# Deep prior in variational assimilation to estimate ocean circulation without explicit regularization

Arthur Filoche, Dominique Béréziat, Anastase Alexandre Charantonis

*Environmental Data Science* (2020), 4: 1–10
doi:10.1017/eds.2020.xx

CAMBRIDGE UNIVERSITY PRESS

RESEARCH ARTICLE

# Deep prior in variational assimilation to estimate ocean circulation without explicit regularization

Arthur Filoche[1], Dominique Béréziat[1] and Anastase Charantonis[2]

[1]Sorbonne Université, CNRS, LIP6, Paris, 75005, France
[2]Laboratoire d'Océanographie et du Climat (LOCEAN), Paris, 75005 France

Received xx xxx xxxx

**Keywords:** variational data assimilation, regularization, deep prior, ocean circulation, twin experiment

## Abstract

Many applications in geosciences require solving inverse problems to estimate the state of a physical system. Data assimilation provides a strong framework to do so when the system is partially observed and its underlying dynamics is known to some extent. In the variational flavor, it can be seen as an optimal control problem where initial conditions are the control parameters. Such problems being often ill-posed, regularization may be needed using explicit prior knowledge to enforce satisfying solution. In this work we propose to use a deep prior, a neural architecture that generates potential solution and acts as implicit regularization. The architecture is trained in an fully-unsupervised manner using the variational data assimilation cost so that gradients are backpropagated through the dynamical model and then through the neural network. To demonstrate its use, we set a twin experiment using a shallow-water toy model, where we test various variational assimilation algorithms on a ocean-like circulation estimation.

## 1. Introduction

Physics-driven numerical weather prediction requires to estimate initial conditions before making a forecast. To do so, one should exploit all the knowledge at disposal which can be observations, a dynamical model or errors statistics. It formalizes as an inverse problem and data assimilation offers a large panel of methods to solve it [2]. A subset of these methods are variational so that the system state is estimated via the minimization of a cost function as in the 4D-Var algorithm [10]. Many similarities with machine learning have been highlighted [1] as both can be used to perform Bayesian inversion by gradient descent.

Even though variational data assimilation has a long standing experience in model-constrained optimization, deep learning techniques have revolutionized ill-posed inverse problems solving [14]. And methods combining neural architectures and differentiable physical models already exists [6, 17, 12]. Most of the time, a large database is leveraged to learn a regularization adapted to the task.

Fitting observations respecting the dynamical model can be seen as a form of regularization but an additional regularizer may be required to promote acceptable solution [9]. Recently a very original idea called deep prior [18] has been developed. A neural architecture is used to generate the solution of an inverse problem and acts like an implicit regularization. Astonishingly the whole architecture is trained in an unsupervised manner on one example and provides results comparable to supervised methods.

In this work, we propose a hybrid methodology bridging deep prior and variational data assimilation and we test it in a twin experiment. The algorithm is evaluated on an ocean-like motion estimation task requiring regularization, then compared to adapted data assimilation algorithms [3, 4]. All algorithms are implemented using tools from the deep learning community. The code is available in Github[1].

## 2. Methodology

### *2.1. Data assimilation Framework*

A dynamical system is considered where a state $\mathbf{X}$ evolves over time following perfectly-known dynamics $\mathbb{M}$, see Eq. (1). Partial and noisy observations $\mathbf{Y}$ are available through an observation operator $\mathbb{H}$, Eq. (2). A background $\mathbf{X}_b$ gives prior information about the initial system state, Eq. (3).

$$\text{Evolution:} \quad \mathbf{X}_{t+1} = \mathbb{M}_t(\mathbf{X}_t) \tag{1}$$

$$\text{Observation:} \quad \mathbf{Y}_t = \mathbb{H}_t(\mathbf{X}_t) + \varepsilon_{R_t} \tag{2}$$

$$\text{Background:} \quad \mathbf{X}_0 = \mathbf{X}_B + \varepsilon_B \tag{3}$$

Additive noise $\varepsilon_B$ and $\varepsilon_R$ represent uncertainties about the observations and the background, respectively. These noises are quantified by their assumed known covariance matrices $\mathbf{B}$ and $\mathbf{R}$, respectively. The dynamics is here considered perfect, but the framework could easily be extended to an imperfect dynamics. For any given matrix $\mathbf{A}$, we note $\|x - y\|_A^2 = \langle (x-y) | \mathbf{A}^{-1}(x-y) \rangle$ the associated Mahalanobis distance.

[1] https://github.com/ArFiloche/Deepprior4DVar_CI22

### *2.2. Variational assimilation*

The objective of data assimilation is to provide an estimation of the system state $\mathbf{X}$ by optimally combining available data $\mathbf{X}_B$, $\mathbf{Y}$ and the dynamical model $\mathbb{M}$. In the variational formalism [10], this is done via the minimization of a cost function which is the sum of background errors and observational errors, $\mathcal{J}_{\text{4DVar}} = \frac{1}{2}\|\varepsilon_b\|^2_{\mathbf{B}} + \frac{1}{2}\sum_{t=0}^{T}\|\varepsilon_{R_t}\|^2_{\mathbf{R}_t}$. The optimization problem is model-constrained as described in Eq. (4). What motivates this cost function is that minimizing it leads to the maximum *a posteriori* estimation of the state under independent Gaussian errors, linear observation operator and linear model hypothesis. The corresponding optimization algorithm is named 4D-Var.

$$\begin{aligned} \underset{\mathbf{X}_0}{\arg\min} \quad & \mathcal{J}_{\text{4DVar}}(\mathbf{X}_0) = \frac{1}{2}\|\mathbf{X}_0 - \mathbf{X}_b\|^2_{\mathbf{B}} + \frac{1}{2}\sum_{t=0}^{T}\|\mathbf{Y}_t - \mathbb{H}_t(\mathbf{X}_t)\|^2_{\mathbf{R}_t} \\ \text{s.t.} \quad & \mathbf{X}_{t+1} = \mathbb{M}_t(\mathbf{X}_t) \end{aligned} \tag{4}$$

The link between variational assimilation and Tikhonov regularization is well described in [9]. For example, choosing a particular matrix $\mathbf{B}$ will promote a particular set of solutions. Therefore, making alike choices can be seen as a handcrafted regularization to take advantage of expert prior knowledge.

### *2.3. Deep prior 4D-Var*

The idea behind deep prior is that using a well-suited neural architecture to generate a solution of the variational problem can act as a handcrafted regularization. This means that the control parameters are shifted from the system state space to the neural network parameters space. From a practical standpoint, a latent variable $z$ is fixed and a generator network $g_{\boldsymbol{\theta}}$ outputs the solution from it such that $g_{\boldsymbol{\theta}}(z) = \mathbf{X}_0$.

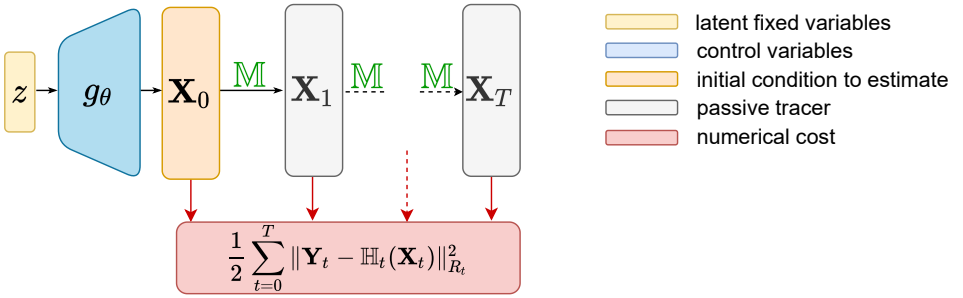


***Figure 1.*** *Schematic view of the forward integration in Deep Prior 4D-Var.*

#### *2.3.1. Cost function*

The generator is then trained with the variational assimilation cost $\mathcal{J}(\boldsymbol{\theta})$. To emphasize the regularizing effect of the deep prior method, we choose to fix $\mathbf{B} = 0$ so that no background information is used. It means that $\mathcal{J}(\boldsymbol{\theta}) = \frac{1}{2}\sum_{t=0}^{T}\|\varepsilon_{R_t}\|^2_{R_t}$ and by denoting multiple integration between two times $\mathbb{M}_{t_1\to t_2}$, the cost can be developed as in Eq. (5).

$$\mathcal{J}(\boldsymbol{\theta}) = \frac{1}{2}\sum_{t=0}^{T}\|\mathbf{Y}_t - \mathbb{H}_t(\mathbb{M}_{0\to t}(g_{\boldsymbol{\theta}}(z)))\|^2_{R_t} \tag{5}$$

It is important to note that this approach is unsupervised, the architecture being trained from scratch on one assimilation window with no pre-training. All the prior information should be contained in the architecture choice.

#### *2.3.2. Gradient*

The gradients of this cost function can be determined analytically. First, the chain rule gives Eq. (6). Then using the adjoint state method we can develop $\nabla_{\mathbf{X}_0}\mathcal{J}(\mathbf{X}_0)$ as in Eq. (7), a detailed proof can be found in [2]. In the differentiable programming paradigm, such analytical expression is not needed to obtain gradients, adjoint modeling is implicitly performed as gradients are backpropagated automatically.

$$\nabla_{\boldsymbol{\theta}}\mathcal{J}(\boldsymbol{\theta}) = \nabla_{\mathbf{X}_0}\mathcal{J}(\mathbf{X}_0)\nabla_{\boldsymbol{\theta}}\mathbf{X}_0 = \nabla_{\mathbf{X}_0}\mathcal{J}(\mathbf{X}_0)\nabla_{\boldsymbol{\theta}}g_{\boldsymbol{\theta}}(z) \tag{6}$$

$$\nabla_{\mathbf{X}_0}\mathcal{J}(\mathbf{X}_0) = \sum_{t=0}^{T}\left[\frac{\partial(\mathbb{H}_t\mathbb{M}_{0\to t})}{\partial\mathbf{X}_0}\right]^{\top}\mathbf{R}_t{}^{-1}\varepsilon_{R_t} \tag{7}$$

#### *2.3.3. Algorithm*

The algorithm seeks to numerically optimize the cost function and simply consists of alternating forward and backward integration to update control parameters by gradient descent (see Algorithm 1). A schematic view of the forward integration can be found in Figure 1.

**Algorithm 1 – Deep prior 4D-Var**

Initialize fixed latent variables $z$
Initialize control variables $\boldsymbol{\theta}$
**while** stop criterion **do**
  **forward**: integrate $\mathbb{M}_{0\to T}(g_{\boldsymbol{\theta}}(z))$ and compute $\mathcal{J}$
  **backward**: automatic differentiation returns $\nabla_{\boldsymbol{\theta}}\mathcal{J}$
  **update**: $\boldsymbol{\theta}$ = optimizer($\boldsymbol{\theta}$ , $\mathcal{J}$ , $\nabla_{\boldsymbol{\theta}}\mathcal{J}$)
**end while**
**return** $\boldsymbol{\theta}, \mathbf{X}_0$

## 3. Case study

### *3.1. Twin experiment*

The proposed methodology is tested within a twin experiment where data are generated from a numerical dynamical model. Observations are then created by sub-sampling and adding noise. The aim of this experiment is to highlight the implicit regularizing effect of the deep generative network. To do so, we compare various algorithms on the observation assimilation task. The considered algorithms are 4D-Var with no regularization, 4D-Var with Tikhonov regularization, and deep prior 4D-Var. All the algorithms are implemented with tools based on automatic differentiation such that no adjoint modeling is required, as described in [7]. In all the assimilation experiments, the dynamical model is perfectly known.

### *3.2. Dynamical system*

#### *3.2.1. State*

State variables of the considered system are $\eta$, the height deviation of the horizontal pressure surface from its mean height, and $\mathbf{w}$, the associated velocity field. $\mathbf{w}$ can be decomposed in $u$ and $v$, the zonal

and meridional velocity, respectively. At each time $t$, the system state is then $\mathbf{X}_t = \left(\eta_t \ \mathbf{w}_t^\top\right)^\top$. The considered temporal window has a fixed size.

### *3.2.2. Shallow water model*

The dynamical model used here corresponds to a discretization of the shallow water equations system in Eq. (8) with first order upwind numerical schemes. These schemes are implemented using a natively differentiable software. $H$ represents the mean height of the horizontal pressure surface and $g$ the acceleration due to gravity. After reaching an equilibrium starting from Gaussian random initial conditions, system trajectories are simulated as shown in Figure 2.

$$\begin{cases} \dfrac{\partial \eta}{\partial t} + \dfrac{\partial (\eta + H)u}{\partial x} + \dfrac{\partial (\eta + H)v}{\partial y} = 0 \\ \dfrac{\partial u}{\partial t} + g\dfrac{\partial \eta}{\partial x} = 0 \\ \dfrac{\partial v}{\partial t} + g\dfrac{\partial \eta}{\partial y} = 0 \end{cases} \tag{8}$$

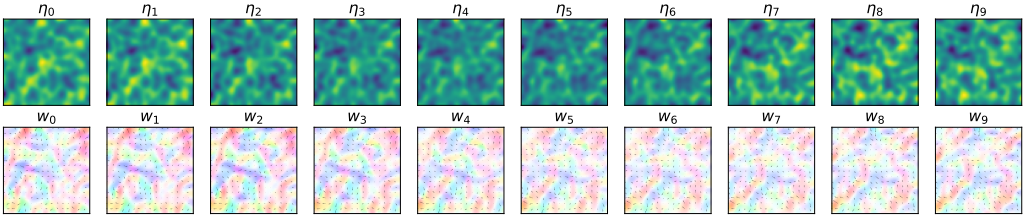


***Figure 2.*** *Example of simulated trajectory with the shallow water numerical model.*

### *3.2.3. Observations*

At regular observational dates, $\eta$ is fully observed up to an additive white noise, see Figure 3. The velocity field $\mathbf{w}$ is never observed. This means that at observational date $t$, the observation operator $\mathbb{H}_t$ is then a linear projector so that $\eta_t = \mathbb{H}_t \mathbf{X}_t$.

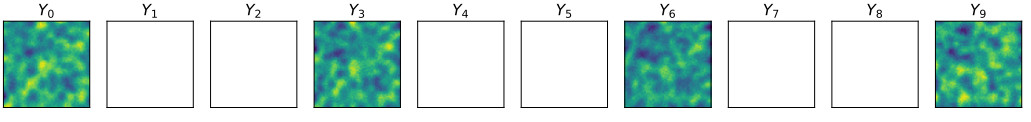


***Figure 3.*** *Example of simulated system observations.*

## *3.3. Regularization*

The role of the assimilation task is then to estimate the velocity field from successive observations of $\eta$. Such motion estimation inverse problem can be ill-posed and may need regularization. The dynamical model being considered perfect, all the velocity fields within the window are determined by the initial field $\mathbf{w}_0$.

#### *3.3.1. No regularization*

The 4D-Var version without regularization only optimizes the fit-to-data term in the cost function. This means that no prior knowledge on the solution can be used, the background covariance matrix $\mathbf{B}$ vanishes.

#### *3.3.2. Tikhonov regularization*

On the other hand, the "Tikhonov" 4D-Var algorithm optimizes the fit-to-data term and also a penalty term. The estimated motion field is forced to be smooth by constraining $\|\nabla \mathbf{w}_0\|_2^2$ and $\|\nabla . \mathbf{w}_0\|_2^2$ to be small. As proved in [11], these terms can be directly included in the background error using a particular matrix $\mathbf{B}$ such that $\alpha\|\nabla \mathbf{w}_0\|_2^2 + \beta\|\nabla . \mathbf{w}_0\|_2^2 = \|\mathbf{X}_0 - \mathbf{X}_b\|^2_{B_{\alpha,\beta}}$ where $\alpha$ and $\beta$ are parameters to be tuned. Such regularization is a classical optical flow penalty [8] and can be used for ocean motion estimation [3].

#### *3.3.3. Deep prior*

As depicted in the method, the only assumption made is about the architecture of the network $g_{\boldsymbol{\theta}}$ generating the solution $\mathbf{w}_0$. Obviously, the chosen architecture is critical for performances. In this experiment, we use a neural architecture similar to the generative convolutional network presented in [16], but replacing deconvolution operations to avoid checkerboard artifacts as described in [13]. The exact architecture is provided in Appendix A.1.

## 4. Results

The first results to look at is the plot of the velocity fields estimated by the algorithms, see Figure 4. The 2D-field of arrows represents the direction and the intensity of the velocity, the colormap provides the same information but helps visualization. Without regularization, the 4D-Var estimation is sharp and seems to suffer from numerical optimization artifacts. On the contrary the deep prior estimate looks less precise but far smoother. The regularized provides the most accurate and smooth estimation. Others examples can be found in Appendix A.2.

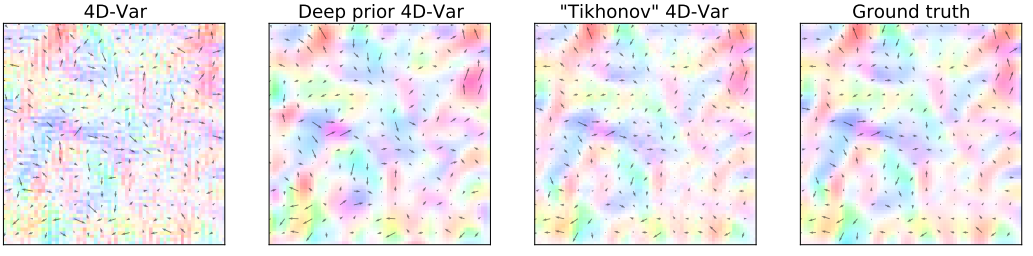


***Figure 4.*** *Example of estimated motion fields* $\mathbf{w}_0$ *with various algorithms.*

Several metrics are calculated to quantify the quality of the estimations. The endpoint error and the angular error are classical optical flow scores, they calculate the Euclidean distance and the average angular deviation between the estimation and the ground truth, respectively. At first glance, there is no statistical difference between deep prior 4D-Var and 4D-Var without regularization.
However, if we dig into smoothness statistics of the estimated fields, $\|\nabla . \mathbf{w}_0\|_2$, $\|\nabla . \mathbf{w}_0\|_2$ and $\|\Delta \mathbf{w}_0\|_2$, it seems that deep prior is able to capture complex statistics of the true motion field. Similar behavior has been noticed in [19]. Looking closer in the histograms, Figure 5, we see that deep prior and "Tikhonov" 4D-Var estimations have smoothness statistics very close to that of the original motion field. Whether it has been explicitly constrained or by deep network design, smooth solutions are

***Table 1.*** *Metrics quantifying the quality of estimated motion field* $\mathbf{w}_0$ *over the assimilated database*

| | Assimilation Score | | Smoothness statistics | | |
|---|---|---|---|---|---|
| Metric[1] | Endpoint error ($\times 10^2$) | Angular error | $\|\|\nabla \mathbf{w}_0\|\|_2$ | $\|\|\nabla . \mathbf{w}_0\|\|_2$ | $\|\|\Delta \mathbf{w}_0\|\|_2$ |
| 4D-Var | $4.2 \pm 0.4$ | $28.4 \pm 9.8$ | $6.1 \pm 0.6$ | $5.3 \pm 0.5$ | $9.9 \pm 1.0$ |
| Deep Prior 4D-Var | $4.6 \pm 2.0$ | $26.7 \pm 5.0$ | $1.9 \pm 0.1$ | $1.6 \pm 0.9$ | $1.0 \pm 0.3$ |
| "Tikhonov" 4D-Var | $1.6 \pm 0.6$ | $9.9 \pm 9.8$ | $2.0 \pm 0.1$ | $1.8 \pm 0.1$ | $1.9 \pm 0.1$ |
| Ground truth | 0 | 0 | $1.7 \pm 0.9$ | $1.6 \pm 0.1$ | $0.7 \pm 0.3$ |

[1] All the metrics are averaged on images

unforced by regularization.

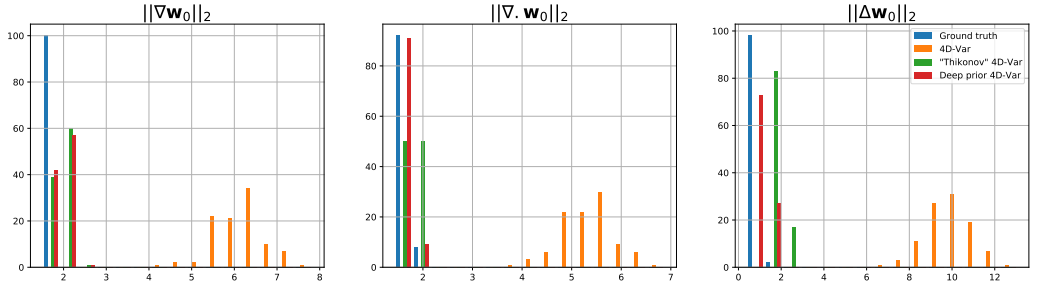


***Figure 5.*** *Histograms of smoothness statistics from estimated motion field* $\mathbf{w}_0$ *with various algorithms.*

It has to be noted that the "Tikhonov" 4D-Var is the only algorithm here that has been treated with hyper-parameters tuning which can explain the large differences in scores. It can be argued that the neural architecture, which is known as a good baseline to generate images, has been tuned through many image processing experiments. However, grid-searching hyper-parameters particularly suited for this experiment should enhance performances.

## 5. Conclusion

We proposed an original method bridging ideas from the image processing and the geosciences communities to solve a variational inverse problem. More precisely we used a neural network as implicit regularization to generate the solution of an initial value problem. To demonstrate its efficiency, we set up a twin experiment comparing different algorithms in a data assimilation task derived from shallow water model. The results show that this kind of regularization can provide an interesting alternative when prior knowledge is not available. However in our case, we observed that expert-driven handcrafted regularization provides better performances. Finally, this work opens the way for further developments on the architecture design, but also in a more realistic context where the numerical dynamics is imperfect.

## References


1. H. Abarbanel, P. Rozdeba, and S. Shirman. Machine learning: Deepest learning as statistical data assimilation problems. *Neural Computation*, 30(8):2025–2055, 2018.
2. M. Asch, M. Bocquet, and M. Nodet. *Data assimilation: methods, algorithms, and applications*. Fundamentals of Algorithms. SIAM, 2016.
3. D. Béréziat and I. Herlin. Image-based modelling of ocean surface circulation from satellite acquisitions. In *International Conference on Computer Vision Theory and Application VISAPP*, pages 1–8, January 2014.
4. D. Béréziat and I. Herlin. Motion and Acceleration from Image Assimilation with evolution models. *Digital Signal Processing*, 83:45–58, 2018.
5. D. Bereziat, I. Herlin, and L. Younes. A generalized optical flow constraint and its physical interpretation. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No.PR00662)*, volume 2, pages 487–492 vol.2, 2000.
6. Emmanuel de Bezenac, Arthur Pajot, and Patrick Gallinari. Deep learning for physical processes: Incorporating prior scientific knowledge. In *International Conference on Learning Representations ICLR*, 2018.
7. Arthur Filoche, Dominique Béréziat, Julien Brajard, and Anastase Alexandre Charantonis. Variational assimilation of Geophysical images leveraging deep learning tools. In *ORASIS 2021*, September 2021.
8. Berthold K. P. Horn and Brian G. Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.
9. Christine Johnson, Brian J. Hoskins, and Nancy K. Nichols. A singular vector perspective of 4D-Var: Filtering and interpolation. *Quarterly Journal of the Royal Meteorological Society*, 131:1–19, 2005.
10. F-X. Le Dimet and O. Talagrand. Variational algorithms for analysis and assimilation of meteorological observations: theoretical aspects. *Tellus*, 38A(10):97, 1986.
11. Y. Lepoittevin and I. Herlin. Regularization terms for motion estimation. Links with spatial correlations. In *International Conference on Computer Vision Theory and Applications VISAPP*, volume 3, pages 458–466, 2016.
12. Lukas Mosser, Olivier Dubrule, and Martin J. Blunt. Stochastic seismic waveform inversion using generative adversarial networks as a geological prior, 2018.
13. Augustus Odena, Vincent Dumoulin, and Chris Olah. Deconvolution and checkerboard artifacts. *Distill*, 2016.
14. Gregory Ongie, Ajil Jalal, Christopher A. Metzler, Richard G. Baraniuk, Alexandros G. Dimakis, and Rebecca Willett. Deep learning techniques for inverse problems in imaging. *IEEE Journal on Selected Areas in Information Theory*, 1(1):39–56, 2020.
15. A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic Differentiation in PyTorch. In *NIPS 2017 Workshop on Autodiff*, 2017.
16. Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *International Conference on Learning Representation ICLR*, 2016.
17. Jonathan Tompson, Kristofer Schlachter, Pablo Sprechmann, and Ken Perlin. Accelerating eulerian fluid simulation with convolutional networks. In *International Conference on Machine Learning, ICML*, pages 5258–5267, 2017.
18. Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. Deep image prior. In *Conference on Vision and Pattern Recognition (CVPR)*, 2018.
19. Jaejun Yoo, Kyong Jin, Harshit Gupta, Jérôme Yerly, Matthias Stuber, and Michael Unser. Time-dependent deep image prior for dynamic MRI. *IEEE Transactions on Medical Imaging*, PP:1–1, 05 2021.

## A. Appendix. Experiment details

### *A.1. Neural network architecture*

```
----------------------------------------------------------------
        Layer (type)               Output Shape          Param #
================================================================
   ConvTranspose2d-1             [-1, 512, 4, 4]         819,200
       BatchNorm2d-2             [-1, 512, 4, 4]           1,024
              ReLU-3             [-1, 512, 4, 4]               0
          Upsample-4             [-1, 512, 8, 8]               0
   ReflectionPad2d-5          [-1, 512, 10, 10]               0
            Conv2d-6             [-1, 256, 8, 8]       1,179,904
       BatchNorm2d-7             [-1, 256, 8, 8]             512
              ReLU-8             [-1, 256, 8, 8]               0
          Upsample-9          [-1, 256, 16, 16]               0
  ReflectionPad2d-10          [-1, 256, 18, 18]               0
           Conv2d-11          [-1, 128, 16, 16]         295,040
      BatchNorm2d-12          [-1, 128, 16, 16]             256
             ReLU-13          [-1, 128, 16, 16]               0
         Upsample-14          [-1, 128, 32, 32]               0
  ReflectionPad2d-15          [-1, 128, 34, 34]               0
           Conv2d-16           [-1, 64, 32, 32]          73,792
      BatchNorm2d-17           [-1, 64, 32, 32]             128
             ReLU-18           [-1, 64, 32, 32]               0
         Upsample-19           [-1, 64, 64, 64]               0
  ReflectionPad2d-20           [-1, 64, 66, 66]               0
           Conv2d-21            [-1, 3, 64, 64]           1,731
             Tanh-22            [-1, 3, 64, 64]               0
================================================================
Total params: 2,371,587
Trainable params: 2,371,587
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.00
Forward/backward pass size (MB): 11.03
Params size (MB): 9.05
Estimated Total Size (MB): 20.08
----------------------------------------------------------------
```

### *A.2. Supplementary examples*

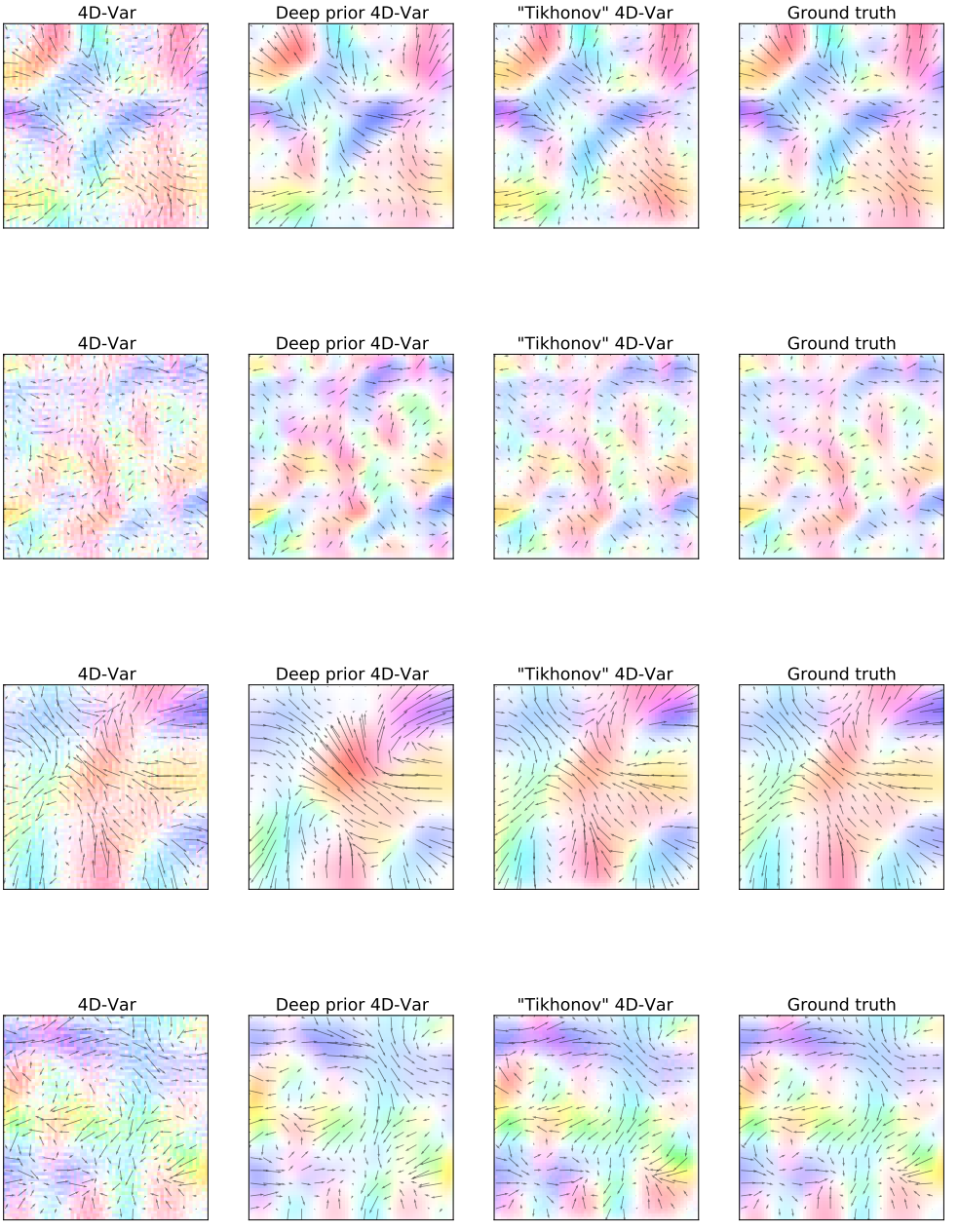


***Figure 6.*** *Examples of estimated motion fields* $\mathbf{w}_0$ *with various algorithms, each line corresponds to a different assimilation window.*