AGU ADVANCING EARTH AND SPACE SCIENCE

JAMES | Journal of Advances in Modeling Earth Systems®

RESEARCH ARTICLE
10.1029/2021MS002572

Learning Variational Data Assimilation Models and Solvers

R. Fablet[1], B. Chapron[2], L. Drumetz[1], E. Mémin[3], O. Pannekoucke[4], and F. Rousseau[5]

[1]IMT Atlantique, UMR CNRS Lab-STICC, Brest, France, [2]Ifremer, UMR CNRS LOPS, Brest, France, [3]INRIA Rennes, UMR CNRS IRMAR, Rennes, France, [4]INPT-ENM, UMR CNRS CNRM, CERFACS, Toulouse, France, [5]IMT Atlantique, UMR INSERM Latim, Brest, France

Key Points:

- We develop a end-to-end neural architecture to design variational data assimilation models and solvers
- We may jointly calibrate all the trainable components of the proposed scheme to optimize a data assimilation performance criterion
- Applied to Lorenz-63 and Lorenz-96 case-studies, the proposed approach may improve and speed up the reconstruction performance

Correspondence to:
R. Fablet,
ronan.fablet@imt-atlantique.fr

Citation:
Fablet, R., Chapron, B., Drumetz, L., Mémin, E., Pannekoucke, O., & Rousseau, F. (2021). Learning variational data assimilation models and solvers. *Journal of Advances in Modeling Earth Systems*, *13*, e2021MS002572. https://doi.org/10.1029/2021MS002572

Received 30 APR 2021
Accepted 29 SEP 2021

Abstract Data assimilation is a key component of operational systems and scientific studies for the understanding, modeling, forecasting and reconstruction of earth systems informed by observation data. Here, we investigate how physics-informed deep learning may provide new means to revisit data assimilation problems. We develop a so-called end-to-end learning approach, which explicitly relies on a variational data assimilation formulation. Using automatic differentiation embedded in deep learning framework, the key novelty of the proposed physics-informed approach is to allow the joint training of the representation of the dynamical process of interest as well as of the solver of the data assimilation problem. We may perform this joint training using both supervised and unsupervised strategies. Our numerical experiments on Lorenz-63 and Lorenz-96 systems report significant gain w.r.t. a classic gradient-based minimization of the variational cost both in terms of reconstruction performance and optimization complexity. Intriguingly, we also show that the variational models issued from the true Lorenz-63 and Lorenz-96 ODE representations may not lead to the best reconstruction performance. We believe these results may open new research avenues for the specification of assimilation models for earth systems, both to speed-up the inversion problem with trainable solvers but possibly more importantly in the way data assimilation systems are designed, for instance regarding the representation of geophysical dynamics.

Plain Language Summary Data assimilation is a key component in the modeling of earth systems to simulate their dynamics, forecast their evolution in the short-term or the long-term as well as to reconstruct earth systems' states from observation data. State-of-the-art data assimilation schemes generally blend prior knowledge on the underlying governing laws with available observation data. Here, we turn data assimilation into a physics-informed machine learning problem. Within a differentiable framework, we can learn from data not only a data assimilation solver but also jointly some representation of the inverse problem. Numerical experiments support the relevance of this end-to-end approach for chaotic dynamics informed by noisy and irregularly-sampled observations. This opens new research avenues for the design of physics-informed and data-constrained simulation, forecasting and reconstruction schemes for earth systems.

1. Introduction

In earth science, the reconstruction of the dynamics of a given state or process from a sequence of partial and noisy observations is referred to as a data assimilation issue. Data assimilation is at the core of a wide range of applications, including operational ones, with the aim to make the most of available observation data sets, including for instance both in situ and satellite-derived data, and state-of-the-art numerical models (Evensen, 2009). Broadly speaking, a vast family of data assimilation methods comes to the minimization of some energy or functional which involves two terms, a dynamical prior and an observation term. We may distinguish two main categories of data assimilation approaches (Evensen, 2009): variational data assimilation and statistical data assimilation. Variational data assimilation relies on a continuous state-space formulation and results in a gradient-based minimization of the defined variational cost. By contrast, statistical data assimilation schemes generally relies on iterative formulations of stochastic filtering techniques such as Kalman and particle filters.

Whereas data assimilation naturally applies to model-driven settings, where the formulation of the dynamical and observation models follow from some physical expertise, data-driven strategies have emerged relatively recently (Bocquet et al., 2020; Lguensat et al., 2017) as possible alternatives. The focus has mainly been given to the identification of data-driven representations of the dynamical model, which may be directly plugged into state-of-the-art assimilation frameworks, especially statistical ones (Bocquet et al., 2020; Lguensat et al., 2017; Ouala et al., 2018). Recent advances have also been reported for the data-driven identification of the dynamical model from partial, irregularly-sampled and/or noisy observation data (Bocquet et al., 2020; Nguyen et al., 2020; Raissi, 2018).

Here, we further explore how data-driven strategies and associated machine learning schemes may be of interest for data assimilation issues. Especially, end-to-end learning strategies aim to build so-called end-to-end neural network (NN) architectures so that one can learn all the trainable components of the architecture w.r.t. some target to be predicted from input data, whereas the traditional approach usually relies on designing each component relatively independently. Many fruitful applications of end-to-end learning strategies have been recently reported in the deep learning literature (Busta et al., 2017; Dieleman & Schrauwen, 2014; Schwartz et al., 2019), including for solving inverse problems in image and signal processing. In this work, we introduce a physics-informed end-to-end learning framework for data assimilation. We rely on a variational data assimilation formulation. The resulting end-to-end architecture uses as inputs a sequence of observations and delivers as outputs a reconstructed state sequence. The associated main contributions are as follows:

1. The proposed physics-informed end-to-end learning architecture combines two main components, a neural and differentiable implementation of the variational data assimilation cost and a gradient-based neural solver of some target loss function. The latter exploits ideas similar to optimizer learning (Andrychowicz et al., 2016; Li et al., 2018; Vilalta & Drissi, 2002) and directly benefits from automatic differentiation tools embedded in deep learning frameworks to implement an adjoint method for the considered dynamical model;
2. Given some training criterion, this end-to-end learning architecture provides new means to train all the trainable parameters of the considered neural schemes. Interestingly, we may consider as training criterion the classic observation-driven data assimilation cost in an unsupervised setup, as well as a reconstruction error assuming we are provided with groundtruthed data in a supervised setup;
3. We report numerical experiments on Lorenz-63 and Lorenz-96 dynamics, which support the relevance of the proposed framework w.r.t. classic variational data assimilation schemes. Trained schemes may speed up the assimilation process as neural solvers may greatly reduce the number of gradient-based iterations. They may also lead to a significant improvement of the reconstruction performance if groundtruthed data are available;
4. Intriguingly, our experiments question the way dynamical priors are defined and calibrated in data assimilation schemes and suggest that approaches based on the sole forecasting performance of the dynamical model may not be optimal for assimilation purposes.

This study is organized as follows. Section 2 introduces the considered variational formulation within a deep learning paradigm. We detail the proposed end-to-end learning framework in Section 3. Numerical experiments on Lorenz-63 and Lorenz-96 dynamics are reported in Section 4. We further discuss our key contributions in Section 6.

2. Problem Statement

We introduce in this section the considered variational formulation which provides the basis for the proposed end-to-end learning framework. Let us consider the following continuous state-space formulation

$$\begin{cases} \dfrac{\partial x(t)}{\partial t} &= \mathcal{M}\Big(x(t)\Big) + \eta(t) \\ y(t,p) &= x(t,p) + \epsilon\,(t), \forall t \in \{t_0, t_0 + \Delta t \ldots., t_0 + N\Delta t\}, \forall p \in \Omega_t \end{cases} \tag{1}$$

with x the considered time-dependent state in some space $\mathcal{X}$ such that $x(t) \in \mathcal{X}, \forall t$, $\mathcal{M}$ the dynamical model and $\{t_0, t_0 + \Delta t \ldots., t_0 + N\Delta t\}$ the observation times. The proposed framework applies to n D+ t processes, including for instance multivariate time processes as well as space-time processes. Observations $\{y(t_i)\}$ may

only be partial, meaning that some components of $y(t_i)$ are only observed over some observation domain Ω_{t_i} at time t_i. Ω_{t_i} may refer to a spatial domain for spatio-temporal dynamics or a list of indices for multivariate time process. Processes η and ϵ represent respectively model errors and observation errors (Evensen, 2009). They are usually assumed to be white noises.

The data assimilation issue, that is, the reconstruction of the hidden state sequence x given a series of observations $\{y(t_i)\}$ at time steps $\{t_i\}$, may be stated as the minimization of the following cost

$$U_\Phi\left(x,y,\Omega\right) = \lambda_1 \sum_i \left\| x(t_i) - y(t_i) \right\|^2_{\Omega_{t_i}} + \lambda_2 \sum_i \left\| x(t_i) - \Phi(x)(t_i) \right\|^2 \tag{2}$$

where $\|.\|^2_\Omega$ stands for the evaluation of the quadratic norm restricted to domain Ω, for example, $\|u\|^2_\Omega = \int_\Omega u(p)^2 dp$ for a scalar 2 d state u. This accounts both for an irregular space-time sampling of the observations as well as an assimilation time step smaller than the observation one. Φ is the flow operator

$$\Phi(x)(t) = x(t-\Delta t) + \int_{t-\Delta t}^{t} \mathcal{M}\left(x(u)\right) du \tag{3}$$

In the cost function (Equation 2), the first term is a weighted measure of the distance from x to the data while the second term evaluates the distance between the empirical and the theoretical dynamics considering the forecast model as imperfect. This cost function is known as the weak-constraint 4D-Var (see e.g. Trmolet, 2007). Note here, for the sake of simplicity we do not consider a background term often used to measure the distance from x to a given background state, and corresponding to a Tikhonov regularization. We also assume spherical covariance hypotheses for the model and observation error. More complex and spatial-correlated covariance structures could be accounted for in the proposed variational formulation. Overall, we may rewrite variational cost $U_\Phi(x,y,\Omega)$ according to the following formulation where the norms are implicitly evaluated over some predefined time window, typically from t_0 to $t_0 + N\Delta t$:

$$U_\Phi\left(x,y,\Omega\right) = \lambda_1 \left\| x - y \right\|^2_\Omega + \lambda_2 \left\| x - \Phi(x) \right\|^2 \tag{4}$$

with $\left\| x - y \right\|^2_\Omega$ the observation term and $\left\| x - \Phi(x) \right\|^2$ the dynamical prior. In a continuous-time formulation, x and y refer here to the continuous-time processes, where as in a discrete-time setting x and y refers to the concatenation of the states x_{t_i} and y_{t_i} from t_0 to $t_0 + N\Delta t$. Similarly, $\Omega = \{\Omega_{t_0},\ldots,\Omega_{t_0+N\Delta t}\}$ accounts for the observation patterns with possible gaps over the considered time window. Within a variational assimilation setting, the minimization of this variational cost typically exploits an iterative gradient-based scheme given some initial estimate $x^{(0)}$, the simplest one being a fixed-step gradient descent

$$x^{(k+1)} = x^{(k)} - \alpha \nabla_x U_\Phi\left(x^{(k)},y,\Omega\right) \tag{5}$$

with α the gradient step coefficient. In a discrete-time setting, $\nabla_x U_\Phi$ refers to the gradient of variational cost U_Φ w.r.t. x. In a time-continuous setting, $\nabla_x U_\Phi$ stands for the functional derivative of variational cost U_Φ w.r.t. x. The computation of gradient operator $\nabla_x U$ typically relies on the adjoint method (Blayo, 2008; Blum et al., 2009). The computational implementation of the adjoint method for operator $\mathbf{Id} - \Phi$ may derive from the continuous formulation of Φ as well as from the discretized version of dynamical model $\mathcal{M}$ to ensure the full consistency between the forward integration of model $\mathcal{M}$ and the adjoint formulation (Bannister, 2017). We may also point out that data assimilation schemes commonly use optimization schemes which are more complex than a fixed-step gradient descent but they also require the computation of the gradient operator.

Within a deep learning framework, given a neural implementation for operator Φ, one may straightforwardly use automatic differentiation tools, typically referred to as the backward step in deep learning, to compute $\nabla_x U$. Among the state-of-the-art deep learning framework, we may cite pytorch (https://www.pytorch.org/), tensorflow (https://www.tensorflow.org/) and jax (https://jax.readthedocs.io/) in Python. We may also cite automatic differentiation tools in Julia (https://www.juliadiff.org/). Here, we use pytorch. In this paper, we investigate such differentiable numerical schemes for operator Φ which may explicitly exploit physics-informed differential and integral operators as well as state-of-the-art neural network architectures. We also design neural architectures for the associated solver, i.e., an iterative gradient-based inversion algorithm based on variational cost (Equation 4). Interestingly, the resulting end-to-end neural architecture provides means both to learn a solver for a predefined variational setting such as (Equation 4) as well as to jointly learn operator Φ and the solver w.r.t. some performance criterion, which may be different from variational cost (Equation 4).

3. End-to-End Learning Framework

In this section, we detail the proposed end-to-end learning framework based on the neural implementation of variational formulation (Equation 4). We first introduce two different types of parameterizations for operator Φ in (Equation 4), which refer to explicit ODE-based and PDE-based representations (Section 3.1) and constrained CNN representations (Section 3.2). We then detail the neural solver in the proposed end-to-end scheme.

3.1. Explicit ODE/PDE-Based Formulation for Operator Φ

Here, we assume that we know the dynamical operator $\mathcal{M}$ in (Equation 1), or at least the parametric family it belongs to $\mathcal{M} \in \mathcal{F} = \left\{\mathcal{M}_\theta \text{ with } \theta \in \mathcal{A} \subset \mathcal{R}^N\right\}$ with N the number of parameters of the model:

$$\frac{\partial x(t)}{\partial t} = \mathcal{M}_\theta\left(x(t)\right) \tag{6}$$

Given the neural implementation of model $\mathcal{M}_\theta$, we can implement operator Φ as a residual network (He et al., 2016) using explicit integration schemes. Here, we consider a fourth-order Runge-Kutta scheme (Dormand & Prince, 1980), which leads to:

$$\Phi(x)(t) = x(t-\Delta) + \Delta \cdot \sum_{i=1}^{4} \alpha_i k_i\left(x(t-\Delta), \mathcal{M}_\theta\right) \tag{7}$$

where $\alpha_1 = \alpha_4 = 1/6, \quad \alpha_2 = \alpha_3 = 2/6, \quad$ and $\quad k_i\left(x(t), \mathcal{M}_\theta\right) = \mathcal{M}_\theta(x(t) + \beta_i \cdot \Delta \cdot k_{i-1}) \quad$ with $\quad k_0 = 0$, $\beta_1 = \beta_2 = \beta_3 = 1/2$ and $\beta_4 = 1$. In such formulations, operator Φ computes a one-step ahead prediction of the input state sequence. We may emphasize that the size of the output of the neural implementation of operator Φ is the same size as the input sequence. In a similar fashion, this also applies to PDE formulations and could be combined to the automatic generation of neural architectures from symbolic PDEs as proposed in (Pannekoucke & Fablet, 2020).

3.2. Constrained CNN Formulation for Operator Φ

From the interpretation of term $\left\|x - \Phi(x)\right\|^2$ in Equation 4 as a projection error, state-of-the-art neural network architectures may also seem appealing. Here, we rely on a discrete-time formulation for x over the considered time window. CNN architectures whose possible parameterization includes the identity operator $\Phi(x) = x$ shall be excluded as they would result in meaningless priors in minimization of energy (Equation 4). We consider the following parameterization (Fablet, Drumetz, & Rousseau, 2021; Fablet, Ouala, et al., 2021):

$$\Phi(x) = \phi\left(\psi(x)\right) \tag{8}$$

where operator ψ s a convolutional layer where the central values of all convolution kernels is set to zero such that $\psi(x)(s)$ at position s does not depend on variable $x(s)$ where s refers to a tuple (k, p) of a discrete time index k and a feature index . It would also apply to multivariate space-time tensors. ϕ is a CNN which composes a number of convolution and activation layers where the kernel size of all convolution layers is 1 along time and/or space dimensions of process x. This guarantees that identity is excluded from the possible parameterization for operator Φ. We refer to these architectures as GENN (Gibbs Energy Neural Network) as they can be regarded as a neural implementation of Gibbs energies (Perez, 1998). Contrary to the ODE-based parameterization introduced in the previous Section, the latter parameterization does not derive from a prior parameterization of dynamical operator $\mathcal{M}$. In addition, it does not involve a sequential representation as the ODE-derived one. Through the convolutional operator, $\Phi(x)(t)$ depends on some time neighborhood $x(t-\delta),\ldots,x(t+\delta)$ where δ relates to the kernel width of the convolution operators as well as the number of convolution layers. We may however interpret operator Φ as a discretized numerical scheme of an ODE or a PDE (Pannekoucke & Fablet, 2020) as $\forall t, p, \Phi(x)(t,p)$ only involves a local neighborhood in the time or space-time domain.

Interestingly, within this category of CNN representations for operator Φ, we may easily consider multi-scale representations to capture patterns of interest at different scales. It comes to combine the architecture

defined by (Equation 8) to upsampling and downsampling operators, which are available in deep learning frameworks. Here, we consider two-scale representations

$$\Phi(x) = \mathcal{U}\Big(\Phi_1\big(\mathcal{D}(x)\big)\Big) + \Phi_2(x) \tag{9}$$

where operators $\Phi_{1,2}$ follow the constrained parameterization introduced above. $\mathcal{D}$ is a downsampling operator implemented as an average pooling layer and $\mathcal{U}$ an upsampling operator implemented as a ConvTranspose layer. In the deep learning literature, such architectures are referred as U-Nets (Cicek et al., 2016). Given that operator Φ_1 uses as inputs only a downsampled version of the input data through operator Dw, using the same convolution kernel widths for operators $\Phi_{1,2}$ they do not access the same scales. Though there might be some scale overlap between Φ_1 and Φ_2, they also inform specific scale ranges, the lowest frequencies for Φ_1 and the highest frequencies for Φ_2, such that we expect they do not solely arbitrarily compensate and embed a representation of key multi-scale features.

3.3. End-to-End Architecture

Given a neural formulation for operator Φ, we design a neural solver for the targeted minimization based on criterion (Equation 4). Using automatic differentiation tools embedded in deep learning frameworks, we can evaluate the gradient of variational cost (Equation 4) w.r.t. variable x, denoted as $\nabla_x U$. Inspired by meta-learning schemes (Andrychowicz et al., 2016; Hospedales et al., 2020), we can design recurrent neural networks to implement gradient-based solvers for the targeted data assimilation issue. Here, we investigate two types of solvers corresponding to the following iterative updates:

1. A LSTM-based solver, which updates the state at the k^{th} iteration as

$$\begin{cases} g^{(k+1)} &= LSTM\Big[\alpha \cdot \nabla_x U_\Phi\Big(x^{(k)}, y, \Omega\Big), h(k), c(k)\Big] \\ x^{(k+1)} &= x^{(k)} - \mathcal{L}\Big(g^{(k+1)}\Big) \end{cases} \tag{10}$$

where α is a scalar parameter, $\{h(k), c(k)\}$ the internal states of the LSTM model and $\mathcal{L}$ a linear layer to map the LSTM output to the space spanned by state x. LSTM-based updates are the classical parameterization of meta-learning schemes. Through the ability of LSTM models to capture long-term and short-term correlations, the resulting gradient-based updates can be regarded as an alternative to momentum-based gradient descent algorithms (LeCun et al., 2015).

2. A CNN solver, which updates the state at the k^{th} iteration as

$$\begin{cases} g^{(k+1)} &= \mathcal{C}\Big[\alpha \cdot \nabla_x U_\Phi\Big(x^{(k)}, y, \Omega\Big), g^{(k)}\Big] \\ x^{(k+1)} &= x^{(k)} - \mathcal{A}\Big(g^{(k+1)}\Big) \end{cases} \tag{11}$$

where $\mathcal{C}$ is a CNN with a sequence of convolutional layers with Relu activations which uses as inputs the concatenation of gradient $\nabla_x U_\Phi\Big(x^{(k)}, y\Omega\Big)$ and previous update $g^{(k)}$. Here, $\mathcal{A}$ refers to an atan activation to avoid exploding updates, especially in the early steps of the learning process.

These two types of updates are used as residual blocks to design a residual network (ResNet) (He et al., 2016) with a predefined number of iterations (typically, from 5 to 20 in our experiments). Overall, as sketched in Figure 1 the resulting end-to-end architecture uses as inputs an initial state $x^{(0)}$, an observation series y and the associated observation domain Ω to account for missing values and aims to reconstruct the hidden state x. In terms of deep learning architectures, it combines the neural implementation of variational cost $U_\Phi(x, y)$ and the iterative gradient-based neural solver, denoted as Γ. Let us denote by $\Psi_{\Phi,\Gamma}(x^{(0)}, y, \Omega)$ the resulting end-to-end model.

3.4. Learning Setting

Given the proposed end-to-end architecture, we may consider different learning strategies. The first strategy assumes that a calibrated implementation of operator Φ is available in the chosen differentiable framework. We may be provided with an ODE or PDE representation for dynamical model $\mathcal{M}$ or we may perform as a preliminary step the supervised identification of operator Φ using some representative data set for state

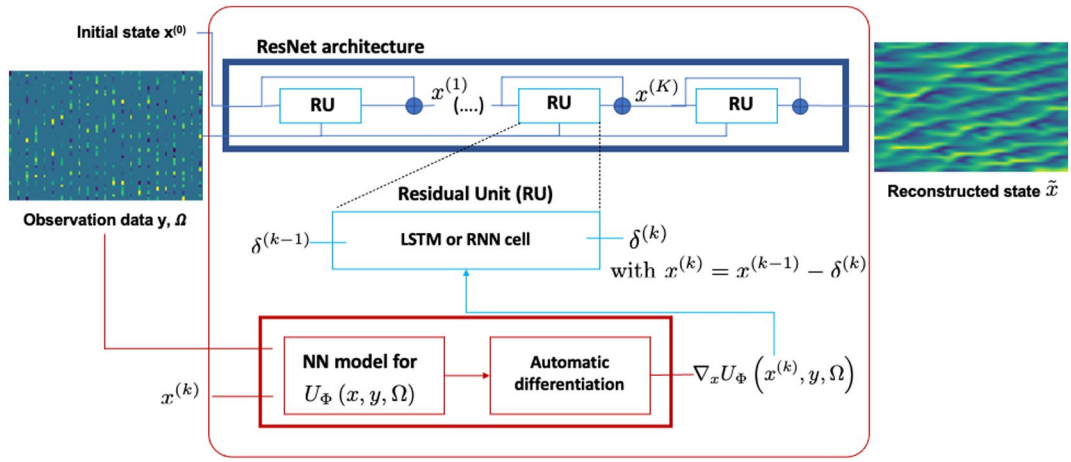

Figure 1. Sketch of the proposed end-to-end architecture in the inference mode: given a partial observation y with missing data mask Ω and an initialization $x^{(0)}$ for the unknown state x to be reconstructed, the proposed neural network architecture relies on an iterative gradient-based solver. It exploits a residual architecture with a predefined number of residual blocks, where the kth residual unit uses as input the gradient $\nabla_x U_\Phi\left(x^{(k-1)}, y, \Omega\right)$ of variational cost U_Φ w.r.t. state x evaluated for the output of the previous residual step. Gradient $\nabla_x U_\Phi\left(x^{(k-1)}, y, \Omega\right)$ derives from automatic differentiation tools embedded in deep learning frameworks, e.g., autograd function in pytorch.

sequence x, for example, (Fablet et al., 2018; Raissi et al., 2019; Pannekoucke & Fablet, 2020). In such a situation, given some observation data set comprising a number of observation series $\{y_1,\ldots,y_N\}$ with associated missing data masks $\{\Omega_1,\ldots,\Omega_N\}$, we can address the learning of NN solver Γ as the minimization of variational cost (Equation 4). For the sake of simplicity, in the following, we drop the subscript index corresponding to a time index and we only include a subscript referring to the index of the sample in the considered data set. For instance, Ω_n refers to observation mask for the n^{th} sample of the data set, which comprises a series of masks $\{\Omega_{n,t_0},\ldots,\Omega_{n,t_0+N\Delta}\}$ at time steps t_0 to $t_0 + N\Delta$. This first strategy then leads to the following learning loss

$$\mathcal{L} = \sum_n U_\Phi\left(\Psi_{\Phi,\Gamma}(x_n^{(0)}, y_n, \Omega_n), y_n, \Omega_n\right) \tag{12}$$

where parameter $\lambda_{1,2}$ in the definition of energy U_Φ in (Equation 4) are set *a priori*. This learning strategy is the standard criterion considered in variational data assimilation. It may be regarded as a non-supervised setting in the sense no groundtruthed data is available for the reconstruction of state x from observation data (y,Ω).

Interestingly, we may also consider a second strategy with a supervised setting where the training data set comprises observation series $\{y_1,\ldots,y_N\}$, associated missing data masks $\{\Omega_1,\ldots,\Omega_N\}$ and true states $\{x_1,\ldots,x_N\}$. Here, we can consider as learning loss the minimization of the reconstruction error

$$\mathcal{L} = \sum_n \left\| x_n - \Psi_{\Phi,\Gamma}\left(x_n^{(0)}, y_n, \Omega_n\right)\right\|^2 \tag{13}$$

this is a classic supervised strategy where we aim to train the end-to-end architecture so that the reconstruction error of the true state given the observation sequence is minimized. We may point out that, in this supervised setting, the gradient used as input in the trainable solver is not the gradient of the training loss but the gradient of the variational cost, whose computation only involves observation data and not the true states.

We may point out that both for the unsupervised and supervised strategies the backpropagation of the gradient of the end-to-end architecture involves a gradient of a gradient, more precisely the gradient (w.r.t. trainable parameters) of the gradient of variational cost (Equation 4) (w.r.t. state variable x). These two gradient stages exploit the automatic differentiation. This means we apply an automatic differentation onto the computational graph derived from an automatic differentiation of variational cost (Equation 4).

We noted experimentally that additional losses corresponding to projection error $\|x - \Phi(x)\|^2$ averaged over true and reconstructed states act as regularization terms for the training and were considered in our experiments. Overall, we implement all models using pytorch framework and the Adam optimizer. We typically increase incrementally the number of iterations of the gradient-based NN Solver (typically from 5 iterations to 20 ones). We let the reader refer to the code available online (https://doi.org/10.5281/zenodo.5266407) for additional details on the implementation.

4. Numerical Experiments

This Section reports numerical experiments with the proposed framework for Lorenz-63 and Lorenz-96 systems, which are widely considered for demonstration and evaluation purposes in data assimilation and are among the typical case-studies considered in recent data-driven and learning-based studies (Bocquet et al., 2020; Lguensat et al., 2017; Raissi, 2018).

When addressing a specific case-study, the proposed framework involves the definition of three main components:

1. The definition of the variational cost, especially operator Φ (Equation 4). Operator Φ may be known *a priori* with predefined parameters or involves trainable parameters within a chosen parametric family of operators;
2. The definition of the parameterization of the trainable solver Γ;
3. The selection of a supervised training loss (Equation 13) or of an unsupervised loss (Equation 12), which depends on the availability of groundtruthed data sets.

While we can consider any combination of these three components, we specifically investigate in these experiments the relevance of a joint learning of operator Φ and solver Γ as well as of the impact of the selected training loss.

4.1. Lorenz-63 Dynamics

We first perform numerical experiments for the assimilation of Lorenz-63 dynamics from partial and noisy observations. Lorenz-63 system is governed by the following three-dimensional ODE:

$$\begin{cases} \dfrac{dX_{t,1}}{dt} = \sigma\left(X_{t,2} - X_{t,2}\right) \\ \dfrac{dX_{t,2}}{dt} = \rho X_{t,1} - X_{t,2} - X_{t,1}X_{t,3} \\ \dfrac{dX_{t,3}}{dt} = X_{t,1}X_{t,2} - \beta X_{t,3} \end{cases} \tag{14}$$

with the following parameterization: $\sigma = 10$, $\rho = 28$, and $\beta = 8/3$. It generates chaotic patterns (Lorenz, 1963). In our experiments, we simulate Lorenz-63 time series with a 0.01 time step using a RK45 integration scheme (Dormand & Prince, 1980). We consider time series with 200 time steps. The observation data are sampled every 8 time steps for the first component of the Lorenz-63 state only and involve a Gaussian additive noise with a variance of 2. The test data set comprises 2,000 sequences and the training data with groundtruthed states contain 10,000 sequences.

Regarding operator Φ, we consider two approaches as discussed in Section 3:

1. A differentiable implementation of an ODE-based representation using a 4^{th}-order Runge-Kutta integration scheme of the ODE. The ODE operator is implemented as bilinear convolutional layers with a total of 9 parameters as proposed in (Fablet et al., 2018);
2. A constrained CNN representation. We consider a two-scale neural architecture introduced in (Equation 9), where operators $\Phi_{1,2}$ involve bilinear convolutional blocks with 30 channels. This architecture involves a total of ≈15,000 parameters.

Table 1
Lorenz-63 Experiments

Learning scheme	Model	Optim	R-score	4DVar-score	ODE-score
No learning	L63-RK4	FSGD	3.55	7.70e−3	<1e−4
Unsupervised learning	L63-RK4	aG-Conv	3.52	1.07e−2	<1e−4
		aG-LSTM	4.52	1.40e−2	<1e−4
	$L63\text{-}GENN_1$	aG-Conv	25.3	1.98e−2	5e−3
		aG-LSTM	11.6	1.85e−2	5e−3
Supervisd learning	L63-RK4	aG-Conv	2.82	9.24e−2	**<1e−4**
		aG-LSTM	4.00	9.36e−2	**<1e−4**
	$L63\text{-}GENN_1$	aG-Conv	**1.34**	1.10e−1	0.16
		aG-LSTM	1.62	1.27e−1	0.14

Note. We report the reconstruction performance of the proposed framework using unsupervised and supervised learning schemes. We consider two types of representations of the dynamics through operator Φ (See the main text for details): a differentiable implementation of the known ODE using a RK4 integration scheme (L63-RK4) and a two-scale GENN representation (L63-GENN). Regarding NN solver Γ, we consider two types of architectures using automatic differentiation to compute the gradient of assimilation cost (Equation 4) w.r.t. State x: LSTM-based solver (aG-LSTM) and CNN-based ones (aG-CNN). As baseline for a standard variational data assimilation scheme, referred to as FSGD, We report the performance of a fixed-step gradient descent for assimilation cost (Equation 4) with the known ODE model and a fourth-order Runge-Kutta (RK4) Integration scheme. We evaluate three performance metrics: The mean square error of the reconstruction of the true state (R-score), the value of the assimilation cost (Equation 4) for the reconstructed state x (4DVar-score) and the mean square error of the one-step-ahead prediction of the consider dynamical prior Φ (ODE-score). Bold values refer to the best score.

We consider both LSTM-based and CNN-based versions of solver Γ. They involve respectively ≈3,000 and ≈4,000 parameters. We let the reader refer to the Python notebook available online (https://doi.org/10.5281/zenodo.5266407) for implementation details.

We synthesize the performance metrics of these experiments in Table 1 for the different parameterizations of operator Φ and the associated solvers using either a supervised learning strategy or an unsupervised one (See Section 3). As baseline for a standard variational data assimilation scheme, we consider a fixed-step gradient descent (FSGD) of assimilation cost (Equation 4) using a fourth-order Runge-Kutta integration scheme for the known ODE model. This baseline is also implemented in Pytorch. All trained solvers involve 20 gradient-based iterations. For the FSGD, we let the minimization converge and chose a small gradient step to guarantee the convergence. As evaluation metrics, we consider the reconstruction error of the true state (mean square error) (R-score), the assimilation cost (4DVar-score) and the mean square error for the one-step-ahead dynamical prior $\|x - \Phi(x)\|^2$ (ODE-score). To compute comparable assimilation costs across methods, we report the assimilation cost for all schemes with $\lambda_1 = 0.01$ and $\lambda_2 = 1.0$, although these parameters may differ from the values learned during the training phase for supervised settings.

From Table 1, we may first notice that all schemes, which aim at minimizing assimilation cost (Equation 4), that is, the FSGD and unsupervised solvers, lead to worse reconstruction performance (R-score about 3.5) compared with supervised schemes (R-score up to 1.34). Conversely, the former leads to significantly better 4DVar-scores. These results emphasize that the assimilation cost may not be a very good indicator of the reconstruction performance. For instance, the trained aG-Conv solver leads to a slightly better performance than the FSGD (R-score of 3.52 vs. 3.55) but with a greater assimilation cost (4DVar-score of 1.05e−2 vs. 7.70e−3). In the unsupervised setting, the use of a GENN prior results in very poor reconstruction performance, which may relate to a worse prediction performance (ODE-score of 2.0e−4 vs. 3.7e−5 for a fourth-order Runge-Kutta scheme with the true ODE model).

Importantly, the supervised schemes greatly improve the reconstruction performance compared with the baseline (R-score of 1.34 for the best supervised scheme vs. 3.55 for the FSGD scheme). The best performance is indeed reported for a joint learning of operator Φ and of the associated solver, using a GENN parameterization for operator Φ. The improvement is also significant compared to the supervised learning of

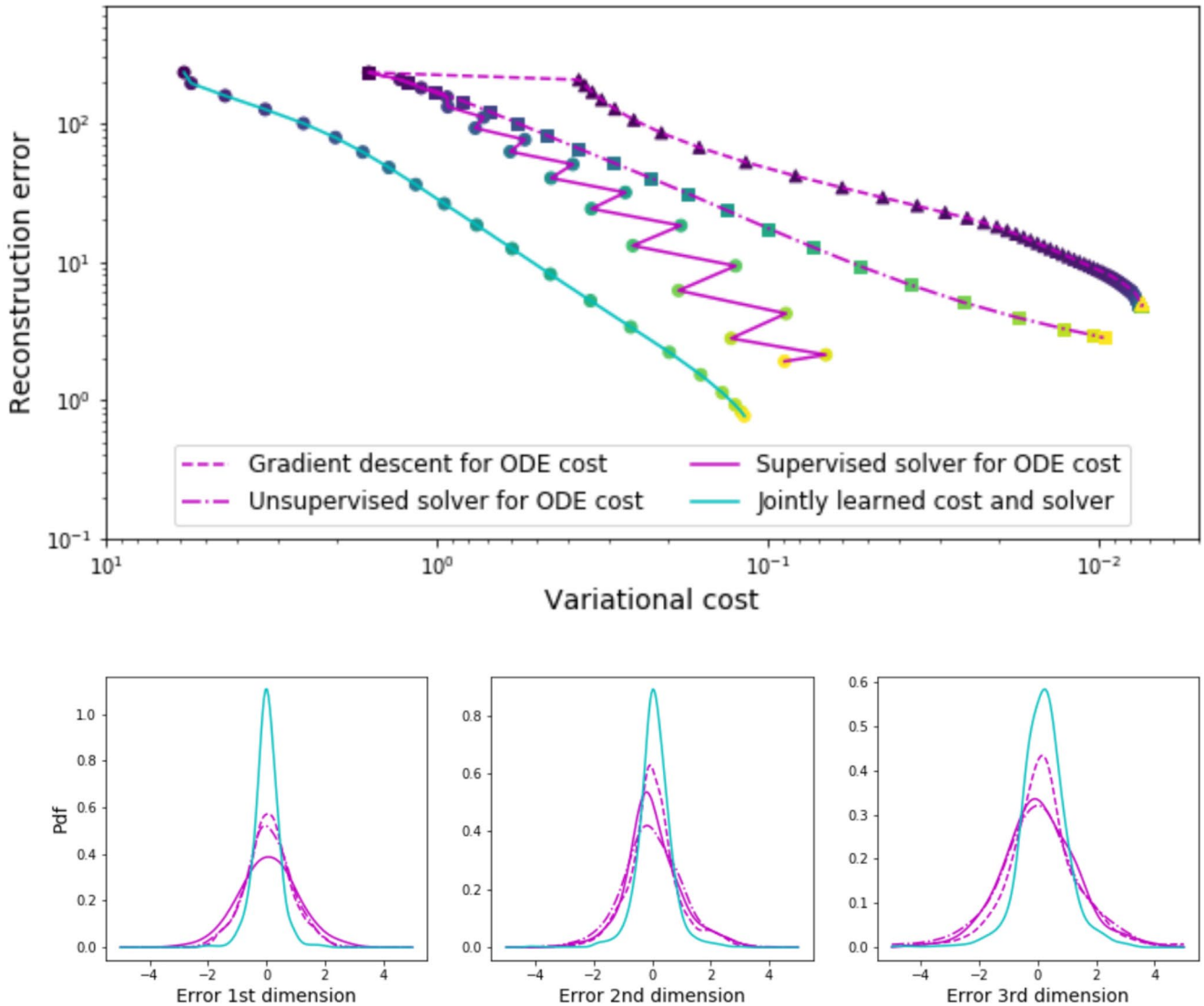

Figure 2. Reconstruction of Lorenz-63 dynamics. Upper panel: solvers' energy pathways using different parameterizations for operator Φ in (Equation 4) and the iterative gradient-based solver. We depict along the sequence of iterations of a solver the evolution of the reconstruction error and of the variational cost. We first consider a neural implementation of cost (Equation 4), referred to as ODE cost, with a parameterization of operator Φ based on a fourth-order Runge-Kutta scheme for the known Lorenz-63 ODE. For this ODE cost, we report the energy pathways for a fixed-step gradient descent (magenta dashed) and neural solvers trained using unsupervised (magenta, dashed-dotted) and supervised (magenta, solid) learning schemes. We also report the energy pathway for a joint supervised learning of the variational model and of the solver. Lower panel: associated pdfs of the reconstruction error for each component of the Lorenz-63 state.

a solver using the discretized version of the true ODE model for operator Φ. Intriguingly, the learnt GENN-based operator Φ is characterized by a relatively poor one-step-ahead prediction error (ODE-score of 0.16 vs. 3.7e−5 for the ODE-based schemes). These results suggest that the best dynamical prior for assimilation purposes might not be the discretization of the true ODE model. The detailed analysis of the values of the assimilation cost further points out that its direct minimization may not be the best strategy to optimize the reconstruction performance.

We further illustrate these results in Figure 2 and Figure 3. For a randomly-sampled subset of the test data set, we report in Figure 2 the energy pathways of different solvers: namely the baseline (FSGD), the best unsupervised and supervised solvers using the true ODE model for operator Φ, and the best supervised solver for a GENN-based operator Φ. Here, the FSGD scheme involves more than 150,000 steps, whereas all the learned solvers involve only 20 gradient-based iterations. In these experiments, the CNN-based solver

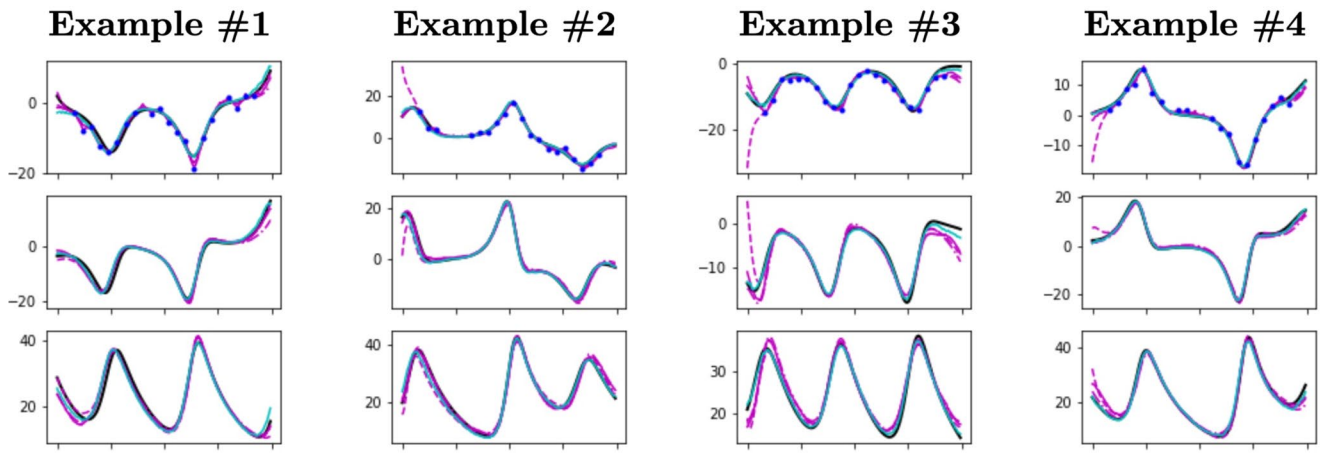

Figure 3. Reconstruction examples for Lorenz-63 dynamics: from left to right, we report four examples of 200-step Lorenz-63 sequences with true states (black, solid), observed values for the first component (blue dots). We refer the reader to Figure 2 for the details of the different reconstruction schemes (magenta and cyan).

always reach a better performance than the LSTM-solver. Interestingly, the joint learning of operator Φ and of the associated solver leads to a smooth descent, which may indicate that the training phase converges towards a new variational cost with a better agreement between the minimization of this cost and the reconstruction performance.

The distributions of the error of the reconstruction for the three components of the Lorenz-63 state in Figure 2 along with different examples in Figure 3 further emphasize the better reconstruction performance of the joint learning of a GENN-based operator Φ and a aG-Conv solver, which leads to reconstruction patterns very similar to the true ones.

4.2. Lorenz-96 Dynamics

The second experiment involves Lorenz-96 dynamics which are governed by the following ODE

$$\frac{dx_{t,i}}{dt} = \left(x_{t,i+1} - x_{t,i-2}\right)x_{t,i-1} - x_{t,i} + F \tag{15}$$

with i the index from 1 to 40 and F a scalar parameter set to 8. The above equation involves a periodic boundary constraint along the multivariate dimension of state $x(t)$. In our experiments, we simulate Lorenz-96 time series with a 0.05 time step using a RK45 integration scheme (Dormand & Prince, 1980). We consider time series with 200 time steps. The observation data are sampled every four time steps and involve a Gaussian additive noise with a variance of 2. Only 20 of the 40 components of the states are observed according to a random sampling. The test data set comprises 256 sequences and the training data with groundtruthed states 2,000 sequences.

Regarding operator Φ, we consider two architectures as discussed in Section 3:

1. A differentiable implementation of an ODE-based representation, referred to as L96-RK4 based on a 4^{th}-order Runge-Kutta integration scheme of the ODE. Our implementation involves bilinear convolutional layers with a periodic boundary conditions. This representation comprises a total of nine parameters;
2. A constrained CNN representation referred to as L96-GENN. We consider a two-scale neural architecture (Equation 9), where operators $\Phi_{1,2}$ involve bilinear convolutional architectures with a total of $\approx$ 50,000 parameters.

As described in Section 3, we consider both LSTM-based and CNN-based versions of solver Γ. They involve respectively $\approx$1,000 and x $\approx$ 1,500 parameters. We let the reader refer to the code made available for implementation details (https://doi.org/10.5281/zenodo.5266407).

Table 2
Lorenz-96 Experiments

Learning scheme	Model	Optim	R-score	4DVar-score	ODE-score
No learning	L96-RK4	4DVar	1.06	1.47e−2	<1e−4
Unsupervised learning	L96-RK4	aG-Conv	1.00	1.66e−2	<1e−4
		aG-LSTM	1.32	1.91e−2	<1e−4
Supervised learning	L96-RK4	aG-Conv	0.82	2.12e−2	**<1e−4**
		aG-LSTM	0.97	3.15e−2	**<1e−4**
	L96-GENN	aG-Conv	0.49	7.47e−2	12.20e−2
		aG-LSTM	**0.38**	5.30e−2	7.20e−2

Note. We report the reconstruction performance of the proposed framework using unsupervised and supervised learning schemes. We let the reader refer to Table 1 and the main text for details on the considered schemes and evaluation criteria. Bold values refer to the best score.

Similarly to Lorenz-63 experiments, we first report in Table 2 a quantitative comparison of different parameterizations of operator Φ and solver Γ. The best unsupervised solver for a variational cost based on the known ODE reaches the same reconstruction performance as the baseline using only 20 gradient-based iterations, to be compared with several thousands for the FSGD solver. Again, the CNN-based solver outperforms the LSTM-based one in the unsupervised setting. We do not report the performance for the GENN-based representation in the unsupervised setting as it behaves very poorly similarly to Lorenz-63 experiments. Besides, the reconstruction performance is greatly improved when considering a supervised setting, especially when we jointly learn operator Φ and solver Γ with a relative gain greater than 50% compared with the baseline. In this case, the best performance is achieved with a LSTM-based solver. Again, the best reconstruction performance does not come with a good one-step-ahead prediction score for the learned GENN-based representation.

We further illustrate these experiments in Figure 4. Here, all solvers' energy pathways are consistent with minimization patterns both for the reconstruction error and the assimilation cost. These pathways are very similar for the three solvers when using the variational cost based on the known ODE. Visually, they lead to similar error distributions and patterns though the FSGD solver involves greater errors. Visually, we can note a clear improvement when considering a joint supervised learning of the variational model and solver. The latter combines a GENN-based parameterization for operator Φ and a LSTM-based solver.

5. Related Work

In this section, we further discuss how the proposed framework relates to and complements previous works according to five different aspects: end-to-end learning framework for inverse problems, learning schemes for gradient-based solvers, representation learning for data assimilation, data assimilation for forecasting issues and learning for computational fluid dynamics.

5.1. End-to-End Learning for Inverse Problems

A variety of end-to-end architectures have been proposed for solving inverse problems in signal and image processing, for instance for denoising, deconvolution or super-resolution issues (Chen et al., 2015; Lucas et al., 2018; McCann et al., 2017; Xie et al., 2012). In these studies, proposed schemes generally rely on a global convolutional architecture, but do not explicitly state on one hand a differentiable representation of an energy-based setting and, on the other hand, the gradient-based solver of the consisdered variational formulation. For instance, in Chen et al. (2015), the end-to-end architecture is inspired by reaction-diffusion PDEs derived from the Euler-Lagrange equation of a variational formulation, but does not explicitly embed such a variational formulation. Conversely, deep learning frameworks and embedded automatic differentiation tools have been considered to solve for inverse problems through the minimization of variational models, where the prior can be given by a pre-trained NN representation (McCann et al., 2017). An other

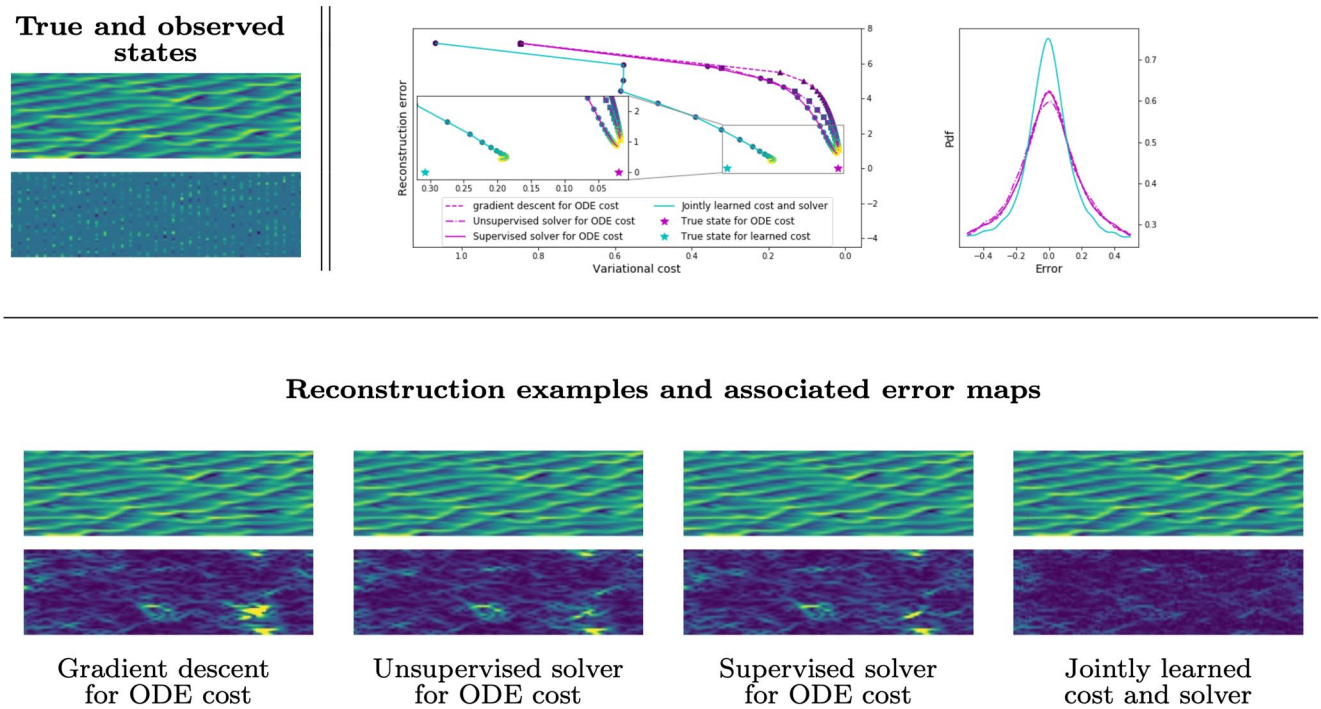

Figure 4. Reconstruction of Lorenz-96 dynamics. Upper left panel: example of true state sequence and associated partial observations; Upper center panel: solvers' energy pathways with the same setting as in Figure 2 adapted to Lorenz-96 case-study; Upper right panel: associated pdfs of the reconstruction errors. Lower panel: We depict the first half of an example of 200-time-step series of 40-dimensional Lorenz-96 states, the *x*-axis being the time axis; first row, reconstructed state sequence for the unsupervised and supervised solvers for a variational cost based on the known Lorenz-96 ODE as well as for the jointly learned variational model and solver according to a supervised setup; second row, absolute error maps of each of the four reconstructions. All states and errors are displayed with the same colormap.

strategy explored in the literature relies on the definition of two independent networks, one corresponding to the generative model and the other one to the inverse model. Raissi et al. (2019) and Raissi (2018) were among the first to explore such end-to-end architectures for the data-driven identification of ODE/PDE representations. This may be regarded as similar in spirit to auto-encoder architectures: the decoder aims to solve the inverse of the encoder but it does not explicitly depend on the representation chosen for the encoder. Here, the considered end-to-end architecture exploits automatic differentiation to explicitly relate the gradient-based inversion model to the considered differentiable variational representation. The latter makes explicit the definition of an observation model and of a dynamical prior. This is regarded as of great interest to improve the overall interpretability of the architecture. Besides, our experiments support that jointly learning the representation of the dynamics and the associated solver can lead to a very significant improvement compared with pre-training the dynamical prior and solving the resulting variational minimization. This is illustrated in this study for a supervised setting but could also apply to unsupervised settings as suggested in Fablet, Drumetz, and Rousseau (2021); Fablet, Ouala, et al. (2021). Applications to the forecasting and reconstruction of sea surface dynamics (Fablet, Drumetz, & Rousseau, 2021; Fablet, Ouala, et al., 2021) support the relevance of the proposed framework to better exploit satellite-derived ocean remote sensing data and potential applications in operational systems.

5.2. Learning Gradient-Based Solvers for Inverse Problems and Data Assimilation

A key novelty of the proposed end-to-end architecture is the gradient-based neural solver for the targeted minimization. This is similar to meta-learning and learning-to-learn strategies (Andrychowicz et al., 2016; Hospedales et al., 2020), where one can learn jointly some targeted representation and the optimizer of the training loss. The LSTM-based architecture of the considered gradient-based neural solver is similar to the one considered in learning-to-learn schemes. Such trainable solvers may speed up optimization strategies. For instance, in our experiments, we only exploit 20 gradient steps. Interestingly, we do not require as training

data for the neural solver minimization path exemplars from some predefined gradient descent algorithms. By contrast, we train the solver in an end-to-end fashion so that we optimize a reconstruction score. We may also notice that we apply here automatic differentiation to compute the gradient w.r.t. the hidden state and not the parameters of the NN representations. Besides, in the supervised learning case, the gradient computed using automatic differentiation is not the gradient of the loss to be minimized during the training process but the gradient of assimilation criterion (Equation 4). We expect the latter to be a good proxy to minimize the reconstruction error of the true states. The reported numerical experiments support this assumption and suggest extensions to other training losses which could benefit from additional data, not provided as inputs for the computation of variational cost (Equation 4). Note that in general, the 4DVar cost can present multiple local and global minima, except in the particular case where the dynamics are linear (as well as the observational operator that maps the state space to the observational space). In order to avoid falling into local minima or to select the absolute minimum a quasi-static approach can be considered (Pires et al., 1996), based on a successive small increments of the assimilation period. While this issue is not addressed in the present work, a similar quasi-static approach could be considered in the design of the learning gradient-based solver.

5.3. Representation Learning for Data Assimilation

As stated in the introduction, the data-driven identification of representations of geophysical dynamics is a very active research area. Numerous recent studies have investigated different machine learning schemes for the identification of governing equations of geophysical processes, including sparse regression techniques (Brunton et al., 2016), neural ODE and PDE schemes (Chen et al., 2018; Pannekoucke & Fablet, 2020; Raissi et al., 2019) as well as analog methods (Lguensat et al., 2017). Interestingly, advances have been proposed for the data-driven identification of such representations, when the processes of interest are not fully observed. This covers noisy and irregularly-sampled observations and partially-observed systems. Reduced-order modeling, which aims to identify lower-dimensional governing equations, also involves very similar studies (Champion et al., 2019). These previous works mainly rely on the identification of an ordinary or partial differential equation as this mathematical representation is a very generic one for modeling geophysical dynamics. The proposed end-to-end framework explicitly embeds a representation of the considered geophysical dynamics. In line with previous works cited above, we may consider neural ODE/PDE representations (Chen et al., 2018; Fablet et al., 2018; Pannekoucke & Fablet, 2020). As we rely on an energy-based setting, we can explore other types of representation. In the reported experiments, we have shown that a multi-scale energy-based representation for the dynamical prior leads to significantly better reconstruction performance than an ODE-based representation at the expense of the relatively coarse approximation of the dynamics. These experiments suggest that, for a given process, different representations might be relevant depending on the targeted applications or geophysical metrics of interest. For instance, we may expect that reconstruction or forecasting schemes for extremes may lead to different representations. Similarly, it also raises the question whether the representation should be adapted to the observation operator.

5.4. Data Assimilation for Forecasting Problems

This work can be adapted to the different classical applications of data assimilation to forecasting problems in earth science. Compared with (Equation 4), the data assimilation cost usually involves a discrepancy term on the initial condition, called the background term. This term is usually expressed as a Mahalanobis distance and involves the inverse of the background covariance matrix. This matrix plays the role of a smoothing prior on the solution and its specification is essential, yet difficult to fix or parameterize based on physical grounds. In the proposed framework, this would correspond to parameterizing and learning this discrepancy term with respect to the initial condition, while the dynamics is fixed and given. While the considered formulation (Equation 4) relates to the weak-constraint data assimilation, which accounts for modeling errors, we could also explore the proposed framework for the so-called strong constraint data assimilation, which amounts to stating operator Φ as the forecasting of the state sequence $\{x(t_i)\}_i$ over the entire time interval from the initial condition $x(t_0)$. Within a weak-constraint data assimilation with a known physical model, the proposed framework may provide new means to fit a correction error term in the dynamics from a parameterization of operator Φ as $\Phi^* + d\Phi$ with Φ^* defined as in the strong constraint case and $d\Phi$ as in the reported experiments.

5.5. Learning for Computational Fluid Dynamics

In the present study, one of the considered NN architecture for operator Φ derives from a general time evolution model interpreted in terms of iterative integration schemes (i.e., Runge-Kutta, Euler etc.). In computational flow dynamics, splitting methods are usually introduced to deal with incompressiblity and the computation of the pressure forcing term (Issa et al., 1986; Patankar, 2018). The introduction of such splitting in the NN architecture might enable us to enforce the incompressibility of the solution and thus to strengthen the physical relevance of the solutions. The synergy between NN architectures and numerical schemes is a natural way to enforce some numerical constraints imposed by the deep physical features of the dynamics at stake. For instance, splitting in terms of slow 3D internal baroclinic and fast 2D (depth averaged) barotropic motions, as it is usually done in numerical ocean models, also arises as a natural strategy to enforce the physical relevance of learning-based schemes applied to ocean dynamics.

6. Conclusion

We have introduced an end-to-end learning framework for variational assimilation problems. Assuming the observation operator is known, it combines a neural-network representation of the dynamics and a neural-network solver for the considered variational cost. We may derive the latter from the known differential operator for the considered dynamics. Here, for Lorenz-63 and Lorenz-96 dynamics, we considered NN representations which implement fourth-order Runge-Kutta integration schemes. A similar approach can be considered for PDEs, including with automatic NN generation tools from symbolic PDE expressions as proposed in (Pannekoucke & Fablet, 2020). Depending on the learning setting, we may learn jointly the two NN representations or only the NN solver assuming the NN representation of the dynamics has been calibrated previously. We provide a pytorch implementation of the proposed framework.

We report numerical experiments which support the relevance of the proposed framework. For Lorenz-63 and Lorenz-96 dynamics, we have shown that we may learn fast gradient-based solver that can reach state-of-the-art performance with only a few gradient iterations (20 in the reported experiments). Interestingly, our findings suggest that, when ground truthed data sets are available, the joint learning of the NN representation of the dynamics and of the associated NN solver may lead to a very significant improvement of the reconstruction performance. Illustrated here for Lorenz systems, future work shall further explore whether these findings generalize to other systems, especially higher-dimensional ones.

Our findings also question the design and selection of the dynamical prior in variational assimilation system. They suggest that numerical ODE-based representations optimal in terms of forecasting performance may not be optimal for reconstruction purposes. Future work shall further explore this question on the synergy between the representation of the dynamics and the gradient-based solver to retrieve the best possible state estimate. The extension of the proposed framework to stochastic representations is also of great interest and could be investigated both in the formulation of the variational costs as well as through some stochastic conditioning of the iterative solver.

Data Availability Statement

The authors provide the source code to generate all the data sets and experiments used in the manuscript, including trained models through the following link https://zenodo.org/badge/latestdoi/248528211.

Acknowledgments
This work was supported by LEFE program (LEFE MANU project IA-OAC), CNES (grant OSTST-MANATEE) and ANR Projects Melody and OceaniX. It benefited from HPC and GPU resources from Azure (Microsoft EU Ocean awards) and from GENCI-IDRIS (Grant 2020-101030).

References

Andrychowicz, M., Denil, M., Gomez, S., Hoffman, M., Pfau, D., Schaul, T., et al. (2016). Learning to learn by gradient descent by gradient descent. In *NIPS* (pp. 3981–3989).

Bannister, R. N. (2017). A review of operational methods of variational and ensemble-variational data assimilation. *Quarterly Journal of the Royal Meteorological Society*, *143*(703), 607–633. https://doi.org/10.1002/qj.2982

Blayo, E. (2008). Deep textspotter: An end-to-end trainable scene text localization and recognition framework. In *Proceedings of 2nd International workshop on industrial applications of low order model based on POD* (pp. 2204–2212).

Blum, J., Le Dimet, F.-X., & Navon, I. M. (2009). Data assimilation for geophysical fluids. In *Handbook of numerical analysis* (Vol. 14, pp. 385–441). Elsevier. https://doi.org/10.1016/s1570-8659(08)00209-3

Bocquet, M., Brajard, J., Carrassi, A., & Bertino, L. (2020). *Foundations of Data Science*, *2*(1), 55–80.

Brunton, S. L., Proctor, J. L., & Kutz, J. N. (2016). Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, *113*(15), 3932–3937. https://doi.org/10.1073/pnas.1517384113

Busta, M., Neumann, L., & Matas, J. (2017). Deep textspotter: An end-to-end trainable scene text localization and recognition framework. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2204–2212). https://doi.org/10.1109/iccv.2017.242

Champion, K., Lusch, B., Kutz, J., & Brunton, S. (2019). Data-driven discovery of coordinates and governing equations. *Proceedings of the National Academy of Sciences*, *116*(45), 22445–22451. https://doi.org/10.1073/pnas.1906995116

Chen, T., Rubanova, Y., Bettencourt, J., & Duvenaud, D. (2018). Neural ordinary differential equations. In *NIPS* (pp. 6571–6583).

Chen, Y., Yu, W., & Pock, T. (2015). On learning optimized reaction diffusion processes for effective image restoration. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 5261–5269). https://doi.org/10.1109/cvpr.2015.7299163

Cicek, Ö., Abdulkadir, A., Lienkamp, S. S., Brox, T., Ronneberger, O. (2015). 3D U-Net: Learning dense volumetric segmentation from sparse annotation. In *MICCAI* (pp. 424–432). Springer.

Dieleman, S., & Schrauwen, B. (2014). End-to-end learning for music audio. In *IEEE ICASSP* (pp. 6964–6968). https://doi.org/10.1109/ICASSP.2014.6854950

Dormand, J. R., & Prince, P. J. (1980). A family of embedded Runge-Kutta formulae. *Journal of Computational and Applied Mathematics*, *6*(1), 19–26. https://doi.org/10.1016/0771-050X(80)90013-3

Evensen, G. (2009). *Data assimilation*. Springer. Retrieved from http://link.springer.com/10.1007/978-3-642-03711-5

Fablet, R., Drumetz, L., & Rousseau, F. (2021). Joint interpolation and representation learning for irregularly-sampled satellite-derived geophysical fields. *Frontiers in Applied Mathematics and Statistics*. https://doi.org/10.3389/fams.2021.655224

Fablet, R., Ouala, M. M., Febvre, Q., Beauchamp, M., & Chapron, B. (2021). End-to-end physics-informed representation learning for satellite ocean remote sensing data: Applications to satellite altimetry and sea surface currents. In *Proceedings of XXIV ISPRS congress*. https://doi.org/10.5194/isprs-annals-v-3-2021-295-2021

Fablet, R., Ouala, S., & Herzet, C. (2018). Bilinear residual neural network for the identification and forecasting of dynamical systems. In *EUSIPCO* (pp. 1–5). Retrieved from https://hal.archives-ouvertes.fr/hal-01686766

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *IEEE CVPR* (pp. 770–778). https://doi.org/10.1109/cvpr.2016.90

Hospedales, T., Antoniou, A., Micaelli, P., & Storkey, A. (2020). *Meta-learning in neural networks: A survey*.

Issa, R., Gosman, A., & Watkins, A. (1986). The computation of compressible and incompressible recirculating flows by a non-iterative implicit scheme. *Journal of Computational Physics*, *62*(1), 66–82. https://doi.org/10.1016/0021-9991(86)90100-2

Lecun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444.

Lguensat, R., Tandeo, P., Aillot, P., & Fablet, R. (2017). The analog data assimilation. *Monthly Weather Review*. https://doi.org/10.1175/mwr-d-16-0441.1

Li, D., Yang, Y., Song, Y.-Z., & Hospedales, T. (2018). Learning to generalize: Meta-learning for domain generalization. In *Proceedings of AAAI*. Retrieved from https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16067

Lorenz, E. (1963). Deterministic nonperiodic flow. *Journal of the Atmospheric Sciences*, *20*(2), 130–141. https://doi.org/10.1175/1520-0469(1963)020<0130:dnf>2.0.co;2

Lucas, A., Iliadis, M., Molina, R., & Katsaggelos, A. (2018). Using deep neural networks for inverse problems in imaging: Beyond analytical methods. *IEEE Signal Processing Magazine*, *35*(1), 20–36. https://doi.org/10.1109/MSP.2017.2760358

McCann, M., Jin, K., & Unser, M. (2017). Convolutional neural networks for inverse problems in imaging: A review. *IEEE Signal Processing Magazine*, *34*(6), 85–95. https://doi.org/10.1109/MSP.2017.2739299

Nguyen, D., Ouala, S., Drumetz, L., & Fablet, R. (2020). Assimilation-based learning of chaotic dynamical systems from noisy and partial data. In *IEEE ICASSP* (pp. 3862–3866). https://doi.org/10.1109/icassp40776.2020.9054718

Ouala, S., Fablet, R., Herzet, C., Chapron, B., Pascual, A., Collard, F., & Gaultier, L. (2018). Neural-network-based Kalman filters for the spatio-temporal interpolation of satellite-derived sea surface temperature. *Remote Sensing*. https://doi.org/10.3390/rs10121864

Pannekoucke, O., & Fablet, R. (2020). PDE-NetGen 1.0: From symbolic PDE representations of physical processes to trainable neural network representations. *Geoscientific Model Development*, 1–14. https://doi.org/10.5194/gmd-2020-35

Patankar, S. (2018). *Numerical heat transfer and fluid flow*. Taylor & Francis.

Perez, P. (1998). Markov random fields and images. *CWI Quarterly*, 413–415. https://doi.org/10.4000/books.pupvd.8881

Pires, C., Vautard, R., & Talagrand, O. (1996). On extending the limits of variational assimilation in nonlinear chaotic systems. *Tellus A*, *48*(1), 96–121. https://doi.org/10.1034/j.1600-0870.1996.00006.x

Raissi, M. (2018). Deep hidden physics models: Deep learning of nonlinear partial differential equations. *Journal of Machine Learning Research*, *19*, 1–24.

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. https://doi.org/10.1016/j.jcp.2018.10.045

Schwartz, E., Giryes, R., & Bronstein, A. (2019). DeepISP: Toward learning an end-to-end image processing pipeline. *IEEE Transactions on Image Processing*, *28*(2), 912–923. https://doi.org/10.1109/TIP.2018.2872858

Trmolet, Y. (2007). Model error estimation in 4DVar. *Quarterly Journal of the Royal Meteorological Society*, *133*(626), 1267–1280.

Vilalta, R., & Drissi, Y. (2002). A perspective view and survey of meta-learning. *Artificial Intelligence Review*, *18*(2), 77–95. https://doi.org/10.1023/A:1019956318069

Xie, J., Xu, L., & Chen, E. (2012). Image denoising and inpainting with deep neural networks. In *NIPS* (pp. 341–349).