\epsilon\|_\infty \leq \alpha$. Denoting $\widetilde{X}$ the corresponding trajectory generated by $\widetilde{F}$, we then have, for all $X \in \mathcal{D}$:

$$\forall t, \ \frac{\mathrm{d}(X - \widetilde{X})_t}{\mathrm{d}t} = F(X_t) - F(\widetilde{X}_t) - \epsilon(\widetilde{X}_t)$$

Integrating over a finite time interval $[0, T]$ and using the triangular inequality as well as the mean value inequality, supposing that F has uniformly bounded spatial derivatives:

$$\forall t \in [0, T], \ \|(X - \widetilde{X})_t\| \leq \|\nabla F\|_\infty \int_0^t \|X_s - \widetilde{X}_s\| + \alpha t$$

which, using a variant of the Grönwall lemma, gives us the inequality:

$$\|X - \widetilde{X}\|_\infty \leq \frac{\alpha}{\|\nabla F\|_\infty} (\exp(\|\nabla F\|_\infty T) - 1)$$

When α tends to 0, we do recover, as expected, the true trajectories X. However, as α is bounded away from 0 by the available temporal resolution, this inequality gives a rough estimate of the way $\widetilde{X}$ diverges from them, and it can be an equality in many cases. This exponential behaviour explains our choice of a trajectory-based optimization and we do obtain better performing model in practice with this approach.

[1] Here the initial conditions can be omitted as we find F directly from finite difference of the trajectory.

3.2.2 Adaptively constrained optimization

Another common aspect to both models presented in the following sections of this chapter is that we are solving constrained optimization problems, where we minimize a certain function over the parametrized function while wanting this function to fit the trajectories from the studied dataset.

Considering a global trajectory loss $\mathcal{L}_{traj}$ and the loss being minimized is $\mathcal{C}$,[2] we can formulate the family of optimization problems we consider in the following simplified way:

$$\begin{aligned} \min_{\theta} \quad & \mathcal{C}(\theta) \\ \text{s.t.} \quad & \forall t,\ \frac{\mathrm{d}\widetilde{X}_t^\theta}{\mathrm{d}t} = F^\theta(\widetilde{X}_t^\theta), \\ & \mathcal{L}_{traj}(\theta) = 0 \end{aligned}$$

There is no generic tool to solve this kind of optimization problem and, because of the constraint over $\mathcal{L}_{traj}$, Theorem 1.1 of Chapter 1 doesn't apply here. However, using this last result, it is possible to use gradient descent for the relaxed version of this problem:

$$\begin{aligned} \min_{\theta} \quad & \mathcal{L}_\lambda(\theta) = \mathcal{C}(\theta) + \lambda \mathcal{L}_{traj}(\theta) \\ \text{s.t.} \quad & \forall t,\ \frac{\mathrm{d}\widetilde{X}_t^\theta}{\mathrm{d}t} = F^\theta(\widetilde{X}_t^\theta) \end{aligned}$$

This last form fits into the framework defined in Chapter 1 where we have taken:

- D_0 as the initial state and $D_{>0}$ as the future states of the system;
- $\mathcal{J}$ as in eq. 1.4 with $\alpha = \lambda$, $\beta = 0$, $\gamma = 1$ and j as the squared L^2 loss over Ω.

Obviously, for a given value of λ, this is not equivalent to the original problem. However, when λ increases, $\mathcal{L}_{traj}(\theta)$ should become closer to 0 and thus the constraint becomes approximately verified.

We could then simply choose a sufficiently high value for our purposes and solve the corresponding problem. This is possible of course but, in general, solving the problem becomes harder with higher values of λ and there is a risk of not obtaining a good enough minimizer.

A solution is then to pick an increasing sequence $(\lambda_i)_i$ and then solve iteratively the problems associated with losses $\mathcal{L}_{\lambda_i}$. This means that, for each $i > 1$, we use the optimal $\theta^\star_{i-1}$ associated with $\mathcal{L}_{\lambda_{i-1}}$ as the initial θ when solving the problem associated with $\mathcal{L}_{\lambda_i}$. This helps as, with increasing i, each $\theta^\star_i$ should become closer to the target $\theta^\star$ so that solving the problem with a higher λ_{i+1} becomes easier.

The remaining question is then about designing the increasing sequence $(\lambda_i)_i$ which should increase as fast as possible in order to have faster global convergence, in terms of the number of iterative steps, but not too quickly so that the convergence of each step stays reasonably quick. It is of course possible to handcraft such a sequence, e.g. by increasing the values linearly, exponentially, etc. but we chose to update the value of λ depending on the current value of $\mathcal{L}_{traj}$: when large, we can make bigger increases as we are still far from satisfying the constraint; when small, the constraint is nearly met so that we can and have to make smaller steps. This translates into an update of the form:

$$\lambda_{i+1} = \lambda_i + \tau \mathcal{L}_{traj}(\theta_{i+1})$$

which is akin to a gradient ascent over the λs.

In fact, this method we chose belongs to the broader family of *Lagrange Multipliers Methods* of constrained optimization algorithms for which a good presentation can be found in [36].

In our case, while convergence is not guaranteed because of the non-convexity of our neural parametrization, in practice, a few steps of increasing λ were in general sufficient to obtain satisfying solutions.

[2] $\mathcal{C}$ is often chosen as a norm over the parametrized F in our work.

3.3 *APHYNITY*: Augmenting Physical Models with Deep Networks for Complex Dynamics Forecasting

While purely data-driven approaches are arguably insufficient when modelling complex phenomena in real world settings, as they ignore the wealth of scientific knwoledge available in the relevant fields, so are standard physical models which tend to ignore some aspects when constructing the model in order to obtain a tractable representation of the studied phenomenon.

In this section, 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. In order to do so, 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 3.1 on the pendulum problem, is to augment the physical model when—and only when—it falls short.

In our framework, the dynamics are thus decomposed into two components: a physical component accounting for the dynamics for which we have some prior knowledge, and a data-driven one accounting for errors of the physical model. The learning problem is then carefully formulated such that the physical model explains as much of the data as possible, while the data-driven component only describes information that cannot be captured otherwise.

This allows to show that the decomposition exists and is unique under mild assumptions and ensures some interpretability while benefiting generalization. Experiments made on three important use cases, each representative of a different family of phenomena, i.e. reaction-diffusion equations, wave equations and the non-linear damped pendulum, show that APHYNITY can efficiently leverage approximate physical models to accurately forecast the evolution of the system and correctly identify relevant physical parameters.

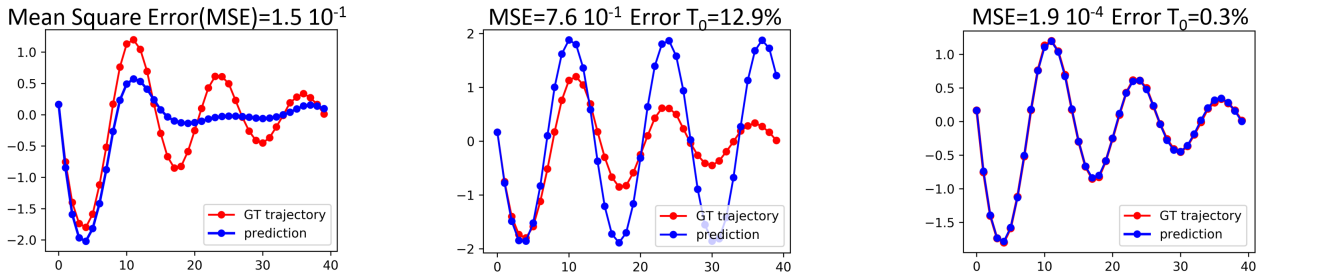
Our contributions are the following:

- 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 3.3.2) for the decomposition given mild conditions, and show that this formulation ensures interpretability and benefits generalization.

- We propose an algorithm and 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 3.3.3). 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. In particular, APHYNITY is robust both to cases where full knowledge is available (i.e. it doesn't hurt when it is unnecessary) and to cases where very little prior knowledge is available.

3.3.1 Related work

Modeling and forecasting complex dynamical systems is a major challenge in domains such as environment and climate [228], health science [57], and in many industrial applications [262]. Model Based (MB) approaches typically rely on partial or ordinary differential equations (PDE/ODE) and stem from a deep understanding of the underlying physical phenomena. Machine learning (ML) and deep learning methods are more prior agnostic yet have become state-of-the-art for several spatio-temporal prediction tasks [242, 272, 207, 74] and connections have been drawn between deep



(a) Data-driven Neural ODE (b) Simple physical model (c) Our APHYNITY framework

Figure 3.1: Predicted dynamics for the damped pendulum vs. ground truth (GT) trajectories $\frac{\mathrm{d}^2\theta}{\mathrm{d}t^2} + \omega_0^2 \sin\theta + \alpha\frac{\mathrm{d}\theta}{\mathrm{d}t} = 0$. We show that in (a) the data-driven approach [51] 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).

architectures and numerical ODE solvers, e.g. neural ODEs [51] and our work presented in previous chapters.

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 [99], computational constraints prescribing a fine grain modeling of the system [17], unknown external factors, forces and sources which are present [146].

Correction in data assimilation Prediction under approximate physical models has been tackled by traditional statistical calibration techniques, which often rely on Bayesian methods [211]. Data assimilation techniques, e.g. the Kalman filter [135, 26], 4D-var [63], prediction errors are modeled probabilistically and a correction using observed data is applied after each prediction step. Similar residual correction procedures are commonly used in robotics and optimal control [52, 157]. However, these sequential (two-stage) procedures prevent the cooperation between prediction and correction. Besides, in model-based reinforcement learning, model deficiencies are typically handled by considering only short-term rollouts [130] or by model predictive control [195]. 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.

Augmented physical models This task of combining physical models with machine learning (*gray-box or hybrid* modeling) was actually first explored from the 1990's: [213, 260, 226] use neural networks to predict the unknown parameters of physical models. The challenge of proper MB/ML cooperation was already raised as a limitation of gray-box approaches but not addressed. Moreover these methods were evaluated on specific applications with a residual targeted to the form of the equation. In the last few years, there has been a renewed interest in deep hybrid models bridging data assimilation techniques and machine learning to identify complex PDE parameters using cautiously constrained forward model [44, 169]. [220, 244] use NNs as implicit methods for solving PDEs, [239] learn spatial differences with a graph network, [264] introduce continuous convolutions for fluid simulations, [70] learn the velocity field of an advection-diffusion system, [106, 53] enforce conservation laws in the network architecture or in the loss function. Recently, some approaches have specifically targeted the MB/ML cooperation. HybridNet [167] and PhICNet [233] both use data-driven networks to learn additive perturbations or source terms to a given PDE. The former considers the favorable context where the perturbations can be accessed, and the latter the special case of additive noise

on the input. [269, 181] propose several empirical fusion strategies with deep neural networks but lack theoretical groundings. PhyDNet [147] tackles augmentation in partially-observed settings, but with specific recurrent architectures dedicated to video prediction. Crucially, all the aforementioned approaches do not address the issues of uniqueness of the decomposition or of proper cooperation for correct parameter identification. Besides, we found experimentally that this vanilla cooperation is inferior to the APHYNITY learning scheme in terms of forecasting and parameter identification performances (see experiments in Section 3.3.3). Designing a general method for combining MB and ML approaches is still a widely open problem, and a clear problem formulation is lacking [224] and our work aims to propose an approach bridging the gap between the two worlds.

3.3.2 The APHYNITY Model

In this section, the unknown F has $\mathcal{A}$ as domain and we only assume that $F \in \mathcal{F}$, with $(\mathcal{F}, \|\cdot\|)$ a normed vector space.

- *Decomposing dynamics into physical and augmented terms*

We consider in this work the common situation where incomplete information is available on the dynamics. This available information is represented in our case in the form of a family of ODEs or PDEs characterized by their temporal evolution $F_p \in \mathcal{F}_p \subset \mathcal{F}$. The core idea here is twofold:

- This set should represent the state-of-the-art scientific knowledge about the studied dynamics. In other words, we know $\mathcal{F}_p$ contains a model which is quite close to the unknown F.
- $\mathcal{F}_p$ is small enough so that it is informative. In general, we assume that functions in $\mathcal{F}_p$ are characterised by a small number of parameters θ_p.

The APHYNITY framework aims to leverage the knowledge of $\mathcal{F}_p$ while mitigating the approximations induced by this simplified model through its combination with a data-driven component. Indeed, $\mathcal{F}$ being a vector space, we can write:

$$F = F_p + F_a$$

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 parametric family, but the physical parameters (e.g. propagation speed for the wave equation) are unknown and need to be estimated from data. The parameters of both F_p and F_a are estimated by fitting the trajectories from $\mathcal{D}$.

It can easily be seen that 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 to act as a complement, so as to model only the information that cannot be captured by the physical prior. In particular, 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 null; 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 the 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:

$$\min_{F_p \in \mathcal{F}_p, F_a \in F} \quad \|F_a\| \quad \text{subject to} \quad \forall X \in \mathcal{D}, \forall t, \frac{\mathrm{d}X_t}{\mathrm{d}t} = (F_p + F_a)(X_t) \tag{3.4}$$

• *Existence and uniqueness*

A first key question is whether the minimum in eq. 3.4 is indeed well-defined, in other words whether there always exists a decomposition with a minimal norm of F_a. The answer actually depends on the geometry of $\mathcal{F}_p$, and is formulated in the following proposition:

Proposition 3.1 (*Existence of a minimizing pair*)

If $\mathcal{F}_p$ is a proximinal set, there exists a decomposition minimizing eq. 3.11.

Proof: The idea is to reconstruct the full functional from the trajectories of $\mathcal{D}$. By definition, $\mathcal{A}$ is the set of points reached by trajectories in $\mathcal{D}$ so that:

$$\mathcal{A} = \{x \in \mathbb{R}^d \mid \exists X_\cdot \in \mathcal{D}, \exists t,\ X_t = x\}$$

Then let us define a function $F^{\mathcal{D}}$ in the following way: For $a \in \mathcal{A}$, we can find $X_\cdot \in \mathcal{D}$ and t_0 such that $X_{t_0} = a$. Differentiating X at t_0, which is possible by definition of $\mathcal{D}$, we take:

$$F^{\mathcal{D}}(a) = \left.\frac{\mathrm{d}X_t}{\mathrm{d}t}\right|_{t=t_0}$$

For any (F_p, F_a) satisfying the constraint in eq. 3.4, we then have that $(F_p + F_a)(a) = \frac{\mathrm{d}X_t}{\mathrm{d}t}_{|t_0} = F^{\mathcal{D}}(a)$ for all $a \in \mathcal{A}$. Conversely, any pair such that $(F_p, F_a) \in \mathcal{F}_p \times \mathcal{F}$ and $F_p + F_a = F^{\mathcal{D}}$, verifies the constraint. Thus we have the equivalence between eq. 3.4 and the metric projection formulated as:

$$\min_{F_p \in F_p} \quad \left\| F^{\mathcal{D}} - F_p \right\| \tag{3.5}$$

If $\mathcal{F}_p$ is proximinal, the projection problem admits a solution which we denote $F_p^\star$. Taking $F_a^\star = F^{\mathcal{D}} - F_p^\star$, we have that $F_p^\star + F_a^\star = F^{\mathcal{D}}$ so that $(F_p^\star, F_a^\star)$ verifies the constraint of eq. 3.4. Moreover, if there is (F_p, F_a) satisfying the constraint of eq. 3.4, we have that $F_p + F_a = F^{\mathcal{D}}$ by what was shown above and $\|F_a\| = \|F^{\mathcal{D}} - F_p\| \geq \|F^{\mathcal{D}} - F_p^\star\|$ by definition of $F_p^\star$. This shows that $(F_p^\star, F_a^\star)$ is minimal. □

Proximinality is a mild condition which, as can be seen 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. 3.4 leads to a unique decomposition into a physical and a data-driven component. The following result provides guarantees on the uniqueness of the decomposition with an additional albeit still quite mild condition.

Proposition 3.2 (*Uniqueness of the minimizing pair*)

If $\mathcal{F}_p$ is a Chebyshev set, eq. 3.11 admits a unique minimizer. The F_p in this minimizer pair is the metric projection of the unknown F onto $\mathcal{F}_p$.

Proof: Going back to the proof of Proposition 3.1, if $\mathcal{F}_p$ is a Chebyshev set, by uniqueness of the projection, if $F_p \neq F_p^\star$ then $\|F_a\| > \|F_a^\star\|$. Thus the minimal pair is unique. □

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

Propositions 3.3 and 3.2 provide, under mild conditions, the theoretical guarantees for the APHYNITY formulation to infer the correct MB/ML decomposition, thus enabling us both to recover the proper physical parameters of the system and to accurately forecast its future states.

Reminder on proximinal and Chebyshev sets Let us start by defining proximinal and Chebyshev sets (those definitions are taken from [92]):

Definition 3.1

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 3.2

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 more general spaces, it is a weaker condition: Boundedly compact sets verify this property for example. In Euclidean spaces, Chebyshev sets are simply the 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 exists examples of non-convex Chebyshev sets, a famous one being presented in [133] for a non-complete inner-product space.

Given the importance of this topic in approximation theory, finding 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, and in particular any finite dimensional subspace, is Chebyshev.
- In a Hilbert space, every closed convex set is Chebyshev.

• *Solving APHYNITY with deep neural networks*

In the following, both terms of the decomposition are parametrized and are denoted as $F_p^{\theta_p}$ and $F_a^{\theta_a}$. Solving APHYNITY then amounts to 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 explore all of $\mathcal{F}$: we thus use deep neural networks, which have shown promising performances for the approximation of differential equations [220, 18].

When learning the parameters of $F_p^{\theta_p}$ and $F_a^{\theta_a}$, we have access to a finite dataset of trajectories discretized with a given temporal resolution Δt: $\mathcal{D}_{\text{train}} = \{(X^{(i)}_{k\Delta t})_{0 \leq k \leq \lfloor \frac{T}{\Delta t} \rfloor}\}_{1 \leq i \leq N}$. Solving eq. 3.4 requires estimating the state derivative $\frac{\mathrm{d}X_t}{\mathrm{d}t}$ appearing in the constraint term. One solution is to approximate this derivative using e.g. finite differences as in [44, 106, 65]. 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 earlier in Section 3.2.1.

We propose instead to solve eq. 3.4 using an integral trajectory-based approach as argued before: we compute $\widetilde{X}^i_{k\Delta t, X_0}$ from an initial state $X_0^{(i)}$ using the current $F_p^{\theta_p} + F_a^{\theta_a}$ dynamics, then enforce the constraint $\widetilde{X}^i_{k\Delta t, X_0} = X^i_{k\Delta t}$. This leads to our final objective function on (θ_p, θ_a):

$$\min_{\theta_p, \theta_a} \quad \left\| F_a^{\theta_a} \right\| \quad \text{subject to} \quad \forall i, \forall k, \widetilde{X}^{(i)}_{k\Delta t} = X^{(i)}_{k\Delta t} \tag{3.6}$$

where $\widetilde{X}^{(i)}_{k\Delta t} := \text{SolveODE}(X_0^{(i)}, F_p^{\theta_p} + F_a^{\theta_a}, k\Delta t)$ is the prediction at $k\Delta t$ of an ODE solver.

Algorithm 2: APHYNITY

Initialization: $\lambda_0 \geq 0, \tau_1 > 0, \tau_2 > 0$;
for *epoch* $= 1 : N_{epochs}$ **do**
  for *iter in* $1 : N_{iter}$ **do**
    for *batch in* $1 : B$ **do**
      $\theta_{j+1} = \theta_j - \tau_1 \nabla \left[\lambda_j \mathcal{L}_{traj}(\theta_j) + \|F_a\| \right]$
    end
  end
  $\lambda_{j+1} = \lambda_j + \tau_2 \mathcal{L}_{traj}(\theta_{j+1})$
end

In our setting, where we consider situations for which $F_p^{\theta_p}$ only partially describes the physical phenomenon, this coupled MB + ML formulation leads to different parameter estimates than using the MB formulation alone.

As expected, our experiments show that using this formulation not only improves forecasting but also leads to a better identification of the physical parameters θ_p than when fitting the simplified physical model $F_p^{\theta_p}$ alone. Indeed, with only an incomplete knowledge of the physics, the estimator for θ_p is hindered by the non modelled dynamics. This aspect and others are explored in the experiments of the following section 3.3.3.

- *Parameter estimation in incomplete physical models*

Classically, when a set $\mathcal{F}_p \subset \mathcal{F}$ summarising 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:

$$\begin{aligned} \min_{F_p \in F_p} \quad & \mathbb{E}_{X \sim \mathcal{D}} L(\widetilde{X}^g, X) \\ \text{s.t.} \quad & \forall g \in \mathcal{I},\ \widetilde{X}_0^g = g \text{ and } \forall t,\ \frac{\mathrm{d}\widetilde{X}_t^g}{\mathrm{d}t} = F_p(\widetilde{X}_t^g) \end{aligned} \tag{3.7}$$

where L is a discrepancy measure between trajectories. Recall that $\widetilde{X}^{X_0}$ is the result trajectory of an ODE solver taking X_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 estimation of the function becomes easier, there is then a residual part which is left unexplained and this can be a non negligible issue in at least two ways:

- When $F \notin \mathcal{F}_p$, the loss is strictly positive at the minimum. This means that reducing the space of functions $\mathcal{F}_p$ makes us lose in terms of accuracy.[3]
- The obtained function F_p might not even be the most meaningful function from $\mathcal{F}_p$ as it would try to capture phenomena which are not explainable with functions in $\mathcal{F}_p$, 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 $\mathcal{F}_p$ but not the dampening, the estimated period will account for the dampening and will thus be biased.

This is confirmed in Section 3.3.3: the incomplete physical models augmented with APHYNITY get different and experimentally better physical identification results than the physical models alone.

[3]This is true in theory, although not necessarily in practice when F overfits a small dataset.

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:

$$\frac{\mathrm{d}^2\theta}{\mathrm{d}t^2} + \omega_0^2\theta + \alpha\frac{\mathrm{d}\theta}{\mathrm{d}t} = 0$$

We take the same notations as before and parametrize the simplified physical models as:

$$F_p^a : X \mapsto (\frac{\mathrm{d}\theta}{\mathrm{d}t}, -a\theta)$$

where $a > 0$ corresponds to ω_0^2. The corresponding solution for an initial state X_0, which we denote X^a, can then written explicitly as:

$$\theta_t^a = \theta_0 \cos\sqrt{a}t$$

Let us consider damped pendulum solutions X written as:

$$\theta_t = \theta_0 e^{-t}\cos t$$

which corresponds to:

$$F : X \mapsto (\frac{\mathrm{d}\theta}{\mathrm{d}t}, -2(\theta + \frac{\mathrm{d}\theta}{\mathrm{d}t}))$$

It is then easy to see that the estimate of a with the physical model alone can be obtained by minimizing:

$$\int_0^T |e^{-t}\cos t - \cos\sqrt{a}t|^2$$

This expression depends on T. Therefore, depending on the chosen time interval and the way the integral is discretized, it will almost always give biased estimates. In other words, the estimated value of a will not give us the solution $t \mapsto \cos t$ with the right frequency.

On the other hand, for a given a, in the APHYNITY framework, the residual must be equal to:

$$F_r^a : X \mapsto (0, (a-2)\theta - 2\frac{\mathrm{d}\theta}{\mathrm{d}t})$$

in order to satisfy the fitting constraint. Here a corresponds to $1 + \omega_0^2$ not to ω_0^2 as in the simplified case. Minimizing its norm, we obtain $a = 2$ which gives us the desired solution:

$$\theta_t = \theta_0 e^{-t}\cos t$$

with the right period.

3.3.3 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 neuro-biology (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.

• *Experimental setting*

We describe the three families of equations studied in the experiments. In all experiments, $\mathcal{F} = \mathcal{L}^2(\mathcal{A})$ where $\mathcal{A}$ is the set of all admissible states for each problem, and the $\mathcal{L}^2$ norm is computed on $\mathcal{D}_{train}$ by: $\|F\|^2 \approx \sum_{i,k} \|F(X^{(i)}_{k\Delta t})\|^2$. All considered sets of physical functionals $\mathcal{F}_p$ are closed and convex in $\mathcal{F}$ and thus are Chebyshev. In order to enable the evaluation on 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 X_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)[4].

Reaction-diffusion equations The system is driven by a FitzHugh-Nagumo type PDE [141]

$$\frac{\partial u}{\partial t} = a\Delta u + R_u(u, v; k), \frac{\partial v}{\partial t} = b\Delta v + R_v(u, v)$$

where a and b are respectively the diffusion coefficients of u and v, Δ is the Laplace operator. The local reaction terms are $R_u(u, v; k) = u - u^3 - k - v, R_v(u, v) = u - v$. The state $X = (u, v)$ 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 $X_{\text{init}} \in [0, 1]^{2\times32\times32}$, we generate states by integrating the true PDE with fixed a, b, and k ($a = 1 \times 10^{-3}, b = 5 \times 10^{-3}, k = 5 \times 10^{-3}$). We 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 final dataset. We set the time of the random initial states 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 X_0 and predict the sequence until the horizon (equivalent to 25 time steps) in all reaction-diffusion experiments.

Neural network architectures Our F_a here is a 3-layer convolution network (ConvNet). The two input channels are (u, v) and two output ones are $(\frac{\partial u}{\partial t}, \frac{\partial v}{\partial t})$. The purely data-driven Neural ODE uses this ConvNet as its F. The detailed architecture is provided in Table 3.2. 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: $Niter = 1, \lambda_0 = 1, \tau_1 = 1 \times 10^{-3}, \tau_2 = 1 \times 10^3$.

The considered physical models are:

- *Param PDE (a, b)*, with unknown (a, b) diffusion terms and without reaction terms: $\mathcal{F}_p = \{F^{a,b}_p : (u, v) \mapsto (a\Delta u, b\Delta v) \mid a \geq a_{\min} > 0, b \geq b_{\min} > 0\}$;

- *Param PDE (a, b, k)*, the full PDE with unknown parameters: $\mathcal{F}_p = \{F^{a,b,k}_p : (u, v) \mapsto (a\Delta u + R_u(u, v; k), b\Delta v + R_v(u, v) \mid a \geq a_{\min} > 0, b \geq b_{\min} > 0, k \geq k_{\min} > 0\}$.

[4]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.

Damped wave equations The damped wave equation is defined by

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

where c is the wave speed and k is the damping coefficient. The state is $X = (w, \frac{\partial w}{\partial t})$. 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 condition are imposed for generation.

Dataset δt was set to 0.001 to respect the Courant number and provide stable integration. The simulation was integrated using a 4th order finite difference Runge-Kutta scheme for 300 steps from an initial Gaussian state, i.e for all sequence at $t = 0$, we have:

$$w(x, y, t = 0) = C \times \exp^{\frac{(x-x_0)^2 + (y-y_0)^2}{\sigma^2}} \tag{3.8}$$

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 while 50 are reserved as a test set. We have $c = 330$ and $k = 50$. As with the reaction diffusion case, the algorithm takes as input a state $X_{t_0} = (w, \frac{\mathrm{d}w}{\mathrm{d}t})(t_0)$ 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 3.2. 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 3.2.

Optimization hyperparameters We use the same hyperparameters for the experiments: $Niter = 3, \lambda_0 = 1,$

The physical models are:

- *Param PDE (c)*, without damping term: $\mathcal{F}_p = \{F_p^c : (u, v) \mapsto (v, c^2 \Delta u) \mid c \in [\epsilon, +\infty) \text{ with } \epsilon > 0\}$;
- *Param PDE (c, k)*: $\mathcal{F}_p = \{F_p^{c,k} : (u, v) \mapsto (v, c^2 \Delta u - kv) \mid c, k \in [\epsilon, +\infty) \text{ with } \epsilon > 0\}$.

Table 3.1: Hyperparameters of the damped pendulum experiments.

Method	Niter	λ_0	τ_1	τ_2
APHYNITY Hamiltonian	5	1	1	0.1
APHYNITY ParamODE (ω_0)	5	1	1	10
APHYNITY ParamODE (ω_0, λ)	5	1000	1	100

Table 3.2: ConvNet architecture in reaction-diffusion and wave equation experiments, used as data-driven derivative operator in APHYNITY and Neural ODE [51].

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

Damped pendulum We consider the non-linear damped pendulum problem, governed by the ODE

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

where $\theta(t)$ is the angle, $\omega_0 = \frac{2\pi}{T_0}$ is the proper pulsation (T_0 being the period) and α is the damping coefficient. With the state $X = (\theta, \frac{\mathrm{d}\theta}{\mathrm{d}t})$, the ODE can be written as $\frac{\mathrm{d}X_t}{\mathrm{d}t} = F(X_t)$ with $F : X \mapsto (\frac{\mathrm{d}\theta}{\mathrm{d}t}, -\omega_0^2 \sin\theta - \alpha\frac{\mathrm{d}\theta}{\mathrm{d}t})$.

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 ($T_0 = 12, \alpha = 0.2$) and varying initial conditions. The simulation integrator is Dormand-Prince Runge-Kutta method of order (4)5 (DOPRI5, [75]). 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 [106].

Neural network architectures We detail in Table 3.3 the neural architectures used for the damped pendulum experiments. All data-driven augmentations for approximating the mapping $X_t \mapsto F(X_t)$ are implemented by multi-layer perceptrons (MLP) with 3 layers of 200 neurons and ReLU activation functions (except at the last layer: linear activation). The Hamiltonian [106, 261] is implemented by a MLP that takes the state X_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 $\mathcal{H}$ with respect to the input: $F(X_t) = \left(\frac{\partial\mathcal{H}}{\partial(\mathrm{d}\theta/\mathrm{d}t)}, -\frac{\partial\mathcal{H}}{\mathrm{d}\theta}\right)$.

Optimization hyperparameters The hyperparameters of the APHYNITY optimization algorithm $(Niter, \lambda_0, \tau_1, \tau_2)$ were cross-validated on the validation set and are shown in Table 3.1. All models were trained with a maximum number of 5000 steps with early stopping.

The physical models are:

- *Hamiltonian* [106], a conservative approximation, with $\mathcal{F}_p = \{F_p^{\mathcal{H}} : (u, v) \mapsto (\partial_y\mathcal{H}(u, v), -\partial_x\mathcal{H}(u, v)) \mid \mathcal{H} \in H^1(\mathbb{R}^2)\}$, $H^1(\mathbb{R}^2)$ is the first order Sobolev space.
- *Param ODE (ω_0)*, the frictionless pendulum: $\mathcal{F}_p = \{F_p^{\omega_0,\alpha=0} \mid \omega_0 \in [\epsilon, +\infty) \text{ with } \epsilon > 0\}$
- *Param ODE (ω_0, α)*, the full pendulum equation: $\mathcal{F}_p = \{F_p^{\omega_0,\alpha} \mid \omega_0, \alpha \in [\epsilon, +\infty) \text{ with } \epsilon > 0\}$.

Table 3.3: Neural network architectures for the damped pendulum experiments. n/a corresponds to non-applicable cases.

Method	Physical model	Data-driven model
Neural ODE	n/a	MLP(in=2, units=200, layers=3, out=2)
Hamiltonian	MLP(in=2, units=200, layers=3, out=1)	n/a
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	n/a
APHYNITY Param ODE (ω_0)	1 trainable parameter ω_0	MLP(in=2, units=200, layers=3, out=2)
Param ODE (ω_0, α)	2 trainable parameters ω_0, λ	n/a
APHYNITY Param ODE (ω_0, α)	2 trainable parameters ω_0, λ	MLP(in=2, units=200, layers=3, out=2)

Baselines As purely data-driven baselines, we use Neural ODE [51] for the three problems and PredRNN++ ([272], 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, we compare for all problems, to the vanilla MB/ML cooperation scheme found in [269, 181]. We also show results for *True PDE/ODE*, which corresponds to the equation for data simulation (which do not lead to zero error due to the difference between simulation and training integration schemes). For the pendulum, we compare to Hamiltonian neural networks [106, 261] and to the the deep Galerkin method (DGM, [244]).

- *Results*

We analyze and discuss below the results obtained for the three datasets 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): -4.6 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 3.4. The Deep Galerkin method for the pendulum in complete physics *DGM* (ω_0, α), 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 wave equation in Table 3.4. Thus APHYNITY is able to augment incomplete physical models with a learned data-driven component as we wanted it to.

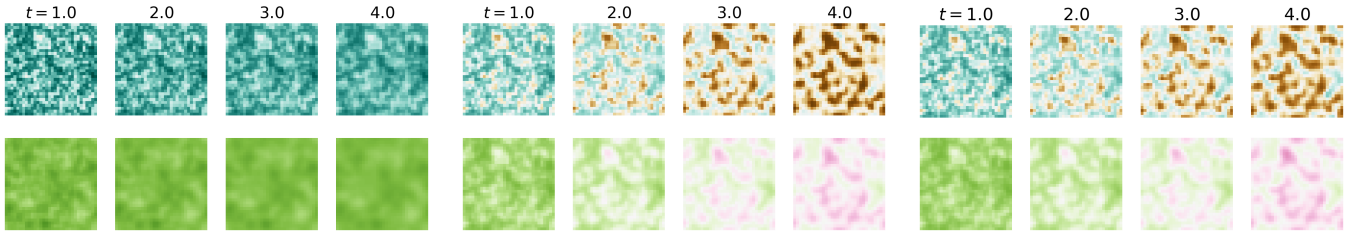
Physical parameter estimation Confirming the phenomenon mentioned in the introduction, incomplete physical models can lead to bad estimates for the relevant physical parameters: an error respectively up to 67.6% and 10.4% for parameters in the reaction-diffusion and wave equations, and an error of more than 13% for parameters for the pendulum in Table 3.4. APHYNITY is able to significantly improve physical parameters identification: down to 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 its approximations is not only beneficial for prediction, but also

Table 3.4: Forecasting and identification results on the (a) reaction-diffusion, (b) wave equation, and (c) damped pendulum datasets. We set for (a) $a = 1 \times 10^{-3}, b = 5 \times 10^{-3}, k = 5 \times 10^{-3}$, for (b) $c = 330$, $k = 50$ and for (c) $T_0 = 6$, $\alpha = 0.2$ as true parameters. log MSEs are computed respectively over 25, 25, and 40 predicted time-steps. %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. n/a 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	n/a	n/a
		PredRNN++	-4.60±0.01	n/a	n/a
	Incomplete physics	Param PDE (a, b)	-1.26±0.02	67.6	n/a
		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	n/a
		APHYNITY Param PDE (a, b, k)	**-9.35±0.02**	**0.096**	1.5e-6
		True PDE	-8.81±0.05	n/a	n/a
		APHYNITY True PDE	**-9.17±0.02**	n/a	1.4e-7
(b) Wave equation	Data-driven	Neural ODE	-2.51±0.29	n/a	n/a
	Incomplete physics	Param PDE (c)	0.51±0.07	10.4	n/a
		APHYNITY Param PDE (c)	**-4.64±0.25**	**0.31**	71.
	Complete physics	Param PDE (c, k)	-4.68±0.55	1.38	n/a
		APHYNITY Param PDE (c, k)	**-6.09±0.28**	**0.70**	4.54
		True PDE	-4.66±0.30	n/a	n/a
		APHYNITY True PDE	**-5.24±0.45**	n/a	0.14
(c) Damped pendulum	Data-driven	Neural ODE	-2.84±0.70	n/a	n/a
	Incomplete physics	Hamiltonian	-0.35±0.10	n/a	n/a
		APHYNITY Hamiltonian	**-3.97±1.20**	n/a	623
		Param ODE (ω_0)	-0.14±0.10	13.2	n/a
		Deep Galerkin Method (ω_0)	-3.10±0.40	22.1	n/a
		APHYNITY Param ODE (ω_0)	**-7.86±0.60**	**4.0**	132
	Complete physics	Param ODE (ω_0, α)	**-8.28±0.40**	**0.45**	n/a
		Deep Galerkin Method (ω_0, α)	-3.14±0.40	7.1	n/a
		APHYNITY Param ODE (ω_0, α)	**-8.31±0.30**	**0.39**	8.5
		True ODE	**-8.58±0.20**	n/a	n/a
		APHYNITY True ODE	**-8.44±0.20**	n/a	2.3

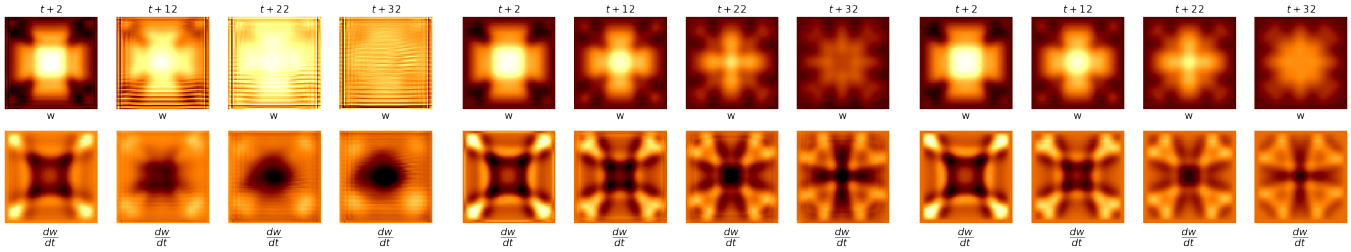
helps to limit errors for parameter identification when dynamical models do not fit data well. This is crucial for interpretability and explainability of the estimates.

Flexibility and Robustness When applied to complete physical models, APHYNITY does not degrade accuracy, contrary to a vanilla cooperation scheme (see ablations below). 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 3.4 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 the error introduced by different numerical integration methods for data simulation and training without overfitting.



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

Figure 3.2: 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 3.3: Comparison between the prediction of APHYNITY when c is estimated and Neural ODE for the damped wave equation. Note that $t + 32$, last column for (a, b, c) is already beyond the training time horizon ($t + 25$), showing the consistency of APHYNITY method.

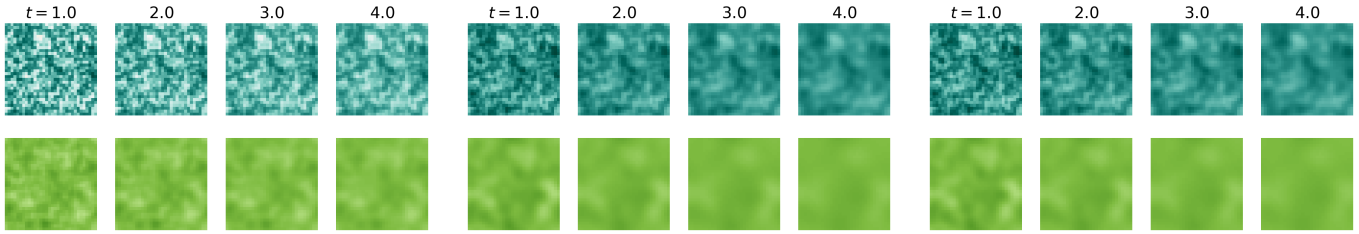
Qualitative visualizations Results in Figure 3.2 for reaction-diffusion show that the incomplete diffusion parametric PDE in Figure 3.2(a) is unable to properly match ground truth simulations: the behavior of the two components in Figure 3.2(a) is reduced to simple independent diffusions due to the lack of interaction terms between u and v. By using APHYNITY in Figure 3.2(b)the formation of patterns is similar to the ground truth. This confirms that F_a can learn the reaction terms and improve prediction quality. In Figure 3.3, 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. APHYNITY incorporates a Laplacian term as well as the data-driven F_a thus capturing the damping phenomenon and succeeding in maintaining physically sound results for long term forecasts, unlike Neural ODE.

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 (Fig. 3.4(a)) and the true one (Fig. 3.4(c)) while APHYNITY gives a good estimation of those diffusion parameters as shown in Fig. 3.4(b).

Ablation study We conduct ablation studies[5] to validate the importance of the APHYNITY augmentation as implemented in the previous experiments compared to a naive strategy consisting in learning $F = F_p + F_a$ without taking care on the quality of the decomposition, to validate trajectory-based learning and the adaptive optimization scheme. Those three ablations are presented in Tables 3.5 and 3.6:

- *Ablation to vanilla MB/ML cooperation*: In Table 3.5, we consider the ablation case with the vanilla augmentation scheme found in [147, 269, 181], which does not present any proper

[5]Reviewers for the NeurIPS 2020 and ICLR 2021 conferences are to be thanked for this section as they helped by suggesting the ablations we conducted.



(a) $a = 0.33 \times 10^{-3}, b = 0.94 \times 10^{-3}$, diffusion estimated with Param PDE (a, b)

(b) $a = 0.97 \times 10^{-3}, b = 4.75 \times 10^{-3}$, diffusion estimated with APHYNITY Param PDE (a, b)

(c) $a = 1.0 \times 10^{-3}, b = 5.0 \times 10^{-3}$, true diffusion

Figure 3.4: Diffusion predictions using coefficient learned with (a) incomplete physical model Param PDE (a, b) and (b) APHYNITY-augmented Param PDE(a, b), compared with the (c) true diffusion

decomposition guarantee. We observe that the APHYNITY cooperation scheme outperforms this vanilla scheme in all case, 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.

- *derivative supervision*: In Table 3.6 $F_p + F_a$ is trained with supervision over approximated derivatives on ground truth trajectory, as performed in [106, 65]. More precisely, APHYNITY's $\mathcal{L}_{\text{traj}}$ is here replaced with $\mathcal{L}_{\text{deriv}} = \|\frac{\mathrm{d}X_t}{\mathrm{d}t} - F(X_t)\|$ as in eq. 3.3, where $\frac{\mathrm{d}X_t}{\mathrm{d}t}$ is approximated by finite differences on X_t.

- *non-adaptive optim.*: In Table 3.6 in which we train APHYNITY by minimizing $\|F_a\|$ without the adaptive optimization of λ shown in Algorithm 2. This case is equivalent to $\lambda = 1, \tau_2 = 0$.

Those results highlight the importance to use a principled adaptive optimization algorithm: 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.

Extension to varying dynamics across the dataset We also tested APHYNITY in a setting where the parameters determining the equation defining the dataset vary from one trajectory to another. In order to build a meaningful model, we used an encoder (with varying architectures depending on the experiment) to determine, from an initial sequence of a few states, a value of those parameters characterising the sequence. Results show that APHYNITY accommodates well to this setting, with similar trends as those reported in this section.[6]

- *Reaction-diffusion systems with varying diffusion parameters* Here, 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. In order 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 3.2 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 3.7, we observe in this more challenging setting the exact same trends as

[6]Note that those last experiments have initiated the reflexion underlying the following section.

Table 3.5: Ablation study comparing APHYNITY to the vanilla augmentation scheme [269, 181] 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	(7.5±1.4)e1
	APHYNITY Param. PDE (a, b)	**-5.10±0.21**	**2.3**	(6.7±0.4)e1
	Param. PDE (a, b, k) with vanilla aug.	-8.04±0.03	25.4	(1.5±0.2)e-2
	APHYNITY Param. PDE (a, b, k)	**-9.35±0.02**	**0.096**	(1.5±0.4)e-6
	True PDE with vanilla aug.	-8.12±0.05	n/a	(6.1±2.3)e-4
	APHYNITY True PDE	**-9.17±0.02**	n/a	(1.4±0.8)e-7
Wave equation	Param PDE (c) with vanilla aug.	-3.90 ± 0.27	0.51	88.66
	APHYNITY Param PDE (c)	**-4.64±0.25**	**0.31**	71.0
	Param PDE (c, k) with vanilla aug.	-5.96 ± 0.10	0.71	25.1
	APHYNITY Param PDE (c, k)	**-6.09±0.28**	**0.70**	4.54
Damped pendulum	Hamiltonian with vanilla aug.	-0.35±0.1	n/a	837±117
	APHYNITY Hamiltonian	**-3.97±1.2**	n/a	623±68
	Param ODE (ω_0) with vanilla aug.	-7.02±1.7	4.5	148±49
	APHYNITY Param ODE (ω_0)	**-7.86±0.6**	**4.0**	132±11
	Param ODE (ω_0, α) with vanilla aug.	-7.60±0.6	4.65	35.5±6.2
	APHYNITY Param ODE (ω_0, α)	**-8.31±0.3**	**0.39**	8.5±2.0
	Augmented True ODE with vanilla aug.	**-8.40±0.2**	n/a	3.4±0.8
	APHYNITY True ODE	**-8.44±0.2**	n/a	2.3±0.4

before, as APHYNITY improves the prediction precision and parameter estimation as well as the same decreasing tendency of $\|F_a\|$.

- *Additional results for the wave equation* Here, each sequence is generated with a different wave celerity. For each simulated sequence, an initial condition is sampled as described previously, along with a wave celerity c also sampled uniformly in $[300, 400]$. k is fixed for all sequences and $k = 50$. 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 3.2 but with 10 input channels. The hyper-parameters used in these experiments are the same as before. The results are consistent with what was observed before.

Table 3.8: 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	$\|F_a\|^2$
Data-driven	Neural ODE	0.056±0.34	n/a	n/a	n/a
Incomplete physics	Param PDE (c)	-1.32±0.27	23.9	n/a	n/a
	APHYNITY Param PDE (c)	**-4.51±0.38**	3.2	n/a	171
Complete physics	Param PDE (c, k)	-4.25±0.28	3.54	1.43	n/a
	APHYNITY Param PDE (c, k)	**-4.84±0.57**	2.41	0.064	3.64
	True PDE (c, k)	**-4.51±0.29**	n/a	n/a	n/a
	APHYNITY True PDE (c, k)	**-4.49±0.22**	n/a	n/a	0.0005

Table 3.6: 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	(6.8±0.6)e1
	Augmented Param. PDE (a, b) non-adaptive optim.	-4.55±0.11	7.5	(7.6±1.0)e1
	APHYNITY Param. PDE (a, b)	**-5.10±0.21**	**2.3**	(6.7±0.4)e1
	Augmented Param. PDE (a, b, k) derivative supervision	-4.90±0.06	11.7	(1.9±0.3)e-1
	Augmented Param. PDE (a, b, k) non-adaptive optim.	-9.10±0.02	0.21	(5.5±2.9)e-7
	APHYNITY Param. PDE (a, b, k)	**-9.35±0.02**	**0.096**	(1.5±0.4)e-6
	Augmented True PDE derivative supervision	-6.03±0.01	n/a	(3.1±0.8)e-3
	Augmented True PDE non-adaptive optim.	-9.01±0.01	n/a	(1.5±0.8)e-6
	APHYNITY True PDE	**-9.17±0.02**	n/a	(1.4±0.8)e-7
Wave equation	Augmented Param PDE (c) derivative supervision	-1.16±0.48	12.1	0.00024
	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	0.00012
	Augmented Param PDE (c, k) non-adaptive optim.	-4.93±0.51	1.32	0.054
	APHYNITY Param PDE (c, k)	**-6.09±0.28**	**0.70**	4.54
	Augmented True PDE derivative supervision	-4.42 ± 0.33	n/a	6.02e-5
	Augmented True PDE non-adaptive optim.	-4.97±0.49	n/a	0.23
	APHYNITY True PDE	**-5.24±0.45**	n/a	0.14
Damped pendulum	Augmented Hamiltonian derivative supervision	-0.83±0.3	n/a	642±121
	Augmented Hamiltonian non-adaptive optim.	-0.49±0.58	n/a	165±30
	APHYNITY Hamiltonian	**-3.97±1.2**	n/a	623±68
	Augmented Param ODE (ω_0) derivative supervision	-1.02±0.04	5.8	136±13
	Augmented Param ODE (ω_0) non-adaptive optim.	-4.30±1.3	4.4	90.4±27
	APHYNITY Param ODE (ω_0)	**-7.86±0.6**	**4.0**	132±11
	Augmented Param ODE (ω_0, α) derivative supervision	-2.61±0.2	5.0	3.2±1.7
	Augmented Param ODE (ω_0, α) non-adaptive optim.	-7.69±1.3	1.65	4.8±7.7
	APHYNITY Param ODE (ω_0, α)	**-8.31±0.3**	**0.39**	8.5±2.0
	Augmented True ODE derivative supervision	-2.14±0.3	n/a	4.1±0.6
	Augmented True ODE non-adaptive optim.	**-8.34±0.4**	n/a	1.4±0.3
	APHYNITY True ODE	**-8.44±0.2**	n/a	2.3±0.4

- *Damped pendulum with varying parameters* Here we vary the parameters (T_0, α) 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 to 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 [207]. Results shown in Table 3.9 are consistent with those presented before as augmenting them with APHYNITY significantly and consistently improves both forecasting results and parameter identification precision.

Table 3.7: Results of the dataset of reaction-diffusion with varying (a, b). $k = 5 \times 10^{-3}$ is shared across the dataset.

	Method	log MSE	%Err a	%Err b	%Err k	$\|F_a\|^2$
Data-driven	Neural ODE [51]	-3.61±0.07	n/a	n/a	n/a	n/a
Incomplete physics	Param PDE (a, b)	-1.32±0.02	55.6	54.1	n/a	n/a
	APHYNITY Param PDE (a, b)	**-4.32±0.32**	**11.8**	**18.7**	n/a	(4.3±0.6)e1
Complete physics	Param PDE (a, b, k)	**-5.54±0.38**	1.55	3.10	0.59	n/a
	APHYNITY Param PDE (a, b, k)	**-5.72±0.25**	**1.29**	**1.23**	**0.39**	(5.9±4.3)e-1
	True PDE	**-8.86±0.02**	n/a	n/a	n/a	n/a
	APHYNITY True PDE	**-8.82±0.15**	n/a	n/a	n/a	(1.8±0.6)e-5

Table 3.9: 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. n/a corresponds to non-applicable cases.

	Method	log MSE	%Error T_0	%Error α	$\|F_a\|^2$
data-driven	Neural ODE [51]	-4.35±0.9	n/a	n/a	n/a
	N-Beats [207]	-4.57±0.5	n/a	n/a	n/a
Incomplete physics	Hamiltonian [106]	-1.31±0.4	n/a	n/a	n/a
	APHYNITY Hamiltonian	**-4.72±0.4**	n/a	n/a	5.6±0.6
	Param ODE (ω_0)	-2.66±0.9	21.5±19	n/a	n/a
	APHYNITY Param ODE (ω_0)	**-5.94±0.7**	**5.0±1.8**	n/a	0.49±0.1
Complete physics	Param ODE (ω_0, α)	**-5.71±0.4**	4.08±0.8	152±129	n/a
	APHYNITY Param ODE (ω_0, α)	**-6.22±0.7**	**3.26±0.6**	**62±27**	(5.39±0.1)e-10
	True ODE	**-8.58±0.1**	n/a	n/a	n/a
	APHYNITY True ODE	**-8.58±0.1**	n/a	n/a	(2.15±1.6)e-4

3.4 *LEADS*: Learning Dynamical Systems that Generalize Across Environments

Despite promising results, current ML models aiming to learn real world dynamics from data 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 [177]; or, in health science, parameters need to be personalized for each patient as for cardiac computational models [199]. More generally, data acquisition and modeling are affected by different factors such as geographical position, sensor variability, measuring circumstances, etc. In other words, within the datasets used to learn models of dynamical systems of real-world phenomena, the distribution of data often changes according to the environment in which they are captured, the dynamics of the system itself varying from one environment to another. Generalizing across those different environments thus challenges the conventional frameworks.

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 other words, both are suboptimal.

In this work, we consider a setting where it is explicitly assumed that the trajectories are collected from different environments. The i.i.d. hypothesis is thus removed twice: by considering the temporality of the data as is done in all other models of this thesis and, specifically to this particular work, by considering the existence of multiple environments. Moreover, because in many useful contexts the dynamics in each environment, while being distinct, share similarities with one another, our objective is also to leverage those similarities in order to improve the modeling capacity and generalization performance. 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;
- ***One-Per-Env.***, where a specific function is fitted for each environment?

RQ2 How does it help in extrapolating to novel environments which has not been seen during training?

The general learning framework we propose here is *LEarning Across Dynamical Systems* (*LEADS*), which decomposes the learned differential equations 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 of 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.

Our contributions are the following:

- We show, under mild conditions, that the learning problem is well-posed, as the resulting decomposition exists and is unique (Section 3.4.2) and analyze the properties of this decomposition from a sample complexity perspective.
- Those statistical bounds are shown to closely coincide with practical experiments in the linear case.
- The framework is instanciated for more general hypothesis spaces and dynamics, following a heuristic for the control of generalization which is then 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 while achieving better performance than *One-For-All* (*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*). Our experiments are done on three representative cases (Section 3.4.4) 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. Up to our knowledge, it is the first time that generalization in multiple dynamical systems is addressed from an ML perspective[7].

[7]Code is available at `https://github.com/yuan-yin/LEADS`.

3.4.1 Related work

Recent approaches linking invariances to Out-of-Distribution (OoD) Generalization, such as [11, 142, 258], aim at finding a single classifier that predicts well invariantly across environments with the power of extrapolating outside the known distributions. However, in our dynamical systems context, the optimal regression function should be different in each environment, and modeling environment bias is as important as modeling the invariant information, as both are indispensable for prediction. Thus such invariant learners are incompatible with our setting. Meta-learning methods have recently been considered for dynamical systems as in [91, 152]. Their objective is to train a single model that can be quickly adapted to a novel environment with a few data-points in limited training steps. However, in general these methods do not focus on leveraging the commonalities and discrepencies in data and may suffer from overfitting at test time [185]. Multi-task learning [293] seeks for learning shared representations of inputs that exploit the domain information. Up to our knowledge current multi-task methods have not been considered for dynamical systems. [248] apply multi-task learning for interactive physical environments but do not consider the case of dynamical systems. Other approaches like [284, 202] integrate probabilistic methods into a Neural ODE, to learn a distribution of the underlying physical processes. Their focus is on the uncertainty of a single system. In APHYNITY, we have also considered an additive decomposition but have instead focused on the combination of physical and statistical components for a single process and not on learning from different environments.

3.4.2 Approach

• *Problem setting*

We consider the problem of learning models of dynamical physical processes with data acquired from a set of environments E. Throughout the paper, we will assume that the dynamics in an environment $e \in E$ 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 environments 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. X_t^e denotes the state of the equation for environment e, taking its values from a bounded set $\mathcal{A}$, with evolution term $F_e : \mathcal{A} \to T\mathcal{A}$, $T\mathcal{A}$ being the tangent bundle of $\mathcal{A}$. In other words, over a fixed time interval $[0, T]$, we have:

$$\frac{\mathrm{d}X_t^e}{\mathrm{d}t} = F_e(X_t^e) \tag{3.9}$$

We assume that, for any e, F_e lies in a functional vector space $\mathcal{F}$. In the experiments, we will consider one ODE, in which case $\mathcal{A} \subset \mathbb{R}^d$, and two PDEs, in which case $\mathcal{A}$ is a d'-dimensional vector field over a bounded spatial domain $S \subset \mathbb{R}^{d'}$.

The differential term of the data-generating dynamical system in Eq. 3.9 is sampled from a distinct distribution for each e, i.e. $F_e \sim Q_e$. From F_e, we define $\mathcal{T}_e$, the data distribution of trajectories $X_\cdot^e$ verifying Eq. 3.9, induced by a distribution of initial states $X_0^e \sim P_0$. The data for this environment is then composed of l trajectories sampled from $\mathcal{T}_e$, and is denoted as $\hat{\mathcal{T}}_e$ with $X_\cdot^{e,i}$ the i-th trajectory. We will denote the full dataset by $\hat{\mathcal{T}} = \bigcup_{e \in E} \hat{\mathcal{T}}_e$.

The classical empirical risk minimization (ERM) framework suggests to model the data dynamics either at the global level (*One-For-All*), taking trajectories indiscriminately from $\hat{\mathcal{T}}$, or at the specific environment level (*One-Per-Env.*), training one model for each $\hat{\mathcal{T}}_e$. 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.

• *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

$$\forall e \in E, F_e = F + G_e \tag{3.10}$$

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[8].

However, as in the previous APHYNITY section, learning this decomposition from data defines an ill-posed problem: for any choice of F, there is a $\{G_e\}_{e\in E}$ such that Eq. 3.10 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 $\Omega(G_e)$, 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:

$$\min_{F,\{G_e\}_{e\in E}\in\mathcal{F}} \quad \sum_{e\in E} \Omega(G_e) \quad \text{subject to} \quad \forall X^{e,i} \in \hat{\mathcal{T}}, \forall t, \frac{\mathrm{d}X_t^{e,i}}{\mathrm{d}t} = (F + G_e)(X_t^{e,i}) \tag{3.11}$$

Minimizing Ω 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. 3.11 is well-defined. The following proposition provides an answer:

Proposition 3.3 (*Existence and Uniqueness*)

Assume Ω is convex, then the existence of a minimal decomposition $F^\star, \{G_e^\star\}_{e\in E} \in \mathcal{F}$ of Eq. 3.11 is guaranteed. Furthermore, if Ω is strictly convex, this decomposition is unique.

Proof: The optimization problem is:

$$\min_{F,G_e\in\mathcal{F}} \quad \sum_{e\in E} \Omega(G_e) \quad \text{subject to} \quad \forall X^{e,i} \in \hat{\mathcal{T}}, \forall t, \frac{\mathrm{d}X_t^{e,i}}{\mathrm{d}t} = (F + G_e)(X_t^{e,i}) \tag{3}$$

The idea is to first reconstruct the full functional from the trajectories of $\hat{\mathcal{T}}$. By definition, $\mathcal{A}^e$ is the set of points reached by trajectories in $\hat{\mathcal{T}}$ from environment e so that:

$$\mathcal{A}^e = \{x \in \mathbb{R}^d \mid \exists X_\cdot^e \in \hat{\mathcal{T}}, \exists t,\ X_t^e = x\}$$

Then we define a function F_e^{data} in the following way: for all $e \in E$, take $a \in \mathcal{A}^e$, for which we can find $X_\cdot^e \in \hat{\mathcal{T}}$ and t_0 such that $X_{t_0}^e = a$. Differentiating $X_\cdot^e$ at t_0, which is possible by definition of $\hat{\mathcal{T}}$, we take:

$$F_e^{\text{data}}(a) = \left.\frac{\mathrm{d}X_t^e}{\mathrm{d}t}\right|_{t=t_0}$$

For any (F, G_e) satisfying the constraint in Eq. 3, we then have $(F + G_e)(a) = \left.\frac{\mathrm{d}x_t}{\mathrm{d}t}\right|_{t_0} = F_e^{\text{data}}(a)$ for all $a \in \mathcal{A}^e$. 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. 3 and the following objective:

$$\min_{F\in\mathcal{F}} \quad \sum_e \Omega(F_e^{\text{data}} - F)$$

[8]This is emphatically shown in the case of the Gray-Scott equations.

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. □

In practice, while we could use, as for APHYNITY, an adaptive optimization scheme converging a constrained solution, we consider in the following this relaxed formulation of Eq. 3.11:

$$\min_{F,\{G_e\}_{e\in E}\in\mathcal{F}} \quad \sum_{e\in E}\left(\frac{1}{\lambda}\Omega(G_e)+\sum_{i=1}^{l}\int_0^T\left\|\frac{\mathrm{d}X_t^{e,i}}{\mathrm{d}t}-(F+G_e)(X_\tau^{e,i})\right\|^2\mathrm{d}t\right) \tag{3.12}$$

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. In practice, this relaxed formulation gave satisfying results.

Note that, as argued in Section 3.2.1, the loss term supervising on derivatives is replaced in the experiments with a trajectory-based loss. However, the derivatives formulation was more convenient for the sample complexity analysis.

3.4.3 Improving generalization with *LEADS*

Defining an appropriate Ω 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 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 Ω for NNs.

More precisely:

- Proposition 3.4 shows that the number of examples necessary to ensure a certain level of test error decreases with an increasing number of environments for any generic learners while Proposition 3.6 tackles generalization w.r.t. new environments, unseen during learning.
- Propositions 3.7 and 3.8 apply those bounds in the linear case to find and justify an appropriate choice for Ω. The bounds are then shown to be consistent with the experiments.
- Propositions 3.9 and 3.10 extend those results for Neural Networks.

• *Preliminary Definitions to Study Generalization with LEADS*

We introduce here the necessary definitions for the different proofs of generalization proofs presented later.

Table 3.10 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 [240]. 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 3.10. For simplicity, we therefore omit the notation of loss function for the hypothesis spaces.

As in [25], covering numbers are based on pseudo-metrics. We can verify that all distances in Table 3.10 are pseudo-metrics.

• *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

Table 3.10: Capacity definitions of different sets by covering number with associated metric or pseudo-metric.

Capacity	Metric or pseudo-metric	Mentioned in
$\mathcal{C}(\varepsilon, \mathbb{H}^m) := \sup_{\boldsymbol{\mathcal{P}}} \mathcal{N}(\varepsilon, \mathbb{H}^m, d_{\boldsymbol{\mathcal{P}}})$	$d_{\boldsymbol{\mathcal{P}}}((F+G_1,\dots,F+G_m),(F'+G'_1,\dots,F'+G'_m)) = \int_{(\mathcal{A}\times T\mathcal{A})^m} \frac{1}{m}\lvert\sum_{e\in E}\lVert(F+G_e)(x^e)-y^e\rVert^2 - \sum_{e\in E}\lVert(F'+G'_e)(x^e)-y^e\rVert^2\rvert \mathrm{d}\boldsymbol{\mathcal{P}}(\mathbf{x},\mathbf{y})$	Theorem 3.1; Prop. 3.5
$\mathcal{C}_{\hat{\mathcal{G}}}(\varepsilon, \hat{\mathcal{F}}) := \sup_{\mathcal{P}} \mathcal{N}(\varepsilon, \hat{\mathcal{F}}, d_{[\mathcal{P},\hat{\mathcal{G}}]})$	$d_{[\mathcal{P},\hat{\mathcal{G}}]}(f,F') = \int_{\mathcal{A}\times T\mathcal{A}} \sup_{g\in\hat{\mathcal{G}}} \lvert\lVert(F+g)(x)-y\rVert^2 - \lVert(F'+g)(x)-y\rVert^2\rvert \mathrm{d}\mathcal{P}(x,y)$	Prop. 3.4, 3.5, 3.10; Cor. 3.1
$\mathcal{C}_{\hat{\mathcal{F}}}(\varepsilon, \hat{\mathcal{G}}) := \sup_{\mathcal{P}} \mathcal{N}(\varepsilon, \hat{\mathcal{G}}, d_{[\mathcal{P},\hat{\mathcal{F}}]})$	$d_{[\mathcal{P},\hat{\mathcal{F}}]}(g,g') = \int_{\mathcal{A}\times T\mathcal{A}} \sup_{f\in\hat{\mathcal{F}}} \lvert\lVert(F+g)(x)-y\rVert^2 - \lVert(F+g')(x)-y\rVert^2\rvert \mathrm{d}\mathcal{P}(x,y)$	Prop. 3.4, 3.5, 3.9
$\mathcal{C}(\varepsilon, F+\hat{\mathcal{G}}) := \sup_{\mathcal{P}} \mathcal{N}(\varepsilon, F+\hat{\mathcal{G}}, d_{\mathcal{P}})$	$d_{\mathcal{P}}(F+g,F+g') = \int_{\mathcal{A}\times T\mathcal{A}} \lvert\lVert(F+g)(x)-y\rVert^2 - \lVert(F+g')(x)-y\rVert^2\rvert \mathrm{d}\mathcal{P}(x,y)$	Prop. 3.6
$\mathcal{C}(\varepsilon, \hat{\mathcal{G}}, L^1) := \sup_{\mathcal{P}} \mathcal{N}(\varepsilon, \hat{\mathcal{G}}, d_{L^1(\mathcal{P})})$	$d_{L^1(\mathcal{P})}(g,g') = \int_{\mathbb{R}^d} \lVert(g-g')(x)\rVert_1 \mathrm{d}\mathcal{P}(x)$	Prop. 3.9; Theorem 3.3
$\mathcal{C}(\varepsilon, \hat{\mathcal{G}}, L^2) := \sup_{\mathcal{P}} \mathcal{N}(\varepsilon, \hat{\mathcal{G}}, d_{L^2(\mathcal{P})})$	$d_{L^2(\mathcal{P})}(g,g') = \sqrt{\int_{\mathbb{R}^d} \lVert(g-g')(x)\rVert_2^2 \mathrm{d}\mathcal{P}(x)}$	Prop. 3.7; Lemma 3.1

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 on a new and unseen environment. The bounds here are inspired by ideas initially introduced in [25]. 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 x we want to predict some target y. In our setting, we want to learn trajectories $(X_t^e)_{0\leq t\leq T}$ starting from an initial condition X_0. We reformulate this problem and cast it as a standard supervised learning problem: $\mathcal{T}_e$ being the data distribution of trajectories for environment e, as defined in 3.4.2, let us consider a trajectory $X_\cdot^e \sim \mathcal{T}_e$, and time $\tau \sim \mathrm{Unif}([0,T])$; we define system states $X = X_\tau^e \in \mathcal{A}$ as input, and the corresponding values of derivatives $Y = F_e(X_\tau^e) \in T\mathcal{A}$ as the associated target. We will denote $\mathcal{P}_e$ the underlying distribution of (X,Y), and $\hat{\mathcal{P}}_e$ the associated dataset of size n.

We are searching for $F, G_e : \mathcal{A} \to T\mathcal{A}$ 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 $h : \mathcal{A} \to T\mathcal{A}$, the error on some test distribution $\mathcal{P}_e$ is given by $\mathrm{er}_{\mathcal{P}_e}(h) = \int_{\mathcal{A}\times T\mathcal{A}} \lVert h(x)-y\rVert^2 \mathrm{d}\mathcal{P}_e(x,y)$ and the error on the training set by $\hat{\mathrm{er}}_{\hat{\mathcal{P}}_e}(h) = \frac{1}{n}\sum_{(x,y)\in\hat{\mathcal{P}}_e} \lVert h(x)-y\rVert^2$.

***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 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 3.4

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

environment satisfies

$$n \geq \max\left\{\frac{64}{\varepsilon^2}\Big(\frac{1}{m}\Big(\log\frac{4}{\delta} + \log \mathcal{C}_{\hat{\mathcal{G}}}\Big(\frac{\varepsilon_1}{16}, \hat{\mathcal{F}}\Big)\Big) + \log \mathcal{C}_{\hat{\mathcal{F}}}\Big(\frac{\varepsilon_2}{16}, \hat{\mathcal{G}}\Big)\Big), \frac{16}{\varepsilon^2}\right\} \tag{3.13}$$

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 $\frac{1}{m}\sum_{e\in E} \mathrm{er}_{\mathcal{P}_e}(F+G_e) \leq \frac{1}{m}\sum_{e\in E} \hat{\mathrm{er}}_{\hat{\mathcal{P}}_e}(F+G_e) + \varepsilon$.

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 3.1 and Prop. 3.5.

Theorem 3.1 (*[25], 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

$$\frac{1}{m}\sum_{e\in E} \mathrm{er}_{\mathcal{P}_e}(F+G_e) \leq \frac{1}{m}\sum_{e\in E} \hat{\mathrm{er}}_{\hat{\mathcal{P}}_e}(F+G_e) + \varepsilon$$

Note that permissibility (as defined in [25]) is a weak measure-theoretic condition satisfied by many real world hypothesis space families [25]. 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 3.5

For all $\varepsilon, \varepsilon_1, \varepsilon_2 > 0$ such that $\varepsilon = \varepsilon_1 + \varepsilon_2$,

$$\log \mathcal{C}(\varepsilon, \mathbb{H}^m) \leq \log \mathcal{C}_{\hat{\mathcal{G}}}(\varepsilon_1, \hat{\mathcal{F}}) + m \log \mathcal{C}_{\hat{\mathcal{F}}}(\varepsilon_2, \hat{\mathcal{G}}) \tag{3.14}$$

Proof of Proposition 3.5: 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 $(\mathcal{A} \times T\mathcal{A})^m$. Let us then fix such a distribution $\boldsymbol{\mathcal{P}}$. and let $\bar{\mathcal{P}} = \frac{1}{m}\sum_{e\in E} \mathcal{P}_e$ be the average distribution.

Suppose that F^C is an ε_1-cover of $(\hat{\mathcal{F}}, d_{[\bar{\mathcal{P}},\hat{\mathcal{G}}]})$ and $\{G_e^C\}_{e\in E}$ are ε_2-covers of $(\hat{\mathcal{G}}, d_{[\mathcal{P}_e,\hat{\mathcal{F}}]})$. Let $H = \{(x_1, \ldots, x_m) \mapsto ((F+G_1)(x_1), \ldots, (F+G_m)(x_m)),\ F \in F^C,\ G_e \in G_e^C\}$, be a set built from the covering sets aforementioned. Note that by definition $|H| = |F^C| \cdot \prod_{e\in E} |G_e^C| \leq \mathcal{C}_{\hat{\mathcal{G}}}(\varepsilon_1, \hat{\mathcal{F}})\mathcal{C}_{\hat{\mathcal{F}}}(\varepsilon_2, \hat{\mathcal{G}})^m$ 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 $F' \in F^C$ such that $d_{[\bar{\mathcal{P}},\hat{\mathcal{G}}]}(F, F') \leq \varepsilon_1$ and $G'_e \in G_e^C$ 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:

$$\begin{aligned}
&d_{\boldsymbol{\mathcal{P}}}((F+G_1, \ldots, F+G_m), (F'+G'_1, \ldots, F'+G'_m)) \\
&\leq d_{\boldsymbol{\mathcal{P}}}((F+G_1, \ldots, F+G_m), (F'+G_1, \ldots, F'+G_m)) \\
&\quad + d_{\boldsymbol{\mathcal{P}}}((F'+G_1, \ldots, F'+G_m), (F'+G'_1, \ldots, F'+G'_m)) \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(triangular inequality of pseudo-metric)}
\end{aligned}$$

$$\leq \frac{1}{m}\left[\sum_{e\in E} d_{\mathcal{P}_e}(F+G_e, F'+G_e) + \sum_{e\in E} d_{\mathcal{P}_e}(F'+G_e, F'+G'_e)\right]$$

(triangular inequality of absolute value)

$$\leq \frac{1}{m}\sum_{e\in E} d_{[\mathcal{P}_e,\hat{\mathcal{G}}]}(F,F') + \frac{1}{m}\sum_{e\in E} d_{[\mathcal{P}_e,\hat{\mathcal{F}}]}(G_e,G'_e) \qquad \text{(by definition of } d_{[\mathcal{P}_e,\hat{\mathcal{G}}]} \text{ and } d_{[\mathcal{P}_e,\hat{\mathcal{F}}]})$$

$$= d_{[\bar{\mathcal{P}},\hat{\mathcal{G}}]}(F,F') + \frac{1}{m}\sum_{e\in E} d_{[\mathcal{P}_e,\hat{\mathcal{F}}]}(G_e,G'_e) \leq \varepsilon_1+\varepsilon_2$$

(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^C is an ε_1-cover of $\hat{\mathcal{F}}$ and $\{G_e^C\}$ 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. 3.14. □

We can now use the bound developed in Prop. 3.5 and use it together with Theorem 3.1, therefore concluding the proof of Prop. 3.4. □

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. 3.11. Further in this section, we will show for linear ODEs that the optimization of $\Omega(G_e)$ in Eq. 1.9 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 $\hat{\mathcal{T}}$, 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. Fig. 3.12 in Section 3.4.4 confirms this result with empirical evidence.

***LEADS* sample complexity for novel environments.** Suppose that problem Eq. 3.11 has been solved for a set of environments 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 3.6

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

$$n' \geq \max\left\{\frac{64}{\varepsilon^2}\log\frac{4\mathcal{C}(\frac{\varepsilon}{16}, f+\hat{\mathcal{G}})}{\delta}, \frac{16}{\varepsilon^2}\right\}, \tag{3.15}$$

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 3.2 (*[25], 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 the Theorem 3.2 to obtain the proposition. □

In Prop. 3.6 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 Prop. 3.4 with $m = 1$.

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

$$\log \mathcal{C}_{\hat{\mathcal{F}}}(\varepsilon, \hat{\mathcal{G}}) \leq \omega(r, \varepsilon), \quad r = \sup_{g\in\hat{\mathcal{G}}} \Omega(g) \tag{3.16}$$

Minimizing Ω 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 (ω, Ω). In the following, we will first show in Section 3.4.3 how to construct a penalization term Ω based on the covering number bound for linear approximators in the case of linear ODEs. We show with this tractable 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 3.4.3 an effective Ω which can be applied for neurally parametrized G_es.

- *Linear case: theoretical bounds correctly predict the trend of test error*

Results in 3.4.3 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 theoretical bound.

Let us consider a linear ODE $\frac{\mathrm{d}x_t^e}{\mathrm{d}t} = \mathrm{L}_{\boldsymbol{F}_e}(x_t^e)$ where $\mathrm{L}_{\boldsymbol{F}_e} : x \mapsto \boldsymbol{F}_e x$ 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* $\frac{\mathrm{d}x_t^e}{\mathrm{d}t} = (\mathrm{L}_{\boldsymbol{F}} + \mathrm{L}_{\boldsymbol{G}_e})(x_t^e)$, $\mathrm{L}_{\boldsymbol{F}} \in \hat{\mathcal{F}}, \mathrm{L}_{\boldsymbol{G}_e} \in \hat{\mathcal{G}} \subseteq \hat{\mathcal{F}}$. As suggested in [23], we have that:

Proposition 3.7

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 i.e. $\|x\|_2 \leq b$, and the MSE loss function is bounded by c, then

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

Proof: Let us take G^C an $\frac{\varepsilon}{2\sqrt{c}}$-cover of $\hat{\mathcal{G}}$ for the L^2 distance $d_{L^2(\mathcal{P})}$ (see definition in Table 3.10). Then,

for each $\mathrm{L}_{\boldsymbol{G}} \in \hat{\mathcal{G}}$, take $\boldsymbol{G}' \in G^C$ such that $d_{L^2}(\mathrm{L}_{\boldsymbol{G}}, \mathrm{L}_{\boldsymbol{G}'}) \leq \frac{\varepsilon}{2\sqrt{c}}$, then

$$
\begin{aligned}
& d_{[\mathcal{P},\hat{\mathcal{F}}]}(\mathrm{L}_{\boldsymbol{G}}, \mathrm{L}_{\boldsymbol{G}'}) \\
=& \int_{\mathcal{A}\times\mathcal{A}'} \sup_{\mathrm{L}_{\boldsymbol{F}}\in\hat{\mathcal{F}}} |\|(\boldsymbol{F}+\boldsymbol{G})x-y\|_2^2 - \|(\boldsymbol{F}+\boldsymbol{G}')x-y\|_2^2| \mathrm{d}\mathcal{P}(x,y) \\
\leq& \int_{\mathcal{A}\times T\mathcal{A}} \sup_{\mathrm{L}_{\boldsymbol{F}}\in\hat{\mathcal{F}}} \|(\boldsymbol{G}-\boldsymbol{G}')x\|_2 (\|(\boldsymbol{F}+\boldsymbol{G})x-y\|_2 + \|(\boldsymbol{F}+\boldsymbol{G}')(x)-y\|_2) \mathrm{d}\mathcal{P}(x,y) \\
\leq& \sqrt{\int_{\mathcal{A}} \|(\boldsymbol{G}-\boldsymbol{G}')x\|_2 \mathrm{d}\mathcal{P}(x)} \sqrt{\int_{\mathcal{A}\times T\mathcal{A}} \sup_{\mathrm{L}_{\boldsymbol{F}}\in\hat{\mathcal{F}}} (\|(\boldsymbol{F}+\boldsymbol{G})x-y\|_2 + \|(\boldsymbol{F}+\boldsymbol{G}')x-y\|_2)^2 \mathrm{d}\mathcal{P}(x,y)} \\
\leq& 2\sqrt{c}\sqrt{\int_{\mathbb{R}^d} \|(\boldsymbol{G}-\boldsymbol{G}')x\|_2 \mathrm{d}\mathcal{P}(x)} \leq \varepsilon
\end{aligned}
$$

so that $\mathcal{C}_{\mathcal{F}}(\varepsilon, \hat{\mathcal{G}}) \leq \mathcal{C}(\frac{\varepsilon}{2\sqrt{c}}, \hat{\mathcal{G}}, L^2)$. According to the following lemma:

Lemma 3.1 (***[23], Lemma 3.2, Adapted***)

Given positive reals (a, b, ε) and positive integer d. Let vector $x \in \mathbb{R}^d$ be given with $\|x\|_p \leq b$, $\hat{\mathcal{G}} = \{\mathrm{L}_{\boldsymbol{G}} : \boldsymbol{G} \in \mathbb{R}^{d\times d}, \|\boldsymbol{G}\|_F^2 \leq r\}$ where $\|\cdot\|_F$ is the Frobenius norm. Then

$$\log \mathcal{C}(\varepsilon, \hat{\mathcal{G}}, L^2) \leq \left\lceil \frac{rdb^2}{\varepsilon^2} \right\rceil \log 2d^2$$

we obtain

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

where $\omega(r, \varepsilon)$ is a strictly increasing function w.r.t. r. □

This result indicates that we can choose $\Omega(\mathrm{L}_{\boldsymbol{G}}) = \|\boldsymbol{G}\|_F$ as our optimization objective in Eq. 3.11. The sample complexity in Eq. 3.13 will decrease with the size of the largest possible $r = \sup_{\mathrm{L}_{\boldsymbol{G}}\in\hat{\mathcal{G}}} \Omega(\mathrm{L}_{\boldsymbol{G}})$. The optimization process will reduce $\Omega(\mathrm{L}_{\boldsymbol{G}})$ until a minimum is reached. The maximum size of the effective hypothesis space is then bounded and decreases throughout training thanks to the penalty. Therefore, in the linear case, we have:

Proposition 3.8

For linear maps $\mathrm{L}_{\boldsymbol{F}} \in \hat{\mathcal{F}}$, s.t. $\|\boldsymbol{F}\|_F^2 \leq r'$, and $\mathrm{L}_{\boldsymbol{G}} \in \hat{\mathcal{G}}$ s.t. $\|\boldsymbol{G}\|_F^2 \leq r$, for inputs s.t. $\|x\|_2 \leq b$, if the MSE loss function is bounded by c, given m environments and n samples per environment, with probability $1-\delta$, the generalization error upper bound is $\varepsilon = \max\{\sqrt{\frac{(p+\sqrt{p^2+4q})}{2}}, \sqrt{\frac{16}{n}}\}$ where $p = \frac{64}{mn}\log\frac{4}{\delta}$ and $q = \frac{64}{n}\lceil(\frac{r'}{mz^2} + \frac{r}{(1-z)^2})cd(32b)^2\rceil \log 2d^2$ for any $0 < z < 1$.

Proof: This can be derived from Proposition 3.4 with the help of Proposition 3.7 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:

$$n = \max\left\{\frac{64}{\varepsilon^2}\left(\frac{1}{m}\log\frac{4}{\delta} + \frac{1}{m}\left\lceil\frac{r'cd(32b)^2}{\varepsilon_1^2}\right\rceil \log 2d^2 + \left\lceil\frac{rcd(32b)^2}{\varepsilon_2^2}\right\rceil \log 2d^2\right), \frac{16}{\varepsilon^2}\right\}$$

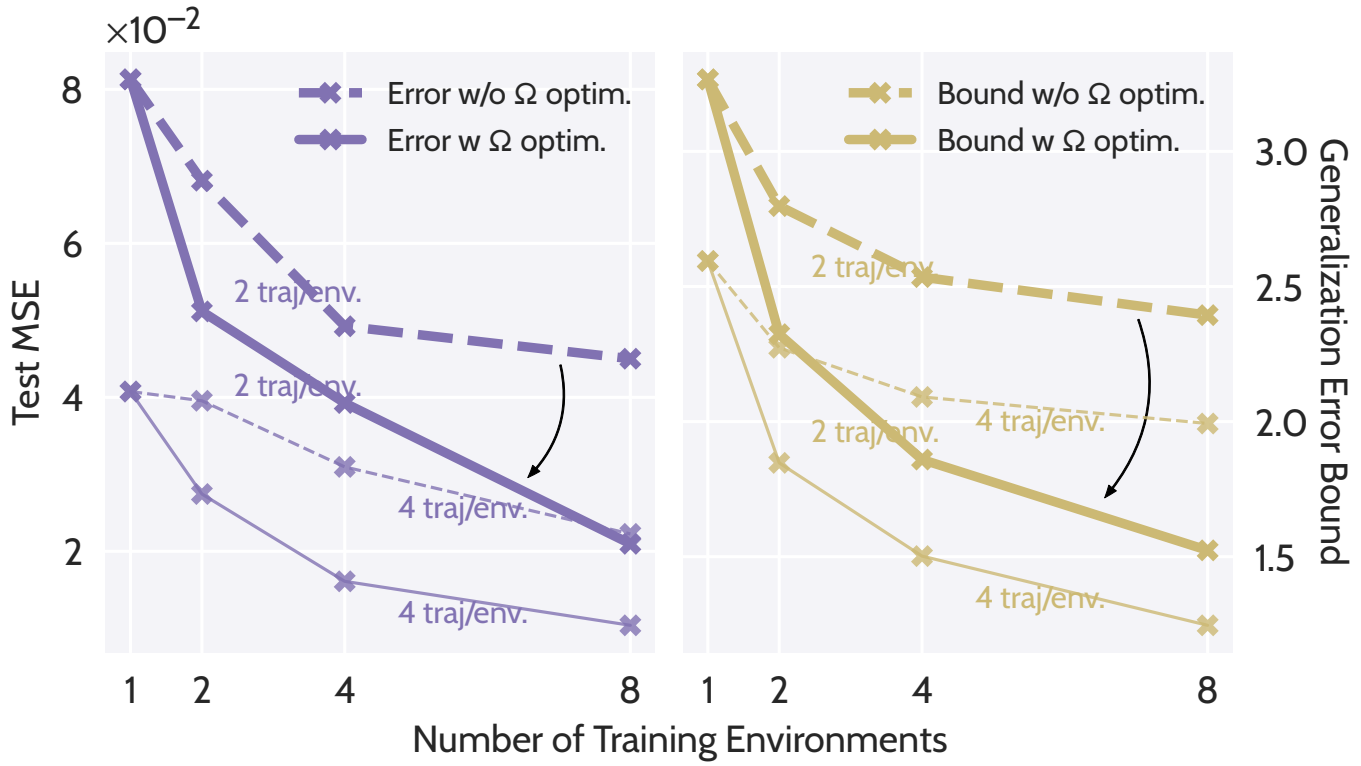


Figure 3.5: Test error compared with corresponding theoretical bound. The arrows indicate the changes after applying $\Omega(G_e)$ penalty.

Table 3.11: Details for the results producing Fig. 3.5 .

Samples/env.	Method	$m = 1$	$m = 2$	$m = 4$	$m = 8$
$n = 2 \cdot K$	*LEADS no min.*	8.13±5.56 e-2	6.81±4.44 e-2	4.92±4.26 e-2	4.50±3.10 e-2
	LEADS (Ours)		**5.11±3.20 e-2**	**3.93±2.88 e-2**	**2.10±0.96 e-2**
$n = 4 \cdot K$	*LEADS no min.*	4.08±2.57 e-2	3.96±2.56 e-2	3.10±2.08 e-2	2.23±1.44 e-2
	LEADS (Ours)		**2.74±1.96 e-2**	**1.61±1.24 e-2**	**1.02±0.74 e-2**

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 $q = \frac{64}{n}\left\lceil\left(\frac{r}{mz^2} + \frac{r'}{(1-z)^2}\right) cd(32b)^2\right\rceil \log 2d^2$. □

Experiment with Linear ODEs We take an example of linear ODE expressed by the following formula:

$$\frac{\mathrm{d}u_t}{\mathrm{d}t} = \mathrm{L}_{\boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^\top}(u_t) = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^\top u_t$$

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 $\Theta_{\mathbf{\Lambda}} = \{\mathbf{\Lambda}_1, \ldots, \mathbf{\Lambda}_8\}$, defined for each $\mathbf{\Lambda}_i$ by:

$$[\mathbf{\Lambda}_i]_{jj} = \begin{cases} 0, & \text{if } i = j \\ -0.5, & \text{otherwise.} \end{cases}$$

which means that the i-th eigenvalue is set to 0, while others are set to a common value -0.5.

In other words, we take an instance of linear ODE defined by $\boldsymbol{F}_e = \boldsymbol{Q}\mathbf{\Lambda}_e\boldsymbol{Q}^\top$ with the diagonal $\mathbf{\Lambda}_e$ specific to each environment. After solving Eq. 3.11 we have at the optimum that $\boldsymbol{G}_e = \boldsymbol{F}_e - \boldsymbol{F}^\star = \boldsymbol{F}_e - \frac{1}{m}\sum_{e' \in E} \boldsymbol{F}_{e'}$. Then we can take $r = \max_{\{\mathrm{L}_{\boldsymbol{G}_e}\}} \Omega(\mathrm{L}_{\boldsymbol{G}_e})$ as the norm bound of $\hat{\mathcal{G}}$ when $\Omega(G_e)$ is optimized.

Fig. 3.5 shows on the left the test error with and without penalty and the corresponding theoretical bound on the right. Corresponding quantitative measures are given in Table 3.11. As expected, we observe that minimizing Ω reduces the test error and closely follows the theoretical generalization bound, as indicated by the arrows from the dashed line to the concrete one.

- *Nonlinear case: instantiation for neural nets*

The above linear case validates the ideas introduced in Prop. 3.4 and provides an instantiation guide and an intuition on the more complex nonlinear case. This motivates us to instantiate the general case by choosing an appropriate approximating space $\hat{\mathcal{F}}$ and a penalization function Ω from the generalization bounds for the corresponding space.

We show in this section how we design a concrete model for nonlinear dynamics following the general guidelines given in 3.4.3. This is mainly composed of the following two parts: (a) choosing an appropriate approximation space and (b) choosing a penalization function Ω for this space. It is important to note that, even if the bounds given in the following sections may be loose in general, we are convinced that they provide useful intuitions on the design of the algorithms which are 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 [139], and the existence of efficient optimization algorithms [55], 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 $\mathbb{R}^d$: $\hat{\mathcal{F}}_{\mathrm{NN}} = \{\nu : x \mapsto \sigma_D(W_D \cdots \sigma_1(W_1 x))) : x, \nu(x) \in \mathbb{R}^d\}$, 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 NNs and convolutional NNs. Note that the Fourier Neural Operator [158] used in the experiments for NS can be also covered by the definition above, as it performs alternatively the convolution in the Fourier space.

Choosing penalization Ω Now we choose an Ω for the space above. Let us first introduce a practical way to bound the capacity of $\hat{\mathcal{G}} \subseteq \hat{\mathcal{F}}_{\mathrm{NN}}$. Proposition 3.9 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 3.9

If for all neural network $g \in \hat{\mathcal{G}}$, $\|g\|_\infty = \operatorname{ess\,sup}|g| \leq R$ and $\|g\|_{\mathrm{Lip}} \leq L$, with $\|\cdot\|_{\mathrm{Lip}}$ the Lipschitz semi-norm, then:

$$\log \mathcal{C}_{\hat{\mathcal{F}}}(\varepsilon, \hat{\mathcal{G}}) \leq \omega(R, L, \varepsilon) \tag{3.17}$$

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 3.3 (*[112], 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 first 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 3.10) 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 3.10). 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

$$\begin{aligned}
d_{[\mathcal{P}, \hat{\mathcal{F}}]}(g, g') &= \int_{\mathcal{A} \times \mathcal{A}'} \sup_{f \in \hat{\mathcal{F}}} |\|(f+g)(x) - y\|_2^2 - \|(f+g')(x) - y\|_2^2| \mathrm{d}\mathcal{P}(x, y) \\
&\leq \int_{\mathcal{A} \times T\mathcal{A}} \sup_{f \in \hat{\mathcal{F}}} \|(g - g')(x)\|_2 (\|(f+g)(x) - y\|_2 + \|(f+g')(x) - y\|_2) \mathrm{d}\mathcal{P}(x, y) \\
&\leq 2\sqrt{c} \int_{\mathbb{R}^d} \|(g - g')(x)\|_1 \mathrm{d}\mathcal{P}(x) \leq \varepsilon
\end{aligned}$$

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 the Theorem 3.3 on $\hat{\mathcal{G}}$, we then have the following inequality

$$\log \mathcal{C}\left(\frac{\varepsilon}{2\sqrt{c}}, \hat{\mathcal{G}}, L^1\right) \leq 2W \log \frac{8e\sqrt{c}DRL}{\varepsilon} \tag{3.18}$$

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:

$$\log \mathcal{C}\left(\frac{\varepsilon}{2\sqrt{c}}, \hat{\mathcal{G}}, L^1\right) \leq c_1 \log \frac{RL}{\varepsilon} + c_2 = \omega(R, L, \varepsilon) \tag{3.19}$$

for $c_1 = 2W$ and $c_2 = 2W \log 8e\sqrt{c}D$. □

This leads us to choose for Ω a strictly increasing function that bounds $\omega(R, L, \varepsilon)$. Given the inequality Eq. 3.17, this choice for Ω will allow us to bound practically the capacity of $\hat{\mathcal{G}}$.

Minimizing Ω will then reduce the effective capacity of the parametric set used to learn G_e. Concretely, we choose for Ω:

$$\Omega(G_e) = \|G_e\|_\infty^2 + \alpha \|G_e\|_{\mathrm{Lip}}^2 \tag{7}$$

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 Prop. 3.4 for our familly of NNs. Let $r = \sup_{g \in \hat{\mathcal{G}}} \Omega(g)$, 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 3.10

If $r = \sup_{g\in\hat{\mathcal{G}}} \Omega(g)$ is finite, the number of samples n in Eq. 3.13, required to satisfy the error bound in Proposition 3.4 with the same $\delta, \varepsilon, \varepsilon_1$ and ε_2 becomes:

$$n \geq \max\left\{\frac{64}{\varepsilon^2}\left(\frac{1}{m}\log\frac{4\mathcal{C}_{\hat{\mathcal{G}}}(\frac{\varepsilon_1}{16},\hat{\mathcal{F}})}{\delta} + \omega\left(r, \frac{\varepsilon_2}{16}\right)\right), \frac{16}{\varepsilon^2}\right\} \tag{3.20}$$

Proof: If $\Omega(G_e) \leq r$, 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 3.9. □

This means that the number of required samples will decrease with the size the largest possible $\Omega(g) = r$. The optimization process will reduce $\Omega(G_e)$ 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 3.1

Optimizing Eq. 1.9 for a given λ, we have that the number of samples n in Eq. 3.13 required to satisfy the error bound in Proposition 3.4 with the same $\delta, \varepsilon, \varepsilon_1$ and ε_2 is:

$$n \geq \max\left\{\frac{64}{\varepsilon^2}\left(\frac{1}{m}\log\frac{4\mathcal{C}_{\hat{\mathcal{G}}}(\frac{\varepsilon_1}{16},\hat{\mathcal{F}})}{\delta} + \omega\left(\lambda\kappa, \frac{\varepsilon_2}{16}\right)\right), \frac{16}{\varepsilon^2}\right\} \tag{3.21}$$

where $\kappa = \sum_{e\in E}\sum_{i=1}^{l}\int_0^T \left\|\frac{\mathrm{d}X_s^{e,i}}{\mathrm{d}t}\right\|^2 \mathrm{d}s$.

Proof: Denote $\mathcal{L}_\lambda(f, \{G_e\})$ the loss function defining Eq. 1.9. Consider a minimizer $(f^\star, \{G_e^\star\})$ of $\mathcal{L}_\lambda$. Then:

$$\mathcal{L}_\lambda(f^\star, \{G_e^\star\}) \leq \mathcal{L}_\lambda(0, \{0\}) = \kappa$$

which gives:

$$\forall e,\ \Omega(G_e^\star) \leq \sum_e \Omega(G_e^\star) \leq \lambda\kappa$$

Defining $\hat{\mathcal{G}} = \{g \in \hat{\mathcal{F}} \mid \Omega(g) \leq \lambda\kappa\}$, we then have that Eq. 1.9 is equivalent to:

$$\min_{f\in\hat{\mathcal{F}}, \{G_e\}_{e\in E}\in\hat{\mathcal{G}}} \quad \sum_{e\in E}\left(\frac{\Omega(G_e)}{\lambda} + \sum_{i=1}^{l}\int_0^T \left\|\frac{\mathrm{d}X_s^{e,i}}{\mathrm{d}t} - (f + G_e)(X_s^{e,i})\right\|^2 \mathrm{d}s\right) \tag{3.22}$$

and the result follows from Proposition 3.10. □

We can thus decrease the sample complexity in the chosen NN family by:

1. increasing the number of training environments engaged in the framework, and
2. decreasing $\Omega(G_e)$ for all G_e, with $\Omega(G_e)$ instantiated as in 3.4.3.

Table 3.12: Test MSE of experiments on LV ($m = 4, n = 1 \cdot K$) with different empirical norms.

Empirical Norm	$p = 1$	$p = 2$	$p = 3$	$p = 10$	$p = \infty$
Test MSE	2.30e-3	2.36e-3	2.34e-3	3.41e-3	6.12e-3

Ω provides a bound based on the largest output norm and the Lipschitz constant for a family of NNs. The experiments of Section 3.4.4 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.

Finally, let us note that constructing tight generalization bounds for neural networks is still an open research problem [197]. The results presented here should thus serve as heuristic arguments which we evaluate successfully in the following on three different datasets with different architectures in the following experiments (Section 3.4.4).

- *Optimizing Ω in practice*

In 3.4.3, 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 Ω defined as $\Omega(G_e) = \|G_e\|_\infty^2 + \alpha\|G_e\|_{\text{Lip}}^2$. This definition ensures the properties required to control the sample complexity.

However, in practice, both terms in $\Omega(G_e)$ are difficult to compute 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 networks, can be bounded by the spectral norms of the weight matrices. More specifically, as suggested in [39], 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 [189] for fast spectral norm approximation.

The infinity norm 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^p and L^∞ norms, respectively defined as $\|G_e\|_{L^p(\hat{\mathcal{P}}_e)} = \left(\frac{1}{n}\sum_{x\in\hat{\mathcal{P}}_e} |G_e(x)|^p\right)^{1/p}$ for $1 \leq p < \infty$ and $\|G_e\|_{L^\infty(\hat{\mathcal{P}}_e)} = \max_{x\in\hat{\mathcal{P}}_e} |G_e(x)|$. Here $|\cdot|$ is an ℓ^2 vector norm. Note that on a finite set of points, these norms reduce to vector norms $\|(|G_e(x_1)|, \ldots, |G_e(x_n)|)^\top\|_p$. They are then all equivalent on the space defined by the training set. Table 3.12 shows the results of experiments performed on LV equation with different $1 \leq p \leq \infty$. Overall we found that L^p for small values of p worked better and chose in our experiments to set $p = 2$.

Moreover, using both minimized quantities $\|G_e\|_{L^2(\hat{\mathcal{P}}_e)}^2$ 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 $\Omega(G_e)$. First, for any x in the compact support of $\mathcal{P}_e$, we have that, fixing some $x_0 \in \hat{\mathcal{P}}_e$:

$$|G_e(x)| \leq |G_e(x) - G_e(x_0)| + |G_e(x_0)|$$

For the first term:

$$|G_e(x) - G_e(x_0)| \leq \|G_e\|_{\text{Lip}}|x - x_0| \leq \Pi(G_e)|x - x_0|$$

and the support of $\mathcal{P}_e$ being compact by hypothesis, denoting by δ its diameter:

$$|G_e(x) - G_e(x_0)| \leq \delta\Pi(G_e)$$

Moreover, for the second term:

$$|G_e(x_0)| = \sqrt{|G_e(x_0)|^2} \leq \sqrt{L(G_e)}$$

and summing both contributions gives us the bound:

$$\|G_e\|_\infty \leq \delta\Pi(G_e) + \sqrt{L(G_e)}$$

so that:

$$\Omega(G_e) \leq (\delta + \alpha)\Pi(G_e) + \sqrt{L(G_e)}$$

Note that this estimation is a crude one and improvements can be made by considering the closest x_0 from x 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:

$$\forall x, \frac{1}{M^2}|G_e(x)|^2 \leq \left|\frac{G_e(x)}{x}\right|^2 \leq \frac{1}{m^2}|G_e(x)|^2$$

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 $|x|$ 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).

3.4.4 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 (MLP, ConvNet, and Fourier Neural Operator [158]). Each architecture is well-adapted to the corresponding dynamics. This also shows that the framework is valid for a variety of approximating functions.

• *Dynamics, environments, and datasets*

Lotka-Volterra (LV). This classical model [171] is used for describing the dynamics of interaction between a predator and a prey. The model dynamics follow the ODE:

$$\frac{\mathrm{d}u}{\mathrm{d}t} = \alpha u - \beta uv, \frac{\mathrm{d}v}{\mathrm{d}t} = \delta uv - \gamma v$$

with u, v the number of prey and predator, $\alpha, \beta, \gamma, \delta > 0$ defining how the two species interact. The initial conditions u_0^i, v_0^i are sampled from a uniform distribution $P_0 = \mathrm{Unif}([1, 2]^2)$. We characterize the dynamics by $\theta = (\frac{\alpha}{\beta}, \frac{\gamma}{\delta}) \in \Theta = \{0.5, 1, 1.44, 1.5, 1.86, 2\}^2$. An environment e is then defined by parameters θ_e sampled from a uniform distribution over the parameter set Θ. 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 (GS). This reaction-diffusion model is known for its complex spatio-temporal behavior with a relatively simple formulation [210]. The governing PDE is:

$$\frac{\partial u}{\partial t} = D_u \Delta u - uv^2 + H(1-u), \frac{\partial v}{\partial t} = D_v \Delta v + uv^2 - (H+k)v$$

where the u, v represent the concentrations of two chemical components in the spatial domain S with periodic boundary conditions. D_u, D_v denote the diffusion coefficients respectively for u, v, and are held constant to $D_u = 0.2097, D_v = 0.105$, and H, k are the reaction parameters depending on the environment. As for the initial conditions $(u_0, v_0) \sim P_0$, we place 3 2-by-2 squares at uniformly sampled positions in S to trigger the reactions. The values of (u_0, v_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 $\theta_e = (H_e, k_e) \in \Theta = \{(0.037, 0.060), (0.030, 0.062), (0.039, 0.058)\}$. We consider a set of θ_e parameters uniformly sampled from the environment distribution Q on Θ.

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

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

where v is the velocity field, $w = \nabla \times v$ is the vorticity, both v, w lie in a spatial domain S with periodic boundary conditions, ν is the viscosity and ξ is the constant forcing term in the domain S. We fix $\nu = 10^{-3}$ across the environments. We sample the initial conditions $w_0^e \sim P_0$ as in [158]. An environment e is defined by its forcing term $\xi_e \in \Theta_\xi = \{\xi_1, \xi_2, \xi_3, \xi_4\}$ with

$$\begin{aligned}
\xi_1(x, y) &= 0.1(\sin(2\pi(x+y)) + \cos(2\pi(x+y))) \\
\xi_2(x, y) &= 0.1(\sin(2\pi(x+y)) + \cos(2\pi(x+2y))) \\
\xi_3(x, y) &= 0.1(\sin(2\pi(x+y)) + \cos(2\pi(2x+y))) \\
\xi_4(x, y) &= 0.1(\sin(2\pi(2x+y)) + \cos(2\pi(2x+y)))
\end{aligned}$$

where $(x, y) \in S$ is the position in the domain S. We uniformly sampled a set of forcing terms from Q on Θ_ξ.

Datasets. For training, we create two datasets for LV 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 8 trajectories (thus corresponding to $n = 8 \cdot K$ data points) 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 on novel environments. We simulate 1 trajectory ($n = 1 \cdot K$ data points) for training. We create two datasets for further validation of *LEADS* with GS and NS. For GS, we simulate trajectories of $K = 10$ steps with $\Delta t = 40$. We choose 3 parameters and simulate 1 trajectory ($n = 1 \cdot K$ data points) for training. For NS, we simulate trajectories of $K = 10$ steps with $\Delta t = 1$. We choose 4 forcing terms and simulate 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 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 $\hat{\mathcal{P}}_e$, $\forall e, X_0^{e,i} = X_0^i$. LV and GS data are simulated with the DOPRI5 solver in NumPy [76, 109]. NS data is simulated with the pseudo-spectral method as in [158].

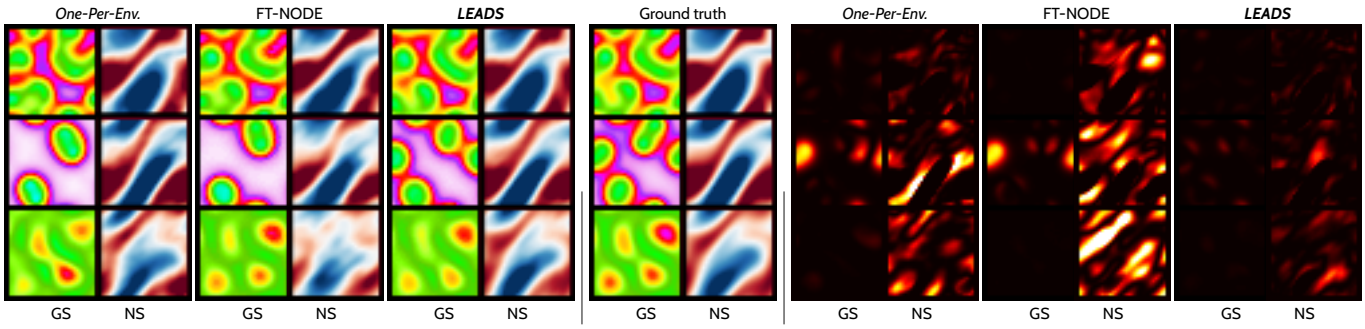


Figure 3.6: Left: final states for GS and NS predicted by the two best baselines (*One-Per-Env.* and FT-NODE) and *LEADS* compared with ground truth. Different environment are arranged by row (3 in total). Right: the corresponding MAE error maps, the scale of the error map is [0, 0.6] for GS, and [0, 0.2] for NS; darker is smaller. Full trajectories are provided in Figures 3.7 to 3.10

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.

- *Experimental settings and baselines*

We validate *LEADS* in two settings: in the first one all the environments in E are available at once and then F and all the G_es are all trained on E. In the second one, training has been performed on E as before, and we consider a novel environment $e' \notin E$: 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*:

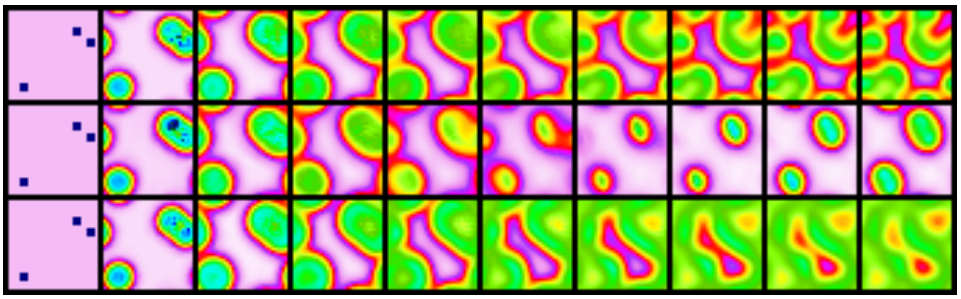
1. ***One-For-All***: learning on the entire dataset $\hat{\mathcal{P}}$ over all environments with the sum of a pair of NNs $F + g$, with the standard ERM principle, as in done in Chapter 2 (but with full supervision here and thus with $\mathcal{H} = \mathrm{id}$). Although this is equivalent to use 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.

2. ***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 $(F + g)_e$.

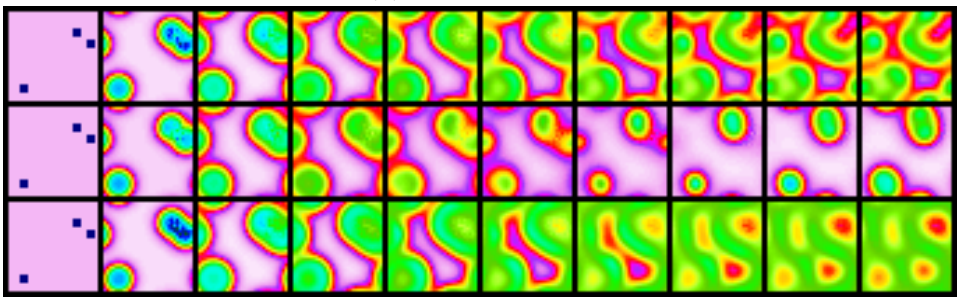
3. Factored Tensor RNN or **FT-RNN** [248]: it modifies the recurrent neural network to integrate a one-hot environment code into each linear transformation of the network. Instead of being encoded in a separate function G_e like in *LEADS*, the environment appears here as an extra one-hot input for the RNN linear transformations. This can be implemented for representative SOTA (spatio-)temporal predictors such as GRU [56] or PredRNN [273].

4. **FT-NODE**: a baseline for which the same environment encoding as FT-RNN is incorporated in a Neural ODE [50].

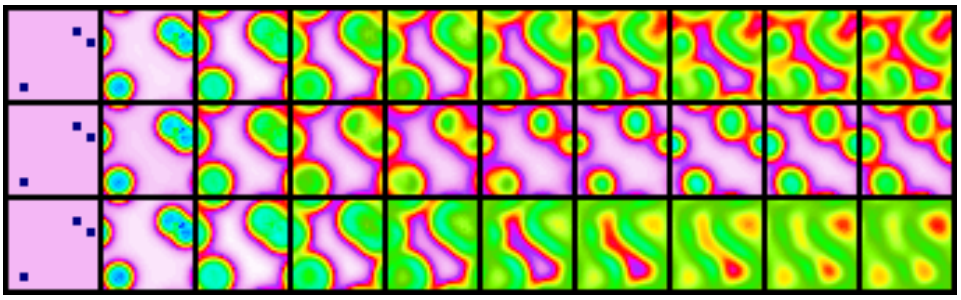
5. Gradient-based Meta Learning or **GBML-like** method: we propose a GBML-like baseline which can directly compare to our framework. It follows the principle of MAML [91], by training



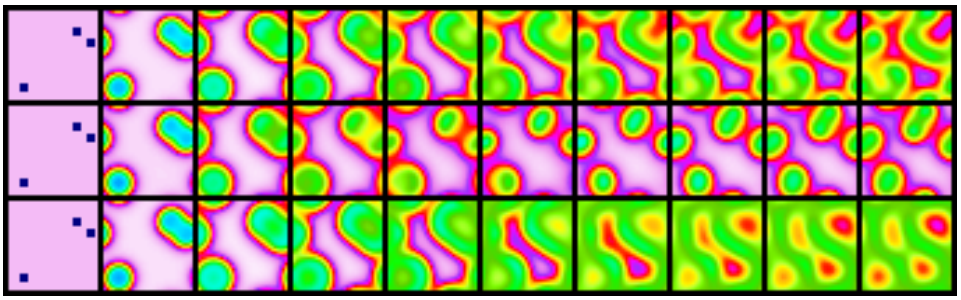
(a) *One-Per-Env.*



(b) FT-NODE

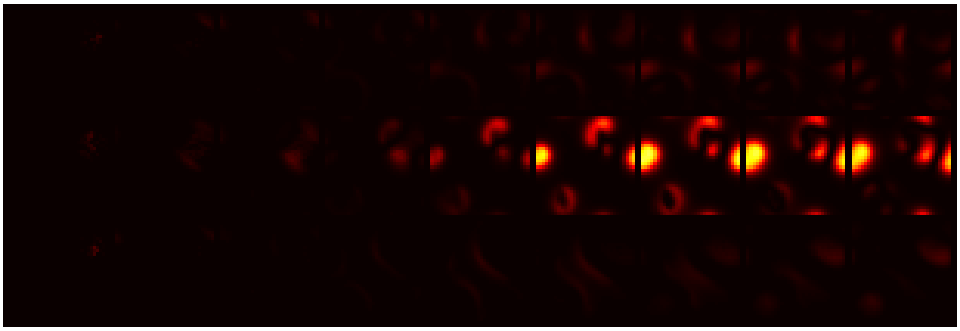


(c) *LEADS*

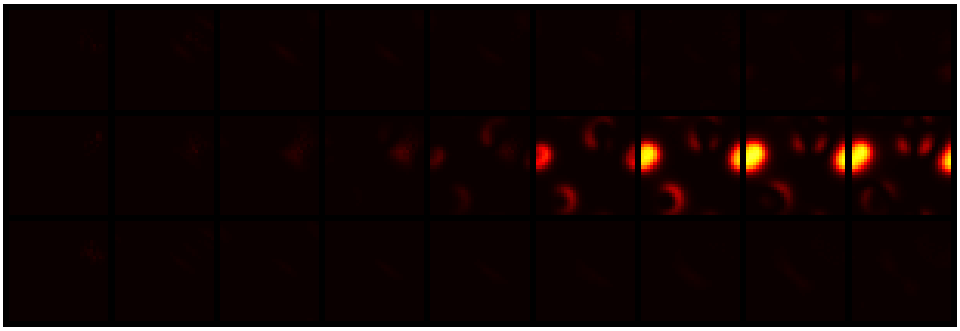


(d) Ground truth

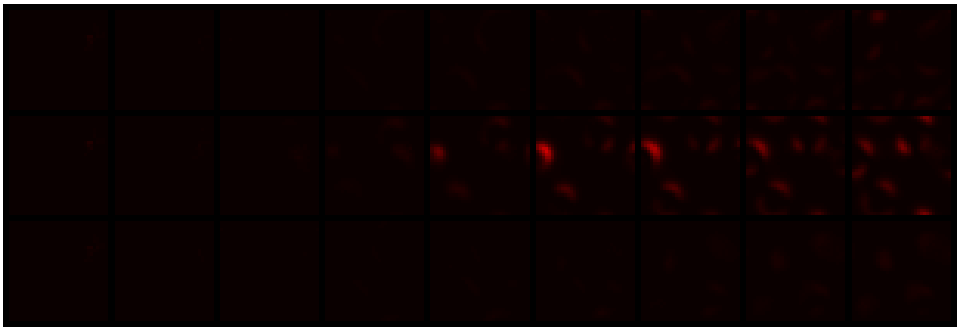
Figure 3.7: Full-length prediction comparison of Fig. 3.6 for GS. In each figure, from top to bottom, the trajectory snapshots are output respectively from 3 training environments. The temporal resolution of each sequence is $\Delta t = 40$.



(a) Difference between *One-Per-Env.* and Ground truth

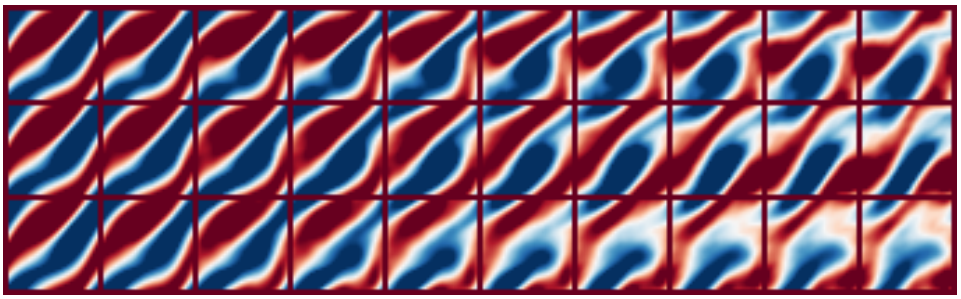


(b) Difference between FT-NODE and Ground truth

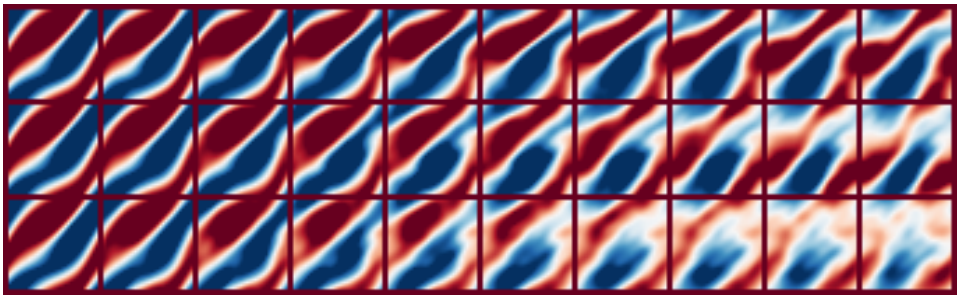


(c) Difference between *LEADS* and Ground truth

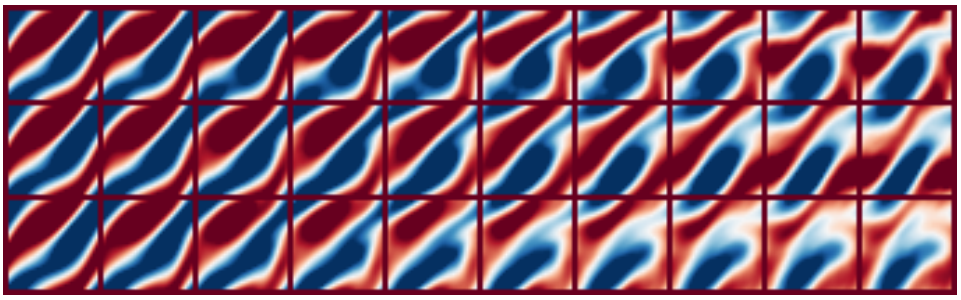
Figure 3.8: Full-length error maps of Fig. 3.6 for GS. In each figure, from top to bottom, the trajectory snapshots correspond to 3 training environments, one per row. The temporal resolution of each sequence is $\Delta t = 40$.



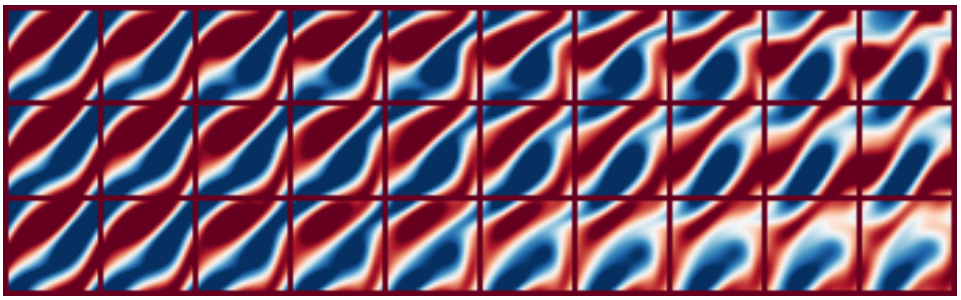
(a) *One-Per-Env.*



(b) FT-NODE

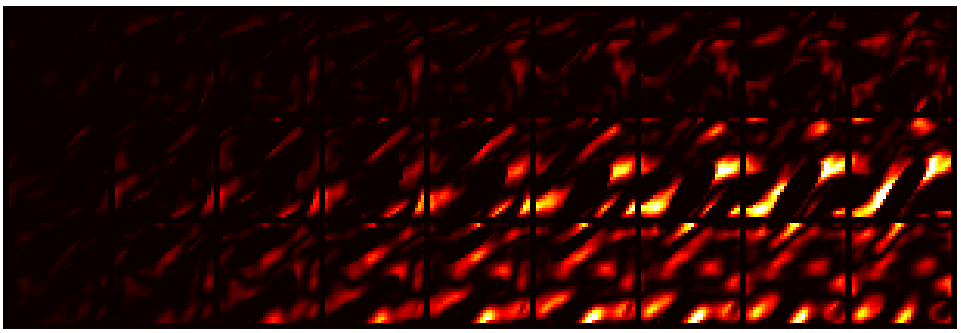


(c) *LEADS*

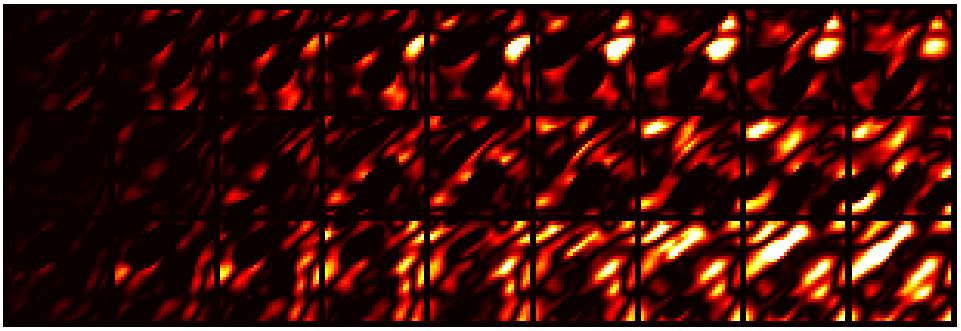


(d) Ground truth

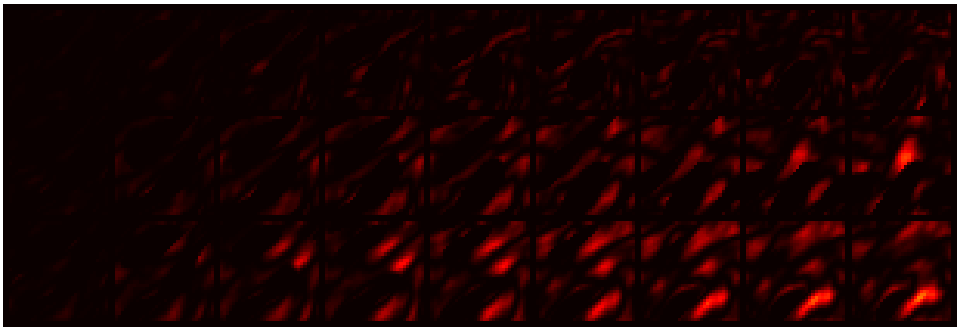
Figure 3.9: Full-length prediction comparison of Fig. 3.6 for NS. In each figure, from top to bottom, the trajectory snapshots correspond to 3 training environments. The temporal resolution of each sequence is $\Delta t = 1$.



(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 3.10: Full-length error maps of Fig. 3.6 for NS. In each figure, from top to bottom, the trajectory snapshots correspond to from 3 training environments. The temporal resolution of each sequence is $\Delta t = 1$.

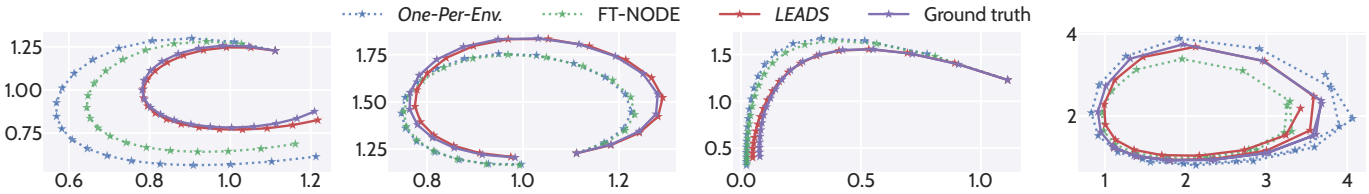


Figure 3.11: Test predicted trajectories in phase space with two baselines (*One-Per-Env.* and FT-NODE) and *LEADS* compared with ground truth for LV for 4 envs., one per figure from left to right. Quantity of the prey u and the predator v respectively on the horizontal and the vertical axis. Initial state is the rightmost end-point of the figures and it is common to all the trajectories.

One-For-All at first which provides an initialization near to the given environments like GBML does, then fitting it individually for each training environment.

6. ***LEADS no min.***: ablation baseline, our proposal without the $\Omega(G_e)$ penalization.

A comparison with the different baselines is proposed in Table 3.14 for the three dynamics. For concision, we provide a selection of results corresponding to 1 training trajectory per environment for LV and GS and 8 for NS. This is the minimal training set size for each dataset. Further experimental results when varying the number of environments from 1 to 8 are provided in Fig. 3.12 and Table 3.15 for LV.

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

1. ***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.

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

3. ***Pre-trained-F-Plus-Trained-G_e***: we train a $G_{e'}$ on each dataset $\hat{\mathcal{P}}_{e'}$ with a fixed pre-trained F, i.e. $F + G_{e'}$ is fitted to data trajectories of the new environment and only $G_{e'}$s is learned.

We compare the test error evolution during training for the 3 schemes above for a comparison of convergence speed and performance. Results are summarised in Fig. 3.13 for LV and Table 3.13 provides similar quantitative results for GS and NS.

Details on experiments with a varying number of environments This paragraph covers the experiments conducted for linear ODEs (Fig. 3.5) and LV (Fig. 3.12) which aim to confront *LEADS* w.r.t. to the theoretical bounds and to the baselines by varying the number of environments available for a given model.

In order to guarantee the comparability of the test-time results, we need to use the same test set when varying the number of environments. Therefore, we have first generated a global set of environments, separated it into subgroups for training, then tested these separately trained models on the global test set. More precisely:

- In the training phase, we consider $M = 8$ environments in total in the environment set E_{total}. The environments are then arranged into $b = 1, 2, 4$ or 8 disjoint groups of the same size, i.e. $\{E_1, \dots, E_b\}$ such that $\bigcup_{i=1}^{b} E_i = E_{\text{total}}$, $\text{card}(E_1) = \cdots = \text{card}(E_b) = \lfloor M/b \rfloor =: m$, where m is the number of environments per group, and $E_i \cap E_j = \emptyset$ 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

Table 3.13: Results on 2 novel environments for LV, GS, and NS at different traning 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
LV ($n = 1 \cdot K$)	*Pre-trained-F-Only*	0.36	⟶	⟶
	One-Per-Env. from scratch	0.23	8.85e-3	3.05e-3
	Pre-trained-F-Plus-Trained-G_e	0.73	**1.36e-3**	**1.11e-3**
GS ($n = 1 \cdot K$)	*Pre-trained-F-Only*	5.44e-3	⟶	⟶
	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**
NS ($n = 8 \cdot K$)	*Pre-trained-F-Only*	1.75e-1	⟶	⟶
	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**

Table 3.14: Results for LV, GS, and NS datasets, trained on m envs. with n data points per env.

Method	LV ($m = 10, n = 1 \cdot K$)		GS ($m = 3, n = 1 \cdot K$)		NS ($m = 4, n = 8 \cdot K$)	
	MSE train	MSE test	MSE train	MSE test	MSE train	MSE test
One-For-All	4.57e-1	5.08±0.56 e-1	1.55e-2	1.43±0.15 e-2	5.17e-2	7.31±5.29 e-2
One-Per-Env.	2.15e-5	7.95±6.96 e-3	8.48e-5	6.43±3.42 e-3	5.60e-6	1.10±0.72 e-2
FT-RNN [248]	5.29e-5	6.40±5.69 e-3	8.44e-6	8.19±3.09 e-3	7.40e-4	5.92±4.00 e-2
FT-NODE	7.74e-5	3.40±2.64 e-3	3.51e-5	3.86±3.36 e-3	1.80e-4	2.96±1.99 e-2
GBML-like	3.84e-6	5.87±5.65 e-3	1.07e-4	6.01±3.62 e-3	1.39e-4	7.37±4.80 e-3
LEADS no min.	3.28e-6	3.07±2.58 e-3	7.65e-5	5.53±3.43 e-3	3.20e-4	7.10±4.24 e-3
LEADS (Ours)	5.74e-6	**1.16±0.99 e-3**	5.75e-5	**2.08±2.88 e-3**	1.03e-4	**5.95±3.65 e-3**

E_i. For example, for *LEADS* with b environment groups, we instantiate $LEADS_1, \ldots, LEADS_b$ respectively on $E_1, \ldots, E_b$. 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 Fig. 3.12 that each group of plots indeed starts from the same point.

- In the test phase, the performance of the model trained with the group 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 figures 3.5 and 3.12 is calculated on the same total test set, which guarantees the comparability between results.

Implementation. We used 4-layer MLPs for LV, 4-layer ConvNets for GS and Fourier Neural Operator (FNO) [158] for NS. For FT-RNN baseline, we adapted GRU [56] for LV and PredRNN [272] for GS and NS. We apply the Swish function [221] as the default activation function. Networks are integrated in time with RK4 (LV, GS) or Euler (NS), using the basic back-propagation through the internals of the solver. We apply an exponential Scheduled Sampling [144] with exponent of 0.99 to stabilize the training. We use the Adam optimizer [140] with the same learning rate 10^{-3} and $(\beta_1, \beta_2) = (0.9, 0.999)$ across the experiments. For the hyperparamters, we chose respectively $\lambda = 5 \times 10^3, 10^2, 10^5$ and $\alpha = 10^{-3}, 10^{-2}, 10^{-5}$ for LV, GS and NS. All experiments are performed with a single NVIDIA Titan Xp GPU.

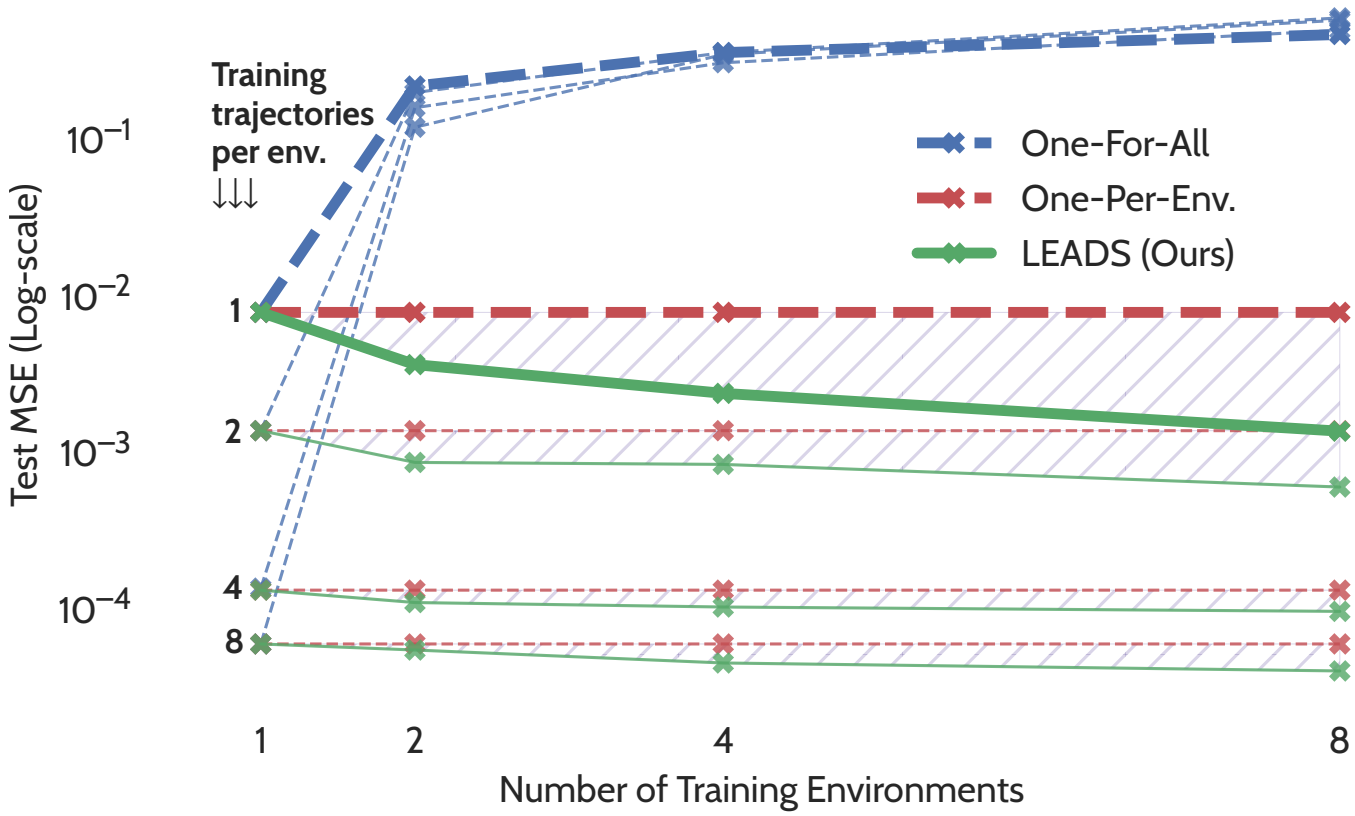


Figure 3.12: Test error for LV w.r.t. the number of environments. We apply the models in 1 to 8 environments. 4 groups of curves correspond to models trained with 1 to 8 trajectories per env. All groups highlight the same tendencies: increasing *One-For-All*, stable *One-Per-Env.*, and decreasing *LEADS*.

• *Experimental results*

All environments available at once. We show the results in Table 3.14. For LV 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. Comparing 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. Fig. 3.11 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 GS and NS systems. The error is reduced by: around 2/3 (GS) and 45% (NS) w.r.t. *One-Per-Env.*; over 60% (GS) and 15% (NS) w.r.t. *LEADS no min.*; 45-75% (GS) and 15-90% (NS) w.r.t. other baselines. In Fig. 3.6, 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 less errors through the integration, which suggests that *LEADS* indeed alleviates overfitting.

The LV experiments (see Fig. 3.12 and Table 3.15) allow us to analyze the behavior of the different 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 1, 2, 4 and 8 training trajectories per environment. The main observations are the following. 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 fit with one single model. The performance of *One-Per-Env.* (in red), for which models are trained independently for each environment, is of course constant when the number of environments changes. *LEADS* (green) circumvents these issues and shows that the

Table 3.15: Detailed results of evaluation error in test on LV systems for Fig. 3.12. 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/env.	Method	$m = 1$	$m = 2$	$m = 4$	$m = 8$
$n = 1 \cdot K$	*One-For-All*	7.87±7.54 e-3	0.22±0.06	0.33±0.06	0.47±0.04
	One-Per-Env.	7.87±7.54 e-3	→	→	→
	FT-RNN	4.02±3.17 e-2	1.62±1.14 e-2	1.62±1.40 e-2	1.08±1.03 e-2
	FT-NODE	7.87±7.54 e-3	7.63±5.84 e-3	4.18±3.77 e-3	4.92±4.19 e-3
	GBML-like	7.87±7.54 e-3	6.32±5.72 e-2	1.44±0.66 e-1	9.85±8.84 e-3
	LEADS (Ours)	7.87±7.54 e-3	**3.65±2.99 e-3**	**2.39±1.83 e-3**	**1.37±1.14 e-3**
$n = 2 \cdot K$	*One-For-All*	1.38±1.61 e-3	0.22±0.04	0.36±0.07	0.60±0.11
	One-Per-Env.	1.38±1.61 e-3	→	→	→
	FT-RNN	7.20±7.12 e-2	2.72±4.00 e-2	1.69±1.57 e-2	1.38±1.25 e-2
	FT-NODE	1.38±1.61 e-3	9.02±8.81 e-3	1.11±1.05 e-3	1.00±0.95 e-3
	GBML-like	1.38±1.61 e-3	9.26±8.27 e-3	1.17±1.09 e-2	1.96±1.95 e-2
	LEADS (Ours)	1.38±1.61 e-3	**8.65±9.61 e-4**	**8.40±9.76 e-4**	**6.02±6.12 e-4**
$n = 4 \cdot K$	*One-For-All*	1.36±1.25 e-4	0.19±0.02	0.31±0.04	0.50±0.04
	One-Per-Env.	1.36±1.25 e-4	→	→	→
	FT-RNN	8.69±8.36 e-4	3.39±3.38 e-4	3.02±1.50 e-4	2.26±1.45 e-4
	FT-NODE	1.36±1.25 e-4	1.74±1.65 e-4	1.78±1.71 e-4	1.39±1.20 e-4
	GBML-like	1.36±1.25 e-4	2.57±7.18 e-3	2.65±3.26 e-3	2.36±3.58 e-3
	LEADS (Ours)	1.36±1.25 e-4	**1.10±0.92 e-4**	**1.03±0.98 e-4**	**9.66±9.79 e-5**
$n = 8 \cdot K$	*One-For-All*	5.98±5.13 e-5	0.16±0.03	0.35±0.06	0.52±0.06
	One-Per-Env.	5.98±5.13 e-5	→	→	→
	FT-RNN	2.09±1.73 e-4	1.18±1.16 e-4	1.13±1.13 e-4	9.13±8.31 e-5
	FT-NODE	5.98±5.13 e-5	6.91±4.46 e-5	7.82±6.95 e-5	6.88±6.39 e-5
	GBML-like	5.98±5.13 e-5	1.02±1.68 e-4	1.41±2.68 e-4	0.99±1.53 e-4
	LEADS (Ours)	5.98±5.13 e-5	**5.47±4.63 e-5**	**4.52±3.98 e-5**	**3.94±3.49 e-5**

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.

Learning on novel environments. We also demonstrate how the pre-trained dynamics can help to fit a model for novel, unseen environments. We took an F pre-trained by *LEADS* on a set of LV environments. Fig. 3.13 shows the evolution of the test loss during training for three systems: a F function pre-trained by *LEADS* on a set of LV 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 environments. 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 dotted line), showing that the pretrained F indeed captures relevant information common to all environments. Similar results are also observed with the GS and NS datasets (cf. Table 3.13). These empirical results clearly show that the learned shared dynamics accelerate and improve the learning in novel environments.

3.5 Conclusion

APHYNITY and *LEADS* offer two case studies of the generalization problem when learning from dynamical data with differential neural models.

The first tackles the case where prior knowledge is available: this, of course, is a natural situation as application domains often have an important body of existing modelling literature. Generalizing this approach so as to make it applicable to real world datasets is the obvious following step. In

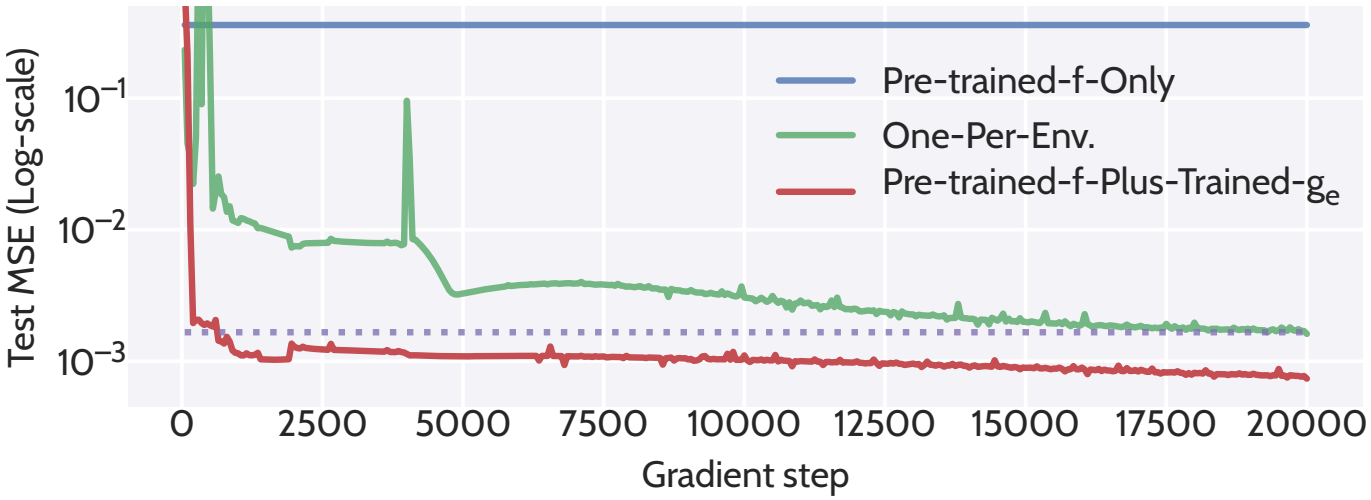


Figure 3.13: Test error evolution during training on 2 novel environments for LV.

particular, this approach should be extended to the more realistic setting of partially observable dynamics, as presented in Chapter 2. This last context however makes it more difficult to offer the theoretical guarantees which we have shown in the fully observed case and finding the right formulation is an essential and difficult challenge for future work, especially whe considering parameter identification.

The second constitutes a first step for the construction of Meta-learning models of dynamical systems. This is still a novel endeavour and natural extensions would study more sophisticated representations of the environments, characterising the geometry of their respective dynamics w.r.t. the common term and exploring few shot extrapolation to unseen dynamics to give a few examples. Quite generally, such a program could lead to more robust representations of the dynamics of the observable world with the ability to construct counter-factuals, tackle changes in the distribution of observed data, etc. which could have many applications in various fields.

PART B

A Dynamical Analysis of Neural Networks

In this second part of the thesis, we take a dual approach to what was presented in previous chapters: instead of using neural networks in order to model real world dynamics, we turn to analyzing the dynamics of deep neural networks. More precisely, we consider the dynamics induced both during their training in Chapter 5 and during inference in Chapter 4. In both cases, we aim to characterize those dynamics appropriately, thus trying to gain useful insights in the use of DNNs for standard learning tasks.

Our main contributions are the following:

- empirical observations allow us to link the dynamics of NNs to those of the construction of the Optimal Transport mappings between data distributions;
- those links are translated into a generic variational formulation which is shown to be more robust, more flexible with a better performance for classification and generative tasks;
- reformulating NNs training in the continuous, infinite width limit, the Neural Tangent Kernel theory reveals key insights about the importance of architectural choices for the discriminator in adversarial generative models.

Chapter 4

A Variational Theory of Deep Neural Networks

This chapter presents a (tentative) variational formulation of the training of Deep Neural Networks. The general message is that "well-trained" NNs, meaning those which successfully accomplish the tasks they are trained for, even though they are not constrained enough to do so, are actually implicitly biased towards a smaller class of functions. Those functions are endowed with desirable properties which we characterize as the solutions to variational problems broadly inspired from Optimal Transport.

This chapter covers the content of the two following publications

- *A Principle of Least Action for the Training of Neural Networks*, S Karkar*, I Ayed*, E de Bézenac*, P Gallinari, ECML 2020, Best student paper award Runner-up;
- *CycleGAN Through the Lens of (Dynamical) Optimal Transport*, E de Bézenac*, I Ayed*, P Gallinari, ECML 2021.

An application to the layer-wise training of classification networks which is also quickly presented is currently under review

- *Block-wise Training of Residual Networks via the Minimizing Movement Scheme*, Skander Karkar, Ibrahim Ayed, Emmanuel de Bézenac, Patrick Gallinari, preprint.

4.1 Introduction

Deep neural networks (DNNs) have repeatedly shown their ability to solve a wide range of challenging tasks, while often having many more parameters than there are training samples. Such a performance of over-parametrized models is counter-intuitive as they seem to adapt their complexity to the given task, systematically achieving a low training error without suffering from over-fitting as could be expected [29, 198, 288]. This is the case for classification which is the task of finding, given an input data-point, the corresponding label among a finite set of possible labels, *e.g.* finding the nature of an object represented in a natural image. For this task, with the availability of reasonably sized supervision datasets, over-parametrized DNN models are able to achieve high accuracies without much constraints or regularization. Classification

Even more spectacularly, DNNs seem to succeed in solving some ill-posed tasks while seemingly not being provided with enough specification to do so. This is the case for the task of Unsupervised Domain Translation. Given pairs of elements from two different domains, *domain translation* consists in learning a mapping from one domain to another, linking paired elements together. A wide range of problems can be formulated as translation, including image-to-image [125], video-to-video [22], image

captioning [290], natural language translation [19], etc. However, obtaining paired examples is often difficult and for this reason has motivated a growing interest towards the more general *unpaired* or *unsupervised* setting where only samples from both domains are available without pairing. A seminal and influential work for solving Unsupervised Domain Translation (UDT) has been the CycleGAN model [295]. It has spurred many variants and extensions leading to impressive results in several application domains [145, 286, 60, 58, 84].

Those observations are in stark contradiction with the classical statistical practice of selecting a class of functions complex enough to represent the coherent patterns in the data, and simple enough to avoid spurious correlations [30, 110]; but also with a naive intuitive understanding of how DNNs work and, as we show, for the case of the UDT task, with a simple mathematical analysis. And, although this behavior has sparked much recent work towards explaining neural networks' success ([72, 127, 203, 216] and [96, 280, 190]), it still remains poorly understood.

Our aim in this work is to focus on the implicit biases which drive the construction and training of the DNN. More precisely, while DNNs have been shown to be universal approximators in many settings [139], our reasoning is that the DNNs which are learned in practice incorporate subtle prior knowledge about the task at hand and the desiderata for it. This translates into the choices made for the parametrization, the architecture, the parameter initialization and the optimization algorithm, which all contribute to the success of the model. Our objective is then to uncover some of these hidden biases and to highlight their link with generalization performance, in particular through the lens of dynamical systems.

Indeed, it is obvious for any Deep Learning practitioner that the effective expressiveness of a given DNN model, even for a fixed architecture and parametrization, depends heavily on other choices such as the way the weights are initialized or the choice of the learning rate. A simple (albeit trivial) example would be to consider what happens when a null learning rate is taken: the only "learnable" function is then the constant one defined at the initialization. Another (almost equally trivial) example would be to consider what happens when all weights are initialized as null matrices in a feed-forward NN: the only function which can be reached is then the null one, thus also making learning impossible. More subtle examples abound and, in practice, once the architecture and the loss are chosen, one has to carefully tune several hyper-parameters, before succeeding in solving a given task.

Our goal will be to uncover what the choices which define successfully trained DNNs lead to in terms of the actually learned functions. We will be more particularly interested in residual networks (ResNets) [113, 116], now ubiquitous in many applications including the two tasks mentioned earlier. Note that our analysis can largely be extended to other architectures (more modern ResNet variants, Transformers, etc.) as long as skip connections are used.

The use of residual connections has made it possible to learn very complex non-linear functions by improving the trainability of very deep networks, and has been found to improve generalization in practice. Links have been derived between these networks and dynamical systems as a ResNet can be seen as a forward Euler scheme discretization of an associated ordinary differential equation (ODE) [275]:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + v_k(\mathbf{x}_k) \;\longleftrightarrow\; \frac{\mathrm{d}\mathbf{x}_t}{\mathrm{d}t} = v_t(\mathbf{x}_t) \tag{4.1}$$

This link has yielded many exciting results, *e.g.* new architectures [174] and reversible networks [48]. Here, we make use of this analogy in order to analyze the behavior of residual networks by studying their associated differential flows. Adopting this dynamical point of view allows us to leverage the theories and mathematical tools developed to study, approximate and apply differential equations and, here in particular, those related to Dynamical Optimal Transport.

More specifically, we start by conducting experiments in which we observe how neural networks displace their inputs through layers which here correspond to the time variable in eq. 4.1 in the case of classification and UDT tasks. For both, we measure a strong empirical correlation between good

test performance and neural networks with low kinetic energy along their transport flow. Those observations lead us to the hypothesis that good choices of hyper-parameters, at least in the cases studied here, lead to neural mappings which minimize the kinetic energy.

Both in order to test this hypothesis and in order to construct better performing, more flexible models which are less sensitive to hyper-parameter choices, we reformulate the training problem as follows: retrieve the network which solves the task using the principle of least action, *i.e.* expending as little kinetic energy as possible. This problem, in its probabilistic formulation, is tightly linked with and inspired by the well-known problem of finding an optimal transportation map [235]. Exploiting this link yields new insights into neural networks' generalization capabilities, and provides a novel algorithm that automatically adapts to the complexity of the data and robustly improves the network's performance, including in low data regimes, without slowing down the training.

To summarize, our contributions are the following:

- Through the dynamic viewpoint, we highlight the *low-energy bias* of ResNets.
- We formulate a Least Action Principle for the training of Neural Networks.
- We prove existence and regularity results for networks with minimal energy.
- We provide an algorithm for retrieving minimal energy networks compatible with different architectures, which leads to better generalization performance on different classification tasks as well as to a well-defined UDT problem[1], without complexifying the architecture.

4.2 Related work

That ResNets [113, 116] are naturally biased towards minimally transforming their input, especially for later blocks and deeper networks, is already shown in [131], which found that the first blocks learn new representations while later blocks only slowly refine those representations. [111] found that the deeper the network the more its blocks minimally move their input. Both were inspirations for this work.

The ODE point of view of ResNets has inspired new architectures [48, 108, 174, 232]. Others were inspired by numerical schemes to improve stability, e.g. [48] add a penalty term that encourages the weights to vary smoothly from layer to layer and [291] replicate an Euler scheme and study the effect of diminishing the discretization step-size. More recently, [279] accelerate the training of [50]'s model for generative tasks using the link with dynamical transport. But most often, regularization is achieved by penalization of the weights (e.g. spectral norm regularization [285], smoothly varying weights [48]).

OT theory was used in [247] to analyse deep gaussian denoising autoencoders (not necessarily implemented through residual networks) as transport systems. In the continuous limit, they are shown to transport the data distribution so as to decrease its entropy. Closer to this work, the dynamical formulation of OT is used in [71] for the problem of unsupervised domain translation.

Regarding UDT and as discussed in the remainder of the text, our work has been motivated in part by the observations in papers such as [96, 31] which have linked well-behaved UDT models with a notion of simplicity. Our formalism here tries to frame it in a more rigorous and more useful formulation, making it in particular task-dependent. Moreover, similarly to us, [96, 32] show that learning a one-sided mapping is possible but do not directly obtain the inverse mapping as we do. Others have tried a hybrid approach between paired and unpaired translation [263], which still doesn't solve the ill-posedness problem as there generally still are infinitely many possible mappings. Also similar to us, [100] uses a progressive interpolation. In the domain adaptation field, using Optimal

[1]Whereas the seminal CycleGAN model is not.

Transport to help a classifier extrapolate has been around for some years, e.g. [64, 68] use a transport cost to align two distributions. The task, although related, is clearly different and so are the methods they develop. Finally, [172] also try to regularize CycleGAN through OT but use barycenters from the optimal plan obtained in the discrete, static setting in order to guide the mapping instead of seeing it directly as an OT map (or as biased towards it as we claim), thus not explaining why CycleGAN works in practice.

4.3 Mathematical preliminaries

4.3.1 Transporting measures

The viewpoint in this chapter is to study the transformation of data through a DNN from an initial distribution $\mu_0 = \alpha$ into a final distribution $\mu_1 = \beta$ while accomplishing a given task. In order to do so, we need to formalize the notion of transforming distributions of data.

This is done here with the notion of *push-forward distributions* which are defined in the following way:

Definition 4.1 (*Push-forward distributions*)

Let μ a distribution and f a measurable map. Then the push-forward distribution $f_\sharp\mu$ is defined, for every measurable set A, with:

$$f_\sharp\mu(A) = \mu(f^{-1}(A))$$

Equivalently, it can also be defined with:

$$\int \phi \mathrm{d} f_\sharp\mu = \int \phi \circ f \mathrm{d}\mu$$

for any measurable function ϕ.

In the following, we will be more specifically interested in the properties of the trajectory $\{\mu_t\}$ between α and β which, in our case, correspond to the intermediary layers of the studied DNN and are thus defined via push-forward transformations of μ_0.

Note that in the following we will interchangeably use the notions of probability measures and distributions, the distribution associated with a certain probability measure μ being defined through

$$\forall \phi, < \mu, \phi >= \int \phi \mathrm{d}\mu$$

where ϕ is taken in an appropriate dual function space according to the context.

4.3.2 ODEs, flows and continuity equations

Following through with the point of view outlined in the previous paragraph as well as in Chapter 1, we see a DNN as a way to *transport* data-points in a way such that solving the task at hand becomes easy. For example, a neural classifier moves the input data-points so that a simple classifier, *e.g.* a linear one, can separate the points belonging to different classes.

There can then be two ways to look at this transformation: either as the DNN transporting data point by point or as a function acting globally on the initial data distribution. Continuity equations are a convenient tool for us to change from one point of view to the other.

Let us start from a residual DNN, seen as the flow of an ODE, acting on a data-point x by putting it in a trajectory $\{X_t\}$ which verifies:

$$X_0 = x \qquad\qquad \forall t \in [0,1], \frac{\mathrm{d}X_t}{\mathrm{d}t} = v_t^\theta(X_t)$$

Here θ refers to all trainable parameters of the NN while the vs typically designates the action of the residual blocks. In other words, time here refers to the depth of the network, meaning that time t can be equated with layer tT in a purely residual network of overall depth T while changing θ allows to control the trajectory of the data-point x.

Because most standard activation functions are globally lipschitz, it is reasonable to assume that $v_t^\theta(\cdot)$ is globally lipschitz. By standard argument, this ensures the ODE above has a unique solution which is defined over $[0, 1]$. The output of the DNN is X_1.

Consequently, the flow of the above ODE is well-defined and we denote it $\phi_t^x = X_t$, which means that ϕ_t^x is the point at time t of the trajectory starting from x at time 0 and following the above ODE defined by the velocity field v^θ. In particular, $\phi_t^\cdot$ gives the transformation of the data after time t and the data distribution after time t is then $(\phi_t^\cdot)_\sharp \mu_0$. Let us put $\mu_t = (\phi_t^\cdot)_\sharp \mu_0$. The immediate question which can then be asked is about how we can characterize the trajectory $\{\mu_t\}$ of the data distribution through the network.

First, suppose μ_0 is absolutely continuous and has a density ρ_0 w.r.t. the Lebesgue measure. Then, because v_t^θ is lipschitz for all t, by an argument for example outlined in [6], μ_t also has a density $\rho_t(\cdot)$ which verifies:

$$\frac{\mathrm{d}\rho_t}{\mathrm{d}t} + \nabla \cdot (\rho_t v_t^\theta) = 0$$

which is the classical *continuity equation* and where $\nabla\cdot$ is the (spatial) divergence operator[2].

In the general case, μ_t is solution to the continuity equation driven by v^θ

$$\frac{\mathrm{d}\mu_t}{\mathrm{d}t} + \nabla \cdot (\mu_t v_t^\theta) = 0$$

with initial condition μ_0 in the distributional sense [238] which means that, for any smooth and compactly supported test function $f(t, x)$, we have

$$\int (\frac{\partial f}{\partial t}(t, \cdot) + \nabla f(t, \cdot) \cdot v_t^\theta(\cdot))\mathrm{d}\mu_t \mathrm{d}t = 0$$

where ∇ is the (spatial) gradient operator[3].

An important point is that, in our case, this distributional formulation of the continuity equation gives the same solution as the ODE driven by v^θ.

4.4 A quick introduction to Optimal Transport

This section presents a few notion of Optimal Transport (OT) theory which will be an essential building block in the following sections, both in terms of intuition and of theoretical tools. In particular, we recall the dynamical formulation of OT as well as a few relatively recent regularity results.

4.4.1 From the Monge problem to Dynamical Optimal Transport

- *The static OT problem*

We state here the most important results of Optimal Transport theory and its dynamical formulation. Our main reference is [235]. [266] is another classical modern reference. The dynamical

[2]To be perfectly accurate, one should denote this operator $\nabla_x\cdot$ but there is no confusion in this context and we will keep this notation consistent throughout the text.

[3]Again, this is actually ∇_x but we choose this notation thanks to the lack of confusion and for the sake of readability.

formulation of OT has been of great importance, both theoretically and practically. It stems mainly from the work of Benamou and Brenier [33].

The principle of least action is central to many fields in physics, mathematics and economics. It is found in classical and relativistic mechanics, thermodynamics, quantum mechanics [87, 97, 103], etc.. It broadly states that the dynamical trajectory of a system between an initial and final configuration is one that makes a certain action associated with the system locally stationary [103]. One mathematical theory which can be associated with this general idea is the theory of Optimal Transport which was initially introduced as a way of finding a transportation map minimizing the cost of displacing mass from one configuration to another [235].

Formally, let α and β be absolutely continuous distributions compactly supported in $\mathbb{R}^d$, and $c : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ a cost function. Consider a transportation map $T : \mathbb{R}^d \to \mathbb{R}^d$ that satisfies $T_\sharp \alpha = \beta$, *i.e.* that pushes α to β. The total cost of the transportation then depends on all the individual contributions of costs of transporting (infinitesimal) mass from each point x to $T(x)$, and finding the optimal transportation map, if such a map exists, amounts to solving:

$$\begin{aligned} \min_{T} \quad & \mathcal{C}^{\text{stat}}(T) = \int_{\mathbb{R}^d} c(x, T(x)) \mathrm{d}\alpha(x) \\ \text{s.t.} \quad & T_\sharp \alpha = \beta \end{aligned} \tag{4.2}$$

A standard choice for c is the p-th power of a norm of $\mathbb{R}^d$, *i.e.* $c(x, y) = \|x - y\|^p$, but other costs can be used, defining different variants of the problem. This cost induces, through the p-th root of the minimal value of eq. 4.2, a distance W_p between any two distributions α and β of finite $p-$th moment, called the p-Wasserstein distance [212].

We start with the following definition:

Definition 4.2 (*Twist condition*)

c verifies the Twist condition if

$$\forall x, \frac{\partial c}{\partial x}(x, \cdot) \text{ is injective on its domain}$$

We then have the following result, proven for example in Theorem 1.17 of [235] along with Remark 1.24. of the same reference, which gives a condition on the cost under which problem eq. 4.2 has a unique minimum.

Theorem 4.1

If c verifies the Twist condition, there exists a unique T such that $\mathcal{C}(T)$ is minimal.

- *The dynamical formulation*

Instead of directly pushing samples of α to β in $\mathbb{R}^d$, we can view α and β as points in a space of measures, and consider trajectories from α to β in this space. Every curve joining those two points in measure space then defines a way to transport the probability mass from α to β.

By definition, the shortest curve joining those two points in measure space, in terms of an adequately defined metric allowing comparisons between those curves, is the *geodesic curve* between α and β. Intuitively, one would want the mapping defined by this curve to be the static OT mapping defined earlier between the two measures.

More formally, we introduce the *Wasserstein metric space* $\mathbb{W}_p(\mathbb{R}^d)$, *i.e.* the space of absolutely continuous measures of $\mathbb{R}^d$ with finite p-th moment endowed with the Wasserstein distance:

$$W_p(\mu, \nu) = \min_{T_\sharp \mu = \nu} \mathcal{C}(T)^{\frac{1}{p}}$$

where $\mathcal{C}$ is defined with the ground cost $c(x, y) = \|x - y\|^p$ and $p > 1$. We then have the following result (which summarises a few results proven for example in Chapter 5 of [235]):

Proposition 4.1

$(\mathbb{W}_p, W_p)$ is a geodesic space, meaning that, for any measures $\mu, \nu \in \mathbb{W}_p$, there exists a geodesic curve $\{\mu_t\}_{t\in[0,1]}$ between μ and ν.

Moreover, there exists a corresponding curve of vector fields $v_t \in L^p(\mu_t)$ so that $\{\mu_t\}_{t\in[0,1]}$ solves the continuity equation driven by v_t. We then have $W_p^p(\alpha,\beta) = \int_0^1 \|v_t\|^p_{L^p(\mu_t)}\mathrm{d}t$.

In other words, finding the geodesic in the adequate measure space between two measures allows to compute the OT distance for the associated cost.

It is actually possible, using the equivalence between the continuity equation and the ODE driven by the same velocity curve, to find the OT map associated with the geodesic. More specifically, considering the flow $\phi_\cdot$ corresponding to the velocities v from the continuity equation verified by the geodesic curve $\{\mu_t\}$ between α and β, the OT map between the two measures is $T = \phi_1^\cdot$ where ϕ is defined through the equation

$$\forall t, x, \frac{\mathrm{d}\phi_t^x}{\mathrm{d}t} = v_t(\phi_t^x)$$

and $\phi_0 = \mathrm{id}$. This completes the equivalence, in the case of power ground costs with $p > 1$, of the static and dynamical formulation of OT, the latter being defined as

$$\begin{aligned} \min_v \quad & \mathcal{C}^{\mathrm{dyn}}(v) = \int_0^1 \|v_t\|^p_{L^p((\phi_t^\cdot)_\sharp\alpha)}\mathrm{d}t \\ \text{s.t.} \quad & \frac{\mathrm{d}\phi_t^x}{\mathrm{d}t} = v_t(\phi_t^x), \phi_0 = \mathrm{id}, (\phi_1^\cdot)_\sharp\alpha = \beta \end{aligned} \tag{4.3}$$

which thus verifies $\mathcal{C}(T^\star) = \mathcal{C}^{\mathrm{dyn}}(v^\star)$ where $T^\star$ and $v^\star$ are the optima corresponding, respectively, to the static and dynamical formulations of the OT problem.

Intuitively, one may think about the dynamical formulation of OT in terms of fluid dynamics where the velocity field would move efficiently some blob in the fluid from one configuration to another as represented in Figure 4.1.

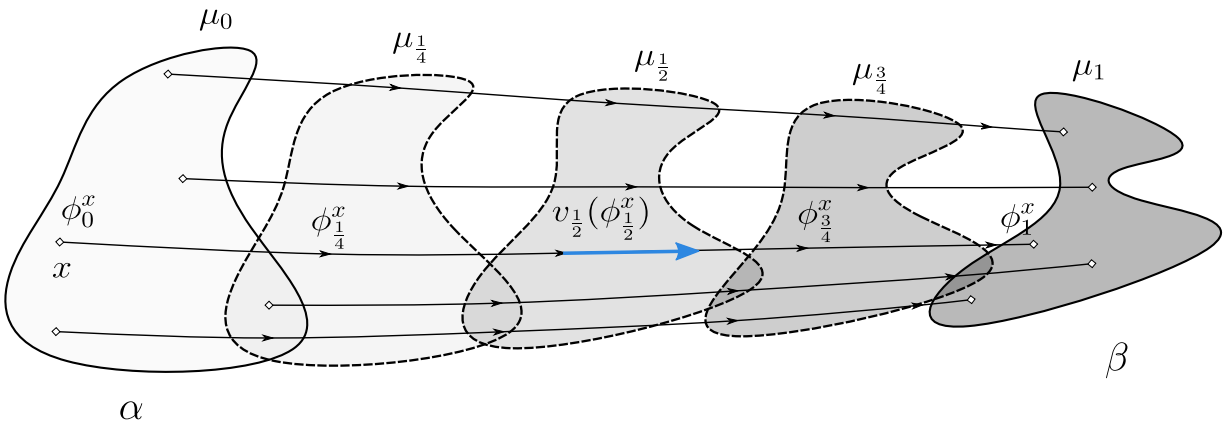


Figure 4.1: The Figure illustrates successive steps of the dynamic transportation of α to β together with the notations used in the text.

4.4.2 Regularity of Optimal Transport Mappings

In this section, we recall some classical and more recent results of regularity for Optimal Transport mappings. This is an intricate subject and the problem had been open for some time after OT theory had been established. The most important results have been established through the study of the

Monge-Ampère equation by Caffarelli then De Philippis and Figalli. Extensions for larger families of costs were developed by Ma, Trudinger and Wang [176] but this is out of the scope of this work.

In particular, Theorem 6.27 of [7] gives a classical almost-everywhere regularity result:

Theorem 4.2

If $c(x, y) = \|x - y\|^p$ for $p > 1$, and α and β have compact supports with $d(supp(\alpha), supp(\beta)) > 0$, then the Optimal Transport map T between α and β is $\alpha - a.e.$ differentiable and its Jacobian $\nabla T(x)$ has non-negative eigenvalues $\alpha - a.s.$

More recently, results summarized below, which correspond to Theorems 4.23, 4.24 and Remark 4.25 of [88], state that the OT map has one degree of regularity more than the initial transported density:

Theorem 4.3

Suppose there are X, Y, bounded open sets, such that the densities of α and β are null in their respective complements and bounded away from zero and infinity over them respectively.

Then, if Y is convex, there exists $\eta > 0$ such that the OT map T between α and β is $C^{0,\eta}$ over X.

If Y isn't convex, there exists two relatively closed sets A, B in X, Y respectively such that $T \in C^{0,\eta}(X \setminus A, Y \setminus B)$, where A and B are of null Lebesgue measure.

Moreover, if the densities are in $C^{k,\eta}$, then $C^{0,\eta}$ can be replaced by $C^{k+1,\eta}$ in the conclusions above. In particular, if the densities are smooth, then the transport map is a diffeomorphism (between the reduced input and target domains if the target support is not convex).

4.5 General Setting

This section defines the formalism in which we present our work. In particular, we propose a model decomposition of a typical DNN and and formulate the research questions our work will aim to answer, particularly in the case of classification and UDT tasks.

4.5.1 Decomposing a DNN

In order to gain a better understanding of the inner workings of a DNN, it is essential to adopt a viewpoint in which the different driving mechanisms become apparent and are decoupled.

We consider the following model of a deep neural network f where computations are separated into the three steps, *i.e.* $f = F \circ T \circ \varphi$ (this is similar to [156] and corresponds to the general structure of recent deep models or to the structure of components of a deep model [116, 277, 287]):

1. **Dimensionality change:** Starting from an input distribution $\mathcal{D}$ in $\mathbb{R}^n$, a transformation φ is applied, transforming it into $\alpha = \varphi_\sharp \mathcal{D}$, a distribution in $\mathbb{R}^d$. This corresponds to the first few layers present in most recent architectures and represents a change of dimensionality. φ is often referred to as the *encoder* or as the *encoding layers* in the litterature.

2. **Data Transport:** Then α is transformed by a mapping $T : \mathbb{R}^d \to \mathbb{R}^d$, which we see as a transport map. Here, the dimensionality doesn't change. In particular, if this part of the network is a sequence of residual blocks, T can be written as the discretized flow of an ODE.

3. **Task-specific final layers:** A final function $F : \mathbb{R}^d \to \mathcal{Y}$ is applied to $T_\sharp \alpha$ in order to compute the loss $\mathcal{L}$ associated with the task at hand, *e.g.* F could be a perceptron classifier. Like φ, F is typically made up of a few layers.

The focus of this work is on analyzing the second phase, Data Transport. Typically, we will assume that the encoder φ is pre-trained and fixed even though this assumption is relaxed in some experiments.

The core idea here is that, in order to solve a complex non-linear task for which a DNN is needed, the data has to be transformed in a non-trivial way, meaning that this is an essential phase, *e.g.* in the case of classification, $T_\sharp\alpha$ needs to be linearly separable if F is linear.

This model is quite general, as many ResNet-based architectures [277, 287] alternate modules that change the dimensionality (step 1) and transport modules that keep the dimensionality fixed (step 2) and according to [131], the transport modules have similar behaviour. The model can then be considered as a simplified ResNet, sometimes called a *single representation* ResNet. Note that [234] finds that those networks which keep the same resolution remain competitive.

As recent neural architectures have systematically achieved near-zero training error [29, 30, 127, 288], we place ourselves in this regime, which makes it possible to model a null loss as a hard constraint: in other words, we consider all trained models to reach a null loss. For some tasks, the meaning of this constraint over T is obvious: in a generative setting for example, $T_\sharp\alpha$ must be equal to some prescribed distribution β which is the target of the generation process. But in general, T is less strictly constrained and the condition depends on the final layers function F and the loss $\mathcal{L}$. This leads us to the following definition:

Definition 4.3 (*The set of admissible targets*)

The set of admissible targets for a final task-specific function F and a loss $\mathcal{L}$ is defined as

$$S_{F,\mathcal{L}} = \{\beta \in \mathcal{P}(\mathbb{R}^d) \mid \mathcal{L}(F, \beta) = 0\} \tag{4.4}$$

The idea here is that this set represents all transformed distributions $\beta = T_\sharp\alpha$ which can be reached while performing gradient descent in order to reach a null loss. In general, $\mathcal{L}$ is fixed while F is learned jointly with T. This set is supposed to be non-empty for some F and, in general, it will contain many distributions.

With this definition, the goal of the learning task can then be reformulated as:

$$\text{Find } (T, F) \text{ such that } T_\sharp\alpha \in S_{F,\mathcal{L}} \tag{4.5}$$

An important observation is that, even when $S_{F,\mathcal{L}}$ is reduced to a singleton, the problem is still strongly under-constrained and it is possible to obtain many such (T, F) that lead to poor generalization for typical parametrizations. In other words, the question which is being asked here is why is good generalization performance usually achieved even with such an under-constrained optimization problem.

The following section instantiates our formalism in our two application settings, namely classification and UDT tasks.

4.5.2 First application: UDT tasks

• *Formal definition of UDT*

Unsupervised Domain Translation (UDT) is the problem of finding, for any element a of a domain $\mathbb{A}$, its best representative b in another given domain $\mathbb{B}$. Both domains are generally provided in the form of a finite number of samples and we will model them here as absolutely continuous probability measures, respectively α and β. We will make the additional hypothesis that both domain are compact in $\mathbb{R}^d$, with regular boundaries, which is reasonable given typical data-sets.

As explained in the introduction, the CycleGAN model [295] has succeeded in achieving remarkable results for this task with many follow-up works building models along the same principles. CycleGAN-like models can then be framed as follows: Given samples from the two probability measures α and β, learn transformations T and S that map one distribution onto the other, while being each other's mutual inverse. The original paper claimed that this cycle-consistency constraint should be enough

to guarantee the meaningfulness of the model. This problem thus involves minimizing the following loss:

$$\mathcal{L}(T, S, \mathbb{A}, \mathbb{B}) = \mathcal{L}_{\text{gan}}(T, S, \mathbb{A}, \mathbb{B}) + \mathcal{L}_{\text{cyc}}(T, S, \mathbb{A}, \mathbb{B}) \tag{4.6}$$

where $\mathcal{L}_{\text{gan}}$ ensures, at optimality, that

$$T_\sharp \alpha = \beta \quad \text{and} \quad S_\sharp \beta = \alpha$$

while $\mathcal{L}_{\text{cyc}}$ ensures cycle-consistency, namely that both transformations are mutual inverses.

However, despite its popularity and empirical successes, there is no clear understanding on why CycleGAN is so effective. Indeed, as shown in [96, 280, 190], the kernel or null space of the CycleGAN loss, *i.e.* the set of couples (T, S) such that $\mathcal{L}(T, S, \mathbb{A}, \mathbb{B}) = 0$, is not reduced to a singleton except in trivial cases and is often infinite in most cases of interest. By studying the kernel of the loss, [190] show more precisely that elements of the null space as well as solutions obtained through the extended version of the loss, where the loss is regularized so that the transformations are close to the identity function, can lead to arbitrarily undesirable solutions of UDT. Thus, there is a discrepancy between what the loss of CycleGAN-like models captures and their practical usefulness.

Intuitively, it is easy to see how such a loss falls short from defining a well-posed problem: one can choose any invertible mapping T which pushes α into β, of which there exists an infinity in general, and (T, T^{-1}) would be a null loss solution. Among those possible null loss mappings, most would be meaningless and the question is then how to characterize meaningful one and, most importantly, how are those selected by CycleGAN-like models in practice.

Here, if we can consider that only a transport mapping T is learned, the corresponding set of admissible targets which it induces reduces to a singleton containing the target distribution β.

- *What should be the properties of a UDT solution?*

Qualitatively, good solutions of a UDT problem are the ones which translate an input a from $\mathbb{A}$ to $\mathbb{B}$ while still conserving as much as possible the characteristics of a, and conversely from $\mathbb{B}$ to $\mathbb{A}$. The CycleGAN seminal paper tries to enforce this through the cycle-consistency loss but, as discussed above and in previous papers on the subject, this loss is null for any invertible mapping T by taking the couple (T, T^{-1}), without necessarily conserving any characteristics across domains. In other words, this loss doesn't really add any constraints on the mapping and infinitely many undesirable can still be theoretically recovered by the model.

This intuition has already been formulated in [96] in the notion of "semantics preserving mappings". The authors, recognizing that preserving semantics is a vague notion, propose to measure it through the minimal number of layers necessary for neural networks to represent the transformation. However, while we think that it provides a useful step forward in understanding UDT, such a formulation has several shortcomings: there is no reason why complexity should always be measured as the number of layers of a non-residual NN [289] and it is not even clear whether such a minimal number is always finite for relevant transformations. Moreover, this notion doesn't provide theoretical insights on how and why CycleGAN performs so well in practice or why it seems to work well even with very deep networks. Crucially, there are no guarantees regarding the uniqueness of the so-called minimal complexity mappings.

It is also interesting to consider the extended loss for CycleGAN introduced in the original paper [295] as a regularization forcing T and S to be close from the identity mapping. While, as shown theoretically and empirically in [190], adding this regularization doesn't prevent undesirable mappings to be reached by the model, the fact that it was necessary to further constrain the objective for certain tasks in this way shows that it can be helpful to have transformations which do not transform inputs too much. This is coherent with the view of [96]. We aim to extend the reasoning of both approaches in a more adaptive, robust and theoretically grounded formalism.

Generalizing those discussions, in our view, there are two main important desirable features in UDT models as used in many practical settings:

- The mapping T (and, symmetrically S) should be constrained to be as conservative as possible, in the sense that they should be as close to the identity as possible.

- The mappings T and S should also be regular. Indeed, in the case of image-to-image translation from paintings to photographs for example, if we take two paintings a, a' representing nearly the same scene then we would want the corresponding photos $T(a), T(a')$ to be similar as well. This property would mean that T and S are endowed with some functional regularity, at least a form of continuity.

While the first feature extends the points of view already discussed in previous works, the second one is novel, up to our knowledge. It seems difficult to enforce directly the regularity of the estimated mappings but we show in the following that our approach seamlessly satisfies both properties.

4.5.3 Second application: classification tasks

Our second application setting is that, more familiar of supervised classification which we formalize in the following way. Consider N classes. For each class i, we have a training data distribution α_i. We assume that those classes have mutually disjoint supports, which means in practice that there is no ambiguity in the class of data points. We represent the full data distribution as $\alpha = \frac{1}{N}\sum_i \alpha_i \times \delta_i$ where δ_i is the dirac distribution concentrated at i thus indicating the label of α_i. For obvious technical reasons, we slightly adapt our notations here by writing $T_\sharp\alpha = \frac{1}{N}\sum_i (T_\sharp\alpha_i) \times \delta_i$ as the labels are not changed along with the data dsitributions.

In the case of UDT, the set of target distributions was easy to define as it was just the distribution associated with the opposite domain and is implied by the definition of the task. However, in the case of classification, this set becomes more difficult to characterize and must, instead, be defined implicitly for a given choice of final loss and classifying layers of the DNN.

More precisely, one wants to find a transformation T of these distributions α_i such that all transported distributions can be correctly classified by a classifier F which is often learned jointly with transport function. For example, if the classifier F is chosen among linear ones, $S_{F,\mathcal{L}}$ is the set of distributions which have N components that are linearly separated by F (while being coherent with the labelling). Note that we place ourselves in a noiseless ideal setting where perfect classification is possible.

Note that we are taking here the problem from an *a posteriori* point of view by considering how (T, F) acts on the underlying infinite data distribution which is that α is transported into an admissible distribution in $S_{F,\mathcal{L}}$. However, the usual setting is to have a finite number of training samples from α and trying to solve the task with those while hoping to generalize to unseen data points. So the learning problem is even more under-constrained than in the case of UDT as both the transport and the classification functions are learned jointly, with each classification function defining an infinite number of admissible targets. The goal here at this point of our analysis is then similar to the UDT case, namely to characterize the properties of "good" mappings T able to generalize well and how they relate to actually trained DNNs.

The question we examine in the following, for both tasks, is then twofold:

- What are the properties characterizing mappings reached by standard residual architectures with common hyper-parameters?

- Can we find a criteria to *automatically select* mappings with desirable properties in order to improve performance and robustness?

4.6 Empirical Analysis of Transport Dynamics in ResNets

In this section, we conduct an empirical analysis, both for low and high dimensional datasets. The first ones allow to visualize how data is actually transported while the second ones are closer to practice.

4.6.1 CycleGAN is Biased Towards Low Energy Transformations

In practice, the success of CycleGAN models must be made possible by the presence of inductive biases that constrain the set of solutions and that are imposed through the combination of the choices made for SGD-based methods, networks architectures, weight parameterization and initialization. In order to develop a better understanding and identify implicit biases, we have conducted an exploratory analysis to characterize the influence of CycleGAN hyperparameters. Our main finding is that the initialization gain σ, *i.e.* the standard deviation of the weights of the residual network (along with a fixed small learning rate), has the most substantial and consistent impact, among all the hyperparameters, on the retrieved mappings. These findings are illustrated in the following experiments.

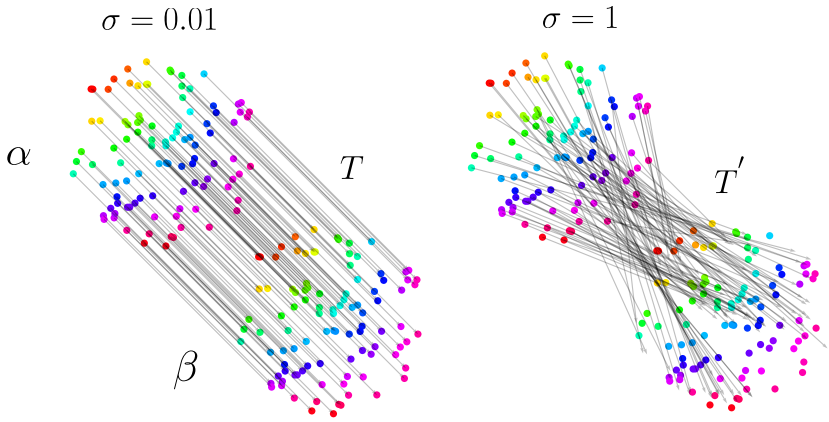


Figure 4.2: Pairings between domains obtained with CycleGAN. Both domains correspond to uniform distributions on a 2d-sphere with shifted centers. Small initialization values lead to simple and ordered mappings (Left), whereas larger ones yield complex and disordered ones (Right). Colors highlight original pairing between domains, before shifting.

2D Toy Example Figure 4.2 shows the effect of changing the gain from a small value, $\sigma = 0.01$, to a higher one, $\sigma = 1$ when learning to map one circular distribution to another. This changes the obtained mapping from a simple translation aligning the two distributions with a minimum displacement to a more disorderly one. In other words, it seems that higher initialization gains lead to higher energy mappings. Further quantifying the effect of initialization gain on the retrieved mappings, we use a natural characterisation of *disorder / complexity* of a mapping: the average distance between a sample x from α and its image $T(x)$. Using the squared Euclidean distance, this corresponds to the *kinetic energy* of the displacement and can be written as $\int_{\mathbb{R}^d} \|x - T(x)\|_2^2 \, \mathrm{d}\alpha(x)$. This quantity is also the *quadratic transport cost* used in Optimal Transport [235]. Using the mapping minimizing this cost as a reference, we see, on the left of Figure 4.3, that the larger σ becomes, the further CycleGAN's mapping (blue curve) is from it. The right curve confirms this finding: As σ grows, so does the transport cost of the trained CycleGAN. For both experiments, the variance across runs increases i.e. the model yields very different mappings across runs, corroborating the ill-posed nature of CycleGAN's optimization problem.

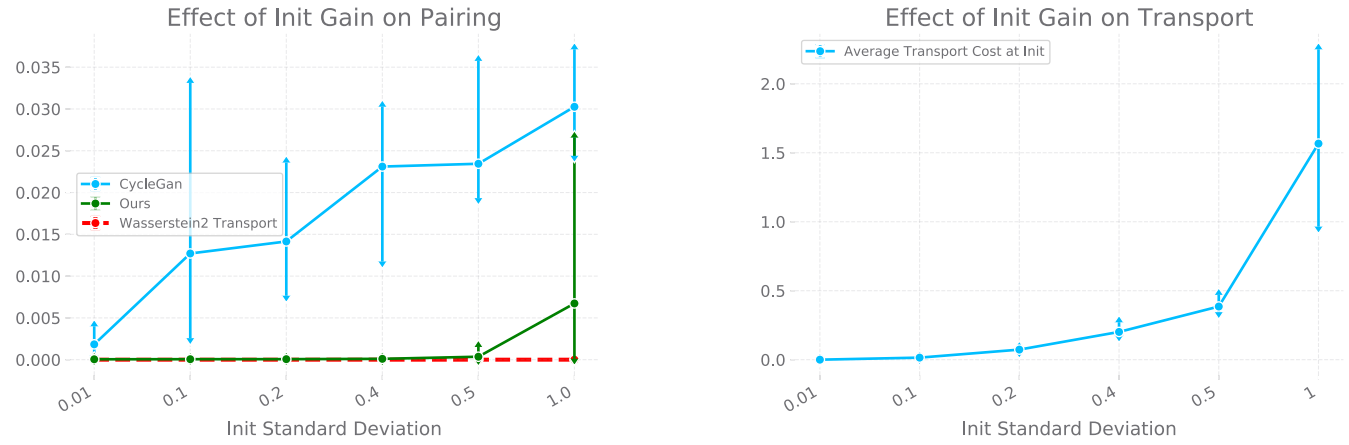


Figure 4.3: Left: L^2 distance to the Optimal Transport mapping "*Wasserstein 2 Transport*" as a function of the initialization gain (domains are illustrated in Figure 4.2). "Ours" refers to the model presented subsequently. Right: Transport cost of the CycleGAN mapping as a function of initialization gain. Metrics are averaged across 5 runs, and the standard deviation is plotted.

High-Dimensional Analysis We also conducted a similar analysis with high-dimensional distributions of images on the CelebA dataset. While in this case calculating the exact OT map is intractable, we can visualize samples obtained with the CycleGAN mapping for different values of σ. The task is male to female transformation where one wants to keep as many characteristics as possible from the original image in the generated one. The result in Figure 4.9 confirms the low-dimensional findings: while in all cases the distributions have been successfully aligned, as all males are transformed into females, the mappings initialized with low σ values perform a minimal transformation of the input while high σ values produce unwanted changes in the features (hair color, face, skin color,...). This is corroborated by measuring the transportation cost incurred by CycleGAN which goes from 0.15 for $\sigma = 0.01$ to 9.7 for $\sigma = 1.5$, showing that this behaviour is linked with high transport costs.

In summary, for common UDT tasks where the input is to be preserved as much as possible, successful CycleGAN models tend to consistently converge to low energetic mappings and this bias is induced by a small initialization gain. However, the CycleGAN model doesn't give any explicit control over this bias, thus warranting a blind hyper-parameter / architecture search for each new task.

Moreover, considering again the experiment in Figure 4.2, the "disordered" mapping could actually be the desirable one in some application where the practitioner doesn't necessarily wish to preserve the input but has another criterion in mind. The problem of not having explicit control over which mapping is retrieved then becomes even more glaring.

In the following, we will show how our better constrained formulation allows to circumvent those issues, thus defining a class of explicitly controllable models with theoretical guarantees.

4.6.2 Classification

We now conduct a similar exploratory analysis of the impact of the network's inner dynamics on generalization in the case of classification. We present below two experiments. The first one highlights how good generalization performance is closely related to low transport cost for classification tasks on MNIST and CIFAR10. This cost therefore appears as a natural characterization of the complexity and disorder of a network. The second experiment, performed on a toy 2D dataset, visualizes the transport induced by the blocks of a ResNet.

We consider ResNets where, after encoding, a data point x_0 is transported by applying $x_{k+1} = x_k + v_k(x_k)$ for K residual blocks and then classified using F. We measure the disorder/complexity

of a network by its transport cost which is the sum of the displacements induced by its residual blocks: $\mathcal{C}(v) = \sum_k \|v_k(x_k)\|_2^2$. This quantity roughly corresponds to the kinetic energy of the total displacement.

Transportation cost and generalization on MNIST and CIFAR10. In order to study the correlation between the transport cost of a residual network and its generalization ability on image data, we train convolutional 9-block ResNets with different initializations (orthogonal and normal with different gains), for 10-class classification tasks MNIST and CIFAR10. In Figure 4.4, each point represents a trained network and gives the transport cost $\mathcal{C}$ as a function of the test accuracy of the network. This experiment clearly highlights the strong negative correlation between transport cost and good generalization. This illustrates the importance of the implicit initialization bias and motivates initialization schemes which favour a low kinetic energy.

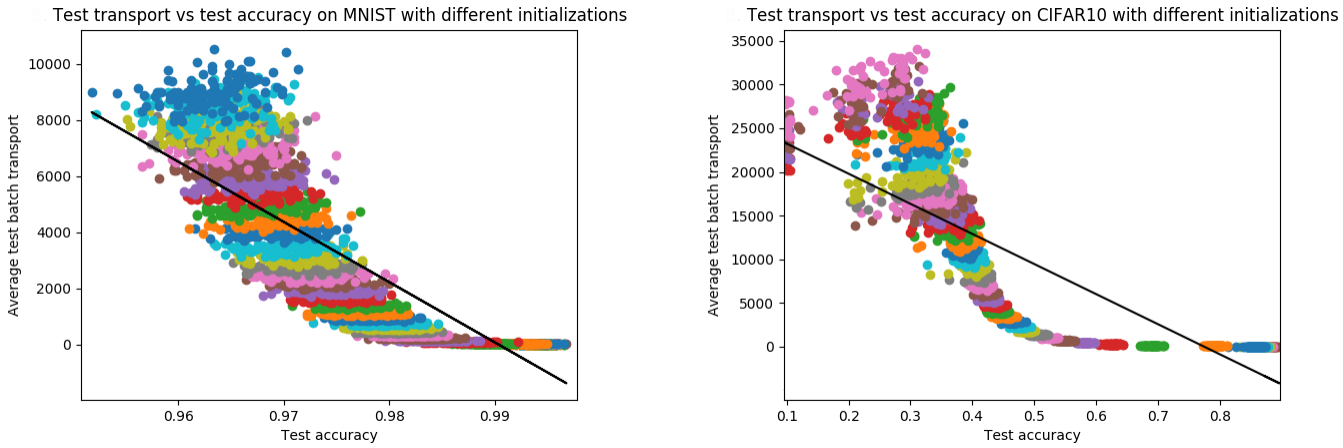


Figure 4.4: Test transport against test accuracy of ResNet9 models on MNIST (left) and CIFAR10 (right) with fitted linear regressions, where each color indicates a different initialization (either orthogonal or normal with varying gains)

Visualizing network dynamics on 2D toy data. This experiment provides a 2D visualization of the transport dynamics inside a network. The task is 2-class classification of a non-linearly separable dataset (two concentric circles, from `sklearn`) which contains 1000 points with a train-test split of 80%-20%, see Figure 4.5, top left. The network is a ResNet containing 9 residual blocks, followed by a fixed linear classifier. Each residual block contains two fully connected layers separated by a batch normalization and a ReLU activation.

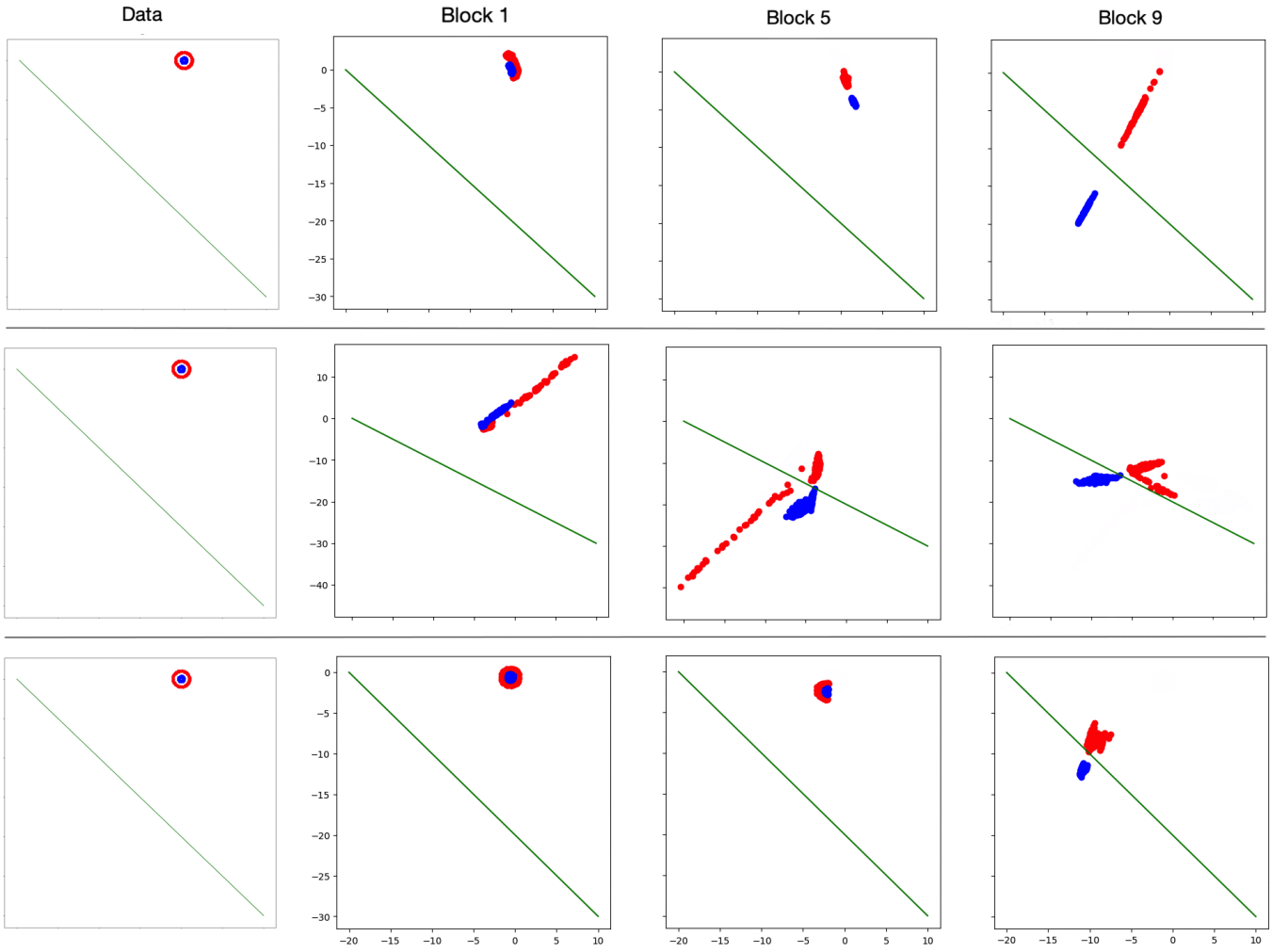


Figure 4.5: Transformed circles test set by a ResNet9 after blocks 1, 5 and 9 after training; first row with good initialization; second row with a $\mathcal{N}(0, 5)$ initialization; third row with a $\mathcal{N}(0, 5)$ initialization and the transport cost added to the loss

With the cross-entropy loss alone, the behaviour of a well trained and carefully initialized network achieving 100% test accuracy is illustrated in the first row of Figure 4.5. With a $\mathcal{N}(0, 5)$ initialization, significantly bigger than the usually small initialization gains, the test accuracy drops to 98% (average of 100 runs) and the transport becomes more chaotic (Figure 4.5, second row). Adding the transport cost to the loss improves the test accuracy (99.7% on average) of this badly initialized network and the movement becomes qualitatively smoother (third row of Figure 4.5). Thus, controlling transport cost seems to improve the behavior and generalization ability of the network. This allows to explicitly control the network whereas implicit biases such as "good" initialization rely on heuristics and prior knowledge of how to correctly set hyper-parameters.

4.7 A Least Action Principle for Training Neural Networks

The previous section has shed some light on the low energy bias of networks as well as on its potential benefits on a model's performance. In this section, we take a step further and make this implicit bias explicit by reformulating the optimization problem defining the DNN so that a predetermined energy is minimized, thus taking inspiration from the problem of Optimal Transport. This allows us to prove the existence of minimizers, uniqueness in some settings and exhibit interesting regularity properties of the minimal energy neural networks which may explain the properties of successfully

trained DNNs such as good generalization performance and other desirable qualitative properties.

Let us emphasize that the formulation we propose here serves two purposes:

- on the one hand, it allows us to test our hypothesis regarding the role of the low energy bias we observed in the previous section, by evaluating the behaviour of models where this bias is explicitly enforced;

- on the other hand, this formulation can be useful in itself as the two applications in the following sections will show by allowing to construct models with improved performance, robustness and flexibility.

4.7.1 Intuition

Let us look again at how a ResNet acts on data, focusing on the transport phase. Taking an initial distribution α considered as a point cloud, it is gradually transformed through the residual blocks in order to reach a certain target distribution β. Obviously, and as seen previously, such a target distribution is not necessarily uniquely defined. Indeed, depending on which task is considered, many such targets could lead to perfectly solving the task at hand. This is what led us to consider the set of admissible targets in Definition 4.3.

However, assuming that such a target is determined, the NN is basically computing a transport map from α to β. Now, if we look at the form of the residual updates

$$x_{k+1} = x_k + v_k(x_k)$$

and consider that small initialization gains with relatively small learning rates are often used to achieve successful training, this translates into small values of the norms $\|v_k(x_k)\|_2^2$, which means that the overall discrete quadratic transport cost $\sum_k \|v_k(x_k)\|_2^2$ is small at initialization and tends to stay so throughout training as many tricks are used to stabilize the training procedure and otherwise there might be problems of gradients diverging.

Looking at this from the continuous point of view, this means that we are transporting α to β along the continuity equation $\frac{\mathrm{d}\mu_t}{\mathrm{d}t} + \nabla \cdot (\mu_t v_t) = 0$ while keeping a small transport cost $\int_0^1 \|v_t\|_{L^p(\mu_t)}^p \,\mathrm{d}t$. In other words, the NN is approximately solving the dynamical formulation of the OT problem from the initial distribution α to an admissible target distribution β which allows to solve the task at hand.

Obviously, the reasoning presented here only has heuristic value as it greatly simplifies the process. Our aim, in the formulation which we give in the following part of this section, is to follow along the lines of this intuition by enforcing the minimization of certain cost explicitly. We will show that this is actually beneficial, both in terms of guaranteeing theoretical properties and in terms of performance.

4.7.2 Formulation

We consider costs $c(x, y) = \|x - y\|^p$ (where $\|.\|$ is a norm of $\mathbb{R}^d$), with $p > 1$, and suppose that $\alpha \in \mathcal{P}_p(\mathbb{R}^d)$ (the set of absolutely continuous measures on $\mathbb{R}^d$ with finite p-th moment). We assume that the space of classifiers is compact, that the loss $\mathcal{L}$ is continuous, that the set $\cup_{F\in\mathcal{F}} S_{F,\mathcal{L}}$ is at a finite p-Wasserstein distance W_p from α (in particular, it is non-empty) and that all its bounded subsets are totally bounded (*i.e.* can be covered by finitely many subsets of any fixed size).

These properties depend on the choice of the loss $\mathcal{L}$ and of a class of functions $\mathcal{F}$ for the classifier F. In particular, they are verified in practice for most UDT and classification instances.

In coherence with our intuition, a natural way to select a robust model, given the empirical observations of Section 4.6, is to select, among the maps which transport α to a point in $S_{F,\mathcal{L}}$ and thus allow to solve the task, one with a minimal transportation cost. This gives us the following

optimization problem:

$$\begin{aligned} \inf_{T,F} \quad & \mathcal{C}(T) = \int_{\mathbb{R}^d} c(x, T(x)) \mathrm{d}\alpha(x) \\ \text{subject to} \quad & T_\sharp \alpha \in S_{F,\mathcal{L}} \end{aligned} \tag{4.7}$$

The equivalent dynamical version for $c(x, y) = \|x - y\|^p$, which is useful especially for residual networks and will be used in the coming experiments is then

$$\begin{aligned} \inf_{v,F} \quad & \int_0^1 \|v_t\|^p_{L^p((\phi_t^\cdot)_\sharp \alpha)} \, \mathrm{d}t \\ \text{subject to} \quad & \frac{\partial \phi_t^x}{\partial t} = v_t(\phi_t^x) \\ & \phi_0^\cdot = \mathrm{id} \\ & (\phi_1^\cdot)_\sharp \alpha \in S_{F,\mathcal{L}} \end{aligned} \tag{4.8}$$

where $\|v_t\|^p_{L^p((\phi_t^\cdot)_\sharp \alpha)} = \int_{\mathbb{R}^d} \|v_t\|^p \mathrm{d}(\phi_t^\cdot)_\sharp \alpha$ and ϕ is the flow of the ODE driven by v_t as defined in Section 4.3.2.

The result below shows that these two problems are equivalent and that the infima are realized as minima:

Theorem 4.4

The infima of eq. 4.7 and eq. 4.8 are finite and are realized through a map T which is (or a velocity field v which induces) an optimal transportation map. When $c(x, y) = \|x - y\|^p$, then eq. 4.7 and eq. 4.8 are equivalent.

Proof: From the hypothesis above, there exists $\beta \in S_{F,\mathcal{L}}$ at a finite distance from α. Taking any transport map between α and β, we see that the infima are finite.

Consider eq. 4.7 and take a minimizing sequence $(T_i, F_i)_i$. Set $\beta_i = (T_i)_\sharp \alpha$. Then $(\mathcal{C}(T_i))_i$ converges to the infimum which is strictly bounded by $M > 0$. Then, by definition, for i large enough, $W_p^p(\alpha, \beta_i) \leq \mathcal{C}(T_i) \leq M$. So that $(\beta_i)_i$ is a bounded sequence in $\cup_F S_{F,\mathcal{L}}$. By the hypothesized total boundedness of bounded subsets and as $\mathcal{P}_p(\mathbb{R}^d)$ endowed with W_p is a complete metric space (see [41] for a proof), up to an extraction, $(\beta_i)_i$ converges to $\beta^\star$ in the closure of $\cup_F S_{F,\mathcal{L}}$. Moreover, up to an extraction, $(F_i)_i$ also converges to $F^\star$ by compactness of the class of classifiers. Taking $T^\star$ the OT map between α and $\beta^\star$, we then have, by continuity of $\mathcal{L}$,

$$T^\star_\sharp \alpha = \beta^\star \in S_{F^\star,\mathcal{L}}$$

and $\mathcal{C}(T^\star) \leq \lim \mathcal{C}(T_i)$ by optimality of $T^\star$, which means, since $(\mathcal{C}(T_i))_i$ is a minimizing sequence, that $\mathcal{C}(T^\star)$ minimizes eq. 4.7. So $(T^\star, F^\star)$ is a minimizer and $T^\star$ is an OT map.

Finally, there exists, by dynamical OT theory, a velocity field $v_t^\star$ inducing the OT map between α and $\beta^\star$ which then gives a minimizer $(v^\star, F^\star)$ for eq. 4.8. By the same reasoning, taking a minimizing sequence $(v^{(i)}, F_i)_i$ and the induced maps T_i shows that both problems are equivalent. □

Note that uniqueness doesn't hold in the general case contrary to standard OT problem, as the constraint $T_\sharp \alpha \in S_{F,\mathcal{L}}$ in eq. 4.7 and eq. 4.8 is looser than in the latter setting because $S_{F,\mathcal{L}}$, which is task-dependent, is not necessarily a singleton, e.g. in the case of classification. However, as we show in the following section, the optimization problems are solved by OT maps will give regularity properties for the models induced by these optimization problems.

Moreover, in the particular case where $S_{F,\mathcal{L}}$ is reduced to a singleton, which is the case for UDT and in general for generative tasks, uniqueness does hold:

Proposition 4.2

If $S_{F,\mathcal{L}}$ is a singleton, eq. 4.7 and eq. 4.8 reduce to the standard OT and dynamical OT problems.

This means in particular that the minimal transport map is unique.

For UDT, this property is essential as it solves the indeterminacy issue of CycleGAN-like models which was mentioned in Section 4.5.2.

Finally, note that, by considering eq. 4.7, we can go for costs c other than power ones.

4.7.3 Regularity

Intuitively, the fact that we minimize the energy of the transport map transforming the data is akin to the core idea of Occam's razor: among all the possible networks that correctly solve the the task, the one transforming the data in the simplest way is selected. We show here that this actually endows maps solving eq. 4.7 and eq. 4.8 with regularity. More precisely, our formulation yields mappings which are as regular as the initial data distribution allows for, and thus provides an alternate view on generalization for modern deep learning architectures in the overparametrized regime.

Indeed, OT maps can be as irregular as needed in order to fit the target distribution, however in much the same way as for successfully trained DNNs, those optimal maps are still surprisingly regular. In a way, they are as regular as possible given the constraints which is exactly the type of flexibility needed in modern ML tasks. However, the constraints in eq. 4.7 and eq. 4.8 are looser than in the standard definitions of Optimal Transport. Still, supposing that the input data distribution has a nicely behaved density, namely bounded and of compact support, with the same hypothesis as above, we have the following, which is mainly a corollary of Theorem 4.4:

Proposition 4.3

Consider $T^\star$ the OT map induced by eq. 4.7 (or eq. 4.8) given by Theorem 4.4. Take X, respectively Y, an open neighborhood of the support of α, respectively of $T^\star_\sharp\alpha$, then $T^\star$ is differentiable, except on a set of null α measure.

Additionally, if $T^\star$ doesn't have singularities, there exists $\eta > 0$ and A, respectively B, relatively closed in X, respectively Y, such that $T^\star$ is η-Hölder continuous from $X \setminus A$ to $Y \setminus B$. Moreover, if the two distributions, initial and target, have smooth densities, $T^\star$ is a diffeomorphism from $X \setminus A$ to $Y \setminus B$.

Proof: This is a consequence of Theorem 4.4, the hypothesis made in this section and the regularity theorems stated in Section 4.4.2. □

There are two main results in Proposition 4.3: the first gives α-$a.e.$ differentiability. This is already as strong as might be expected from a classifier: there are necessarily discontinuities at the frontiers between different classes. The second is even more interesting: it gives Hölder continuity over as large a domain as possible, and even a diffeomorphism if the data distribution is well-behaved enough. We recall that a function f is η-Hölder continuous for $\eta \in]0, 1]$ if $\exists\ M > 0$ such that $\|f(x) - f(y)\| \leq M\|x - y\|^\eta$ for all x, y. η measures the smoothness of f, the higher its value, the more regular the function. In particular, in the case of classification, this means that the Hausdorff dimension, which is a standard generalization of dimensionality for non-smooth spaces, along the frontiers between the different classes is scaled by less than a factor of $1/\eta$ in the transported domain. If the densities are smooth, the dimension thus even becomes provably smaller by this result.

Intuitively, this means that, in these models, the data is transported in a way that preserves and simplifies the patterns in the input distribution. This notion of regularity is exactly the one that one wants as the regularity of the mappings has to be linked to that of their underlying domains.

Moreover, the fact that regularity excludes a negligible set of points of the domains is also coherent with what we should expect: for UDT for example, in the two domains, there can be points which are close but nevertheless represent elements from different classes and thus should be transported far from each other. For example, in image-to-image translation between photographs and paintings, two

images with the same background can represent different objects and thus be translated into very different paintings. Thus, this regularity result supports our claim for the transport cost to be the right measure of "complexity" for UDT mappings.

In the following, we propose a practical algorithm implementing these models so that we can study it empirically for UDT and classification tasks.

4.7.4 General Algorithm

We propose an algorithm for training models using the proposed least action principle formulation by minimizing a transport cost.

We will tackle the case of the dynamical formulation, thus with ground costs of the form $c(x, y) = \|x - y\|^p$, which was the form used in almost all of our experiments in the following sections. The algorithm for more general costs can be easily deduced both in cases where the cost also admits a dynamical equivalent and in others where it doesn't.

We start by considering eq. 4.8 and taking a neural parametrization v^θ, F^θ of the velocity fields and of the task-specific decision function. Not that θ covers the parameters of both here so that we can write that

$$((\phi^\theta)_1^\cdot)_\sharp \alpha \in S_{F,\mathcal{L}} \Leftrightarrow \mathcal{L}(\theta) := \mathcal{L}(F^\theta, ((\phi^\theta)_1^\cdot)_\sharp \alpha) = 0$$

where ϕ^θ is the flow associated with v^θ.

We can now write the optimization problem as the following

$$\begin{aligned} \min_\theta \quad & \mathcal{C}(\theta) = \int_0^1 \|v_t^\theta\|^p_{L^p(((\phi^\theta)_t^\cdot)_\sharp \alpha)} \, \mathrm{d}t \\ \text{s.t.} \quad & \frac{\partial (\phi^\theta)_t^x}{\partial t} = v_t((\phi^\theta)_t^x), \\ & \forall x, (\phi^\theta)_0^x = x, \\ & \mathcal{L}(\theta) = 0 \end{aligned} \tag{4.9}$$

which is very similar to the general optimization problem presented in Chapter 1. However, there is the additional constraint $\mathcal{L}(\theta) = 0$ which has to be enforced.

In order to do so and be able to write this optimization problem in a form for which gradient descent is possible, we actually consider a sequence of optimization problems $\mathcal{C}(\theta_i) + \lambda_i \ \mathcal{L}(\theta_i)$ in the same way as in Chapter 3 which should converge to an approximate solution where the null loss constraint is enforced with a chosen tolerance. The updates are conducted for (θ_i, λ_i) in the following way:

$$\begin{cases} \theta_{i+1} = \arg\min_\theta \ \mathcal{C}(\theta) + \lambda_i \ \mathcal{L}(\theta) \\ \lambda_{i+1} = \lambda_i + \tau \ \mathcal{L}(\theta_{i+1}) \end{cases} \tag{4.10}$$

The minimization in the first update is done via SGD with a gradient computed as in Chapter 1 for a number of steps s, where a step means a batch, starting from the previous parameter value θ_i. Moreover, in practice, we have found that it is more stable to divide the minimization objective in

eq. 4.10 by λ_i, yielding Algorithm 3.

Algorithm 3: Training neural networks with Least Action Principle (LAP-Net)

Input: Training samples, step size τ, number of steps s, initial weight λ_0
Initialization: Initialize the parameters θ_0 and set $i = 0$
while *not converged* **do**
1. Starting from θ_i, perform s steps of stochastic gradient descent:
   1.1. $\theta^0_{i+1} = \theta_i$
   1.2. $\theta^l_{i+1} = \theta^{l-1}_{i+1} - \epsilon\nabla_\theta(\mathcal{C}(\theta^{l-1}_{i+1})/\lambda_i + \mathcal{L}(\theta^{l-1}_{i+1}))$ for l from 1 to s
   1.3. $\theta_{i+1} = \theta^s_{i+1}$
2. Update the weight $\lambda_{i+1} = \lambda_i + \tau\ \mathcal{L}(\theta_{i+1})$ and increment $i \leftarrow i + 1$

Output: Learned parameters θ

While the high non-convexity makes it difficult to ensure exact optimality, we can still have some induced regularity when reaching a "good" local minimum:

Proposition 4.4

Suppose $(F^{\theta^\star}, T^{\theta^\star})$ is reached by the optimization algorithm such that $T^{\theta^\star}$ is an $\epsilon-$OT map between α and its push-forward[4]. Then we have, with the same notations as in Proposition 4.3,

$$\forall x, y \in X \setminus A, \|T^{\theta^\star}(x) - T^{\theta^\star}(y)\| \leq O(\epsilon + \|x - y\|^\eta)$$

Proof: We simply write the decomposition:

$$T^{\theta^\star}(x) - T^{\theta^\star}(y) = T^{\theta^\star}(x) - T^\star(x) + T^\star(x) - T^\star(y) + T^\star(y) - T^{\theta^\star}(y)$$

and use the triangular inequality: the first and third terms are smaller than ϵ by hypothesis while Hölder continuity applies for the second by Proposition 4.3. □

This shows that minimizing the transport cost still endows the model with some regularity, even in situations where the global minimum is not reached and the null loss contraint is not verified.

4.8 Application to UDT

We look here at how our LAP formulation tackles the UDT task. Let us recall that we have already showed in the previous section that this formulation provides uniqueness of the map between the domains in the UDT setting. Here, we show that the dynamical formulation also provides the inverse of this unique map without additional training or costly computation. Finally, we evaluate the corresponding model with two variants of the UDT task.

4.8.1 Instanciation of the algorithm

- *Computing the Inverse*

Consider the optimal vector field $v^\star$ of eq. 4.8 associated to the OT map $T^\star$ from α to β and the following system of differential equations, for all $x \in \mathbb{B}$:

$$\begin{cases} \frac{\partial \psi^x_t}{\partial t} = -v^\star_t(\psi^x_t) \\ \psi^x_0 = x \end{cases} \tag{4.11}$$

Then we have the following:

Proposition 4.5

The solution curve $\{\psi_t^\cdot\}_t$ of eq. 4.11 induces a trajectory of measures $\{(\psi_t^\cdot)_\sharp\beta\}$ which geodesically interpolates between β and α. In particular, $S^\star = \psi_1^\cdot$ is the inverse of $T^\star$, verifies $S^\star_\sharp\beta = \alpha$ and is the OT map between β and α.

Proof: Let us consider $\nu_t = (\psi_t^\cdot)_\sharp\beta$. Then $(\nu_t, -v_t)$ solves the continuity equation. On the other hand, taking previous notations, we have that, for all test functions f:

$$\int \left(\frac{\partial f}{\partial t}(t,\cdot) + \nabla f(t,\cdot)\cdot v_t^\star(\cdot) \right) \mathrm{d}\mu_t \mathrm{d}t = 0$$

which becomes

$$\int \left(\frac{\partial f}{\partial t}(1-t,\cdot) + \nabla f(1-t,\cdot)\cdot v_{1-t}^\star(\cdot) \right) \mathrm{d}\mu_{1-t} \mathrm{d}t = 0$$

and then, with the change of variables $t' = 1-t$:

$$\int \left(-\frac{\partial f}{\partial t'}(t',\cdot) + \nabla f(t',\cdot)\cdot v_{t'}^\star(\cdot) \right) \mathrm{d}\mu_{1-t} \mathrm{d}t' = 0$$

which can also be written as

$$\int \left(\frac{\partial f}{\partial t'}(t',\cdot) + \nabla f(t',\cdot)\cdot (-v_{t'}^\star(\cdot)) \right) \mathrm{d}\mu_{1-t} \mathrm{d}t' = 0$$

This means that $(\mu_{1-t}, -v_{1-t})$ solves the continuity equation with the same initial condition as ν_t which gives us $\nu_t = \mu_{1-t}$ for all t. The distance induced by ν_t is then the same as the one induced by v_t which is the same as the one induced by $T^\star$. This means that $S^\star$ is indeed the OT map between β and α by symmetry of the Wasserstein metric which finishes the proof. □

This result shows that $(T^\star, S^\star)$ does indeed solve the UDT problem and is in the null space of the CycleGAN loss. Moreover, in order to compute $S^\star$, there is no need to parametrize it nor to solve a difficult optimization problem. It is only necessary to discretize the associated differential equation which is of the same nature as the one for the forward mapping, meaning that the same scheme can be used.

• *Practical details*

In CycleGAN, the first boundary condition $\phi_0^\cdot = \mathrm{id}$ is satisfied by construction, while the second $(\phi_1^\cdot)_\sharp\alpha = \beta$ is enforced with the GAN loss.

Thus we recover CycleGAN as a particular implementation of this model when there is no transport cost minimization. We actually construct our instantiation in a similar fashion in order to have meaningful comparisons: The differential equations are discretized using an Euler scheme and boundary conditions are enforced using an iterative penalization of the GAN loss. More involved schemes can be used here such as any suitable parametrized solver [50].

Discretization and enforcing boundary conditions We implement the forward map with a Euler discretization[5] of the dynamical formulation thus giving us the following fully discretized optimization problem:

$$\begin{aligned} \min_\theta \quad & \mathcal{C}_d(\theta) = \sum_{k=1}^{K} \sum_{x\in \mathrm{Data}_\alpha} \|v^{\theta_k}(\phi_k^x)\|_p^p \\ \text{s.t.} \quad & \forall x,\ \forall k,\ \phi_{k+1}^x = \phi_k^x + \Delta t\, v^{\theta_k}(\phi_k^x), \\ & \phi_0^\cdot = \mathrm{id},\ \ (\phi_1^\cdot)_\sharp\alpha = \beta \end{aligned} \tag{4.12}$$

[5]Other schemes could be used, which would lead to other architectures, and could arguably be more suited for stability reasons but this is beyond the scope of this work.

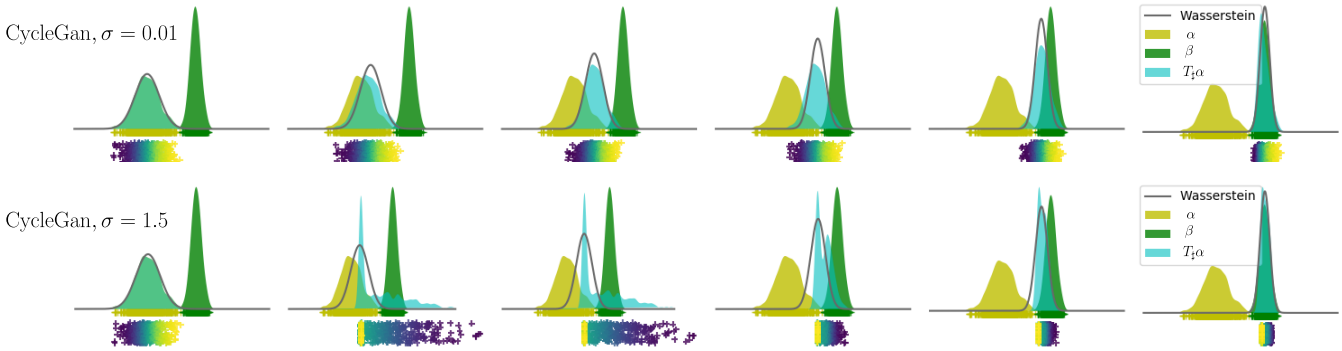


Figure 4.6: Visualization of the hidden layers of CycleGAN when mapping the yellow gaussian distribution to the green one with different initializations. The points below the histograms are colored when initially sampled thus allowing to see the trajectories of individual sampels. Thus, when σ increases, the mapping goes from a simple translation (Top) to a more complicated mapping (Bottom), thus inducing an increase in transport cost.

In other words, this corresponds to a slightly simplified version of the original vanilla CycleGAN implementation with an added transport cost minimization. Figure 4.6 illustrates the already observed fact of low initialization gains CycleGAN being close to dynamical OT maps: our implementation is coherent with this.

The constraint $(\phi_1^{\cdot})_\sharp \alpha = \beta$ ensuring that input domain α maps to the target domain β isn't straightforward to implement as already noted in the presentation general algorithm. We do so by optimizing an iterative Lagrangian relaxation associated to eq. 4.12, introducing a measure of discrepancy D between output and target domains:

$$\min_{\theta} \quad \mathcal{C}_d(\theta) + \frac{1}{\lambda_i} D\Big((\phi_1^{\cdot})_\sharp \alpha,\, \beta\Big) \tag{4.13}$$

where the sequence of Lagrange multipliers $(\lambda_i)_i$ converges linearly to 0 during optimization. At the limit, as the sequence of multipliers converges to 0, the constraint is satisfied. Note that we do not use here the Uzawa-like iterates here for the λ_i as this simpler heuristic was sufficient.

Each λ_i induces an optimization problem which is solved using stochastic gradient based techniques. As in most approaches for UDT, D may be implemented using generative adversarial networks, or any other appropriate measure of discrepancy between measures, such as kernel distances. Moreover, in order to stabilize the adversarial training which enforces boundary conditions for both our model and CycleGAN, we use an auto-encoder to a lower dimensional latent space. This limits the sharpness of output images but produces consistent and reproducible results, thus allowing meaningful comparisons which is the objective here.

Algorithm Training is done only for the forward equation and an approximation of the reverse is obtained by iterating $y_{k-1} = y_k - \Delta t\, v^{\theta_k}(y_k)$, starting from a sample y_K from β, as Section 4.8.1

allows to. Algorithm 1 gives all necessary details of the procedure.

Algorithm 4: Training Procedure for UDT tasks

Input: Dataset of unpaired images $(I_{\mathbb{A}}, I_{\mathbb{B}})$ sampled from (α, β),
initial coefficient λ_0, decay parameter d, initial parameters θ, minimal penalization ϵ
Pretrain Encoder E and decoder D
Make dataset of encodings $(x = E(I_{\mathbb{A}}), y = E(I_{\mathbb{B}}))$
while *not converged* **do**
  Randomly sample a mini-batch of x, y
  Solve forward equation $\phi^x_{k+1} = \phi^x_k + \Delta t\, v^{\theta_k}(\phi^x_k)$, starting from $\phi^x_0 = x$
  Estimate loss $\mathcal{L} = \mathcal{C}_d(\theta) + \frac{1}{\lambda_i} D\Big((\phi^{\cdot}_1)_\sharp \alpha,\, \beta\Big)$ on mini-batch
  Compute gradient $\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\theta}$ backpropagating through forward equation
  Update θ in the steepest descent direction
  $\lambda_{i+1} \leftarrow \max(\lambda_i - d, \epsilon)$

Output: Learned parameters θ.

Architectures. Implementation is performed via DCGAN and ResNet architectures as described below. For the Encoder, we use a standard DCGAN architecture[6], augmenting it with 2 self-attention layers, mapping the images to a fixed, 128 dimensional latent vector. For the Decoder, we use residual up-convolutions, also augmented with 2 self-attention layers. We use 9 temporal steps, corresponding to as many residual blocks which consist of a linear layer, batch normalization, a non-linearity, and a final linear layer. The discrepancy D is implemented using generative adversarial networks with the discriminator being a simple MLP architecture of depth 3, consisting of linear layers with spectral normalization, and LeakyReLU($p = 0.2$).

Moreover, in the experiments below, our dataset is the CelebA dataset, resizing images to 128×128 pixels, without any additional transformation. The initial coefficient is $\lambda_0 = 1$, and the decay factor is set depending on the number of total iterations M, so as to be ϵ on the final iteration. Throughout all the experiments, we use the Adam optimizer with $\beta_1 = 0.5$ and $\beta_2 = 0.999$.

4.8.2 A Typical UDT Task

Taking the CelebA dataset, we consider the male to female task where the objective is to change the gender of the input image while keeping other characteristics of the image unchanged as much as possible.

Figure 4.7 illustrates how our model works for Male to Female translation (forward) and back (reverse) on the CelebA dataset, displaying intermediate images as the input distribution gradually transforms into the target distribution. **No cycle-consistency is being explicitly enforced** here and the **reverse is not directly parametrized nor trained** but still performs well. The model changes relevant high-level attributes when progressively aligning the distributions but doesn't change non-relevant features (hair or skin color, background,...) which is coherent to what is expected for an optimal map w.r.t. an attractive cost function (here the squared Euclidean one). Additional samples are available in Section 4.8 of the Appendix. All the experiments conducted in this work with our proposed OT framework have been implemented using this dynamical formulation.

Figure 4.9 shows that for a low initialization gain, both our method and CycleGAN give satisfying and similar solutions. When changing the value of this hyper-parameter, the CycleGAN mapping becomes unstable, producing outputs very different from the inputs.

The non-uniqueness of the solution of CycleGAN's optimization problem is highlighted here by the multiple mappings found for different initializations. It is also worth noting that, for CycleGAN,

[6] `https://github.com/pytorch/examples/tree/master/dcgan`

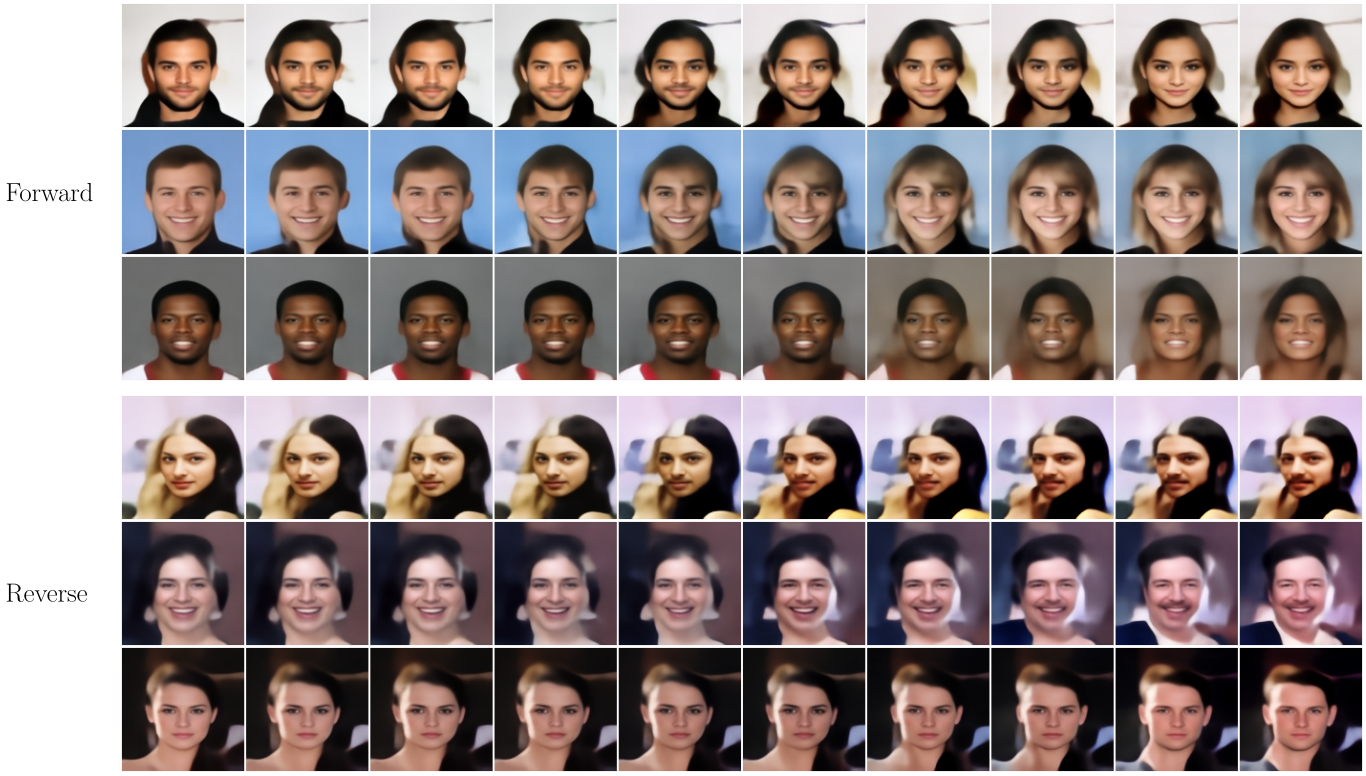


Figure 4.7: Male to Female translation (top) and the inverse (bottom). Intermediate images are the interpolations provided by the network's intermediate layers. The reverse mapping is not trained. Additional samples are available in Figure 4.8.

using a large σ made convergence of the optimization harder. As already observed before, the chaotic behavior of the CycleGAN model correlates with an increase in the transport cost of the obtained mappings. This validates the quadratic OT bias of CycleGAN, showing that this model only works as an implicit OT mapping for a quadratic cost given a certain architecture, initialized and trained in a certain way. For this example, the prior induced by the quadratic transport cost is the right one and correctly captures the geometry of the task, as one wants to preserve as much as possible the characteristics of the input. By explicitly enforcing optimality w.r.t. the quadratic cost, the model becomes robust to changes in the initialization as the OT problem admits a unique solution for this cost.

4.8.3 Imbalanced CelebA Task

Here, we tackle the case of a corrupted dataset where **structural bias** is present in the target domain, which can be an important use case of UDT when fairness of the datasets is an issue [79]: samples from the target dataset are systematically corrupted. Here, our variational theory of DNN models for UDT tells us that vanilla CycleGAN should not succeed in finding the right mapping as it would fall prey to the corruption which is not taken into account in its formulation, explicitly or implicitly. The strength of our approach is in this case to allow replacing the implicit minimal kinetic energy bias with an explicit one suitable to the context.

More specifically, we consider a subset of the CelebA dataset, where domains correspond to:

- α is a distribution over $\mathbb{A}$ which contains female faces with **black hair** which are **non-smiling**;
- β is a distribution over $\mathbb{B}$ which contains female faces with **black hair** which are **smiling**

However, we only have access to biased samples from β, where female faces have **blond** instead of

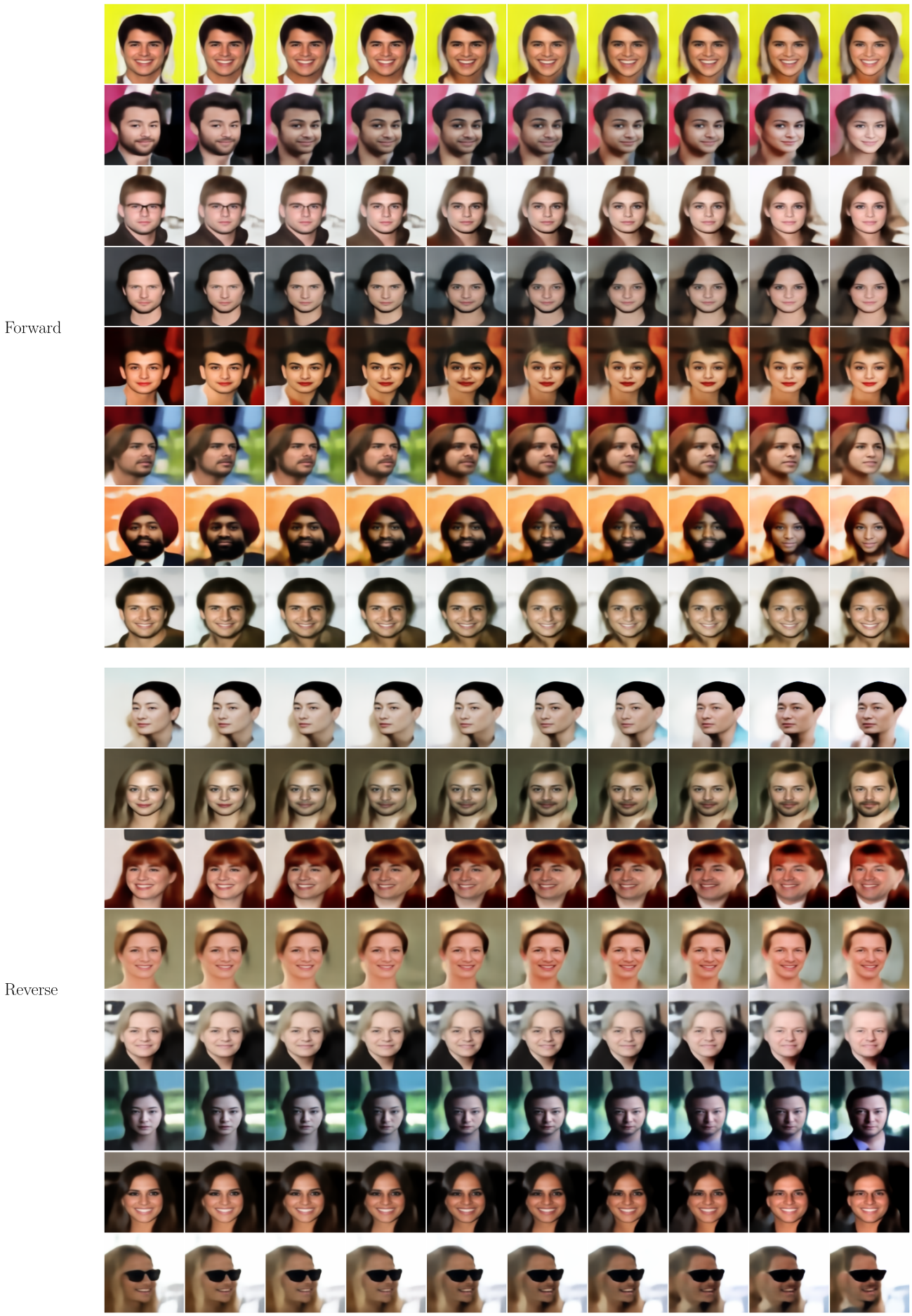


Figure 4.8: Additional Samples: Male to Female, and Back.

Figure 4.9: Each column associates one input image to its outputs for different models: CycleGAN and our model with different initial gain parameters. We have ensured convergence of all models to the same fit to the target distribution.

black hair. In other words, the task here is about adding smiles while not changing other features of the face.

In Fig. 4.10, we report results with CycleGAN and our approach with the quadratic cost: the hair color is modified along with the *smile* feature, and black-haired non smiling faces are mapped to blond smiling ones as should be expected from both. This highlights a particular case where CycleGAN's implicit bias fails as well as its explicitly biased quadratic cost counterpart.

Using our presented formulation, we are able to solve this task by changing the cost function: We use a a non-standard cost which is more suited to the geometry of the problem:

$$c(x, y) = \|H(x) - H(y)\|_2^2 \tag{4.14}$$

where $H(I)$ is a histogram function of the image I. More precisely, H is computed as a soft histogram over the colors of the image of 20 bins, using a Gaussian kernel with $\sigma = 0.05$ for the smoothing. This cost allows to take into account the colorimetry of the image, thus helping to find an OT map which preserves hair color in this case and re-balances the dataset as needed.

This task is an example of a case where a simple cost may help achieve non-trivial results when appropriate information is injected into it. In other words, by using prior knowledge on the corruption of the dataset, a cost function can be tailored to correct it.

More generally, it is not difficult to prove that a cost can almost always be designed to find the right solution for a given task between two distributions α and β among the infinity of candidates in the kernel of CycleGAN's loss. This is shown in the next subsection.

Figure 4.10: Results for Imbalanced CelebA Task. We wish to map faces that have the **Non-Smiling** and **Black Hair** attributes to **Smiling**, **Black-Hair** faces, while only accessing **Smiling**, **Blond Hair** faces for the target domain.

4.8.4 Finding the right cost

The following proposition shows that, under some technical assumptions, there always exists a ground transport cost which finds the desired mapping:

Proposition 4.6

With the same notations as before, suppose that a given UDT task between α and β is solved by a mapping T.

Then, if we assume that T is a differentiable map with a Jacobian everywhere invertible, there exists a cost function c such that the corresponding Monge problem yields T as its unique optimal transport map.

Proof: Suppose that T verifies the assumption. Let us define the following cost:

$$c(x, y) = \|T(x) - y\|_2^2$$

In this case, c is differentiable w.r.t. its first variable by differentiability of T. For $x_0 \in \mathbb{A}$, we then have that:

$$\forall y,\ \nabla_{x_0} c(x_0, y) = 2\ {}^t(\mathrm{Jac}_{x_0} T)(T(x_0) - y)$$

which is injective in y by invertibility of $\mathrm{Jac}_{x_0} T$. Thus c verifies the Twist condition and we can conclude about the existence and uniqueness of the OT map of the corresponding Monge problem.

Moreover, for any transport map T', we have that $\mathcal{C}(T') \geq 0$ and we also have that $\mathcal{C}(T) = 0$ which shows that T is indeed the OT map for c by unicity of the optimum. □

Note that the proof is not constructive. In practice, such a cost can handcrafted using knowledge about the task, as has been done in the Imbalanced dataset experiment, or it could even be learned if

some paired samples are available as a supervision (along with some additional assumptions on the desired mapping for example).

4.9 Application to classification

This section studies the application of LAP NN training in the case of classification tasks. We start by outlining a few practical details then evaluate different instances of our framework for the standard MNIST, CIFAR10 and CIFAR100 datasets.

4.9.1 Practical details

Instanciation of the algorithm In the following experiments, we instanciate the general LAP model with $p = 2$, meaning that we minimize the kinetic energy. Starting from problem eq. 4.8 with $p = 2$ and the Euclidean norm, we discretize the differential equation via a forward Euler scheme, which yields $\phi^x_{k+1} = \phi^x_k + v_k(\phi^x_k)$ similarly to the UDT case. The discretized derivative v_k is parameterized by a residual block, giving us a standard residual architecture. This choice was in part made to mimick the actual architecture of standard, single representation ResNets. Finally, as we only have access to a finite set $\mathcal{X}$ of samples x from α, we use a Monte-Carlo approximation of the integral $w.r.t$ the distributions $(\phi^i_t)_\sharp\alpha$, to obtain the optimization problem:

$$\begin{aligned} \min_{\theta} \quad & \mathcal{C}(\theta) = \sum_{x\in\mathcal{X}} \sum_{k=0}^{K-1} \|v_k(\phi^x_k)\|_2^2 \\ \text{s.t.} \quad & \phi^x_{k+1} = \phi^x_k + v_k(\phi^x_k), \\ & \phi^x_0 = x, \ \forall\, x \in \mathcal{X}, \\ & \mathcal{L}(\theta) = 0 \end{aligned}$$

This problem is solved using the iterative method described in Section 3.2.2 of Chapter 3.

Hyper-parameters Orthogonal initialization with gain 0.01 is used for all ResNet models. Kaiming initialization is used for all ResNeXt models. SGD is used for training all models. The momentum is always set to 0.9 and weight decay to 0.0001. For ResNet models, the learning rate is 0.01 and is divided by 5 at epochs 120, 160 and 200 (when the training goes that far). For ResNeXt models, the learning rate is 0.1 and is divided by 10 at epochs 150, 225 and 250. Batch size is 128 for all experiments. Architectures of ResNet [113] and ResNeXt [277] blocks are standard and exactly as in the references. The ResNets used are single representation ResNets (i.e. containing one residual stage) with 9 residual blocks. The ResNeXt architecture used is the ResNeXt-50-32×4d from [277]. As a reminder, the residual block of a ResNeXt applies $x + \sum_i w_i(x)$ with the functions w_i having the same architecture but independent weights, followed by a ReLU activation. More specific details are given for each experiment.

4.9.2 MNIST Experiments

The base model is a ResNet with 9 residual blocks. Two convolutional layers first encode the image of shape $1 \times 28 \times 28$ into shape $32 \times 14 \times 14$. A residual block contains two convolutional layers, each preceded by a ReLU activation and batch normalization. The classifier is made up of two fully connected layers separated by batch normalization and a ReLU activation. We use an orthogonal initialization [236] with gain 0.01. This and all vanilla models and their training regimes are implemented by following closely the cited papers that first introduced them and our method is added on top of the standard training regimes. For performance comparisons, we average the highest

test accuracy achieved over 30 training epochs (over random orthogonal weight initializations and random subsets of the complete training set).

From the experiments in two dimensions, we suspect that adding the transport cost helps when the training set is small. Indeed, when using the entire training set, the task is essentially solved (99.4% test accuracy). We penalize the transport cost iteratively, using $\lambda_0 = 5$, $\tau = 1$ and $s = 5$. The performance essentially stays the same(99.3% test accuracy).

On the other hand, as shown in Table 4.1, we find that adding the transport cost improves generalization when the training set is very small: the improvement becomes more important as the training set becomes smaller and reaches an increase of almost 14 percentage points in the average test accuracy for the smallest training set size.

Table 4.1: Average test accuracy and 95% confidence interval of ResNet9 over 50 instances on MNIST with training sets of different sizes (in %)

Training set size	ResNet	LAP-ResNet (Ours)
500	90.8, [90.4, 91.2]	**90.9**, [90.7, 91.1]
400	88.4, [88.0, 88.8]	**88.4**, [88.0, 88.8]
300	83.5, [83.0, 84.1]	**86.2**, [85.8, 86.6]
200	74.9, [73.9, 75.9]	**82.0**, [81.5, 82.5]
100	56.4, [54.9, 58.0]	**70.0**, [69.0, 71.0]

4.9.3 CIFAR10 Experiments

We run the same experiments on CIFAR10 as with the MNIST dataset. The architecture is exactly the same except that the encoder transforms the input which is of shape $3 \times 32 \times 32$ into shape $100 \times 16 \times 16$. For our method, we use $\lambda_0 = 0.1$, $\tau = 0.1$ and $s = 50$. We average the highest test accuracy achieved after 200 training epochs over random orthogonal weight initializations and random subsets of the complete train set.

Here, we find that adding the transport cost helps for all sizes of the train set (which has 50 000 images in total) which is to be expected as the task is more complex and benefits from minimizing the transport cost. However, the increase in average precision becomes more important as the train set becomes smaller (Table 4.2).

Table 4.2: Average test accuracy and 95% confidence interval of ResNet9 over 20 instances on CIFAR10 with training sets of different sizes (in %)

Training set size	ResNet	LAP-ResNet	Regularized ResNet, $\lambda = 0.2$
50 000	91.49, [91.40, 91.59]	**91.94**, [91.84, 92.04]	91.36, [91.28, 91.44]
30 000	88.61, [88.47, 88.75]	**89.41**, [89.31, 89.50]	88.50, [88.38, 88.61]
20 000	85.73, [85.59, 85.87]	**86.74**, [86.61, 86.87]	85.82, [85.70, 85.93]
10 000	79.25, [79.00, 79.49]	**80.90**, [80.74, 81.06]	80.15, [80.02, 80.28]
5 000	70.32, [70.00, 70.63]	**72.58**, [72.36, 72.79]	72.03, [71.71, 72.34]
4 000	67.80, [67.55, 68.07]	**70.12**, [69.81, 70.42]	69.64, [69.35, 69.94]
1 000	49.22, [48.69, 49.74]	**51.14**, [50.69, 51.59]	50.38, [49.92, 50.82]
500	41.55, [41.14, 41.96]	**42.92**, [42.54, 43.29]	42.30, [41.88, 42.73]
100	26.98, [25.98, 27.97]	25.34, [24.63, 26.10]	**27.53**, [26.59, 28.47]

4.9.4 CIFAR100 experiments

We used both the ResNet and the ResNeXt architecture for this more difficult task, namely the ResNeXt-50-32×4d architecture detailed in [277]. This is a much bigger and state-of-the-art network, as compared with the single representation ResNet used so far. It also extends the experimental results beyond the theoretical framework in three ways: the embedding dimension changes between the residual blocks, a block applies $x_{k+1} = \texttt{ReLU}(x_k + \sum_i w_{k,i}(x_k))$ and the encoder is no longer fixed. We found that penalizing $\sum_i w_{k,i}(x_k)$ or $x_{k+1} - x_k$ is essentially equivalent. Table 4.3 shows consistent accuracy gains as our method (with $\lambda_0 = 1$, $\tau = 0.1$ and $s = 5$) corrects a slight overfitting of the bigger ResNeXt compared to the ResNet.

Table 4.3: Average highest test accuracy and 95% confidence interval of ResNeXt50 over 10 instances on CIFAR100 with training sets of different sizes (in %)

Training set size	ResNeXt	LAP-ResNeXt (Ours)
50 000	72.97, [71.79, 74.14]	**76.11**, [75.32, 76.89]
25 000	62.55, [60.18, 64.92]	**64.11**, [62.25, 65.96]
12 500	45.90, [43.16, 48.67]	**48.23**, [46.39, 50.07]

An important observation is that adding the transport cost significantly reduces the variance in the results. This is expected as the model becomes more constrained and can be seen as an advantage, especially in cases where the results vary more with the initialization (*e.g.* transfer learning). This is illustrated by the width of the 95% confidence intervals in the tables above often becoming narrower when the transport cost is penalized.

Finally, we have also considered a relaxation of the optimization program by considering a fixed weight λ, which provides a simpler and quite competitive benchmark as there is no need for iterative optimization in this case.

Table 4.4: Average highest test accuracy and 95% confidence interval of ResNet9 over 10 instances on CIFAR100 with training sets of different sizes (in %)

Training set size	ResNet	LAP-ResNet	Reg. ResNet[7], $\lambda \in \{0.05, 0.2\}$
50 000	72.32, [72.08, 72.56]	72.43, [72.25, 72.61]	**72.62**, [72.41, 72.83]
25 000	64.34, [64.10, 64.57]	64.34, [64.11, 64.58]	**64.76**, [64.52, 65.00]
10 000	49.27, [48.84, 49.69]	**50.57**, [50.34, 50.80]	50.46, [50.19, 50.72]
5 000	34.74, [33.90, 35.58]	37.97, [37.68, 38.27]	**38.44**, [37.99, 38.89]
1 000	15.66, [15.23, 16.08]	**16.42**, [16.10, 16.75]	16.03, [15.55, 16.52]

Table 4.5: Average highest test accuracy and 95% confidence interval of ResNeXt50 over 10 instances on CIFAR100 with training sets of different sizes (in %)

Training set size	ResNeXt	LAP-ResNeXt	Reg. ResNeXt, $\lambda = 0.01$
50 000	72.97, [71.79, 74.14]	**76.11**, [75.32, 76.89]	75.96, [74.92, 77.01]
25 000	62.55, [60.18,64.92]	**64.11**, [62.25, 65.96]	64.10, [62.36, 65.84]
12 500	45.90, [43.16, 48.67]	**48.23**, [46.39, 50.07]	47.77, [45.93, 49.62]

4.9.5 Discussion of the results

The first observation here is that, similarly to what has been observed in the UDT task, in a standard setting where hyper-parameters are carefully chosen and with enough training data points, standard residual architectures behave essentially similarly to models instanciating our LAP framework. This validates our claim that "well trained" NNs are actually implicitly minimizing a transport energy.

Moreover, we have seen, as predicted by our theoretical development, that, in settings where data is more scarce, LAP networks tend to improve on standard architectures, all else being equal, by alleviating overfitting. This means that transport cost minimization does capture some of the regularity needed to model the data in those tasks. The most interesting factor here is that LAP training can come on top of usual practices without much changes to the code or to training time, even when the architecture is not the simplified single representation ResNet which is closer to our theoretical formalism as the experiments with the ResNeXt have shown.

Note that in all experiments we have used the quadratic cost. The most natural question is then to ask whether it is possible to use this LAP framework with customized costs as a way to model prior knowledge of a certain classification task or a certain dataset. We leave this question for future research.

4.10 An Extension: Layerwise Training of DNNs

In this section, we provide a natural extension to our variational framework for the classification application to the layerwise training of deep classifiers.

4.10.1 Motivation and General Idea

Layer-wise or module-wise training of neural networks is compelling in constrained and on-device settings, as it circumvents a number of problems of end-to-end backpropagration. However, this method suffers itself from the stagnation problem, whereby early layers tend to overfit while deeper layers stop increasing test accuracy after a certain depth. We propose to solve this issue by introducing a simple layer-wise training scheme inspired by the minimizing movement scheme for gradient flows in distribution space. This method is particularly well-adapted to residual networks, and is amenable to theoretical study. Experimentally, we show improved accuracy of our networks compared to classical module-wise training schemes.

End-to-end backpropagation is the standard training method of neural nets. But there are reasons to look for alternatives:

- it requires loading the whole model during training which can be impossible in constrained settings such as training on mobile devices [259, 257];
- it forces the training of systems of cooperative networks to be sequential and synchronous, which is not flexible enough when the networks are distributed between a central agent and clients that operate at different rates [129];
- it prohibits training the layers in parallel which can be useful to accelerate training.

These limitations follow from the three locking problems that end-to-end backpropagation suffers from: forward locking (each layer must wait for the previous layers to process its input), update locking (each layer must wait for the end of the forward pass to be updated) and backward locking (each layer must wait for errors to backpropagate from the last layer to be updated) [129]. Dividing the network into modules, a module being made up of one or more layers, and greedily solving module-wise optimization problems sequentially (i.e. one after the other) or in parallel (i.e. batch-wise), solves update locking

(and backward locking). When combined with batch buffers, parallel module-wise training solves all three problems [28] and allows parallel training of the modules. Module-wise training is appealing in memory-constrained settings as it works without most gradients and activations needed in end-to-end training. When done sequentially, it only requires loading and training one module (so possibly one layer) at a time. Despite its simplicity, module-wise training has been shown to scale well [28, 214], outperforming more complicated methods addressing the locking problems e.g. synthetic [129, 67] and delayed [123, 122] gradients.

The typical setting of (sequential) module-wise training for minimizing a loss $\mathcal{L}$ is, given a dataset $\mathcal{D}$, to solve one after the other, for $1 \leq k \leq K$, Problems

$$(T_k, F_k) \in \arg\min_{T,F} \sum_{x \in \mathcal{D}} \mathcal{L}(F, T(G_{k-1}(x))) \tag{4.15}$$

where $G_k = T_k \circ ... \circ T_1$ for $1 \leq k \leq K$ and $G_0 = \texttt{id}$. Here, T_k is the k-th module (one or many layers) thus receiving the output of module T_{k-1}, and F_k is an auxiliary classifier that processes the outputs of T_k so the loss can be computed, which is consistent with the notations of the other classification tasks in this chapter. With those notations, each $F_i \circ G_i$ is an intermediate classifier and $F_K \circ G_K$ is the final one. A well-known fact in the literature is that, when the modules are numerous and shallow, module-wise training suffers from a *stagnation problem*, whereby greedy early modules overfit and learn more discriminative features than end-to-end training [179], and deeper modules don't improve the test accuracy [270], or even sometimes degrade it.

Many approaches exist to tackle this issue, [270] proposing for example to maximize the mutual information that each module keeps with the input, in addition to minimizing the loss. We propose a different one, leveraging our previous variational formulation for the training of neural classifiers. Indeed, module-wise training can be seen as the gradual transport of the initial data distribution towards the set of admissible target distributions as formalized previously. We can then apply the same ideas as before by making the trajectory follow a gradient flow in an appropriately chosen space which corresponds to the Wasserstein Gradient Flow when using the same kinetic cost.

More precisely, given $\mathcal{Z} : \mathbb{W}_2(\Omega) \to \mathbb{R}$, which in our case is induced by the classification loss $\mathcal{L}$, the minimizing movement scheme is a discretized gradient flow that is well-defined in non-Euclidean metric spaces and minimizes (under some conditions) $\mathcal{Z}$ starting from $\rho_1^\tau \in \mathcal{P}(\Omega)$. It is given by

$$\rho_{k+1}^\tau \in \arg\min_{\rho \in \mathcal{P}(\Omega)} \ \mathcal{Z}(\rho) + \frac{1}{2\tau} W_2^2(\rho, \rho_k^\tau) \tag{4.16}$$

This equation can be interpreted as a non-Euclidean version of the implicit Euler scheme for following the gradient flow of $\mathcal{Z}$, or as a Wasserstein proximal step to minimize $\mathcal{Z}$. Under mild technical conditions on $\mathcal{Z}$, we have the existence of the scheme eq. 4.16 and we can have global convergence under stricter conditions. Note in particular that τ corresponds here to the time-step of the scheme and thus should be taken as small as possible: the continuous gradient flow corresponds to τ tending to 0.

Because parametrizing distributions in $\mathcal{P}(\Omega)$ isn't doable in practice, a more practical solution is to consider instead the optimization problem

$$T_k^\tau \in \arg\min_T \mathcal{Z}(T_\sharp \rho_k^\tau) + \frac{1}{2\tau} \int_\Omega \|T(x) - x\|^2 \mathrm{d}\rho_k^\tau(x) \tag{4.17}$$

where T is represented, as usual, with a Neural Network. This takes us closer to the layerwise learning problem for which we recover a regularized version by replacing $\mathcal{Z}$ with $\mathcal{L}$

$$\begin{aligned} &(T_k^\tau, F_k^\tau) \in \\ &\arg\min_{T,F} \ \mathcal{L}(F, T_\sharp \rho_k^\tau) + \frac{1}{2\tau} \int_\Omega \|T(x) - x\|^2 \mathrm{d}\rho_k^\tau(x) \end{aligned} \tag{4.18}$$

with $\rho_{k+1}^\tau = (T_k^\tau)_\sharp \rho_k^\tau$ and $\rho_1^\tau = \alpha$ the initial data distribution.

4.10.2 Practical Method

eq. 4.17 thus defines our proposed training method: each (T_k, F_k) is obtained by solving a discretized version of this optimization problem. The intuition here is that we keep greedily-trained modules from overfitting and destroying information needed by deeper modules by penalizing their kinetic energy to force them to preserve the geometry of the problem as much as possible at each step of the transformation.

Given $\tau > 0$, we now solve, for $1 \leq k \leq K$, Problems

$$(T_k^\tau, F_k^\tau) \in \arg\min_{T,F} \tag{4.19}$$
$$\sum_{x \in \mathcal{D}} \mathcal{L}(F, T(G_{k-1}^\tau(x)) + \frac{1}{2\tau}\|T(G_{k-1}^\tau(x)) - G_{k-1}^\tau(x)\|^2$$

where $G_k^\tau = T_k^\tau \circ ... \circ T_1^\tau$ for $1 \leq k \leq K$ and $G_0^\tau = \texttt{id}$. The final network is now $F_K^\tau \circ G_K^\tau$. Intuitively, we can think that this biases the modules towards moving the points as little as possible, thus at least keeping the performance of the previous module. In the following, we instanciate our method with ResNets but it could be applied to any architecture where the quantity $T(x) - x$ can be computed.

In particular, if each module T_k is made up of M ResBlocks, i.e. applies $x_{m+1} = x_m + r_m(x_m)$ for $0 \leq m < M$, then its total discrete kinetic energy for a single data point x_0 is the sum of its squared residue norms $\sum\|r_m(x_m)\|^2$, by the standard equivalence between a ResNet and the discrete Euler scheme for an ordinary differential equation [275] with velocity field r:

$$x_{m+1} = x_m + r_m(x_m) \iff \frac{\mathrm{d}x_t}{\mathrm{d}t} = r_t(x_t) \tag{4.20}$$

and $\sum\|r_m(x_m)\|^2$ is then the discretization of the total kinetic energy $\int_0^1 \|r_t(x)\|^2 \mathrm{d}t$ of the ODE. If ψ_m^x denotes the position of a point x after m ResBlocks, then regularizing the kinetic energy of multi-block modules means solving

$$(T_k^\tau, F_k^\tau) \in \tag{4.21}$$
$$\arg\min_{T,F} \sum_{x \in \tilde{\rho}_0} (\mathcal{L}(F, T(G_{k-1}^\tau(x)) + \frac{1}{2\tau} \sum_{m=0}^{M-1} \|r_m(\psi_m^x)\|^2)$$
$$\text{s.t. } T = (\texttt{id} + r_{M-1}) \circ ... \circ (\texttt{id} + r_0)$$
$$\psi_0^x = G_{k-1}^\tau(x), \psi_{m+1}^x = \psi_m^x + r_m(\psi_m^x)$$

We also want to minimize this sum of squared residue norms instead of $\|T(x) - x\|^2$ which no longer exactly coincide, as it works better in practice, which we assume is because it offers a better and more localized control of the transport.

We consider two ways of solving the module-wise training problems:

Sequential module-wise training where each module is trained for N epochs before training the next module, which receives as input the output of the previous trained module, thus allowing to load only one module at a time in memory;

Parallel module-wise training where a complete forward pass is done for each batch, updating the parameters of each module separately and allowing for the parallelization of training across different modules as described in [28].

Given its meaning, we might want to consider a fixed τ which would be as small as possible. However, instead, we might want to vary it along the depth k to further constrain with a smaller τ_k the earlier modules in order to avoid overfitting or the later modules in order to maintain the accuracy of earlier modules. In practice, we found that using a value which is twice as small for the first half of the modules is a simple heuristic which works well. Note that more involved choices could be used and might improve performance.

Table 4.6: Test accuracy of 4-4 ResNet models on TinyImageNet with parallel TRGL (ours) and parallel BaselineGL, compared to methods DGL and PredSim from [214] that also split their networks in 4 module-wise-parallel-trained modules.

Parallel BaselineGL	Parallel TRGL (ours)	DGL ResNet-152	PredSim ResNet-152
59.72	**61.13**	57.64	51.76

4.10.3 Experiments

We test our method on classification tasks, with $\mathcal{L}$ being the cross-entropy loss. For the ResBlocks, we use the architecture from [114] with two convolutional layers in each block. For the auxiliary classifiers, unless stated otherwise, we use the architecture from [27, 28], that is a convolutional layer followed by an average pooling layer and a fully connected layer. The experiments below then show that our method combines well with theirs and improves its results when using ResNets. Moreover, we use rather simple and wide ResNets with 256 filters initially and downsampling and doubling of the filters at the midpoint, no matter the depth. If the network is divided in K modules of M residual blocks each, we call the network a $K-M$ ResNet. We call our method TRGL for Transport-Regularized Greedy Learning. We use orthogonal initialization [236] with a gain of 0.05 and standard data augmentation. For sequential and multi-lap sequential training, we use SGD with a learning rate of 0.007. For parallel training we use SGD with learning rate of 0.003. These architectural and training choices have been made to improve the baseline performance of greedy module-wise training without transport regularization, which we will call BaselineGL when included in the results for comparison and ablation study purposes.

We compare our method with other standard ones by focusing on parallel module-wise training with few modules, as it performs better than sequential module-wise training and is therefore the most explored in recent papers. We train a 16-block ResNet divided in 4 modules of 4 blocks each (we denote this a 4-4 ResNet) module-wise on the classical TinyImageNet dataset. Parallel module-wise training in this case consumes about 25% less memory than end-to-end training (6951MiB vs 9241MiB), and we train for 200 epochs. We compare in Table 4.6 our results in this setup to those of two of the best recent parallel module-wise training methods: DGL [28] and PredSim [206] from Table 2 in [214]. The benefit of the regularization is clear as it adds 1.4 percentage points of accuracy to the baseline which uses the exact same architecture. While our network has around 51 million parameters, our method easily beats the first two methods, even when they use a bigger ResNet-152 (around 59 million parameters), also divided in four modules.

In Fig. 4.11, we look at the test accuracy of each block after block-wise training with and without our regularization. On the left, from experiments with sequential block-wise training on a train set of 1000 CIFAR10 images, we see a large decline in performance after the first block (block 0 being the encoder) that the regularization completely avoids. On the right, from experiments with parallel block-wise training on a train set of 5000 CIFAR100 images, we see a steeper increase in test accuracy along the blocks with the regularization. These observations suggest that our regularization helps most when looking at the accuracy of the last block in a deep block-wise-trained model. We also observe that with the regularization the difference between the accuracy of the last block and that of the best block is smaller than without the regularization.

Those experiments show that our approach holds some promise for blockwise training for any variant and many standard underlying architectures with substantial improvement over strong state-of-the-art baselines. It remains to be tested on larger datasets, for which we expect it to give an even stronger improvement and tasks other than image classification. In particular, it would be interesting to find cases where a ground cost other than the quadratic one might be useful.

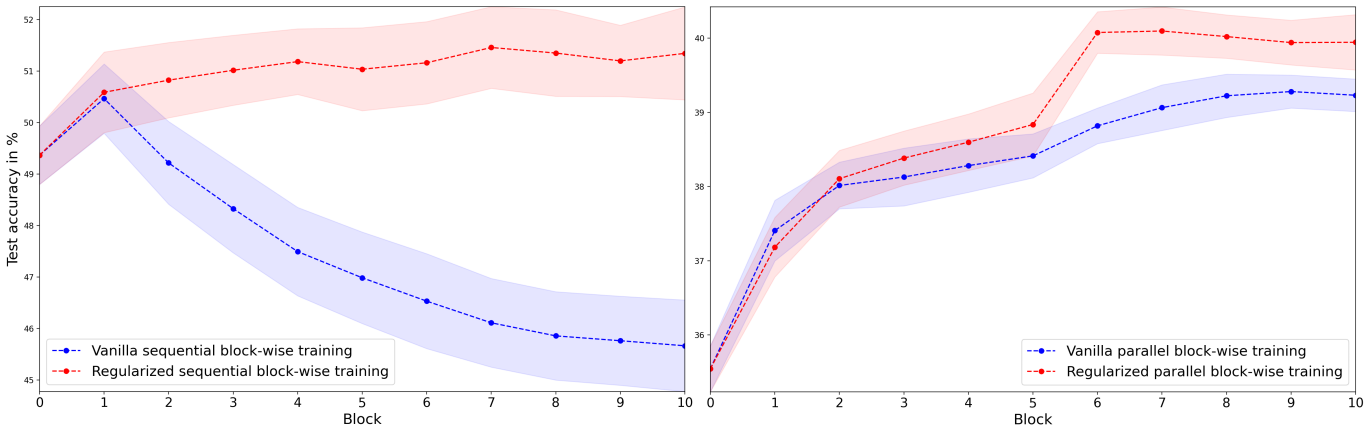


Figure 4.11: Highest test accuracy after each block of 10-1 ResNet models averaged over 10 runs with 95% confidence intervals. Left: sequential vanilla (BaselineGL, in blue) and regularized (TRGL, in red) block-wise training on 2% of the CIFAR10 training set. Right: parallel vanilla (BaselineGL, in blue) and regularized (TRGL, in red) block-wise training on 10% of the CIFAR100 training set.

4.11 Variational DNNs and the Navier-Stokes Experiments

This section shortly discusses the findings observed in Section 2.5.5 of Chapter 2. The goal here is to better our understanding of those puzzling observations in the light of the variational theory formulated in this chapter. Note in particular that the considerations presented here are specific to the exact setting of those experiments, in particular regarding the nature of the observation operator $\mathcal{H}$ which projects the states onto the density of the fluid denoted Y in Chapter 2.

Because we took a relatively small viscosity constant in our experiments, let us consider the incompressible Euler PDE on a time interval $[0, T]$ and a domain D with smooth boundary δD:

$$\begin{aligned} \frac{\partial u}{\partial t} + (u \cdot \nabla)u &= -\frac{\nabla p}{\rho} \\ \nabla \cdot u &= 0 \\ u_{|\delta D} \cdot n &= 0 \end{aligned} \tag{4.22}$$

Given a smooth solution u, we can then consider the corresponding flow, defined with the following

$$\begin{aligned} \forall t, x \in [0, T] \times D, \frac{\partial g_t(x)}{\partial t} &= u(t, g_t(x)) \\ g_0 &= \mathrm{id} \end{aligned} \tag{4.23}$$

In other words, g_t represents the transformation of the initial density distribution of the fluid in the domain D to reach the configuration of time t for the velocity u which is solution to the PDE.

Following [13, 89], under technical conditions out of the scope of this informal discussion, it is known that this flow solves the following variational formulation

$$\begin{aligned} \min_{g} \quad & \int_0^T \int_D \left| \frac{\partial g_t}{\partial t}(x) \right|^2 \mathrm{d}x\mathrm{d}t \\ \text{s.t.} \quad & \forall t,\ g_t \in \mathrm{SDiff}(D), \\ & g_0 = \mathrm{id}, \\ & (g_T)_\sharp Y_0 = Y_T \end{aligned} \tag{4.24}$$

where $\mathrm{SDiff}(D)$ is the set of smooth diffeomorphisms of D onto itself and Y_0, resp. Y_T, is the distribution of fluid at time 0, resp. at time T. The rever is also true, meaning that, again under

technical assumptions which we won't delve into here, the minimizing g of this variational problem is the flow of the corresponding incompressible Euler equation.

Let us recall Equation (2.3) of Chapter 2, in the Setting where an initial state is provided

$$\begin{aligned} \underset{\theta}{minimize} \quad & \mathbb{E}_{(X_0,Y_{1:T})\in \text{Dataset}}\left[\mathcal{J}(Y_{1:T}, Y^\theta_{1:T})\right] \\ \text{subject to} \quad & \frac{\mathrm{d}X^\theta_t}{\mathrm{d}t} = F_\theta(X^\theta_t), \\ & X^\theta_0 = X_0 \end{aligned} \tag{4.25}$$

which is still, as shown in Chapter 2, under-determined as an infinite number of functions could lead to a null loss over the dataset. In the following, for the sake of coherence, we redefine notations by denoting the trajectory $(X_t)_t$ starting at a state X_0 as $\phi^{X_0}_t$.

Going back to the hypothesis formulated in this chapter, which is that DNNs tend to solve a given task by minimizing the associated dynamic kinetic energy, when the DNN parametrizing this optimization problem is learnt, the variational reformulation can then be written as

$$\begin{aligned} \underset{\theta}{minimize} \quad & \sum_{X_0\in\text{Dataset}} \int_0^T \int_D |F_\theta(\phi^{X_0}_t)(x)|^2 \mathrm{d}x\mathrm{d}t \\ \text{subject to} \quad & \mathbb{E}_{(X_0,Y_{1:T})\in \text{Dataset}}\left[\mathcal{J}(Y_{1:T}, Y^\theta_{1:T})\right] = 0, \\ & \frac{\mathrm{d}\phi^{X_0}_t}{\mathrm{d}t} = F_\theta(\phi^{X_0}_t), \\ & (\phi^{X_0}_t)_0 = X_0 \end{aligned} \tag{4.26}$$

The constraint enforced by the nullity of the loss here ensures that for all initial states in the dataset, the learned flow adequately transforms the initial distribution of the fluid Y_0 into the supervision fluid densities Y_t. In other words, we can rewrite the optimization problem as

$$\begin{aligned} \underset{\theta}{minimize} \quad & \sum_{X_0\in\text{Dataset}} \int_0^T \int_D |F_\theta(\phi^{X_0}_t)(x)|^2 \mathrm{d}x\mathrm{d}t \\ \text{subject to} \quad & \forall (X_0, Y_{1:T}) \in \text{Dataset}, \forall t \in \{1, ..., T\}, (\phi^{X_0}_t)_\sharp Y_0 = Y_t, \\ & \frac{\mathrm{d}\phi^{X_0}_t}{\mathrm{d}t} = F_\theta(\phi^{X_0}_t), \\ & (\phi^{X_0}_t)_0 = X_0 \end{aligned} \tag{4.27}$$

This means in particular that each $\phi^{X_0}_t$ solves Equation (4.24). It follows that it is the flow of the incompressible Euler equation and thus the first component of $F_\theta(X_t)$ is equal to the solution velocity field defined by the PDE with initial state X_0.

While those considerations do not constitute a proof, this helps us understand why it would be useful for the DNN F_θ to preserve the structure of the velocity field given in X_0 as, acccording to the variational reformulation we propose, it is biased towards using it for the observed part of the state.

4.12 Conclusion

While many results exist to show that DNNs are universal approximators and practice shows that they are able indeed to express complex functions of their inputs, this seems to make it impossible to generalize to unseen data points. This is contrary to the empirical success observed for many tasks. Our goal here has thus been to show that, when taking into account the implicit biases neural models are endowed with, the actual class of functions which is actually learned in practice is smaller than

in theory. We present a characterization of the functions of this class as minimizers of a variational problem linked with Optimal Transport theory and coherent with empirical observations. On two application cases, this has allowed us to show that explicitly enforcing those implicit biases allows for more robustness and flexibility while exhibiting better theoretical properties for each task.

A natural extension to our work would be to consider the variational formalism introduced as a way to construct new models for challenging tasks. More precisely, one could take a very generic architecture then tailor an adequate cost for a given task and dataset. Conceiving a principled methodology for the construction of such a cost would be an important subject of future exploration. For example, building on the general existence and uniqueness results obtained for generalized costs on manifolds in [89], it should be possible to extend this trend of work in order to take into account the geometry of a given problem and dataset.

Chapter 5

A Neural Tangent Kernel view on GANs

The work presented in the previous chapter allowed us to pinpoint several important properties of "well-trained" DNNs in various standard tasks. While some hyper-parameter choices appear to be important in enforcing those properties in practice, two crucial aspects could not be studied under the presented variational framework: training dynamics and architectural choices. Both are obviously essential in the success of DNNs and the goal in this chapter is to sketch a way of analysing them.

More precisely, this chapter proposes a novel framework for the study of GANs, which is shown to be more consistent with practice, within the Neural Tangent Kernel Deep Learning theory. It corresponds to the following publication:

- *A Neural Tangent Kernel Perspective of GANs*, J-Y Franceschi*, E de Bézenac*, I Ayed*, P Gallinari, ICML 2022.

5.1 Introduction

Generative Adversarial Networks [102] have become a canonical approach to generative modeling as they produce realistic samples for numerous data types, with a plethora of variants [274]. These models are notoriously difficult to train and require extensive hyper-parameter tuning [43, 138, 163]. To alleviate these shortcomings, much effort has been put in gaining a better understanding of the training process, resulting in a vast literature on theoretical analyses of GANs. In particular, a large portion of those focus on studying GAN loss functions to conclude about their comparative advantages.

However, empirical evaluations [175, 143] showed that different GAN formulations can yield approximately the same performance in terms of sample quality and stability of the training algorithm, regardless of the chosen loss. This indicates that by focusing exclusively on the loss function, theoretical studies might not model practical settings adequately.

In particular, the discriminator being a trained neural network is not taken into account, nor are the corresponding inductive biases which might considerably alter the generator's loss landscape. Moreover, it can be shown, as we do, that neglecting this constraint hampers the analysis of gradient-based learning of the generator on finite training sets, since the gradient from the associated discriminator is ill-defined almost everywhere. These limitations thus hinder the potential of theoretical analyses to explain GANs' empirical behavior.

In this work, we provide a framework of analysis for GANs solving these issues by explicitly incorporating the discriminator's architecture. To this end, we leverage the advances in deep learning theory driven by Neural Tangent Kernels (NTKs) [128], and develop theoretical results showing the relevance of our approach. Under mild conditions on its architecture and loss, we are able to characterize the trained infinite-width discriminator for which we then establish differentiability properties. We do so by proving novel regularity results on its NTK. In other word, our framework

does overcome the limitations of previous analyses, making it more coherent with GAN as instanciated and trained in practice.

This more accurate formalism enables us to derive new insights about the generator. In particular, we formulate the dynamics of the generated distribution via the generator's NTK, and discuss its consequences by linking it to gradient flows in probability spaces. We deduce that, under the Integral Probability Metric loss, the generated distribution minimizes its Maximum Mean Discrepancy given by the discriminator's NTK w.r.t. the target distribution. Moreover, we release an analysis toolkit based on our framework, GAN(TK)^2, which we use to empirically validate our analysis. As an important illustration, we study the role of the ReLU activation in GAN architectures.

Let us give a reading guide for the following pages.

- Section 5.3 points a fundamental flaw in usual analysis of GANs as the gradient of the discriminator is discarded, thus not taking into account its parametrization and its spatial gradient field, which we show to be essential to the GAN's gradient descent dynamics.

- Section 5.4 introduces the NTK framework within which our work is conducted as well as the functional differential equation it implies which defines the trajectory of the discriminator function during training.

- Under the previously stated assumptions, Section 5.5 provides a thorough theoretical analysis of the discriminator function properties, starting from its definition as the unique solution of the training differential equation. In particular, we show how it evolves w.r.t. the RKHS of the NTK and study its differentiability properties for which we give broad sufficient conditions. Complete proofs of those results are provided as well.

- With the discriminator dynamics understood, it remains to see how this affects the generated distributions. Section 5.6 studies those consequences. We start by stating the general equation governing their trajectories in probability space which shows how such a presentation encompasses several previous analysis of GAN dynamics. Because the equation is intractable and its general study is largely out of the scope of this work, we give a few considerations about important specific GAN losses including LSGAN and those involving IPMs. We conclude the section with a proof of optimality under restrictive assumptions.

- Section 5.7 explores empiricially a few aspects of the theoretical analysis conducted before, generally showing quite a close correspondence between experiments and our analysis, even in finite width settings both using low dimensional easier to visualize datasets and high dimensional datasets closer to practice. We verify in particular the impact of architectural choices such as the presence of bias, the activation or the specificity of the NTK as compared to other standard kernels, on the convergence of the generator and the quality of the spatial gradient field generated by the discriminator.

- Section 5.8 offers miscrellaneous discussions regarding the previous presented work.

In the following, while complete technical proofs are provided, the reader can skip them without hampering his understanding of the work and when necessary the results are always stated separately beforehand.

5.2 Related Work

GAN Theory. A first line of research, started by [102] and pursued by many others [205, 294, 253], studies the loss minimized by the generator. Assuming that the discriminator is optimal and can take arbitrary values, different families of divergences can be recovered. However, as noted by [10], these

divergences should be ill-suited to GAN training, contrary to empirical evidence. Our framework addresses this discrepancy, as it properly characterizes the generator's loss and gradient.

Another line of work analyzes the dynamics and convergence of the generated distribution [196, 183, 182]. As the studied dynamics are highly non-linear, this approach typically requires strong simplifying assumptions, e.g. restricting to linear neural networks or reducing datasets to a single datapoint. The most advanced analyses taking into account the discriminator's parameterization are specialized to specific GAN models [20] and discriminators, such as a linear one, or random feature models [165, 21]. In contrast to these works, our framework of analysis provides a more comprehensive modeling as we establish generally applicable results about the influence of the discriminator's architecture on the generator's dynamics.

Neural Tangent Kernel. NTKs were introduced by [128], who showed that a trained neural network in the infinite-width regime equates to a kernel method, thereby making the dynamics of the training algorithm tractable and amenable to theoretical study. This fundamental work has been followed by a thorough line of research generalizing and expanding its initial results [14, 38, 151, 162, 246], developing means of computing NTKs [204, 281], further analyzing these kernels [82, 37, 49], studying and leveraging them in practice [294, 15, 150, 161, 256], and more broadly exploring infinite-width networks [160, 282, 3]. These prior works validate that NTKs can encapsulate the characteristics of neural network architectures, providing a solid theoretical basis to study the effect of architecture on learning problems.

Other works studied the regularity of NTKs [38, 283, 24] but, as far as we know, ours is the first to state general differentiability results for NTKs and infinite-width networks. While [126] sought to improve generators by investigating checkerboard artifacts in the light of NTKs and [59] introduced preliminary results in a simplified setting for the generator only, our contribution is the first to employ NTKs to comprehensively study GAN training.

5.3 Limits of Previous Studies

We present in this section the usual GAN formulation and illustrate the limitations of prior analyses.

First, we introduce some notations. Let $\Omega \subseteq \mathbb{R}^n$ be a closed convex set, $\mathcal{P}(\Omega)$ the set of probability distributions over Ω, and $L^2(\mu)$ the set of square-integrable functions from the support $\operatorname{supp}\mu$ of μ to $\mathbb{R}$ with respect to measure μ, with scalar product $\langle \cdot, \cdot \rangle_{L^2(\mu)}$. If $\Lambda \subseteq \Omega$, we write $L^2(\Lambda)$ for $L^2(\lambda)$, with λ the Lebesgue measure on Λ.

5.3.1 Generative Adversarial Networks

GAN algorithms seek to produce samples from an unknown target distribution $\beta \in \mathcal{P}(\Omega)$. To this extent, a generator function $g \in \mathcal{G}\colon \mathbb{R}^d \to \Omega$ parameterized by θ is learned to map a latent variable $z \sim p_z$ to the space of target samples such that the generated distribution α_g and β are indistinguishable for a discriminator $f \in \mathcal{F}$ parameterized by ϑ. The generator and the discriminator are trained in an adversarial manner as they are assigned conflicting objectives.

Many GAN models consist in solving the following optimization problem, with $a, b, c\colon \mathbb{R} \to \mathbb{R}$:

$$\inf_{g \in \mathcal{G}} \left\{ \mathcal{C}_{f^\star_{\alpha_g}}\left(\alpha_g\right) \triangleq \mathbb{E}_{x \sim \alpha_g}\left[c_{f^\star_{\alpha_g}}(x) \right] \right\} \tag{5.1}$$

where $c_f = c \circ f$, and $f^\star_{\alpha_g}$ is chosen to solve, or approximate, the following optimization problem:

$$\sup_{f \in \mathcal{F}} \left\{ \mathcal{L}_{\alpha_g}(f) \triangleq \mathbb{E}_{x \sim \alpha_g}\left[a_f(x) \right] - \mathbb{E}_{y \sim \beta}\left[b_f(y) \right] \right\}. \tag{5.2}$$

For instance, [102] originally used $a(x) = \log(1 - \sigma(x))$, $b(x) = c(x) = -\log(\sigma(x))$; in LSGAN [178], $a(x) = -(x+1)^2$, $b(x) = (x-1)^2$, $c(x) = x^2$; and for Integral Probability Metrics [194] used e.g. by [12], $a = b = c = \mathrm{id}$. Many more fall under this formulation [205, 159].

Equation (5.1) is then solved using gradient descent on the generator's parameters, with at each step $j \in \mathbb{N}$:

$$\theta_{j+1} = \theta_j - \eta \mathbb{E}_{z \sim p_z} \left[\nabla_\theta g_{\theta_j}(z)^\top \nabla_x c_{f^\star_{\alpha_{g_{\theta_j}}}}(x) \Big|_{x = g_{\theta_j}(z)} \right]. \tag{5.3}$$

This is obtained via the chain rule from the generator's loss $\mathcal{C}_{f^\star_{\alpha_g}}\left(\alpha_g\right)$ in Equation (5.1). However, we highlight that the gradient applied in Equation (5.3) differs from $\nabla_\theta \mathcal{C}_{f^\star_{\alpha_g}}\left(\alpha_g\right)$: the terms taking into account the dependency of the optimal discriminator $f^\star_{\alpha_{g_\theta}}$ on the generator's parameters are discarded. This is because the discriminator is, in practice, considered to be independent of the generator in the alternating optimization between the generator and the discriminator.

Since $\nabla_x c_{f^\star_\alpha}(x) = \nabla_x f^\star_\alpha(x) \cdot c'(f^\star_\alpha(x))$, and as highlighted e.g. by [102] and [10], the gradient of the discriminator plays a crucial role in the convergence of GANs. For example, if this vector field is null on the training data when $\alpha \neq \beta$, the generator's gradient is zero and convergence is impossible. For this reason, this paper is devoted to developing a better understanding of this gradient field and its consequences when the discriminator is a neural network. In order to characterize this gradient field, we must first study the discriminator itself.

5.3.2 On the Necessity of Modeling the Discriminator Parameterization

For each GAN formulation, it is customary to elucidate the loss implemented by Equation (5.2), often assuming that $\mathcal{F} = L^2(\Omega)$, i.e. the discriminator can take arbitrary values. Under this assumption, the original GAN yields the Jensen-Shannon divergence between α_g and β, and LSGAN a Pearson χ^2-divergence, for instance.

However, as pointed out by [16], the discriminator is trained in practice with a finite number of samples: both fake and target distributions are finite mixtures of Diracs, which we respectively denote as $\hat{\alpha}_g$ and $\hat{\beta}$. Let $\hat{\gamma}_g = \frac{1}{2}\hat{\alpha}_g + \frac{1}{2}\hat{\beta}$ be the distribution of training samples.

Assumption 5.1 (*Finite training set*)

> $\hat{\gamma}_g \in \mathcal{P}(\Omega)$ *is a finite mixture of Diracs.*

In this setting, the Jensen-Shannon and χ^2-divergence are constant since $\hat{\alpha}_g$ and $\hat{\beta}$ generally do not have the same support. This is the theoretical reason given by [10] to introduce new losses, such as in WGAN [12]. However, this is inconsistent with empirical results showing that GANs can be trained even without the latter losses.

Actually, in the alternating optimization setting used in practice as in Equation (5.3), the constancy of $\mathcal{L}_{\hat{\alpha}_g}$, or even of $\mathcal{C}_{f^\star_{\alpha_g}}$, does not imply that $\nabla_x c_{f^\star_{\alpha_g}}$ in Equation (5.3) is zero on these points. This stems from the gradient of Equation (5.3) ignoring the dependency of the optimal discriminator on the generator's parameters: while $\nabla_\theta \mathcal{C}_{f^\star_{\alpha_g}}\left(\alpha_g\right)$ might be null, the gradient of Equation (5.3) differs and may not be zero, thereby changing the actual loss optimized by the generator. This fact is unaccounted for in many prior analyses, like the ones of [12] and [16]. We refer to Sections 5.6.2 and 5.8.1 for further discussion.

Yet, in the previous theoretical frameworks where the discriminator can take arbitrary values, this gradient field is not even defined for any loss $\mathcal{L}_{\hat{\alpha}_g}$. Indeed, when the discriminator's loss $\mathcal{L}_{\hat{\alpha}_g}(f)$ is only computed on the empirical distribution $\hat{\gamma}_g$ (as it is the case for most GAN formulations), the discriminator optimization problem of Equation (5.2) never yields a unique optimal solution outside $\hat{\gamma}_g$. This is illustrated by the following straightforward result.

Proposition 5.1 (*Ill-Posed Problem in* $L^2(\Omega)$)

Suppose that $\mathcal{F} = L^2(\Omega)$, $\operatorname{supp} \hat{\gamma}_g \subsetneq \Omega$. *Then, for all* $f, h \in \mathcal{F}$ *coinciding over* $\operatorname{supp} \hat{\gamma}_g$, $\mathcal{L}_{\hat{\alpha}_g}(f) = \mathcal{L}_{\hat{\alpha}_g}(h)$ *and Equation* (5.2) *has either no or infinitely many optimal solutions in* $\mathcal{F}$, *all coinciding over* $\operatorname{supp} \hat{\gamma}_g$.

In particular, the set of solutions, if non-empty, contains non-differentiable discriminators as well as discriminators with null or non-informative gradients. This underspecification of the discriminator over Ω makes the gradient of the optimal discriminator in standard GAN analyses ill-defined.

This signifies that the loss alone does not impose any constraint on the values that $f_{\hat{\alpha}_g}$ takes outside $\operatorname{supp} \hat{\gamma}_g$, and more particularly that there are no constraints on the gradients. Therefore, an analysis beyond the loss function is necessary to precisely determine the learning problem of the generator defined by the discriminator.

5.4 A NTK framework for GANs

To tackle the aforementioned issues, we notice that, in practice, the inner optimization problem of Equation (5.2) is not solved exactly. Instead, using alternating optimization, a proxy neural discriminator is trained using several steps of gradient ascent for each generator update [101]. For a learning rate ε and a fixed generator g, this results the optimization procedure, from $i = 0$ to N:

$$\vartheta^g_{i+1} = \vartheta^g_i + \varepsilon \nabla_\vartheta \mathcal{L}_{\hat{\alpha}_g}\big(f_{\vartheta_i}\big), \qquad f^\star_{\hat{\alpha}_g} = f_{\vartheta^g_N}. \tag{5.4}$$

This parameterization and training of the discriminator as a neural network solve the gradient indeterminacy of the previous section, but make a theoretical analysis of its impact unattainable. We propose to facilitate it by using the theory of NTKs, which we briefly introduce in Section 5.4.1, in order to build a more consistent theory of GAN discriminator training. We then leverage those results for the discriminator to analyze the dynamics of the generated distribution via the generator's NTK in Section 5.6.

5.4.1 From Finite to Infinite-Width Networks

We study the continuous-time version of Equation (5.4):

$$\partial_t \vartheta^g_t = \nabla_\vartheta \mathcal{L}_{\hat{\alpha}_g}\big(f_{\vartheta^g_t}\big), \tag{5.5}$$

which we consider in the infinite-width limit of the discriminator, making its analysis more tractable.

In the limit where the width of the hidden layers of $f_t \triangleq f_{\vartheta^g_t}$ tends to infinity, [128] showed that its so-called NTK $k_{\vartheta^g_t}$ remains constant during a gradient ascent such as Equation (5.5), i.e. there is a limiting kernel k such that:

$$\begin{aligned} &\forall \tau \in \mathbb{R}_+, \ \ \forall x, y \in \mathbb{R}^n, \ \ \forall t \in [0, \tau], \\ &\quad k_{\vartheta^g_t}(x, y) \ \triangleq \ \partial_\vartheta f_t(x)^\top \partial_\vartheta f_t(y) \ = \ k(x, y). \end{aligned} \tag{5.6}$$

In particular, k only depends on the architecture of f and the initialization distribution of its parameters. The constancy of the NTK of f_t during gradient descent holds for many standard architectures, typically without bottleneck and ending with a linear layer [162], which is the case of most standard discriminators in the setting of Equation (5.2).

Indeed, the constancy of the neural tangent kernel during training when the width of the network becomes increasingly large is broadly applicable. As summarized by [162], typical neural networks with the building blocks of multilayer perceptrons and convolutional neural networks comply with this property, as long as they end with a linear layer and do not have any bottleneck, this constancy

needsing the minimum internal width to grow unbounded [14]. This includes, for example, residual convolutional neural networks [117]. The requirement of a final linear activation can be circumvented by transferring this activation into the loss function, as we did for the original GAN formulation in Section 5.3. Our framework thus encompasses a wide range of discriminator architectures and many building blocks of state-of-the-art discriminators can be studied in this infinite-width regime with a constant NTK, as highlighted by the exhaustiveness of the Neural Tangents library [204].

In particular, assumptions about the used activation functions are mild and include many standard activations such as ReLU, sigmoid and tanh. Beyond fully connected linear layers and convolutions, NTK constancy also includes typical operations such as self-attention [120], layer normalization and batch normalization [281]. This variety of networks supports the generality of our approach, as it includes powerful discriminator architectures such as BigGAN [43].

Moreover, let us emphasize the fact that the NTK of the discriminator remains constant throughout the whole GAN optimization process, and not only under a fixed generator. Indeed, if it remains constant in-between generator updates, then it also remains constant when the generator changes. This is because, for a finite training time, the constancy of the NTK solely depends on the network architecture and initialization, regardless of the training loss which is the only aspect affected by the generator's change and may thus change in the course of training without affecting the NTK.

There are nevertheless some limits to the NTK approximation, as we are not aware of works studying the application of the infinite-width regime to some operations such as spectral normalization, and networks in the regime of a constant NTK cannot perform feature learning as they are equivalent to kernel methods [98, 282]. However, our framework remains general and constitutes, in our knowledge, the most advanced attempt at theoretically modeling the discriminator's architecture in GANs.

5.4.2 Modeling Inductive Biases of the Discriminator in the Infinite-Width Limit

Let us start by the following assumption regarding the constant NTK kernel we defined in Section 5.4.1.

Assumption 5.2 (*Kernel*)

$k\colon \Omega^2 \to \mathbb{R}$ is a symmetric positive semi-definite kernel with $k \in L^2\big(\Omega^2\big)$.

The constancy of the NTK simplifies the dynamics of training in the functional space. However, in order to express these dynamics, we must first introduce some preliminary definitions.

Definition 5.1 (*Functional gradient*)

Whenever a functional $\mathcal{L}\colon L^2(\mu) \to \mathbb{R}$ has sufficient regularity, its gradient with respect to μ evaluated at $f \in L^2(\mu)$ is defined in the usual way as the element $\nabla^\mu \mathcal{L}(f) \in L^2(\mu)$ such that for all $\psi \in L^2(\mu)$:

$$\lim_{\varepsilon \to 0} \frac{1}{\varepsilon}(\mathcal{L}(f + \varepsilon\psi) - \mathcal{L}(f)) = \langle \nabla^\mu \mathcal{L}(f), \psi \rangle_{L^2(\mu)}. \tag{5.7}$$

Definition 5.2 (*RKHS w.r.t. μ and kernel integral operator [250]*)

If k follows Assumption 5.2 and $\mu \in \mathcal{P}(\Omega)$ is a finite mixture of Diracs, we define the Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}_k^\mu$ of k with respect to μ given by the Moore–Aronszajn theorem as the linear span of functions $k(x, \cdot)$ for $x \in \operatorname{supp}\mu$. Its kernel integral operator from Mercer's theorem is defined as:

$$\mathcal{T}_{k,\mu}\colon L^2(\mu) \to \mathcal{H}_k^\mu, \qquad h \mapsto \int_x k(\cdot, x)h(x)\mathrm{d}\mu(x). \tag{5.8}$$

Note that $\mathcal{T}_{k,\mu}$ generates $\mathcal{H}_k^\mu$, and elements of $\mathcal{H}_k^\mu$ are functions defined over all Ω as $\mathcal{H}_k^\mu \subseteq L^2(\Omega)$.

The results of [128] imply that the infinite-width discriminator f_t trained by Equation (5.5) obeys the following differential equation in-between generator updates:

$$\partial_t f_t = \mathcal{T}_{k,\hat{\gamma}_g}\Big(\nabla^{\hat{\gamma}_g}\mathcal{L}_{\hat{\alpha}}(f_t)\Big). \tag{5.9}$$

Within the alternating optimization of GANs at generator step j, f_0 would correspond to the previous discriminator step $f^\star_{\alpha_{g_{\theta_j}}} \triangleq f^j$, and $f^{j+1} = f_\tau$, with τ being the training time of the discriminator in-between generator updates.

In the following, we rely on this differential equation to gain a better understanding of the discriminator during training as well as its implications for training the generator.

5.5 Analysis of the Discriminator: Characterization and Differentiability

Starting from Equation (5.9), we first prove, under mild assumptions on the discriminator loss function, that Equation (5.9) admits a unique solution for a given initial condition. We then study the differentiability of neural networks in this regime, a necessary condition for the trainability of GANs. Note that these results are not specific to GANs but generalize to networks trained under empirical losses such as Equation (5.2), e.g. for binary classification and regression. Moreover, while results are presented in the context of a discrete distribution $\hat{\gamma}_g$, they are generalizable to broader distributions.

5.5.1 Existence, Uniqueness and Characterization of the Discriminator

The following gives a general result on the existence and uniqueness of the discriminator which then allows to characterize its general form, amenable to theoretical analysis.

We start by stating the results before delving into the complete proofs.

- *Statement of the results*

Assumption 5.3 (*Loss regularity*)

a and b from Equation (5.2) are differentiable with Lipschitz derivatives over $\mathbb{R}$.

Theorem 5.1 (*Solution of gradient descent*)

Under Assumptions 5.1 to 5.3, Equation (5.9) with initial value $f_0 \in L^2(\Omega)$ admits a unique solution $f_\cdot : \mathbb{R}_+ \to L^2(\Omega)$. Moreover, the following holds for all $t \in \mathbb{R}_+$:

$$\begin{aligned} \forall t \in \mathbb{R}_+, f_t &= f_0 + \int_0^t \mathcal{T}_{k,\hat{\gamma}_g}\Big(\nabla^{\hat{\gamma}_g}\mathcal{L}_{\hat{\alpha}_g}(f_s)\Big)\mathrm{d}s \\ &= f_0 + \mathcal{T}_{k,\hat{\gamma}_g}\left(\int_0^t \nabla^{\hat{\gamma}_g}\mathcal{L}_{\hat{\alpha}_g}(f_s)\mathrm{d}s\right). \end{aligned} \tag{5.10}$$

Because, for any given training time t, there exists a unique $f_t \in L^2(\Omega)$, defined over all of Ω and not only the training set, the aforementioned issue in Section 5.3.2 of determining the discriminator associated to $\hat{\gamma}_g$ is now resolved. It is now possible to study the discriminator in its general form thanks to Equation (5.10). It involves two terms: the previous discriminator state $f_0 = f^j$, as well as the kernel operator of an integral. This integral is a function that is undefined outside $\operatorname{supp}\hat{\gamma}_g$, as by definition $\nabla^{\hat{\gamma}_g}\mathcal{L}_{\hat{\alpha}_g}(f_s) \in L^2\big(\hat{\gamma}_g\big)$. Fortunately, the kernel operator behaves like a smoothing operator, as it not only defines the function on all of Ω but also embeds it in a highly structured space.

Corollary 5.1 (*Training and RKHS*)

Under Assumptions 5.1 to 5.3, $f_t - f_0$ belongs to the RKHS $\mathcal{H}_k^{\hat{\gamma}_g}$ for all $t \in \mathbb{R}_+$.

In our setting, this space is generated from the NTK k, which only depends on the discriminator architecture, and not on the loss function. This highlights the crucial role of the discriminator's implicit biases, and enables us to characterize its regularity for a given architecture.

• *Proofs*

The methods used in this section are adaptations to our setting of standard methods of proof. In particular, they can be easily adapted to slightly different contexts, the main ingredient being the structure of the kernel integral operator. Moreover, it is also worth noting that, although we relied on Assumption 5.1 for $\hat{\gamma}$, the results are essentially unchanged if we take a compactly supported measure γ instead.

We decompose the proof into several intermediate results. Propositions 5.2 and 5.3, stated and demonstrated in this section, correspond when combined to Theorem 5.1.

Let us first prove the following two intermediate lemmas.

Lemma 5.1

Let $\delta T > 0$ and $\mathcal{F}_{\delta T} = \mathcal{C}\Big([0, \delta T], B_{L^2(\hat{\gamma})}(f_0, 1)\Big)$ endowed with the norm:

$$\forall u \in \mathcal{F}_{\delta T},\ \|u\| = \sup_{t \in [0, \delta T]} \|u_t\|_{L^2(\hat{\gamma})}. \tag{5.11}$$

Then $\mathcal{F}_{\delta T}$ is complete.

Proof: Let $(u^n)_n$ be a Cauchy sequence in $\mathcal{F}_{\delta T}$. For a fixed $t \in [0, \delta T]$:

$$\forall n, m, \|u_t^n - u_t^m\|_{L^2(\hat{\gamma})} \leq \|u^n - u^m\|, \tag{5.12}$$

which shows that $(u_t^n)_n$ is a Cauchy sequence in $L^2(\hat{\gamma})$. $L^2(\hat{\gamma})$ being complete, $(u_t^n)_n$ converges to a $u_t^\infty \in L^2(\hat{\gamma})$. Moreover, for $\varepsilon > 0$, because (u^n) is Cauchy, we can choose N such that:

$$\forall n, m \geq N, \|u^n - u^m\| \leq \varepsilon. \tag{5.13}$$

We thus have that:

$$\forall t, \forall n, m \geq N, \|u_t^n - u_t^m\|_{L^2(\hat{\gamma})} \leq \varepsilon. \tag{5.14}$$

Then, by taking m to ∞, by continuity of the $L^2(\hat{\gamma})$ norm:

$$\forall t, \forall n \geq N, \|u_t^n - u_t^\infty\|_{L^2(\hat{\gamma})} \leq \varepsilon, \tag{5.15}$$

which means that:

$$\forall n \geq N, \|u^n - u^\infty\| \leq \varepsilon. \tag{5.16}$$

so that $(u^n)_n$ tends to u^∞.

Moreover, as:

$$\forall n, \|u_t^n\|_{L^2(\hat{\gamma})} \leq 1, \tag{5.17}$$

we have that $\|u_t^\infty\|_{L^2(\hat{\gamma})} \leq 1$.

Finally, let us consider $s, t \in [0, \delta T]$. We have that:

$$\forall n, \|u_t^\infty - u_s^\infty\|_{L^2(\hat{\gamma})} \leq \|u_t^\infty - u_t^n\|_{L^2(\hat{\gamma})} + \|u_t^n - u_s^n\|_{L^2(\hat{\gamma})} + \|u_s^\infty - u_s^n\|_{L^2(\hat{\gamma})}. \tag{5.18}$$

The first and the third terms can then be taken as small as needed by definition of u^∞ by taking n high enough, while the second can be made to tend to 0 as t tends to s by continuity of u^n. This proves the continuity of u^∞ and shows that $u^\infty \in \mathcal{F}_{\delta T}$. □

Lemma 5.2

For any $F \in L^2(\hat{\gamma})$, we have that $F \in L^2(\hat{\alpha})$ and $F \in L^2\big(\hat{\beta}\big)$ with:

$$\|F\|_{L^2(\hat{\alpha})} \leq \sqrt{2}\|F\|_{L^2(\hat{\gamma})} \text{ and } \|F\|_{L^2\left(\hat{\beta}\right)} \leq \sqrt{2}\|F\|_{L^2(\hat{\gamma})}. \tag{5.19}$$

Proof: For any $F \in L^2(\hat{\gamma})$, we have that

$$\|F\|^2_{L^2(\hat{\gamma})} = \frac{1}{2}\|F\|^2_{L^2(\hat{\alpha})} + \frac{1}{2}\|F\|^2_{L^2\left(\hat{\beta}\right)}, \tag{5.20}$$

so that $F \in L^2(\hat{\alpha})$ and $F \in L^2\big(\hat{\beta}\big)$ with:

$$\|F\|^2_{L^2(\hat{\alpha})} = 2\|F\|^2_{L^2(\hat{\gamma})} - \|F\|^2_{L^2\left(\hat{\beta}\right)} \leq 2\|F\|^2_{L^2(\hat{\gamma})}, \quad \|F\|^2_{L^2\left(\hat{\beta}\right)} = 2\|F\|_{L^2(\hat{\gamma})} - \|F\|_{L^2(\hat{\alpha})} \leq 2\|F\|^2_{L^2(\hat{\gamma})}, \tag{5.21}$$

which allows us to conclude. □

From this, we can prove the existence and uniqueness of the initial value problem from Equation (5.9).

Proposition 5.2 (*Existence and Uniqueness*)

Under Assumptions 5.1 to 5.3, Equation (5.9) *with initial value f_0 admits a unique solution $f_\cdot : \mathbb{R}_+ \to L^2(\Omega)$.*

Proof:

A few inequalities. We start this proof by proving a few inequalities.

Let $f, g \in L^2(\hat{\gamma})$. We have, by the Cauchy-Schwarz inequality, for all $z \in \Omega$:

$$\left|\Big(\mathcal{T}_{k,\hat{\gamma}}\Big(\nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(f)\Big) - \mathcal{T}_{k,\hat{\gamma}}\Big(\nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(g)\Big)\Big)(z)\right| \leq \|k(z,\cdot)\|_{L^2(\hat{\gamma})}\left\|\nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(f) - \nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(g)\right\|_{L^2(\hat{\gamma})}. \tag{5.22}$$

Moreover, by definition:

$$\Big\langle \nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(f) - \nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(g), h\Big\rangle_{L^2(\hat{\gamma})} = \int \Big(a'_f - a'_g\Big)h\mathrm{d}\hat{\alpha} - \int \Big(b'_f - b'_g\Big)h\mathrm{d}\hat{\beta}, \tag{5.23}$$

so that:

$$\left\|\nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(f) - \nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(g)\right\|^2_{L^2(\hat{\gamma})} \leq \left\|\nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(f) - \nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(g)\right\|_{L^2(\hat{\gamma})}\left(\left\|a'_f - a'_g\right\|_{L^2(\hat{\alpha})} + \left\|b'_f - b'_g\right\|_{L^2\left(\hat{\beta}\right)}\right), \tag{5.24}$$

and then, along with Lemma 5.2:

$$\left\|\nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(f) - \nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(g)\right\|_{L^2(\hat{\gamma})} \leq \left\|a'_f - a'_g\right\|_{L^2(\hat{\alpha})} + \left\|b'_f - b'_g\right\|_{L^2\left(\hat{\beta}\right)} \leq \sqrt{2}\Big(\left\|a'_f - a'_g\right\|_{L^2(\hat{\gamma})} + \left\|b'_f - b'_g\right\|_{L^2(\hat{\gamma})}\Big). \tag{5.25}$$

By Assumption 5.3, we know that a' and b' are Lipschitz with constants that we denote K_1 and K_2. We can then write for all x:

$$\Big|a'(f(x)) - a'(g(x))\Big| \leq K_1|f(x) - g(x)|, \qquad \Big|b'(f(x)) - b'(g(x))\Big| \leq K_2|f(x) - g(x)|, \tag{5.26}$$

so that:

$$\left\|a'_f - a'_g\right\|_{L^2(\hat{\gamma})} \leq K_1\|f-g\|_{L^2(\hat{\gamma})}, \qquad \left\|b'_f - b'_g\right\|_{L^2(\hat{\gamma})} \leq K_2\|f-g\|_{L^2(\hat{\gamma})}. \tag{5.27}$$

Finally, we can now write, for all $z \in \Omega$:

$$\left|\left(\mathcal{T}_{k,\hat{\gamma}}\left(\nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(f)\right) - \mathcal{T}_{k,\hat{\gamma}}\left(\nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(g)\right)\right)(z)\right| \leq \sqrt{2}(K_1+K_2)\|f-g\|_{L^2(\hat{\gamma})}\|k(z,\cdot)\|_{L^2(\hat{\gamma})}, \tag{A}$$

and then:

$$\left\|\mathcal{T}_{k,\hat{\gamma}}\left(\nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(f)\right) - \mathcal{T}_{k,\hat{\gamma}}\left(\nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(g)\right)\right\|_{L^2(\hat{\gamma})} \leq K\|f-g\|_{L^2(\hat{\gamma})}, \tag{B}$$

where $K = \sqrt{2}(K_1+K_2)\sqrt{\int\|k(z,\cdot)\|^2_{L^2(\hat{\gamma})}\mathrm{d}\hat{\gamma}(z)}$ is finite as a finite sum of finite terms from Assumptions 5.1 and 5.2. In particular, putting $g=0$ and using the triangular inequality also gives us:

$$\left\|\mathcal{T}_{k,\hat{\gamma}}\left(\nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(f)\right)\right\|_{L^2(\hat{\gamma})} \leq K\|f\|_{L^2(\hat{\gamma})} + M, \tag{B'}$$

where $M = \left\|\mathcal{T}_{k,\hat{\gamma}}\left(\nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(0)\right)\right\|_{L^2(\hat{\gamma})}$.

Existence and uniqueness in $L^2(\hat{\gamma})$. We now adapt the standard fixed point proof to prove existence and uniqueness of a solution to the studied equation in $L^2(\hat{\gamma})$.

We consider the family of spaces $\mathcal{F}_{\delta T} = \mathcal{C}\Big([0,\delta T], B_{L^2(\hat{\gamma})}(f_0,1)\Big)$. $\mathcal{F}_{\delta T}$ is defined, for $\delta T > 0$, as the space of continuous functions from $[0,\delta T]$ to the closed ball of radius 1 centered around f_0 in $L^2(\hat{\gamma})$ which we endow with the norm:

$$\forall u \in \mathcal{F}_{\delta T},\ \|u\| = \sup_{t\in[0,\delta T]}\|u_t\|_{L^2(\hat{\gamma})}. \tag{5.28}$$

We now define the application Φ where $\Phi(u)$ is defined as, for any $u \in \mathcal{F}_{\delta T}$:

$$\Phi(u)_t = f_0 + \int_0^t \mathcal{T}_{k,\hat{\gamma}}\left(\nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(u_s)\right)\mathrm{d}s. \tag{5.29}$$

We have, using Equation (B'):

$$\|\Phi(u)_t - f_0\|_{L^2(\hat{\gamma})} \leq \int_0^t \Big(K\|u_s\|_{L^2(\hat{\gamma})} + M\Big)\mathrm{d}s \leq (K+M)\delta T. \tag{5.30}$$

Thus, taking $\delta T = \big(2(K+M)\big)^{-1}$ makes Φ an application from $\mathcal{F}_{\delta T}$ into itself. Moreover, we have:

$$\forall u, v \in \mathcal{F}_{\delta T}, \|\Phi(u) - \Phi(v)\| \leq \frac{1}{2}\|u-v\|, \tag{5.31}$$

which means that Φ is a contraction of $\mathcal{F}_{\delta T}$. Lemma 5.1 and the Banach-Picard theorem then tell us that Φ has a unique fixed point in $\mathcal{F}_{\delta T}$. It is then obvious that such a fixed point is a solution of Equation (5.9) over $[0,\delta T]$.

Let us now consider the maximal $T > 0$ such that a solution f_t of Equation (5.9) is defined over $[0,T)$. We have, using Equation (B'):

$$\forall t \in [0,T), \|f_t\|_{L^2(\hat{\gamma})} \leq \|f_0\|_{L^2(\hat{\gamma})} + \int_0^t \Big(\|f_s\|_{L^2(\hat{\gamma})} + M\Big)\mathrm{d}s, \tag{5.32}$$

which, using Grönwall's lemma, gives:

$$\forall t \in [0,T), \|f_t\|_{L^2(\hat{\gamma})} \leq \|f_0\|_{L^2(\hat{\gamma})} \mathrm{e}^{KT} + \frac{M}{K}\Big(\mathrm{e}^{KT} - 1\Big). \tag{5.33}$$

Define $g^n = f_{T-\frac{1}{n}}$. We have, again using Equation (B'):

$$\forall m \geq n, \|g^n - g^m\|_{L^2(\hat{\gamma})} \leq \int_{T-\frac{1}{n}}^{T-\frac{1}{m}} (K\|f_s\| + M)\mathrm{d}s \leq \left(\frac{1}{n} - \frac{1}{m}\right)\left(\|f_0\|_{L^2(\hat{\gamma})}\mathrm{e}^{KT} + \frac{M}{K}\Big(\mathrm{e}^{KT} - 1\Big)\right), \tag{5.34}$$

which shows that $(g^n)_n$ is a Cauchy sequence. $L^2(\hat{\gamma})$ being complete, we can thus consider its limit g^∞. Clearly, f_t tends to g^∞ in $L^2(\hat{\gamma})$. By considering the initial value problem associated with Equation (5.9) starting from g^∞, we can thus extend the solution f_t to $[0, T + \delta T)$, thus contradicting the maximality of T, which proves that the solution can be extended to $\mathbb{R}_+$.

Existence and uniqueness in $L^2(\Omega)$. We now conclude the proof by extending the previous solution to $L^2(\Omega)$. We keep the same notations as above and, in particular, f is the unique solution of Equation (5.9) with initial value f_0.

Let us define $\tilde{f}$ as:

$$\forall t, \forall x, \tilde{f}_t(x) = f_0(x) + \int_0^t \mathcal{T}_{k,\hat{\gamma}}\Big(\nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(f_s)\Big)(x)\mathrm{d}s, \tag{5.35}$$

where the r.h.s. only depends on f and is thus well-defined. By remarking that $\tilde{f}$ is equal to f on $\operatorname{supp}\hat{\gamma}$ and that, for every s,

$$\mathcal{T}_{k,\hat{\gamma}}\left(\nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}\Big(\tilde{f}_s\Big)\right) = \mathcal{T}_{k,\hat{\gamma}}\left(\nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}\Big(\tilde{f}_s\Big|_{\operatorname{supp}\hat{\gamma}}\Big)\right) = \mathcal{T}_{k,\hat{\gamma}}\Big(\nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(f_s)\Big), \tag{5.36}$$

we see that $\tilde{f}$ is solution to Equation (5.9). Moreover, from Assumption 5.2, we know that, for any $z \in \Omega$, $\int k(z,x)^2 \mathrm{d}\Omega(x)$ is finite and, from Assumption 5.1, that $\|k(z,\cdot)\|^2_{L^2(\hat{\gamma})}$ is a finite sum of terms $k(z,x_i)^2$ which shows that $\int \|k(z,\cdot)\|^2_{L^2(\hat{\gamma})}\mathrm{d}\Omega(z)$ is finite, again from Assumption 5.2. We can then say that $\tilde{f}_s \in L^2(\Omega)$ for any s by using the above with Equation (A) taken for $g = 0$.

Finally, suppose h is a solution to Equation (5.9) with initial value f_0. We know that $h|_{\operatorname{supp}\hat{\gamma}}$ coincides with f and thus with $\tilde{f}\Big|_{\operatorname{supp}\hat{\gamma}}$ in $L^2(\hat{\gamma})$ as we already proved uniqueness in the latter space. Thus, we have that $\left\| h_s|_{\operatorname{supp}\hat{\gamma}} - \tilde{f}_s\Big|_{\operatorname{supp}\hat{\gamma}} \right\|_{L^2(\hat{\gamma})} = 0$ for any s. Now, we have:

$$\begin{aligned} &\forall z \in \Omega, \forall s, \left| \left(\mathcal{T}_{k,\hat{\gamma}}\Big(\nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(h_s)\Big) - \mathcal{T}_{k,\hat{\gamma}}\left(\nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}\Big(\tilde{f}_s\Big)\right)\right)(z)\right| \\ &\quad = \left| \left(\mathcal{T}_{k,\hat{\gamma}}\left(\nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}\Big(h_s|_{\operatorname{supp}\hat{\gamma}}\Big)\right) - \mathcal{T}_{k,\hat{\gamma}}\left(\nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}\Big(\tilde{f}_s\Big|_{\operatorname{supp}\hat{\gamma}}\Big)\right)\right)(z)\right| \leq 0, \end{aligned} \tag{5.37}$$

by Equation (A). This shows that $\partial_t\Big(\tilde{f} - h\Big) = 0$ and, given that $h_0 = \tilde{f}_0 = f_0$, we have $h = \tilde{f}$ which concludes the proof.

□

There only remains to prove for Theorem 5.1 the inversion between the integral over time and the integral operator. We first prove an intermediate lemma and then conclude with the proof of the inversion.

Lemma 5.3

Under Assumptions 5.1 to 5.3, $\int_0^T \Big(\|a'\|_{L^2\left((f_s)_\sharp \hat{\alpha}\right)} + \|b'\|_{L^2\left((f_s)_\sharp \hat{\beta}\right)} \Big) \mathrm{d}s$ is finite for any $T > 0$.

Proof: Let $T > 0$. We have, by Assumption 5.3 and the triangular inequality:

$$\forall x, \Big| a'(f(x)) \Big| \leq K_1 |f(x)| + M_1, \tag{5.38}$$

where $M_1 = |a'(0)|$. We can then write, using Lemma 5.2 and the inequality from Equation (5.33):

$$\forall s \leq T, \Big\| a' \Big\|_{L^2\left((f_s)_\sharp \hat{\alpha}\right)} \leq K_1 \sqrt{2} \|f_s\|_{L^2(\hat{\gamma})} + M_1 \leq K_1 \sqrt{2} \left(\|f_0\|_{L^2(\hat{\gamma})} \mathrm{e}^{KT} + \frac{M}{K} \Big(\mathrm{e}^{KT} - 1 \Big) \right) + M_1, \tag{5.39}$$

the latter being constant in s and thus integrable on $[0, T]$. We can then bound $\|b'\|_{L^2\left((f_s)_\sharp \hat{\beta}\right)}$ similarly, which concludes the proof. □

Proposition 5.3 (*Integral inversion*)

Under Assumptions 5.1 to 5.3, the following integral inversion holds:

$$f_t = f_0 + \int_0^t \mathcal{T}_{k_f, \hat{\gamma}} \Big(\nabla^{\hat{\gamma}} \mathcal{L}_{\hat{\alpha}, \hat{\beta}}(f_s) \Big) \mathrm{d}s = f_0 + \mathcal{T}_{k_f, \hat{\gamma}} \left(\int_0^t \nabla^{\hat{\gamma}} \mathcal{L}_{\hat{\alpha}, \hat{\beta}}(f_s) \mathrm{d}s \right). \tag{5.40}$$

Proof: By definition, a straightforward computation gives, for any function $h \in L^2(\hat{\gamma})$:

$$\Big\langle \nabla^{\hat{\gamma}} \mathcal{L}_{\hat{\alpha}}(f), h \Big\rangle_{L^2(\hat{\gamma})} = \mathrm{d}\mathcal{L}_{\hat{\alpha}}(f)[h] = \int a'_f h \mathrm{d}\hat{\alpha} - \int b'_f h \mathrm{d}\hat{\beta}. \tag{5.41}$$

We can then write:

$$\Big\| \nabla^{\hat{\gamma}} \mathcal{L}_{\hat{\alpha}}(f_t) \Big\|^2_{L^2(\hat{\gamma})} = \Big\langle \nabla^{\hat{\gamma}} \mathcal{L}_{\hat{\alpha}}(f_t), \nabla^{\hat{\gamma}} \mathcal{L}_{\hat{\alpha}}(f_t) \Big\rangle_{L^2(\hat{\gamma})} = \int a'_{f_t} \nabla^{\hat{\gamma}} \mathcal{L}_{\hat{\alpha}}(f_t) \mathrm{d}\hat{\alpha} - \int b'_{f_t} \nabla^{\hat{\gamma}} \mathcal{L}_{\hat{\alpha}}(f_t) \mathrm{d}\hat{\beta}, \tag{5.42}$$

so that, with the Cauchy-Schwarz inequality and Lemma 5.2:

$$\begin{aligned} \Big\| \nabla^{\hat{\gamma}} \mathcal{L}_{\hat{\alpha}}(f_t) \Big\|^2_{L^2(\hat{\gamma})} &\leq \int \Big| a'_{f_t} \Big| \Big| \nabla^{\hat{\gamma}} \mathcal{L}_{\hat{\alpha}}(f_t) \Big| \mathrm{d}\hat{\alpha} + \int \Big| b'_{f_t} \Big| \Big| \nabla^{\hat{\gamma}} \mathcal{L}_{\hat{\alpha}}(f_t) \Big| \mathrm{d}\hat{\beta} \\ &\leq \Big\| a'_{f_t} \Big\|_{L^2(\hat{\alpha})} \Big\| \nabla^{\hat{\gamma}} \mathcal{L}_{\hat{\alpha}}(f_t) \Big\|_{L^2(\hat{\alpha})} + \Big\| b'_{f_t} \Big\|_{L^2\left(\hat{\beta}\right)} \Big\| \nabla^{\hat{\gamma}} \mathcal{L}_{\hat{\alpha}}(f_t) \Big\|_{L^2\left(\hat{\beta}\right)} \\ &\leq \sqrt{2} \Big\| \nabla^{\hat{\gamma}} \mathcal{L}_{\hat{\alpha}}(f_t) \Big\|_{L^2(\hat{\gamma})} \left[\Big\| a'_{f_t} \Big\|_{L^2(\hat{\alpha})} + \Big\| b'_{f_t} \Big\|_{L^2\left(\hat{\beta}\right)} \right], \end{aligned} \tag{5.43}$$

which then gives us:

$$\Big\| \nabla^{\hat{\gamma}} \mathcal{L}_{\hat{\alpha}}(f_t) \Big\|_{L^2(\hat{\gamma})} \leq \sqrt{2} \left[\Big\| a' \Big\|_{L^2\left((f_t)_\sharp \hat{\alpha}\right)} + \Big\| b' \Big\|_{L^2\left((f_t)_\sharp \hat{\beta}\right)} \right]. \tag{5.44}$$

By the Cauchy-Schwarz inequality and Equation (5.44), we then have for all z:

$$\begin{aligned} \int_0^t \int_x \Big| k(z, x) \nabla^{\hat{\gamma}} \mathcal{L}_{\hat{\alpha}}(f_s)(x) \Big| \mathrm{d}\hat{\gamma}(x) \mathrm{d}s &\leq \int_0^t \|k(z, \cdot)\|_{L^2(\hat{\gamma})} \Big\| \nabla^{\hat{\gamma}} \mathcal{L}_{\hat{\alpha}}(f_s) \Big\|_{L^2(\hat{\gamma})} \mathrm{d}s \\ &\leq \sqrt{2} \|k(z, \cdot)\|_{L^2(\hat{\gamma})} \int_0^t \left[\Big\| a' \Big\|_{L^2\left((f_s)_\sharp \hat{\alpha}\right)} + \Big\| b' \Big\|_{L^2\left((f_s)_\sharp \hat{\beta}\right)} \right] \mathrm{d}s. \end{aligned} \tag{5.45}$$

The latter being finite by Lemma 5.3, we can now use Fubini's theorem to conclude that:

$$\begin{aligned}\int_0^t \mathcal{T}_{k_f,\hat{\gamma}}\Big(\nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(f_s)\Big)\mathrm{d}s &= \int_0^t \int_x k(\cdot,x)\nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(f_s)(x)\mathrm{d}\hat{\gamma}(x)\mathrm{d}s \\ &= \int_x k(\cdot,x)\left[\int_0^t \nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(f_s)(x)\mathrm{d}s\right]\mathrm{d}\hat{\gamma}(x) \\ &= \mathcal{T}_{k_f,\hat{\gamma}}\left(\int_0^t \nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(f_s)(x)\mathrm{d}s\right).\end{aligned} \tag{5.46}$$

□

5.5.2 Differentiability of the Discriminator

Here, we study the differentiability of the discriminator. We start by giving a summary of the assumptions and corresponding results before providing detailed proofs of the latter.

• *Summary of the Differentiability Results*

We study in this section the smoothness, i.e. infinite differentiability, of the discriminator. It mostly relies on the differentiability of the kernel k, by Equation (5.10), which is obtained by characterizing the regularity of the corresponding conjugate kernel [149]. Therefore, we prove the differentiability of the NTKs of standard architectures, and then conclude about the differentiability of f_t.

Assumption 5.4 (*Discriminator architecture*)

The discriminator is a standard architecture (fully connected, convolutional or residual). Any activation ϕ in the network satisfies the following properties:

- *ϕ is smooth everywhere except on a finite set D;*
- *for all $j \in \mathbb{N}$, there exist scalars $\lambda_1^{(j)}$ and $\lambda_2^{(j)}$ such that:*

$$\forall x \in \mathbb{R} \setminus D, \left|\phi^{(j)}(x)\right| \leq \lambda_1^{(j)}|x| + \lambda_2^{(j)}, \tag{5.47}$$

where $\phi^{(j)}$ is the j-th derivative of ϕ.

Assumption 5.5 (*Discriminator regularity*)

The activation function is smooth.

Assumption 5.6 (*Discriminator bias*)

Linear layers have non-null bias terms. Moreover, for all $x, y \in \mathbb{R}$ such that $x \neq y$, the following holds:

$$\mathbb{E}_{\varepsilon\sim\mathcal{N}(0,1)}\phi(x\varepsilon)^2 \neq \mathbb{E}_{\varepsilon\sim\mathcal{N}(0,1)}\phi(y\varepsilon)^2. \tag{5.48}$$

Remark 5.1 (*Typical activations*)

Assumptions 5.4 to 5.6 cover multiple standard activation functions, including tanh*, softplus, ReLU, leaky ReLU and sigmoid.*

We first prove the differentiability of the NTK.

Proposition 5.4 (*Differentiability of k*)

Let k be the NTK of an infinite-width network from Assumption 5.4. For any $y \in \Omega$, $k(\cdot, y)$ is

smooth everywhere over Ω under Assumption 5.5, or almost everywhere if Assumption 5.6 holds instead.

From Proposition 5.4, NTKs satisfy Assumption 5.2. Using Corollary 5.1, we thus conclude on the differentiability of f_t.

Theorem 5.2 (*Differentiability of f_t*)

Suppose that k is the NTK of an infinite-width network following Assumption 5.4. Then f_t is smooth everywhere over Ω under Assumption 5.5, or almost everywhere when Assumption 5.6 holds instead.

Remark 5.2 (*Bias-free ReLU networks*)

ReLU networks with hidden layers and no bias are not differentiable at 0. However, by introducing non-zero bias, this non-differentiability at 0 disappears in the NTK and the infinite-width discriminator. This observation explains some experimental results in Section 5.7. Note that [38] state that the bias-free ReLU kernel is not Lipschitz even outside 0. However, we find this result to be incorrect. We further discuss this matter in Section 5.8.2.

This result demonstrates that, for a wide range of GANs, e.g. vanilla GAN and LSGAN, the optimized discriminator indeed admits gradients, making the gradient flow given to the generator well-defined in our framework. This supports our motivation to bring the theory closer to the empirical evidence that many GAN models do work in practice while their theoretical interpretation until now has been stating the opposite [10].

• *Detailed proofs*

Given Theorem 5.1, establishing the desired differentiability of f_t can be done by separately proving similar results on both $f_t - f_0$ and f_0.

In both cases, this involves the differentiability of the following activation kernel $\mathcal{K}_\phi(A)$ given another differentiable kernel A:

$$\mathcal{K}_\phi(A)\colon x, y \mapsto \mathbb{E}_{f\sim\mathcal{GP}(0,A)}\Big[\phi(f(x))\phi(f(y))\Big], \tag{5.49}$$

where $\mathcal{GP}(0, A)$ is a univariate centered Gaussian Process (GP) with covariance function A. Indeed, the kernel-transforming operator $\mathcal{K}_\phi$ is central in the recursive computation of the neural network conjugate kernel which determines the NTK (involved in $f_t - f_0 \in \mathcal{H}_k^{\hat{\gamma}_g}$) as well as the behavior of the network at initialization (which follows a GP with the conjugate kernel as covariance).

Hence, our proof of Theorem 5.2 relies on the preservation of kernel smoothness through $\mathcal{K}_\phi$, proved in Section 5.5.2, which ensures the smoothness of the conjugate kernel, the NTK and, in turn, of f_t as addressed in Section 5.5.2 which concludes the overall proof.

Before developing these two main steps, we first need to state the following lemma showing the regularity of samples of a GP from the regularity of the corresponding kernel.

Lemma 5.4 (*GP regularity*)

Let $A\colon \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ be a symmetric kernel. Let V an open set such that A is $\mathcal{C}^\infty$ on $V \times V$. Then the GP induced by the kernel A has a.s. $\mathcal{C}^\infty$ sample paths on V.

Proof: Because A is $\mathcal{C}^\infty$ on $V \times V$, we know, from Theorem 2.2.2 of [1] for example, that the corresponding GP f is mean-square smooth on V. If we take α a k-th order multi-index, we also know, again from [1], that $\partial^\alpha f$ is also a GP with covariance kernel $\partial^\alpha A$. As A is $\mathcal{C}^\infty$, $\partial^\alpha A$ then is differentiable and $\partial^\alpha f$ has partial derivatives which are mean-square continuous. Then, by the Corollary 5.3.12 of [237], we can say that $\partial^\alpha f$ has continuous sample paths a.s. which means that $f \in \mathcal{C}^k(V)$. This proves the lemma. □

$\mathcal{K}_\phi$ Preserves Kernel Differentiability Given the definition of $\mathcal{K}_\phi(A)$ in Equation (5.49), we choose to prove its differentiability via the dominated convergence theorem and Leibniz integral rule. This requires to derive separate proofs depending on whether ϕ is smooth everywhere or almost everywhere.

The former case allows us to apply strong GP regularity results leading to $\mathcal{K}_\phi$ preserving kernel smoothness without additional hypothesis in Lemma 5.5. The latter case requires a careful decomposition of the expectation of Equation (5.49) via two-dimensional Gaussian sampling to circumvent the non-differentiability points of ϕ, yielding additional constraints on kernels A for $\mathcal{K}_\phi$ to preserve their smoothness in Lemma 5.6; these constraints are typically verified in the case of neural networks with bias (cf. Section 5.5.2).

In any case, we emphasize that these differentiability constraints may not be tight and are only sufficient conditions ensuring the smoothness of $\mathcal{K}_\phi(A)$.

Lemma 5.5 ($\mathcal{K}_\phi$ *with smooth* ϕ)

Let $A\colon \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ be a symmetric positive semi-definite kernel and $\phi\colon \mathbb{R} \to \mathbb{R}$. We suppose that ϕ is an activation function following Assumptions 5.4 and 5.5; in particular, ϕ is smooth.

Let $y \in \mathbb{R}^n$ and U be an open subset of $\mathbb{R}^n$ such that $x \mapsto A(x,x)$ and $x \mapsto A(x,y)$ are infinitely differentiable over U. Then, $x \mapsto \mathcal{K}_\phi(A)(x,x)$ and $x \mapsto \mathcal{K}_\phi(A)(x,y)$ are infinitely differentiable over U as well.

Proof: In order to prove the smoothness results over the open set U, it suffices to prove them on any open bounded subset of U. Let then $V \subseteq U$ be an open bounded set. Without loss of generality, we can assume that its closure $\operatorname{cl} V$ is also included in U.

We define B_1 and B_2 from Equation (5.49) as follows, for all $x \in V$:

$$B_1(x) \triangleq \mathcal{K}_\phi(A)(x,y) = \mathbb{E}_{f\sim\mathcal{GP}(0,A)}\Big[\phi(f(x))\phi(f(y))\Big], \quad B_2(x) \triangleq \mathcal{K}_\phi(A)(x,x) = \mathbb{E}_{f\sim\mathcal{GP}(0,A)}\Big[\phi(f(x))^2\Big]. \tag{5.50}$$

In the previous expressions, Lemma 5.4 tells us that we can take f to be $\mathcal{C}^\infty$ over $\operatorname{cl} V$ with probability one. Hence, B_1 and B_2 are expectations of smooth functions over V. We seek to apply the dominated convergence theorem to prove that B_1 and B_2 are, in turn, smooth over V. To this end, we prove in the following the integrability of the derivatives of their integrands.

Let $\alpha = (\alpha_1, \dots, \alpha_n) \in \mathbb{N}^n$. Using the usual notations for multi-indexed partial derivatives, via a multivariate Faà di Bruno formula [153], we can write the derivatives $\partial^\alpha(\psi \circ f)$ at $x \in V$ for $\psi \in \Big\{\phi, \phi^2\Big\}$ as a weighted sum of terms of the form:

$$\psi^{(j)}(f(x))g_1(x)\cdots g_N(x), \tag{5.51}$$

where the g_is are partial derivatives of f at x. As A is $\mathcal{C}^\infty$ over V, each of the g_is is thus a GP with a $\mathcal{C}^\infty$ covariance function by Lemma 5.4. We can also write for all $x \in V$:

$$\begin{aligned}\Big|\psi^{(j)}(f(x))g_1(x)\cdots g_N(x)\Big| &\leq \sup_{z\in\operatorname{cl} V}\Big|\psi^{(j)}(f(z))g_1(z)\cdots g_N(z)\Big| \\ &\leq \sup_{z_0\in\operatorname{cl} V}\Big|\psi^{(j)}(f(z_0))\Big| \sup_{z_1\in\operatorname{cl} V}|g_1(z_1)|\cdots\sup_{z_N\in\operatorname{cl} V}|g_N(z_N)|.\end{aligned} \tag{5.52}$$

For each i, because the covariance function of g_i is smooth over the compact set $\operatorname{cl} V$, its variance admits a maximum in $\operatorname{cl} V$ and we take σ_i^2 the double of its value. We then know from [2], that there is an M_i such that:

$$\forall m \in \mathbb{N}, \mathbb{E}_{f\sim\mathcal{GP}(0,A)}\left[\sup_{z_i\in\operatorname{cl} V}|g_i(z_i)|^m\right] \leq M_i^m \mathbb{E}|Y_i|^m, \tag{5.53}$$

where Y_i is a Gaussian distribution which variance is σ_i^2, the right-hand side thus being finite.

We also have, by Assumption 5.4, that:

$$\sup_{z\in\operatorname{cl} V}\left|\phi^{(j)}(f(z))\right|^2 \leq \sup_{z\in\operatorname{cl} V}\left(\lambda_1^{(j)}|f(z)|+\lambda_2^{(j)}\right)^2, \tag{5.54}$$

which is shown to be integrable over f by the same arguments as for the g_is. Moreover, the Faà di Bruno formula decomposes $\psi^{(j)}$ when $\psi=\phi^2$ as a weighted sum of terms of the form $\phi^{(l)}\phi^{(l')}$ with $l,l'\in\mathbb{N}$. Therefore, thanks to similar arguments, for any $\psi\in\left\{\phi,\phi^2\right\}$:

$$\mathbb{E}_{f\sim\mathcal{GP}(0,A)}\left[\sup_{z\in\operatorname{cl} V}\left|\psi^{(j)}(f(z))\right|^2\right]<\infty. \tag{5.55}$$

Now, by using the Cauchy-Schwarz inequality, we have that:

$$\begin{aligned}&\mathbb{E}\left[\sup_{z_0\in\operatorname{cl} V}\left|\psi^{(j)}(f(z_0))\right|\sup_{z_1\in\operatorname{cl} V}|g_1(z_1)|\cdots\sup_{z_N\in\operatorname{cl} V}|g_N(z_N)|\right]\\&\quad\leq\sqrt{\mathbb{E}\left[\sup_{z_0\in\operatorname{cl} V}\left|\psi^{(j)}(f(z_0))\right|^2\right]}\sqrt{\mathbb{E}\left[\sup_{z_1\in\operatorname{cl} V}|g_1(z_1)|^2\cdots\sup_{z_N\in\operatorname{cl} V}|g_N(z_N)|^2\right]}.\end{aligned} \tag{5.56}$$

By iterated applications of the Cauchy-Schwarz inequality and using the previous arguments, we can then show that:

$$\sup_{z_0\in\operatorname{cl} V}\left|\psi^{(j)}(f(z_0))\right|\sup_{z_1\in\operatorname{cl} V}|g_1(z_1)|\cdots\sup_{z_N\in\operatorname{cl} V}|g_N(z_N)| \tag{5.57}$$

is integrable over f. Additionally, note that by the same arguments for the case of $\psi=\phi$, a multiplication by $\phi(f(y))$ preserves this integrability.

We can then write for all $x\in V$, by a standard corollary of the dominated convergence theorem:

$$\partial^\alpha B_1(x)=\mathbb{E}_{f\sim\mathcal{GP}(0,A)}\left[\partial^\alpha\left(\phi\circ f\right)\big|_x\phi(f(y))\right],\qquad \partial^\alpha B_2(x)=\mathbb{E}_{f\sim\mathcal{GP}(0,A)}\left[\partial^\alpha\left(\phi^2\circ f\right)\Big|_x\right], \tag{5.58}$$

which shows that B_1 and B_2 are $\mathcal{C}^\infty$ over V. This in turn means that B_1 and B_2 are $\mathcal{C}^\infty$ over U. □

Lemma 5.6 ($\mathcal{K}_\phi$ *with piecewise smooth* ϕ)

Let $A\colon\mathbb{R}^n\times\mathbb{R}^n\to\mathbb{R}$ be a symmetric positive semi-definite kernel and $\phi\colon\mathbb{R}\to\mathbb{R}$. We suppose that ϕ is an activation function following Assumptions 5.4 and 5.6. Let us define the matrix $\Sigma_A^{x,y}$ as:

$$\Sigma_A^{x,y}\triangleq\begin{pmatrix}A(x,x)&A(x,y)\\A(x,y)&A(y,y)\end{pmatrix}. \tag{5.59}$$

Let $y\in\mathbb{R}^n$ and U be an open subset of $\mathbb{R}^n$ such that $x\mapsto A(x,x)$ and $x\mapsto A(x,y)$ are infinitely differentiable over U. Then, $x\mapsto\mathcal{K}_\phi(A)(x,x)$ and $x\mapsto\mathcal{K}_\phi(A)(x,y)$ are infinitely differentiable over $U'\triangleq\left\{x\in U\;\middle|\;\Sigma_A^{x,y}\text{ is invertible}\right\}$.

Proof: Since $\det\Sigma_A^{x,y}$ is smooth over U and $U'=\left\{x\in U\;\middle|\;\det\Sigma_A^{x,y}>0\right\}$, U' is an open subset of U. Hence, similarly to the proof of Lemma 5.5, it suffices to prove the smoothness of B_1 and B_2 defined in Equation (5.50) on any open bounded subset of U'. Let then $V\subseteq\mathbb{R}^n$ be an open bounded set such that $\operatorname{cl} V\subseteq U'$. Note that $\det\Sigma_A^{x,y}>0$ implies that $A(x,x)>0$ and $A(y,y)>0$.

We will conduct in the following the proof that B_1 is smooth over V. Like in the proof of Lemma 5.5, the smoothness of B_2 follows the same reasoning with little adaptation; in particular, it relies on the fact that $A(x,x)>0$ for all $x\in U'$, making its square root smooth over $\operatorname{cl} V$.

Since the dominated convergence theorem cannot be directly applied from Equation (5.50) because of ϕ's potential non-differentiability points D, let us decompose its expression for all $x \in U'$:

$$B_1(x) = \mathbb{E}_{f\sim\mathcal{GP}(0,A)}\Big[\phi\big(f(x)\big)\phi\big(f(y)\big)\Big] = \mathbb{E}_{(z,z')\sim\mathcal{N}\left((0,0),\Sigma_A^{x,y}\right)}\Big[\phi(z)\phi\Big(z'\Big)\Big] \tag{5.60}$$

$$= \mathbb{E}_{z'\sim\mathcal{N}\left(0,A(y,y)\right)}\left[\phi\Big(z'\Big)\mathbb{E}_{z\sim\mathcal{N}\left(\frac{A(x,y)}{A(y,y)}z',A(x,x)-\frac{A(x,y)^2}{A(y,y)}\right)}\big[\phi(z)\big]\right] \tag{5.61}$$

$$= \mathbb{E}_{z'\sim\mathcal{N}\left(0,A(y,y)\right)}\Big[\phi\Big(z'\Big)h\Big(z',x\Big)\Big], \tag{5.62}$$

where h is defined as:

$$h\Big(z',x\Big) \triangleq \int_{-\infty}^{+\infty}\phi(z)\cdot\frac{1}{\sigma(x)\sqrt{2\pi}}\mathrm{e}^{-\frac{1}{2}\left(\frac{z-\mu(x)z'}{\sigma(x)}\right)^2}\mathrm{d}z, \quad \mu(x)=\frac{A(x,y)}{A(y,y)}, \quad \sigma(x)=\sqrt{\frac{\det\Sigma_A^{x,y}}{A(y,y)}}. \tag{5.63}$$

Now, if $D=\{c_1,\dots,c_L\}$ with $L\in\mathbb{N}$ and $c_1<\dots<c_L$, the c_ls constitute the non-differentiability points of ϕ; we can then decompose the integration of ϕ in Equation (5.63) as a sum of $L+1$ integrals with differentiable integrands, using $c_0=-\infty$ and $c_{L+1}=+\infty$:

$$h(\varepsilon,x) = \frac{1}{\sqrt{2\pi}}\sum_{l=0}^{L}\int_{c_l}^{c_{l+1}}\frac{\phi(z)}{\sigma(x)}\mathrm{e}^{-\frac{1}{2}\left(\frac{z-\mu(x)z'}{\sigma(x)}\right)^2}\mathrm{d}z. \tag{5.64}$$

Therefore, it remains to show the smoothness of all applications $B_{1,l}$ for $l\in[\![0,L]\!]$ defined as:

$$B_{1,j}(x) = \mathbb{E}_{z'\sim\mathcal{N}\left(0,A(y,y)\right)}\left[\int_{c_l}^{c_{l+1}}\frac{\phi(z')\phi(z)}{\sigma(x)\sqrt{2\pi}}\mathrm{e}^{-\frac{1}{2}\left(\frac{z-\mu(x)z'}{\sigma(x)}\right)^2}\mathrm{d}z\right]. \tag{5.65}$$

The rest of this proof unfolds similarly to the one of Lemma 5.5. Indeed, the integrand of Equation (5.65) is smooth over $\operatorname{cl}V$. There remains to show that all derivatives of this integrand are dominated by an integrable function of z and z'. Consider the following integrand:

$$\iota\Big(z,z',x\Big) = \frac{\phi(z')\phi(z)}{\sigma(x)\sqrt{2\pi}}\mathrm{e}^{-\frac{1}{2}\left(\frac{z-\mu(x)z'}{\sigma(x)}\right)^2}. \tag{5.66}$$

By applying the multivariate Faà di Bruno formula and noticing that σ and μ are smooth over the closed set $\operatorname{cl}V$, we know that the derivatives of $\iota(z,z',x)$ with respect to x for any derivation order are weighted sums of terms of the form:

$$z^k z'^{k'}\kappa(x)\frac{\phi(z')\phi(z)}{\sigma(x)\sqrt{2\pi}}\mathrm{e}^{-\frac{1}{2}\left(\frac{z-\mu(x)z'}{\sigma(x)}\right)^2}, \tag{5.67}$$

where κ is a smooth function over $\operatorname{cl}V$ and $k,k'\in\mathbb{N}$. Moreover, because σ, μ and κ are smooth over the closed set $\operatorname{cl}V$ with positive values for σ, there are constants a_1, a_2 and a_3 such that:

$$\left|z^k z'^{k'}\kappa(x)\frac{\phi(z')\phi(z)}{\sigma(x)\sqrt{2\pi}}\mathrm{e}^{-\frac{1}{2}\left(\frac{z-\mu(x)z'}{\sigma(x)}\right)^2}\right| \le \left|z^k z'^{k'}\phi\Big(z'\Big)\phi(z)\right|a_3\mathrm{e}^{-\frac{1}{2}\left(\frac{z-a_1z'}{a_2}\right)^2}, \tag{5.68}$$

which is integrable over z via Assumption 5.4 and Equation (5.47). Finally, let us notice that for some constants b_1, b_2 and b_3:

$$\int_{c_l}^{c_{l+1}}\left|z^k z'^{k'}\phi\Big(z'\Big)\phi(z)\right|a_3\mathrm{e}^{-\frac{1}{2}\left(\frac{z-a_1z'}{a_2}\right)^2} \le b_1\mathbb{E}_{z\sim\mathcal{N}(b_2z',b_3)}\left|z^k z'^{k'}\phi\Big(z'\Big)\phi(z)\right|, \tag{5.69}$$

which is also integrable with respect to $\mathbb{E}_{z'\sim\mathcal{N}\left(0,A(y,y)\right)}$ by similar arguments (see also the integrability of Equation (5.53) in Lemma 5.5). This concludes the proof of integrability required to apply the dominated convergence theorem, allowing us to conclude about the smoothness of all $B_{1,j}$ and, in turn, of B_1 over U'. □

Remark 5.3 (*Relaxed condition for smoothness*)

The invertibility condition of Lemma 5.6 is actually stronger than needed: it suffices to assume that the rank of $\Sigma_A^{x,y}$ remains constant in a neighborhood of x.

Differentiability of Conjugate Kernels, NTKs and Discriminators From the previous lemmas, we can then prove the results of Section 5.5.2. We start by demonstrating the smoothness of the conjugate kernel for dense networks, and conclude in consequence about the smoothness of the NTK and trained network.

Lemma 5.7 (*Differentiability of the conjugate kernel*)

Let k_c be the conjugate kernel [149] of an infinite-width dense non-residual architecture such as in Assumption 5.4. For any $y \in \mathbb{R}^n$, the following holds for $A \in \left\{k_c, \mathcal{K}_{\phi'}(k_c)\right\}$:

- *if Assumption 5.5 holds, then $x \mapsto A(x,x)$ and $x \mapsto A(x,y)$ are smooth everywhere over $\mathbb{R}^n$;*
- *if Assumption 5.6 holds, then $x \mapsto A(x,x)$ and $x \mapsto A(x,y)$ are smooth over an open set whose complement has null Lebesgue measure.*

Proof: We define the following kernel:

$$C_L^\phi(x,y) = \mathbb{E}_{f\sim\mathcal{GP}\left(0,C_{L-1}^\phi\right)}\Big[\phi(f(x))\phi(f(y))\Big] + \beta^2 = \mathcal{K}_\phi\Big(C_{L-1}^\phi\Big) + \beta^2, \tag{5.70}$$

with:

$$C_0^\phi(x,y) = \frac{1}{n}x^\top y + \beta^2. \tag{5.71}$$

We have that $k_c = C_L^\phi$, with L being the number of hidden layers in the network. Therefore, Lemma 5.5 ensures the smoothness result under Assumption 5.5.

Let us now consider Assumption 5.6; in particular, $\beta > 0$. We prove by induction over L in the following that:

- $B_1 \colon x \mapsto C_L^\phi(x,y)$ is smooth over $U = \{x \in \mathbb{R}^n \mid \|x\| \neq \|y\|\}$;
- $B_2 \colon x \mapsto C_L^\phi(x,x)$ is smooth;
- for all $x, x' \in \mathbb{R}^n$ with $\|x\| \neq \|x'\|$, $B_2(x) \neq B_2(x')$.

The result is immediate for $L = 0$. We now suppose that it holds for some $L \in \mathbb{N}$ and prove that it also holds for $L+1$ hidden layers. Let us express B_2:

$$B_2(x) = \mathbb{E}_{\varepsilon\sim\mathcal{N}(0,1)}\left[\phi\left(\varepsilon\sqrt{C_L^\phi(x,x)+\beta^2}\right)^2\right]. \tag{5.72}$$

Using Lemma 5.6 and Remark 5.3, the fact that $\beta > 0$ and the induction hypothesis ensures that B_2 is smooth. Moreover, Assumption 5.6, in particular Equation (5.48), allows us to assert that $\|x\| \neq \|x'\|$ implies $B_2(x) \neq B_2(x')$.

Finally, in order to apply Lemma 5.6 to prove the smoothness of B_1 over U, there remains to show that the following matrix is invertible:

$$\Sigma_\beta^{x,y} \triangleq \begin{pmatrix} C_L^\phi(x,x)+\beta^2 & C_L^\phi(x,y)+\beta^2 \\ C_L^\phi(x,y)+\beta^2 & C_L^\phi(y,y)+\beta^2 \end{pmatrix}. \tag{5.73}$$

Let us compute its determinant:

$$\begin{aligned} \det\Sigma_\beta^{x,y} &= \Big(C_L^\phi(x,x)+\beta^2\Big)\Big(C_L^\phi(y,y)+\beta^2\Big) - \Big(C_L^\phi(x,y)+\beta^2\Big)^2 \\ &= \det\Sigma_0^{x,y} + \beta^2\Big(C_L^\phi(x,x) + C_L^\phi(y,y) - 2C_L^\phi(x,y)\Big). \end{aligned} \tag{5.74}$$

C_L^ϕ is a symmetric positive semi-definite kernel, thus:

$$\det \Sigma_\beta^{x,y} - \det \Sigma_0^{x,y} = \beta^2 \cdot \begin{pmatrix} 1 & -1 \end{pmatrix} \Sigma_0^{x,y} \begin{pmatrix} 1 \\ -1 \end{pmatrix} \geq 0. \tag{5.75}$$

Hence, if $\det \Sigma_0^{x,y} > 0$, then $\det \Sigma_\beta^{x,y} > 0$. Besides, if $\det \Sigma_0^{x,y} = 0$, then:

$$\det \Sigma_\beta^{x,y} = \beta^2 \left(\sqrt{B_2(x)} - \sqrt{B_2(y)} \right)^2 > 0, \tag{5.76}$$

for all $x \in U$. This proves that B_1 is indeed smooth over U, and concludes the induction.

Note that U is indeed an open set whose complement in $\mathbb{R}^n$ has null Lebesgue measure. Overall, the result is thus proved for $A = k_c$; a similar reasoning using the previous induction result also transfers the result to $A = \mathcal{K}_{\phi'}(k_c)$. □

Proposition 5.5 (*Differentiability of k*)

Let k be the NTK of an infinite-width architecture following Assumption 5.4. For any $y \in \mathbb{R}^n$:

- *if Assumption 5.5 holds, then $k(\cdot, y)$ is smooth everywhere over $\mathbb{R}^n$;*
- *if Assumption 5.6 holds, then $k(\cdot, y)$ is smooth almost everywhere over $\mathbb{R}^n$, in particular over an open set whose complement has null Lebesgue measure.*

Proof: According to the definitions of [128], [14] and [121], the smoothness of the kernel is guaranteed whenever the conjugate kernel k_c and its transform $\mathcal{K}_{\phi'}(k_c)$ are smooth; the result of Lemma 5.7 then applies. In the case of residual networks, there is a slight adaptation of the formula which does not change its regularity. Regarding convolutional networks, their conjugate kernels and NTKs involve finite combinations of such dense conjugate kernels and NTKs, thereby preserving their smoothness almost everywhere. □

Proposition 5.6 (*Differentiability of f_t*)

Let f_t be a solution to Equation (5.9) under Assumptions 5.1 and 5.3 by Theorem 5.1, with k the NTK of an infinite-width neural network and f_0 an initialization of the latter.

Then, under Assumptions 5.4 and 5.5, f_t is smooth everywhere. Under Assumptions 5.4 and 5.6, f_t is smooth almost everywhere, in particular over an open set whose complement has null Lebesgue measure.

Proof: From Theorem 5.1, we have:

$$f_t - f_0 = \mathcal{T}_{k,\hat{\gamma}} \left(\int_0^t \nabla^{\hat{\gamma}} \mathcal{L}_{\hat{\alpha}}(f_s) \mathrm{d}s \right). \tag{5.77}$$

We observe that $\mathcal{T}_{k,\hat{\gamma}}(h)$ has, for any $h \in L^2(\hat{\gamma})$, a regularity which only depends on the regularity of $k(\cdot, y)$ for $y \in \operatorname{supp} \hat{\gamma}$. Indeed, if $k(\cdot, y)$ is smooth in a certain neighborhood V for every such y, we can bound $\partial^\alpha k(\cdot, y)$ over V for every y and any multi-index α and then use dominated convergence to prove that $\mathcal{T}_{k,\hat{\gamma}}(h)(\cdot)$ is smooth over V. Therefore, the regularity of $k(\cdot, y)$ transfers to $f_t - f_0$. Given Proposition 5.4, there remains to prove the same result for f_0.

The theorem then follows from the fact that f_0 has the same regularity as its conjugate kernel k_c thanks to Lemma 5.4 because f_0 is a sample from the GP with kernel k_c. Lemma 5.7 shows the smoothness almost everywhere over an open set of applications $x \mapsto k_c(x, y)$; to apply Lemma 5.4 and concludes this proof, this result must be generalized to prove the smoothness of k_c with respect to both its inputs. This can be done by generalizing the proofs of Lemmas 5.5 and 5.6 to show the smoothness of kernels with respect to both x and y, with the same arguments than for x alone. □

Remark 5.4

In the previous theorem, f_0 is considered to be the initialization of the network. However, we highlight that, without loss of generality, this theorem encompasses the change of training distribution $\hat{\gamma}$ during GAN training. Indeed, as explained in Section 5.4.2, f_0 after j steps of generator training can actually be decomposed as, for some $h_k \in L^2(\hat{\gamma}_k)$, $k \in [\![1, j]\!]$:

$$f_0 = f^0 + \sum_{k=1}^{j} \mathcal{T}_{k,\hat{\gamma}_k}(h_k), \tag{5.78}$$

by taking into account the updates of the discriminators over the whole GAN optimization process. The proof of Theorem 5.2 can then be applied similarly in this case by showing the differentiability of $f_0 - f^0$ on the one hand and of f^0, being the initialization of the discriminator at the very beginning of GAN training, on the other hand.

5.6 Dynamics of the Generated Distribution

In this section, starting from the results obtained previously for the discriminator, we assess their implication on the generated distribution and its evolution during training. In particular, we derive the general continuity equation it follows, thus determining its trajectory, and then study the case of some specific loss functions. Finally, we end our analysis by providing optimality results under stricter conditions.

5.6.1 General considerations

The previous differentiability results allow us to study Equation (5.3), by ensuring the existence of $\nabla f^\star_{\hat{\alpha}_g}$. We consider Equation (5.3) in continuous-time like Equation (5.5), with training time ℓ as well as $g_\ell \triangleq g_{\theta_\ell}$ and $\alpha_\ell \triangleq \alpha_{g_\ell}$. NTKs enable us to describe the generated distribution's dynamics.

Proposition 5.7 (*Dynamics of α_ℓ*)

Under Assumptions 5.4 and 5.5, Equation (5.3) *is well-posed and yields in continuous-time, with k_{g_ℓ} the NTK of the generator g_ℓ:*

$$\partial_\ell g_\ell = -\mathcal{T}_{k_{g_\ell}, p_z}\left(z \mapsto \nabla_x c_{f^\star_{\hat{\alpha}_{g_\ell}}}(x)\Big|_{x=g_\ell(z)}\right). \tag{5.79}$$

Equivalently, the following continuity equation holds for the joint distribution α^z_ℓ of $(z, g_\ell(z))$ under $z \sim p_z$:

$$\partial_\ell \alpha^z_\ell = -\nabla_x \cdot \left(\alpha^z_\ell \mathcal{T}_{k_{g_\ell}, p_z}\left(z \mapsto \nabla_x c_{f^\star_{\hat{\alpha}_{g_\ell}}}(x)\Big|_{x=g_\ell(z)}\right)\right), \tag{5.80}$$

where α_ℓ can be recovered as the marginalization of α^z_ℓ over $z \sim p_z$.

Proof: Assumptions 5.4 and 5.5 ensure, via Proposition 5.4 and Theorem 5.2 that the trained discriminator is differentiable everywhere at all times, whatever the state of the generator. Therefore, Equation (5.3) is well-posed.

Let us consider its continuous-time version with discriminators trained on discrete distributions as described above:

$$\partial_\ell \theta_\ell = -\mathbb{E}_{z\sim p_z}\left[\nabla_\theta g_\ell(z)^\top \nabla_x c_{f^\star_{\hat{\alpha}_{g_\ell}}}(x)\Big|_{x=g_\ell(z)}\right]. \tag{5.81}$$

By following [192, Equation (5)]'s reasoning on a similar equation, Equation (5.81) yields the following generator dynamics for all inputs $z \in \mathbb{R}^d$:

$$\partial_\ell g_\ell(z) = -\mathbb{E}_{z' \sim p_z}\left[\nabla_{\theta_\ell} g_\ell(z)^\top \nabla_{\theta_\ell} g_\ell\Big(z'\Big) \nabla_x \, c_{f^\star_{\hat{\alpha}_{g_\ell}}}(x)\Big|_{x=g_\ell(z')}\right]. \tag{5.82}$$

We recognize the NTK k_{g_ℓ} of the generator as:

$$k_{g_\ell}\Big(z, z'\Big) \triangleq \nabla_{\theta_\ell} g_\ell(z)^\top \nabla_{\theta_\ell} g_\ell\Big(z'\Big). \tag{5.83}$$

From this, we obtain the dynamics of the generator:

$$\partial_\ell g_\ell = -\mathcal{T}_{k_{g_\ell}, p_z}\left(z \mapsto \nabla_x \, c_{f^\star_{\hat{\alpha}_{g_\ell}}}(x)\Big|_{x=g_\ell(z)}\right). \tag{5.84}$$

In other words, the transported particles $(z, g_\ell(z))$ have trajectories X_ℓ which are solutions of the Ordinary Differential Equation (ODE):

$$\frac{\mathrm{d}X_\ell}{\mathrm{d}\ell} = (0, v_\ell(X_\ell)), \tag{5.85}$$

where:

$$v_\ell = -\mathcal{T}_{k_{g_\ell}, p_z}\left(z \mapsto \nabla_x \, c_{f^\star_{\hat{\alpha}_{g_\ell}}}(x)\Big|_{x=g_\ell(z)}\right). \tag{5.86}$$

Then, because $\alpha^z_\ell \triangleq (\mathrm{id}, g_\ell)_\sharp p_z$ is the induced transported density, following [6], whenever the ODE above is well-defined and has unique solutions (which is necessarily the case for any trained g), α^z_ℓ verifies the continuity equation with the velocity field v_ℓ:

$$\begin{aligned}
\partial_\ell \alpha^z_\ell &= -\nabla_{z,x} \cdot \left(\alpha^z_\ell \left(0, \mathcal{T}_{k_{g_\ell}, p_z}\left(z \mapsto \nabla_x \, c_{f^\star_{\hat{\alpha}_{g_\ell}}}(x)\Big|_{x=g_\ell(z)}\right)\right)\right) \\
&= -\nabla_x \cdot \left(\alpha^z_\ell \mathcal{T}_{k_{g_\ell}, p_z}\left(z \mapsto \nabla_x \, c_{f^\star_{\hat{\alpha}_{g_\ell}}}(x)\Big|_{x=g_\ell(z)}\right)\right).
\end{aligned} \tag{5.87}$$

This yields the desired result. □

In the infinite-width limit of the generator, the generator's NTK is also constant: $k_{g_\ell} = k_g$; this is the setting that we consider to study the implications of the latter proposition. Suppose that there exists a functional $\mathscr{C}$ over $L^2(\Omega)$ such that $c_{f^\star_{\hat{\alpha}}} = \partial_{\hat{\alpha}} \mathscr{C}(\hat{\alpha})$. Standard results in gradient flows theory – see [8, Chapter 10] for a detailed exposition, or [9, Appendix A.3] for a summary – state that $\nabla c_{f^\star_{\hat{\alpha}}}$ is in this case the strong subdifferential of $\mathscr{C}(\hat{\alpha})$ for the Wasserstein geometry.

When $k_g(z, z') = \delta_{z-z'} I_n$ with δ a Dirac centered at 0, we have $\mathcal{T}_{k_g, p_z} = \mathrm{id}$. Then, from Equation (5.80), α^z_ℓ follows the Wasserstein gradient flow with $c_{f^\star_{\hat{\alpha}}}$ as potential. This implies that $\mathscr{C}(\hat{\alpha}_\ell)$ is a decreasing function of the generator's training time ℓ. In other words, the generator g is trained to minimize $\mathscr{C}\Big(\hat{\alpha}_g\Big)$, which is the implicit objective induced by the discriminator.

In the general case, $\mathcal{T}_{k_g, p_z}$ introduces interactions between generated particles as a consequence of the neural parameterization of the generator. Then, Equation (5.80) amounts to following the same gradient flow as before, but in a Stein geometry [78] – instead of a Wasserstein geometry – determined by the generator's integral operator, directly implying that in this case $\mathscr{C}(\hat{\alpha}_\ell)$ also decreases during training. This geometrical understanding opens interesting perspectives for theoretical analysis, e.g. we see that GAN training in this regime generalizes Stein variational gradient descent [164], with the

Kullback-Leibler minimization objective between generated and target distributions being replaced with $\mathscr{C}(\hat{\alpha})$.

Improving our understanding of Equation (5.80) is fundamental towards elucidating the open problem of the neural generator's convergence. Our study enables us to shed light on these dynamics and highlights the necessity of pursuing the study of GANs via NTKs to obtain a more comprehensive understanding of them, which is the purpose of the rest of this paper. In particular, the non-interacting case where $\mathcal{T}_{k_g,p_z} = \mathrm{id}$ already yields particularly useful insights, that we explore in Section 5.7. Moreover, we discuss in the following section standard GAN losses and attempt to determine the minimized functional $\mathscr{C}$ in these cases.

5.6.2 Study of Specific Losses

Armed with the general framework of the previous section, we derive in this section more fine-grained results thanks to additional assumptions on the loss function covering standard GAN models.

- *Notations and Preliminaries*

We first need to introduce some definitions.

The presented solutions to Equation (5.9) leverage a notion of functions of linear operators, similarly to functions of matrices [118]. We define such functions in the simplified case of non-negative symmetric compact operators with a finite number of eigenvalues, such as $\mathcal{T}_{k,\hat{\gamma}}$.

Definition 5.3 (*Linear operator*)

Let $\mathcal{A}\colon L^2(\hat{\gamma}) \to L^2(\Omega)$ be a non-negative symmetric compact linear operator with a finite number of eigenvalues, for which the spectral theorem guarantees the existence of a countable orthonormal basis of eigenfunctions with non-negative eigenvalues. If $\varphi\colon \mathbb{R}_+ \to \mathbb{R}$, we define $\varphi(\mathcal{A})$ as the linear operator with the same eigenspaces as $\mathcal{A}$, with their respective eigenvalues mapped by φ; in other words, if λ is an eigenvalue of $\mathcal{A}$, then $\varphi(\mathcal{A})$ admits the eigenvalue $\varphi(\lambda)$ with the same eigenspace.

In the case where $\mathcal{A}$ is a matrix, this amounts to diagonalizing $\mathcal{A}$ and transforming its diagonalization elementwise using φ. Note that $\mathcal{T}_{k,\hat{\gamma}}$ has a finite number of eigenvalues since it is generated by a finite linear combination of linear operators (see Definition 5.2).

We also need to define the following Radon–Nikodym derivatives with inputs in $\operatorname{supp}\hat{\gamma}$:

$$\rho = \frac{\mathrm{d}\left(\hat{\beta} - \hat{\alpha}\right)}{\mathrm{d}\left(\hat{\beta} + \hat{\alpha}\right)}, \qquad \rho_1 = \frac{\mathrm{d}\hat{\alpha}}{\mathrm{d}\hat{\gamma}}, \qquad \rho_2 = \frac{\mathrm{d}\hat{\beta}}{\mathrm{d}\hat{\gamma}}, \tag{5.88}$$

knowing that

$$\rho = \frac{1}{2}(\rho_2 - \rho_1), \qquad \rho_1 + \rho_2 = 2. \tag{5.89}$$

These functions help us to compute the functional gradient of $\mathcal{L}_{\hat{\alpha}}$, as follows.

Lemma 5.8 (*Loss derivative*)

Under Assumption 5.3:

$$\nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(f) = \rho_1 a'_f - \rho_2 b'_f = \rho_1 \cdot \left(a' \circ f\right) - \rho_2 \cdot \left(b' \circ f\right). \tag{5.90}$$

Proof: We have from Equation (5.2):

$$\mathcal{L}_{\hat{\alpha}}(f) = \mathbb{E}_{x\sim\hat{\alpha}}\left[a_f(x)\right] - \mathbb{E}_{y\sim\hat{\beta}}\left[b_f(y)\right] = \left\langle \rho_1, a_f \right\rangle_{L^2(\hat{\gamma})} - \left\langle \rho_2, b_f \right\rangle_{L^2(\hat{\gamma})}, \tag{5.91}$$

hence by composition:

$$\nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(f) = \rho_1 \cdot \Big(a' \circ f\Big) - \rho_2 \cdot \Big(b' \circ f\Big) = \rho_1 a'_f - \rho_2 b'_f. \tag{5.92}$$

□

• *The IPM as an NTK MMD Minimizer*

We study the case of the IPM loss, with the following remarkable discriminator expression, from which we deduce the objective minimized by the generator.

Proposition 5.8 (*IPM discriminator*)

Under Assumptions 5.1 and 5.2, the solutions of Equation (5.9) *for* $a = b = \mathrm{id}$ *are* $f_t = f_0 + t f^*_{\hat{\alpha}_g}$, *where* $f^*_{\hat{\alpha}_g}$ *is the unnormalized MMD witness function [105] with kernel* k, *yielding:*

$$\begin{aligned} f^*_{\hat{\alpha}_g} &= \mathbb{E}_{x\sim\hat{\alpha}_g}[k(x,\cdot)] - \mathbb{E}_{y\sim\hat{\beta}}[k(y,\cdot)], \\ \mathcal{L}_{\hat{\alpha}_g}(f_t) &= \mathcal{L}_{\hat{\alpha}_g}(f_0) + t \cdot \mathrm{MMD}_k^2\Big(\hat{\alpha}_g, \hat{\beta}\Big). \end{aligned} \tag{5.93}$$

Proof: Assumptions 5.1 and 5.2 are already assumed and Assumption 5.3 holds for the given a and b of the IPM loss. Thus, Theorem 5.1 applies, and there exists a unique solution $t \mapsto f_t$ to Equation (5.9) over $\mathbb{R}_+$ in $L^2(\Omega)$ for a given initial condition f_0. Therefore, in order to find the solution of Equation (5.9), there remains to prove that, for a given initial condition f_0,

$$g\colon t \mapsto g_t = f_0 + t f^*_{\hat{\alpha}} \tag{5.94}$$

is a solution to Equation (5.9) with $g_0 = f_0$ and $g_t \in L^2(\Omega)$ for all $t \in \mathbb{R}_+$.

Let us first express the gradient of $\mathcal{L}_{\hat{\alpha}}$. We have from Lemma 5.8, with $a_f = b_f = f$:

$$\nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(f) = \rho_1 a'_f - \rho_2 b'_f = -2\rho. \tag{5.95}$$

So Equation (5.9) equates to:

$$\partial_t f_t = -2\mathcal{T}_{k,\hat{\gamma}}(\rho) = 2\int_x k(\cdot,x)\rho(x)\mathrm{d}\hat{\gamma}(x) = \int_x k(\cdot,x)\mathrm{d}\hat{\alpha}(x) - \int_y k(\cdot,y)\mathrm{d}\hat{\beta}(y), \tag{5.96}$$

by definition of ρ (see Equation (5.88)), yielding:

$$\partial_t f_t = f^*_{\hat{\alpha}}. \tag{5.97}$$

Clearly, $t \mapsto g_t = f_0 + t f^*_{\hat{\alpha}}$ is a solution of the latter equation, $g_0 = f_0$ and $g_t \in L^2(\Omega)$ given that $\operatorname{supp}\hat{\gamma}$ is finite and $k \in L^2\Big(\Omega^2\Big)$ by assumption. The set of solutions for the IPM loss is thus characterized.

Finally, let us compute $\mathcal{L}_{\hat{\alpha}}(f_t)$. By linearity of $\mathcal{L}_{\hat{\alpha}}$ for $a = b = \mathrm{id}$:

$$\mathcal{L}_{\hat{\alpha}}(f_t) = \mathcal{L}_{\hat{\alpha}}(f_0) + t \cdot \mathcal{L}_{\hat{\alpha}}(f^*_{\hat{\alpha}}) = \mathcal{L}_{\hat{\alpha}}(f_0) + t \cdot \mathcal{L}_{\hat{\alpha}}\Big(\mathcal{T}_{k,\hat{\gamma}}(-2\rho)\Big). \tag{5.98}$$

But, from Equation (5.91), $\mathcal{L}_{\hat{\alpha}}(f) = \langle -2\rho, f\rangle_{L^2(\hat{\gamma})}$, hence:

$$\mathcal{L}_{\hat{\alpha}}(f_t) = \mathcal{L}_{\hat{\alpha}}(f_0) + t \cdot \Big\langle -2\rho, \mathcal{T}_{k,\hat{\gamma}}(-2\rho)\Big\rangle_{L^2(\hat{\gamma})} = \mathcal{L}_{\hat{\alpha}}(f_0) + t \cdot \Big\|\mathcal{T}_{k,\hat{\gamma}}(-2\rho)\Big\|^2_{\mathcal{H}_k^{\hat{\gamma}}}. \tag{5.99}$$

By noticing that $\mathcal{T}_{k,\hat{\gamma}}(-2\rho) = f^*_{\hat{\alpha}}$ and that $\|f^*_{\hat{\alpha}}\|_{\mathcal{H}_k^{\hat{\gamma}}} = \mathrm{MMD}_k\Big(\hat{\alpha}, \hat{\beta}\Big)$ since $f^*_{\hat{\alpha}}$ is the unnormalized MMD witness function, the expression of $\mathcal{L}_{\hat{\alpha}}(f_t)$ in the proposition is obtained. □

The latter result signifies that the direction of the gradient given to the discriminator at each of its optimization step is optimal within the RKHS of its NTK, stemming from the linearity of the IPM loss. The connection with MMD is especially interesting as it has been thoroughly studied in the literature [193]. If k is characteristic, as discussed in Section 5.8.4, then it defines a distance between distributions. Moreover, the statistical properties of the loss induced by the discriminator directly follow from those of the MMD: it is an unbiased estimator with a squared sample complexity that is independent of the dimension of the samples [104].

Remark 5.5 (*Instance smoothing*)

We show for IPMs that modeling the discriminator's architecture amounts to smoothing out the input distribution using the kernel integral operator $\mathcal{T}_{k,\hat{\gamma}_g}$ and can thus be seen as a generalization of the regularization technique for GANs called instance noise [254]. This is discussed in Section 5.8.3.

Suppose that the discriminator is reinitialized at every step of the generator, with $f_0 = 0$ in Equation (5.9); this is possible with the initialization scheme of [292]. Then, as $c = \mathrm{id}$ and from Proposition 5.8, $\nabla c_{f_{\hat{\alpha}}} = \tau \nabla f^*_{\hat{\alpha}_g}$, where τ is the training time of the discriminator. The latter gradient constitutes the gradient flow of the squared MMD, as shown by [9] with convergence guarantees and discretization properties in the absence of generator. This signifies that $\mathscr{C}(\hat{\alpha}) = \tau \mathrm{MMD}^2_k\big(\hat{\alpha}_g, \hat{\beta}\big)$ (see Section 5.6).

Therefore, in the IPM case, the discriminator leads the generator to be trained to minimize the MMD between the empirical generated and target distributions, with respect to the NTK of the discriminator. This is the subject of study of [191], who derive convergence results about the generator trained in such conditions, considerations about the discriminator's NTK aside. Our work is, to our knowledge, the first to consider NTKs as kernels for the MMD, concurrently with [54].

Remark 5.6 (*IPM and WGAN*)

Along with a constraint on the set of functions, the IPM is involved in the earth mover's distance $\mathcal{W}_1$ [267] – used in WGAN and StyleGAN [137], close to the hinge loss of BigGAN [43] –, the MMD – used in MMD GAN [155] –, the total variation, etc. In Proposition 5.8, we consider the IPM with the sole constraint of having a neural discriminator. Our analysis implies that this suffices to ensure relevant gradients, given the aforementioned convergence results. This contradicts the recurrent assertion that the Lipschitz constraint of WGAN [12] is necessary to solve the gradient issues of prior approaches. Indeed, these issues originate from the analyses inadequacy, as shown in this work. Hence, while WGAN tackles these issues by changing the loss, we fundamentally address them with a refined framework. An analysis of WGAN, left for future work, would require combining the neural discriminator and Lipschitz constraints.

• *LSGAN and New Divergences*

Optimality of the discriminator can be proved when assuming that its loss function is well-behaved. Let us consider the case of LSGAN, for which Equation (5.9) can be solved by slightly adapting the results from [128] for regression.

Proposition 5.9 (*LSGAN discr.*)

Under Assumptions 5.1 and 5.2, the solutions of Equation (5.9) *for $a = -(\mathrm{id}+1)^2$ and $b = -(\mathrm{id}-1)^2$ are defined for all $t \in \mathbb{R}_+$ as:*

$$f_t = \exp\Big(-4t\mathcal{T}_{k,\hat{\gamma}_g}\Big)(f_0 - \rho) + \rho, \qquad \rho = \frac{\mathrm{d}\Big(\hat{\beta} - \hat{\alpha}_g\Big)}{\mathrm{d}\Big(\hat{\beta} + \hat{\alpha}_g\Big)}. \tag{5.100}$$

Proof: Assumptions 5.1 and 5.2 are already assumed and Assumption 5.3 holds for the given a and b in LSGAN. Thus, Theorem 5.1 applies, and there exists a unique solution $t \mapsto f_t$ to Equation (5.9) over $\mathbb{R}_+$ in $L^2(\Omega)$ for a given initial condition f_0. Therefore, there remains to prove that, for a given initial condition f_0,

$$g\colon t \mapsto g_t = f_0 + \varphi_t\Big(\mathcal{T}_{k,\hat{\gamma}}\Big)(f_0 - \rho) \tag{5.101}$$

is a solution to Equation (5.9) with $g_0 = f_0$ and $g_t \in L^2(\Omega)$ for all $t \in \mathbb{R}_+$.

Let us first express the gradient of $\mathcal{L}_{\hat{\alpha}}$. We have from Lemma 5.8, with $a_f = -(f+1)^2$ and $b_f = -(f-1)^2$:

$$\nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(f) = \rho_1 a'_f - \rho_2 b'_f = -2\rho_1(f+1) - 2\rho_2(f-1) = 4\rho - 4f. \tag{5.102}$$

So Equation (5.9) equates to:

$$\partial_t f_t = 4\mathcal{T}_{k,\hat{\gamma}}(\rho - f_t). \tag{5.103}$$

Now let us prove that g_t is a solution to Equation (5.103). We have:

$$\partial_t g_t = -4\Big(\mathcal{T}_{k,\hat{\gamma}} \circ \exp\Big(-4t\mathcal{T}_{k,\hat{\gamma}}\Big)\Big)(f_0 - \rho) = -4\Big(\mathcal{T}_{k,\hat{\gamma}} \circ \exp\Big(-4t\mathcal{T}_{k,\hat{\gamma}}\Big)\Big)(f_0 - \rho). \tag{5.104}$$

Restricted to $\operatorname{supp}\hat{\gamma}$, we can write from Equation (5.101):

$$g_t = f_0 + \left(\exp\left(-4t\,\mathcal{T}_{k,\hat{\gamma}}\Big|_{\operatorname{supp}\hat{\gamma}}\right) - \mathrm{id}_{L^2(\hat{\gamma})}\right)(f_0 - \rho), \tag{5.105}$$

and plugging this in Equation (5.104):

$$\partial_t g_t = -4\mathcal{T}_{k,\hat{\gamma}}(g_t - \rho), \tag{5.106}$$

where we retrieve the differential equation of Equation (5.103). Therefore, g_t is a solution to Equation (5.103).

It is clear that $g_0 = f_0$. Moreover, $\mathcal{T}_{k,\hat{\gamma}}$ being decomposable in a finite orthonormal basis of elements of operators over $L^2(\Omega)$, its exponential has values in $L^2(\Omega)$ as well, making g_t belong to $L^2(\Omega)$ for all t. With this, the proof is complete. □

In the previous result, ρ is the optimum of $\mathcal{L}_{\hat{\alpha}_g}$ over $L^2\Big(\hat{\gamma}_g\Big)$. When k is positive definite over $\hat{\gamma}_g$ (see Section 5.8.4), f_t tends to the optimum for $\mathcal{L}_{\hat{\alpha}_g}$ as its limit is ρ over $\operatorname{supp}\hat{\gamma}_g$. Nonetheless, unlike the discriminator with arbitrary values of Section 5.3.2, f_∞ is defined over all Ω thanks to the integral operator $\mathcal{T}_{k,\hat{\gamma}_g}$. It is also the solution to the minimum norm interpolant problem in the RKHS [128], therefore explaining why the discriminator does not overfit in scarce data regimes (see Section 5.7), and consequently has bounded gradients despite large training times. We also prove a generalization of this optimality conclusion for concave bounded losses in Section 5.6.3.

Following the discussion initiated in Section 5.3.2 and applying it to LSGAN using Proposition 5.9, similarly to the Jensen-Shannon, the resulting generator loss on discrete training data is constant when the discriminator is optimal. However, the gradients received by the generator are not necessarily null, e.g. in the empirical analysis of Section 5.7. This is because the learning problem of the generator induced by the discriminator makes the generator minimize another loss $\mathscr{C}$, as explained in Section 5.6. This raises the question of determining $\mathscr{C}$ for LSGAN and other standard losses. Furthermore, the same problem arises for gradients obtained from incompletely trained discriminators f_t. Unlike the IPM case for which the results of [9] who leveraged the theory of [8] led to a remarkable solution, this connection remains to be established for other adversarial losses. We leave this as future work.

5.6.3 Optimality Analysis

In this section, we derive an optimality result for concave bounded loss functions of the discriminator and positive definite kernels. We finish by some considerations for the original GAN loss which unfortunately falls short from providing optimality.

- *Assumptions*

We first assume that the NTK is positive definite over the training dataset.

Assumption 5.7 (*Positive definite kernel*)

k is positive definite over $\hat{\gamma}$.

This positive definiteness property equates for finite datasets to the invertibility of the mapping

$$\begin{aligned} \mathcal{T}_{k,\hat{\gamma}}\big|_{\operatorname{supp}\hat{\gamma}} \colon L^2(\hat{\gamma}) &\to L^2(\hat{\gamma}) \\ h &\mapsto \mathcal{T}_{k,\hat{\gamma}}(h)\big|_{\operatorname{supp}\hat{\gamma}} \end{aligned}, \tag{5.107}$$

that can be seen as a multiplication by the invertible Gram matrix of k over $\hat{\gamma}$. We further discuss this hypothesis in Section 5.8.4.

We also assume the following properties on the discriminator loss function.

Assumption 5.8 (*Concave loss*)

$\mathcal{L}_{\hat{\alpha}}$ is concave and bounded from above, and its supremum is reached on a unique point $y^\star$ in $L^2(\hat{\gamma})$.

Moreover, we need for the sake of the proof a uniform continuity assumption on the solution to Equation (5.9).

Assumption 5.9 (*Solution continuity*)

$t \mapsto f_t|_{\operatorname{supp}\hat{\gamma}}$ is uniformly continuous over $\mathbb{R}_+$.

Note that these assumptions are verified in the case of LSGAN, which is the typical application of the optimality results that we prove in the following.

- *Optimality Result*

Proposition 5.10 (*Asymptotic optimality*)

Under Assumptions 5.1 to 5.3 and 5.7 to 5.9, f_t converges pointwise when $t \to \infty$, and:

$$\mathcal{L}_{\hat{\alpha}}(f_t) \xrightarrow[t\to\infty]{} \mathcal{L}_{\hat{\alpha}}(y^\star), \quad f_\infty = f_0 + \mathcal{T}_{k,\hat{\gamma}}\Big(\mathcal{T}_{k,\hat{\gamma}}\big|^{-1}_{\operatorname{supp}\hat{\gamma}}\big(y^\star - f_0|_{\operatorname{supp}\hat{\gamma}}\big)\Big), \quad f_\infty|_{\operatorname{supp}\hat{\gamma}} = y^\star, \tag{5.108}$$

where we recall that:

$$y^\star = \operatorname*{arg\,max}_{y \in L^2(\hat{\gamma})} \mathcal{L}_{\hat{\alpha}}(y). \tag{5.109}$$

This result ensures that, for concave losses such as LSGAN, the optimum for $\mathcal{L}_{\hat{\alpha}}$ in $L^2(\Omega)$ is reached for infinite training times by neural network training in the infinite-width regime when the NTK of the discriminator is positive definite. However, this also provides the expression of the optimal network outside $\operatorname{supp}\hat{\gamma}$ thanks to the smoothing of $\hat{\gamma}$.

In order to prove this proposition, we need the following intermediate results: the first about the functional gradient of $\mathcal{L}_{\hat{\alpha}}$ on the solution f_t; the second about a direct application of positive definite kernels showing that one can retrieve $f \in \mathcal{H}_k^{\hat{\gamma}}$ over all Ω from its restriction to $\operatorname{supp}\hat{\gamma}$.

Lemma 5.9

Under Assumptions 5.1 to 5.3 and 5.7 to 5.9, $\nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(f_t) \to 0$ when $t \to \infty$. Since $\operatorname{supp}\hat{\gamma}$ is finite, this limit can be interpreted pointwise.

Proof: Assumptions 5.1 to 5.3 ensure the existence and uniqueness of f_t, by Theorem 5.1.

$t \mapsto \hat{f}_t \triangleq f_t|_{\operatorname{supp}\hat{\gamma}}$ and $\mathcal{L}_{\hat{\alpha}}$ being differentiable, $t \mapsto \mathcal{L}_{\hat{\alpha}}(f_t)$ is differentiable, and:

$$\partial_t \mathcal{L}_{\hat{\alpha}}(f_t) = \left\langle \nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(f_t), \partial_t \hat{f}_t \right\rangle_{L^2(\hat{\gamma})} = \left\langle \nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(f_t), \mathcal{T}_{k,\hat{\gamma}}\Big(\nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(f_t)\Big) \right\rangle_{L^2(\hat{\gamma})}, \tag{5.110}$$

using Equation (5.9). This equates to:

$$\partial_t \mathcal{L}_{\hat{\alpha}}(f_t) = \left\| \mathcal{T}_{k,\hat{\gamma}}\Big(\nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(f_t)\Big) \right\|^2_{\mathcal{H}_k^{\hat{\gamma}}} \geq 0, \tag{5.111}$$

where $\|\cdot\|_{\mathcal{H}_k^{\hat{\gamma}}}$ is the semi-norm associated to the RKHS $\mathcal{H}_k^{\hat{\gamma}}$. Note that this semi-norm is dependent on the restriction of its input to $\operatorname{supp}\hat{\gamma}$ only. Therefore, $t \mapsto \mathcal{L}_{\hat{\alpha}}(f_t)$ is increasing. Since $\mathcal{L}_{\hat{\alpha}}$ is bounded from above, $t \mapsto \mathcal{L}_{\hat{\alpha}}(f_t)$ admits a limit when $t \to \infty$.

We now aim at proving from the latter fact that $\partial_t \mathcal{L}_{\hat{\alpha}}(f_t) \to 0$ when $t \to \infty$. We notice that $\|\cdot\|^2_{\mathcal{H}_k^{\hat{\gamma}}}$ is uniformly continuous over $L^2(\hat{\gamma})$ since $\operatorname{supp}\hat{\gamma}$ is finite, $\nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}$ is uniformly continuous over $L^2(\hat{\gamma})$ since a' and b' are Lipschitz-continuous, $\mathcal{T}_{k,\hat{\gamma}}\Big|_{\operatorname{supp}\hat{\gamma}}$ is uniformly continuous as it amounts to a finite matrix multiplication, and Assumption 5.9 gives that $t \mapsto f_t|_{\operatorname{supp}\hat{\gamma}}$ is uniformly continuous over $\mathbb{R}_+$. Therefore, their composition $t \mapsto \partial_t \mathcal{L}_{\hat{\alpha}}(f_t)$ (from Equation (5.111)) is uniformly continuous over $\mathbb{R}_+$. Using Barbălat's Lemma [83], we conclude that $\partial_t \mathcal{L}_{\hat{\alpha}}(f_t) \to 0$ when $t \to \infty$.

Furthermore, k is positive definite over $\hat{\gamma}$ by Assumption 5.7, so $\|\cdot\|_{\mathcal{H}_k^{\hat{\gamma}}}$ is actually a norm. Therefore, since $\operatorname{supp}\hat{\gamma}$ is finite, the following pointwise convergence holds:

$$\nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(f_t) \xrightarrow[t\to\infty]{} 0. \tag{5.112}$$

□

Lemma 5.10 ($\mathcal{H}_k^{\hat{\gamma}}$ *determined by* $\operatorname{supp}\hat{\gamma}$)

Under Assumptions 5.1, 5.2 and 5.7, for all $f \in \mathcal{H}_k^{\hat{\gamma}}$, the following holds:

$$f = \mathcal{T}_{k,\hat{\gamma}}\left(\mathcal{T}_{k,\hat{\gamma}}\Big|^{-1}_{\operatorname{supp}\hat{\gamma}}\Big(f|_{\operatorname{supp}\hat{\gamma}}\Big)\right). \tag{5.113}$$

Proof: Since k is positive definite by Assumption 5.7, then $\mathcal{T}_{k,\hat{\gamma}}\Big|_{\operatorname{supp}\hat{\gamma}}$ from Equation (5.107) is invertible.

Let $f \in \mathcal{H}_k^{\hat{\gamma}}$. Then, by definition of the RKHS in Definition 5.2, there exists $h \in L^2(\hat{\gamma})$ such that $f = \mathcal{T}_{k,\hat{\gamma}}(h)$. In particular, $f|_{\operatorname{supp}\hat{\gamma}} = \mathcal{T}_{k,\hat{\gamma}}\Big|_{\operatorname{supp}\hat{\gamma}}(h)$, hence $h = \mathcal{T}_{k,\hat{\gamma}}\Big|^{-1}_{\operatorname{supp}\hat{\gamma}}\Big(f|_{\operatorname{supp}\hat{\gamma}}\Big)$. □

We can now prove the desired proposition.

Proof of Proposition 5.10: Let us first show that f_t converges to the optimum $y^\star$ in $L^2(\hat{\gamma})$. By applying Lemma 5.9, we know that $\nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(f_t) \to 0$ when $t \to \infty$. Given that the supremum of the differentiable concave function $\mathcal{L}_{\hat{\alpha}}\colon L^2(\hat{\gamma}) \to \mathbb{R}$ is achieved at a unique point $y^\star \in L^2(\hat{\gamma})$ with finite $\operatorname{supp}\hat{\gamma}$, then the latter convergence result implies that $\hat{f}_t \triangleq f_t|_{\operatorname{supp}\hat{\gamma}}$ converges pointwise to $y^\star$ when $t \to \infty$.

Given this convergence in $L^2(\hat{\gamma})$, we can deduce convergence on the whole domain Ω by noticing that $f_t - f_0 \in \mathcal{H}_k^{\hat{\gamma}}$, from Corollary 5.1. Thus, using Lemma 5.10:

$$f_t - f_0 = \mathcal{T}_{k,\hat{\gamma}}\left(\mathcal{T}_{k,\hat{\gamma}}\Big|^{-1}_{\operatorname{supp}\hat{\gamma}}\Big((f_t - f_0)|_{\operatorname{supp}\hat{\gamma}}\Big)\right). \tag{5.114}$$

Again, since $\operatorname{supp}\hat{\gamma}$ is finite, and $\mathcal{T}_{k,\hat{\gamma}}\big|^{-1}_{\operatorname{supp}\hat{\gamma}}$ can be expressed as a matrix multiplication, the fact that f_t converges to $y^\star$ over $\operatorname{supp}\hat{\gamma}$ implies that:

$$\mathcal{T}_{k,\hat{\gamma}}\big|^{-1}_{\operatorname{supp}\hat{\gamma}}\Big((f_t - f_0)|_{\operatorname{supp}\hat{\gamma}}\Big) \xrightarrow[t\to\infty]{} \mathcal{T}_{k,\hat{\gamma}}\big|^{-1}_{\operatorname{supp}\hat{\gamma}}\Big(y^\star - f_0|_{\operatorname{supp}\hat{\gamma}}\Big). \tag{5.115}$$

Finally, using the definition of the integral operator in Definition 5.2, the latter convergence implies the following desired pointwise convergence:

$$f_t \xrightarrow[t\to\infty]{} f_0 + \mathcal{T}_{k,\hat{\gamma}}\Big(\mathcal{T}_{k,\hat{\gamma}}\big|^{-1}_{\operatorname{supp}\hat{\gamma}}\Big(y^\star - f_0|_{\operatorname{supp}\hat{\gamma}}\Big)\Big). \tag{5.116}$$

We showed at the beginning of this proof that f_t converges to the optimum $y^\star$ in $L^2(\hat{\gamma})$, so $\mathcal{L}_{\hat{\alpha}}(f_t) \to \mathcal{L}_{\hat{\alpha}}(y^\star)$ by continuity of $\mathcal{L}_{\hat{\alpha}}$ as claimed in the proposition. □

- *The case of the original GAN formulation*

Unfortunately, finding the solutions to Equation (5.9) in the case of the original GAN formulation, i.e. $a = \log(1-\sigma)$ and $b = -\log\sigma$, remains to the extent of our knowledge an open problem. We provide in the rest of this section some leads that might prove useful for more advanced analyses.

Let us first determine the expression of Equation (5.9) for vanilla GAN.

Lemma 5.11

For $a = \log(1-\sigma)$ and $b = -\log\sigma$, Equation (5.9) equates to:

$$\partial_t f_t = \mathcal{T}_{k,\hat{\gamma}}(\rho_2 - 2\sigma(f)). \tag{5.117}$$

Proof: We have from Lemma 5.8, with $a_f = b_f = f$:

$$\nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(f) = \rho_1 a'_f - \rho_2 b'_f = -\rho_1 \frac{\sigma'(f)}{1-\sigma(f)} + \rho_2 \frac{\sigma'(f)}{\sigma(f)}. \tag{5.118}$$

By noticing that $\sigma'(f) = \sigma(f)(1-\sigma(f))$, we obtain:

$$\nabla^{\hat{\gamma}}\mathcal{L}_{\hat{\alpha}}(f) = \rho_1 a'_f - \rho_2 b'_f = -\rho_1\sigma(f) + \rho_2(1-\sigma(f)) = \rho_2 - 2\sigma(f). \tag{5.119}$$

By plugging the latter expression in Equation (5.9), the desired result is achieved. □

Note that Assumption 5.3 holds for these choices of a and b. Therefore, under Assumptions 5.1 and 5.2, there exists a unique solution to Equation (5.117) in $\mathbb{R}_+ \to L^2(\Omega)$ with a given initialization f_0.

Let us first study Equation (5.117) in the simplified case of a one-dimensional ordinary differential equation.

Proposition 5.11

Let $r \in \{0, 2\}$ and $\lambda \in \mathbb{R}$. The set of differentiable solutions over $\mathbb{R}$ to the ODE:

$$\partial_t y_t = \lambda(r - 2\sigma(y_t)) \tag{5.120}$$

is the following one:

$$S = \left\{ y : t \mapsto (1-r)\Big(W\big(\mathrm{e}^{2\lambda t + C}\big) - 2\lambda t - C\Big) \;\middle|\; C \in \mathbb{R} \right\}, \tag{5.121}$$

where W the is principal branch of the Lambert W function [62].

Proof: The theorem of Cauchy-Lipschitz ensures that there exists a unique global solution to Equation (5.120) for a given initial condition $y_0 \in \mathbb{R}$. Therefore, we only need to show that all elements of S are solutions of Equation (5.120) and that they can cover any initial condition.

Let us first prove that $y\colon t \mapsto (1-r)\Big(W\big(\mathrm{e}^{2\lambda t+C}\big) - 2\lambda t - C\Big)$ is a solution of Equation (5.120). Let us express the derivative of y:

$$\frac{1}{1-r}\partial_t y_t = 2\lambda\Big(\mathrm{e}^{2\lambda t+C}W'\big(\mathrm{e}^{2\lambda t+C}\big) - 1\Big). \tag{5.122}$$

$W'(z) = \frac{W(z)}{z\big(1+W(z)\big)}$, so:

$$\frac{1}{1-r}\partial_t y_t = 2\lambda\left(\frac{W\big(\mathrm{e}^{2\lambda t+C}\big)}{1+W\big(\mathrm{e}^{2\lambda t+C}\big)} - 1\right) = -\frac{2\lambda}{1+W\big(\mathrm{e}^{2\lambda t+C}\big)}. \tag{5.123}$$

Moreover, $W(z) = z\mathrm{e}^{-W(z)}$, and with $r-1 \in \{1,-1\}$:

$$\frac{1}{1-r}\partial_t y_t = -\frac{2\lambda}{1+\mathrm{e}^{2\lambda t+C}\mathrm{e}^{-W\left(\mathrm{e}^{2\lambda t+C}\right)}} = -\frac{2\lambda}{1+\mathrm{e}^{(r-1)y_t}}. \tag{5.124}$$

Finally, we notice that, since $r \in \{0,2\}$:

$$\lambda(r-2\sigma(y_t)) = -\frac{2\lambda(1-r)}{1+\mathrm{e}^{(r-1)y_t}}. \tag{5.125}$$

Therefore:

$$\partial_t y_t = \lambda(r-2\sigma(y_t)) \tag{5.126}$$

and y_t is a solution to Equation (5.120).

Since $y_0 = (1-r)\Big(W\big(\mathrm{e}^C\big) - C\Big)$ and $z \mapsto W(\mathrm{e}^z) - z$ can be proven to be bijective over $\mathbb{R}$, the elements of S can cover any initial condition. With this, the result is proved. □

Suppose that $f_0 = 0$ in Equation (5.117) and that ρ_2 has values in $\{0,2\}$ – i.e. $\hat{\alpha}$ and $\hat{\beta}$ have disjoint supports (which is the typical case for distributions with finite support). From Proposition 5.11, a candidate solution would be:

$$f_t = \varphi_t(x)(\rho_2 - 1) = -\varphi_t(x)(\rho), \tag{5.127}$$

where:

$$\varphi_t\colon x \mapsto W\big(\mathrm{e}^{2tx+1}\big) - 2tx - 1, \tag{5.128}$$

since the initial condition $y_0 = 0$ gives the constant value $C = 1$ in Equation (5.121). Note that the Lambert W function of a symmetric linear operator is well-defined, as we choose the principal branch of the Lambert function in our case; see the work of [61] for more details. Note also that the estimation of $W(\mathrm{e}^z)$ is actually numerically stable using approximations from [124].

However, Equation (5.127) cannot be a solution of Equation (5.117). Indeed, one can prove by following essentially the same reasoning as the proof of Proposition 5.11 that:

$$\partial_t f_t = 2\left(\mathcal{T}_{k,\hat{\gamma}} \circ \left(\psi_t\big(\mathcal{T}_{k,\hat{\gamma}}\big)\right)^{-1}\right)(\rho_2 - 1), \tag{5.129}$$

with:

$$\psi_t : x \mapsto 1 + W\left(e^{2tx+1}\right) > 0. \tag{5.130}$$

However, this does not allow us to obtain Equation (5.117) since in the latter the sigmoid is taken coordinate-wise, where the exponential in Equation (5.129) acts on matrices.

Nonetheless, for t small enough, f_t as defined in Equation (5.129) should approximate the solution of Equation (5.117), since the sigmoid function is approximately linear around 0 and $f_t \approx 0$ when t is small enough. We find in practice that for reasonable values of t, e.g. $t \leq 5$, the approximate solution of Equation (5.129) is actually close to the numerical solution of Equation (5.117) obtained using an ODE solver. In other words, we have provided here a candidate approximate expression for the discriminator in the setting of the original GAN formulation – i.e., for binary classifiers. This shows the promise of the research directions hinted at here but we leave for future work a more in-depth study.

5.7 Empirical Exploration

We present in this section a few experiments spanning different losses and architectures and applied to a few different datasets. Obviously, this exploration is not supposed to be exhaustive, the aim being to validate the theoretical results obtained in the previous sections while also illustrating the power of our framework and of the proposed toolkit in studying GAN training. In particular, we argue that the infinite width limit, while certainly losing some information compared to what happens in actual finite width implementations of GANs, as used in practice, can still provide interesting and useful insights into their practical properties.

More specifically, we focus on the IPM and LSGAN losses for the discriminator since they are the two losses for which we know the analytic behavior of the discriminator in the infinite-width limit, but other losses can be studied as well in GAN(TK)^2. A large-scale empirical study of our framework is out of the scope of this paper, and is left for future work.

5.7.1 Practical Details

All experiments are performed with the proposed Generative Adversarial Neural Tangent Kernel ToolKit GAN(TK)^2 that we publicly release at `https://github.com/emited/gantk2` in the hope that the community leverages and expands it for principled GAN analyses. It is based on the JAX Neural Tangents library [204], and is convenient to evaluate novel architectures and losses based on different visualizations and analyses.

For the sake of efficiency and for these experiments only, we choose $f_0 = 0$ using the antisymmetrical initialization [292]. Indeed, in the analytical computations of the infinite-width regime, taking into account all previous discriminator states for each generator step if we were not to reinitialize to 0 after each training of the discriminator would be prohibitively costly computationally. This choice also allows us to ignore residual gradients from the initialization, which introduce noise in the optimization process.

The following lines delve further into the practicalities of our implementation and datasets.

- *GAN(TK)^2 Specifications and Computing Resources*

GAN(TK)^2 is implemented in Python (tested on versions 3.8.1 and 3.9.2) and based on JAX [42] for tensor computations and Neural Tangents [204] for NTKs. We refer to the code released at `https://github.com/emited/gantk2` for detailed specifications and instructions.

All experiments presented in this paper were run on Nvidia GPUs (Nvidia Titan RTX – 24GB of VRAM – with CUDA 11.2 as well as Nvidia Titan V – 12GB – and Nvidia GeForce RTX 2080 Ti – 11

GB – with CUDA 10.2). All two-dimensional experiments require only a few minutes of computations on a single GPU. Experiments on MNIST and CelebA were run using simultaneously four GPUs for parallel computations, for at most a couple of hours.

• *Datasets*

8 **Gaussians.** The target distribution is composed of 8 Gaussians with their means being evenly distributed on the centered sphere of radius 5, and each with a standard deviation of 0.5. The input fake distribution is drawn at initialization from a standard normal distribution $\mathcal{N}(0, 1)$. We sample in our experiments 500 points from each distribution at each run to build $\hat{\alpha}$ and $\hat{\beta}$.

AB and Density. These two datasets are taken from the Geomloss library examples [86][1] and are distributed under the MIT license. To sample a point from a distribution based on these greyscale images files, we sample a pixel (considered to lie in $[-1, 1]^2$) in the image from a distribution where each pixel probability is proportional to the darkness of this pixel, and then apply a Gaussian noise centered at the chosen pixel coordinates with a standard deviation equal to the inverse of the image size. We sample in our experiments 500 points from each distribution at each run to build $\hat{\alpha}$ and $\hat{\beta}$.

MNIST and CelebA. We preprocess each MNIST image [148] by extending it from 28×28 frames to 32×32 frames (by padding it with black pixels). CelebA images [166] are downsampled from a size of 178×218 to 32×39 and then center-cropped to 32×32.

For both datasets, we normalize pixels in the $[-1, 1]$ range. For our experiments, we consider a subset of 1024 elements of each dataset, which are randomly sampled for each run.

• *Parameters*

Sinkhorn divergence. The Sinkhorn divergence is computed using the Geomloss library [86], with a blur parameter of 0.001 and a scaling of 0.95, making it close to the Wasserstein $\mathcal{W}_2$ distance.

RBF kernel. The RBF kernel used in our experiments is the following:

$$k(x, y) = \mathrm{e}^{\frac{\|x-y\|^2}{2n}}, \tag{5.131}$$

where n is the dimension of x and y, i.e. the dimension of the data.

Architecture. We used for the neural networks of our experiments the standard NTK parameterization [128], with a scaling factor of 1 for matrix multiplications and, when bias in enabled, a multiplicative constant of 1 for biases (except for sigmoid where this bias factor is lowered to 0.2 to avoid saturating the sigmoid, and for CelebA where it is equal to 4). All considered networks are composed of 3 hidden layers and end with a linear layer. In the finite-width case, the width of these hidden layers is 128. We additionally use antisymmetrical initialization [292], except for the finite-width LSGAN loss.

Discriminator optimization. Discriminators in the finite-width regime are trained using full-batch gradient descent without momentum, with one step per update to the distributions and the following learning rates ε:

- for the IPM loss: $\varepsilon = 0.01$;

[1]They can be downloaded at `https://github.com/jeanfeydy/geomloss/tree/master/geomloss/examples/optimal_transport/data`: AB corresponds to files `A.png` (source) and `B.png` (target), and Density corresponds to files `density_a.png` (source) and `density_a.png` (target).

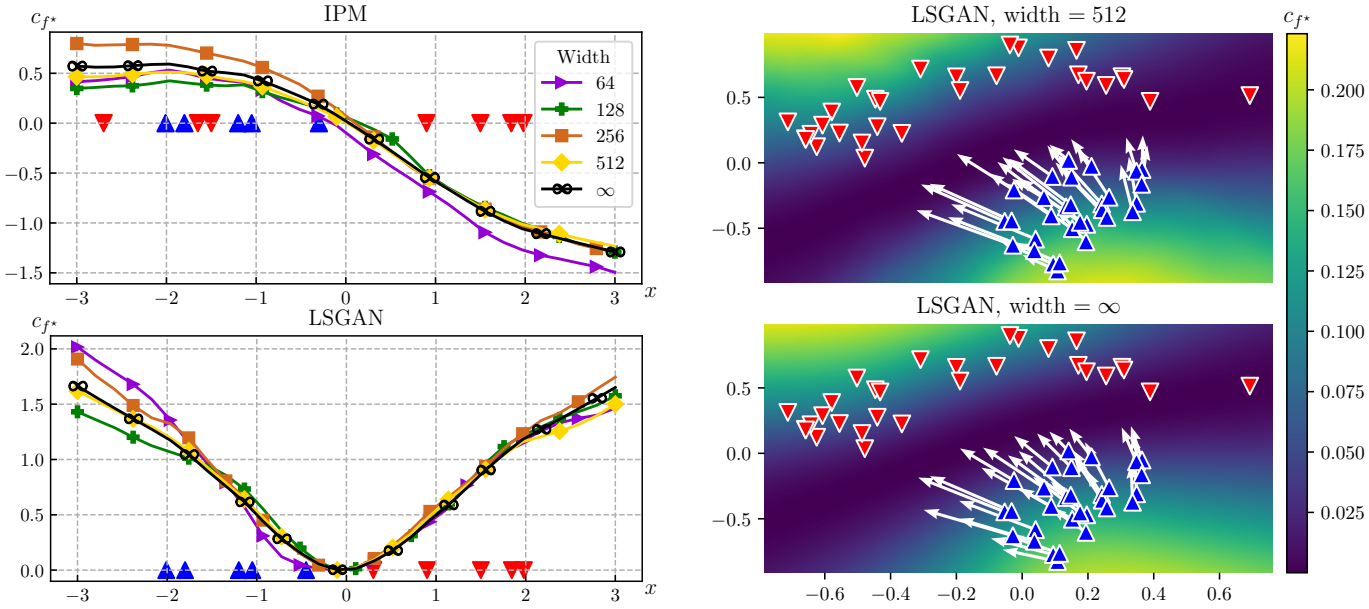


Figure 5.1: Values of $c_{f^\star}$ for LSGAN and IPM, where $f^\star$ is a 3-layer ReLU MLP with bias and varying width trained on the dataset represented by ▼ (real) and ▲ (fake) markers, initialized at $f_0 = 0$. The infinite-width network is trained for a time $\tau = 1$ and the finite-width networks using 10 gradient descent steps with learning rate $\varepsilon = 0.1$, to make training times correspond. The gradients $\nabla_x c_{f^\star}$ are shown with white arrows on the two-dimensional plots for the fake distribution.

- for the IPM loss with reset and LSGAN: $\varepsilon = 0.1$.

In the infinite-width limit, we use the analytic expression derived in Section 5.6.2 with training time $\tau = 1$ (except for MNIST and CelebA where $\tau = 1000$) and $f_0 = 0$ (through the initialization of [292]) to avoid the computational cost of accumulating discriminators' analytic expressions across the generator's optimization steps.

Point cloud descent. The multiplicative constant η over the gradient applied to each datapoint for two-dimensional problems is chosen as follows:

- for the IPM loss in the infinite-width regime: $\eta = 1000$;
- for the IPM loss in the finite-width regime: $\eta = 100$;
- for the IPM loss in the finite-width regime and discriminator reset: $\eta = 1000$;
- for LSGAN in the infinite-width regime: $\eta = 1000$;
- for LSGAN in the finite-width regime: $\eta = 1$.

We multiply η by 1000 when using sigmoid activations, because of the low magnitude of the gradients it provides. We choose for MNIST $\eta = 100$.

Training is performed for the following number of iterations:

- for 8 Gaussians: 20 000;
- for Density and AB: 10 000;
- for MNIST: 50 000.

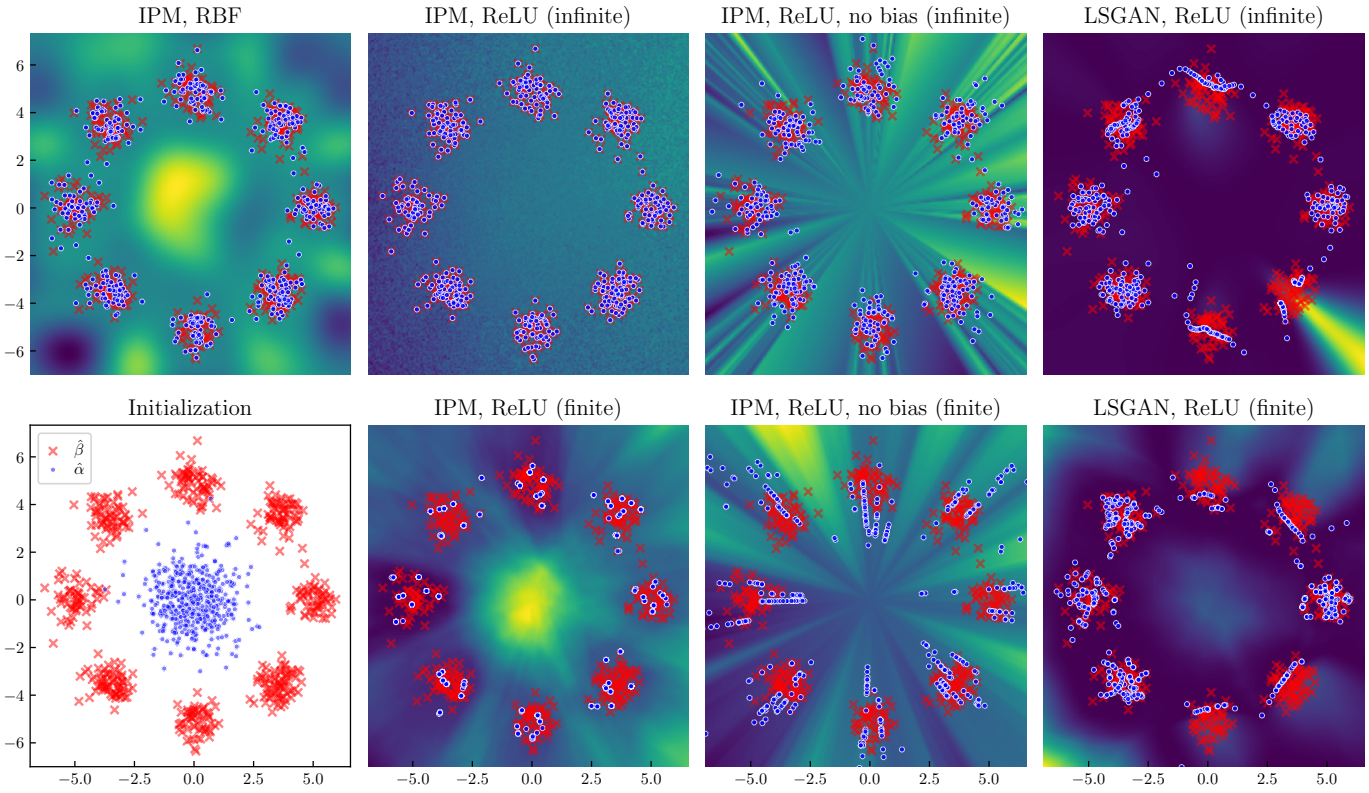


Figure 5.2: Generator (●) and target (×) samples for different methods. In the background, $c_{f^\star}$.

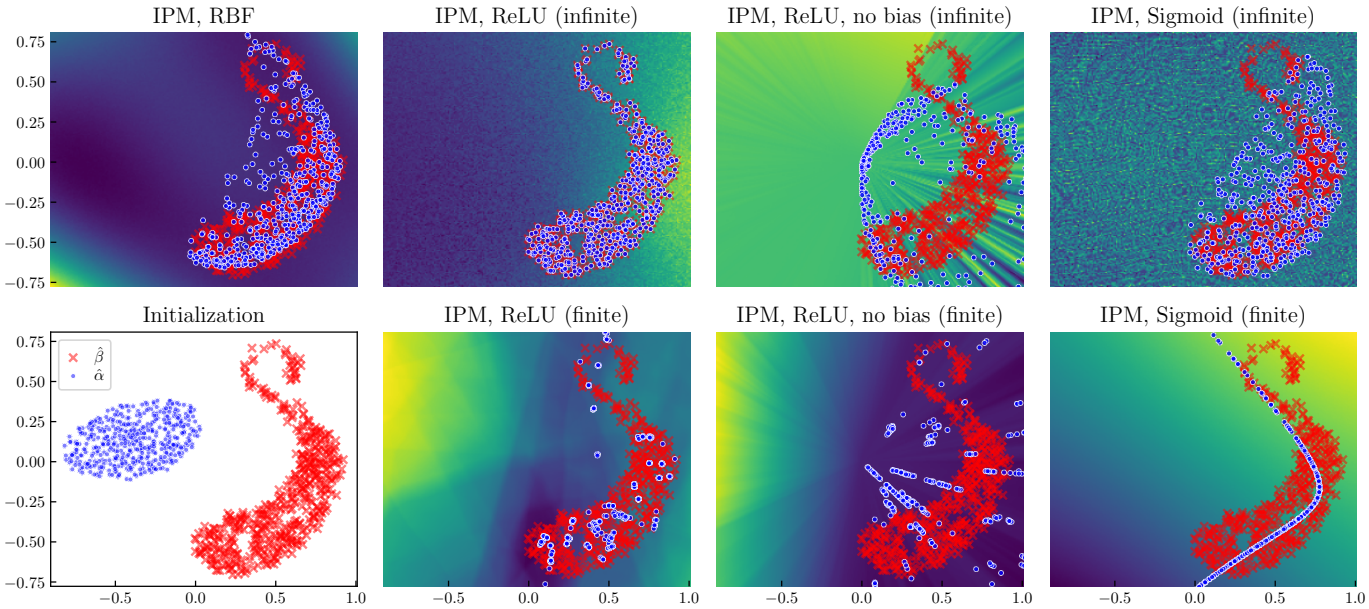


Figure 5.3: Generator (●) and target (×) samples for different methods applied to the Density problem. In the background, $c_{f^\star}$.

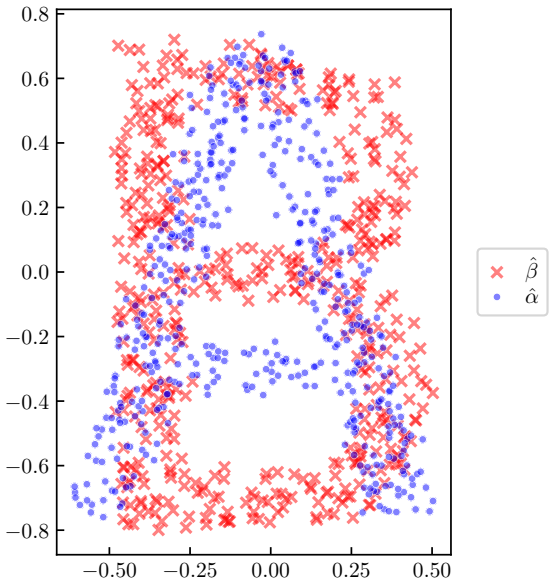


Figure 5.4: Initial generator (●) and target (×) samples for the AB problem.

5.7.2 One- and Two-Dimensional Datasets

• *A first experiment with simple, fixed distributions*

We first study the case where generated and target distributions are fixed, in order to visualize the gradient fields. In this setting, we qualitatively study the similarity between the finite- and infinite-width regimes of the discriminator. Figure 5.1 shows $c_{f^\star}$ and its gradients on one- and two-dimensional data for LSGAN and IPM losses with a ReLU MLP with 3 hidden layers of varying widths. We find the behavior of finite-width discriminators to be close to their infinite-width counterpart for standard widths, and converges rapidly to the behaviour in the infinite width limit as the width becomes larger.

• *General considerations for the 8 gaussians, AB and Density datasets*

Experimental setting. We first consider the 8 Gaussians as target distribution, as described before, as represented in Figure 5.2) and in a setup similar to that of [184], [251] and [12], as well as the more complex, in terms of shape, AB and Density datasets. We alleviate the complexity of the analysis by following Equation (5.80) with $\mathcal{T}_{k_{g_\ell},p_z} = \mathrm{id}$, similarly to [192] and [9], thereby modeling the generator's evolution by considering a finite number of samples, initially Gaussian.

For IPM and LSGAN losses, we evaluate the convergence of the generated distributions for a discriminator with ReLU activations in the finite- and infinite-width regime, either with or without bias. We also comparatively evaluate the advantages of this architecture by considering the case where the infinite-width loss is not given by an NTK, but by the popular Radial Basis Function (RBF) kernel, which is characteristic and presents attractive properties [193]. We refer to Figures 5.2 and 5.3 for qualitative results and Tables 5.1 to 5.3 for a quantitative evaluation.

Adequacy. We observe that the behaviour of the different architectures are quite similar between the finite- and infinite-width regimes, ReLU networks generally performing considerably better in the latter case. Remarkably, for the infinite-width IPM, generated and target distributions perfectly match. This can be explained by the high capacity of infinite-width networks as it has already been shown that NTKs benefit from low-data regimes [15].

Impact of bias. The bias-free discriminator performs worse than with bias, for both regimes and both losses. This is in line with findings of e.g. [24], and can be explained in our theoretical framework

Table 5.1: Sinkhorn divergence [86, lower is better, similar to $\mathcal{W}_2$] averaged over three runs between the final generated distribution and the target dataset for the 8 Gaussians problem.

Loss	RBF kernel	ReLU	ReLU (no bias)	Sigmoid
IPM (inf.)	$(2.60 \pm 0.06) \times 10^{-2}$	$(9.40 \pm 2.71) \times 10^{-7}$	$(9.70 \pm 1.88) \times 10^{-2}$	$(8.40 \pm 0.02) \times 10^{-2}$
IPM	—	$(1.21 \pm 0.14) \times 10^{-1}$	1.20 ± 0.60	$(7.40 \pm 1.30) \times 10^{-1}$
LSGAN (inf.)	$(4.21 \pm 0.10) \times 10^{-1}$	$(7.56 \pm 0.45) \times 10^{-2}$	$(1.27 \pm 0.01) \times 10^{1}$	7.35 ± 0.11
LSGAN	—	3.07 ± 0.68	7.52 ± 0.01	7.41 ± 0.54

Table 5.2: Sinkhorn divergence averaged over three runs between the final generated distribution and the target dataset for the Density problem.

Loss	RBF kernel	ReLU	ReLU (no bias)	Sigmoid
IPM (inf.)	$(2.37 \pm 0.32) \times 10^{-3}$	$(3.34 \pm 0.49) \times 10^{-9}$	$(7.34 \pm 0.34) \times 10^{-2}$	$(6.25 \pm 0.31) \times 10^{-3}$
IPM	—	$(5.02 \pm 1.19) \times 10^{-3}$	$(9.25 \pm 0.30) \times 10^{-2}$	$(3.06 \pm 0.57) \times 10^{-2}$
LSGAN (inf.)	$(7.53 \pm 0.59) \times 10^{-3}$	$(1.49 \pm 0.11) \times 10^{-3}$	$(2.80 \pm 0.03) \times 10^{-1}$	$(2.21 \pm 0.01) \times 10^{-1}$
LSGAN	—	$(1.53 \pm 1.08) \times 10^{-2}$	$(1.64 \pm 0.19) \times 10^{-1}$	$(5.88 \pm 0.80) \times 10^{-2}$

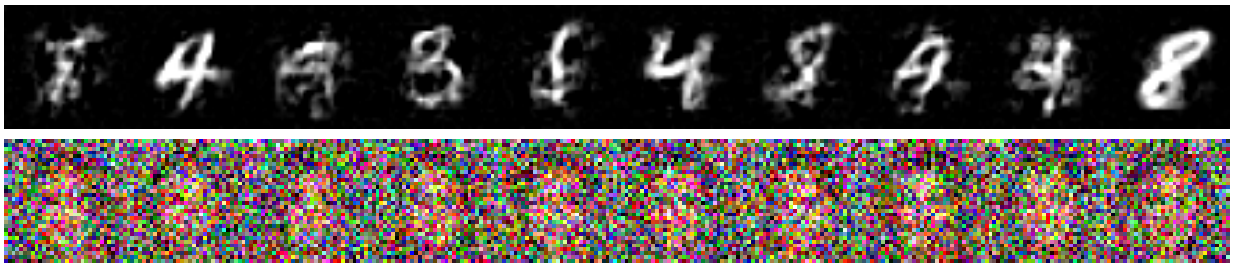
by comparing their NTKs. Indeed, the NTK of a bias-free ReLU network is not characteristic, whereas its bias counterpart was proven to present powerful approximation properties [132]. Furthermore, results of Section 5.5.2 state that the ReLU NTK with bias is differentiable at 0, whereas its bias-free version is not, which can disrupt optimization based on its gradients: note in Figures 5.2 and 5.3 the abrupt streaks of the discriminator's spatial gradient directed towards 0 which must have consequences on convergence.

NTK vs. RBF. We also notice the clear superiority, both qualitatively and quantitatively, of NTKs over the RBF kernel. Because of their similar properties with their finite regime counterparts, this tends to support the fact that the gradients of a ReLU network with bias are particularly well adapted to GANs. Visualizations of these gradients in the infinite-width limit are also available in Section 5.7.4 and further corroborate these findings. More generally, we believe that the NTK of ReLU networks could be of particular interest for kernel methods requiring the computation of a spatial gradient, such as methods based on the Stein Variational Gradient Descent [164].

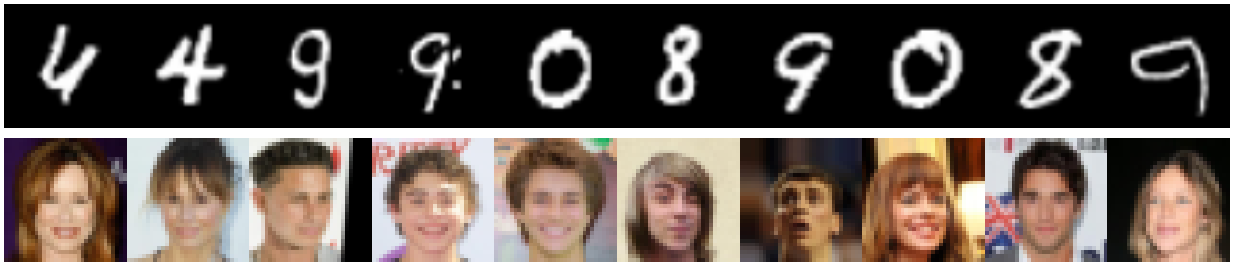
Note that the observations made here follow the same tendencies for all three datasets and are further corroborated by the experiments of the following subsections.

Table 5.3: Sinkhorn divergence averaged over three runs between the final generated distribution and the target dataset for the AB problem.

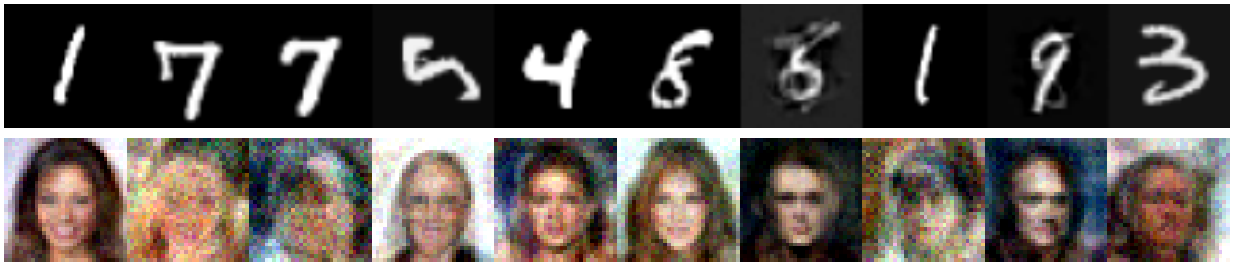
Loss	RBF kernel	ReLU	ReLU (no bias)	Sigmoid
IPM (inf.)	$(4.65 \pm 0.82) \times 10^{-3}$	$(2.64 \pm 2.13) \times 10^{-9}$	$(6.11 \pm 0.19) \times 10^{-3}$	$(5.69 \pm 0.38) \times 10^{-3}$
IPM	—	$(2.75 \pm 0.20) \times 10^{-3}$	$(3.65 \pm 1.44) \times 10^{-2}$	$(1.25 \pm 0.32) \times 10^{-2}$
LSGAN (inf.)	$(1.13 \pm 0.05) \times 10^{-2}$	$(8.63 \pm 2.24) \times 10^{-3}$	$(1.02 \pm 0.40) \times 10^{-1}$	$(1.40 \pm 0.06) \times 10^{-2}$
LSGAN	—	$(1.32 \pm 1.30) \times 10^{-1}$	$(2.57 \pm 0.73) \times 10^{-2}$	$(8.78 \pm 2.23) \times 10^{-2}$



(a) RBF kernel: blurry digits on MNIST, prohibitively noisy images on CelebA.



(b) ReLU: sharp digits on MNIST, high-quality images on CelebA.



(c) ReLU (no bias): mostly sharp digits with some artifacts and blurry images on MNIST, blurry and noisy images on CelebA.

Figure 5.5: Uncurated samples from the results of the descent of a set of 1024 particles over a subset of 1024 elements of MNIST and CelebA, starting from a standard Gaussian. Training is done using the IPM loss in the infinite-width kernel setting.

• *ReLU vs. Sigmoid Activations*

We additionally introduce a new baseline for the 8 Gaussians, Density and AB problems, where we replace the ReLU activation in the discriminator by a sigmoid-like activation $\tilde{\sigma}$, that we abbreviate to sigmoid in this experimental study for readability purposes. We choose $\tilde{\sigma}$ instead of the actual sigmoid σ for computational reasons, since $\tilde{\sigma}$, contrary to σ, allows for analytic computations of NTKs in the Neural Tangents library [204]. $\tilde{\sigma}$ is defined in the latter using the error function erf scaled in order to minimize a squared loss with respect to σ over $[-5, 5]$, with the following expression:

$$\tilde{\sigma}\colon x \mapsto \frac{1}{2}\left(\operatorname{erf}\left(\frac{x}{2.402\,056\,353\,171\,979\,6}\right) + 1\right). \tag{5.132}$$

Results are given in Tables 5.1 to 5.3 and an illustration is available in Figure 5.3. We observe that the sigmoid baseline is consistently outperformed by the RBF kernel and ReLU activation (with bias) for all regimes and losses. This is in accordance with common experimental practice, where internal sigmoid activations are found less effective than ReLU because of the potential activation saturation that they can induce.

We provide a qualitative explanation to this underperformance of sigmoid via our framework in Section 5.7.4.

5.7.3 Qualitative MNIST and CelebA Experiment

An experimental analysis of our framework on complex, large-scale, high-dimensional image datasets is out the scope of our study: it would pose many challenges, including computationally, and one would also have to use more involved architectures to be close from practice. Therefore, we leave it for future work.

Nonetheless, we present an experiment on the MNIST [148] and CelebA [166] image datasets in a setting similar to that of the experiments on two-dimensional point clouds of the previous sections: For each dataset, we make a point cloud $\hat{\alpha}$, initialized as a standard Gaussian, which we move towards a subset of the MNIST dataset following the gradients of the IPM loss in the infinite-width regime.

Qualitative results are presented in Figure 5.5.

Our conclusions are consistent with what has been observed before. We notice, similarly to the two-dimensional experiments, that the ReLU network with bias outperforms its bias-free counterpart and a standard RBF kernel in terms of sample quality. The difference between the RBF kernel and ReLU NTK is even more flagrant in this complex high-dimensional setting, as the RBF kernel is unable to produce reasonably good samples.

5.7.4 Visualizing the Gradient Field Induced by the Discriminator

We raise in Sections 5.6 and 5.6.2 the open problem of studying the convergence of the generated distribution towards the target distribution with respect to the gradients of the discriminator. The aim in this section is to study qualitatively these gradients in a simplified case that could shed some light on the more general setting and explain some of our experimental results. These gradient fields can be plotted using the provided GAN(TK)^2 toolkit.

• *Setting*

Since we study gradients of the discriminator expressed in Equation (5.10), we assume that $f_0 = 0$ – for instance, using the anti-symmetrical initialization [292] – in order to ignore residual gradients from the initialization.

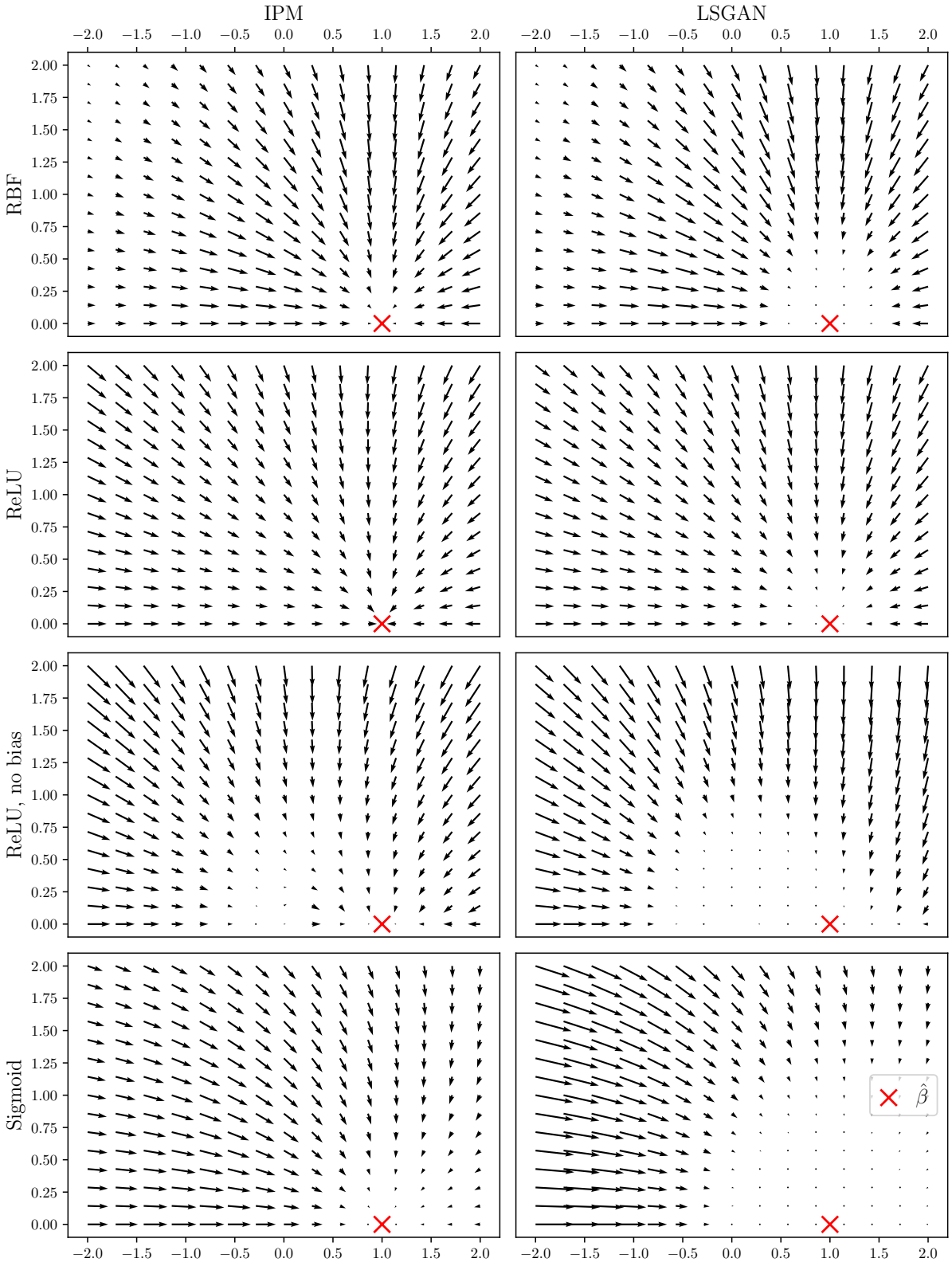


Figure 5.6: Gradient field $\nabla c_{f^\star_{\hat{\alpha}_x}}(x)$ received by a generated sample $x \in \mathbb{R}^2$ (i.e. $\hat{\alpha} = \hat{\alpha}_x = \delta_x$) initialized to x_0 with respect to its coordinates in $\mathrm{Span}\{x_0, y\}$ where y, marked by a $\times$, is the target distribution (i.e. $\hat{\beta} = \delta_y$), with $\|y\| = 1$. Arrows correspond to the movement of x in $\mathrm{Span}\{x_0, y\}$ following $\nabla c_{f^\star_{\hat{\alpha}_x}}(x)$, for different losses and networks; scales are specific for each pair of loss and network. The ideal case is the convergence of x along this gradient field towards the target y. Note that in the chosen orthonormal coordinate system, without loss of generality, y has coordinate $(1, 0)$; moreover, the gradient field is symmetrical with respect to the horizontal axis.

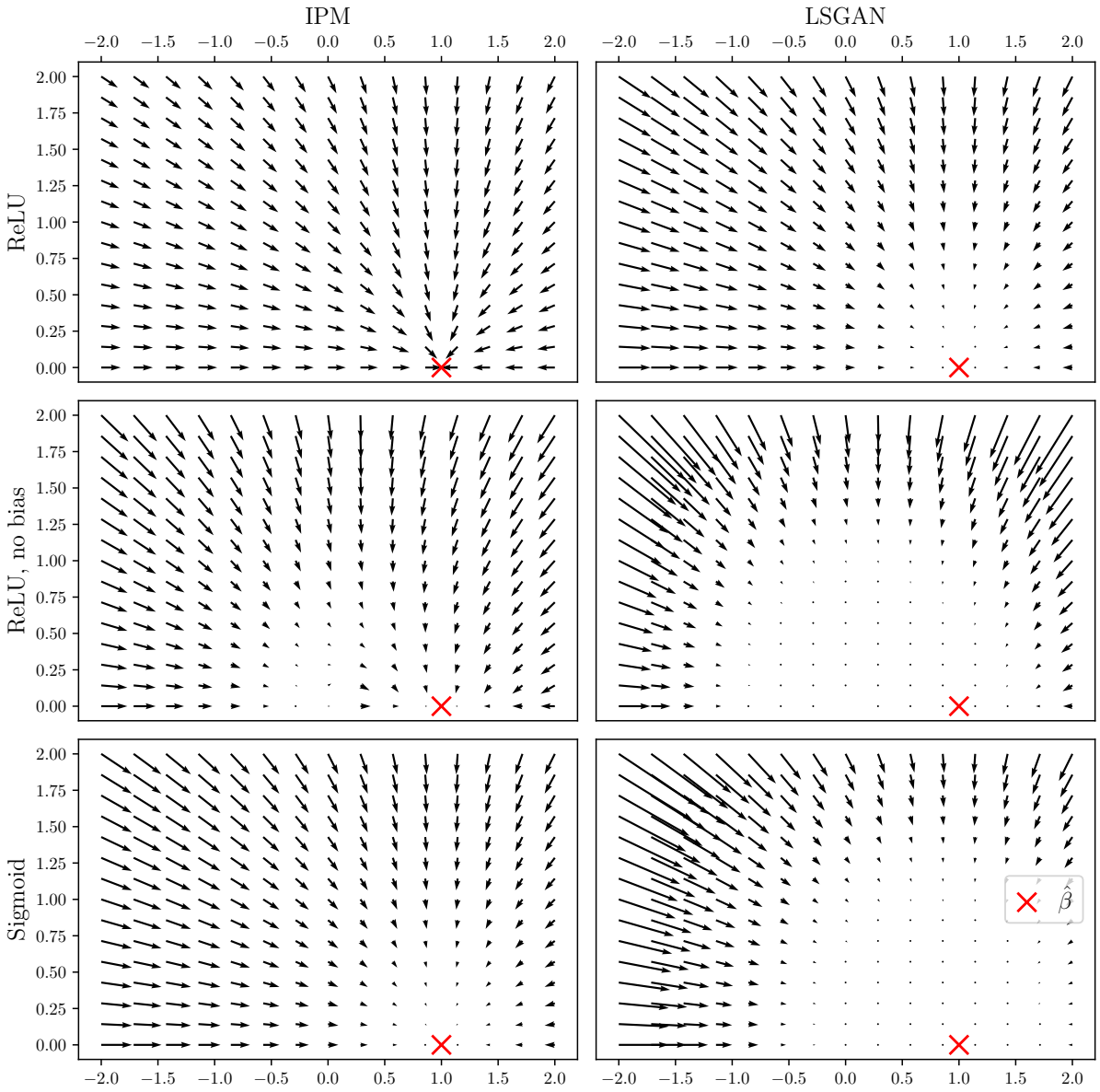


Figure 5.7: Same plot as Figure 5.6 but with underlying points $x, y \in \mathbb{R}^{512}$.

By Theorem 5.1, for any loss and any training time, the discriminator can be expressed as $f^\star_{\hat\alpha} = \mathcal{T}_{k,\hat\gamma}(h_0)$, for some $h_0 \in L^2(\hat\gamma)$. Thus, there exists $h_1 \in L^2(\hat\gamma)$ such that:

$$f^\star_{\hat\alpha} = \sum_{x \in \operatorname{supp}\hat\gamma} h_1(x) k(x, \cdot). \tag{5.133}$$

Consequently,

$$\nabla f^\star_{\hat\alpha} = \sum_{x \in \operatorname{supp}\hat\gamma} h_1(x) \nabla k(x, \cdot), \qquad \nabla c_{f^\star_{\hat\alpha}} = \sum_{x \in \operatorname{supp}\hat\gamma} h_1(x) \nabla k(x, \cdot) c'(f^\star_{\hat\alpha}(\cdot)). \tag{5.134}$$

Dirac-GAN setting. The latter linear combination of gradients indicates that, by examining gradients of $c_{f^\star_{\hat\alpha}}$ for pairs of $(x, y) \in \operatorname{supp}\hat\alpha \times \operatorname{supp}\hat\beta$, one could already develop potentially valid intuitions that can hold even when multiple points are considered. This is especially the case for the IPM loss, as h_0, h_1 have a simple form: $h_1(x) = 1$ if $x \in \operatorname{supp}\hat\alpha$ and $h_1(y) = -1$ if $y \in \operatorname{supp}\hat\beta$ (assuming points from $\hat\alpha$ and $\hat\beta$ are uniformly weighted); moreover, note that $c'(f^\star_{\hat\alpha}(\cdot)) = 1$. Thus, we study here $\nabla c_{f^\star_{\hat\alpha}}$ when $\hat\alpha$ and $\hat\beta$ are only comprised of one point, i.e. the setting of Dirac GAN [182], with $\hat\alpha = \delta_x \triangleq \hat\alpha_x$ and $\hat\beta = \delta_y$.

Visualizing high-dimensional inputs. Unfortunately, the gradient field is difficult to visualize when the samples live in a high-dimensional space. Interestingly, the NTK $k(x, y)$ for any architecture starting with a fully connected layer only depends on $\|x\|$, $\|y\|$ and $\langle x, y\rangle$ [283], and therefore all the information of $\nabla c_{f^\star_{\hat\alpha}}$ is contained in $\operatorname{Span}\{x, y\}$. From this, we show in Figures 5.6 and 5.7 the gradient field $\nabla c_{f^\star_{\hat\alpha}}$ in the two-dimensional space $\operatorname{Span}\{x, y\}$ for different architectures and losses in the infinite-width regime described in Section 5.7 and in this section. Figure 5.6 corresponds to two-dimensional $x, y \in \mathbb{R}^2$, and Figure 5.7 to high-dimensional $x, y \in \mathbb{R}^{512}$. Note that in the plots, the gradient field is symmetric w.r.t. the horizontal axis and for this reason we have restricted the illustration to the case where the second coordinate is positive.

Convergence of the gradient flow. In the last paragraph, we have seen that the gradient field in the Dirac-GAN setting lives in the two-dimensional $\operatorname{Span}\{x, y\}$, independently of the dimensionality of x, y. This means that when training the generated distribution, as in Section 5.7, the position of the particle x always remains in this two-dimensional space, and hence (non-)convergence in this setting can be easily visualized by examining the corresponding two-dimensional gradient field. This is what we do in the following, for different architectures and losses.

• *Qualitative Analysis of the Gradient Field*

x is far from y. When generated outputs are far away from the target, it is essential that their gradient has a large enough magnitude in order to pull these points towards the target. The behavior of the gradients for distant points can be observed in the plots. For ReLU networks, for both losses, the gradients for distant points seem to be well behaved and large enough. Note that in the IPM case, the magnitude of the gradients is even larger when x is further away from y. This is not the case for the RBF kernel when the variance parameter is too small, as the magnitude of the gradient becomes prohibitively small. We highlight that we selected a large variance parameter in order to avoid such a behavior, but diminishing magnitudes can still be observed. Note that choosing an overly large variance may also have a negative impact on the points that are closer to the target, in the more general setting where the target distributions has more than a single point.

x is close to y. A particularity of the NTK of ReLU discriminators with bias that arises from this study is that the gradients vanish more slowly when the generated x becomes close from the

target y, compared to NTKs of ReLU without bias and sigmoid networks, as well as to the RBF kernel. We hypothesize that this is also another distinguishing feature that helps the generated distribution in converging more easily to the target distribution, especially when they are not far apart. On the contrary, this gradient vanishes more rapidly for NTKs of ReLU without bias and sigmoid networks, compared to the RBF kernel. This can explain the worse performance of such NTKs when compared to the RBF kernel in our experiments (see previously and Tables 5.1 to 5.3). Note that this phenomenon is even more pronounced in high-dimensional spaces such as in Figure 5.7.

x **is close to** 0**.** Finally, we highlight gradient vanishing and instabilities around the origin for ReLU networks without bias. This is related to its differentiability issues at the origin exposed in Section 5.5.2, and to its lack of representational power discussed in Section 5.8.4. This can also be retrieved on the larger scale experiments of Figures 5.2 and 5.3 where the origin is the source of instabilities in the descent.

Sigmoid network. It is also possible to evaluate the properties of the discriminator's gradient for architectures that are not used in practice, such as networks using the sigmoid activation. Figures 5.2 and 5.3 provide a clear explanation: as stated above, the magnitudes of the gradients become too small when $x \to y$, and heavily depend on the direction from which x approaches y. Ideally, the induced gradient flow should be insensitive to the direction in order for the convergence to be reliable and robust, which seems to be the case for ReLU networks.

5.8 Further Discussions and Remarks

We develop in this section some additional remarks and explanations.

5.8.1 Loss of the Generator and its Gradient

We highlight in this section the importance of taking into account discriminator gradients in the optimization of the generator. Let us focus on an example similar to the one of [12, Example 1] and choose as β a single Dirac centered at 0 and as $\alpha_g = \alpha_\theta$ single Dirac centered at $x_\theta = \theta$ (the generator parameters being the coordinates of the generated point). Let us focus for the sake of simplicity on the case of LSGAN since it is a recurring example in this work, but a similar reasoning can be done for other GAN instances.

In the theoretical min-max formulation of GANs considered by [12], the generator is trained to minimize the following quantity:

$$\mathcal{C}_{f^\star_{\alpha_\theta}}(\alpha_\theta) \triangleq \mathbb{E}_{x\sim\alpha_\theta}\Big[c_{f^\star_{\alpha_\theta}}(x)\Big] = f^\star_{\alpha_\theta}(x_\theta)^2, \tag{5.135}$$

where:

$$\begin{aligned} f^\star_{\alpha_\theta} &= \underset{f\in L^2\left(\frac{1}{2}\alpha_\theta+\frac{1}{2}\beta\right)}{\arg\max} \Big\{\mathcal{L}_{\alpha_\theta}(f) \triangleq \mathbb{E}_{x\sim\alpha_\theta}\Big[a_f(x)\Big] - \mathbb{E}_{y\sim\beta}\Big[b_f(y)\Big]\Big\} \\ &= \underset{f\in L^2\left(\frac{1}{2}\alpha_\theta+\frac{1}{2}\beta\right)}{\arg\min} \Big\{\Big(f^\star_{\alpha_\theta}(x_\theta)+1\Big)^2 + \Big(f^\star_{\alpha_\theta}(0)-1\Big)^2\Big\}. \end{aligned} \tag{5.136}$$

Consequently, $f^\star_{\alpha_\theta}(0) = 1$ and $f^\star_{\alpha_\theta}(x_\theta) = -1$ when $x_\theta \neq 0$, thus in this case:

$$\mathcal{C}_{f^\star_{\alpha_\theta}}(\alpha_\theta) = 1. \tag{5.137}$$

This constancy of the generator loss would make it impossible to be learned by gradient descent, as pointed out by [12].

However, the setting does not correspond to the actual optimization process used in practice and represented by Equation (5.3). We do have $\nabla_\theta \mathcal{C}_{f^\star_{\alpha_\theta}}(\alpha_\theta) = 0$ when $x_\theta \neq 0$, but the generator never uses this gradient in standard GAN optimization. Indeed, this gradient takes into account the dependency of the optimal discriminator $f^\star_{\alpha_\theta}$ in the generator parameters, since the optimal discriminator depends on the generated distribution. Yet, in practice and with few exceptions such as Unrolled GANs [184] and as done in Equation (5.3), this dependency is ignored when computing the gradient of the generator, because of the alternating optimization setting – where the discriminator is trained in-between generator's updates. Therefore, despite being constant on the training data, this loss can yield non-zero gradients to the generator. However, this requires the gradient of $f^\star_{\alpha_\theta}$ to be defined, which is the issue addressed in Section 5.3.2.

5.8.2 Differentiability of the Bias-Free ReLU Kernel

Remark 5.2 contradicts the results of [38] on the regularity of the NTK of a bias-free ReLU MLP with one hidden layer, which can be expressed as follows (up to a constant scaling the matrix multiplication in linear layers):

$$k(x,y) = \|x\|\|y\|\kappa\left(\frac{\langle x,y\rangle}{\|x\|\|y\|}\right), \tag{5.138}$$

where:

$$\begin{aligned}\kappa\colon [0,1] &\to \mathbb{R}\\ u &\mapsto \frac{2}{\pi}u(\pi - \arccos u) + \frac{1}{\pi}\sqrt{1-u^2}\end{aligned}. \tag{5.139}$$

More particularly, [38, Proposition 3] claim that $k(\cdot, y)$ is not Lipschitz around y for all y in the unit sphere. By following their proof, it amounts to prove that $k(\cdot, y)$ is not Lipschitz around y for all y in any centered sphere. We highlight that this also contradicts empirical evidence, as we did observe the Lipschitzness of such NTK in practice using the Neural Tangents library [204].

We believe that the mistake in the proof of [38] lies in the confusion between functions κ and $k_0\colon x, y \mapsto \kappa\left(\frac{\langle x,y\rangle}{\|x\|\|y\|}\right)$, which have different geometries. Their proof relies on the fact that κ is indeed non-Lipschitz in the neighborhood of $u = 1$. However, this does not imply that k_0 is not Lipschitz, or not derivable. We can prove that it is actually at least locally Lipschitz.

Indeed, let us compute the following derivative for $x \neq y \in \mathbb{R}^n \setminus \{0\}$:

$$\frac{\partial k_0(x,y)}{\partial x} = \frac{y\|x\| - \frac{x}{\|x\|}\langle x,y\rangle}{\|x\|^2\|y\|}\kappa'(u) = \frac{1}{\|x\|\|y\|}\left(y - \langle x,y\rangle\frac{x}{\|x\|^2}\right)\kappa'(u), \tag{5.140}$$

where $u = \frac{\langle x,y\rangle}{\|x\|\|y\|}$ and:

$$\pi\cdot\kappa'(u) = \frac{u}{\sqrt{1-u^2}} + 2(\pi - \arccos u). \tag{5.141}$$

Note that $\kappa'(u) \sim_{u\to 1^-} \frac{\pi u}{\sqrt{1-u^2}} \sim_{u\to 1^-} \frac{\pi}{\sqrt{2}\sqrt{1-u}}$. Therefore:

$$\begin{aligned}\frac{\pi}{\sqrt{2}}\cdot\frac{\partial k_0(x,y)}{\partial x} &\sim_{x\to y} \frac{1}{\|y\|^2}\left(y - \langle x,y\rangle\frac{x}{\|x\|^2}\right)\frac{\sqrt{\|x\|\|y\|}}{\sqrt{\|x\|\|y\| - \langle x,y\rangle}}\\ &\sim_{x\to y} \frac{\|x\|^2 y - \langle x,y\rangle x}{\|y\|^3\sqrt{\|x\|\|y\| - \langle x,y\rangle}}\\ &\sim_{x\to y} \frac{\|y\|^2 - \langle x,y\rangle}{\|y\|^3\sqrt{\|y\|^2 - \langle x,y\rangle}}y \xrightarrow[x\to y]{} 0,\end{aligned} \tag{5.142}$$

which proves that k_0 is actually Lipschitz around points (y, y), as well as differentiable, and confirms our remark.

5.8.3 Integral Operator and Instance Noise

Instance noise [254] consists in adding random Gaussian noise to the input and target samples. This amounts to convolving the data distributions with a Gaussian density, which will have the effect of smoothing the discriminator. In the following, for the case of IPM losses, we link instance noise with our framework, showing that smoothing of the data distributions already occurs via the NTK kernel, stemming from the fact that the discriminator is a neural network trained with gradient descent.

More specifically, it can be shown that if k is an RBF kernel, the optimal discriminators in both case are the same. This is based on the fact that the density of a convolution of an empirical measure $\hat{\mu} = \frac{1}{N}\sum_i \delta_{x_i}$, where δ_z is the Dirac distribution centered on z, and a Gaussian density $\tilde{k}$ with associated RBF kernel k can be written as $\tilde{k} * \hat{\mu} = \frac{1}{N}\sum_i k(x_i, \cdot)$.

Let us consider the following regularized discriminator optimization problem in $L^2(\mathbb{R})$ smoothed from $L^2(\Omega)$ with instance noise, i.e. convolving $\hat{\alpha}$ and $\hat{\beta}$ with $\tilde{k}$.

$$\sup_{f \in L^2(\mathbb{R})} \left\{ \mathcal{L}^{\tilde{k}}_{\hat{\alpha}}(f) \triangleq \mathbb{E}_{x \sim \tilde{k} * \hat{\alpha}}[f(x)] - \mathbb{E}_{y \sim \tilde{k} * \hat{\beta}}[f(y)] - \lambda \|f\|^2_{L^2} \right\} \tag{5.143}$$

The optimum f^{IN} can be found by taking the gradient:

$$\nabla_f \left(\mathcal{L}^{\tilde{k}}_{\hat{\alpha}}\left(f^{\text{IN}}\right) - \lambda \left\| f^{\text{IN}} \right\|^2_{L^2} \right) = 0 \qquad \Leftrightarrow \qquad f^{\text{IN}} = \frac{1}{2\lambda}\left(\tilde{k} * \hat{\alpha} - \tilde{k} * \hat{\beta} \right). \tag{5.144}$$

If we now study the resolution of the optimization problem in $\mathcal{H}^{\hat{\gamma}}_k$ as in Section 5.6.2 with $f_0 = 0$, we find the following discriminator:

$$f_t = t\Big(\mathbb{E}_{x \sim \hat{\alpha}}[k(x, \cdot)] - \mathbb{E}_{y \sim \hat{\beta}}[k(y, \cdot)] \Big) = t\Big(\tilde{k} * \hat{\alpha} - \tilde{k} * \hat{\beta} \Big). \tag{5.145}$$

Therefore, we have that $f^{\text{IN}} \propto f_t$, i.e. instance noise and regularization by neural networks obtain the same smoothed solution.

This analysis was done using the example of an RBF kernel, but it also holds for stationary kernels, i.e. $k(x, y) = \tilde{k}(x - y)$, which can be used to convolve measures. We remind that this is relevant, given that NTKs are stationary over spheres [128, 283], around where data can be concentrated in high dimensions.

5.8.4 Positive Definite NTKs

Optimality results in the theory of NTKs usually rely on the assumption that the considered NTK k is positive definite over the training dataset $\hat{\gamma}$ [128, 292]. This property offers several theoretical advantages.

Indeed, this gives sufficient representational power to its RKHS to include the optimal solution over $\hat{\gamma}$. Moreover, this positive definiteness property equates for finite datasets to the invertibility of the mapping

$$\begin{aligned} \mathcal{T}_{k,\hat{\gamma}}\big|_{\operatorname{supp}\hat{\gamma}} \colon L^2(\hat{\gamma}) &\to L^2(\hat{\gamma}) \\ h &\mapsto \mathcal{T}_{k,\hat{\gamma}}(h)\big|_{\operatorname{supp}\hat{\gamma}} \end{aligned}, \tag{5.146}$$

that can be seen as a multiplication by the invertible Gram matrix of k over $\hat{\gamma}$. From this, one can retrieve the expression of $f \in \mathcal{H}^{\hat{\gamma}}_k$ from its restriction $f|_{\operatorname{supp}\hat{\gamma}}$ to $\operatorname{supp}\hat{\gamma}$ in the following way:

$$f = \mathcal{T}_{k,\hat{\gamma}} \circ \mathcal{T}_{k,\hat{\gamma}}\big|^{-1}_{\operatorname{supp}\hat{\gamma}} \left(f|_{\operatorname{supp}\hat{\gamma}} \right), \tag{5.147}$$

as shown in Lemma 5.10. Finally, as shown by [128] and in Section 5.6.3, this makes the discriminator loss function strictly increase during training.

One may wonder whether this assumption is reasonable for NTKs. [128] proved that it indeed holds for NTKs of non-shallow MLPs with non-polynomial activations if data is supported on the unit sphere, supported by the fact that the NTK is stationary over the unit sphere. Others, such as [82], have observed positive definiteness of the NTK subject to specific assumptions on the networks and data. We are not aware of more general results of this kind. However, one may conjecture that, at least for specific kinds of networks, NTKs are positive definite for any training data.

Indeed, besides global convergence results [4], prior work indicates that MLPs are universal approximators [119, 154]. This property can be linked in our context to universal kernels [252], which are guaranteed to be positive definite over any training data [249]. Universality is linked to the density of the kernel RKHS in the space of continuous functions. In the case of NTKs, previously cited approximation properties can be interpreted as signs of expressive RKHSs, and thus support the hypothesis of universal NTKs. Furthermore, beyond positive definiteness, universal kernels are also characteristic [249], which is interesting when they are used to compute MMDs, as we do in Section 5.6.2. Note that for the standard case of ReLU MLPs, [132] showed universal approximation results in the infinite-width regime, and works such as the one of [49] observed that their RKHS is close to the one of the Laplace kernel, which is positive definite.

Bias-free ReLU NTKs are not characteristic. As already noted by [154], the presence of bias is important when it comes to representational power of MLPs. We can retrieve this observation in our framework. In the case of a ReLU shallow network with one hidden layer and without bias, [38] determine its associated NTK as follows (up to a constant scaling the matrix multiplication in linear layers):

$$k(x, y) = \|x\|\|y\|\kappa\left(\frac{\langle x, y\rangle}{\|x\|\|y\|}\right), \tag{5.148}$$

with in particular $k(x, 0) = 0$ for all $x \in \Omega$; suppose that $0 \in \Omega$. This expression of the kernel implies that k is not positive definite for all datasets: take for example $x = 0$ and $y \in \Omega \setminus \{0\}$; then the Gram matrix of k has a null row, hence k is not strictly positive definite over $\{x, y\}$. Another consequence is that k is not characteristic. Indeed, take probability distributions $\mu = \delta_{\frac{y}{2}}$ and $\nu = \frac{1}{2}\left(\delta_x + \delta_y\right)$ with δ_z being the Dirac distribution centered on $z \in \Omega$, and where $x = 0$ and $y \in \Omega \setminus \{0\}$. Then:

$$\mathbb{E}_{z\sim\mu}k(z, \cdot) = k\left(\frac{1}{2}y, \cdot\right) = \frac{1}{2}k(y, \cdot) = \frac{1}{2}(k(y, \cdot) + k(x, \cdot)) = \mathbb{E}_{z\sim\nu}k(z, \cdot), \tag{5.149}$$

i.e., kernel embeddings of μ and $\nu \neq \mu$ are identical, making k not characteristic by definition.

5.9 Conclusion

Leveraging the theory of infinite-width neural networks, we have proposed in this final chapter a framework for the analysis of GANs which explicitly models important aspects of the choices involved in training a large variety of discriminator architectures. We thus show that the proposed framework models more accurately GAN training compared to prior approaches by deriving properties of the trained discriminator. We demonstrate the analysis opportunities of the proposed modelization by further studying the generated distribution for specific GAN losses and architectures, both theoretically and empirically, notably using our public GAN analysis toolkit. We believe that this work will serve as a basis for more elaborate analyses, thus leading to more principled, better GAN models.

Moreover, this chapter constitutes a natural extension of the previous one as it not only considers the training of the DNNs and its role in determining the function which is actually learned but also explicitly models the choice of architecture and activation by the way of the NTK of the discriminator.

Bibliography

[1] Robert J. Adler. *The Geometry Of Random Fields*. Society for Industrial and Applied Mathematics, dec 1981. (Cited on page 156.)

[2] Robert J. Adler. An introduction to continuity, extrema, and related topics for general gaussian processes. *Lecture Notes-Monograph Series*, 12:i–155, 1990. (Cited on page 157.)

[3] Sina Alemohammad, Zichao Wang, Randall Balestriero, and Richard G. Baraniuk. The recurrent neural tangent kernel. In *International Conference on Learning Representations*, 2021. (Cited on page 145.)

[4] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A convergence theory for deep learning via over-parameterization. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 242–252. PMLR, jun 2019. (Cited on page 186.)

[5] Mauricio A. Alvarez, David Luengo, and Neil D. Lawrence. Linear latent force models using gaussian processes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(11):2693–2705, nov 2013. (Cited on pages 26 and 27.)

[6] Luigi Ambrosio and Gianluca Crippa. Continuity equations and ODE flows with non-smooth velocity. *Proceedings of the Royal Society of Edinburgh: Section A Mathematics*, 144(6):1191–1244, 2014. (Cited on pages 109 and 163.)

[7] Luigi Ambrosio, Nicola Gigli, and Giuseppe Savare. *Gradient Flows in Metric Spaces and in the Space of Probability Measures*. Birkhäuser Basel, 2005. (Cited on page 112.)

[8] Luigi Ambrosio, Nicola Gigli, and Giuseppe Savaré. *Gradient Flows*. Birkhäuser Basel, Basel, Switzerland, 2008. (Cited on pages 163 and 167.)

[9] Michael Arbel, Anna Korba, Adil Salim, and Arthur Gretton. Maximum mean discrepancy gradient flow. In Hanna Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché Buc, Emily Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32, pages 6484–6494. Curran Associates, Inc., 2019. (Cited on pages 163, 166, 167, and 176.)

[10] Martin Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks. In *International Conference on Learning Representations*, 2017. (Cited on pages 144, 146, and 156.)

[11] Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant Risk Minimization. *arXiv:1907.02893 [cs, stat]*, mar 2020. arXiv: 1907.02893. (Cited on page 77.)

[12] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International*

Conference on Machine Learning, volume 70 of *Proceedings of Machine Learning Research*, pages 214–223. PMLR, aug 2017. (Cited on pages 146, 166, 176, and 183.)

[13] Vladimir Arnold. Sur la géométrie différentielle des groupes de Lie de dimension infinie et ses applications à l'hydrodynamique des fluides parfaits. *Annales de l'Institut Fourier*, 16(1):319–361, 1966. (Cited on page 139.)

[14] Sanjeev Arora, Simon S. Du, Wei Hu, Zhiyuan Li, Ruslan Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. In Hanna Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché Buc, Emily Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32, pages 8141–8150. Curran Associates, Inc., 2019. (Cited on pages 145, 148, and 161.)

[15] Sanjeev Arora, Simon S. Du, Zhiyuan Li, Ruslan Salakhutdinov, Ruosong Wang, and Dingli Yu. Harnessing the power of infinitely wide deep nets on small-data tasks. In *International Conference on Learning Representations*, 2020. (Cited on pages 145 and 176.)

[16] Sanjeev Arora, Rong Ge, Yingyu Liang, Tengyu Ma, and Yi Zhang. Generalization and equilibrium in generative adversarial nets (GANs). In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 224–232. PMLR, aug 2017. (Cited on page 146.)

[17] 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. (Cited on page 60.)

[18] Ibrahim Ayed, Emmanuel de Bézenac, Arthur Pajot, Julien Brajard, and Patrick Gallinari. Learning dynamical systems from partial observations. *arXiv preprint arXiv:1902.11136*, 2019. (Cited on page 63.)

[19] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. 2015. (Cited on page 106.)

[20] Yu Bai, Tengyu Ma, and Andrej Risteski. Approximability of discriminators implies diversity in GANs. In *International Conference on Learning Representations*, 2019. (Cited on page 145.)

[21] Yogesh Balaji, Mohammadmahdi Sajedi, Neha Mukund Kalibhat, Mucong Ding, Dominik Stöger, Mahdi Soltanolkotabi, and Soheil Feizi. Understanding over-parameterization in generative adversarial networks. In *International Conference on Learning Representations*, 2021. (Cited on page 145.)

[22] Aayush Bansal, Shugao Ma, Deva Ramanan, and Yaser Sheikh. Recycle-gan: Unsupervised video retargeting. 2018. (Cited on page 105.)

[23] 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. (Cited on pages 83 and 84.)

[24] Ronen Basri, Meirav Galun, Amnon Geifman, David Jacobs, Yoni Kasten, and Shira Kritchman. Frequency bias in neural networks for input of non-uniform density. In Hal Daumé, III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 685–694. PMLR, jul 2020. (Cited on pages 145 and 176.)

[25] Jonathan Baxter. A model of inductive bias learning. *J. Artif. Int. Res.*, 12(1):149–198, mar 2000. (Cited on pages 79, 80, 81, and 83.)

[26] Philipp Becker, Harit Pandya, Gregor Gebhardt, Cheng Zhao, James Taylor, and Gerhard Neumann. Recurrent kalman networks: Factorized inference in high-dimensional deep feature spaces. *International Conference on Machine Learning (ICML)*, 2019. (Cited on page 60.)

[27] Eugene Belilovsky, Michael Eickenberg, and Edouard Oyallon. Greedy layerwise learning can scale to imagenet. In *ICML*, 2019. (Cited on page 138.)

[28] Eugene Belilovsky, Michael Eickenberg, and Edouard Oyallon. Decoupled greedy learning of cnns. In *ICML*, 2020. (Cited on pages 136, 137, and 138.)

[29] Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *PNAS*, 2019. (Cited on pages 105 and 113.)

[30] Mikhail Belkin, Siyuan Ma, and Soumik Mandal. To understand deep learning we need to understand kernel learning. In *35th International Conference on Machine Learning*, 2018. (Cited on pages 106 and 113.)

[31] Sagie Benaim, Tomer Galanti, and Lior Wolf. Estimating the success of unsupervised image to image translation. 2018. (Cited on page 107.)

[32] Sagie Benaim and Lior Wolf. One-sided unsupervised domain mapping. 2017. (Cited on page 107.)

[33] J.D. Benamou and Y. Brenier. A computational fluid mechanics solution to the monge-kantorovich mass transfer problem. *Numerische Mathematik*, 2000. (Cited on page 110.)

[34] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. NIPS'15, pages 1171–1179, Cambridge, MA, USA, 2015. MIT Press. (Cited on pages 46 and 49.)

[35] Dominique Béréziat and Isabelle Herlin. *Coupling Dynamic Equations and Satellite Images for Modelling Ocean Surface Circulation*, pages 191–205. Springer International Publishing, Cham, 2015. (Cited on page 27.)

[36] Dimitri P. Bertsekas. *Constrained Optimization and Lagrange Multiplier Methods (Optimization and Neural Computation Series)*. Athena Scientific, 1 edition, 1996. (Cited on page 58.)

[37] Alberto Bietti and Francis Bach. Deep equals shallow for ReLU networks in kernel regimes. In *International Conference on Learning Representations*, 2021. (Cited on page 145.)

[38] Alberto Bietti and Julien Mairal. On the inductive bias of neural tangent kernels. In Hanna Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché Buc, Emily Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32, pages 12893–12904. Curran Associates, Inc., 2019. (Cited on pages 145, 156, 184, and 186.)

[39] 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*, volume 97 of *Proceedings of Machine Learning Research*, pages 664–674, Long Beach, California, USA, 09–15 Jun 2019. PMLR. (Cited on page 89.)

[40] M Bocquet. Parameter-field estimation for atmospheric dispersion: Application to the Chernobyl accident using 4D-Var. *Quarterly Journal of the Royal Meteorological Society*, 138(664):664–681, 2012. (Cited on page 20.)

[41] François Bolley. Separability and completeness for the wasserstein distance. In *Séminaire de Probabilités XLI*. Springer, Berlin, Heidelberg, 2008. (Cited on page 121.)

[42] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. (Cited on page 172.)

[43] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *International Conference on Learning Representations*, 2019. (Cited on pages 143, 148, and 166.)

[44] 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. (Cited on pages 13, 55, 60, and 63.)

[45] Alberto Carrassi, Marc Bocquet, Laurent Bertino, and Geir Evensen. Data assimilation in the geosciences: An overview of methods, issues, and perspectives. *Wiley Interdisciplinary Reviews: Climate Change*, 9(5):e535, 2018. (Cited on page 7.)

[46] Alberto Carrassi, Marc Bocquet, Laurent Bertino, and Geir Evensen. Data assimilation in the geosciences: An overview of methods, issues, and perspectives. *Wiley Interdisciplinary Reviews: Climate Change*, 9(5):e535, 2018. (Cited on page 26.)

[47] Alberto Carrassi, Marc Bocquet, Laurent Bertino, and Geir Evensen. Data assimilation in the geosciences: An overview of methods, issues, and perspectives. *Wiley Interdisciplinary Reviews: Climate Change*, 9(5):e535, 2018. (Cited on page 27.)

[48] Bo Chang et al. Reversible architectures for arbitrarily deep residual neural networks. In *AAAI Conference on Artificial Intelligence*, 2018. (Cited on pages 106 and 107.)

[49] Lin Chen and Sheng Xu. Deep neural tangent kernel and Laplace kernel have the same RKHS. In *International Conference on Learning Representations*, 2021. (Cited on pages 145 and 186.)

[50] R.T.Q Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, 2018. (Cited on pages 9, 14, 20, 29, 55, 92, 107, and 125.)

[51] Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David K. Duvenaud. Neural ordinary differential equations. In *Advances in neural information processing systems (NeurIPS)*, pages 6571–6583, 2018. (Cited on pages 60, 68, 69, 75, and 210.)

[52] Wen-Hua Chen. Disturbance observer based control for nonlinear systems. *IEEE/ASME transactions on mechatronics*, 9(4):706–710, 2004. (Cited on page 60.)

[53] Zhengdao Chen, Jianyu Zhang, Martin Arjovsky, and Léon Bottou. Symplectic recurrent neural networks. *International Conference on Learning Representations (ICLR)*, 2020. (Cited on page 60.)

[54] Xiuyuan Cheng and Yao Xie. Neural tangent kernel maximum mean discrepancy. *arXiv preprint arXiv:2106.03227*, 2021. (Cited on page 166.)

[55] Lénaïc Chizat and Francis Bach. On the global convergence of gradient descent for over-parameterized models using optimal transport. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31, pages 3036–3046. Curran Associates, Inc., 2018. (Cited on page 86.)

[56] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, 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)*, pages 1724–1734, Doha, Qatar, oct 2014. Association for Computational Linguistics. (Cited on pages 92 and 98.)

[57] Edward Choi, Mohammad Taha Bahadori, Jimeng Sun, Joshua Kulas, Andy Schuetz, and Walter Stewart. RETAIN: An interpretable predictive model for healthcare using reverse time attention mechanism. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 3504–3512, 2016. (Cited on page 59.)

[58] Yunjey Choi, Min-Je Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. 2018. (Cited on page 106.)

[59] Casey Chu, Kentaro Minami, and Kenji Fukumizu. The equivalence between Stein variational gradient descent and black-box variational inference. *arXiv preprint arXiv:2004.01822*, 2020. (Cited on page 145.)

[60] Yu-An Chung, Wei-Hung Weng, Schrasing Tong, and James Glass. Unsupervised cross-modal alignment of speech and text embedding spaces. 2018. (Cited on page 106.)

[61] Robert M. Corless, Hui Ding, Nicholas J. Higham, and David J. Jeffrey. The solution of $S\exp(S) = A$ is not always the Lambert W function of A. In *Proceedings of the 2007 International Symposium on Symbolic and Algebraic Computation*, ISSAC '07, pages 116–121, New York, NY, USA, 2007. Association for Computing Machinery. (Cited on page 171.)

[62] Robert M. Corless, Gaston H. Gonnet, David E. G. Hare, David J. Jeffrey, and Donald E. Knuth. On the Lambert W function. *Advances in Computational Mathematics*, 5(1):329–359, dec 1996. (Cited on page 170.)

[63] Philippe Courtier, J-N 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. (Cited on page 60.)

[64] Nicolas Courty, Rémi Flamary, Devis Tuia, and Alain Rakotomamonjy. Optimal transport for domain adaptation. *CoRR*, abs/1507.00504, 2015. (Cited on page 108.)

[65] Miles Cranmer, Sam Greydanus, Stephan Hoyer, Peter Battaglia, David Spergel, and Shirley Ho. Lagrangian neural networks. *ICLR 2020 Deep Differential Equations Workshop*, 2020. (Cited on pages 63 and 72.)

[66] James P. Crutchfield and Bruce S. Mcnamara. Equations of motion from a data series. *Complex Systems*, page 452, 1987. (Cited on pages 26 and 27.)

[67] Wojciech Marian Czarnecki, Grzegorz Świrszcz, Max Jaderberg, Simon Osindero, Oriol Vinyals, and Koray Kavukcuoglu. Understanding synthetic gradients and decoupled neural interfaces. In *ICML*, 2017. (Cited on page 136.)

[68] Bharath Bhushan Damodaran, Benjamin Kellenberger, Rémi Flamary, Devis Tuia, and Nicolas Courty. Deepjdot: Deep joint distribution optimal transport for unsupervised domain adaptation. 2018. (Cited on page 108.)

[69] Emmanuel de Bézenac, Arthur Pajot, and Patrick Gallinari. Deep learning for physical processes: Incorporating prior scientific knowledge. In *ICLR*, 2018. (Cited on pages 27, 35, 36, and 55.)

[70] Emmanuel de Bézenac, Arthur Pajot, and Patrick Gallinari. Deep learning for physical processes: Incorporating prior scientific knowledge. *International Conference on Learning Representations (ICLR)*, 2018. (Cited on page 60.)

[71] Emmanuel de Bézennac, Ibrahim Ayed, and Patrick Gallinari. Optimal unsupervised domain translation. *arXiv*, 2019. (Cited on page 107.)

[72] Giacomo De Palma, Bobak Kiani, and Seth Lloyd. Random deep neural networks are biased towards simple functions. In *Advances in Neural Information Processing Systems*, 2019. (Cited on page 106.)

[73] Emily Denton and Rob Fergus. Stochastic video generation with a learned prior. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1174–1183, Stockholmsmässan, Stockholm, Sweden, jul 2018. PMLR. (Cited on page 6.)

[74] Jérémie Donà, Jean-Yves Franceschi, Sylvain Lamprier, and Patrick Gallinari. Pde-driven spatiotemporal disentanglement. *International Conference on Learning Representations (ICLR)*, 2020. (Cited on page 59.)

[75] 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. (Cited on page 68.)

[76] J.R. Dormand and P.J. Prince. A family of embedded runge-kutta formulae. *Journal of Computational and Applied Mathematics*, 6(1):19 – 26, 1980. (Cited on page 91.)

[77] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox. FlowNet: Learning optical flow with convolutional networks. pages 2758–2766. IEEE, 2015. (Cited on page 35.)

[78] Andrew Duncan, Nikolas Nüsken, and Lukasz Szpruch. On the geometry of Stein variational gradient descent. *arXiv preprint arXiv:1912.00894*, 2019. (Cited on page 163.)

[79] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard S. Zemel. Fairness through awareness. 2012. (Cited on page 128.)

[80] Weinan E. A proposal on machine learning via dynamical systems. *Communications in Mathematics and Statistics*, 5:1–11, 02 2017. (Cited on page 8.)

[81] Ronan Fablet, Said Ouala, and Cédric Herzet. Bilinear residual neural network for the identification and forecasting of dynamical systems. *CoRR*, abs/1712.07003, 2017. (Cited on page 27.)

[82] Zhou Fan and Zhichao Wang. Spectra of the conjugate kernel and neural tangent kernel for linear-width neural networks. 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 7710–7721. Curran Associates, Inc., 2020. (Cited on pages 145 and 186.)

[83] Bálint Farkas and Sven-Ake Wegner. Variations on Barbălat's lemma. *The American Mathematical Monthly*, 123(8):825–830, 2016. (Cited on page 169.)

[84] Rafael Felix, Vijay B. G. Kumar, Ian Reid, and Gustavo Carneiro. Multi-modal cycle-consistent generalized zero-shot learning. 2018. (Cited on page 106.)

[85] John Ferguson. Geological applications of differential equations. In Springer, editor, *Mathematics in Geology*, chapter 8, pages 216–237. 1988. (Cited on page 25.)

[86] Jean Feydy, Thibault Séjourné, François-Xavier Vialard, Shun-ichi Amari, Alain Trouve, and Gabriel Peyré. Interpolating between optimal transport and MMD using Sinkhorn divergences. In Kamalika Chaudhuri and Masashi Sugiyama, editors, *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pages 2681–2690. PMLR, apr 2019. (Cited on pages 173 and 177.)

[87] Richard P. Feynman. The principle of least action in quantum mechanics. In *Feynman's Thesis - A New Approach to Quantum Theory*. World Scientific Publishing, 2005. (Cited on page 110.)

[88] A. Figalli. *The Monge-Ampère Equation and Its Applications*. Zurich lectures in advanced mathematics. European Mathematical Society, 2017. (Cited on page 112.)

[89] Alessio Figalli. *Optimal transportation and action-minimizing measures*. Tesi 8. Ed. della normale, Pisa, 2008. (Cited on pages 139 and 141.)

[90] Chris Finlay, Jorn-Henrik Jacobsen, Levon Nurbekyan, and Adam M Oberman. How to train your neural ode. In *ICML*, 2020. (Cited on page 20.)

[91] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *CoRR*, abs/1703.03400, 2017. (Cited on pages 77 and 92.)

[92] James Fletcher and Warren Moors. Chebyshev sets. *Journal of the Australian Mathematical Society*, 98:161–231, 04 2014. (Cited on page 63.)

[93] C. Foias, O. Manley, R. Rosa, and R. Temam. *Navier-Stokes Equations and Turbulence*. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 2001. (Cited on page 33.)

[94] Jean-Yves Franceschi, Edouard Delasalles, Mickaël Chen, Sylvain Lamprier, and Patrick Gallinari. Stochastic latent residual video prediction. *arXiv preprint arXiv:2002.09219*, 2020. (Cited on page 6.)

[95] Stefania Fresca, Andrea Manzoni, Luca Dedè, and Alfio Quarteroni. *Deep learning-based reduced order models in cardiac electrophysiology*, volume 15. 2020. (Cited on page 55.)

[96] Tomer Galanti, Lior Wolf, and Sagie Benaim. The role of minimal complexity functions in unsupervised learning of semantic mappings. 2018. (Cited on pages 106, 107, and 114.)

[97] V. Garcia-Morales, J. Pellicer, and J. Manzanares. Thermodynamics based on the principle of least abbreviated action. *Annals of Physics*, 2008. (Cited on page 110.)

[98] Mario Geiger, Stefano Spigler, Arthur Jacot, and Matthieu Wyart. Disentangling feature and lazy training in deep neural networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2020(11), nov 2020. (Cited on page 148.)

[99] P. Gentine, M. Pritchard, S. Rasp, G. Reinaudi, and G. Yacalis. Could machine learning break the convection parameterization deadlock? *Geophysical Research Letters*, 45(11):5742–5751, 2018. (Cited on page 60.)

[100] Rui Gong, Wen Li, Yuhua Chen, and Luc Van Gool. DLOW: domain flow for adaptation and generalization. (Cited on page 107.)

[101] Ian Goodfellow. NIPS 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016. (Cited on page 147.)

[102] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Zoubin Ghahramani, Max Welling, Corinna Cortes, Neil D. Lawrence, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27, pages 2672–2680. Curran Associates, Inc., 2014. (Cited on pages 143, 144, and 146.)

[103] C. G. Gray. Principle of least action. *Scholarpedia*, 2009. (Cited on page 110.)

[104] Arthur Gretton, Karsten M. Borgwardt, Malte Rasch, Bernhard Schölkopf, and Alex Smola. A kernel method for the two-sample-problem. In Bernhard Schölkopf, John C. Platt, and Thomas Hoffman, editors, *Advances in Neural Information Processing Systems*, volume 19, pages 513–520. MIT Press, 2007. (Cited on page 166.)

[105] Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13(25):723–773, 2012. (Cited on page 165.)

[106] Samuel Greydanus, Misko Dzamba, and Jason Yosinski. Hamiltonian neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 15353–15363, 2019. (Cited on pages 60, 63, 68, 69, 72, and 75.)

[107] Max D. Gunzburger. *Perspectives in Flow Control and Optimization*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2002. (Cited on page 19.)

[108] Eldad Haber, Keegan Lensink, Eran Treister, and Lars Ruthotto. IMEXnet a forward stable deep neural network. In *36th International Conference on Machine Learning*, 2019. (Cited on page 107.)

[109] Charles R. Harris, K. Jarrod Millman, St'efan J. 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 Fern'andez del R'ıo, Mark Wiebe, Pearu Peterson, Pierre G'erard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, sep 2020. (Cited on page 91.)

[110] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer-Verlag New York, 2001. (Cited on page 106.)

[111] Michael Hauser. On residual networks learning a perturbation from identity. *arXiv*, 2019. (Cited on page 107.)

[112] David Haussler. Decision theoretic generalizations of the pac model for neural net and other learning applications. *Information and Computation*, 100(1):78 – 150, 1992. (Cited on page 87.)

[113] K. He, X. Zhan, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *ECCV*. Springer, Cham, 2016. (Cited on pages 106, 107, and 132.)

[114] K. He, X. Zhan, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *ECCV*, 2016. (Cited on page 138.)

[115] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778, 2016. (Cited on pages 35 and 46.)

[116] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. (Cited on pages 106, 107, and 112.)

[117] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, jun 2016. (Cited on page 148.)

[118] Nicholas J. Higham. *Functions of matrices: theory and computation*. Society for Industrial and Applied Mathematics, 2008. (Cited on page 164.)

[119] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989. (Cited on page 186.)

[120] Jiri Hron, Yasaman Bahri, Jascha Sohl-Dickstein, and Roman Novak. Infinite attention: NNGP and NTK for deep attention networks. In Hal Daumé, III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 4376–4386. PMLR, jul 2020. (Cited on page 148.)

[121] Kaixuan Huang, Yuqing Wang, Molei Tao, and Tuo Zhao. Why do deep residual networks generalize better than deep feedforward networks? — a neural tangent kernel perspective. 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 2698–2709. Curran Associates, Inc., 2020. (Cited on page 161.)

[122] Zhouyuan Huo, Bin Gu, and Heng Huang. Training neural networks using features replay. In *NeurIPS*, 2018. (Cited on page 136.)

[123] Zhouyuan Huo, Bin Gu, Qian Yang, and Heng Huang. Decoupled parallel backpropagation with convergence guarantee. In *ICML*, 2018. (Cited on page 136.)

[124] Roberto Iacono and John P. Boyd. New approximations to the principal real-valued branch of the Lambert W-function. *Advances in Computational Mathematics*, 43(6):1403–1436, 2017. (Cited on page 171.)

[125] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *arxiv*, 2016. (Cited on page 105.)

[126] Arthur Jacot, Franck Gabriel, François Ged, and Clément Hongler. Order and chaos: NTK views on DNN normalization, checkerboard and boundary artifacts. *arXiv preprint arXiv:1907.05715*, 2019. (Cited on page 145.)

[127] Arthur Jacot, Franck Gabriel, and Clement Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in Neural Information Processing Systems*, 2018. (Cited on pages 106 and 113.)

[128] Arthur Jacot, Franck Gabriel, and Clement Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In Samy Bengio, Hanna Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31, pages 8580–8589. Curran Associates, Inc., 2018. (Cited on pages 143, 145, 147, 149, 161, 166, 167, 173, 185, and 186.)

[129] M. Jaderberg, W. M. Czarnecki, S. Osindero, O. Vinyals, A. Graves, D. Silver, and K. Kavukcuoglu. Decoupled neural interfaces using synthetic gradients. In *ICML*, 2017. (Cited on pages 135 and 136.)

[130] Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 12519–12530, 2019. (Cited on page 60.)

[131] Stanislaw Jastrzebski, Devansh Arpit, Nicolas Ballas, Vikas Verma, Tong Che, and Yoshua Bengio. Residual connections encourage iterative inference. In *ICLR*, 2018. (Cited on pages 107 and 113.)

[132] Ziwei Ji, Matus Telgarsky, and Ruicheng Xian. Neural tangent kernels, transportation mappings, and universal approximation. In *International Conference on Learning Representations*, 2020. (Cited on pages 177 and 186.)

[133] Gordon G Johnson. A nonconvex set which has the unique nearest point property. *Journal of Approximation Theory*, 51(4):289 – 332, 1987. (Cited on page 63.)

[134] Ekaterina Kalinicheva, Dino Ienco, Jérémie Sublime, and Maria Trocan. Unsupervised change detection analysis in satellite image time series using deep learning combined with graph-based approaches. *IEEE J. Sel. Top. Appl. Earth Obs. Remote. Sens.*, 13:1450–1466, 2020. (Cited on page 27.)

[135] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. 1960. (Cited on page 60.)

[136] Anuj Karpatne, William Watkins, Jordan Read, and Vipin Kumar. Physics-guided Neural Networks (PGNN): An Application in Lake Temperature Modeling. In *http://arxiv.org/abs/1710.11431*, 2017. (Cited on page 27.)

[137] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4396–4405, jun 2019. (Cited on page 166.)

[138] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of StyleGAN. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8107–8116, jun 2020. (Cited on page 143.)

[139] Patrick Kidger and Terry Lyons. Universal Approximation with Deep Narrow Networks. In Jacob Abernethy and Shivani Agarwal, editors, *Proceedings of Thirty Third Conference on Learning Theory*, volume 125 of *Proceedings of Machine Learning Research*, pages 2306–2327. PMLR, 09–12 Jul 2020. (Cited on pages 86 and 106.)

[140] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *Proceedings of International Conference for Learning Representations, San Diego, 2015*. (Cited on pages 46 and 98.)

[141] 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. (Cited on page 66.)

[142] David Krueger, Ethan Caballero, Jörn-Henrik Jacobsen, Amy Zhang, Jonathan Binas, Rémi Le Priol, and Aaron C. Courville. Out-of-distribution generalization via risk extrapolation (rex). *CoRR*, abs/2003.00688, 2020. (Cited on page 77.)

[143] Karol Kurach, Mario Lucic, Xiaohua Zhai, Marcin Michalski, and Sylvain Gelly. A large-scale study on regularization and normalization in GANs. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 3581–3590. PMLR, jun 2019. (Cited on page 143.)

[144] Alex Lamb, Anirudh Goyal, Ying Zhang, Saizheng Zhang, Aaron Courville, and Yoshua Bengio. Professor Forcing: A New Algorithm for Training Recurrent Networks. *arXiv:1610.09038 [cs, stat]*, oct 2016. arXiv: 1610.09038. (Cited on page 98.)

[145] Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. Unsupervised machine translation using monolingual corpora only. 2018. (Cited on page 106.)

[146] William Large and Stephen Yeager. Diurnal to decadal global forcing for ocean and sea-ice models: The data sets and flux climatologies. 05 2004. (Cited on page 60.)

[147] Vincent Le Guen and Nicolas Thome. Disentangling physical dynamics from unknown factors for unsupervised video prediction. In *Computer Vision and Pattern Recognition (CVPR)*. 2020. (Cited on pages 61 and 71.)

[148] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, nov 1998. (Cited on pages 173 and 179.)

[149] Jaehoon Lee, Yasaman Bahri, Roman Novak, Samuel S. Schoenholz, Jeffrey Pennington, and Jascha Sohl-Dickstein. Deep neural networks as Gaussian processes. In *International Conference on Learning Representations*, 2018. (Cited on pages 155 and 160.)

[150] Jaehoon Lee, Samuel S. Schoenholz, Jeffrey Pennington, Ben Adlam, Lechao Xiao, Roman Novak, and Jascha Sohl-Dickstein. Finite versus infinite neural networks: an empirical study. In Hanna Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché Buc, Emily Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 15156–15172. Curran Associates, Inc., 2020. (Cited on page 145.)

[151] Jaehoon Lee, Lechao Xiao, Samuel S. Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. In Hanna Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché Buc, Emily Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32, pages 8572–8583. Curran Associates, Inc., 2019. (Cited on page 145.)

[152] Seungjun Lee, Haesang Yang, and Woojae Seong. Identifying physical law of hamiltonian systems via meta-learning. *CoRR*, abs/2102.11544, 2021. (Cited on page 77.)

[153] Roy B. Leipnik and Charles E. M. Pearce. The multivariate Faà di Bruno formula and multivariate Taylor expansions with explicit integral remainder term. *The ANZIAM Journal*, 48(3):327–341, 2007. (Cited on page 157.)

[154] Moshe Leshno, Vladimir Ya. Lin, Allan Pinkus, and Shimon Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Networks*, 6(6):861–867, 1993. (Cited on page 186.)

[155] Chun-Liang Li, Wei-Cheng Chang, Yu Cheng, Yiming Yang, and Barnabás Póczos. MMD GAN: Towards deeper understanding of moment matching network. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30, pages 2200–2210. Curran Associates, Inc., 2017. (Cited on page 166.)

[156] Qianxiao Li, Long Chen, Cheng Tai, and Weinan E. Maximum principle based algorithms for deep learning. *Journal of Machine Learning Research*, 2018. (Cited on page 112.)

[157] Shihua Li, Jun Yang, Wen-Hua Chen, and Xisong Chen. *Disturbance observer-based control: methods and applications*. CRC press, 2014. (Cited on page 60.)

[158] Zongyi Li, Nikola Borislavov Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*, 2021. (Cited on pages 13, 86, 90, 91, and 98.)

[159] Jae Hyun Lim and Jong Chul Ye. Geometric GAN. *arXiv preprint arXiv:1705.02894*, 2017. (Cited on page 146.)

[160] Etai Littwin, Tomer Galanti, Lior Wolf, and Greg Yang. On infinite-width hypernetworks. 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 13226–13237. Curran Associates, Inc., 2020. (Cited on page 145.)

[161] Etai Littwin, Ben Myara, Sima Sabah, Joshua Susskind, Shuangfei Zhai, and Oren Golan. Collegial ensembles. 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 18738–18748. Curran Associates, Inc., 2020. (Cited on page 145.)

[162] Chaoyue Liu, Libin Zhu, and Misha Belkin. On the linearity of large non-linear models: when and why the tangent kernel is constant. 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 15954–15964. Curran Associates, Inc., 2020. (Cited on pages 145 and 147.)

[163] Ming-Yu Liu, Xun Huang, Jiahui Yu, Ting-Chun Wang, and Arun Mallya. Generative adversarial networks for image and video synthesis: Algorithms and applications. *Proceedings of the IEEE*, 109(5):839–862, 2021. (Cited on page 143.)

[164] Qiang Liu and Dilin Wang. Stein variational gradient descent: A general purpose Bayesian inference algorithm. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29, pages 2378–2386. Curran Associates, Inc., 2016. (Cited on pages 163 and 177.)

[165] Shuang Liu, Olivier Bousquet, and Kamalika Chaudhuri. Approximation and convergence properties of generative adversarial learning. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30, pages 5551–5559. Curran Associates, Inc., 2017. (Cited on page 145.)

[166] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *IEEE International Conference on Computer Vision (ICCV)*, pages 3730–3738, dec 2015. (Cited on pages 173 and 179.)

[167] Yun Long, Xueyuan She, and Saibal Mukhopadhyay. Hybridnet: integrating model-based and data-driven learning to predict evolution of dynamical systems. *Conference on Robot Learning (CoRL)*, 2018. (Cited on page 60.)

[168] Zichao Long, Yiping Lu, Xianzhong Ma, and Bin Dong. PDE-Net: Learning PDEs from Data. In *ICML*, pages 3214–3222, 2018. (Cited on page 13.)

[169] Zichao Long, Yiping Lu, Xianzhong Ma, and Bin Dong. PDE-Net: Learning PDEs from data. In *International Conference on Machine Learning (ICML)*, 2018. (Cited on pages 27, 55, and 60.)

[170] A. C. Lorenc. Analysis methods for numerical weather prediction. *Quarterly Journal of the Royal Meteorological Society*, 112(474):1177–1194, 1986. (Cited on page 27.)

[171] Alfred J. Lotka. Elements of physical biology. *Science Progress in the Twentieth Century (1919-1933)*, 21(82):341–343, 1926. (Cited on page 90.)

[172] Guansong Lu, Zhiming Zhou, Yuxuan Song, Kan Ren, and Yong Yu. Guiding the one-to-one mapping in cyclegan via optimal transport. 2018. (Cited on page 108.)

[173] Lu Lu, Pengzhan Jin, and George Em Karniadakis. Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. *CoRR*, abs/1910.03193, 2019. (Cited on page 13.)

[174] Yiping Lu, Aoxiao Zhong, Quanzheng Li, and Bin Dong. Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations. In *35th International Conference on Machine Learning*, 2018. (Cited on pages 106 and 107.)

[175] Mario Lucic, Karol Kurach, Marcin Michalski, Sylvain Gelly, and Olivier Bousquet. Are GANs created equal? a large-scale study. In Samy Bengio, Hanna Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31, pages 698–707. Curran Associates, Inc., 2018. (Cited on page 143.)

[176] X. Ma, N. Trudinger, and X. Wang. Regularity of potential functions of the optimal transportation problem. *Archive for Rational Mechanics and Analysis*, 2005. (Cited on page 112.)

[177] G. Madec. *NEMO ocean engine*. Note du Pôle de modélisation, Institut Pierre-Simon Laplace (IPSL), France, No 27, ISSN No 1288-1619, 2008. (Cited on pages 41 and 75.)

[178] Xudong Mao, Qing Li, Haoran Xie, Raymond Y. K. Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2813–2821, oct 2017. (Cited on page 146.)

[179] Enrique S. Marquez, Jonathon S. Hare, and Mahesan Niranjan. Deep cascade learning. *IEEE Transactions on Neural Networks and Learning Systems*, 2018. (Cited on page 136.)

[180] Michael Mathieu, Camille Couprie, and Yann LeCun. Deep multi-scale video prediction beyond mean square error. In *International Conference on Learning Representations*, 2016. (Cited on page 6.)

[181] Viraj Mehta, Ian Char, Willie Neiswanger, Youngseog Chung, and Jeff Schneider. Neural dynamical systems. *ICLR 2020 Deep Differential Equations Workshop*, 2020. (Cited on pages 61, 69, 71, and 73.)

[182] Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for GANs do actually converge? In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3481–3490. PMLR, jul 2018. (Cited on pages 145 and 182.)

[183] Lars Mescheder, Sebastian Nowozin, and Andreas Geiger. The numerics of GANs. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30, pages 1823–1833. Curran Associates, Inc., 2017. (Cited on page 145.)

[184] Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. Unrolled generative adversarial networks. In *International Conference on Learning Representations*, 2017. (Cited on pages 176 and 184.)

[185] Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. Meta-learning with temporal convolutions. *CoRR*, abs/1707.03141, 2017. (Cited on page 77.)

[186] Colleen C. Mitchell and David G. Schaeffer. A two-current model for the dynamics of cardiac membrane. *Bulletin of mathematical biology*, 65(5):767–793, Sep 2003. (Cited on page 25.)

[187] Colleen C. Mitchell and David G. Schaeffer. A two-current model for the dynamics of cardiac membrane. *Bulletin of mathematical biology*, 65(5):767–793, Sep 2003. (Cited on page 45.)

[188] Colleen C Mitchell and David G Schaeffer. A two-current model for the dynamics of cardiac membrane. *Bull. Math. Biol*, 65(5):767–793, 2003. (Cited on page 48.)

[189] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *CoRR*, abs/1802.05957, 2018. (Cited on page 89.)

[190] Nikita Moriakov, Jonas Adler, and Jonas Teuwen. Kernel of cyclegan as a principle homogeneous space, 2020. (Cited on pages 106 and 114.)

[191] Youssef Mroueh and Truyen Nguyen. On the convergence of gradient descent in GANs: MMD GAN as a gradient flow. In Arindam Banerjee and Kenji Fukumizu, editors, *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 1720–1728. PMLR, apr 2021. (Cited on page 166.)

[192] Youssef Mroueh, Tom Sercu, and Anant Raj. Sobolev descent. In Kamalika Chaudhuri and Masashi Sugiyama, editors, *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pages 2976–2985. PMLR, apr 2019. (Cited on pages 163 and 176.)

[193] Krikamol Muandet, Kenji Fukumizu, Bharath Sriperumbudur, and Bernhard Schölkopf. Kernel mean embedding of distributions: A review and beyond. *Foundations and Trends® in Machine Learning*, 10(1–2):1–141, 2017. (Cited on pages 166 and 176.)

[194] Alfred Müller. Integral probability metrics and their generating classes of functions. *Advances in Applied Probability*, 29(2):429–443, 1997. (Cited on page 146.)

[195] Anusha Nagabandi, Gregory Kahn, Ronald S Fearing, and Sergey Levine. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7559–7566. IEEE, 2018. (Cited on page 60.)

[196] Vaishnavh Nagarajan and J. Zico Kolter. Gradient descent GAN optimization is locally stable. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30, pages 5591–5600. Curran Associates, Inc., 2017. (Cited on page 145.)

[197] 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. (Cited on page 89.)

[198] P. Nakkiran, G. Kaplun, Y. Bansal, T. Yang, B. Barak, and I. Sutskever. Deep double descent: Where bigger models and more data hurt. In *ICLR*, 2020. (Cited on page 105.)

[199] Aurel Neic, Fernando O. 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. (Cited on page 75.)

[200] Oliver Nelles. *Nonlinear System Identification*, volume 13. 01 2001. (Cited on page 27.)

[201] Duong Nguyen, Said Ouala, Lucas Drumetz, and Ronan Fablet. EM-like Learning Chaotic Dynamics from Noisy and Partial Observations. 3 2019. (Cited on page 27.)

[202] Alexander Norcliffe, Cristian Bodnar, Ben Day, Jacob Moss, and Pietro Liò. Neural ODE processes. In *International Conference on Learning Representations*, 2021. (Cited on page 77.)

[203] R. Novak, Y. Bahri, D. A. Abolafia, J. Pennington, and J. Sohl-Dickstein. Sensitivity and generalization in neural networks: an empirical study. In *ICLR*, 2018. (Cited on page 106.)

[204] Roman Novak, Lechao Xiao, Jiri Hron, Jaehoon Lee, Alexander A. Alemi, Jascha Sohl-Dickstein, and Samuel S. Schoenholz. Neural Tangents: Fast and easy infinite neural networks in Python. In *International Conference on Learning Representations*, 2020. (Cited on pages 145, 148, 172, 179, and 184.)

[205] Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-GAN: Training generative neural samplers using variational divergence minimization. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29, pages 271–279. Curran Associates, Inc., 2016. (Cited on pages 144 and 146.)

[206] Arild Nøkland and Lars Hiller Eidnes. Training neural networks with local error signals. In *ICML*, 2019. (Cited on page 138.)

[207] Boris N. Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. *International Conference on Learning Representations (ICLR)*, 2020. (Cited on pages 59, 74, and 75.)

[208] S. Ouala, C. Herzet, and R. Fablet. Sea surface temperature prediction and reconstruction using patch-level neural network representations. In *IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium*, pages 5628–5631, 2018. (Cited on page 27.)

[209] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017. (Cited on page 35.)

[210] John E. Pearson. Complex patterns in a simple system. *Science*, 261(5118):189–192, 1993. (Cited on page 91.)

[211] 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. (Cited on page 60.)

[212] G. Peyre and M. Cuturi. *Computational Optimal Transport*. Now Publishers, 2019. (Cited on page 110.)

[213] Dimitris C Psichogios and Lyle H Ungar. A hybrid neural network-first principles approach to process modeling. *AIChE Journal*, 38(10):1499–1511, 1992. (Cited on page 60.)

[214] Myeongjang Pyeon, Jihwan Moon, Taeyoung Hahn, and Gunhee Kim. Sedona: Search for decoupled neural networks toward greedy block-wise learning. In *ICLR*, 2021. (Cited on pages 136 and 138.)

[215] Evan Racah, Christopher Beckham, Tegan Maharaj, Samira Ebrahimi Kahou, Mr. Prabhat, and Christopher Pal. Extremeweather: A large-scale climate dataset for semi-supervised detection, localization, and understanding of extreme weather events. In *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, pages 3405–3416, dec 2017. (Cited on page 27.)

[216] Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. In *36th International Conference on Machine Learning*, 2019. (Cited on page 106.)

[217] M. Raissi, P. Perdikaris, and G. 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. (Cited on pages 29 and 55.)

[218] Maziar Raissi. Deep hidden physics models: Deep learning of nonlinear partial differential equations. *Journal of Machine Learning Research*, 19, 2018. (Cited on page 27.)

[219] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Machine learning of linear differential equations using gaussian processes. *Journal of Computational Physics*, 348:683 – 693, 2017. (Cited on page 27.)

[220] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 473:686–707, 2019. (Cited on pages 60 and 63.)

[221] Prajit Ramachandran, Barret Zoph, and Quoc V. Le. Searching for activation functions. *CoRR*, abs/1710.05941, 2017. (Cited on page 98.)

[222] Saikiran Rapaka, Tommaso Mansi, Bogdan Georgescu, Mihaela Pop, G. Wright, Ali Kamen, and Dorin Comaniciu. LBM-EP: Lattice-Boltzmann method for fast cardiac electrophysiology simulation from 3d images. *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2012*, pages 33–40, 2012. (Cited on page 45.)

[223] Saikiran Rapaka, Tommaso Mansi, Bogdan Georgescu, Mihaela Pop, Graham A Wright, Ali Kamen, and Dorin Comaniciu. LBM-EP: Lattice-boltzmann method for fast cardiac electrophysiology simulation from 3d images. In *Int. Conf. MICCAI*, pages 33–40. Springer, 2012. (Cited on page 48.)

[224] 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*, 566:195–204, 2019. (Cited on pages 55 and 61.)

[225] Jatin Relan, Phani Chinchapatnam, Maxime Sermesant, Kawal Rhode, Matt Ginks, Hervé Delingette, C. Aldo Rinaldi, Reza Razavi, and Nicholas Ayache. Coupled personalization of cardiac electrophysiology models for prediction of ischaemic ventricular tachycardia. *Interface Focus*, 1(3):396–407, 2011. (Cited on page 45.)

[226] R Rico-Martinez, JS Anderson, and IG 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. (Cited on page 60.)

[227] James C. Robinson. *Dimensions, Embeddings, and Attractors*. Cambridge Tracts in Mathematics. Cambridge University Press, 2010. (Cited on page 28.)

[228] 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. (Cited on page 59.)

[229] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015. (Cited on pages 35, 42, and 46.)

[230] Samuel H. Rudy, Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Data-driven discovery of partial differential equations. *Science Advances*, 3(4):e1602614, apr 2017. (Cited on pages 13 and 27.)

[231] Lars Ruthotto and Eldad Haber. Deep Neural Networks motivated by Partial Differential Equations. 2018. (Cited on page 29.)

[232] Lars Ruthotto and Eldad Haber. Deep neural networks motivated by partial differential equations. *Journal of Mathematical Imaging and Vision*, 2020. (Cited on page 107.)

[233] Priyabrata Saha, Saurabh Dash, and Saibal Mukhopadhyay. PhICNet: Physics-incorporated convolutional recurrent neural networks for modeling dynamical systems. *arXiv preprint arXiv:2004.06243*, 2020. (Cited on page 60.)

[234] M. Sandler, J. Baccash, A. Zhmoginov, and A. Howard. Non-discriminative data or weak model? on the relative importance of data and model resolution. In *International Conference on Computer Vision Workshop (ICCVW)*, 2019. (Cited on page 113.)

[235] Filippo Santambrogio. *Optimal transport for Applied Mathematicians: Calculus of Variations, PDEs and Modeling*. 2015. (Cited on pages 107, 109, 110, and 116.)

[236] Andrew M. Saxe, James L. Mcclelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural network. In *ICLR*, 2014. (Cited on pages 132 and 138.)

[237] Michael Scheuerer. *A Comparison of Models and Methods for Spatial Interpolation in Statistics and Numerical Analysis*. PhD thesis, Georg-August-Universität Göttingen, oct 2009. (Cited on page 156.)

[238] Christian Seis. A quantitative theory for the continuity equation. 2017. (Cited on page 109.)

[239] Sungyong Seo, Chuizheng Meng, and Yan Liu. Physics-aware difference graph networks for sparsely-observed dynamics. *International Conference on Learning Representations (ICLR)*, 2020. (Cited on page 60.)

[240] Shai Shalev-Shwartz and Shai Ben-David. *Covering Numbers*, page 337–340. Cambridge University Press, 2014. (Cited on page 79.)

[241] 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 *Advances in Neural Information Processing Systems 28*, pages 802–810, 2015. (Cited on page 27.)

[242] 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 *Advances in neural information processing systems (NeurIPS)*, pages 802–810, 2015. (Cited on page 59.)

[243] Justin Sirignano and Konstantinos Spiliopoulos. DGM: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics*, 375(Dms 1550918):1339–1364, 2018. (Cited on page 29.)

[244] Justin Sirignano and Konstantinos Spiliopoulos. Dgm: A deep learning algorithm for solving partial differential equations. *Journal of computational physics*, 375:1339–1364, 2018. (Cited on pages 60 and 69.)

[245] Ziv Sirkes and Eli Tziperman. Finite difference of adjoint or adjoint of finite difference? *Monthly weather review*, 125(12):3373–3378, 1997. (Cited on page 19.)

[246] Jascha Sohl-Dickstein, Roman Novak, Samuel S. Schoenholz, and Jaehoon Lee. On the infinite width limit of neural networks with a standard parameterization. *arXiv preprint arXiv:2001.07301*, 2020. (Cited on page 145.)

[247] Sho Sonoda and Noboru Murata. Transport analysis of infinitely deep neural network. *Journal of Machine Learning Research*, 2019. (Cited on page 107.)

[248] 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. Learning for Visual Semantic Understanding in Big Data ESANN 2014 Industrial Data Processing and Analysis. (Cited on pages 77, 92, and 98.)

[249] Bharath K. Sriperumbudur, Kenji Fukumizu, and Gert R. G. Lanckriet. Universality, characteristic kernels and RKHS embedding of measures. *Journal of Machine Learning Research*, 12(70):2389–2410, 2011. (Cited on page 186.)

[250] Bharath K. Sriperumbudur, Arthur Gretton, Kenji Fukumizu, Bernhard Schölkopf, and Gert R. G. Lanckriet. Hilbert space embeddings and metrics on probability measures. *Journal of Machine Learning Research*, 11(50):1517–1561, 2010. (Cited on page 148.)

[251] Akash Srivastava, Lazar Valkov, Chris Russell, Michael U. Gutmann, and Charles Sutton. VEEGAN: Reducing mode collapse in GANs using implicit variational learning. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30, pages 3310–3320. Curran Associates, Inc., 2017. (Cited on page 176.)

[252] Ingo Steinwart. On the influence of the kernel on the consistency of support vector machines. *Journal of Machine Learning Research*, 2:67–93, nov 2001. (Cited on page 186.)

[253] Ruoyu Sun, Tiantian Fang, and Alexander Schwing. Towards a better global loss landscape of GANs. 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 10186–10198. Curran Associates, Inc., 2020. (Cited on page 144.)

[254] Casper Kaae Sønderby, Jose Caballero, Lucas Theis, Wenzhe Shi, and Ferenc Huszár. Amortised MAP inference for image super-resolution. In *International Conference on Learning Representations*, 2017. (Cited on pages 166 and 185.)

[255] F. Takens. Detecting strange attractors in fluid turbulence. *Symposium on dynamical systems and turbulence*, pages 366–381, 1981. (Cited on page 28.)

[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. (Cited on page 145.)

[257] Yin Tang, Qi Teng, Lei Zhang, Fuhong Min, and Jun He. Layer-wise training convolutional neural networks with smaller filters for human activity recognition using wearable sensors. *IEEE Sensors Journal*, 2021. (Cited on page 135.)

[258] Damien Teney, Ehsan Abbasnejad, and Anton van den Hengel. Unshuffling Data for Improved Generalization. 2020. (Cited on page 77.)

[259] Qi Teng, Kun Wang, Lei Zhang, and Jun He. The layer-wise training convolutional neural networks using local loss for sensor-based human activity recognition. *IEEE Sensors Journal*, 2020. (Cited on page 135.)

[260] Michael L Thompson and Mark A Kramer. Modeling chemical processes using prior knowledge and neural networks. *AIChE Journal*, 40(8):1328–1340, 1994. (Cited on page 60.)

[261] Peter Toth, Danilo Jimenez Rezende, Andrew Jaegle, Sébastien Racanière, Aleksandar Botev, and Irina Higgins. Hamiltonian generative networks. *International Conference on Learning Representations (ICLR)*, 2020. (Cited on pages 68 and 69.)

[262] 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. (Cited on page 59.)

[263] Soumya Tripathy, Juho Kannala, and Esa Rahtu. Learning image-to-image translation using paired and unpaired training samples. 2018. (Cited on page 107.)

[264] Benjamin Ummenhofer, Lukas Prantl, Nils Thuerey, and Vladlen Koltun. Lagrangian fluid simulation with continuous convolutions. *International Conference on Learning Representations (ICLR)*, 2020. (Cited on page 60.)

[265] Thomas Vandal, Evan Kodra, Sangram Ganguly, Andrew Michaelis, Ramakrishna Nemani, and Auroop R Ganguly. Generating high resolution climate change projections through single image super-resolution: An abridged version. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 5389–5393. International Joint Conferences on Artificial Intelligence Organization, 7 2018. (Cited on page 27.)

[266] Cédric Villani. *Optimal Transport: Old and New*. Springer-Verlag, 2008. (Cited on page 109.)

[267] Cédric Villani. *The Wasserstein distances*, pages 93–111. Grundlehren der mathematischen Wissenschaften. Springer Berlin Heidelberg, Berlin - Heidelberg, Germany, 2009. (Cited on page 166.)

[268] Henning Voss, J. Timmer, and Juergen Kurths. Nonlinear dynamical system identification from uncertain and indirect measurements. *International Journal of Bifurcation and Chaos*, 14, 01 2004. (Cited on page 29.)

[269] 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. (Cited on pages 61, 69, 71, and 73.)

[270] Yulin Wang, Zanlin Ni, Shiji Song, and Gao Huang Le Yang. Revisiting locally supervised learning: an alternative to end-to-end training. In *ICLR*, 2021. (Cited on page 136.)

[271] 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, 2018. (Cited on pages 35 and 36.)

[272] 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 *International Conference on Machine Learning (ICML)*, 2018. (Cited on pages 59, 69, and 98.)

[273] 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 (NeurIPS)*, pages 879–888. 2017. (Cited on page 92.)

[274] Zhengwei Wang, Qi She, and Tomás E. Ward. Generative adversarial networks in computer vision: A survey and taxonomy. *ACM Computing Surveys*, 54(2), apr 2021. (Cited on page 143.)

[275] E Weinan. A proposal on machine learning via dynamical systems. *Communications in Mathematics and Statistics*, 2017. (Cited on pages 106 and 137.)

[276] Jared D. Willard, Xiaowei Jia, Shaoming Xu, Michael Steinbach, and Vipin Kumar. Integrating physics-based modeling with machine learning: A survey. volume 1, pages 1–34, 2020. (Cited on pages 27 and 55.)

[277] Saining Xie et al. Aggregated residual transformations for deep neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. (Cited on pages 112, 113, 132, and 134.)

[278] O. Bokanowski Y. Achdou and T. Lelievre. Partial differential equations in finance. 2007. (Cited on page 25.)

[279] Hanshu Yan, Jiawei Du, Vincent Tan, and Jiashi Feng. On robustness of neural ordinary differential equations. In *ICLR*, 2020. (Cited on pages 20 and 107.)

[280] Chao Yang, Taehwan Kim, Ruizhe Wang, Hao Peng, and C.-C. Jay Kuo. ESTHER: extremely simple image translation through self-regularization. 2018. (Cited on pages 106 and 114.)

[281] Greg Yang. Tensor programs II: Neural tangent kernel for any architecture. *arXiv preprint arXiv:2006.14548*, 2020. (Cited on pages 145 and 148.)

[282] Greg Yang and Edward J. Hu. Tensor programs iv: Feature learning in infinite-width neural networks. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 11727–11737. PMLR, jul 2021. (Cited on pages 145 and 148.)

[283] Greg Yang and Hadi Salman. A fine-grained spectral perspective on neural networks. *arXiv preprint arXiv:1907.10599*, 2019. (Cited on pages 145, 182, and 185.)

[284] Cagatay Yildiz, Markus Heinonen, and Harri Lahdesmaki. ODE2VAE: Deep generative second order odes with bayesian neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 13412–13421, 2019. (Cited on page 77.)

[285] Yuichi Yoshida and Takeru Miyato. Spectral norm regularization for improving the generalizability of deep learning. *arXiv*, 2017. (Cited on page 107.)

[286] Yuan Yuan, Siyuan Liu, Jiawei Zhang, Yongbing Zhang, Chao Dong, and Liang Lin. Unsupervised image super-resolution using cycle-in-cycle generative adversarial networks. 2018. (Cited on page 106.)

[287] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *BMVC*, 2016. (Cited on pages 112 and 113.)

[288] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning requires rethinking generalization. In *ICLR*, 2017. (Cited on pages 105 and 113.)

[289] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. 2017. (Cited on page 114.)

[290] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaolei Huang, Xiaogang Wang, and Dimitris N. Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. 2016. (Cited on page 106.)

[291] Jingfeng Zhang et al. Towards robust resnet: A small step but a giant leap. In *Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI)*, 2019. (Cited on page 107.)

[292] Yaoyu Zhang, Zhi-Qin John Xu, Tao Luo, and Zheng Ma. A type of generalization error induced by initialization in deep neural networks. In Jianfeng Lu and Rachel Ward, editors, *Proceedings of The First Mathematical and Scientific Machine Learning Conference*, volume 107 of *Proceedings of Machine Learning Research*, pages 144–164, Princeton University, Princeton, NJ, USA, jul 2020. PMLR. (Cited on pages 166, 172, 173, 174, 179, and 185.)

[293] Yu Zhang and Qiang Yang. A survey on multi-task learning. *CoRR*, abs/1707.08114, 2017. (Cited on pages 27 and 77.)

[294] Zhiming Zhou, Jiadong Liang, Yuxuan Song, Lantao Yu, Hongwei Wang, Weinan Zhang, Yong Yu, and Zhihua Zhang. Lipschitz generative adversarial nets. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 7584–7593, Long Beach, California, USA, jun 2019. PMLR. (Cited on pages 144 and 145.)

[295] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *CoRR*, abs/1703.10593, 2017. (Cited on pages 9, 106, 113, and 114.)

List of Figures

3 Improving Generalization and Robustness for Neural Differential Models of Dynamical Systems