Data Learning: Integrating Data Assimilation and Machine Learning

Caterina Buizza[b], César Quilodrán Casas[a], Philip Nadler[a], Julian Mack[a], Stefano Marrone[a], Zainab Titus[c], Clémence Le Cornec[d], Evelyn Heylen[e], Tolga Dur[a], Luis Baca Ruiz[a], Claire Heaney[c], Julio Amador Díaz Lopez[a], K S Sesh Kumar[a], Rossella Arcucci[c,a]

[a]*Data Science Institute, Department of Computing, Imperial College London, UK*
[b]*Personal Robotics Lab, Department of EEE, Imperial College London, UK*
[c]*Department of Earth Science and Engineering, Imperial College London, UK*
[d]*Department of Civil and Environmental Engineering, Imperial College London, UK*
[e]*Control and Power Group, Department of EEE, Imperial College London, UK*

Abstract

Data Assimilation (DA) is the approximation of the true state of some physical system by combining observations with a dynamic model. DA incorporates observational data into a prediction model to improve forecasted results. These models have increased in sophistication to better fit application requirements and circumvent implementation issues. Nevertheless, these approaches are incapable of fully overcoming their unrealistic assumptions. Machine Learning (ML) shows great capability in approximating nonlinear systems and extracting meaningful features from high–dimensional data. ML algorithms are capable of assisting or replacing traditional forecasting methods. However, the data used during training in any Machine Learning (ML) algorithm include numerical, approximation and round off errors, which are trained into the forecasting model. Integration of ML with DA increases the reliability of prediction by including information with a physical meaning. This work provides an introduction to Data Learning, a field that integrates Data Assimilation and Machine Learning to overcome limitations in applying these fields to real-world data. The fundamental equations of DA and ML are presented and developed to show how they can be combined into Data Learning. We present a number of Data Learning methods and results for some test cases, though the equations are general and can easily be applied elsewhere.

Keywords: Data Learning, Data Assimilation, Machine Learning

1. Introduction and Motivation

All numerical models introduce uncertainty through the selection of scales and parameters, and any computational method contributes to uncertainty due to linearization, discretization, finite precision and round-off errors [15]. It is essential to account for these uncertainties when producing an acceptable numerical simulation.

Numerical forecasting models often use Data Assimilation (DA) methods for uncertainty quantification as they allow the inclusion of observational data into a prediction model to improve forecasted results [6]. In this way, problems with uneven data distribution or redundancy can be addressed. In the past 20 years, DA methodology has become the main component in the development and validation of mathematical models in meteorology, climatology, geophysics, geology and hydrology [38, 6]. More recently, DA has also been applied to numerical simulations of geophysical applications, medicine and biological science [4].

Conventional methods for DA include Kalman filters and variational approaches [6]. They have increased in sophistication to better fit their application requirements and circumvent their implementation issues. Nevertheless, these approaches are incapable of fully overcoming their unrealistic assumptions: in particular linearity, normality and zero error covariances [51]. However, with the rapid developments in recent years, ML shows great capability in approximating nonlinear systems and extracting meaningful features from high-dimensional data [7, 60].

ML algorithms are capable of assisting or replacing traditional forecasting methods, without the need to know explicit equations describing the underlying system. However, the training data required for machine learning include numerical approximation and round off errors, which are trained into the forecasting model. In addition, building ML models becomes difficult in many real-world scenarios [7] due to:

1. Dimensionality constraints: matrices become so large that they are difficult to work with;
2. Noisy data: uncertainty and noise in the data creates serious error propagation;

3. Low-quality data: the data do not provide meaningful information over the whole field.

To improve the efficiency and accuracy of numerical simulations, we introduce *Data Learning*, which integrates DA and ML methods. This increases the reliability of prediction and reduces error by including information with a physical meaning from observed data. The resulting cohesion produces a collection of faster and more accurate predictive models, based on the idea that ML can be used to learn the past experiences of a DA process. This follows the principles of the Bayesian approach and in fact both fields present inverse methods that can be united under a Bayesian framework, as presented formally in [31].

In this paper, we present an overview of combining ML and DA into Data Learning (DL) to resolve the three limitations mentioned above, i.e., dimensionality constraints, noisy data and low-quality data. We will give an overview of the proposed method of DL, and the differences to existing proposals of this kind in Section 2. In this section we will also present a number of related works and the contribution of this work. We show different approaches of DL, along with results in particular application areas in Sections 3, 4 and 5. Conclusions are summarised in Section 6.

1.1. Limitations in applying Machine Learning to Real-World Scenarios

Machine learning is successfully used in a wide range of applications when the conditions allow it. However, these methods still struggle in cases where either the dimensions are very large, the data is noisy or the data do not adequately cover the whole domain. In this section, we introduce the limitations of ML models in their use to solve real world problems. We will first address Neural Networks (NNs) [32] and then Gaussian Processes (GPs) [60].

NNs are stochastic rather than deterministic [69, 42]. Thus, when applied to deterministic systems, they may well succeed, but without ever learning the relationship between the variables. For example, in weather forecasting, NN algorithms do not know when they are violating the laws of physics [11, 26]. They also require large amounts of data before they begin to produce reliable results, and the larger the architecture, the more data is needed. In addition, if the data available is too noisy, too scarce or there is a lack of salient features to represent the problem, the network will not perform well. This can happen either if there is a lack of good features or a lack of good ground truth data. In [41], a large dataset is used to develop a new emission model using NN. However, with only vehicle speed and acceleration as

inputs, the NN is not able to find a better representation than a non-linear regression model. These problems can be overcame with the use of machine learning approaches based on small datasets. One-shot learning was introduced in [28] with the intention of learning the classification of images based on a small number of samples. In [67] this idea was extended to the concept of matching networks which maps a small labelled support set and an unlabelled example to its label, skipping fine-tuning to new classes. Furthermore, physics-informed neural networks (PINN) were introduced in [58]. These are neural networks that are trained to solve supervised learning problems with the constraints of given laws of physics described by general nonlinear PDEs. This aims to solve the problem of the cost of data acquisition for data-driven models based, for example, on computational fluid dynamics simulations, and making decisions under partial information.

A fundamental assumption of GPs is that the measurement values have Gaussian uncertainty. In addition, GPs are non–parametric, meaning not only that training data must be kept at inference time, but also that the computation cost of prediction scales cubically with the number of training samples [60]. GPs also do not deal well with discontinuities in data, such as, edges in images, a financial crash of stock market [68]. These caveats can be solved by using inducing points. In [65] a variational formulation for sparse approximations was introduced. This approach jointly infers the inducing inputs and the kernel hyperparameters. [70] introduced the concept of convolutional Gaussian Processes to tackle the handling of large amounts of data, like high-dimensional inputs like images, giving the GP a convolutional structure.

In order to overcome these limitations, we propose new methods of DL that combine ML methods with DA in the next section.

2. Data Learning Approach and Related Works

In this section, we will give an overview of DL methods that combine ML and DA. We also explain the difference with the existing methods of this kind. We introduce related works for the main methodologies we use in this paper: Machine Learning, Reduced Order Modelling and Data Assimilation and we present the contribution of the present work.

The novelty of DL is in the integration approach that is proposed in [9]. We have seen in section 1 how methods from each of the fields of ML and DA bring different strengths and weaknesses to a system. In Data Learning, we

propose to combine modules from either of these fields into a larger complete system as shown in Figure 1. This modular approach is facilitated by recent

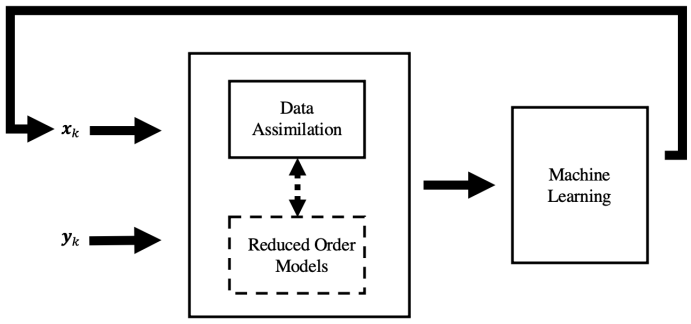


Figure 1: General Data Learning framework.

advances in understanding ML methods, especially NN [45]. The difference from existing approaches that attempt to fuse the two fields is that our methods do not require specific conditions to be successful, as detailed in Section 2.1. In our pipeline approach, we can rely on methods that are already well defined and stable, without the need to change how they work. This is the reason that DL can be applied to so many fields.

DA is used to infer and learn dynamical systems. ML methods can be interpreted as dynamical systems. This facilitates the use of dynamical system theory to understand the behaviour of ML systems [73]. This also means that DA can be used to interpret ML.

Thus ML systems can be trained using DA methods, as shown in recent advances such as [69] and [42]. In [21], the *Adjoint method* is used for NN training. Furthermore, there are a number of applications, ranging from oceanography to finance and epidemiology ([50, 39, 49]), that can be tackled by modelling them as dynamical systems using NN.

With the increasing number of ML models applied to real world applications, the field must develop a way for information from real observations to constrain them. Without this input from real observations, most ML systems are not suited to realistic cases and detach quickly from the underlying model they are based on. Observations can also be noisy and sparse, which disrupts the interpretation of these data by ML models not specifically designed to take this into account [12, 13]. In addition, some models do not take the variation of observations in space and time into account. This is

where DA becomes necessary. Through the DL framework, a DA system can be applied to help the dynamical model (in this case, to help the ML model) to find the optimal initial conditions. While DA can be computationally expensive, Reduced Order Models (ROMs) such as dimensionality reduction make them computationally more efficient. ROMs based on ML methods also preserve most of the variance of the original model [20]. Therefore, DA helps to improve ML model, while also benefiting from ML techniques [55].

2.1. Other approaches to combining ML and DA

The relationship between the learning, interpretation and application of data in ML and DA has been explored in several works already. Other methods fuse both ML and DA in such a way that they are very problem-specific. We propose a modular approach to build a framework that can solve a large class of problems.

Unifying the frameworks of DA and ML from a Bayesian perspective highlights the interface between probabilistic ML methods and differential equations. This equivalence is presented formally in [31], and shows the similarities between the two fields. Here the equivalence of four-dimensional Variational Data Assimilation (4D-Var) and Recurrent Neural Networks, and how approximate Bayesian inverse methods (i. e. Variational Data Assimilation (VarDA) in DA and back-propagation in ML) can be used to merge the two fields. A slightly different perspective has been shown in [10] where Variational Data Assimilation and Machine Learnings are seen as already integrated in a Weak Constraint Variational Data Assimilation. These methods are also particularly suited to systems involving Gaussian process, as presented in [8, 57, 53]. These are data-driven algorithms capable of learning nonlinear, space-dependent cross-correlations and of estimating model statistics under a unified approach.

Application of these fusion algorithms have been presented both in real-world applications such as air quality forecasting [43] using a data-driven ML approach, and in more theoretical problems, such as that of spatio-temporal oscillations of the FitzHugh - Nagumo PDE [71], using numerically computed approximations of Koopman eigenfunctions and eigenvalues. Other approaches are more similar to the work presented in this paper, for example in [12], where an Ensemble Kalman filter (EnKF) and a NN are applied iteratively to emulate hidden dynamics and predict future states.

The common theme in fusion-based approaches is that they are specific to particular methods, rather than allowing the flexibility of choice between

different modules depending on the needs of the overall system.

We will discuss the application of different DL integrations in detail in Section 3 and Section 4. First however, we will discuss relevant background literature in more detail in the next Section.

2.2. Related works for the main methodologies we use in this paper

Several methods exist that use ROM with ML. In [72], the authors use Gaussian Process Regression (GPR) to predict flow in an urban neighbourhood. Principal Component Analysis (PCA) was performed to obtain basis functions used to train the GPR. This predicted reasonable results with similar performance to the high-fidelity model. In [36], the authors present a method using Proper Orthogonal Decomposition (POD), which uses multilayer perceptrons to approximate the POD coefficients of the reduced model and apply this to a nonlinear Poisson equation and lid-driven cavity flow. [64] presents a similar method using POD and compares four machine learning techniques (neural networks, multivariate polynomial regression, k-nearest-neighbours and decision trees). These methods are used to predict the pressure field of an airfoil and the strain field over a damaged composite panel. The authors observe that in engineering applications there is less data available than in other machine learning applications and, as a result, it can be beneficial to incorporate knowledge from physical models.

ROM of reservoir systems have been developed using machine learning techniques such as Autoencoder (AE) [16] and generative adversarial networks [48]. Investigation into the use of Convolutional Variational AEs for parameterisation in an Ensemble Smoother with Multiple Data Assimilation (ES-MDA), have demonstrated reasonable potential for the integration of ML methods into DA frameworks for reservoir simulation [17], or a chaotic system [75, 5]. More recent papers use the term 'data driven' to emphasise the role of ML: for example, in [33], GPR is used to predict POD coefficients for unseen parameters in 2D flow past a cylinder. The 'Digital Twin' framework is presented in [59], within which many of the methods described here fit.

To overcome noise or to incorporate time-series information, methods exist that merge Kalman filters and ML. In [22], the authors account for temporal information in human motion analysis by integrating a Kalman filter into a Long Short Term Memory (LSTM) neural network. Instead in [14], the authors use Kalman tracking to speed up any learning-based

motion tracking method to real-time and to correct some common inconsistencies in camera-based motion tracking methods. VarDA is often applied in any system where a reasonable cost function can be defined. In recent years, an advanced hybrid data assimilation technique, the Iterative Ensemble Kalman Smoother (IEnKS), which integrates ensemble-based methods with variational techniques, has been shown to improve the accuracy of the assimilation process [27]. However, the application of this method can be computationally expensive due to the use of adjoints in gradient-based optimisation for the variational analysis [6]. Using a computationally convenient goal-based sensitivity method [15] with ensembles and explicit time windows for computing adjoints [35], rapid convergence to a solution has been achieved for an advection test case to highlight the capability of the method for large scale problems.

In weather forecast systems with ML and DA [25], deep learning emulators could be used to generate tangent linear and adjoint model codes for 4D-Var DA [34]. Papers describing similarities between ML and DA in a Bayesian framework, suggesting ways to combine the two [30].

Recent research in the field of public opinion prediction has developed techniques to reduce misclassifications using weakly supervised methods. Such methods have been successful in eliminating and managing temporal biases. More recently, DA methods have been leveraged to analyze how assimilating offline polling data to online public opinion data can correct for such biases. Even if both these approaches are still in their infancy, preliminary results appears to successfully remove part of the biases present in Twitter data. Both weakly supervised pruning of misclassifications and data assimilation appear to be promising avenues of research.

2.3. Contribution

As mentioned in Section 1.1, the limitations of traditional ML methods can be grouped into three areas, namely: dimensionality constraints, noisy data and low-quality data. In this paper, we present methods for dealing with one or more of these three constraints using DL. In particular:

1. *Data Learning for Dimensionality Constraints and Noisy Data*: We introduce
   (a) a DL model based on the integration of Kalman Filters with Reduced Order Models and Neural Networks, described in Section 3.1 and tested in Section 5.1 on an oceanographic problem.

(b) a DL model based on the integration of Variational Data Assimilation with Encoder-Decoder technologies, as shown in Section 3.2 and tested on an air-pollution forecasting problem in Section 5.2.

2. *Data Learning for Noisy Data and Low-Quality Data*: In particular we faced these problems introducing
   (a) a DL approach that integrates Convolutional Neural Networks with Kalman Filters as presented in Section 4.1 and tested for human motion tracking in Section 5.3. A variation of this DL model which uses a Variational Data Assimilation instead of a Kalman Filter is also tested in Section 5.6 for pharmacokinetic modelling.
   (b) a DL approach that integrates Variational Data Assimilation with Neural Networks for Parameter Estimation in Section 4.2 and tested for an Economic system in Section 5.4.
   (c) a DL approach that integrates Gaussian Processes with Variational Data Assimilation, as shown in Section 4.3 and tested in Section 5.5 for an optimal sensor placement problem.

For these DL models, we introduce the mathematical formulation and evaluate the accuracy of the results compared to results provided by standard ML or DA methods. We will discuss the results and performance of these methods in more detail in Section 5. Validation is provided for some test cases, though the algorithms and numerical methods proposed in this work can be applied to other physical, engineering, economic or medical problems involving other equations or state variables, as shown in Figure 7.

3. Data Learning for Dimensionality Constraints and Noisy Data

The dimensionality of data increases very quickly in real-world scenarios, leading to unmanageable computational costs. ROM has established itself within computational modelling as a very useful tool, especially for solving optimisation problems or for making real-time predictions. A ROM is a surrogate for a high fidelity model (HFM)[1] and aims to reproduce the behaviour of the HFM in much less time.

Typically ROM can be split into an offline stage (where the HFM is run many times) and an online stage (which runs quickly due to the reduction in

[1]Models that do not perform any reduction

the number of degrees of freedom). In the offline stage, solutions of the HFM are produced for different parameter values, known as snapshots, to which PCA or an Autoencoder is applied.

The choice between these depends on whether we require nonlinearity in the reduced model. AEs allow the use of traditional DA methods on a lower-dimensional space, whilst maintaining the underlying nonlinearity of the data.

3.1. Integration of Kalman Filters with Reduced Order Models and Neural Networks

In this section, we present a DL model that combines Data Assimilation, Reduced Order Modelling and Neural Networks. Typically, this is made up of two steps: a *prediction* step and a *correction* step. At step k the result of the previous forecast is denoted by $\mathbf{x}_k^f$, and the measurement by $\mathbf{y}_k$. Based on these two vectors, we compute $\mathbf{x}_k^a$ (a for *analysis*). We then use the evolution model, which is usually (partial) differential equation-based, to obtain a prediction of the state at step $k+1$. The result of the forecast is denoted $\mathbf{x}_k^f$ and becomes the initial guess for the next time step. The goal of the DA, in this case a Kalman filter (KF), is to compute an optimal a posteriori (analysis) estimate, $\mathbf{x}_k^a$, that is a linear combination of an a priori (forecast) estimate, $\mathbf{x}_k^f$, and a weighted difference between the actual measurement, $\mathbf{y}_k$, and the measurement prediction, $\mathbf{H}_k\mathbf{x}_k^f$. The filter must be of the form

$$\mathbf{x}_k^a = \mathbf{x}_k^f + \mathbf{K}_k\left(\mathbf{y}_k - \mathbf{H}_k\mathbf{x}_k^f\right) \tag{3.1}$$

where $\mathbf{K}$ is the Kalman gain. The difference $\left(\mathbf{y}_k - \mathbf{H}_k\mathbf{x}_k^f\right)$ is called the *innovation* and reflects the discrepancy between the actual and the predicted measurements at k. The Kalman gain, the measurement update and the error covariance update equations are given by (3.1) with:

$$\mathbf{K}_k = \mathbf{P}_k^f\mathbf{H}_k^T(\mathbf{H}_k\mathbf{P}_k^f\mathbf{H}_k^T + \mathbf{R})^{-1} \tag{3.2}$$

$$\mathbf{P}_k^a = (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_k^f \tag{3.3}$$

and the prediction update is given by:

$$\mathbf{x}_{k+1}^f = \mathbf{M}\mathbf{x}_k^a \tag{3.4}$$

sometimes coupled with an error covariance update, where $\mathbf{H}_k$ is the observation model, $\mathbf{R}$ is the observation noise and $\mathbf{y}_k$ is a measurement [6]. The

prediction-correction steps as implemented in equations (3.1)-(3.3) are computationally very expensive because of the inversion of big matrices and the presence of the forecasting model $\mathbf{M}$ in (3.4).

The model presented here aims to produce fast DA with sufficiently accurate results. This is achieved by building a reduced-space model that acts

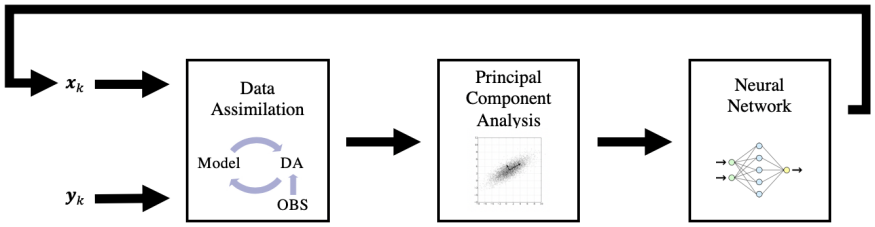


Figure 2: Integrating DA, PCA and NNs to address dimensionality constraints, noisy data and low-quality data. The PCA allows the DA to be performed in the reduced space.

as a substitute for the full-state (FS) model. This is represented in Figure 2 and can be divided into three steps: (a) DA, (b) PCA, and (c) NN. The first step is to compute

$$\mathbf{x}^a = [\mathbf{x}_1^a, \ldots, \mathbf{x}_m^a] \tag{3.5}$$

where

$$\mathbf{x}_k^a = \mathbf{x}_k^f + \mathbf{K}_k \left(\mathbf{y}_k - \mathbf{H}_k \mathbf{x}_k^f \right)$$

as in Equation (3.1). The second step is the dimension reduction using PCA. PCA (equivalently POD) is a commonly used technique to reduce dimensionality by extracting only the components that contribute most to the variance of a system. Because of its simplicity and analytic derivation, PCA has become a widely popular statistical tool in atmospheric, ocean and climate science.

PCA is an unsupervised learning method that simplifies high-dimensional data by geometrically projecting the data into lower dimensions called Principal Components (PCs), which are a linear combination of the original variables.

Before performing PCA, each variable is reshaped into a two-dimensional matrix $\mathbf{x}_k^a$:

$$\mathbf{x}_k = \begin{bmatrix} x_{1,1}^a & x_{1,2}^a & \cdots & x_{1,m}^a \\ x_{2,1}^a & x_{2,2}^a & \cdots & x_{2,m}^a \\ \vdots & \vdots & \ddots & \vdots \\ x_{n,1}^a & x_{n,2}^a & \cdots & x_{n,m}^a \end{bmatrix}_k \tag{3.6}$$

where m is the number of data points of the k^{th} variable and n is the number of time steps.

In order to not bias the results, it is necessary to standardise each $\mathbf{x}_k^a$ by subtracting the column-wise mean and dividing by the standard deviation $\sigma_{\mathbf{x}_k^a}$. This gives the standardised matrices $\bar{\mathbf{x}}_k^a$, where each variable is weighted with equal importance while maintaining the relative magnitude of variability at each model grid point within a variable. The horizontal concatenation of these matrices results in

$$\bar{\mathbf{x}}_k^a = [\bar{\mathbf{x}}_1^a, \cdots, \bar{\mathbf{x}}_q^a]. \tag{3.7}$$

PCA is performed by singular value decomposition, after which $\bar{\mathbf{x}}_k^a$ can be expressed as

$$\bar{\mathbf{x}}_k^a = \mathbf{P}\mathbf{\Pi} \tag{3.8}$$

where $\mathbf{P} \in \mathbb{R}^{n\times n}$ contains the time series of the PCs and $\mathbf{\Pi} \in \mathbb{R}^{n\times m}$ are the PCs of $\bar{\mathbf{x}}_k^a$.

The dimension reduction of the system is then obtained by truncating $\mathbf{P}$ and $\mathbf{\Pi}$. Let ν be the number of selected PCs, then the reduced space anomaly matrix $\bar{\mathbf{x}}_{\nu,k}^a$ is defined by:

$$\bar{\mathbf{x}}_{\nu,k}^a = \mathbf{P}_\nu \Pi_\nu = [\bar{\mathbf{x}}_1^a, \cdots, \bar{\mathbf{x}}_\nu^a] \tag{3.9}$$

with $\mathbf{P}_\nu \in \mathbb{R}^{n\times\nu}$ and $\mathbf{\Pi}_\nu \in \mathbb{R}^{\nu\times m}$, where $\mathbf{P}_\nu$ is computed by considering the first ν columns of $\mathbf{P}$ and $\mathbf{\Pi}_\nu$ is computed from $\mathbf{\Pi}$ by considering the first ν rows.

The final step consists in training a NN to learn the time series $\{\mathbf{x}_{\nu,k}^a\}_{k=1}^m$. This is achieved by training a NN that will emulate the physics of reconstructed full-state (RFS) with an acceptable degree of accuracy, i. e. the training set becomes

$$\big(\mathbf{x}^{tr}(l_0), \mathbf{x}^{tr}(l_F)\big) = \Big(\mathbf{x}_{\nu,k}^a, \mathbf{x}_{\nu,k+1}^a\Big) \tag{3.10}$$

and the loss function [7] becomes

$$C\big(\mathbf{a}(l_i), \mathbf{x}_{\nu,k}^a\big) = \frac{1}{N_p}\sum_{n=1}^{N_p}\frac{1}{2N_d}\sum_{j=1}^{N_d}\mathbf{R}_j(l_i)\big[\mathbf{a}_j(l_i) - \mathbf{x}_{\nu,k,j}^a\big]^2. \tag{3.11}$$

This model has been tested for an application in Oceanography; results are shown in Section 5.1.

3.2. Integration of Variational Data Assimilation with Encoder-Decoder technologies

VarDA is based on optimal control theory, which is in turn derived from the *calculus of variations*. The analysed state does not aim to maximise a probability density function (pdf), but to minimise a *cost function*. This is usually done using numerical methods based on the gradient of the cost function, which can be obtained with the aid of adjoint methods [6]. At step k the result of the previous forecast is denoted by $\mathbf{x}_k^f$, and the measurement by $\mathbf{y}_k$. Based on these two vectors, we compute $\mathbf{x}_k^a$ (a for *analysis*) by minimising the cost function:

$$J(\mathbf{x}_k) = \frac{1}{2}\big(\mathbf{x}_k - \mathbf{x}_k^f\big)^T \mathbf{B}^{-1}\big(\mathbf{x}_k - \mathbf{x}_k^f\big) + \frac{1}{2}\big(\mathbf{H}\mathbf{x}_k - \mathbf{y}_k\big)^T \mathbf{R}^{-1}\big(\mathbf{H}\mathbf{x}_k - \mathbf{y}_k\big). \quad (3.12)$$

where $\mathbf{H}_k$ is the observation model, $\mathbf{R}$ is the observation noise and $\mathbf{y}_k$ is a measurement [6]. The matrices involved in this calculation are often so large they cannot be easily manipulated or stored in memory [38]. In most useful DA problems the model state space is large: $n \geq \mathcal{O}(10^6)$. Whether using VarDA, or KFs [37], the naive formulations have n and n^2 dependencies[2]. Thus we must perform DA in a reduced space.

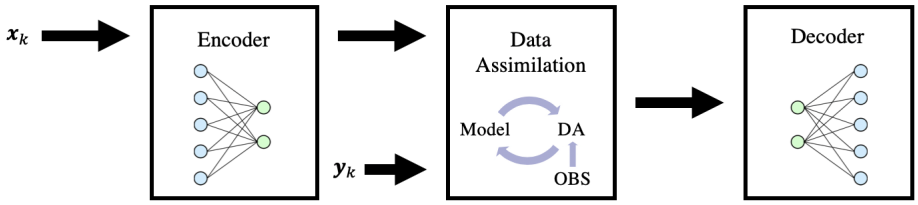


Figure 3: Integration of VarDA and AE for dealing with dimensionality constraints and noisy data. The Encoder-Decoder allows the DA to be performed in the reduced space.

If Equation (3.12) is linearised around the background state, the VarDA problem can be written in the following form:

$$\boldsymbol{\delta}\mathbf{x}_k^{DA} = \text{argmin}_{\boldsymbol{\delta}\mathbf{x}_k} J(\boldsymbol{\delta}\mathbf{x}_k) \quad (3.13)$$

[2]In fact KFs require the inversion of the background covariance matrix of size n which is in $\mathcal{O}(n^3)$.

with

$$J(\boldsymbol{\delta}\mathbf{x}_k) = \frac{\alpha}{2}\,\boldsymbol{\delta}\mathbf{x}_k^T\mathbf{B}^{-1}\boldsymbol{\delta}\mathbf{x}_k + \frac{1}{2}(\mathbf{H}\boldsymbol{\delta}\mathbf{x}_k - \mathbf{d}_k)^T\mathbf{R}^{-1}(\mathbf{H}_k\boldsymbol{\delta}\mathbf{x}_k - \mathbf{d}_k) \tag{3.14}$$

where $\mathbf{d}_k = [\mathbf{y}_k - \mathbf{H}_k\mathbf{x}_k^f]$ is the innovation, $\mathbf{H}_k$ is here the linearised observational operator, and $\boldsymbol{\delta}\mathbf{x}_k = \mathbf{x}_k - \mathbf{x}_k^f$ are the increments.

In equation (3.14), the minimisation problem is defined on the field of increments [24]. In order to avoid the inversion of $\mathbf{B}$, as $\mathbf{B} = \mathbf{V}\mathbf{V}^T$ [2], the minimisation can be computed with respect to a new variable $\mathbf{w}_k = \mathbf{V}^+\boldsymbol{\delta}\mathbf{x}_k$ [44], where $\mathbf{V}^+$ denotes the generalised inverse of $\mathbf{V}$, yielding:

$$\mathbf{w}_k^{DA} = \text{argmin}_{\mathbf{w}_k}\, J(\mathbf{w}_k) \tag{3.15}$$

with

$$J(\mathbf{w}_k) = \frac{\alpha}{2}\mathbf{w}_k^T\mathbf{w}_k + \frac{1}{2}(\mathbf{H}\mathbf{V}\mathbf{w}_k - \mathbf{d}_k)^T\mathbf{R}^{-1}(\mathbf{H}\mathbf{V}\mathbf{w}_k - \mathbf{d}_k). \tag{3.16}$$

As the background error covariance matrix is ill-conditioned, in order to improve the conditioning only PCs of the largest eigenvalues of the error covariance matrix are considered. PCA allows a decomposition of a data function into a set of orthogonal functions, which are designed so that only a few of these functions are needed in lower-dimensional approximations. Furthermore, since the PCs are the eigenvectors of the error covariance matrix, the condition number of the latter is reduced as well. Nevertheless, the accuracy of the solution obtained by truncating PCs exhibits a severe sensibility to the variation of the value of the truncation parameter, and so a suitable choice of the number of PCs is strongly recommended [15]. This issue introduces a severe drawback to the reliability of PCA truncation, and hence to the usability of the operative software in different scenarios [2]. To tackle this issue, we use AEs.

AEs are an alternative to using PCA [46]. AEs are a self-supervised machine learning method that can be used to produce a compressed representation of data. AEs are useful in any problem in which a latent data representation is required and whenever the reduced space must maintain the nonlinear properties of the full space. They are trained using historical or model-generated data to perform a significant reduction of state dimensions. AEs consist of two neural networks: an encoder $f(\mathbf{x})$ that acts on the

state $\mathbf{x}$ to produce latent representation $\mathbf{z}$ and a decoder $g(\mathbf{z})$ which creates reconstruction $\hat{\mathbf{x}}$ from $\mathbf{z}$:

$$\begin{aligned} f(\mathbf{x}) &= \mathbf{z} \\ g(\mathbf{z}) &= \hat{\mathbf{x}} \end{aligned} \tag{3.17}$$

where $\mathbf{x}, \hat{\mathbf{x}} \in \mathbb{R}^n$, $\mathbf{z} \in \mathbb{R}^m$, $m < n$.

Typically f and g are jointly trained to minimise the L1 or L2 reconstruction error. The 'information bottleneck' at the latent dimension encourages the networks to find and exploit redundancies in the training data to implicitly model the data distribution.

One possible AE-based approach is to work in the control space, where the variable $\mathbf{w}_k$ is defined, and use an AE trained on the model states to produce a latent space $\boldsymbol{V}_l$ such that:

$$\begin{aligned} \mathbf{V}_l = f(\mathbf{V}) &= \left[f(\mathbf{x}_0^f),\ f(\mathbf{x}_1^f),\ ...,\ f(\mathbf{x}_S^f)\right] \\ \mathbf{V}_l &= \left[\mathbf{z}_0^f,\ \mathbf{z}_1^f,\ ...,\ \mathbf{z}_S^f\right] \in \mathbb{R}^{m \times S} \end{aligned} \tag{3.18}$$

The function we are going to minimise is as in Equation (3.16), with respect to the new variable $\mathbf{w}_l = \mathbf{V}_l^+ \boldsymbol{\delta}\mathbf{x}_k$:

$$J(\mathbf{w}_l) = \frac{1}{2}\alpha \mathbf{w}_l^T \mathbf{w}_l + \frac{1}{2}(\mathbf{H}\mathbf{V}_l\mathbf{w}_l - \mathbf{d})^T \mathbf{R}^{-1}(\mathbf{H}\mathbf{V}_l\mathbf{w}_l - \mathbf{d}). \tag{3.19}$$

The minimisation can be performed via an L-BFGS method, as also implemented in Keras or Tensorflow, [74, 46].

This approach achieves a reduction in the size of $\boldsymbol{B}$ by a factor of $\mathcal{O}(\frac{n^2}{mS})$. In PCA, for truncation at mode ν (see Equation (3.9)), this approach achieves a reduction by a factor of $\mathcal{O}(\frac{n^2}{n\nu}) = \mathcal{O}(\frac{n}{\nu})$.

The reduction in size using an AE in comparison with PCA is accompanied by a reduction in computational complexity and decreases latency but there is also another class of advantages offered by AEs in this case.

There are a number of reasons why the AE approach produces better reconstructions than the PCA approach:

1. PCA on the control space projects data to the reduced space with an affine transformation. In other words, latent features must be created from linear combinations of the inputs. This approach is only optimal for compression when the data is drawn from a Gaussian distribution. The non-linear activation functions in AEs relax this constraint meaning less constrained data distributions can be modelled.

2. The mean of the training data distribution can be stored 'for free' by the decoder leaving space in the latent representation to encode the sample variations. This is the explanation for how such a substantial decrease in the implicit size of $\boldsymbol{B}$ can be achieved. Note that in contrast, the mean is the first mode obtained in eigenanalysis techniques.
3. By design, in PCA, some of the information is intentionally discarded. This is not the case in the AE framework.

Finally, when using AEs, location information is used for compression, meaning that properties such as local smoothness can be utilised to compress the state more efficiently. Validation of this approach is provided in Section 5 for modelling of pollutant dispersion within an urban environment.

4. Data Learning for Noisy Data and Low-Quality Data

DA is a fundamental tool in tackling problems with noisy or low-quality data. Kalman filters are used when dealing with sequential data, and VarDA methods are used when working with a cost function. We will now show that these approaches can be combined with different ML methods to tackle these common constraints when working with real-world data.

4.1. Integration of Convolutional Neural Networks with Kalman Filters

DL has been used to build a module that reduces problems related to noise and data inconsistency. Kalman filters are structured to include some information about the underlying structure of the data and leverage this to make predictions for future time steps and reduce the effect of noisy data. If the data is known to follow a time series, then successive frames give information about the motion that can be helpful to understanding how the time series is evolving. If the data comes from several sources, one data source may be more reliable, but can provide information less frequently; the other might be faster but gives noisier measurements. In this scenario, both sources are CNNs architectures. CNNs feature convolution cells (or pooling layers) and kernels. Convolution kernels process input data and pooling layers simplify it to reduce unnecessary features. CNNs are typically used for image recognition, with an input window sliding along the image, and the convolution layer compressing the detected features. For example, in image recognition the first convolution layer might detect gradients, the second lines, the third shapes, continuing until the scale of particular objects is

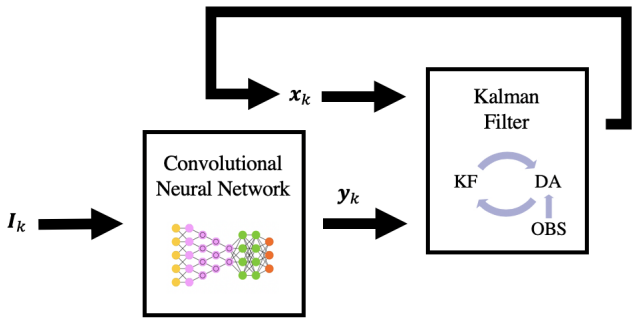


Figure 4: The Convolutional Neural Network (CNN) takes an image as input and extracts joint positions and skeletons. The output from the CNN is used as the measurement for the Kalman filter, which filters out noise and incorrect data.

reached. Often a deep feed-forward network is added to the final convolution layer for further data processing.

If we let $\mathcal{I} = \mathbf{x}_k \in \mathbb{R}^{n \times m}$ be the input state (e. g. an image) and $\mathcal{K} \in \mathbb{R}^{k_r \times k_c}$ be the kernel, and let the rows and columns of the result matrix be denoted by $n_r \in [1, n]$ and $n_c \in [1, m]$ respectively, then we have

$$C(n_r, n_c) = (\mathcal{I} \circ \mathcal{K})(n_r, n_c) = \sum_{i=1}^{k_r} \sum_{j=1}^{k_c} \mathcal{K}(i, j)\mathcal{I}(n_r - i, n_c - j). \tag{4.1}$$

The primary input data are images, which are then run through one of the two CNNs to obtain a noisy measurement $\mathbf{y}_k$. Within this measurement vector, there will be missing data points and inaccurate data points.

To combine the two sources of information a Kalman filter is used, as introduced in Equations (3.1) and (3.2). The aim is also to correct any inconsistencies and fill in missing data points using the predictions from the Kalman filter.

In addition, we want to include a simple model of the expected motion kinematics. Neither a constant velocity or constant acceleration assumption describes the motion we might expect. Thus, to capture the changing acceleration, we can make use of the process noise matrix. We define the state vector to include position and velocity, so that $\mathbf{x} = \left[\begin{smallmatrix} x \\ v \end{smallmatrix}\right]$, and our state equation to be:

$$\hat{x}_{k+1}^{-} = \begin{bmatrix} I & \Delta T \\ \mathbf{0} & I \end{bmatrix} \hat{x}_k + \begin{bmatrix} \frac{1}{2}\Delta T^2 \\ \Delta T \end{bmatrix} \hat{a}_k. \tag{4.2}$$

If we let

$$\mathbf{G} = \begin{bmatrix} \frac{1}{2}\Delta T^2 \\ \Delta T \end{bmatrix} \tag{4.3}$$

we can define the noise term $\mathbf{w}_k \sim \mathcal{N}(0, \mathbf{Q})$ to have a structure that takes into account the acceleration of the system, with

$$\mathbf{Q} = \begin{bmatrix} \frac{1}{4}\Delta T^4 & \frac{1}{2}\Delta T^3 \\ \frac{1}{2}\Delta T^3 & \Delta T^2 \end{bmatrix}. \tag{4.4}$$

In this way, the process covariance matrix of the Kalman filter can be used to include acceleration in the system.

This model has been tested in Section 5.3 on a human pose tracking problem. A modified version using VarDA instead of Kalman Filters can be applied to pharmacokinetic modelling as shown in Section 5.6.

4.2. Integration of Variational Data Assimilation with Neural Networks for Parameter Estimation

If the forecasting model $\mathcal{M}$ is such that its parameters are unreliable, a DL model based on DA for parameter estimation can be applied. Assuming that the structure of the model is such that

$$\mathbf{x}_{k+1} = \mathcal{M}(\mathbf{x}_k, \boldsymbol{\beta}_k, \boldsymbol{\sigma}_k, \boldsymbol{\epsilon}_k, \boldsymbol{\xi}_k) \tag{4.5}$$

where $\mathbf{x}_k$ is a vector of predictors and lagged dependent variables and $\boldsymbol{\beta}_k$ the coefficient vector of corresponding dimension. The errors follow the distributions: $\boldsymbol{\epsilon}_k \sim \mathcal{N}(0, 1)$, $\boldsymbol{\nu}_k \sim \mathcal{N}(0, \mathbf{U}_k)$ and $\boldsymbol{\xi}_k \sim \mathcal{N}(0, \mathbf{V}_k)$. where U_k and V_k denote the parameter error covariance matrices [6]. To obtain statistical optimal parameter values of $\boldsymbol{\beta}$ and $\boldsymbol{\sigma}$ we developed the following data assimilation methodology by solving the corresponding normal equations and maximizing the following custom cost function:

$$J(\boldsymbol{\beta}, \boldsymbol{\sigma}) = \left|\left|\boldsymbol{\beta} - \boldsymbol{\beta}_{k-1} + \boldsymbol{\nu}_k\right|\right|_{\mathbf{U}_k^{-1}} + \left|\left| \log\left(\frac{\boldsymbol{\sigma}_{k-1}^2}{\boldsymbol{\sigma}^2}\right) + \boldsymbol{\xi}_k\right|\right|_{\mathbf{V}_k^{-1}} \tag{4.6}$$

The solution to this leads to a filtering algorithm, corresponding to a modified Kalman filter. In order to minimise the function, we let $\nabla(J(\boldsymbol{\beta}, \boldsymbol{\sigma})) = 0$ and

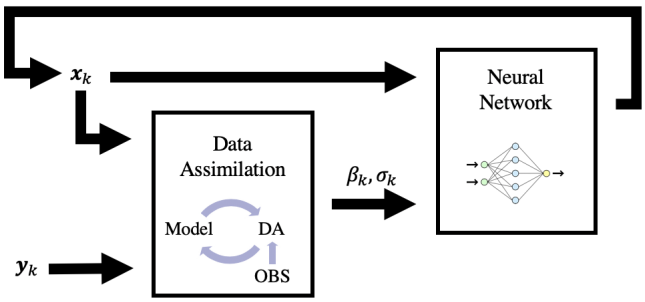


Figure 5: Data Learning based on the integration of Variational Data Assimilation with Neural Networks for Parameter Estimation. In this way it is possible to improve a model with unreliable parameters.

we solve the normal equations which conduct to a modified version of the Kalman Filter:

$$\boldsymbol{\beta}_k = \boldsymbol{\beta}_{k-1} + \mathbf{K}_k(\mathbf{y}_k - \mathbf{x}_k\boldsymbol{\beta}_{k-1}) \tag{4.7}$$

$$\log(\boldsymbol{\sigma}_k^2) = \log(\boldsymbol{\sigma}_{k-1}^2) + \mathbf{K}_k^*(\mathbf{y}_k^* - \log(\boldsymbol{\sigma}_{k-1})^2) \tag{4.8}$$

where

$$\mathbf{K}_k = \mathbf{Q}_{k-1}\mathbf{x}_k'(\mathbf{x}_k\mathbf{Q}_{k-1}\mathbf{x}_k' + \boldsymbol{\sigma}_k^2)^{-1} \tag{4.9}$$

and

$$\mathbf{K}_k^* = \mathbf{R}_{k-1}\mathbf{R}_{k-1}^{-1}, \quad \mathbf{y}_k^* = \log((\mathbf{y}_k - \mathbf{x}_k\boldsymbol{\beta}_{k-1})^2) - \ln(\boldsymbol{\xi}_k^2) \tag{4.10}$$

The first term in (4.6) is a parameter vector that describes the linear relationship between observations, the second term is non-linear volatility term derived from econometric theory, which is linearized in the optimization (see [50] for details). The resulting parameter can be used as input in a network architecture to forecast latent dynamics of the system [39, 49]. An example application of DL for parameter estimation applied to econometric inference is presented in Section 5.4.

4.3. Integration of Gaussian Processes with Variational Data Assimilation

Gaussian Processes are statistical models used to describe observations taken from a continuous domain such as time or space, with the main advantage being the uniformity of the underlying structure.

More formally, a GP is a collection of random variables (a stochastic process) such that any finite subgroup of these random variables has a multivariate normal distribution: i. e. a time-continuous stochastic process $\{\mathbf{X}_k\}$ is Gaussian if and only if for every finite set of indices $k = 1, \ldots, n$ in the index set,

$$\mathbf{X}_{1\ldots n} = (\mathbf{X}_1, \ldots, \mathbf{X}_n) \tag{4.11}$$

is a multivariate Gaussian random variable. The distribution of the GP is the joint distribution of the whole collection of random variables, i. e. it is a distribution over functions with a continuous domain. A key characteristic of Gaussian processes is that they can be completely defined by their covariance function (or kernel).

A GP aims to obtain the predictive distribution for the realisation of a latent function, described by a zero-mean Gaussian random field with covariance function is formulated as follows: $k(\cdot, \cdot)$ on a domain $\Omega \subset \mathbb{R}^{n \times n}$ given a training data set $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^n$ of n pairs of noisy inputs $\mathbf{x}_i$ and noisy observations $\mathbf{y}_i$. The observations $\mathbf{y}_i$ are given by

$$\mathbf{y}_i = f(\mathbf{x}_i + e_{xi}) + e_{yi} \tag{4.12}$$

where $e_{xi} \sim \mathcal{N}(0, \sigma_x^2)$ and $e_{yi} \sim \mathcal{N}(0, \sigma_y^2)$. The joint distribution of $(f_k, \mathbf{y}_k)$ is

$$P(f_k, \mathbf{y}_k) \sim \mathcal{N}\left(0, \begin{bmatrix} \mathcal{K}_{\mathbf{x}_k\mathbf{x}_k} & \mathcal{K}_{\mathbf{x}\mathbf{x}_k}^T \\ \mathcal{K}_{\mathbf{x}\mathbf{x}_k} & \sigma_{\mathbf{y}_k}^2 \mathbf{I} + \mathcal{K}_{\mathbf{x}\mathbf{x}} \end{bmatrix}\right) \tag{4.13}$$

where

$$\mathcal{K}_{\mathbf{x}\mathbf{x}} = \{k(\mathbf{x}_i, \mathbf{x}_j)\}_{i,j=1\ldots n} \in \mathbb{R}^{n \times n}, \tag{4.14}$$

and

$$\mathcal{K}_{\mathbf{x}\mathbf{x}_k} = (\mathcal{K}(\mathbf{x}_1, \mathbf{x}_k) \ldots \mathcal{K}(\mathbf{x}_n, \mathbf{x}_k))^T. \tag{4.15}$$

Given the conditional distribution of Gaussian variables, the predictive distribution of f_k given $\mathbf{x}_k$ is

$$P(f_k|\mathbf{y}_k) \sim \mathcal{N}\left(\mathcal{K}_{\mathbf{x}\mathbf{x}_k}^T \mathbf{A}^{-1} \mathbf{y}_k, \mathcal{K}_{\mathbf{x}_k\mathbf{x}_k} - \mathcal{K}_{\mathbf{x}\mathbf{x}_k}^T \mathbf{A}^{-1} \mathcal{K}_{\mathbf{x}\mathbf{x}_k}\right) \tag{4.16}$$

where

$$A = \sigma_{y_k}^2 \mathbf{I} + \mathcal{K}_{\mathbf{x}\mathbf{x}} \tag{4.17}$$

In some applications, we may have several sources of observations that need to be combined in an optimal way. At each step k, let $\mathbf{o}_k = \mathcal{H}(\mathbf{x}_k)$ be a vector of observations in addition to $\mathbf{y}_k$, where $\mathcal{H}$ is a non-linear operator collecting the assimilated observations at each step. The aim of the DA used here is to find an optimal trade-off between the current estimate of the system state (background) and the available assimilated observations. Let $\mathbf{R}$

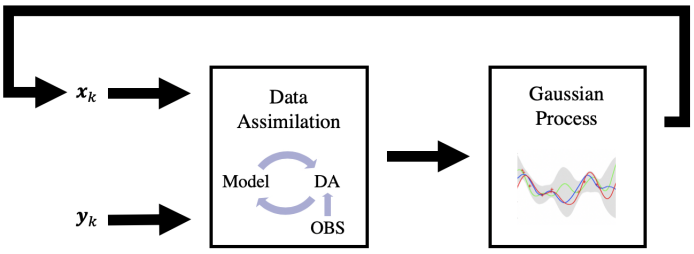


Figure 6: Data Learning for Data Selection problems based on the integration of Gaussian Processes with Variational Data Assimilation.

be a covariance matrix whose elements provide an estimate of the errors[3] on $\mathbf{o}_k$. The assumption and assimilation of input noise by a data assimilation approach (see [6]) introduces a corrective term

$$\mathbf{O} = \mathbf{H}^T \mathbf{R}^{-1} \mathbf{H} \tag{4.18}$$

to the output noise. Thus the resulting Gaussian process becomes

$$P(f_k|\mathbf{y}_k, \mathbf{o}_k) = \mathcal{N}\left(\mathcal{K}_{\mathbf{x}\mathbf{x}_*}^T \hat{\mathbf{A}}^{-1}\mathbf{y}, \mathcal{K}_{\mathbf{x}_*\mathbf{x}_*} - \mathcal{K}_{\mathbf{x}\mathbf{x}_*}^T \hat{\mathbf{A}}^{-1}\mathcal{K}_{\mathbf{x}\mathbf{x}_*}\right) \tag{4.19}$$

where

$$\hat{\mathbf{A}} = \mathbf{A} + \mathbf{O} \tag{4.20}$$

with $\mathbf{A}$ and $\mathbf{O}$ defined in (4.17) and (4.18) respectively, and where we assume that each input dimension is independently corrupted by noise, thus $\mathbf{R}$ is diagonal:

$$\mathbf{R} = \sigma_{\mathbf{x}}^2 \mathbf{I} \tag{4.21}$$

[3] i.e., the assimilated observations are not verified exactly, they are a weak constraint over the inputs to the Gaussian Process

The predictive mean $\mathcal{K}_{\mathbf{x}\mathbf{x}_k}^T \hat{\mathbf{A}}^{-1}\mathbf{y}_k$ gives the point prediction of $f_k(\mathbf{x}_k)$ at location $\mathbf{x}_k$, whose uncertainty is measured by the predictive variance

$$\mathcal{K}_{\mathbf{x}_k\mathbf{x}_k} - \mathcal{K}_{\mathbf{x}\mathbf{x}_k}^T \hat{\mathbf{A}}^{-1}\mathcal{K}_{\mathbf{x}\mathbf{x}_k}. \tag{4.22}$$

The point prediction given above is the Best Linear Unbiased Estimator (BLUE) in the following sense: consider all linear predictors

$$\mu(\mathbf{x}_k) = u(\mathbf{x}_k)^T \mathbf{y}_k, \tag{4.23}$$

satisfying the unbiasedness requirement $E[\mu(\mathbf{x}_k)] = 0$. We want to find the vector $u(\mathbf{x}_k)$ which minimises the mean squared prediction error $E[\mu(\mathbf{x}_k) - f_k(\mathbf{x}_k)]^2$. Since $E[\mu(\mathbf{x}_k)] = 0$ and $E[f_k(\mathbf{x}_k)] = 0$, the mean squared prediction error equals the error variance $var[\mu(\mathbf{x}_k) - f(\mathbf{x}_k)]$ and can be expressed as

$$\begin{aligned}\sigma(\mathbf{x}_k) &= u(\mathbf{x}_k)^T E[\mathbf{y}_k \mathbf{y}_k^T] u(\mathbf{x}_k) - 2u(\mathbf{x}_k)^T E[\mathbf{y}_k f_k(\mathbf{x}_k)] + E[f_k(\mathbf{x}_k)^2] \\ &= u(\mathbf{x}_k)^T (\sigma_y^2 \mathbf{I} + \mathcal{K}_{xx} + \mathbf{H}^T \mathbf{R}^{-1} \mathbf{H}) u(\mathbf{x}_k) - 2u(\mathbf{x}_k)^T \mathcal{K}_{\mathbf{x}\mathbf{x}_k} + \mathcal{K}_{\mathbf{x}_k\mathbf{x}_k}\end{aligned} \tag{4.24}$$

This model as been tested in Section 5.5 for an application in optimal sensor placement.

5. Testing

We will now present some results obtained using our DL methods. Note that the algorithms and numerical methods proposed in this work can also be applied to other physical, engineering, economic or medical problems involving other equations or state variables: our focus on particular applications does not affect the generality of our studies as the underlying equations of the systems are identical. In fact, we know that DL can be applied in any of the fields shown in Figure 7.

The accuracy of our results is evaluated by the Mean Squared Error (MSE) on each sub-domain:

$$\text{MSE}(\mathbf{x}_k) = \frac{\|\mathbf{x}_k - \mathbf{x}_k^C\|_{L^2}^2}{N} \tag{5.1}$$

computed with respect to a control variable $\mathbf{x}_k^C$ where N is the dimension of the state vector.

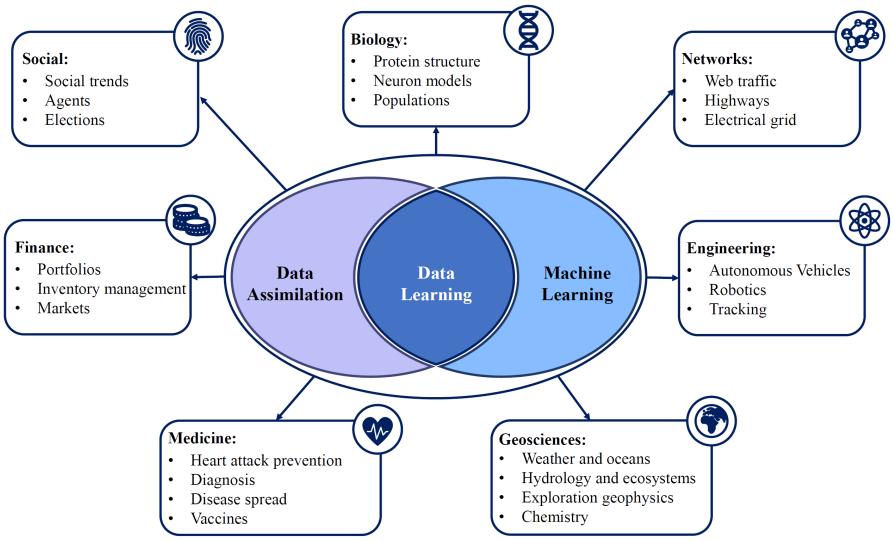


Figure 7: We can use Data Learning in any field where observations and models coexist, and methods are already being applied in several areas.

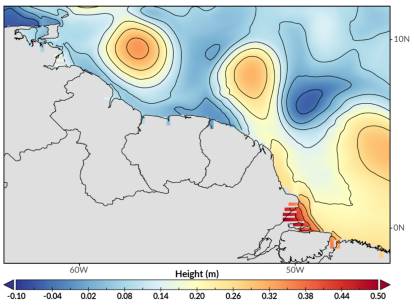
5.1. Data Learning for Dimensionality Constraints and Noisy Data: An application to oceanography

In this section we show an application of the method presented in Section 3.1, which combines DA, PCA and NNs to improve accuracy and efficiency of ocean forecasting [54]. Fast and reliable ocean assimilation methods are useful in multiple applications, such as emergency scenarios, search and rescue, and oil spills. Speed improvement is achieved by performing DA on a reduced space. A surface 10km resolution hindcast of the North Brazil current from the Regional Ocean Modelling System (ROMS) serves as the full-space state. The target variables are sea surface height, sea surface temperature, and surface currents.

Experiments use data and forecasts from July 2015 and January 2016. The analysis and forecasts in the presented framework yield a higher skill score and higher spatial correlation when compared to the operational dataset Global Ocean Physics Analysis and Forecast by the UK Met Office. The reduced model approach could be a useful tool when full physics regional models are not available to make a forecast.

In this study, observations of Sea Surface Height (SSH) and Sea Surface Temperature (SST) are assimilated together, with further validation using data of SSH, SST, zonal (U), and meridional (V) components of surface

(a) SSH in FS



(b) SSH in RFS

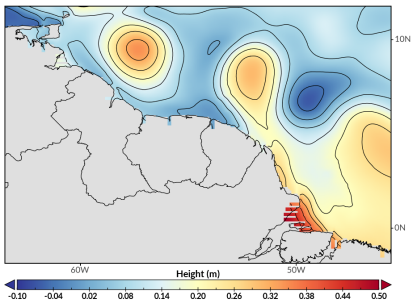


Figure 8: Full-state (FS) and reconstructed full-state (RFS) models. Snapshot of December 1st, 2009. This day shows the genesis and advection of the North Brazil Current rings travelling north-west and the retroflection of the NBC to supply the North Equatorial Counter Current

currents.

SSH (Figure 8a and 8b) gives a clear representation of the North Brazil Current (NBC) rings. The main features of the NBC rings can be observed in both model solutions, thus the mesoscale features are well represented in both. Values of annual averaged normalised root squared mean errors (Equation (5.1)) and execution time for different ocean variables before and after data learning are shown in Table 1. Only SSH and SST were assimilated. The initial size of the domain is $n = 120.000$ grid points and the number of PCs defining the reduced data set in Equation (3.10), computed by Equation (3.9), is $\nu = 100$. In Skitka et al. [62], for less than 0.2% of the modes retained, the authors implement a reduced quasilinear model which able to reproduce vertical profiles of horizontal mean fields as well as certain energetically important second-order turbulent transport statistics and energies to within 30% error. In our case, we use only the 0.08% keeping the 93% of variance, which means within 7% error. In terms of execution time, the model takes 10 seconds to perform a 50 ensemble KF with 30K observations (two variables) onto a 60K vector (four variables). The full model on one processor takes 30.52 *hours* to produce a 1-day forecast. This DL model takes approximately 0.1 *seconds* on a single processor. Considering that the full model produces 50 levels and the DL model produces only the surface level, there is a linear gain in speed of 21960 times (assuming a linear relationship between levels and processing time). For the DA, there is a speedup of 11000 times (considering 50 ensembles). For a daily time-step basis, the fast

DA framework presented here outperforms the fullspace ROMS run of this region by 3 to 4 orders of magnitude in computational efficiency. In terms of computing time, [29] reports a speed-up of two orders of magnitude when comparing the DA on the full-space estuary model of the Columbia River to the one performed on the model surrogate. In a similar study using only DA in the full-space, [23] reported RMSE of SSH and SST validation datasets to range from 0.05 to 0.15 m, and 0.5 to 3.5 °C. The RMSE results of the assimilation of altimetry and temperature are similar to the ones presented by [29] for an estuary: between 0.04 to 0.07 m in SSH and 0.4 to 1.4 °C for SST. Our fast DA framework has a similar magnitude of errors compared to these full-space systems after performing DA.

Ocean Variable	MSE		Execution time	
	Before	After	Before	After
SSH	1.034	0.575	30.52 hours	0.1 secs
SST	0.024	0.012		
U	0.954	0.813		
V	1.168	0.998		

Table 1: *An application to oceanography* - Annual averaged normalised root squared mean errors and execution time for different ocean variables before and after data learning. Only SSH and SST were assimilated.

5.2. Data Learning for Dimensionality Constraints and Noisy Data: An application to air pollution prediction

In this section, we apply the DL model presented in Section 3.2. Experimental results are provided for pollutant dispersion within an urban environment. The model has been tested on an urban environment located near London South Bank University (LSBU), UK [3]. The computational domain has a size of $[0, 2041] \times [0, 2288] \times [0, 250]$ metres. In this work, the forecasting model $\mathcal{M}$ is based on the 3D non-hydrostatic Navier-Stokes equations,

$$\nabla \cdot \mathbf{x} = 0, \tag{5.2}$$

$$\frac{\partial \mathbf{x}}{\partial t} + \mathbf{x} \cdot \nabla \mathbf{x} = -\nabla p + \nabla \cdot \tau, \tag{5.3}$$

where $\mathbf{x} \equiv (x, y, z)^T$ is the velocity, $p = \tilde{p}/\rho_0$ is the normalised pressure ($\tilde{p}$ is pressure and ρ_0 is the constant reference density) and τ denotes the stress tensor.

The dispersion of pollutant is described by the classic advection-diffusion equation, with pollutant concentration treated as a passive scalar. A source term was added to the advection-diffusion equation to mimic a constant release of pollutant generated by traffic in a busy intersection (Figure 9). Table 2 shows a comparison of the MSE (Equation (5.1)) and execution time

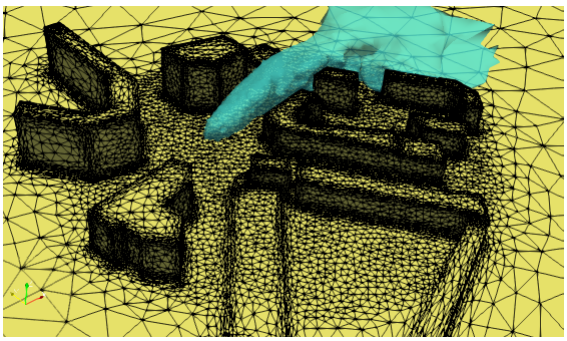


Figure 9: Isosurface of the pollution generated by traffic in a busy intersection. The domain is the London South Bank University, London, UK.

of our Data Learning model, where we used the Tucodec-NeXt AE with $\nu = 32$ PCs in equation (3.9). Here m denotes the number of assimilated time steps considered with respect to the number of grid points n. The Data Learning MSE is 37% lower than the state of the art of the system in [3].

Further research will explore whether the improved performance of the bi-reduced space approach is explained by poor conditioning in the mono-reduced space and the resulting numerical errors.

Model	MSE	Execution Time (s)
Ref MSE	1.0001	-
PCA, $\nu = 32$, $m = n$	0.1270	1.8597
PCA, $\nu = 32$, $m = 0.1n$	0.1270	0.2627
PCA, $\nu = 32$, $m = 0.01n$	0.1334	0.0443
PCA, $\nu = 32$, $m = 0.001n$	0.1680	0.0390
Data Learning with Tucodec-NeXt	0.0787	0.0537

Table 2: *An application to air pollution prediction* - Comparison of MSE and Execution time of our DataLearning model where we used the Autoencoder named Tucodec-NeXt with optimal PCA setting the truncation parameter $\nu = 32$ [3]. m denotes the number of assimilated time steps we consider with respect to the number of grid points n.

5.3. Data Learning for Noisy Data and Low-Quality Data: An application to human motion tracking

Here we show results for the method presented in Section 4.1. The module is tested on a dataset with ground truth joint annotations in real-world images in a range of lighting conditions and situations: Posetrack 2018 [1]. The evaluation is based on the MSE as defined in Equation (5.1), but with

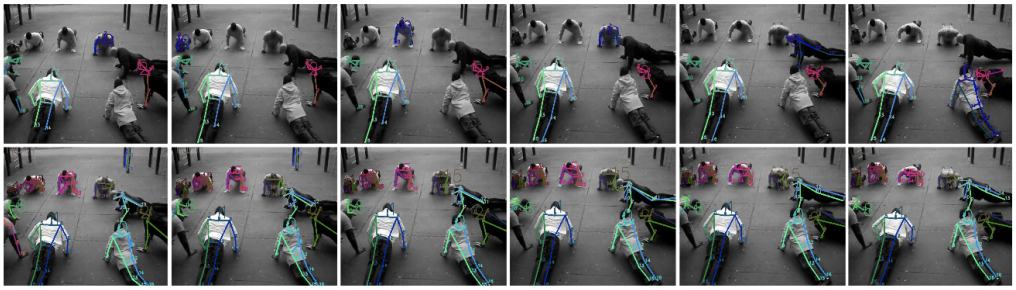


Figure 10: Difference between unfiltered (top) and filtered (bottom) sequence. The filtering, introduced by the Data Learning approach, clearly allows more joints to be tracked throughout several frames, allowing the poses to reach the required number of keypoints to be considered valid.

added metrics. The error estimation should take into account whether 1) the number of people in the scene is correctly identified, 2) the identification of these people is consistent, and 3) all 25 joints have been identified for each person. These are compared with the ground truth labels of the images. Thus, MSE (Equation (5.1)) is replaced by Multiple Object Tracking Accuracy (MOTA), defined as:

$$\text{MOTA} = 1 - \frac{\sum_k (m_k + fp_k + mme_k)}{\sum_k g_k} \tag{5.4}$$

where m_k, fp_k and mme_k are the number of misses, false positives and of mismatches respectively for frame k, and g_k is the total number of objects present in the frame. Table 3 shows PoseTrack2018 Multiple Object Tracking metrics and average computing time in frames per second (FPS) with an increasing number of tracking frames N for our Data Learning method, described in Section 4.1, compared to the original algorithms.

The performance is also tested the number of frames N over which we track the poses is increased.

Model	MOTA Head	MOTA Shoul.	MOTA Elbow	MOTA Wrist	MOTA Hip	MOTA Knee	MOTA Ankle	MOTA Total	Speed FPS
Original Algorithm									
Openpose	-77.9	6.2	-26.4	-53.8	-10.1	-34.3	-62.3	-39.7	25.5
STAF	39.5	67.7	54.4	44.4	55.9	50.0	42.3	49.8	15.5
LightTrack	67.7	72.6	67.3	57.8	63.5	63.8	57.7	64.6	8.5
Data Learning: Tracking 5 Frames									
Openpose	23.2	20.5	19.5	15.7	24.8	22.6	21.7	21.2	42.5
STAF	34.6	55.8	46.1	35.9	56.2	46.6	46.0	45.8	36.5
LightTrack	56.9	57.7	57.5	56.9	58.2	57.8	56.9	57.4	25.0
Data Learning: Tracking 10 Frames									
Openpose	20.6	18.8	19.3	15.9	22.2	20.4	20.9	19.8	45.0
STAF	35.6	51.9	43.2	32.7	54.0	44.4	47.0	44.1	41.0
LightTrack	52.5	54.4	49.6	48.1	53.7	51.4	48.7	51.3	33.5

Table 3: *An application to human motion tracking* - PoseTrack2018 Multiple Object Tracking metrics and average computing time in frames per second (FPS) with increasing number of tracking frames N for our Data Learning method (described in Section 4.1) compared to the original algorithms.

The module is run alongside the pose estimation methods OpenPose [19], LightTrack [52] and STAF [56]. LightTrack is currently the best performing publicly available system on the PoseTrack 2018 leaderboard. Figure 10 shows the difference between unfiltered (top) and filtered (bottom) sequence. The filtering, introduced by the Data Learning approach, clearly allows more joints to be tracked throughout several frames, allowing the poses to reach the required number of key points to be considered valid.

5.4. Data Learning for Noisy Data and Low-Quality Data: An application to econometric inference

In this section, we implement the Data Learning model presented in Section 4.2 and apply it to an econometric problem. Both econometrics and DA aim to incorporate new observations into a numerical model, in fact, DA can be applied for the estimation of latent model parameters [63]. The combination of the two approaches yields a novel method for economic analysis in a digitised economy. Other contributions to econometric analysis are the derivation of a cost function which can be solved using both statistical and variational approaches. The class of economic models considered are Time-

Varying Parameter Vector Autoregressive Models (TVP-VARs). TVP-VARs are a class of state-space models used for the forecasting of economic time series. They are also used to study the effects of economic policy by analysing the variables of the latent state equation [18, 50, 39]. The structure of the forecasting model is given by:

$$\mathbf{y}_k = \mathbf{x}_k \boldsymbol{\beta}_k + \boldsymbol{\epsilon}_k \boldsymbol{\sigma}_k \tag{5.5}$$

$$\boldsymbol{\beta}_k = \boldsymbol{\beta}_{k-1} + \boldsymbol{\nu}_k \tag{5.6}$$

$$\log(\boldsymbol{\sigma}_k^2) = \log(\boldsymbol{\sigma}_{k-1}^2) + \boldsymbol{\xi}_k \tag{5.7}$$

where scalar $\mathbf{y}_k$ is the value of the dependent variable at k, $\mathbf{x}_k$ is a $1 \times q$ vector of predictors and lagged dependent variables and $\boldsymbol{\beta}_k$ the coefficient vector of corresponding dimension. The errors follow the distributions: $\boldsymbol{\epsilon}_k \sim \mathcal{N}(0, 1)$, $\boldsymbol{\nu}_k \sim \mathcal{N}(0, \mathbf{U}_k)$ and $\boldsymbol{\xi}_k \sim \mathcal{N}(0, \mathbf{V}_k)$. where $\mathbf{U}_k$ and $\mathbf{V}_k$ denote the parameters error covariance matrices.

Equation (5.5) describes the relationship of economic variables with other series of interest. The latent state over time is given by Equation (5.6), whereas Equation (5.7) models changes in the volatility of variables over time. To obtain statistical optimal parameter values of $\boldsymbol{\beta}$ and $\boldsymbol{\sigma}$ we use solve the corresponding normal equations and maximise the custom cost function in (4.6). This leads to a filtering algorithm, corresponding to a modified Kalman filter. Equations (5.5) to (5.7) can thus be used to analyse how the inflow of assets into smart contracts or other entities on a blockchain describe, for example, changes in trading volume or usage of a decentralised application. Results are provided for several bitcoins in Figure 11 and shown in Table 4 where the MSE defined in Equation (5.1) is computed before and after the assimilation process. In terms of execution time, the code runs in 10.87×10^2 seconds for a data set of 65600 nodes and scales to 0.99×10^0 seconds for a data set of 1100 nodes on 32 cores.

5.5. Data Learning for Noisy Data and Low-Quality Data: An application to optimal sensor placement

Finally, we present results for the method discussed in Section 4.3. The algorithm developed for optimal sensors placement is based on the work in [40] and essentially minimises the joint entropy as defined in [61]. As this is a combinatorial optimisation problem that is not feasible, the authors in [40] suggest to greedily place sensors at the point of highest mutual information, and prove that this is a constant approximation of the optimal solution.

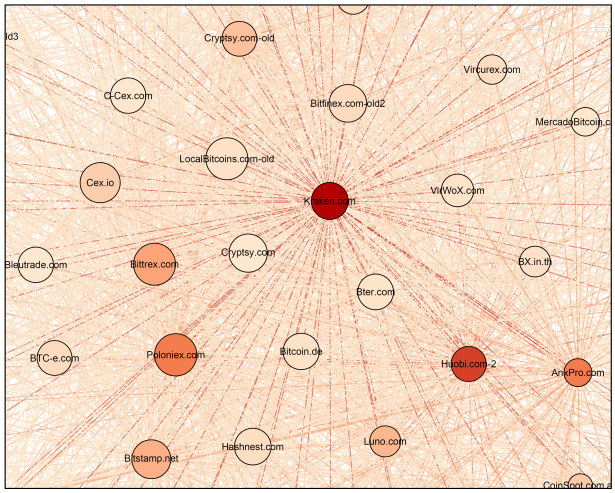


Figure 11: The bitcoin ecosystem: aggregated bitcoins addresses belonging to the exchanges. Detailed graph representing on-chain Bitcoin flows of exchanges.

MSE	Balance			Returns			Volume		
	Bef.	Aft.	Ratio	Bef.	Aft.	Ratio	Bef.	Aft.	Ratio
Kraken	1.40	$6.00e^{-6}$	$4.28e^{-6}$	1.98	$6.90e^{-6}$	$3.48e^{-6}$	0.71	$1.28e^{-5}$	$1.80e^{-5}$
Poloniex	8.55	$3.51e^{-5}$	$4.10e^{-6}$	1.75	$1.39e^{-6}$	$7.93e^{-7}$	0.31	$3.57e^{-5}$	$1.14e^{-4}$
Bittrex	12.66	$4.72e^{-5}$	$3.73e^{-6}$	1.76	$2.08e^{-6}$	$1.18e^{-6}$	1.20	$4.55e^{-5}$	$3.80e^{-5}$
HitBtc	-	-	-	2.11	$6.47e^{-6}$	$3.07e^{-6}$	0.15	$2.79e^{-5}$	$1.87e^{-4}$
Overall	Before			After			Ratio		
Mean	2.962			$2.064e^{-5}$			$6.967e^{-6}$		

Table 4: *An application to econometric inference* - Mean squared errors before and after data learning, along with corresponding ratios for four different Bitcoin types.

The method is tested for outdoor sensor placement at a test-site comprising 148 buildings around London South Bank University (LSBU), UK.

First, a Gaussian process trained on the data from our selection of sensors is used to predict pollution values at each node; these are then compared to the real pollution values. Second, sensor placement performance when used for data assimilation is validated. In both instances, performance is compared to the mean performance of 100 sets of randomly placed sensors.

Results presented in Table 5 for both validation methodologies show that the Data Learning model, presented in Section 4.3, applied here for a sensor placement problem, performs much better than random placement. The

optimal placements for our model (WCGP) and the model with random assignment (GP) are shown in Figure 12. In terms of execution time, the code runs in 225 *minutes*, compared to 2989 *minutes* for the standard code.

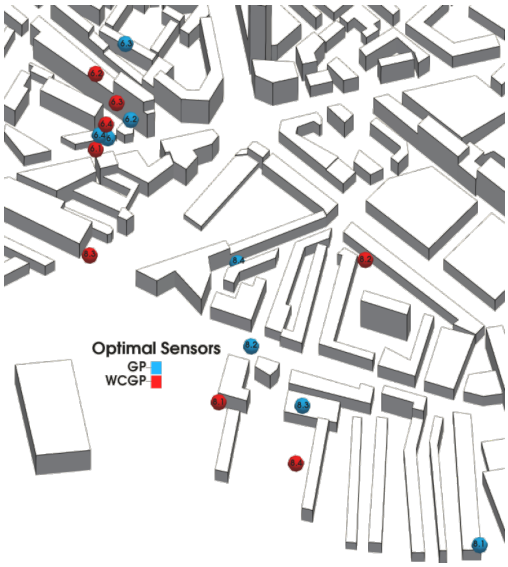


Figure 12: Optimal sensor placements when using a traditional model (GP) approach (blue spheres) or our Data Learning (WCGP) approach (red spheres).

	Real Mean	Estimated Mean	MSE($\mathbf{x}^{\mathcal{M}}$)	MSE($\mathbf{x}^{\mathrm{DA}}$)
Original Algorithm	2.4662e-01	1.9598e-01	2.24e-01	5.25e-02
Data Learning (GP+DA)	2.4662e-01	2.2771e-01	1.77e-01	3.35e-02
Random	2.4662e-01	2.3900e02	6.54e00	8.90e-01

Table 5: *An application to optimal sensor placement* - Estimated and real pollution concentration means and MSE of the predicted model before (MSE($\mathbf{x}^{\mathcal{M}}$)) and after (MSE($\mathbf{x}^{\mathcal{M}}$) data learning. For 'random', the average over 100 random placements was taken.

5.6. Data Learning for Noisy Data and Low-Quality Data: An application to pharmacokinetic modelling

Here we show results for the method presented in Section 4.1, which uses VarDA instead of a KF. Applications in medical domains require the extraction of valuable information from large amounts of data. In medical imaging,

Dynamic Contrast Enhanced-Magnetic Resonance Imaging (DCE-MRI) is one methodology whose analysis is particularly time-consuming (Figure 13).

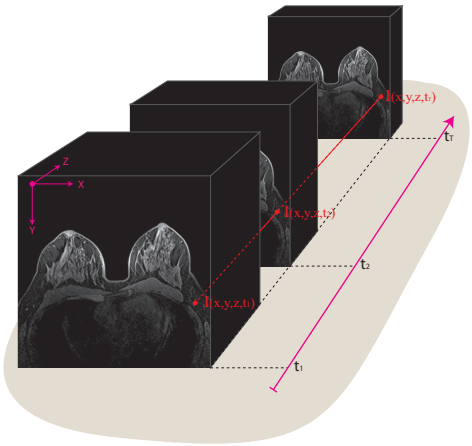


Figure 13: Illustrative example of a breast DCE-MRI, showing the 3 spatial and the temporal dimensions.

In this instance, the forecasting model $\mathcal{M}$ is taken from [66] and uses a compartmental approach to represent blood plasma and the extra-vascular extracellular space. The contrast agent concentration in each voxel $C_v(t)$ is calculated as the convolution of an exponential impulsive response and the Parker arterial input function $C_p(t)$ as

$$\begin{aligned} C_v(t) &= K^{\text{trans}} \int_0^t \mathrm{e}^{-\frac{K^{trans}}{v_e}(t-s)} \cdot C_p(s)\,\mathrm{d}s + v_p C_p(t) \\ C_p(t) &= \sum_{i=1}^{2} \frac{A_i}{\sigma_i\sqrt{2\pi}} \mathrm{e}^{\frac{-(t-T_i)^2}{2\sigma_i^2}} + \alpha \frac{\mathrm{e}^{-\beta t}}{1+\mathrm{e}^{-s(t-\tau)}} \end{aligned} \tag{5.8}$$

where K^{trans} is the tissue permeability; v_e is the extravascular extracellular space volume per unit volume of tissue and v_p is the plasma volume fraction.

Fitting real data points to such a model is very computationally demanding and involves curve fitting procedures. However, as seen in section 3.2, DL can be used to deal with high-dimensional data by leveraging actual measurements to compensate for erroneous data.

In the proposed method [47], the authors speedup the curve fitting evaluation by means of a Look-up Table (LuT) approach. The results are compared to those obtained using the Levenberg-Marquardt algorithm (LMA).

The comparison shows that the proposed approach and LMA are not statistically different in terms of MSE (with respect to the observed data) and in terms of the sum of differences in the parameters and in the solution space.

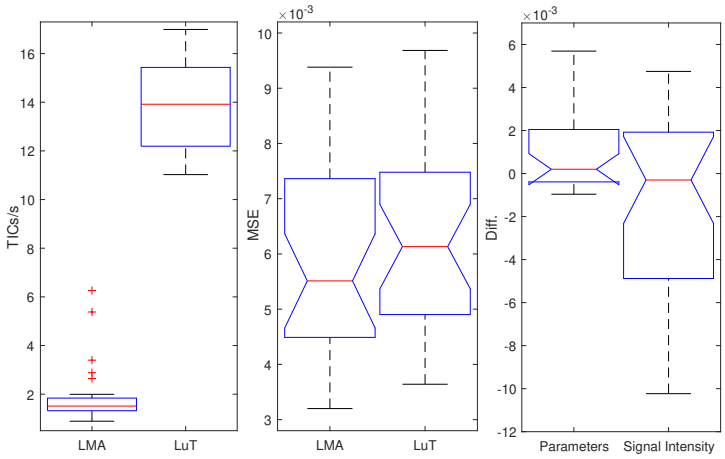


Figure 14: *An application to pharmacokinetic modelling* - LuT performance evaluation. On the left: Performance comparison in terms of throughput (TIC/s). In the middle: MSE comparison between LuT and LMA. On the right: Sum of Differences in the parameters and curve spaces between LuT and LMA. With the aim of highlighting the medians and the corresponding interquartile ranges, the MSE axis has been accordingly centred, causing some outliers to fall out of the plotted interval.

Results show that the proposed approach can be effectively used without affecting the reliability of the fitting results. This is shown in Figure 14, where the box plots of the MSE evaluated between the fitted points and the original DCE-MRI data almost overlap. Results also show that the best fitting parameters obtained by applying the LMA are not statistically different from those obtained by applying the proposed LuT approach. The same can be stated for the fitted values. These assertions can be derived from Figure 14 (on the right), showing that the 0 value is contained within the 95% confidence interval for the sum of differences both in the parameters and in the solutions space (between LuT and LMA). In conclusion, by exploiting Data Learning it is possible to optimize the curve fitting procedure. The resulting method is an order of magnitude less computationally expensive than the LMA on the same dataset, with a fitting error that is not statistically different from that obtained by means of a classical iterative procedure.

6. Conclusion and Future work

We have presented an introduction to Data Learning and how Data Assimilation and Machine learning methods can be integrated successfully. We have introduced a number of methods that integrate ML and DA effectively to solve some of each field's limitations. We have presented the equations for these methods and have shown results for some test cases, though the equations are general and could be applied to other areas as well. We have given the equations for our Data Learning methods and discussed when the different methods can be applied and how to best leverage their strengths. In particular, we have shown how Data Learning can be used to resolve dimensionality constraints and issues with noisy or low-quality data. At the moment, we are increasing the number of methods that rely on this integration in a variety of different applications and we are studying the impact of Data Learning on the accuracy and efficiency with respect to standard Machine Learning and Data Assimilation methods.

Acknowledgements

This work is supported by the EPSRC grant EP/T003189/1 Health assessment across biological length scales for personal pollution exposure and its mitigation (INHALE), the grant EP/N010221/1 Managing Air for Green Inner Cities (MAGIC) consortium, the PREMIERE programme grant (EP/T000414/1) and the RELIANT grant (EP/V036777/1)

References

[1] Andriluka, M., Iqbal, U., Ensafutdinov, E., Pishchulin, L., Milan, A., Gall, J., B., S., 2018. PoseTrack: A benchmark for human pose estimation and tracking, in: IEEE Conference on Computer Vision and Pattern Recognition, pp. 5167–5176.

[2] Arcucci, R., D'Amore, L., Pistoia, J., Toumi, R., Murli, A., 2017. On the variational data assimilation problem solving and sensitivity analysis. Journal of Computational Physics 335, 311–326. doi:10.1016/j.jcp.2017.01.034.

[3] Arcucci, R., Mottet, L., Pain, C., Guo, Y.K., 2019. Optimal reduced space for variational data assimilation. Journal of Computational Physics 379, 51–69. doi:10.1016/j.jcp.2018.10.042.

[4] Arcucci, R., Moutiq, L., Guo, Y.K., 2020. Neural assimilation, in: International Conference on Computational Science, Springer. pp. 155–168.

[5] Arcucci, R., Zhu, J., Hu, S., Guo, Y.K., 2021. Deep data assimilation: integrating deep learning with data assimilation. Applied Sciences 11, 1114.

[6] Asch, M., Bocquet, M., Nodet, M., 2016. Data assimilation: methods, algorithms, and applications. Society for Industrial and Applied Mathematics, Philadelphia. doi:10.1137/1.9781611974546.

[7] Bishop, C.M., 2006. Pattern recognition and machine learning. springer.

[8] Bocquet, M., Brajard, J., Carrassi, A., Bertino, L., 2020. Bayesian inference of chaotic dynamics by merging data assimilation, machine learning and expectation-maximization. Foundations of Data Science 2, 55–80. doi:10.3934/fods.2020004.

[9] Bonavita, M., Arcucci, R., Carrassi, A., Dueben, P., Geer, A., Le Saux, B., Longepe, N., Mathieu, P.P., Raynaud, L., 2020. Machine learning for earth system observation and prediction. Bulletin of the American Meteorological Society (BAMS) , 1–13doi:https://doi.org/10.1175/BAMS-D-20-0307.1.

[10] Bonavita, M., Massart, S., Laloyaux, P., Chrust, M., 2014. Data assimilation and machine learning science at ecmwf .

[11] Boukabara, S.A., Krasnopolsky, V., Stewart, J.Q., Maddy, E.S., Shahroudi, N., Hoffman, R.N., 2019. Leveraging modern artificial intelligence for remote sensing and nwp: Benefits and challenges. Bulletin of the American Meteorological Society .

[12] Brajard, J., Carrassi, A., Bocquet, M., Bertino, L., 2019. Combining data assimilation and machine learning to emulate a dynamical model from sparse and noisy observations: a case study with the lorenz 96 model. Geoscientific Model Development Discussions , 1–21doi:10.5194/gmd-2019-136.

[13] Brajard, J., Carrassi, A., Bocquet, M., Bertino, L., 2020. Combining data assimilation and machine learning to infer unresolved scale parametrisation. arXiv preprint arXiv:2009.04318 .

[14] Buizza, C., Fischer, T., Demiris, Y., 2020. Real-time multi-person pose tracking using data assimilation, in: IEEE Winter Conference on Applications of Computer Vision, pp. 1–8.

[15] Cacuci, D.G., Ionescu-Bujor, M., Navon, I.M., 2005. Sensitivity and uncertainty analysis, volume II: applications to large-scale systems. volume 2. CRC press.

[16] Canchumuni, S.A., Emerick, A.A., Pacheco, M.A., et al., 2017. Integration of ensemble data assimilation and deep learning for history matching facies models, in: Offshore Technology Conference, Offshore Technology Conference. pp. 1–13. doi:10.4043/28015-MS.

[17] Canchumuni, S.W., Emerick, A.A., Pacheco, M.A.C., 2019. Towards a robust parameterization for conditioning facies models using deep variational autoencoders and ensemble smoother. Computers & Geosciences 128, 87–102. doi:10.1016/j.cageo.2019.04.006.

[18] Canova, F., Gambetti, L., 2009. Structural changes in the us economy: Is there a role for monetary policy? Journal of Economic dynamics and control 33, 477–490.

[19] Cao, Z., Simon, T., Wei, S.E., Sheikh, Y., 2017. Realtime multi-person 2d pose estimation using part affinity fields, in: IEEE Conference on Computer Vision and Pattern Recognition, pp. 7291–7299.

[20] Chao, G., Luo, Y., Ding, W., 2019. Recent advances in supervised dimension reduction: A survey. Machine learning and knowledge extraction 1, 341–358.

[21] Chen, R.T., Rubanova, Y., Bettencourt, J., Duvenaud, D.K., 2018. Neural ordinary differential equations, in: Advances in neural information processing systems, pp. 6571–6583.

[22] Coskun, H., Achilles, F., Dipietro, R., Navab, N., Tombari, F., 2017. Long short-term memory kalman filters: Recurrent neural estimators for pose regularization, in: IEEE International Conference on Computer Vision, pp. 5525–5533. doi:10.1109/ICCV.2017.589, arXiv:arXiv:1708.01885v1.

[23] Costa, FB and Tanajura, CAS, 2015. Assimilation of sea-level anomalies and Argo data into HYCOM and its impact on the 24 hour forecasts in the western tropical and South Atlantic. Journal of Operational Oceanography 8, 52–62.

[24] Courtier, J., 1994. A strategy for operational implementation of 4d-var, using an incremental approach. Q J R Meteorol Soc 120, 1367–1387.

[25] Düben, P., Modigliani, U., Geer, A., Siemen, S., Pappenberger, F., Bauer, P., Brown, A., Palkovic, M., Raoult, B., Wedi, N., et al., 2021. Machine learning

825 at ecmwf: A roadmap for the next 10 years. ECMWF Technical Memoranda
826 .

827 [26] Dueben, P.D., Bauer, P., 2018. Challenges and design choices for global
828 weather and climate models based on machine learning. Geoscientific Model
829 Development 11, 3999–4009.

830 [27] Emerick, A.A., 2018. Deterministic ensemble smoother with multiple data as-
831 similation as an alternative for history-matching seismic data. Computational
832 Geosciences 22, 1175–1186. doi:10.1007/s10596-018-9745-5.

833 [28] Fei-Fei, L., Fergus, R., Perona, P., 2006. One-shot learning of object cate-
834 gories. IEEE transactions on pattern analysis and machine intelligence 28,
835 594–611.

836 [29] Frolov, S., Baptista, A.M., Leen, T.K., Lu, Z., van der Merwe, R., 2009. Fast
837 data assimilation using a nonlinear Kalman filter and a model surrogate: An
838 application to the Columbia River estuary. Dynamics of Atmospheres and
839 Oceans 48, 16–45.

840 [30] Geer, A., 2021. Learning earth system models from observations: machine
841 learning or data assimilation? Philosophical Transactions of the Royal Society
842 A 379, 20200089.

843 [31] Geer, A.J., 2020. Learning earth system models from observations: machine
844 learning or data assimilation? Technical Report 863. ECMWF. doi:10.
845 21957/7fyj2811r.

846 [32] Goodfellow, I., Bengio, Y., Courville, A., 2016. Deep learning. MIT press.

847 [33] Guo, M., Hesthaven, J.S., 2019. Data-driven reduced order modeling for
848 time-dependent problems. Computer Methods in Applied Mechanics and En-
849 gineering 345, 75–99.

850 [34] Hatfield, S.E., Chantry, M., Dueben, P.D., Lopez, P., Geer, A.J., Palmer,
851 T.N., 2021. Building tangent-linear and adjoint models for data assimilation
852 with neural networks. Earth and Space Science Open Archive ESSOAr .

853 [35] Heaney, C.E., Salinas, P., Fang, F., Pain, C.C., Navon, I.M., 2018. Goal-
854 based sensitivity maps using time windows and ensemble perturbations. arXiv
855 preprint arXiv:1804.04457 .

[36] Hesthaven, J., Ubbiali, S., 2018. Non-intrusive reduced order modeling of nonlinear problems using neural networks. Journal of Computational Physics 363, 55–78. doi:10.1016/j.jcp.2018.02.037.

[37] Kalman, R.E., 1960. A new approach to linear filtering and prediction problems. Journal of basic Engineering 82, 35–45.

[38] Kalnay, E., 2003. Atmospheric modeling, data assimilation and predictability. Cambridge University Press, Cambridge, MA .

[39] Khandelwal, P., Nadler, P., Arcucci, R., Knottenbelt, W., Guo, Y.K., 2020. A scalable inference method for large dynamic economic systems, in: NeuRIPS ML for Economic Policy Workshop, p. Forthcoming.

[40] Krause, A., Singh, A., Guestrin, C., 2008. Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. Journal of Machine Learning Research 9, 235–284.

[41] Le Cornec, C.M., Molden, N., van Reeuwijk, M., Stettler, M.E., 2020. Modelling of instantaneous emissions from diesel vehicles with a special focus on nox: Insights from machine learning techniques. Science of The Total Environment 737, 139625.

[42] Li, Q., Chen, L., Tai, C., Weinan, E., 2017. Maximum principle based algorithms for deep learning. The Journal of Machine Learning Research 18, 5998–6026.

[43] Lin, H., Jin, J., Herik, H., 2019. Air quality forecast through integrated data assimilation and machine learning, in: International Conference on Agents and Artificial Intelligence, pp. 787–793. doi:10.5220/0007555207870793.

[44] Lorenc, A., 1997. Development of an operational variational assimilation scheme. Journal of the Meteorological Society of Japan 75, 339–346.

[45] Ma, C., Wojtowytsch, S., Wu, L., et al., 2020. Towards a mathematical understanding of neural network-based machine learning: what we know and what we don't. arXiv preprint arXiv:2009.10713 .

[46] Mack, J., Arcucci, R., Molina-Solana, M., Guo, Y.K., 2020. Attention-based convolutional autoencoders for 3d-variational data assimilation. Computer Methods in Applied Mechanics and Engineering 372, 113291.

[47] Marrone, S., Piantadosi, G., Sansone, M., Sansone, C., 2017. Look-up tables for efficient non-linear parameters estimation, in: International Conference on Optimization and Decision Science, Springer. pp. 49–57.

[48] Mosser, L., Dubrule, O., Blunt, M.J., 2019. Deepflow: History matching in the space of deep generative models. CoRR abs/1905.05749. arXiv:1905.05749.

[49] Nadler, P., Arcucci, R., Guo, Y., 2020. A neural sir model for global forecasting, in: Machine Learning for Health, PMLR. pp. 254–266.

[50] Nadler, P., Arcucci, R., Guo, Y.K., 2019. A scalable approach to econometric inference, in: PARCO, pp. 59–68.

[51] Nichols, N., 2010. Data Assimilation - Chapter Mathematical concepts in data assimilation. Springer.

[52] Ning, G., Huang, H., 2019. Lighttrack: A generic framework for online top-down human pose tracking. arXiv preprint arXiv:1905.02822 arXiv:1905.02822.

[53] Perdikaris, P., Raissi, M., Damianou, A., Lawrence, N., Karniadakis, G., 2017. Nonlinear information fusion algorithms for data-efficient multi-fidelity modelling. Proceedings of the Royal Society A: Mathematical, Physical and Engineering Science 473, 20160751. doi:10.1098/rspa.2016.0751.

[54] Quilodrán Casas, C., 2018. Fast ocean data assimilation and forecasting using a neural-network reduced-space regional ocean model of the north Brazil current. Ph.D. thesis. Imperial College London.

[55] Quilodrán Casas, C., Arcucci, R., Wu, P., Pain, C., Guo, Y.K., 2020. A reduced order deep data assimilation model. Physica D: Nonlinear Phenomena 412, 132615.

[56] Raaj, Y., Idrees, H., Hidalgo, G., Sheikh, Y., 2019. Efficient online multi-person 2d pose tracking with recurrent spatio-temporal affinity fields, in: IEEE Conference on Computer Vision & Pattern Recognition, pp. 4620–4628.

[57] Raissi, M., Perdikaris, P., Karniadakis, G., 2016. Inferring solutions of differential equations using noisy multi-fidelity data. Journal of Computational Physics 335. doi:10.1016/j.jcp.2017.01.060.

[58] 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.

[59] Rasheed, A., San, O., Kvamsdal, T., 2019. Digital twin: Values, challenges and enablers. ArXiV pre-print: arXiv:1910.01719 .

[60] Rasmussen, C.E., Williams, C.K.I., 2005. Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning). The MIT Press.

[61] Shannon, C.E., 1948. A mathematical theory of communication. Bell system technical journal 27, 379–423.

[62] Skitka, J., Marston, J., Fox-Kemper, B., 2020. Reduced-order quasilinear model of ocean boundary-layer turbulence. Journal of Physical Oceanography 50, 537–558.

[63] Smith, P.J., Dance, S.L., Baines, M.J., Nichols, N.K., Scott, T.R., 2009. Variational data assimilation for parameter estimation: application to a simple morphodynamic model. Ocean Dynamics 59, 697.

[64] Swischuk, R., Mainini, L., Peherstorfer, B., Willcox, K., 2019. Projection-based model reduction: Formulations for physics-based machine learning. Computers & Fluids 179, 704–717. doi:10.1016/j.compfluid.2018.07.021.

[65] Titsias, M., 2009. Variational learning of inducing variables in sparse gaussian processes, in: Artificial intelligence and statistics, PMLR. pp. 567–574.

[66] Tofts, P.S., Brix, G., Buckley, D.L., Evelhoch, J.L., Henderson, E., Knopp, M.V., Larsson, H.B.W., Lee, T.Y., Mayr, N.A., Parker, G.J.M., Others, 1999. Estimating kinetic parameters from dynamic contrast-enhanced t 1-weighted mri of a diffusable tracer: standardized quantities and symbols. Journal of Magnetic Resonance Imaging 10, 223–232. doi:10.1002/(SICI)1522-2586(199909)10:3<223::AID-JMRI2>3.0.CO;2-S.

[67] Vinyals, O., Blundell, C., Lillicrap, T., Wierstra, D., et al., 2016. Matching networks for one shot learning. Advances in neural information processing systems 29, 3630–3638.

[68] Walter, C.A., 2000. The efficient market hypothesis, the gaussian assumption, and the investment management industry, in: EFMA 2001 Lugano Meetings.

[69] Weinan, E., 2017. A proposal on machine learning via dynamical systems. Communications in Mathematics and Statistics 5, 1–11.

[70] Van der Wilk, M., Rasmussen, C.E., Hensman, J., 2017. Convolutional gaussian processes. arXiv preprint arXiv:1709.01894 .

[71] Williams, M., Rowley, C., Mezic, I., Kevrekidis, I., 2014. Data fusion via intrinsic dynamic variables: An application of data-driven koopman spectral analysis. EPL (Europhysics Letters) 109. doi:10.1209/0295-5075/109/40007.

[72] Xiao, D., Heaney, C., Mottet, L., Fang, F., Lin, W., Navon, I., Guo, Y., Matar, O., Robins, A., Pain, C., 2019. A reduced order model for turbulent flows in the urban environment using machine learning. Building and Environment 148, 323–337. doi:10.1016/j.buildenv.2018.10.035.

[73] Yu, H., Tian, X., Li, Q., et al., 2020. Onsagernet: Learning stable and interpretable dynamics using a generalized onsager principle. arXiv preprint arXiv:2009.02327 .

[74] Zhu, C., Byrd, R.H., Lu, P., Nocedal, J., 1997. Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization. ACM Transactions on Mathematical Software (TOMS) 23, 550–560.

[75] Zhu, J., Hu, S., Arcucci, R., Xu, C., Zhu, J., Guo, Y.k., 2019. Model error correction in data assimilation by integrating neural networks. Big Data Mining and Analytics 2, 83–91.

Acronyms

AE Autoencoder
BLUE Best Linear Unbiased Estimator
CNN Convolutional Neural Network
DA Data Assimilation
DCE-MRI Dynamic Contrast Enhanced-Magnetic Resonance Imaging
DL Data Learning
EnKF Ensemble Kalman filter
FS full-state
RFS reconstructed full-state
GP Gaussian Process
GPR Gaussian Process Regression
HFM high fidelity model
IEnKS Iterative Ensemble Kalman Smoother
KF Kalman filter
LMA Levenberg-Marquardt algorithm
LSTM Long Short Term Memory
LuT Look-up Table
ML Machine Learning
MSE Mean Squared Error
NN Neural Network
PC Principal Component
PCA Principal Component Analysis
pdf probability density function
POD Proper Orthogonal Decomposition
RFS reconstructed full-state
RNN Recurrent Neural Network
ROM Reduced Order Model
SSH Sea Surface Height
SST Sea Surface Temperature
TVP-VAR Time-Varying Parameter Vector Autoregressive Model
VarDA Variational Data Assimilation