Machine Learning (2022) 111:2349–2380
https://doi.org/10.1007/s10994-022-06139-2

Modelling spatiotemporal dynamics from Earth observation data with neural differential equations

Ibrahim Ayed[1,2] · **Emmanuel de Bézenac**[2] · **Arthur Pajot**[2] · **Patrick Gallinari**[2,3]

Received: 16 June 2020 / Revised: 4 February 2022 / Accepted: 11 February 2022 /
Published online: 18 March 2022

Abstract
Forecasting complex spatiotemporal dynamics is central in Earth science for modeling a variety of phenomena ranging from atmospheric dynamics to the evolution of vegetation. Those phenomena are often observed from remote sensing measurements that only provide partial information on the underlying physical equations. In this context, we consider the problem of automatically learning the dynamics of physical spatiotemporal processes from incomplete observations. We propose a new data-driven framework where the dynamics is modeled by an unknown differential equation and where the state representation and evolution is learned only from partial observations. The dynamical model is parametrized by a deep neural network. Since the problem is underconstrained, the model may learn high quality forecasts of the observations while being physically inconsistent. We introduce two settings that help analyze and interpret the learned model states. We evaluate the proposed model on two benchmarks: (1) the incompressible Navier–Stokes equations which underlie transport phenomena in the atmosphere and in the ocean, (2) a challenging problem of sea surface temperature prediction where the underlying dynamics corresponds to a sophisticated ocean dynamics model. The proposed model is able to provide long term forecasts for these complex dynamics and large dimensional observation spaces.

Keywords Deep learning · Forecasting · Partial observations · Spatio-temporal · Dynamical systems

Editor: Thomas Corpetti.

✉ Ibrahim Ayed
ayedibrahim@gmail.com

Emmanuel de Bézenac
emmanueldebezenac@gmail.com

[1] Present Address: Theresis Lab, Thales, Paris, France

[2] Sorbonne Université, CNRS, ISIR, Paris, France

[3] Criteo AI Lab, Paris, France

1 Introduction

Machine learning (ML) has been part of geoscience for a long time. It has been applied to many sub-domains of Earth science, mostly as dedicated ML solutions developed for specific problems. The success of deep learning for large size real world applications in domains like vision or signal processing opens new perspectives for a better and broader integration of ML and Earth science and for bypassing limitations of current ML and assimilation solutions. This has been advocated in several recent prospective papers (Reichstein et al., 2019; Bergen et al., 2019; Gil et al., 2019; Huntingford et al., 2019).

Although there exists a lot of similarities between standard deep learning applicative domains and Earth science applications, the latter have specificities that make them extremely different from the classical playground of deep learning. Let us quote some of them that are particularly relevant for the present work. (1) Earth science is mostly concerned with the modeling of complex phenomena involving spatio-temporal dynamics. This shares similarities with video and motion prediction (Mathieu et al., 2016; Denton and Fergus, 2018; Franceschi et al., 2020), but the underlying phenomena are usually much more complex, involving time-evolving multidimensional structures and observations at different spatio-temporal resolutions. (2) Only raw observations are available and in most scenarios, labels are not available. (3) 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 (Carrassi et al., 2018a): for example when studying ocean's circulation, variables contained in the system's state such as surface temperature, salinity or sea surface heigth are observable via satellite imaging, while subsurface variables characterizing ocean dynamics are substantially much more difficult to observe. In this case, the state is said to be *partially observable*. (4) Extrapolation is not guaranteed since problems in geosciences are often underconstrained, possibly leading to models with high predictive performance on the training/ test sets that do not generalize to new geophysical contexts.

Besides, there are other challenges that are common to all physical modeling problems. An important one is physical consistency and interpretability: predictions achieving good performance may be inconsistent or not physically plausible and then useless for practitioners. It is then essential to constrain the statistical model to be physically consistent. This is usually performed by regularizing the loss function or by constraining the deep learning model dynamics.

In what follows, we attempt to provide answers to some of these problems. 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 Earth observation problems. We make the hypothesis that the unkown dynamics obey a set of differential equations with general form:

$$\frac{dX_t}{dt} = F(X_t) \tag{1}$$

where X is the system state, considered here as a spatio-temporal vector field. Its value at time t is denoted $X_t(x) \in \mathbb{R}^d$. When F is known, predicting and analyzing the dynamics of the system often amounts to using an adequate numerical solver. 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. A data-driven paradigm for modeling dynamical systems has then emerged (Crutchfield and Mcnamara, 1987; Alvarez et al., 2013) for

some years, where the state dynamics is automatically discovered based on the observations. 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

$$\frac{\mathrm{d}X_t}{\mathrm{d}t} = F_\theta(X_t) \tag{2}$$

fits the measured data.

As mentioned before, for most real-world applications, the X variables describing the system are not fully visible to sensors and this is the case considered here. We then suppose available sequences of partial observations $Y_1, \ldots, Y_T$ acquired on a regular spatial grid, providing incomplete information about the unknown underlying process with (full) state variables $X_1, \ldots, X_T$. We make the classical hypothesis that incomplete observation 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 for taking into account complex spatial dependencies and multiscale behavior. Our objective is then to learn parameters θ capturing the dynamics of the system's state and then perform long-term forecasts.

Our approach indeed learns the dynamics of spatio-temporal systems from raw and partial image observations without prior knowledge of the system. We start by presenting our model and analyze its properties (Sect. 3) as well as the adjoint equation used for training it (Sect. 4). We then introduce two instances of this general model (Sect. 5). In the experimental sections, we make use of the model for two problems. First for the well-known Navier–Stokes equations (Sect. 6), which underlie a large amount of physical phenomena, for example in the ocean-atmosphere exchanges. Second, on the prediction of Sea Surface Temperature (Sect. 7), for which we use data coming from a sophisticated ocean dynamics model which is improved using actual observations. Our experiments demonstrate a clear improvement over state-of-the-art deep learning baselines in terms of forecasting accuracy. We further analyze experimentaly how the learned state dynamics characterize the non-observed state variables. This is up to our knowledge the first data driven model able to forecast complex spatio-temporal dynamics characteristics of geophysics applications, in a partially observed context, at this complexity and size levels.

To summarize, our main contributions are the following:

- We propose a framework for learning spatio-temporal dynamics characteristic of geophysics transport phenomena 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 represnetaions with respect to the canonical interpretable physical states.
- We demonstrate its performances on two problems: the incompressible Navier–Stokes equations and a challenging and realistic dataset of Sea Surface Temperatures.

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

2 Related work

In the past, several works have already attempted to learn differential equations from data, such as e.g., Crutchfield and Mcnamara (1987), Alvarez et al. (2013). More recently, Rudy et al. (2017) and Zhang and Lin (2018) use sparse regression on a dictionary of differential terms to recover the underlying PDE. In Raissi et al. (2017), they propose recovering the coefficients of the differential terms by deriving a GP kernel from a linearized form of the PDE. Long et al. (2018) carefully tailor the neural network architecture, based on the discretization of the different terms of the underlying PDE. Raissi (2018) develops a NN framework for learning PDEs from data. Fablet et al. (2017) construct a bilinear network and use an architecture similar to finite difference schemes to learn fully observed dynamical systems. In those approaches, we often see that either the form of the PDE or the variable dependency are supposed to be known and that the context is the unrealistic setting where the state is fully observed.

Related to our work, there is the field of data assimilation (Lorenc, 1986; Carrassi et al., 2018b), 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 (Carrassi et al., 2018b), 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 (Bérèziat and Herlin, 2015), 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 partially observations. Nguyen et al. (2019), 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 de Bézenac et al. (2018), 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 the data, which allows the network hidden state to ressemble the true hidden state.

As mentioned in the introduction, machine learning as been part of geophysics modeling for the past decades. Most machine learning methodologies have been aplied to geophysics and remote sensing. We will focus here on recent developments in the field. The last few years have seen an exponentially increasing number of deep learning applications to geophysics through the use of Earth observation data. We then highlight a few representative applications. Kalinicheva et al. (2020) perform change detection for satellite image time series using autoencoders. One of the first papers for extreme weather event detection is considered in Racah et al. (2017). Convolutional LSTMs were introduced in Shi et al. (2015) for nowcasting. Karpatne et al. (2017) 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 is lake temperature modeling. Vandal et al. (2018) makes use of a super resolution convolutional neural network with multi-scale input channels for statistical downscaling of climate variables. de Bézenac et al. (2018) used as a baseline in this paper introduces physical knowldge under the form of an advection-diffusion equation in order to predict sea surface temperature. Ouala et al. (2018) also tackle the forecasting and assimilation of geophysical fields and consider sea surface temperature as an application.

Moreover, during the last few years, a link has been made between residual networks and dynamical systems (Weinan, 2017): a residual block $h_{t+1} = h_t + f(h_t, \theta_t)$ can be seen as the explicit Euler discretization of the following system: $\frac{dh_t}{dt} = f(h_t, \theta_t)$. Adopting this viewpoint, time t corresponds to the neural network's layer index, the initial condition $h(0)$ to the network's input, and the forward pass as the time integration $h(T) = h(0) + \int_0^T f(h(t), \theta_t)\,\mathrm{dt}$, where $h(T)$ corresponds to its output. Chen et al. (2018) propose computing this intractable integral using an ordinary differential equation (ODE) solver. During training, in order to compute the derivative with respect to the neural network parameters, the corresponding adjoint state equation is solved backward in time. Note that in our work, instead of considering the evolution of the inner dynamics of the neural throughout its layers, we consider the dynamics of the studied process itself, in the context of partially observed states.

3 Learning the dynamics of partially observable systems

Let us first formulate the task of learning partially observed dynamics as an optimization problem and then introduce a training algorithm.

3.1 Partially observable systems: hypothesis

We assume that only partial measurements of the system's state are available.

Our hypothesis are the following:

- We have a dataset $\{Y^{(i)}\}_i = 1 \dots N$ 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 vector field X satisfying equation (1) such that $\mathcal{H}(X) = Y$.

$\mathcal{H}$ represents the loss of information between the state X describing the system and observations Y. Note that in our experiments $\mathcal{H}$ will be a projection operator, i.e. Y_t is a subset of the variables in X_t. A first question is whether it is always possible to reconstruct the state X from the observations, for any function $\mathcal{H}$. This is not the case in general: if $\mathcal{H}$ has a constant value for all inputs for example. However, the Takens theorem, see Takens (1981) for the original statement and Robinson (2010) for a more recent version, states that, for a dense set of observation functions $\mathcal{H}$, there exists an integer K such that $Y_{t-K+1:t}$ can be transoformed into X_t.[1] 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. This implies that there could exist many state

[1] 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.

representations which induce the same observations Y.[2] Our experiments are performed on simulated data, providing access to all the variables of the system. We will denote by **canonical state**, the true state of the physical model, which is not available for training in our context but known from the simulations. Having access to this ground truth 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 evaluation purpose and is not feasible in real situations where we have no access to state variables.

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 the following:

$$\begin{aligned} \underset{\theta}{\text{minimize}} \quad & \mathbb{E}_{Y\in\text{Dataset}}[\mathcal{J}(Y,\mathcal{H}(X))] \\ \text{subject to} \quad & \frac{\mathrm{d}X_t}{\mathrm{d}t} = F_\theta(X_t) \\ & X_0 = g_\theta(Y_{-k+1:0}), \end{aligned} \tag{3}$$

where we take

$$\mathcal{J}(Y,\widetilde{Y}) = \int_0^T \|Y_t - \widetilde{Y}_t\|_{L^2}^2, \tag{4}$$

taking the L^2 norm 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 $\{(Y^{(i)}_{-k+1},\dots,Y^{(i)}_0,\dots,Y^{(i)}_T)\}_i$, 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 for predicting an initial state X_0 from past observations $Y_{-k+1:0}$.

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 (Voss et al., 2004). Closer to us, neural differential equation solvers, e.g., Sirignano and Spiliopoulos (2018), Raissi et al. (2019) or Chen et al. (2018), all assume *having access to the full states* and not only to incomplete observations as we do here. Sirignano and Spiliopoulos (2018), Raissi et al. (2019) furthermore assume that the form of the differential equation is known. Solving problem (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 (Ruthotto and Haber, 2018) unsupervisedly. The time evolution is 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] We give a more rigorous statement regarding the more particular situation of interest in this work in Sect. 5.

3.3 Training and inference algorithms

The formulation above closely resembles control problems where a given controllable dynamical system is constrained to optimize a certain objective. However, our aim here is different as our goal is to find the dynamical system fitting a certain set of constraints, here provided through the observations Y. In practice, the optimization problem defined here can be solved using gradient descent methods. There are many methods to calculate the gradient and we briefly recall in Sect. 4 the adjoint method used in our experiments. In general, F_θ and g_θ can be parameterized as a neural network or as another parametric family, the only constraint being that it is differentiable almost everywhere with respect to θ.

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

Algorithm 1 Training Procedure in (in the experiments)

Input: Training samples $\{(Y_{-k+1:0}), Y_{1:l}\}$.
Guess initial parameters θ
while not converged **do**
  Randomly select sample sequence $\{(Y_{-k+1:0}), Y_{1:l}\}$
  $\widetilde{X}_0 \leftarrow g_\theta(Y_{-k+1:0})$
  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.

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.

The following section details the important step of gradient computation.

4 Calculating the gradient

We start by briefly recalling the adjoint state equation and the corresponding general algorithm. We then discuss the two main methods used to discretize it, and then end the section by discussing stability and robustness properties of the resulting gradient. The definition of the adjoint state equation and the general algorithm highlighted in theorem 1 are classical results, recalled here for completeness since they underlie our approach.

When the dynamics are learned as it is the case in our work, there are no closed formulae for the forward equation which must be discretized, thus inducing discretization errors. Propositions 3 and 4 presented in the Appendix, Sect. A, show that the gradient is still well-defined and doesn't amplify errors.

4.1 The adjoint state equation

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 *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 always continuous. $\langle \cdot, \cdot \rangle$ is the scalar product associated to the L^2 space over the compact spatial domain $\Omega \subset \mathbb{R}^d$ over which the vector fields are defined.

In order to construct a gradient descent algorithm to solve equation (3), we need to find the gradient of the cost functional under the given constraints, *i.e.* the differential of $\theta \to \mathbb{E}_Y \mathcal{J}(Y, \mathcal{H}(X^\theta))$. However, this implies calculating $\dfrac{\partial X^\theta}{\partial \theta}$, which is often computationally demanding, as it implies solving $\dim(\theta)$ forward equations, which is high in our case. The adjoint state method avoids those costly computations by considering the Lagrangian formulation of the constrained optimization problem. A classical calculation gives the expression stated in the following theorem:

Theorem 1 (Adjoint State Equation)

$$\nabla_\theta \mathcal{J} = \left(- \int_0^T \left\langle \lambda_t, \frac{\partial F_\theta(X_t^\theta)}{\partial \theta_i} \right\rangle \mathrm{d}t - \left\langle \lambda_0, \frac{\partial g_\theta}{\partial \theta_i} \right\rangle \right)_i \tag{5}$$

where λ is solution of

$$\frac{\mathrm{d}\lambda_t}{\mathrm{d}t} = A_t \lambda_t + B_t \tag{6}$$

solved backwards, starting with $\lambda_T = 0$, and where

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

and

$$B_t = 2(\partial_X \mathcal{H}(X_t^\theta))^\star (\mathcal{H}(X_t^\theta) - Y_t).$$

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

For completeness, a proof of this result is provided in the Appendix, Sect. B.1.

4.2 Approximate solutions

Theorem 1 gives us a way to calculate, for a given value of θ, the gradient of the constrained problem being solved. However, solving the forward and backward equations, namely Eqs. (2) and (6) isn't generally straightforward. They do not yield a closed form

solution and we must content ourselves with approximations. There are essentially two different ways to tackle this problem (Gunzburger, 2002): the *differentiate-then-discretize* approach, and the *discretize-then-differentiate* approach.[3]

In the *differentiate-then-discretize* approach, one directly approximates the equations using numerical schemes. Here, the approximation error to the gradient comes from the discretization error made in the solver for both the forward and backward equations. This method is used in the black box solvers in Chen et al. (2018). It has the advantage of allowing the use of non-differentiable steps in the solver. However, it can yield inconsistent gradients of the cost functional $\mathcal{J}$, the discretization of the adjoint equations depends on the studied problem and therefore must be carefully selected and tuned (Bocquet, 2012).

In a *discretize-then-differentiate* approach, a differentiable solver for the forward equations is used, *e.g.,* using an explicit Euler scheme $X^\theta_{t+\delta t} \approx X^\theta_t + \delta t F_\theta(X^\theta_t)$. Based on the solver's sequence of operations for the forward equations, the backward equations and the gradient can be directly obtained using automatic differentiation software (Paszke et al., 2017). This algorithm is actually equivalent to backpropagation (LeCun et al., 1988) which can be derived as a special case of it: As the step-size approaches zero, the forward and backward equations are recovered.

While the two methods are consistent and both converge to the equations derived in Theorem 1, they do not always yield the same results as they proceed differently. In our experience, the second one proved more stable and the fact that we were limited to differentiable solvers wasn't an obstacle. Moreover, in the Appendix, Sect. A, we derive some properties of the adjoint equation and the corresponding gradient which are reassuring regarding its stability and robustness to approximation errors.

5 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.

5.1 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 $l \leq d$, 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 Sect. 3.1, the following result shows that there is usually an infinite number of solutions to the optimisation problem.

Proposition 1 *If $l < d$ and the unobserved part of the state is non trivial, the non-parametric version of the optimization problem equation (3) admits an infinite number of null loss solutions which are distinct from canonical state representations.*

[3] The differentiate-then-discretize method is often referred to as the *continuous adjoint method*, and the *discretize-then-differentiate* approach as the *discrete adjoint method* (Sirkes and Tziperman, 1997).

A proof is given in the Appendix, Sect. B.2.

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.

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 Sect. 7, the properties of those two settings can be useful when dealing with real world data.

5.2 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 $\{(Y^{(i)}_{-k+1}, \dots, Y^{(i)}_0, \dots, Y^{(i)}_T)\}_i$. 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 observation forecasts?
- 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 *There exists an invertible function e which transforms jointly learned states into canonical states.*

A proof is given in the Appendix, Sect. B.3.

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.

5.3 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 $\{(X^{(i)}_0, Y^{(i)}_1, \dots, Y^{(i)}_T)\}_i$. 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.

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?
- Can we find a way to conserve the structure of the initial state throughout time-steps so that we **unsupervisedly learn** the dynamics of the hidden components of the state?

A first fact is that, for the same reasons outlined in the proof of proposition 5, 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 was chosen to be structurally conservative, 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.

This is one of the reasons we choose to parametrize F as a residual network in our experiments: ResNets tend to modify only slightly their input (see Hauser, 2019 or Jastrzkebski et al., 2017 for example) and we use this property successfully in Sect. 6.5 to learn canonical state representations in the case of the Navier–Stokes equations.

6 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 (Sect. 7).

6.1 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 Foias et al. (2001) for example.

Those equations are

$$\begin{aligned} \frac{\partial u}{\partial t} + (u \cdot \nabla)u &= -\frac{\nabla p}{\rho} + g + \nu \nabla^2 u, \\ \frac{\partial \rho}{\partial t} + (u \cdot \nabla)\rho &= 0, \\ \nabla \cdot u &= 0, \end{aligned} \tag{7}$$

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. (1) as we still have the pressure variable p as well as the null divergence constraint. However, the Helmholz-Leray decomposition result (Foias et al., 2001), 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. (7), 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.[4]

Thus, if u is solution to Eq. (7), it is also a solution of:

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

which is of the form of Eq. (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[5]:

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

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

In conclusion, we have:

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

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

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

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

6.2 Implementation and dataset details

6.2.1 The dataset

We have taken the observations to be the density of the fluid while the hidden components are the two-dimensional velocity field.

We have produced 600 separate simulations with **independently and randomly generated initial conditions**,[6] 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.[7]

As stated before, one also has to choose a training horizon T, to construct the used dataset of the form $\{(Y^{(i)}_{-k+1}, \dots, Y^{(i)}_0, \dots, Y^{(i)}_T)\}_i$ for **setting 1** and $\{(X^{(i)}_0, Y^{(i)}_1, \dots, Y^{(i)}_T)\}_i$ 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.

6.2.2 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 (He et al., 2016), 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 (Ronneberger et al., 2015). More precisely, we have used a modified variant of the FlowNetS architecture of Dosovitskiy et al. (2015) with:

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

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

- 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 results, 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 (2) in time, we use a simple Euler scheme. Note that the discretization step-size may differ from the time interval between consecutive observations; in our case, we apply 3 Euler steps between two observations, thus giving us *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 (Paszke et al., 2017). The use of a small learning rate was voluntary: in conjunction with the orthogonal initialization, this ensures that the weight matrices do not deviate too much from the orthogonality condition during training thus allowing for good gradient propagation and stable learning dynamics. We have also observed that those choices allow for results which are robust across different runs.

Training usually took a few hours on the datasets we describe here, which is comparable to other baselines (although PRNN took a few more hours to converge satisfyingly).

6.2.3 Baselines and metrics

We compare our models to two different baselines:

- **PKnI** It is a physics-informed deep learning model described in de Bézenac et al. (2018), 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** (Wang et al., 2018) 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:

$$\frac{1}{T} \frac{1}{|\Omega|} \sum_{k=1}^{T} \sum_{x \in \Omega} \frac{\|\mathcal{H}(X_k(x)) - Y_k(x)\|^2}{\|Y_k(x)\|^2} \tag{8}$$

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:

$$\frac{1}{K}\sum_{k=1}^{K}\frac{1}{|\Omega|}\sum_{x\in\Omega}\frac{\left\langle u^1(x),u^2(x)\right\rangle}{\left\|u^1(x)\right\|\left\|u^2(x)\right\|} \tag{9}$$

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).

6.3 Forecasting observations

Figure 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 1. Our model is able to predict observations up to a long forecasting horizon (results are shown for up to 30 steps in Fig. 1 and 50 steps in Table 1), which means that it has managed to learn the dynamical system. Note that for setting 2 in Fig. 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. Recall that the supervision is done here only at the level of observations, 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 Fig. 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 even though it doesn't produce meaningful hidden states, for the first few steps, produces less sharp predictions which explains its worse performance when evaluated on long term predictions. Additional samples shown in Figs. 2 and 3 confirm this observation.

6.4 Restructuring jointly trained states

Proposition 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 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

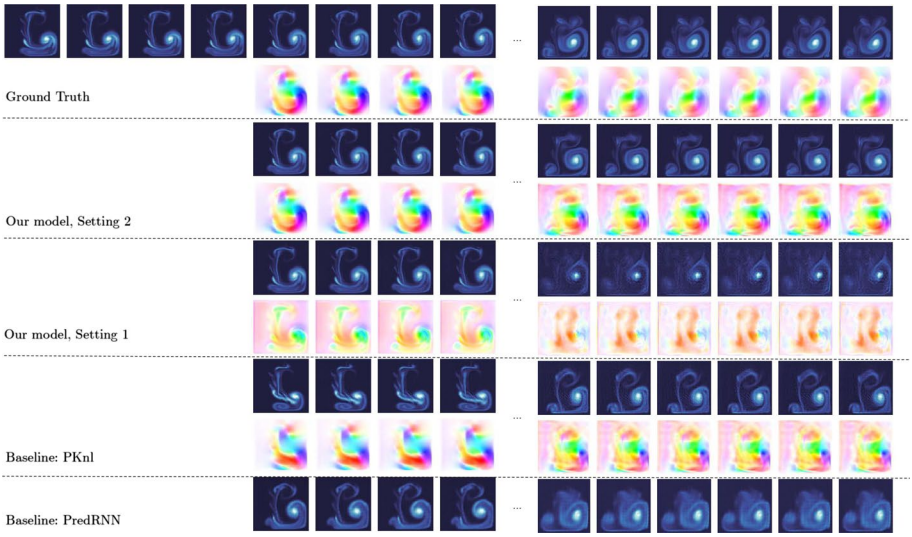

Fig. 1 Forecasting the Navier–Stokes equations 30 time-steps ahead with different models, starting from a given initial condition. In this figure as well as in the following ones, the velocity field is represented using the Middlebury Color Code as implemented in the flowlib library https://github.com/liruoteng/OpticalFlowToolkit/blob/master/lib/flowlib.py

Table 1 Relative MSE as in Eq. (8) for our model and different baselines, at different temporal horizons on the Navier–Stokes equations.

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 (de Bézenac et al., 2018)	0.194	0.221	0.752
PRNN (Wang et al., 2018)	0.170	0.227	0.719

Note that the Setting 2 model uses a full, true initial state as X_0 while the Setting 1 one only relies on observations

Bold values indicate that the best score for a given column

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,[8] it still shows promise and demonstrates that this approach could be applied in many cases.

6.5 Imposing the initial condition prescribes the hidden dynamics

Figure 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.

[8] 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.

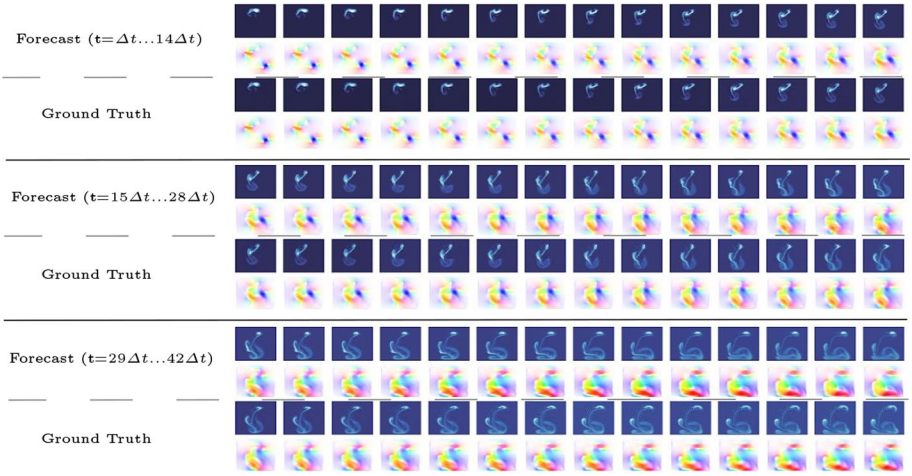

Fig. 2 Setting 2: Forecasting the Navier–Stokes equations, starting from a given initial condition (not shown here). We forecast 42 time-steps ahead and compare results with the ground truth simulation

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, 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 the values obtained with the PKnI model.

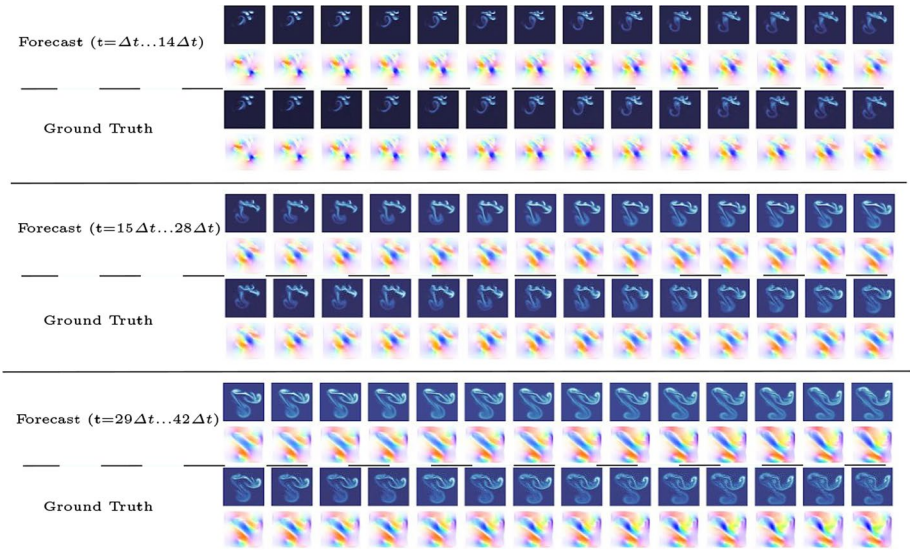

Fig. 3 Setting 2: Forecasting the Navier–Stokes equations, starting from a given initial condition (not shown here). We forecast 42 time-steps ahead and compare results with the ground truth simulation

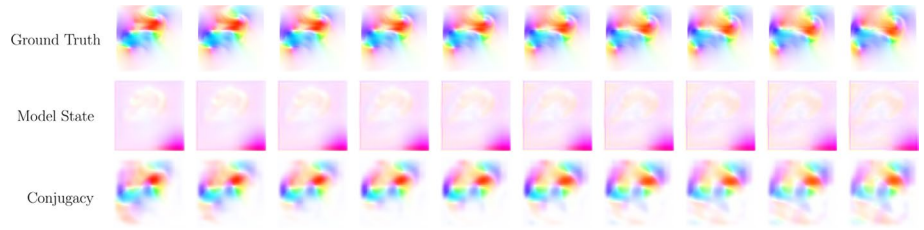

Fig. 4 Setting 1: Example of a sequence of hidden states transformed by the calculated conjugacy

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 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, Projection. 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.

6.6 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.

Table 2 Cosine similarity as in Eq. (9) 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

Bold values indicate that the best score for a given column

Table 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 (Setting 2)	0.118	0.798	0.180	0.679	0.628	0.483
Ours (Setting 2—projection)	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

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 preservation of the input code, which happens to be that of the canonical state because X_0 is fed into it. A similar kind of phenomenon is also empirically observed in the unsupervised domain translation field with the success of the CycleGAN model which is explored from this point of view in (de Bézenac et al., 2019).[9]

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. In practice, systems of interest may present a hybrid setting, the following will study an example of such a situation for a more complex, more realistic dynamical system.

7 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 suppose access to part of the hidden components of the state for the initial condition. This 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 this realistic, state-of-the-art simulation of ocean circulation, we then propose two instances of our model and compare them to standard baselines.

[9] Obviously, the jointly trained model can't be expected to learn a state corresponding to the canonical one as it is never provided with any structure.

7.1 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 (Madec, 2008) 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 Sect. 6, 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 $\tilde{w}_0$.

7.2 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 Fig. 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, taking as initial state

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

- **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:

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

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 $\tilde{w}_0$ nor in T_0. For E_θ, we use the UNet architecture (Ronneberger et al., 2015).

7.3 Results and conclusions

We have used the same hyper-parameters to build and train our architectures as for the Navier–Stokes simulations (described in Sect. 6.2). We also consider the same baselines. As a reviewer of this paper 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 Figs. 5, 6 and 7 as well as Table 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.

8 Discussion

In the machine learning community, the forecasting problem is often seen as a learning a neural network mapping consecutive states in time. In this work, we take an alternate approach, and use the neural network to express the rate of change of the states instead. This task is intrinsically simpler for the network, and is in fact the natural way to model time varying processes. This also allows to accommodate irregularly acquired observations as showed in Chen et al. (2018), and can also allow interpolation between observations.

In Sect. 5, we explore avenues in order to constrain the hidden dynamics. Typically, in Setting 1 (Sect. 5.2), if we have access to a small amount of observations of the full state, it is possible to map the hidden states learned by the neural network onto the canonical coordinate system. This opens up interesting directions for future exploration, as it possible to predict quantities of interest (velocity, pressure, etc...) from the states of the network, rendering the hidden dynamics of the network more interpretable.

Although the theoretical foundations of Setting 1 are well understood (Coudène, 2016), it is not the case for the setting 2. The fact that we learn dynamics closely resembling the dynamics of the underlying system by only giving as input the initial

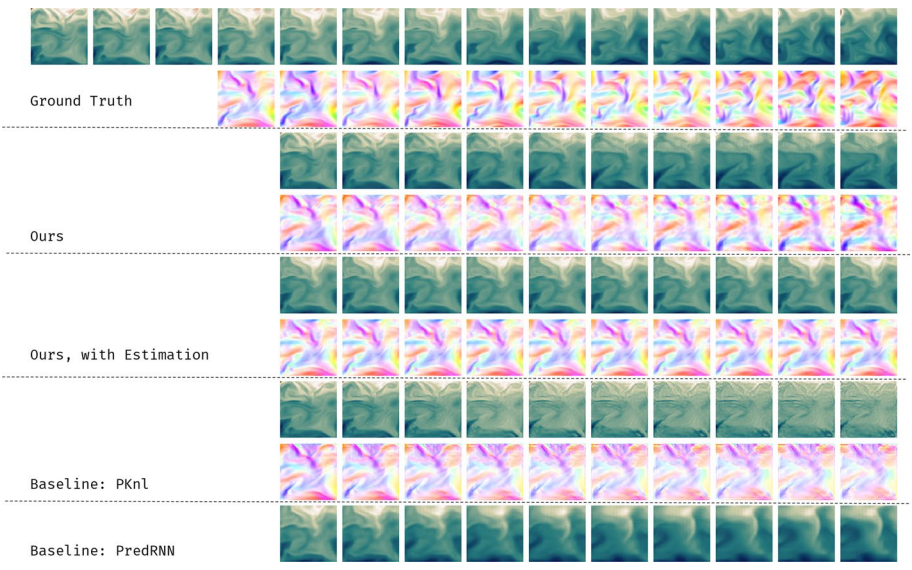

Fig. 5 Forecasting sea surface temperatures 10 time-steps ahead with different models, starting from a given initial condition

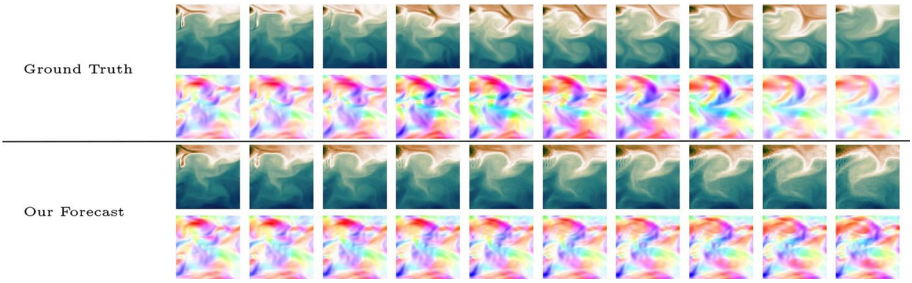

Fig. 6 Forecasting Glorys2v4 10 time-steps ahead, starting from a given, full state, initial condition (not shown here), wthout the estimation

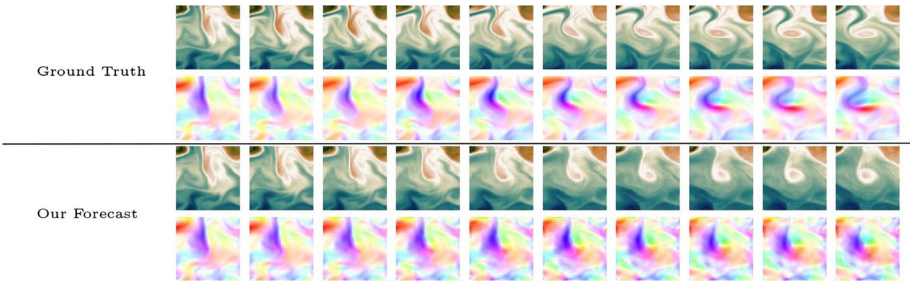

Fig. 7 Forecasting Glorys2v4 10 time-steps ahead, starting from a given, full state, initial condition (not shown here), without the estimation

Table 4 Relative MSE and cosine similarity scores for our models and different baselines, at different temporal horizons on the Glorys2v4 dataset

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

State estimation is not available for PRNN which is based on a recurrent network, hence the crosses in the "cosine" columns

Bold values indicate that the best score for a given column

condition (see Sect. 6.5) is intriguing. We have conducted an ablation study in order to better understand this phenomenon (Sect. 6.5), showing that the success in predicting the hidden states correctly without direct supervision is due to the proposed continuous-time framework and the particular architecture we used. However, the exact underpinnings are not entirely understood. As future work, we wish to develop the

theoretical aspects and implications of these results in order to shed light on the underlying mechanisms.

9 Conclusion

We present in this paper a general data-driven model for space-time processes when the state is only partially observable. We show that partial observability introduces ill-posedness in the determination of an interpretable state representation then propose two methods to solve this issue: This allows to demonstrate that non-structured states can be interpreted when correctly transformed and that the model, when fed with a structured interpretable state with a well parametrized evolution term, can forecast unsupervisedly the hidden dynamics of the state. The theoretical analysis is confirmed through experiments on raw simulations of the Navier–Stokes equations and comparisons with two competitive data-sets.

Appendix A: Properties of the adjoint equation

In this section, we derive some properties of the adjoint equation of Theorem 1.

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

Lemma 1 (Gronwall) *Let u be solution to*:

$$\partial_t u_t = \alpha_t u_t + \beta_t$$

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 3 (Stability of the gradient) Under the hypothesis in *Theorem* 1, λ *is defined and bounded over* $[0, T]$. *Thus*, $\frac{\partial}{\partial\theta}\mathcal{J}(Y, \mathcal{H}(X^\theta))$ *is also well-defined and bounded*.

Proof Using the lemma above, we have, using the same notations as in Theorem 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 the hypothesis above, $\partial_X F_\theta$, $\partial_X \mathcal{H}$ and $\mathcal{H}$ are continuous and $[0, T] \times \Omega$ is compact so that A and B are bounded. 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 $\frac{\partial}{\partial\theta}\mathcal{J}(Y, \mathcal{H}(X^\theta))$ 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 $[0,T]\times\Omega$. Then we have:

Proposition 4 (Robustness of the gradient) *Consider a perturbed solution* $\widetilde{X}$ *of the forward equation.*[10] *Then*:

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

for a real number M.

Proof We have, taking $\epsilon = \widetilde{\lambda} - \lambda$:

$$\partial_t \epsilon_t = \widetilde{A}_t \epsilon_t + (\widetilde{A}_t - A_t)\lambda_t + \widetilde{B}_t - B_t$$

We know that λ is bounded. Moreover:

$$\|\widetilde{A}_t - A_t\| \leq \|\partial^2_{XX} F_\theta\|_\infty \|\widetilde{X} - X\|_\infty$$

and:

$$\|\widetilde{B}_t - B_t\| \leq 2\|\partial^2_{XX}\mathcal{H}\|_\infty \|\mathcal{H}(\widetilde{X}_t) - \mathcal{H}(X)_t\| \leq 2\|\partial^2_{XX}\mathcal{H}\|_\infty \|\partial_X \mathcal{H}\|_\infty \|\widetilde{X} - X\|_\infty$$

Combining all those inequalities, we then have L such that:

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

Taking $\alpha_t = \widetilde{A}_t$, $\beta_t = (\widetilde{A}_t - A_t)\lambda_t + \widetilde{B}_t - B_t$ and recalling that $\epsilon_T = \widetilde{\lambda}_T - \lambda_T = 0$, we can then apply the Grönwall lemma to conlude. □

In other words, the gradient calculated through the adjoint method is robust in the sense that it doesn't amplify perturbations of the forward equation. This is important from a practical point of view as there is bound to be approximation errors when solving the forward equation. Moreover, noise in the data would also result in a perturbation of the resulting solution of the forward equation and has thus to be controlled.

Appendix B: Proofs

Proof of Theorem 1

Theorem 2 (Adjoint State Equation)

$$\partial_\theta \mathcal{J}(Y, \mathcal{H}(X^\theta)) = -\int_0^T \left\langle \lambda_t, \partial_\theta F_\theta(X_t^\theta) \right\rangle \mathrm{dt} - \langle \lambda_0, \partial_\theta g_\theta \rangle \tag{10}$$

[10] The perturbation can for example model approximation errors in solving the equation.

where λ is solution of:

$$\partial_t \lambda_t = A_t \lambda_t + B_t \tag{11}$$

solved backwards, starting with $\lambda_T = 0$, *and where*:

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

and

$$B_t = 2(\partial_X \mathcal{H}(X_t^\theta))^\star(\mathcal{H}(X_t^\theta) - Y_t)$$

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

Proof Let us define:

$$\begin{aligned}\mathcal{L}(X, \lambda, \mu, \theta) = \mathcal{J}(X) + \int_0^T \left\langle \lambda_t, \frac{dX_t}{dt} - F_\theta(X_t) \right\rangle \mathrm{dt} \\ + \langle \mu, X_0 - g_\theta \rangle\end{aligned} \tag{12}$$

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

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

which gives:

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

Straightforward calculus gives us:

$$\partial_\theta \mathcal{J}(X_t^\theta) = \int_0^T 2\left\langle \partial_X \mathcal{H}(X_t^\theta) \cdot \partial_\theta X_t^\theta, \mathcal{H}(X_t^\theta) - Y_t \right\rangle \mathrm{dt}$$

Let us fix θ and a variation $\delta\theta$. Then, we 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:

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

and:

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

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

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

so that:

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

We now have all elements to conclude calculating the derivative of $\mathcal{L}$, with some more easy calculus:

$$\begin{aligned}\partial_\theta \mathcal{L} = \int_0^T \big(&2\big\langle \partial_X \mathcal{H}(X_t^\theta) \cdot \partial_\theta X_t^\theta, \mathcal{H}(X_t^\theta) - Y_t \big\rangle +\\ &\big\langle \lambda_t, \partial_\theta \partial_t X_t^\theta - \partial_X F_\theta(X_t^\theta) \cdot \partial_\theta X_t^\theta - \partial_\theta F_\theta(X_t^\theta) \big\rangle\big) \mathrm{d}t\\ &+ \big\langle \mu, \partial_\theta X_0^\theta - \partial_\theta g_\theta \big\rangle\end{aligned}$$

By the Schwarz theorem, as X is twice continuously differentiable, we have that $\partial_\theta \partial_t X_t^\theta = \partial_t \partial_\theta X_t^\theta$. Integrating by parts, we get:

$$\begin{aligned}\int_0^T \big\langle \lambda_t, \partial_\theta \partial_t X_t^\theta \big\rangle \mathrm{d}t =& \big\langle \lambda_T, \partial_\theta X_T^\theta \big\rangle - \big\langle \lambda_0, \partial_\theta X_0^\theta \big\rangle\\ &- \int_0^T \big\langle \partial_t \lambda_t, \partial_\theta X_t^\theta \big\rangle \mathrm{d}t\end{aligned}$$

Putting all this together and arranging it, we get:

$$\begin{aligned}\partial_\theta \mathcal{L} = &\int_0^T \big\langle \partial_\theta X_t^\theta, 2\partial_X \mathcal{H}(X_t^\theta)^\star \big(\mathcal{H}(X_t^\theta) - Y_t\big)\\ &-\partial_t \lambda_t - \partial_X F_\theta(X_t^\theta)^\star \lambda_t \big\rangle \mathrm{d}t\\ &- \int_0^T \big\langle \lambda_t, \partial_\theta F_\theta(X_t^\theta) \big\rangle \mathrm{d}t + \big\langle \lambda_T, \partial_\theta X_T^\theta \big\rangle + \big\langle \mu - \lambda_0, \partial_\theta X_0^\theta \big\rangle\\ &- \langle \mu, \partial_\theta g_\theta \rangle\end{aligned}$$

We can now define:

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

and

$$B_t = 2(\partial_X \mathcal{H}(X_t^\theta))^\star(\mathcal{H}(X_t^\theta) - Y_t)$$

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

$$\partial_t \lambda_t = A_t \lambda_t + B_t$$

with final condition $\lambda_T = 0$. We also choose $\mu = \lambda_0$ so that, finally, we have:

$$\partial_\theta \mathcal{L} = -\int_0^T \left\langle \lambda_t, \partial_\theta F_\theta(X_t^\theta) \right\rangle \mathrm{dt} - \langle \lambda_0, \partial_\theta g_\theta \rangle$$

which concludes the proof. □

Proof of Proposition 5

Proposition 5 *If $l < d$ and the unobserved part of the state is non trivial, the non-parametric version of the optimization problem equation (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, $X_t = \left(Y_t, Z_t\right)$ 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,[11] and let $\phi_\sharp X$ be defined as:

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

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:

$$\begin{aligned} F^\phi(W) = &\left(F^{(1)}\left((\phi^{-1})_\sharp(X)\right),\right. \\ &\left.\partial_Z \phi\left(P^{(2)}\left((\phi^{-1})_\sharp(W)\right)\right) \cdot F^{(2)}\left((\phi^{-1})_\sharp(W)\right)\right) \end{aligned}$$

[11] 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. □

Proof of Proposition 2

Proposition 6 *There exists an invertible function g 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 $X^{\text{JT}}_t = e(Y_{t-K+1}, \cdots, Y_t)$. Then, if we denote by j the function:

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

we have that:

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

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 $X^{\text{JT}}_t \in S^{\text{JT}}$ there is a sequence of observations such that $X^{\text{JT}}_t = e(Y_{t-K\Delta t+1}, \dots, Y_t)$ and then taking the corresponding canonical state X^{can}_t we have that $X^{\text{JT}}_t = j(X^{\text{can}}_t)$. Thus j is surjective. □

Author contributions IA: Jointly elaborated the theory and conducted experiments, jointly wrote the paper. EB: Jointly elaborated the theory and conducted experiments, jointly wrote the paper. AP: Participated in

conducting some of the experiments, participated in discussions. PG: Supervision, participated in discussions and in writing the paper.

Funding ANR Project LOCUST, Award Number: Project-ANR-15-CE23-0027 Recipient: Patrick Gallinari CLEAR joint Lab. Thales - Sorbonne Universite, Award Number: None Recipient: Ibrahim Ayed

Declarations

Conflict of interest All authors declare taht they have no conflict of interest.

References

Alvarez, M. A., Luengo, D., & Lawrence, N. D. (2013). Linear latent force models using Gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 35*(11), 2693–2705.

Béréziat, D., & Herlin, I. (2015). *Coupling dynamic equations and satellite images for modelling ocean surface circulation* (pp. 191–205). Springer.

Bergen, K. J., Johnson, P. A., De Hoop, M. V., & Beroza, G. C. (2019). Machine learning for data-driven discovery in solid Earth geoscience. *Science, 363*, 6433.

Bocquet, M. (2012). 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. https://doi.org/10.1002/qj.961

Carrassi, A., Bocquet, M., Bertino, L., & Evensen, G. (2018a). Data assimilation in the geosciences: An overview of methods, issues, and perspectives. *Wiley Interdisciplinary Reviews: Climate Change, 9*(5). https://doi.org/10.1002/wcc.535

Carrassi, A., Bocquet, M., Bertino, L., & Evensen, G. (2018b). Data assimilation in the geosciences: An overview of methods, issues, and perspectives. *Wiley Interdisciplinary Reviews: Climate Change, 9*(5). https://doi.org/10.1002/wcc.535

Chen, R. T. Q., Rubanova, Y., Bettencourt, J., & Duvenaud, D. (2018) Neural ordinary differential equations. In: *NIPS*.

Coudène, Y. (2016). Conjugation (pp. 69–78). Springer. https://doi.org/10.1007/978-1-4471-7287-1_7

Crutchfield, J. P., & Mcnamara, B. S. (1987). Equations of motion from a data series. *Complex Systems, 66*, 452.

de Bézenac, E., Ayed, I., & Gallinari, P. (2019). *Optimal unsupervised domain translation*. CoRR arXiv:1906.01292

de Bézenac, E., Pajot, A., & Gallinari, P. (2018). Deep learning for physical processes: Incorporating prior scientific knowledge. In: *ICLR*.

Denton, E., & Fergus, R. (2018). Stochastic video generation with a learned prior. In J. Dy, & A. Krause (Eds.), *Proceedings of the 35th international conference on machine learning, proceedings of machine learning research (PMLR)* (vol. 80, pp 1174–1183). Stockholmsmässan.

Dosovitskiy, A., Fischer, P., Ilg, E., Hausser, P., Hazirbas, C., Golkov, V., Smagt, P.vd, Cremers, D., & Brox, T. (2015) *FlowNet: Learning optical flow with convolutional networks* (pp. 2758–2766). IEEE. https://doi.org/10.1109/ICCV.2015.316

Fablet, R., Ouala, S., & Herzet, C. (2017). *Bilinear residual neural network for the identification and forecasting of dynamical systems*. CoRR arXiv:1712.07003

Foias, C., Manley, O., Rosa, R., & Temam, R. (2001). *Navier–Stokes Equations and Turbulence. Encyclopedia of mathematics and its applications*. Cambridge University Press.

Franceschi, J. Y., Delasalles, E., Chen, M., Lamprier, S., & Gallinari, P. (2020). *Stochastic latent residual video prediction*. arXiv preprint arXiv:200209219

Gil, Y., Hill, M., Horel, J., Hsu, L., Kinter, J., Knoblock, C., Krum, D., Kumar, V., Lermusiaux, P., Liu, Y., North, C., Pierce, S. A., Pankratius, V., Peters, S., Plale, B., Pope, A., Ravela, S., Restrepo, J., Ridley, A., … Gomes, C. (2019). Intelligent systems for geosciences. *Communications of the ACM, 62*(1), 76–84.

Gunzburger, M. D. (2002). *Perspectives in flow control and optimization*. Society for Industrial and Applied Mathematics.

Hauser, M. (2019). *On residual networks learning a perturbation from identity*. CoRR arXiv:1902.04106

He, K., Zhang, X., Ren, S., & Sun, J. (2016). 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* (pp. 770–778).

Huntingford, C., Jeffers, E. S., Bonsall, M. B., Christensen, H. M., Lees, T., & Yang, H. (2019). Machine learning and artificial intelligence to aid climate change research and preparedness. *Environmental Research Letters*. https://doi.org/10.1088/1748-9326/ab4e55.

Jastrzkebski, S., Arpit, D., Ballas, N., Verma, V., Che, T., & Bengio, Y. (2017). *Residual connections encourage iterative inference*. CoRR arXiv:1710.04773

Kalinicheva, E., Ienco, D., Sublime, J., & Trocan, M. (2020). Unsupervised change detection analysis in satellite image time series using deep learning combined with graph-based approaches. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 13*, 1450–1466.

Karpatne, A., Watkins, W., Read, J., & Kumar, V. (2017). *Physics-guided neural networks (PGNN): An application in lake temperature modeling*. arXiv:1710.11431

LeCun, Y., & Touresky, D., Hinton, G., & Sejnowski, T. (1988). A theoretical framework for back-propagation. In *Proceedings of the 1988 connectionist models summer school, CMU* (vol. 1, pp. 21–28). Morgan Kaufmann.

Long, Z., Lu, Y., Ma, X., & Dong, B. (2018). PDE-Net: Learning PDEs from data (pp. 3214–3222). In: *ICML*.

Lorenc, A. C. (1986). Analysis methods for numerical weather prediction. *Quarterly Journal of the Royal Meteorological Society, 112*(474), 1177–1194. https://doi.org/10.1002/qj.49711247414

Madec, G. (2008). *NEMO ocean engine*. Note du Pôle de modélisation, Institut Pierre-Simon Laplace (IPSL), France, No 27, ISSN No 1288-1619.

Mathieu, M., Couprie, C., & LeCun, Y. (2016). Deep multi-scale video prediction beyond mean square error. In *International conference on learning representations*.

Nguyen, D., Ouala, S., Drumetz, L., & Fablet, R. (2019). *EM-like learning chaotic dynamics from noisy and partial observations*. https://doi.org/10.13140/RG.2.2.19493.96483

Ouala, S., Herzet, C., & Fablet, R. (2018). Sea surface temperature prediction and reconstruction using patch-level neural network representations. In *IGARSS 2018—2018 IEEE international geoscience and remote sensing symposium* (pp. 5628–5631).

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., & Lerer, A. (2017). *Automatic differentiation in pytorch*.

Racah, E., Beckham, C., Maharaj, T., Ebrahimi Kahou, S., Prabhat, M., & Pal, C. (2017). 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)* (pp. 3405–3416).

Raissi, M. (2018). Deep hidden physics models: Deep learning of nonlinear partial differential equations. *Journal of Machine Learning Research, 66*, 19.

Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2017). Machine learning of linear differential equations using Gaussian processes. *Journal of Computational Physics, 348*, 683–693.

Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2019). 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.

Reichstein, M., Camps-Valls, G., Stevens, B., Jung, M., Denzler, J., Carvalhais, N., & Prabhat, H. (2019). Deep learning and process understanding for data-driven Earth system science. *Nature, 566,* 195–204.

Robinson, J. C. (2010). *Dimensions, embeddings, and attractors. Cambridge Tracts in Mathematics*. Cambridge University Press. https://doi.org/10.1017/CBO9780511933912

Ronneberger, O., Fischer, P., & Brox, T. (2015). *U-net: Convolutional networks for biomedical image segmentation*. CoRR arXiv:1505.04597

Rudy, S. H., Brunton, S. L., Proctor, J. L., & Kutz, J. N. (2017). Data-driven discovery of partial differential equations. *Science Advances, 3*(4), e1602614.

Ruthotto, L., & Haber, E. (2018). *Deep neural networks motivated by partial differential equations*. arXiv:1804.04272

Shi, X., Chen, Z., Wang, H., Yeung, D. Y., Wk, Wong, & Wc, Woo. (2015). Convolutional LSTM network: A machine learning approach for precipitation nowcasting. *Advances in Neural Information Processing Systems, 28*, 802–810.

Sirignano, J., & Spiliopoulos, K. (2018). DGM: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics, 375*(Dms 1550918), 1339–1364. https://doi.org/10.1016/j.jcp.2018.08.029, arXiv:1708.07469

Sirkes, Z., & Tziperman, E. (1997). Finite difference of adjoint or adjoint of finite difference? *Monthly Weather Review, 125*(12), 3373–3378.

Takens, F. (1981). Detecting strange attractors in fluid turbulence. In *Symposium on dynamical systems and turbulence* (pp. 366–381).

Vandal, T., Kodra, E., Ganguly, S., Michaelis, A., Nemani, R., & Ganguly, A. R. (2018). 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, international joint conferences on artificial intelligence organization* pp (pp. 5389–5393). https://doi.org/10.24963/ijcai.2018/759

Voss, H., Timmer, J., & Kurths, J. (2004). Nonlinear dynamical system identification from uncertain and indirect measurements. *International Journal of Bifurcation and Chaos, 14*, 66.

Wang, Y., Gao, Z., Long, M., Wang, J., & Yu, P. S. (2018). *Predrnn++: Towards a resolution of the deep-in-time dilemma in spatiotemporal predictive learning*. arXiv:1804.06300

Weinan, E. (2017). A proposal on machine learning via dynamical systems. *Communications in Mathematics and Statistics, 5*, 1–11.

Zhang, S., & Lin, G. (2018). Robust data-driven discovery of governing physical laws with error bars. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Science, 474*(2217), 20180305. https://doi.org/10.1098/rspa.2018.0305