JOINT LEARNING OF VARIATIONAL REPRESENTATIONS AND SOLVERS FOR INVERSE PROBLEMS WITH PARTIALLY-OBSERVED DATA

A PREPRINT

R. Fablet*
IMT Atlantique, UMR CNRS Lab-STICC
Brest, FR
ronan.fablet@imt-atlantique.fr

L. Drumetz
IMT Atlantique, UMR CNRS Lab-STICC
Brest, FR
lucas.drumetz@imt-atlantique.fr

F. Rousseau
IMT Atlantique, UMR INSERM Latim
Brest, FR
francois.rousseau@imt-atlantique.fr

June 5, 2020

ABSTRACT

Designing appropriate variational regularization schemes is a crucial part of solving inverse problems, making them better-posed and guaranteeing that the solution of the associated optimization problem satisfies desirable properties. Recently, learning-based strategies have appeared to be very efficient for solving inverse problems, by learning direct inversion schemes or plug-and-play regularizers from available pairs of true states and observations. In this paper, we go a step further and design an end-to-end framework allowing to learn actual variational frameworks for inverse problems in such a supervised setting. The variational cost and the gradient-based solver are both stated as neural networks using automatic differentiation for the latter. We can jointly learn both components to minimize the data reconstruction error on the true states. This leads to a data-driven discovery of variational models. We consider an application to inverse problems with incomplete datasets (image inpainting and multivariate time series interpolation). We experimentally illustrate that this framework can lead to a significant gain in terms of reconstruction performance, including w.r.t. the direct minimization of the variational formulation derived from the known generative model.

1 Introduction

Solving an inverse problem consists in computing an acceptable solution of a model that can explain measured observations. Inverse problems involve so-called forward or generative models describing the generation process of the observed data. Inverting a defined forward model is frequently an ill-posed problem. This leads to identifiability issues, which means that naive inversion schemes based on direct least squares inversion do not yield a neither good nor unique solution. The variational framework is a popular approach to solve inverse problems by defining an energy or cost whose minimization leads to an acceptable solution. This cost can be broken down into two terms: a data fidelity term related to a specific observation model and a regularization term characterizing the space of acceptable solutions. From a Bayesian point of view, one finds a trade-off between the data fidelity term and the regularization term specified independently from a particular observation configuration. The solution sought is then the Maximum a Posteriori (MAP) estimate of the state, given the observation data. In such approaches, the prior or regularization term is specified regardless of the observation configuration.

*Corresponding author)

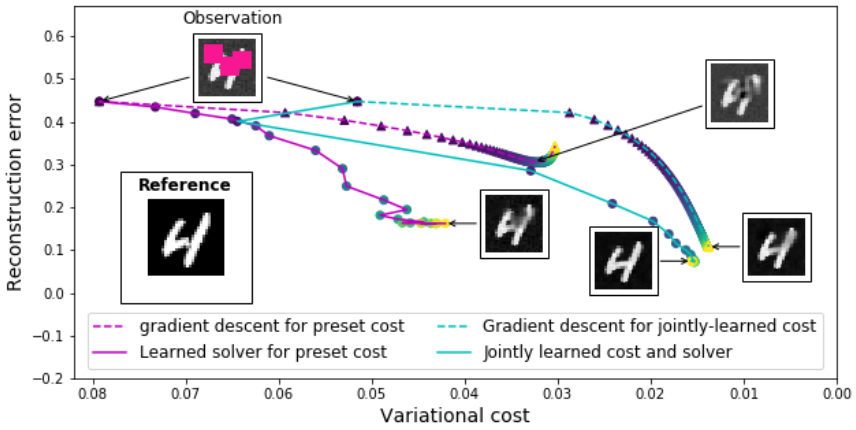


Figure 1: **Solvers' energy pathways depending on the regularization term**: when considering a predefined PCA-based regularization, the minimizer of the corresponding variational cost (magenta, dashed) may not converge to the expected hidden state especially for partially-observed data as illustrated here for an inpainting example. Using an end-to-end formulation, we may learn a gradient-based solver (magenta, solid) to improve the reconstruction performance. It may be further improved when simultaneously learning the variational cost and the associated solver, here using a CNN-parameterized cost (cyan). In such cases, we expect a gradient descent (dashed) and the trained solver (solid) to converge to similar solutions, the latter in very few steps (here, 15).

For a given inverse problem, it is then necessary to specify accurately the observation model, to define an appropriate regularization term and to implement a suitable optimization method. These three steps are the key elements in solving an inverse problem using variational approaches. An important issue is that there is no guarantees in general that the solution of the resulting optimization problem is actually the true state from which the observed data have been generated (see Fig.1). This mismatch can have several causes: discrepancy between the forward model and the actual data generation process, unsuitable regularization term or optimization algorithm stuck in a local minimum.

In this article, we address the design and resolution of inverse problems as a meta-learning problem [18]. With an emphasis on partial-observed data, our main contributions are as follows: (i) a versatile end-to-end framework for solving inverse problems by jointly learning the variational cost and the corresponding solver; (ii) experiments showing that learned iterative solvers can greatly speed up and improve the inversion performance compared to a gradient descent for predefined generative models; (iii) experiments showing that the joint learning of the variational representation and the associated solver may further improve the reconstruction performance, including when the true generative model is known. We also provide the accompanying Pytorch code for our results to be reproducible[2].

2 Background

The resolution of inverse problems is classically stated as the minimization of a variational cost [5]

$$\widehat{x} = \arg\min_{x \in \mathcal{X}} U_O(x, y) + U_R(x) \tag{1}$$

where x is the unknown state, y an observation, U_O the data fidelity term which relates the unknown state to the observation and U_R a regularization term. The latter aims at turning the ill-posed inversion of the observation into a solvable problem.

As case-study, we focus on problems where the observation only conveys a partial information about the true state: especially x and y live in some space $\mathcal{X}$ and observation term writes as $U_O(x, y) = \|x - y\|^2_\Omega$, where Ω is the domain where observation y is truely sampled[3]. Ω may for instance refer to a spatial domain in image interpolation and inpainting problems [10], to components of variable x in matrix completion [7] and data assimilation problems [6, 7]. Regarding the regularization term, a variety of approaches have been proposed, including gradient norms among which the total variation $U_R(x) = \|\nabla x\|$ [5, 2], dictionary-based terms [12, 23] or Markov Random Fields [27, 16]. Most of

[2]The code of the preprint is available at `https://github.com/CIA-Oceanix/DinAE_4DVarNN_torch`

[3]$\|x\|^2_\Omega$ refer to the squared norm of x over domain Ω, e.g. $\|x\|^2_\Omega = \int_\Omega x(p)^2 dp$ for scalar images.

these terms may be rewritten as $U_R(x) = \rho\left(x - \Phi(x)\right)$ where ρ is typically the L_1 or L_2 norm and Φ a projection-like operator. Overall, we consider a variational functional U_Φ for partially-observed data given by

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

where $\lambda_{1,2}$ are preset or learnable scalar weights. Given some parameterization for operator Φ, the minimization of functional U_Φ usually involves iterative gradient-based algorithms, for instance iterative updates given by $x^{(k+1)} = x^{(k)} - \alpha \nabla_x U_\Phi(x^{(k)}, y, \Omega)$ where $\nabla_x U_\Phi$ is the gradient operator of function U_Φ w.r.t. state x, typically derived using Euler-Lagrange equations [5]. For inverse problems with time-related processes, one may consider the adjoint method, at the core of variational data assimilation schemes in geoscience [6]. Interestingly, using automatic differentiation tools as embedded in deep learning frameworks, the computation of this gradient operator is straightforward given that operator Φ can be stated as a composition of elementary operators such as tensorial and convolution operators and non-linear componentwise activation functions.

As detailed hereafter, this property provides us the basis for designing fully-learnable end-to-end formulations, which combine an explicit variational representation (2) and an iterative gradient-based solver (4) to address the reconstruction of the unknown state x from observation y.

3 End-to-end learning framework

This section presents the proposed end-to-end framework for the joint learning of variational representations and associated solvers for inverse problems with partial observations. We first introduce in Section 3.1 the considered parameterizations for operator Φ in (2), the proposed end-to-end architecture in Section 3.2 and the learning strategy in Section 3.3.

3.1 Neural network representations for operator Φ

The design of neural network (NN) architectures for operator Φ is a key component of the proposed framework. We may distinguish physics-informed and purely data-driven representations. Regarding the latter, dictionary-based or auto-encoder representations [12] lead to operator Φ parameterized as $\Phi(x) = \phi(\psi(x))$ with ψ a mapping from the original space to a lower-dimensional space and ϕ the inverse mapping.

Here, the assumption that the solution lies in a lower-dimensional space may be fairly restrictive and typically result in smoothing out fine-scale features in signals and images. We may rather exploit a NN architecture whose inputs and outputs have the same dimension as state x, including any architecture Ψ proposed for directly solving the targeted inverse problem as $x = \Psi(y)$ as in plug-an-play approaches [25, 22, 32]. As an example, for the sake of simplicity, we consider across all case-studies a two-scale CNN architecture (subsequently denoted 2S-CNN):

$$\Phi(x) = Up\left(\Phi_1\left(Dw(x)\right)\right) + \Phi_2(x) \tag{3}$$

where Up and Dw are upsampling (ConvTranspose layer) and downsampling (AveragePooling layer) operators. $\Phi_{1,2}$ are CNNs combining convolutional layers and ReLu activations.

Regarding physics-informed representations, numerous recent studies have stressed the design of neural networks based on ordinary and partial differential equations (ODE/PDE) [28, 8]. Especially, neural ODE representations come to parameterize operator Φ as $\Phi(x)(t + \Delta) = \phi(x(t))$with Δ the integration time step and ϕ the one-step-ahead integration of the considered ODE.

3.2 End-to-end architecture

The proposed end-to-end architecture consists in embedding an iterative gradient-based solver based on the considered variational representation (2). As inputs, we consider an observation y, the associated observation domain Ω and some initialization $x^{(0)}$. Following meta-learning schemes [3], the gradient-based solver involves a residual architecture using a LSTM. More precisely, the k^{th} iterative update within the iterative solver is given by

$$\left\{\begin{array}{lcl} g^{(k+1)} & = & LSTM\left[\alpha \cdot \nabla_x U_\Phi\left(x^{(k)}, y, \Omega\right), h(k), c(k)\right] \\ x^{(k+1)} & = & x^{(k)} - \mathcal{H}\left(g^{(k+1)}\right) \end{array}\right. \tag{4}$$

where $g^{(k+1)}$ is the output of an LSTM cell using as input gradient $\nabla_x U_\Phi(x^{(k)}, y, \Omega)$ for state estimate at iteration k, $h(k), c(k)$ the internal states of the LSTM, α a normalization scalar and $\mathcal{H}$ a linear or convolutional mapping. Let

us denote by Γ this iterative update operator and $\Psi_{\Phi,\Gamma}(x^{(0)}, y, \Omega)$ the output of the solver given initialization $x^{(0)}$, observation y and observed domain Ω.

Overall, the end-to-end architecture $\Psi_{\Phi,\Gamma}$ involves a predefined number of iterative updates typically from 5 to 20 in the reported experiments. We may consider other iterative solvers, including fixed-point procedures and other recurrent cells in place of the LSTM cells [13]. Experimentally, the latter was more efficient. The total number of parameters of the proposed end-to-end architecture comprises the parameters of operator Φ, weights $\lambda_{1,2}$ and the parameters of gradient-based update operator Γ.

3.3 Joint learning scheme

Given pairs of observation data (y_n, Ω_n) and hidden states x_n, we state the joint learning of operator Φ and solver operator Γ as the minimization of a reconstruction cost:

$$\arg\min_{\Phi,\Gamma} \sum_n \mathcal{L}\left(x_n, \tilde{x}_n\right) \text{ s.t. } \tilde{x}_n = \Psi_{\Phi,\Gamma}(x_n^{(0)}, y_n, \Omega_n) \tag{5}$$

where $\mathcal{L}(x, \tilde{x}) = \nu_1\|x - \tilde{x}\|^2 + \nu_2\|x - \Phi(x)\|^2 + \nu_3\|\tilde{x} - \Phi(\tilde{x})\|^2, \forall x, \forall y \in \mathcal{X}$ combines the reconstruction error of the true state with the projection errors of both the reconstructed state and the true state through weights $\nu_{1,2,3}$. Given the end-to-end architecture $\Psi_{\Phi,\Gamma}$, we can directly apply stochastic optimizers such as Adam to solve this minimization. Obviously, we may also consider a sequential learning scheme, where operator Φ is first set *a priori* or learned according to reconstruction loss $\sum_n \|x_n - \Phi(x_n)\|^2$, and the above minimization is carried out w.r.t. Γ only, Φ being fixed.

All reported experiments have been run using Pytorch and the Adam optimizer for the learning stage. Regarding the gradient-based solver, we gradually increase the number of gradient-based iterations during the learning process, typically from 5 to 20. Weights $\nu_{1,2,3}$ are set to 1., 0.05 and 0.05. When jointly learning operators Φ and Γ, we use a halved learning rate for the parameters of operator Φ.

4 Results

We run numerical experiments to illustrate and evaluate the proposed framework. We consider two different case-studies: image inpainting with MNIST data and the reconstruction of downsampled signals governed by ODEs. The latter provides us with means to investigate learnable variational formulations and solvers when the true model which governs the hidden states is known.

Model Φ	Joint learning	Solver	R-Score	I-score	P-score
DICT	No	OMP	0.25	0.90	0.21
	No	Lasso	0.20	0.71	0.21
PCA	No	FSGD	0.39	1.09	**0.12**
	No	LSTM-S	0.15	0.54	**0.12**
AE	Yes	FP(1)	0.4	1.51	**0.12**
	No	LSTM-S	0.195	0.71	**0.12**
	Yes	LSTM-S	0.210	0.76	0.95
2S-CNN	Yes	FP(1)	0.16	0.56	0.03
	Yes	LSTM-S	**0.09**	**0.33**	0.02

Table 1: **MNIST experiment:** The training and test datasets comprise MNIST images corrupted by a Gaussian additive noise and three randomly-sampled 9x9 holes. We consider three parameterizations for operator Φ in the variational formulation defined by (2), a 50-dimensional PCA decomposition (PCA), a 20-dimensional dense auto-encoder (AE) and a bilinear 2S-CNN representation. We also compare a fixed-step gradient descent for variational energy (2), referred to as FSGD, a one-step fixed-point iteration (FP(1)) and a LSTM-based solver (LSTM-S) based on the automatic differentiation of energy (2). We may jointly train operator Φ and the LSTM solver or solely train the latter given a predefined operator Φ. We also include two sparse coding schemes (DICT) adapted from [23]. As evaluation scores, we compute the reconstruction error for the entire image domain (R-score), the reconstruction error for missing data areas only (I-score), the projection error $x - \Phi(x)$ for gap-free and noise-free images (P-score). We let the reader refer to the main text for additional information.

4.1 MNIST data

MNIST data provide a relatively simple image dataset to evaluate the proposed framework with different types of regularization terms in (2). Especially, the MNIST dataset seems well-suited to dictionary and auto-encoder priors [12, 23]. We simulate observed images with 3 randomly sampled 9×9 missing data areas and an additive Gaussian noise of variance $0.1 * \sigma^2$, with σ^2 the pixel-wise variance of the training data. We consider three types of parameterizations for operator Φ:

- **a linear PCA decomposition**: a 50-dimensional PCA operator which amounts to $\approx 90\%$ of the variance of the data. Fitted to the training dataset, it involves 80,000 parameters.
- **a dense auto-encoder (AE)**: a dense AE operator with a 20-dimensional latent representation. The encoder and decoder both involve 3 dense layers with ReLu activations. This AE also amounts to $\approx 90\%$ of the variance of the data. It comprises ≈500,000 parameters.
- **2S-CNN model**: a CNN architecture given by (3) with a bilinear parameterization to account for local non-linearities [14]. It comprises ≈55,000 parameters. Importantly, it combines both global and local features through the considered two-scale representation.

We let the reader refer to the code provided as supplementary material for more details. Given these parameterizations, we investigate different learning strategies and/or optimisation schemes:

- **A fixed-step gradient descent**: assuming operator Φ has been pre-trained, we implement a simple fixed-step gradient descent for variational cost (2). We empirically tune the gradient step and the weighing factors $\lambda_{1,2}$ according to a log-10 scale grid search. We report the best reconstruction performance along the implemented gradient descent pathway. We refer to this optimization scheme as FSGD;
- **A fixed-point procedure**: a classic approach to solve for inverse problems using deep learning architectures [9, 19, 32] is to train a solver using observation y as inputs. Such approaches may be regarded as the application of a one-step fixed-point solver for criterion (2) [13].The resulting interpolator is $\Phi(y)$. We refer to this solver as FP(1).
- **a learnable gradient-based solver**: the LSTM solver given by (4) involves a convolutional LSTM cell whose hidden states involve 5 channels. We refer to this solver as LSTM-S.

For benchmarking purposes, we include sparse coding approaches using OMP (Orthogonal Matching Pursuit) and Lasso solutions [23]. We adapt these schemes for interpolation issues through iterative coding-decoding steps, the observed data being kept after each decoding step, except in the final iteration[4]. Our evaluation relies on three normalized metrics for the test dataset, using σ^2 as normalization factor: the normalized mean square error (NMSE) of the reconstruction over the entire image domain (R-Score), the NMSE of the reconstruction over the unobserved image domain (I-Score) and the NMSE of the projection error $x - \Phi(x)$ for true states x (P-Score).

We report in Tab.1 a synthesis of these experiments. We may first notice that the learnable solver significantly improves the reconstruction error compared with the direct minimization of (2) given a pretrained model (e.g., 0.54 vs. 1.09 (resp. 0.71 vs. 1.51) for the PCA (resp. AE) version of operator Φ)[5]. Despite AE and PCA architectures having very similar representation capabilities for the true states (P-scores of ≈ 0.12), the reconstruction performance is much better with a PCA representation, which may relate to the simpler and convex nature of the associated variational cost (2). As illustrated in Fig.1, the learnable solver identifies a much better descent pathway for the reconstruction error from the differentiation of variational cost (2) than the direct minimization of this cost.

The second key result is the significant gain issued from the joint learning of operator Φ and of the solver (0.33 vs. 0.54 for the best I-scores of 2S-CNN and PCA representations). Note that the 2S-CNN-based operator Φ used as a direct solution for the inversion (FP(1) solver in Tab.1) is also significantly outperformed. Intriguingly, the better reconstruction performance issued from the 2S-CNN representation involves a worse P-score. It suggests that a coarser approximation of true states by operator Φ may be better suited to reconstruction issues than the one best describing the true states. The visual inspection of examples in Fig.2 is in line with these results. The 2S-CNN parameterization in the joint learning setting clearly outperforms the other approaches. The FSGD minimization of the learned 2S-CNN-based variational cost leads to visually relevant reconstructions, though the contrast of interpolated areas is often too low. This, with the consistent descent pathways of the FSGD and learnable solvers for the 2S-CNN case in Fig.1, supports the consistency of the learned variational representation. Fig.1 also stresses the fast convergence of the learnable solvers with 10 to 20 iterations, compared with hundreds or thousands for the FSGD scheme.

[4]These sparse coding schemes are run with MiniBatchDictionaryLearning function in scikitlearn [26].

[5]We do not carry a similar experiment with the 2S-CNN architecture as by construction the associated pre-trained version of operator Φ could lead upon convergence to a meaningless prior $\Phi(x) = x$.

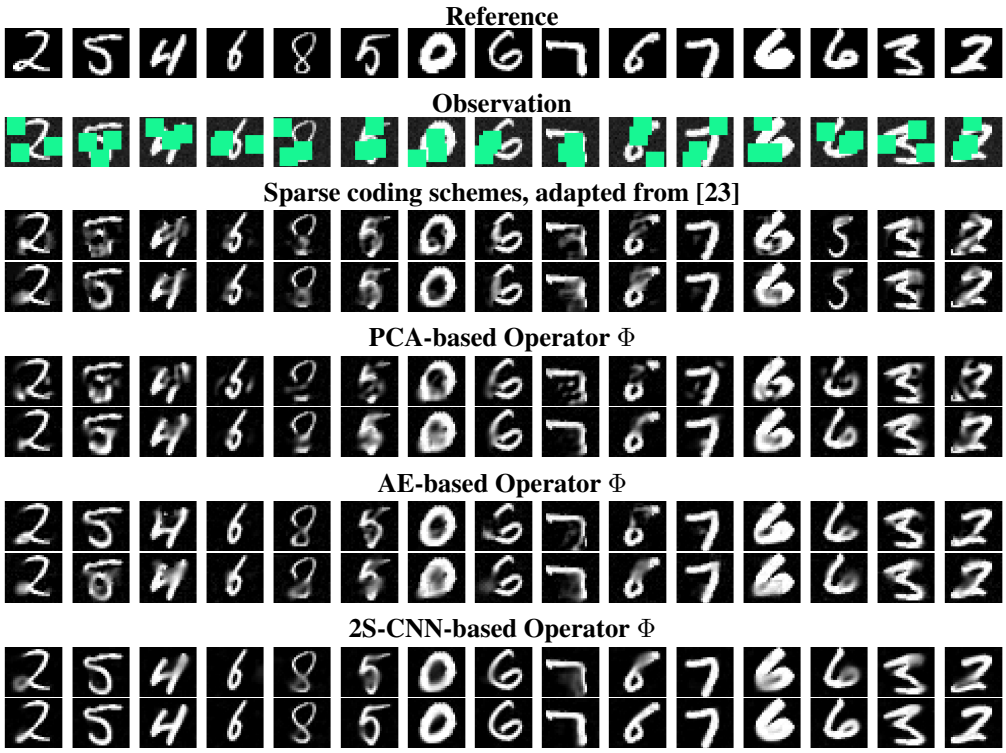


Figure 2: **Reconstruction examples for MNIST experiment**. We randomly sample 15 examples for MNIST test dataset: first row, reference images; second row, observed images; third and fourth rows, sparse decomposition methods [23] using respectively OMP and Lasso solutions. From the fifth row to the tenth row, we report results using the proposed framework using PCA-based, AE-based and 2S-CNN-based parameterizations for operator Φ. For each example, we first report the output of a gradient descent of the considered variational cost (e.g., the fifth row for the PCA-based setting) and of the learnt LSTM solver (e.g., the sixth row for the PCA-based setting). We let the reader refer to the main text for the details on the considered experimental setting and parameterizations.

4.2 Multivariate signals governed by ODEs

The second case-study addresses multivariate signals governed by ODEs so that we are provided with a theoretical ground-truth for the true representation of the data: namely, Lorenz-63 and Lorenz-96 dynamics, which are widely used for benchmarking experiments in geoscience and data assimilation. Both systems involve bilinear ODEs which lead to chaotic patterns under the considered parametrizations. We detail our experimental setting, which involves noisy and undersampled observations, in the Supplementary Material.

We consider two architectures for operator Φ. A Neural ODE representation is first fully-informed by the true ODE using a bilinear ResNet architecture[6] [14], referred to as L63-RK4 and L96-RK4. We also investigate CNN-based representations (3), where operators $\Phi_{1,2}$ involve bilinear convolutional blocks with 30 channels. Referred to as L63-2S-CNN and L96-2S-CNN, they involve respectively 15,000 and 52,000 parameters.

We consider the same optimization schemes as for MNIST case-study. Regarding evaluation metrics, we focus on the R-score and the P-score[7]. Tab.2 further supports the relevance of the proposed framework. Similarly to MNIST case-study, the learnable LSTM solver clearly outperforms the FSGD minimization. The joint learning of the variational representation and of the solver further improves the reconstruction performance. Besides, we also retrieve that the best reconstruction schemes involve a significantly worse representation performance for the true states.

[6] It relies on a 4^{th}-order Runge-Kutta integration which provides a good approximation of the integration scheme considered for data generation.

[7] Given the noise level and regular sampling, R-score and I-score provide relatively similar information.

Model Φ	Joint learning	Solver	R-Score	P-score
L63-RK4	No	FSGD	2.71e-3	**1.41e-7**
	No	LSTM-S	1.52e-3	**1.41e-7**
L63-2S-CNN	Yes	LSTM-S	**1.00e-3**	2.78e-4
L96-RK4	No	FSGD	5.65e-2	**6.70e-7**
	No	LSTM-S	4.46e-2	**6.70e-7**
L96-2S-CNN	Yes	LSTM-S	**1.94e-2**	3.70e-3

Table 2: **Experiment with multivariate signals governed by ODEs:** we report for Lorenz-63 and Lorenz-96 dynamics (7-8) reconstruction experiments from noisy and under-sampled observations. We let the reader refer to the main text for the details on the experimental setting and the parameterizations considered for operator Φ in (2). The evaluation procedure is similar to Tab.1.

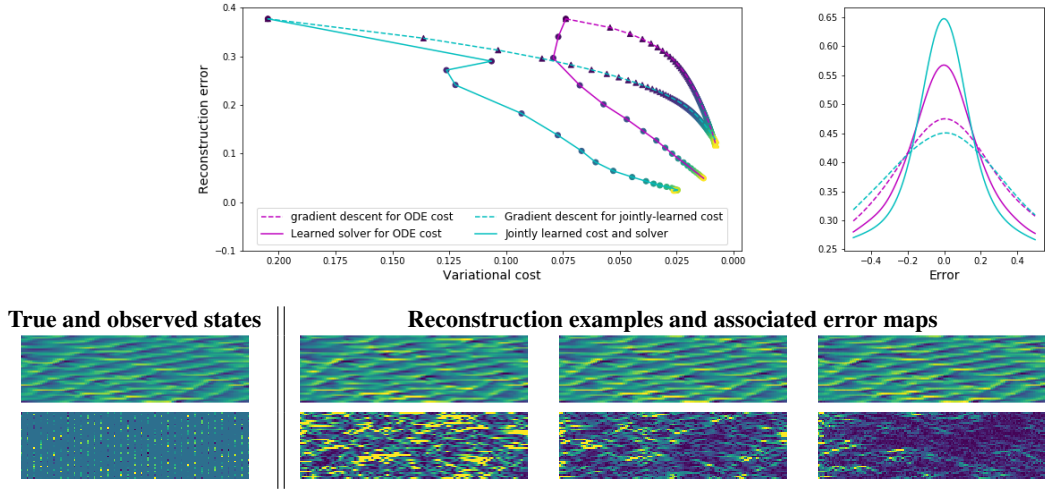


Figure 3: **Reconstruction of Lorenz-96 dynamics.** Upper left panel: solvers energy pathways using a predefined ODE-based cost or a learnable variational formulation. Upper right panel: associated pdfs of the reconstruction errors. Lower panel: We depict the first half of an example of 200-time-step series of 40-dimensional Lorenz-96 states, the x-axis being the time axis; first row, reference states, FSGD-based reconstruction for variational cost (2) with the true ODE model, reconstruction using the learned LSTM solver for this ODE-based setting, reconstruction when jointly learning the variational representation and the LSTM solver; second row, observation with missing data, absolute error maps of each of the 3 reconstructions. All states and errors are displayed with the same colormap.

5 Related work

Direct learning of inverse models: Supervised learning has been increasingly used to address inverse problems by learning the relationship between observations and true states [24, 21]. Interestingly, learnable data-driven solvers have been derived from standard model-driven algorithms, e.g. reaction-diffusion PDEs [9], ADMM algorithm [33]. Such approaches do not however learn some underlying variational representation beyond the learned solver.

Plug-and-play and learning-based regularizers: In order to make learning approaches more versatile, and not specialized to a particular instance of a problem, so-called plug-and-play methods have been designed [31], including for instance denoising neural networks [25] and adversarial regularizers [22]. These regularizers trained offline can be incorporated to proximal optimization algorithms. This method can also lead to convergence guarantees of the resulting variational framework, provided the learned operator is sufficiently regular [30].

Bi-level optimization for inverse problems: Recently, some learning schemes for inverse problems took a meta-learning [18] orientation, in the form of bi-level optimization problems. A given variational cost function (or a pre-trained regularizer such as with plug-and-play algorithms) forms an inner optimization problem, which is not solved directly as in classical variational schemes, but involves a learning-based solver, so that the reconstruction error is

minimized in an outer loop [20, 1]. Note that a recent monograph covers most of the aspects discussed in this section so far [4].

Optimizer learning: Learning solvers to minimize cost functions, in a so-called "learning-to-learn" fashion [3], has attracted a lot of attention recently. The idea is to unroll an optimization algorithm (e.g., a descent direction algorithm), where the descent direction is parameterized by a RNN. The parameters of the RNN are learnt to to minimize the objective function. LSTMs are a popular choice since they allow to keep a memory of the past gradient information, as in momentum-based algorithms such as Adam. They also lead to sample-efficient algorithms and faster convergence than classical optimization algorithms [3], as evidenced in our experiments. Optimizer learning is related to the bi-level optimization framework [15], as the formal bi-level optimization setting can be relaxed by replacing the inner minimization through an unrolled optimization algorithm. Our learning strategy (5) can also be regarded as a relaxed version of a bi-level optimization problem

$$\arg\min_{\Phi} \sum_{n} \mathcal{L}(x_n, \tilde{x}_n) \text{ s.t. } \tilde{x}_n = \arg\min_{x} U_{\Phi}\left(x, y_n, \Omega_n\right) \tag{6}$$

where the inner minimization is replaced with a fixed number of iterations of a gradient-based solver. Here, the inner cost function also has trainable parameters. This relaxation, combined with automatic differentiation, allows our approach to jointly learn the parameters of the variational cost and of the associated solver, instead of resorting to a sequential minimization as in classical bi-level frameworks. If we fix the inner cost function,we can incorporate any direct learning-based inversion scheme or plug-and-play approach to be fined tuned via meta-learning. The potential interest is two-fold: a lower complexity of the trainable operators and a greater interpretability of the learnt variational representations compared with a sole solver.

6 Conclusion

To the best of our knowledge, the problems of learning a variational cost (or more restrictively, a regularizer) from data for inverse problems, and learning a solver for a given variational cost, have always been treated sequentially so far. Our approach allows the joint data-driven discovery of a variational formulation of the inverse problem and the associated solver. Our experiments show that this is a better strategy than only learning the solver using a pre-defined or pre-trained variational cost, or even than directly inverting with the true generative model.

This work opens new research avenues regarding the definition and resolution of variational formulations. A variety of energy-based formulations may be investigated beyond regularization terms $\|x - \Phi(x)\|^2$ as considered here, including physics-informed and theory-guided formulations as well as probabilistic energy-based representations.

Acknowledgements

This work was supported by CNES (grant OSTST-MANATEE), Microsoft (AI EU Ocean awards) and ANR Projects Melody and OceaniX, and exploited HPC resources from GENCI-IDRIS (Grant 2020-101030).

References

[1] J. Adler and O. Öktem. Solving ill-posed inverse problems using iterative deep neural networks. *Inv. Pr.*, 33(12):124007, 2017.

[2] L. Alvarez, F. Guichard, P.-L. Lions, and J.-M. Morel. Axioms and fundamental equations of image processing. *Arch. Rat. Mech. An.*, 123(3):199–257, 1993.

[3] M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. De Freitas. Learning to learn by gradient descent by gradient descent. In *NeurIPS*, 3981–3989, 2016.

[4] S. Arridge, P. Maass, O. Öktem, and C.-B. Schönlieb. Solving inverse problems using data-driven models. *Act. Num.*, 28:1–174, 2019.

[5] G. Aubert and P. Kornprobst. *Mathematical problems in image processing: partial differential equations and the calculus of variations*, 147. Springer, 2006.

[6] J. Blum, F.-X. Le Dimet, and M.I. Navon. Data assimilation for geophysical fluids. In *Handbook Num. An.*, 14:385–441, 2009.

[7] E.J. Candes and Y. Plan. Matrix Completion With Noise. In *Proc. IEEE*, 98(6):925–936, 2010.

[8] T.Q. Chen, Y. Rubanova, J. Bettencourt, and D.K. Duvenaud. Neural Ordinary Differential Equations. In *NeurIPS*, 6571–6583, 2018.

[9] Y. Chen, W. Yu, and T. Pock. On Learning Optimized Reaction Diffusion Processes for Effective Image Restoration. In *CVPR* 5261–5269, 2015.

[10] Q. Cheng, H. Shen, L. Zhang, and P. Li. Inpainting for Remotely Sensed Images With a Multichannel Nonlocal Total Variation Model. *IEEE TGRS*, 52(1):175–187, 2014.

[11] J. R. Dormand and P. J. Prince. A family of embedded Runge-Kutta formulae. *J. Comp. App. Math.*, 6(1):19–26, 1980.

[12] M. Elad and M. Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *IEEE TIP*, 15(12):3736–3745, 2006.

[13] R. Fablet, L. Drumetz, and F. Rousseau. End-to-end learning of optimal interpolators for geophysical dynamics. In *Climate Informatics*, 2019.

[14] R. Fablet, S. Ouala, and C. Herzet. Bilinear residual Neural Network for the identification and forecasting of dynamical systems. In *EUSIPCO*, 2018.

[15] L. Franceschi, P. Frasconi, S. Salzo, R. Grazzi, and M. Pontil. Bilevel programming for hyperparameter optimization and meta-learning. *arXiv preprint arXiv:1806.04910*, 2018.

[16] D. Geman. Random fields and inverse problems in imaging. *LNM*, 1427:113–193, 1990.

[17] G. Holler, K. Kunisch, and R. C Barnard. A bilevel approach for parameter learning in inverse problems. *Inv. Pr.*, 34(11):115012, 2018.

[18] T. Hospedales, A. Antoniou, P. Micaelli, and A. Storkey. Meta-learning in neural networks: A survey. *arXiv preprint arXiv:2004.05439*, 2020.

[19] G. Liu, F.A. Reda, K.J. Shih, T.C. Wang, A. Tao, and B. Catanzaro. Image Inpainting for Irregular Holes Using Partial Convolutions. In *ECCV*, 85–100, 2018.

[20] R. Liu, L. Ma, X. Yuan, S. Zeng, and J. Zhang. Bilevel Integrative Optimization for Ill-posed Inverse Problems. *arXiv preprint arXiv:1907.03083*, 2019.

[21] A. Lucas, M. Iliadis, R. Molina, and A. K Katsaggelos. Using deep neural networks for inverse problems in imaging: beyond analytical methods. *IEEE SPM*, 35(1):20–36, 2018.

[22] S. Lunz, O. Öktem, and C.-B. Schönlieb. Adversarial regularizers in inverse problems. In *NeurIPS*, pages 8507–8516, 2018.

[23] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online dictionary learning for sparse coding. In *ICML*, 689–696, 2009.

[24] M.T. McCann, K.H. Jin and M. Unser. Convolutional neural networks for inverse problems in imaging: A review. In *IEEE SPM*, 34(6):85–95, 2017.

[25] T. Meinhardt, M. Moller, C. Hazirbas, and D. Cremers. Learning proximal operators: Using denoising networks for regularizing inverse imaging problems. In *CVPR*, 1781–1790, 2017.

[26] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, and V. Dubourg. Scikit-learn: Machine learning in Python. *JMLR*, 12:2825–2830, 2011.

[27] Patrick Perez. *Markov random fields and images*. CWI Quart., 1998.

[28] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *JCP*, 378:686–707, 2019.

[29] J.H.R. Chang, C.-L. Li, B. Poczos, B.V.K. Vijaya Kumar, and A.C. Sankaranarayanan. One network to solve them all–solving linear inverse problems using deep projection models. In *ICCV*, 5888–5897, 2017.

[30] E.K. Ryu, J. Liu, S. Wang, X. Chen, Z. Wang, and W. Yin. Plug-and-play methods provably converge with properly trained denoisers. *arXiv preprint arXiv:1905.05406*, 2019.

[31] S.V. Venkatakrishnan, C.A. Bouman, and B. Wohlberg. Plug-and-play priors for model based reconstruction. In *IEEE GlobalSIP*, 945–948, 2013.

[32] J. Xie, L. Xu, and E. Chen. Image Denoising and Inpainting with Deep Neural Networks. In *NIPS*, 341–349, 2012.

[33] J. Xie, L. Xu, and E. Chen. Deep ADMM-Net for Compressive Sensing MRI. In *NIPS*, 10–18, 2016.

A Supplementary Material

A.1 Lorenz-63 case-study

Lorenz-63 dynamics involve a three-dimensional state governed by the following ODE:

$$\begin{cases} \frac{dX_{t,1}}{dt} &= \sigma\left(X_{t,2} - X_{t,2}\right) \\ \frac{dX_{t,2}}{dt} &= \rho X_{t,1} - X_{t,2} - X_{t,1}X_{t,3} \\ \frac{dX_{t,3}}{dt} &= X_{t,1}X_{t,2} - \beta X_{t,3} \end{cases} \tag{7}$$

We consider the standard parameterization, $\sigma = 10$, $\rho = 28$ and $\beta = 8/3$, which leads to a strange attractor. We may refer the reader to [**?**] for additional illustrations.

In our experiments, we simulate time series with a 0.01 time step using a RK45 integration scheme [11]. More precisely, We first simulate a 20000-time-step sequence from an initial condition within the attractor. We use time steps 0 to 150000 to randomly sample 10000 200-time-step sequences for the training dataset, and time steps 15000 to 2000 to randomly sample 2000 200-time-step sequences for the test dataset. Observation data are generated similarly to [**?**] with a sampling every 8 time steps and a Gaussian additive noise with a variance of 2.

A.2 Lorenz-96 case-study

Lorenz-96 dynamics involve a 40-dimensional state with a periodic boundary condition governed by the following ODE

$$\frac{dx_{t,i}}{dt} = \left(x_{t,i+1} - x_{t,i-2}\right) x_{t,i-1} - x_{t,i} + F \tag{8}$$

with i the index from 1 to 40 and F a scalar parameter set to 8. These dynamics involve chaotic wave-like patterns as illustrated in Fig.3

In our experiments, we proceed similarly to Lorenz-63 case-study. We first simulate a 10000-time-step sequence from an initial condition within the attractor.

we simulate time series with a 0.05 time step using a RK45 integration scheme [11]. The training and test datasets involve time series with 200 time steps. We use time steps 0 to 7500 to randomly sample 2000 200-time-step sequences for the training dataset, and time steps 7500 to 10000 to randomly sample 256 200-time-step sequences for the test dataset. Observation data are generated similarly to [**?**] with a sampling every 4 time steps, for only 20 randomly selected components over the 40 components of the state, and a Gaussian additive noise with a variance of 2.