Data Assimilation Networks

Pierre Boudier[1], Anthony Fillion[2], Serge Gratton[2], Selime Gürol[3], Sixin Zhang[2]

[1]NVIDIA / ANITI, Toulouse, France
[2]Université de Toulouse / ANITI, Toulouse, France
[3]CERFACS / ANITI, Toulouse, France

*

Key Points:

- We propose a general framework DAN based on an extended Elman Network for Bayesian Data Assimilation.
- We show that DAN can achieve optimal prior and posterior density estimations by optimizing likelihood-based objective function.
- Numerically DAN can achieve comparable performance to the EnKF on Lorenz-95 system, without tuning of localization or inflation.

---

*Partially supported by 3IA Artificial and Natural Intelligence Toulouse Institute, French “Investing for the Future - PIA3” program under the Grant agreement ANR-19-PI3A-0004

Corresponding author: Sixin Zhang, `sixin.zhang@irit.fr`

Abstract

Data assimilation (DA) aims at forecasting the state of a dynamical system by combining a mathematical representation of the system with noisy observations taking into account their uncertainties. State of the art methods are based on the Gaussian error statistics and the linearization of the non-linear dynamics which may lead to sub-optimal methods. In this respect, there are still open questions how to improve these methods. In this paper, we propose a *fully data driven deep learning architecture* generalizing recurrent Elman networks and data assimilation algorithms which approximate a sequence of prior and posterior densities conditioned on noisy observations. By construction our approach can be used for general nonlinear dynamics and non-Gaussian densities. On numerical experiments based on the well-known Lorenz-95 system and with Gaussian error statistics, our architecture achieves comparable performance to EnKF on both the analysis and the propagation of probability density functions of the system state at a given time without using any explicit regularization technique.

Plain Language Summary

Data assimilation (DA) aims at forecasting the state of a dynamical system by combining information coming from the model dynamics and noisy (sparse) observations based on their error statistics. Bayesian data assimilation uses the random nature of both the physical and observational error which can be described in terms of probability density functions. This is formally accomplished by using Bayes' Theorem, which requires calculation of the densities that may be quite complex. Practical algorithms then perform linearization of nonlinear operators which are optimal for Gaussian statistics and may use limited information due to computational cost. This results in sub-optimal DA algorithms which requires then the use of explicit regularization techniques to increase the performance of the algorithm or obtain stable algorithms.

With the advances in Machine Learning (ML) and deep learning, there has been significant increase in the research of using ML for data assimilation to decrease the computational cost, or to have better estimation of the state. In this paper, we propose a fully data driven algorithm to learn the prior and posterior pdfs conditioned on the given observations. Our learning is based on the reference trajectories of the model and observations, and loss function minimizes the information loss in the sense of the Kullback-Leibler (KL) divergence. Numerical experiments show that we obtain comparable performance to that of EnKF without the need of localisation and inflation techniques. These numerical results shows the potential advantage of NN based algorithms when the used practical algorithms are sub-optimal.

1 Introduction

1.1 Context

In Data assimilation (DA, (Asch et al., 2016)), the time dependent state of a system is estimated using two models that are the *observational model*, which relates the state to physical observations, and the *dynamical model*, that is used to propagate the state along the time dimension. These models can be written as a Hidden Markov Model (HMM).

Observational and dynamical models are described using random variables that account for observation and state errors. Hence DA algorithms are grounded on a Bayesian approach in which observation realizations are combined with the above statistical models to obtain state predictive and posterior density sequences. This estimation is done in two recursive steps: the *analysis* updates a predictive density into a posterior one with an incoming observation; and the *propagation* updates a posterior density into a the next cycle predictive (or prior) density.

DA methods use additional assumptions or approximations to obtain closed expressions for the densities so that they can be handled by computers. Historically in the *Kalman filter* (KF, (Kalman, 1960)) approach, statistical models are supposed to be Gaussian and operators linear. Hence, the propagation and analysis steps consist in updating *mean and covariance matrix* of densities. In the *Ensemble Kalman Filter* (EnKF, (Evensen, 2009)) approach, these densities are represented by a *set of sampling vectors*. EnKF when used with a small number of ensembles results in low-rank representation of the error covariance matrices. This causes some spurious errors in the covariance matrix which are filtered by using regularization techniques such as localization and inflation (Hamill et al., 2001; Houtekamer & Mitchell, 2001; Asch et al., 2016). EnKF can be used for nonlinear dynamics, however due to the truncation of the statistics up to the second order, in the limit of large ensembles the EnKF filter solution differs from the solution of the Bayesian filter (Le Gland et al., 2011), except for linear dynamics and Gaussian statistics. Hence, when using these methods for non-linear and non-Gaussian setting there are still open questions in achieving an optimal prediction error in the Bayesian setting.

In this paper, we propose a general supervised learning framework based on Recurrent Neural Network (RNN) for Bayesian DA to approximate a sequence of prior and posterior densities conditioned on noisy observations. Section 2 explains the sequential Bayesian DA framework with an emphasis on the *time invariant structure* in the Bayesian DA which is the key property for RNNs. The proposed approach, *Data Assimilation Network* (DAN), is then detailed in Section 3 which generalizes both the Elman Neural Network and the Kalman Filter. DAN approximates the prior and posterior densities by minimizing the log-likelihood cost function based on the information loss, related to the cross-entropy. The details of the cost function and the theoretical results for the optimal solution of the cost function are presented in Section 3.4. The practical aspects of the DAN including the architecture and computationally efficient training algorithm are given in Section 4. We then provide the numerical results on the Lorenz-95 system in Section 5 which includes the stability analysis also beyond the time-interval or the initial condition used in the training. Finally, we provide the conclusions in Section 6.

1.2 Related work

With the advances in machine learning and deep learning, there has been significant increase in the research of using ML to forecast the evolution of physical systems with a data-driven approach (Brunton et al., 2016; Rudy et al., 2017; Raissi et al., 2019, 2017a, 2017b; Li et al., 2020; Jia et al., 2021). Recently, this research has its significant impact on the design and use of advanced DA algorithms. We next outline three main directions that are related to our research in the hybridization of DA and ML approaches.

In a first direction, one addresses the traditional DA problem where the goal is to estimate the distribution of a state sequence x_t conditioned on an observation sequence y_t, by using explicitly an underlying dynamical model $\mathcal{M}$. Harter and de Campos Velho (2012) propose to use Elman Neural Network to learn the analysis equation of KF type algorithm where the dynamics are nonlinear. Their main aim is to reduce the computational complexity without affecting the accuracy. McCabe and Brown (2021) focus on the learning of the analysis equation within an EnKF framework. They propose the Amortized Ensemble Filter which aims to improve existing EnKF algorithms by replacing the EnKF analysis equations with a parameterized function in the form of a neural network.

In a second direction, one aims to learn an unknown dynamical model $\mathcal{M}$ from noisy observations of y_t. This direction is more ambitious compared to the first one as the dynamics to be learnt can be non-linear or even chaotic. Bocquet et al. (2019) propose to use the Bayesian data assimilation framework to learn a parametric $\mathcal{M}$ from sequences of observations y_t. The dynamical model is represented by a surrogate model which is formalized as a neural network under locality and homogeneity assumptions. Bocquet et al. (2020) extends this framework to the joint estimation of the state x_t and the dynamical model $\mathcal{M}$ with a model error represented by a covariance matrix. They estimate the ensembles of the state by using a traditional Ensemble Kalman Smoother based on Gaussian assumption, and then with the given posterior ensemble they minimize for the dynamical model and its error statistics. Similarly, Brajard et al. (2020) propose an iterative algorithm to learn a neural-network parametric model of $\mathcal{M}$. With a fixed $\mathcal{M}$, it estimates the state x_t using the observations y_t, and then uses the estimated state to optimize the parameters of $\mathcal{M}$. A related work is from Krishnan et al. (2015), which introduces a deep KF to estimate the mean and the error covariance matrix in KF to model medical data, based on variational autoencoder (Girin et al., 2021).

A third direction, which is what we consider in the present paper, is to estimate the distribution of a state sequence x_t conditioned on a observation sequence y_t, without explicitly using the underlying dynamical model $\mathcal{M}$ in the propagation. This direction often uses training data in a supervised form of (x_t, y_t). For instance, Fablet et al. (2021) propose a joint learning of the NN representation of the model dynamics and of the analysis equation for the sub-problem albeit within a traditional variational data assimilation framework. A related work to learn an implicit model is Revach et al. (2022), which proposes a parametric KF to handle partially known model dynamics, replacing explicit covariance matrices by a parametric NN to estimate the model error.

All these approaches consider improving the DA methodologies which are based on an *existing DA algorithm* within sequential or variational framework. In this work, we propose a fully data driven approach for Bayesian data assimilation without relying on any prior DA algorithm that can be sub-optimal in case of non-Gaussian error statistics and non-linear dynamics.

1.3 Notation

We denote a state random variable at time t as $\boldsymbol{x}_t$ taking their values in some space $\mathbb{X} = \mathbb{R}^n$ of dimension n. An observation random variable at time t is denoted by $\boldsymbol{y}_t$ taking its values in some space $\mathbb{Y}$ of dimension d (often $\mathbb{R}^d$). We write a sequence of random variables $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_t$ as $\boldsymbol{x}_{1:t}$. A joint probability density of two sequence of random variables $\boldsymbol{x}_{1:t}$ and $\boldsymbol{y}_{1:t}$ with respect to the Lebesgue measure on the finite dimensional Euclidean space $\mathbb{X}^t \times \mathbb{Y}^t$ is written as $p(x_{1:t}, y_{1:t}) = p_{\boldsymbol{x}_{1:t}, \boldsymbol{y}_{1:t}}(x_{1:t}, y_{1:t})$. The set of pdfs over $\mathbb{X}$ is denoted by $\mathbb{P}_{\mathbb{X}}$. A conditional pdf for $\boldsymbol{x}_t$ given $\boldsymbol{y}_t = y_t$ is written as $p_{\boldsymbol{x}_t|\boldsymbol{y}_t}(\cdot|y_t) \in \mathbb{P}_{\mathbb{X}}$.

2 Sequential Bayesian Data assimilation

In this section, we review the Bayesian optimal solution of sequential Bayesian data assimilation for an observed dynamical system and use its repetitive time-invariant structure to motivate the introduction of the DAN framework.

2.1 Sequential Bayesian Data assimilation

Data assimilation aims to estimate the state of a dynamical process which is modeled by a discrete-time stochastic equation and observed via available instruments which can be modeled by another stochastic equation (Asch et al., 2016). These equations are given by the following system:

$$\boldsymbol{x}_t = \mathcal{M}\left(\boldsymbol{x}_{t-1}\right) + \boldsymbol{\eta}_t, \qquad \text{(propagation equation)} \qquad (1a)$$

$$\boldsymbol{y}_t = \mathcal{H}\left(\boldsymbol{x}_t\right) + \boldsymbol{\varepsilon}_t, \qquad \text{(observation equation)} \qquad (1b)$$

where $\mathcal{M}(\cdot)$ is the nonlinear propagation operator that acts on the model state random variable vector at time t, $\boldsymbol{x}_t \in \mathbb{X}$ and return the model state vector $\boldsymbol{x}_{t+1} \in \mathbb{X}$. $\mathcal{H}(\cdot)$ is the nonlinear observation operator that acts on the state random variable $\boldsymbol{x}_t$ and approximately returns the observation random variable $\boldsymbol{y}_t \in \mathbb{Y}$ at time t. Both of these steps may involves errors and they are represented by an *additive model error*, $\boldsymbol{\eta}_t$, For example, the observation operator may involve spatial interpolations, physical unit transformations and so on, resulting in measurement errors.

and an *additive observation error*, $\boldsymbol{\varepsilon}_t$. We assume that these stochastic errors are distributed according to the pdf $p_{\boldsymbol{\eta}}$ and $p_{\boldsymbol{\varepsilon}}$ and they are are i.i.d. along time, independent to the initial state x_1. Using these assumptions DA problem can be interpreted as a Hidden Markov Model (Carrassi et al., 2018).

Given such a dynamical model, sequential Bayesian DA aims at quantifying the uncertainty over the system state each time an observation sample becomes available. Such an analysis starts by rewriting, under suitable mathematical assumptions, the DA system in terms of *conditional probability density functions* $p_{\boldsymbol{x}_t|\boldsymbol{x}_{t-1}}(\cdot|x_{t-1}) \in \mathbb{P}_{\mathbb{X}}$ which represents (1a), and $p_{\boldsymbol{y}_t|\boldsymbol{x}_t}(\cdot|x_t) \in \mathbb{P}_{\mathbb{Y}}$ which represents (1b). Using these densities, we can quantify the uncertainty of the state as a function of the observations. This can be done in two steps sequentially using the Bayesian framework: the analysis step and the propagation (forecast) step. Let $p_t^b := p_{\boldsymbol{x}_t|\boldsymbol{y}_{1:t-1}}$ be the posterior distribution of $\boldsymbol{x}_t$ given $\boldsymbol{y}_{1:t-1}$, and $p_t^a := p_{\boldsymbol{x}_t|\boldsymbol{y}_{1:t}}$ be the posterior distribution of $\boldsymbol{x}_t$ given $\boldsymbol{y}_{1:t}$. The *analysis* step computes $p_t^a(\cdot|y_{1:t}) \in \mathbb{P}_{\mathbb{X}}$ from $p_t^b(\cdot|y_{1:t-1}) \in \mathbb{P}_{\mathbb{X}}$ based on Bayes rule,

$$p_t^a(\cdot|y_{1:t}) = \frac{p_{\boldsymbol{y}_t|\boldsymbol{x}_t}(y_t|\cdot)\, p_t^b(\cdot|y_{1:t-1})}{p_{\boldsymbol{y}_{1:t-1}}(y_{1:t-1})} \tag{2}$$

Here, $p_{\boldsymbol{y}_t|\boldsymbol{x}_t}(y_t|\cdot)$ is considered as a likelihood function of x_t, and $p_{\boldsymbol{y}_{1:t-1}}$ is marginal distribution of observations. Similarly, the *propagation* step computes $p_{t+1}^b(\cdot|y_{1:t})$ from $p_t^a(\cdot|y_{1:t})$,

$$p_{t+1}^b(\cdot|y_{1:t}) = \int p_{\boldsymbol{x}_{t+1}|\boldsymbol{x}_t}(\cdot|x) p_t^a(x|y_{1:t})\mathrm{d}x. \tag{3}$$

The analysis and forecast steps are then repeated within a given number of cycles (time interval) in which the forecast step provides a prior density for the next cycle.

Performing the analysis and propagation steps in (2) and (3) with linear dynamics for the propagation operator $\mathcal{M}(\cdot)$ and the observation operator $\mathcal{H}(\cdot)$, and using a Gaussian assumption for the probabilities $p_{\boldsymbol{\varepsilon}}$ and $p_{\boldsymbol{\eta}}$ reduces to the well known *Kalman filter* (KF, (Kalman, 1960)). The challenge is that the calculation of the pdfs become intractable with nonlinear ODS or non-Gaussian pdfs of the error terms. When the dynamics are nonlinear, ensemble type KFs such as Ensemble KF (Evensen, 2009) are widely used alternative methods, but when used with limited number of ensembles, they require additional techniques (see Section 3.3 for further discussions).

2.2 Time-invariant structure in the BDA

We review the *invariant structure* of the BDA for the ODS defined in Section 2.1, which is a key property to motivate the DAN framework. Following the i.i.d. assumptions that we have made on the errors in (1a) and (1b), the conditional pdfs $p_{\boldsymbol{x}_{t+1}|\boldsymbol{x}_t}$ and $p_{\boldsymbol{y}_t|\boldsymbol{x}_t}$ are time invariant, in the sense that for $t = 1, 2, \ldots$

$$\begin{aligned} p_{\boldsymbol{x}_{t+1}|\boldsymbol{x}_t}(u|v) &= p_{\boldsymbol{x}_2|\boldsymbol{x}_1}(u|v) \\ p_{\boldsymbol{y}_t|\boldsymbol{x}_t}(y|v) &= p_{\boldsymbol{y}_1|\boldsymbol{x}_1}(y|v) \end{aligned}$$

for all $u, v \in \mathbb{X}$ and $y \in \mathbb{Y}$.

As a result, the conditional pdfs representing the ODS are time invariant in the following sense. The analysis step (2) can then be considered as a *time invariant function*, a^{BDA}, which operates on the prior cpdf, $p_t^b(\cdot|y_{1:t-1}) \in \mathbb{P}_{\mathbb{X}}$ and a current observation, $y_t \in \mathbb{Y}$, and then return a posterior cpdf $p^a_t(\cdot|y_{1:t}) \in \mathbb{P}_{\mathbb{X}}$:

$$p^a_t(\cdot|y_{1:t}) = a^{BDA}\left[p^b_t(\cdot|y_{1:t-1}), y_t\right].$$

Similarly, according to (3), the propagation transformation can be considered as a *time invariant function*, b^{BDA}, that transforms a posterior pdf to a prior pdf,

$$p^b_{t+1}(\cdot|y_{1:t}) = b^{BDA}\left[p^a_t(\cdot|y_{1:t})\right].$$

This presentation of the sequential BDA allows us to see the DA cycle as the composition of *two time invariant* transformations a^{BDA} and b^{BDA}, i.e. each transformation is produced using the *same update rule* applied to the previous transformations. Exploiting this *repetitive time invariant structure*, corresponding to a *chain of events*, leads to a general framework named as the DAN based on recurrent neural networks (RNNs). We detail these ingredients of the DAN in Section 3 and Section 4.

3 Data Assimilation Networks (DAN)

In section 3.1 we present a general framework for DAN which generalizes both traditional data assimilation algorithms described in Section 3.2 and 3.3. Thanks to the repetitive structure of BDA, it allows one to address nonlinear model dynamics and non-Gaussian error distributions. Section 3.4 presents a key ingredient of DAN, which is the cost function based on the log-likelihood, and its theoretical properties. Instead of calculating the posterior pdfs analytically, DAN aims to learn these pdfs by using sequences of (x_t, y_t) generated from the ODS.

3.1 DAN framework

For a given set $\mathbb{S}$, DAN is defined as a triplet of transformations such that

$$a \in \mathbb{S} \times \mathbb{Y} \to \mathbb{S}, \text{ (analyzer)} \tag{4a}$$

$$b \in \mathbb{S} \to \mathbb{S}, \text{ (propagater)} \tag{4b}$$

$$c \in \mathbb{S} \to \mathbb{P}_{\mathbb{X}}, \text{ (procoder)} \tag{4c}$$

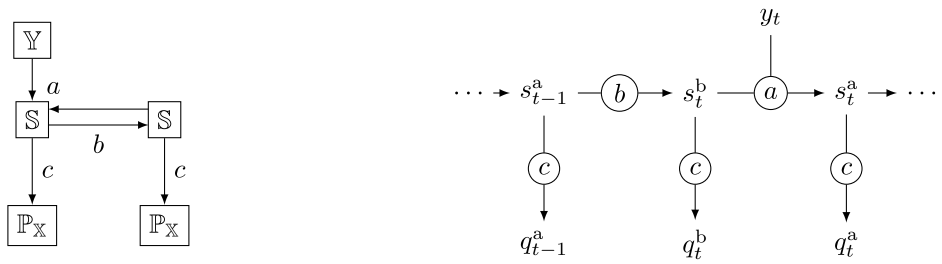
The term “procoder” is a contraction of “probability coder” as the function c transforms an internal representation into an actual pdf over $\mathbb{X}$. A representation of a DAN is given by Figure 1a. When $S = \mathbb{P}_{\mathbb{X}}$ and c is identity, this framework encompasses the transformation of a^{BDA} and b^{BDA} in the BDA as a special case. However, it includes also other DA algorithms such as Kalman Filter and Ensemble Kalman Filter. Such connections are detailed in Section 3.2 and 3.3.

One important ingredient of DAN as a general framework for *cycled* DA algorithms is the use of memory to transform prior and posterior densities from one cycle to the next one. In this respect, $\mathbb{S}$ can be interpreted as a memory space which is a finite-dimensional vector space within the DAN framework. Considering DAN as a RNN with memory usage naturally make the link with the well-known *Elman Network.* This connection is detailed in Section 4.1.

As a recurrent neural network, we can unroll DAN into a sequence of transformations. Given an initial memory $s^{\mathrm{a}}_0 \in \mathbb{S}_0$, and an observation trajectory $y_{1:T} \in \mathbb{Y}^T$, a DAN recursively outputs a predictive and a posterior sequence such that for $1 \le t \le T$,

$$\begin{aligned} s^b_t &\coloneqq b\left(s^{\mathrm{a}}_{t-1}\right), \quad s^{\mathrm{a}}_t \coloneqq a\left(s^b_t, y_t\right) \\ q^b_t &\coloneqq c\left(s^b_t\right), \quad q^{\mathrm{a}}_t \coloneqq c\left(s^{\mathrm{a}}_t\right). \end{aligned}$$

This recursive application is represented in Figure 1b. Note that $\{q^b_t\}^T_{t=1}$ and $\{q^{\mathrm{a}}_t\}^T_{t=1}$ are *candidate conditional densities.* This means that for a given sequence of observations



(a) Scheme of a DAN

(b) Scheme of a recursive DAN application

Figure 1: Representation of a DAN: (a) scheme of a DAN (b) unrolled DAN along time interval

$y_{1:t} = (y_1, \cdots, y_t)$, we have $q^b_t(\cdot|y_{1:t-1}) \in \mathbb{P}_{\mathbb{X}}$ and $q^a_t(\cdot|y_{1:t}) \in \mathbb{P}_{\mathbb{X}}$. However, these candidate conditional densities are not required to be compatible by construction with a joint-distribution over $\mathbb{X}^T \times \mathbb{Y}^T$. As a consequence, we do not assume that there is some joint distribution $q(x_{1:T}, y_{1:T})$ which induces the $q^b_t(\cdot|y_{1:t-1})$ and $q^a_t(\cdot|y_{1:t})$. However, as we shall see in Section 4, the construction of DAN using recurrent neural networks implicitly imposes some relationships between these candidate conditional densities.

3.2 The Kalman Filter as a DAN

In the original *Kalman filter* (KF, (Kalman, 1960), the propagation operator $\mathcal{M}$ is supposed affine with M as linear part and the observation operator $\mathcal{H}$ also affine with H as linear part. In this case, the analysis and propagation transformations preserve Gaussian pdfs that are easily characterized by their mean and covariance matrix. The analysis and propagation transformations then simplify to algebraic expressions on these pairs as we shall see in this section.

Suppose that the internal representation of a Gaussian pdf is formalized by the injective transformation, $c^{\mathrm{KF}} : \mathbb{Z}_{\mathbb{X}} \to \mathbb{G}_{\mathbb{X}}$,

$$c^{\mathrm{KF}}(s) = \mathcal{N}(\mu, \Sigma),$$

where $s \coloneqq (\mu, \Sigma)$, μ and Σ being the mean and covariance matrix respectively and $\mathbb{Z}_{\mathbb{X}}$ is the set of mean and covariance matrix pairs over $\mathbb{X}$, $\mathbb{G}_{\mathbb{X}}$ is the set of Gaussian pdfs over $\mathbb{X}$. The KF analysis transformation is the function that transforms such a prior pair in $\mathbb{Z}_{\mathbb{X}}$ and an observation y in $\mathbb{Y}$ into the posterior pair in $\mathbb{Z}_{\mathbb{X}}$, i.e. $a^{\mathrm{KF}} : \mathbb{Z}_{\mathbb{X}} \times \mathbb{Y} \to \mathbb{Z}_{\mathbb{X}}$, given by

$$a^{\mathrm{KF}}\left(\mu^{\mathrm{b}}, \Sigma^{\mathrm{b}}, y\right) = \left(\mu^{\mathrm{a}}, \Sigma^{\mathrm{a}}\right) \tag{5}$$

with $\Sigma^{\mathrm{a}} = \left(H^{\mathrm{T}} R^{-1} H + \left(\Sigma^{\mathrm{b}}\right)^{-1}\right)^{-1}$, $\mu^{\mathrm{a}} = \mu^{\mathrm{b}} + \Sigma^{\mathrm{a}} H^{\mathrm{T}} R^{-1}\left(y - H\left(\mu^{\mathrm{b}}\right)\right)$. The mapping diagram for the analysis step of the KF is given by the diagram in Figure 2a, which is a commutative diagram.

As well, the KF propagation transformation is the function that transforms a posterior pair in $\mathbb{Z}_{\mathbb{X}}$ into the next cycle prior in $\mathbb{Z}_{\mathbb{X}}$, i.e. $b^{\mathrm{KF}} : \mathbb{Z}_{\mathbb{X}} \to \mathbb{Z}_{\mathbb{X}}$, given by

$$b^{\mathrm{KF}}\left(\mu^{\mathrm{a}}, \Sigma^{\mathrm{a}}\right) = \left(\mu^{\mathrm{b}}, \Sigma^{\mathrm{b}}\right) \tag{6}$$

with $\Sigma^{\mathrm{b}} = M \Sigma^{\mathrm{a}} M^{\mathrm{T}} + Q$, Q being the model error covariance matrix and $\mu^{\mathrm{b}} = M\left(\mu^{\mathrm{a}}\right)$. The mapping diagram for the propagation step of the KF is given by the diagram in Figure 2b, which is a commutative diagram.

$$\begin{tikzcd}
\mathbb{G}_{\mathbb{X}} \times \mathbb{Y} \arrow[r, "a^{\mathrm{BDA}}"] & \mathbb{G}_{\mathbb{X}} \\
\mathbb{Z}_{\mathbb{X}} \times \mathbb{Y} \arrow[r, "a^{\mathrm{KF}}"] \arrow[u, "c^{\mathrm{KF}}"] & \mathbb{Z}_{\mathbb{X}} \arrow[u, "c^{\mathrm{KF}}"']
\end{tikzcd}$$

(a) Commuting diagram for the KF analysis

$$\begin{tikzcd}
\mathbb{G}_{\mathbb{X}} \arrow[r, "b^{\mathrm{BDA}}"] & \mathbb{G}_{\mathbb{X}} \\
\mathbb{Z}_{\mathbb{X}} \arrow[r, "b^{\mathrm{KF}}"] \arrow[u, "c^{\mathrm{KF}}"] & \mathbb{Z}_{\mathbb{X}} \arrow[u, "c^{\mathrm{KF}}"']
\end{tikzcd}$$

(b) Commuting diagram for the KF propagation

Figure 2: Kalman filter mapping diagram

Unfortunately, operators linearity is rarely met in practice and covariance matrices may not be easy to store and manipulate in the case of large scale problems. A popular dimension reduction approach is the *ensemble* Kalman filter that has proven effective in several large scale applications.

3.3 The Ensemble Kalman Filter as a DAN

In the *Ensemble Kalman Filter* (EnKF, (Evensen, 2009)), statistics $(\mu, \Sigma) \in \mathbb{Z}_{\mathbb{X}}$ are estimated from an ensemble matrix $X \in \mathbb{X}^m = \mathbb{R}^{n\times m}$ having m columns with the empirical estimators

$$\mu = Xu, \tag{7a}$$
$$\Sigma = XUX^{\mathrm{T}}, \tag{7b}$$

where $u = \left(\frac{1}{m}, \dots, \frac{1}{m}\right)^{\mathrm{T}} \in \mathbb{R}^m, U = \frac{I_m - m \times uu^{\mathrm{T}}}{m-1} \in \mathbb{R}^{m\times m}$ and $I_m \in \mathbb{R}^{m\times m}$ is the identity matrix. Thus, the algebra over mean and covariance matrices pairs can be represented by operators on ensembles. In this approach nonlinear operators can be evaluated columnwise on ensembles and ensembles with few columns may produce low-rank approximations of large scale covariance matrices. Hence ensembles are an internal representation for the pdfs that are transformed by the function into a Gaussian pdf, $c^{EnKF} : \mathbb{X}^m \to \mathbb{G}_{\mathbb{X}}$,

$$c^{EnKF}(X) = \mathcal{N}\left(Xu, XUX^{\mathrm{T}}\right), \tag{8}$$

when the error covariance matrix XUX^{T} is full-rank, for instance when $m \geq n$.

The EnKF analysis transformation is the function that transforms such a prior ensemble $X_{\mathrm{b}} \in \mathbb{X}^m$ and an observation $y \in \mathbb{Y}$ into the posterior ensemble $X_{\mathrm{a}} \in \mathbb{X}^m$, $a^{\mathrm{EnKF}} : \mathbb{X}^m \times \mathbb{Y} \to \mathbb{X}^m$, given by

$$a^{\mathrm{EnKF}}\left(X_{\mathrm{b}}, y\right) = X_{\mathrm{a}} \quad \text{with} \quad X_{\mathrm{a}} = X_{\mathrm{b}} + K\left(Y - Y_{\mathrm{b}}\right) \tag{9}$$

where $K = X_{\mathrm{b}}UY_{\mathrm{b}}^{\mathrm{T}}\left(Y_{\mathrm{b}}UY_{\mathrm{b}}^{\mathrm{T}} + R\right)^{-1} \in \mathbb{R}^{n\times d}$ is the ensemble Kalman gain, $Y_{\mathrm{b}} = \mathcal{H}\left(X_{\mathrm{b}}\right) \in \mathbb{Y}^m$ and $Y \in \mathbb{Y}^m (= \mathbb{R}^{d\times m})$ is a column matrix with m samples of $\mathcal{N}(y, R)$.

As well, the EnKF propagation transformation is the function that transforms a posterior ensemble $X_{\mathrm{a}} \in \mathbb{X}^m$ into the next cycle prior ensemble $X_{\mathrm{b}} \in \mathbb{X}^m$, $b^{\mathrm{EnKF}} : \mathbb{X}^m \to \mathbb{X}^m$, given by

$$b^{\mathrm{EnKF}}(X_{\mathrm{a}}) = X_{\mathrm{b}} \quad \text{with} \quad X_{\mathrm{b}} = \mathcal{M}\left(X_{\mathrm{a}}\right) + W \tag{10}$$

where $W \in \mathbb{X}^m$ is a column matrix consisting of m samples distributed according to the Gaussian pdf $\mathcal{N}\left(0_n, Q\right)$.

In EnKF, as explained above the mean and the covariance matrix for the Gaussian pdf are calculated through ensembles and propagation is performed through the ensembles using nonlinear dynamics. For large-scale nonlinear systems, when one can use only a limited number of ensembles, the error covariance matrix become a rank deficient matrix. This leads to sub-optimal performance (Asch et al., 2016) and may introduce errors during the propagation. For instance, spurious correlations may appear or ensembles may collapse. As a result, for a stable EnKF regularization techniques like localization and inflation needs to be applied (Hamill et al., 2001; Houtekamer & Mitchell, 2001; Gharamti, 2018). Localization consists in filtering out the long-distance spurious correlations in the error covariance matrix. It is not straightforward to find the optimal parameters for the localization, therefore some tuning is required. This regularization technique also requires observations to be local, i.e. an observation that can be attributed to one model grid point. After filtering out these spurious correlations such that the analysis is updated by the local observations, there may be still problem with the use of limited ensembles along the propagation. These small errors may be problematic when they are accumulated through the cycles. This can still lead to filter divergence. A common solution is to inflate the error covariance matrix by an empirical factor slightly greater than one. The multiplicative inflation compensate errors due to a small size of ensembles and the approximate assumption of Gaussian distribution on the error statistics (Bocquet, 2011).

3.4 DAN log-likelihood cost function

In this section, we introduce a cost function which allows one to optimize the candidate conditional densities, i.e. q_t^{a} and q_t^{b}, based on samples of $\boldsymbol{x}_{1:T}$ and $\boldsymbol{y}_{1:T}$. The distance between the target conditional densities p_t^b and p_t^{a} and the candidate conditional densities q_t^b and q_t^{a} are minimized in the sense of the *information loss*, related to *cross-entropy* (Cover & Thomas, 2005).

Definition 1 (log-likelihood cost function)**.** *Assume* $q = (q_t^b, q_t^{\mathrm{a}})_{t=1}^T \in \mathbb{P} = \left(\Pi_{t=1}^T \mathbb{Y}^{t-1} \to \mathbb{P}_{\mathbb{X}}\right) \times \left(\Pi_{t=1}^T \mathbb{Y}^{t} \to \mathbb{P}_{\mathbb{X}}\right)$ *such that the following log-likelihood cost function is well-defined (i.e. for each* $t \geq 1$*, the Lebesgue integral with respect to* $x_{1:t}$ *and* $y_{1:t}$ *exists)*

$$\mathcal{J}_t(q_t^b, q_t^{\mathrm{a}}) \coloneqq -\int \left[\ln q_t^b(x_t|y_{1:t-1}) + \ln q_t^{\mathrm{a}}(x_t|y_{1:t})\right] p(x_{1:t}, y_{1:t}) \mathrm{d}x_{1:t} \mathrm{d}y_{1:t}. \tag{11}$$

The total log-likelihood cost function is defined as

$$\mathcal{J}(q) \coloneqq \frac{1}{T} \sum_{t=1}^{T} \mathcal{J}_t(q_t^b, q_t^{\mathrm{a}}). \tag{12}$$

The following results shows that if $q \in \mathbb{P}$, the global optima of $\mathcal{J}$ is the Bayesian prior and posterior cpdf trajectories of the ODS.

Theorem 1. *Let* $\bar{q} \in \arg\min_{q \in \mathbb{P}} \mathcal{J}(q)$*, then* $\forall t \in \{1, \cdots, T\}$*,* $\bar{q}_t^b(x|y_{1:t-1}) = p_t^b(x|y_{1:t-1})$ *for* $p_t^b(\cdot|y_{1:t-1})$*-a.e* $x \in \mathbb{X}$ *and p-a.e* $y_{1:t-1} \in \mathbb{Y}^{t-1}$*. Similarly,* $\bar{q}_t^{\mathrm{a}}(x|y_{1:t}) = p_t^{\mathrm{a}}(x|y_{1:t})$ *for* $p_t^{\mathrm{a}}(\cdot|y_{1:t})$*-a.e* $x \in \mathbb{X}$ *and p-a.e* $y_{1:t} \in \mathbb{Y}^t$*.*

Proof. According to (12), it is sufficient to derive the optimal solution of $\mathcal{J}_t(q_t^b, q_t^{\mathrm{a}})$ for each t independently. The proof is an application of the KL-divergence (Kullback & Leibler, 1951) to conditional probability densities. For a function $f(x)$ on a measurable space of $\mathbb{X}$ with probability p, we say $f(x) = 0$ for p-a.e. x (p-almost everywhere shortly p-a.e.) if there exists a measurable set A such that $p(A) = 1$ and $\forall x \in A$, $f(x) = 0$.

We re-write $\mathcal{J}_t(q_t^b, q_t^{\mathrm{a}})$ as

$$\text{-}\int \ln q_t^b(x_t|y_{1:t\text{-}1}) p_t^b(x_t|y_{1:t\text{-}1}) p(y_{1:t\text{-}1}) dx_t \mathrm{d}y_{1:t\text{-}1} - \int \ln q_t^{\mathrm{a}}(x_t|y_{1:t}) p_t^{\mathrm{a}}(x_t|y_{1:t}) p(y_{1:t}) dx_t \mathrm{d}y_{1:t}, \tag{13}$$

using the property $p(x_t, y_{1:t-1}) = p^b_t(x_t|y_{1:t\text{-}1})p(y_{1:t\text{-}1})$ and $p(x_t, y_{1:t}) = p^a_t(x_t|y_{1:t})p(y_{1:t})$. The first term in (13) can be written as conditional relative entropy by including a constant conditional entropy term:

$$\int \left(\int \ln \frac{p^b_t(x_t|y_{1:t\text{-}1})}{q^b_t(x_t|y_{1:t\text{-}1})} p^b_t(x_t|y_{1:t\text{-}1}) dx_t \right) p(y_{1:t\text{-}1}) \mathrm{d}y_{1:t\text{-}1} \geq 0. \tag{14}$$

We have equality in (14) if and only if $q^b_t(x|y_{1:t\text{-}1}) = p^b_t(x|y_{1:t\text{-}1})$ for $p^b_t(\cdot|y_{1:t\text{-}1})$-a.e x, and p-a.e. $y_{1:t\text{-}1}$ (see a proof in (Kullback & Leibler, 1951, Lemma 3.1) and (Bogachev, 2007, Corollary 2.5.4)). Thus, the minimal solution is given by $\bar{q}^b_t$ as stated in the theorem. Similarly, the minimal solution of the second term (13) is given by the $\bar{q}^{\mathrm{a}}_t$ in the statement.

□

The theoretical results in Theorem 1 can not be numerically computed without specifying a functional class of the candidate conditional pdfs $q = (q^b_t, q^{\mathrm{a}}_t)^T_{t=1}$. As a common specific case, we can consider candidate conditional pdfs as the Gaussian pdfs which allows one to match the correct mean and covariance of the target prior and posterior cpdf.

Let $\mathbb{G}_{\mathbb{X}}$ be the set of Gaussian pdfs over $\mathbb{X}$, and $q \in \mathbb{G} = \left(\Pi^T_{t=1}\mathbb{Y}^{t-1} \to \mathbb{G}_{\mathbb{X}}\right) \times \left(\Pi^T_{t=1}\mathbb{Y}^{t} \to \mathbb{G}_{\mathbb{X}}\right)$. For each $J_t(q^b_t, q^{\mathrm{a}}_t)$ in Definition 1 to be well-defined, it is necessary to assume that the target prior and posterior distributions $p^b_t(\cdot|y_{1:t-1})$ and $p^{\mathrm{a}}_t(\cdot|y_{1:t})$ have first-order and second-order moments. Under these assumptions, we have

Theorem 2. *Let $\bar{q} \in \arg\min_{q\in\mathbb{G}} \mathcal{J}(q)$, then $\forall t \in \{1, \cdots, T\}$, the mean and covariance of $\bar{q}^b_t(\cdot|y_{1:t-1})$ equals to the mean and covariance of $p^b_t(\cdot|y_{1:t-1})$ for p-a.e $y_{1:t-1} \in \mathbb{Y}^{t-1}$. Similarly, the mean and covariance of $\bar{q}^a_t(\cdot|y_{1:t})$ equals to the mean and covariance of $p^a_t(\cdot|y_{1:t})$ for p-a.e $y_{1:t} \in \mathbb{Y}^t$.*

Proof. We shall only provide a proof for $\bar{q}^b_t(\cdot|y_{1:t-1})$ as the proof is similar for $\bar{q}^a_t(\cdot|y_{1:t})$. Let $\bar{p}^b_t(\cdot|y_{1:t-1})$ be the Gaussian distribution which has the mean and covariance of $p^b_t(\cdot|y_{1:t-1})$. Following the proof of Theorem 1, we can rewrite the first term, up to a constant, in (13) into

$$\int \left(\int \ln \frac{\bar{p}^b_t(x_t|y_{1:t-1})}{q^b_t(x_t|y_{1:t-1})} p^b_t(x_t|y_{1:t-1}) dx_t \right) p(y_{1:t-1}) \mathrm{d}y_{1:t-1} \tag{15}$$

This is an equivalent minimization problem because we have added a term of $\bar{p}^b_t$ which does not depend on q^b_t. By definition, $q^b_t(\cdot|y_{1:t-1}) \in \mathbb{G}_{\mathbb{X}}, \bar{q}^b_t(\cdot|y_{1:t-1}) \in \mathbb{G}_{\mathbb{X}}$, the logarithm term in (15) is a quadratic function of x_t. As a consequence, we can rewrite (15) as

$$\int \left(\int \ln \frac{\bar{p}^b_t(x_t|y_{1:t-1})}{q^b_t(x_t|y_{1:t-1})} \bar{p}^b_t(x_t|y_{1:t-1}) dx_t \right) p(y_{1:t-1}) \mathrm{d}y_{1:t-1} \geq 0. \tag{16}$$

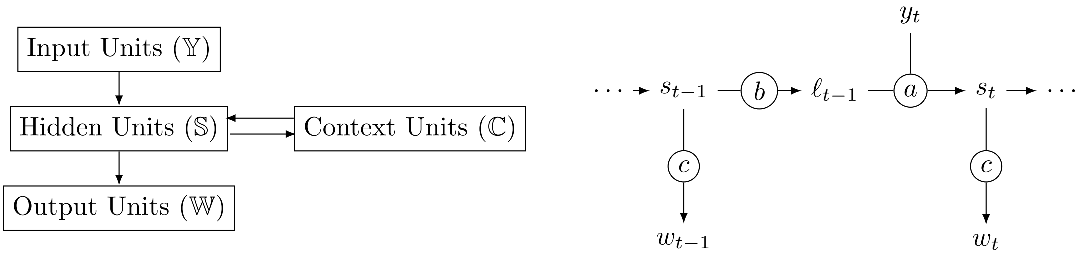
where we have replaced the density p^b_t by $\bar{p}^b_t$ because they have the same first and second order moments. Note that the inner integral in (16) is the KL divergence between $\bar{p}^b_t$ and q^b_t, so its minimal solution $\bar{q}^b_t(\cdot|y_{1:t-1})$ equals almost surely to $\bar{p}^b_t(\cdot|y_{1:t-1})$. Therefore the mean and covariance of $\bar{q}^b_t(\cdot|y_{1:t-1})$ and $p^b_t(\cdot|y_{1:t-1})$ match for p-a.e. $y_{1:t-1}$. □

4 DAN construction and training algorithm

Having specified the cost function in the previous section, we are now going to discuss how to construct the components of a, b, c in DAN in order to fit training data samples. To motivate the DAN construction, we first review its connection with the classical Elman network in Section 4.1. We then specify the construction of the DAN using recurrent neural networks in Section 4.2. Section 4.3 and 4.4 describe how to efficiently train the network.

4.1 Connection with Elman network

DAN can be interpreted as an extension of an *Elman network* (EN) (Elman, 1990) which is a basic structure of recurrent network. An Elman network is a three-layer network (input, hidden and output layers) with the addition of a set of context units. These context units provide memory to the network. Both the input units and context units activate the hidden units; the hidden units then feed forward to activate the output units (Elman, 1990). A representation of a EN is given in Figure 3a.


Input Units ($\mathbb{Y}$)
Hidden Units ($\mathbb{S}$)
Context Units ($\mathbb{C}$)
Output Units ($\mathbb{W}$)
y_t
s_{t-1}
b
ℓ_{t-1}
a
s_t
c
c
w_{t-1}
w_t

(a) Scheme of a Elman Network

(b) Scheme of a recursive Elman application

Figure 3: Representation of a Elman Network: (a) scheme of a EN (b) unrolled EN along time interval

The context units make the Elman network able to process variable length sequences of inputs to produce sequences of outputs as shown in Figure 3b. Indeed, given a new input $y_t \in \mathbb{Y}$ in the input sequence, the function a updates a context memory from $\ell_{t-1} \in \mathbb{C}$ to a hidden state memory $s_t = a\left(\ell_{t-1}, y_t\right) \in \mathbb{S}$. And the function c decodes the hidden state memory into an output $w_t = c\left(s_t\right) \in \mathbb{W}$ in the output sequence. The updated hidden state memory is transferred to the context unit via function b. In a way, the context memory of an Elman network is expected to gather relevant information from the past inputs to perform satisfactory predictions. The training process in machine learning will optimally induce how to manipulate the memory from data.

The similarity between DAN and EN can be made explicit with the analogy that the hidden layer is connected to the context units by the function b, which includes *time propagation* for DAN. In DAN the hidden unit memory $\mathbb{S}$ is considered as the same set as the context unit memory $\mathbb{C}$, and c function decodes both the hidden and the context unit memory into a probability density function.

The EN can not perform DA operations in all its generality. For instance, EN can not make *predictions* without observations, that is estimating strict future states from past observations. This is because the function a performs both the propagation and the analysis at once. In a way, the EN only produces posterior outputs and no prior outputs while the DAN produces prior or posterior outputs by applying the procoder c before or after the propagater b (see Figure 1b and Figure 3b). DAN can also produce strict future predictions without observations by applying the propagater b multiple times before applying the procoder c. Second, the DAN provides a probabilistic representation of the state i.e. an element in $\mathbb{P}_{\mathbb{X}}$ instead of an element in $\mathbb{X}$. Also, note that the compositions of b and c make a generalized propagation operator as it propagates in time probabilistic representations of the state rather than punctual realizations.

4.2 Construct DAN using Recurrent Neural networks (RNN)

We propose to use neural networks to construct a parameterized family of DANs. Let θ denote all the weights in neural networks, and the memory space $\mathbb{S}$ be a finite-dimensional Euclidean space. The parametric family of the analyzers and propagators are L-layer fully connected neural networks:

$$a_\theta : \quad \underbrace{\mathbb{S}\times\mathbb{Y}\to\cdots\to\mathbb{S}\times\mathbb{Y}}_{L\ times}\to\mathbb{S}, \tag{17a}$$

$$b_\theta : \quad \underbrace{\mathbb{S}\to\cdots\to\mathbb{S}}_{L\ times}, \tag{17b}$$

The construction of a_θ is built upon L fully-connected layers with residual connections. It is based on the LeakyReLU activation function and the ReZero trick (Bing et al., 2015; Bachlechner et al., 2020) to improve the trainability when L is large. For layer ℓ, the input $v_{\ell-1}\in\mathbb{S}\times\mathbb{Y}$ is transformed into $v_\ell\in\mathbb{S}\times\mathbb{Y}$ by

$$v_\ell = v_{\ell-1} + \alpha_\ell \mathrm{LeakyReLU}\left(W_\ell v_{\ell-1}+\beta_\ell\right). \tag{18}$$

An extra linear layer is then applied to the output v_L in order to compute a memory state as the output of a_θ. The trainable parameters of a_θ are $(\alpha_\ell, W_\ell, \beta_\ell)_{\ell\leq L}$ and the weight and bias in the linear layer. As illustrated in Figure 1b, the input a_θ at time t is a concatenation of s_t^b and y_t, i.e. $v_0=(s_t^b, y_t)$. Similarly, the b_θ is constructed from the same L fully-connected layers as in (18) by using a different set of trainable parameters. The input of b_θ at time t is set to s_t^a.

The procoder c_θ is specified with respect to the pdf choice of candidate conditional densities. For instance, for the Gaussian case studied in Theorem 2, c_θ can be defined as:

$$c_\theta : \quad \mathbb{S}\to\mathbb{R}^{n+\frac{n(n+1)}{2}}\to\mathbb{G}_{\mathbb{X}} \tag{19}$$

which is a linear layer from $\mathbb{S}$ to $\mathbb{R}^{n+\frac{n(n+1)}{2}}$, followed by a function that transforms the $n+\frac{n(n+1)}{2}$ dimension vector into the mean and the covariance of a Gaussian distribution. This transformation is detailed in Appendix A.

4.3 Training and test loss from unrolled RNN

In order to train a DAN, we will unroll the RNN defined by $(a_\theta, b_\theta, c_\theta)$ so as to define the training computing from I i.i.d trajectories of $(\boldsymbol{x}_{1:T}, \boldsymbol{y}_{1:T})$. We also define the test loss for training performance evaluation.

To be clear on how the states s_t^{a} and s_t^b depend on a_θ, b_θ and a given trajectory $y_{1:t}$, we will denote the state (memory) at time t informed by the data up to time t-1 and generated using a θ-parametric function as $s_{t|t\text{-}1}^{b,\theta}$. Then we can rewrite s_t^b and s_t^{a} more explicitly as:

$$s_{t|t\text{-}1}^{b,\theta} = b_\theta\left(s_{t\text{-}1|t\text{-}1}^{\mathrm{a},\theta}\right), \quad and \quad s_{t|t}^{\mathrm{a},\theta} = a_\theta\left(s_{t|t\text{-}1}^{b,\theta}, y_t\right), \tag{20}$$

where $s_{0|0}^{\mathrm{a},\theta}=s_0$ is an initial memory of RNN independent of θ. The procoder c_θ outputs the pdf

$$q_{t|t\text{-}1}^{b,\theta}(\cdot|y_{1:t-1}) = c_\theta\left(s_{t|t\text{-}1}^{b,\theta}\right), \quad and \quad q_{t|t}^{\mathrm{a},\theta}(\cdot|y_{1:t}) = c_\theta\left(s_{t|t}^{\mathrm{a},\theta}\right). \tag{21}$$

To define the training loss computed from the I trajectories, we introduce a *trajectory-dependent* loss function which will be needed to define our online training strategy. Let $\left(x_{1:T}^{(i)}, y_{1:T}^{(i)}\right)$ be the i-th trajectory, we write the loss function for the i-th trajectory as:

$$J_t^{(i)}\left(q_{t|t\text{-}1}^{b,\theta}, q_{t|t}^{\mathrm{a},\theta}\right) = -\log q_{t|t\text{-}1}^{b,\theta}\left(x_t^{(i)}|y_{1:t-1}^{(i)}\right) - \log q_{t|t}^{\mathrm{a},\theta}\left(x_t^{(i)}|y_{1:t}^{(i)}\right).$$

The training loss is defined accordingly as a function of θ,

$$\frac{1}{TI}\sum_{t=1}^{T}\sum_{i=1}^{I} J_t^{(i)}\left(q^{b,\theta}_{t|t\text{-}1}, q^{\mathrm{a},\theta}_{t|t}\right) \tag{22}$$

We define the test loss $J(\theta)$, as in (22), by using another I independent trajectories of $(\boldsymbol{x}_{1:T}, \boldsymbol{y}_{1:T})$. It allows one to evaluate how well the DAN learns the underlying dynamics of ODS beyond the training trajectories.

4.4 Online training algorithm: TBPTT

Direct optimization of the training loss in (22) is impractical for large-scale problems since the gradient backpropagation through time generates a large computational graph that consumes a lot of memory (Jaeger, 2002). This limits the time length T and batch size I which, in turn, might lead to overfitting due to limited data. A workaround is to resort to gradient descent with truncated backpropagation through time (TBPTT, (Williams & Peng, 1990; Williams & Zipser, 1995)). It is commonly used in the machine learning community to train recurrent neural networks (Tang & Glass, 2018; Aicher et al., 2020).

Starting from θ_0, the TBPTT is an online method which generates a sequence of model parameters θ_k for $k = 1, 2, \cdots, T$. Each θ_k is obtained from θ_{k-1} based on the information of I training trajectories $\{(x_k^{(i)}, y_k^{(i)})\}_{i\leq I}$ on-the-fly.

More precisely, given the initial memories $\{\bar{s}_0^{(i)}\}_{i\leq I}$ and θ_0, we update the memory

$$\bar{s}_k^{(i)} = a_{\theta_{k-1}}(b_{\theta_{k-1}}(\bar{s}_{k-1}^{(i)}), y_k^{(i)}), \quad k \geq 1$$

and then we perform the following gradient update,

$$\theta_{k+1} = \theta_k - \eta_k \frac{1}{I}\sum_{i=1}^{I} \nabla_\theta J_{k+1}^{(i)}(c_\theta \cdot b_\theta(\bar{s}_k^{(i)}), c_\theta \cdot a_\theta(b_\theta(\bar{s}_k^{(i)}), y_{k+1}^{(i)}))|_{\theta=\theta_k} \tag{23}$$

where η_k is the learning rate. The gradient is computed over the I training trajectories at time $k+1$. As a result, the optimization is not anymore limited in time due to computer memory constraints.

To adjust the learning rate η_k adaptively, we apply the Adam optimizer (Kingma & Ba, 2014) to the gradient in (23). This simultaneously adjusts the updates of θ_k based on an average gradient computed from the gradients at previous steps.

5 Numerical experiments

In this section, we present results of DAN on the Lorenz-95 system (Lorenz, 1995) using the Gaussian conditional posteriors presented in Theorem 2. We first explain Lorenz dynamics in Section 5.1, and provide experimental details in Section 5.2. Then, Section 5.3 evaluates the effectiveness of the online training method TBPTT to minimize the test loss (defined in Section 4.3). Section 5.4 compares standard rmses performance of DAN to a state-of-the-art DA method IEnKF-Q using a limited ensemble memory. We further study the robustness of DAN in terms of its performance on future sequences beyond the horizon T of the training sequences, as well as its sensitivity to the initial distribution of x_1.

5.1 The Lorenz-95 system

The Lorenz-95 system introduced by Lorenz (1995) contains n variables $x_i, i = 1, \ldots, n$ and is governed by the n equations:

$$\frac{dx_i}{dt} = -x_{i-2}x_{i-1} + x_{i-1}x_{i+1} - x_i + F. \tag{24}$$

In Eq. (24) the quadratic terms represent the advection that conserves the total energy, the linear term represents the damping through which the energy decreases, and the constant term represents external forcing keeping the total energy away from zero. The n variables may be thought of as values of some atmospheric quantity in n sectors of a latitude circle.

In this study, we take $n = 40$ and $F = 8$ which results in some chaotic behaviour. The boundary conditions are set to be periodic, i.e., $x_0 = x_{40}$, $x_{-1} = x_{39}$ and $x_{41} = x_1$. The equations are solved using the fourth-order Runge-Kutta scheme, with $\Delta t = 0.05$ (a 6 hour time step).

5.2 Experiment setup

We study the performance of DAN when trained to map to Gaussian posteriors, i.e. the procoder c function is given by (19). This is compared to a state-of-art method of EnKF. For comparison, we implemented the Iterative EnKF with model error (IEnKF-Q (Sakov et al., 2018)), which handles non-linearities better and accepts additive model error.

A batch of I trajectories of $x \in \mathbb{R}^{40}$ is simulated from the resolvant (propagation operator) $\mathcal{M} : \mathbb{R}^{40} \to \mathbb{R}^{40}$ of the 40 dimensional Lorenz-95 system. To start from a stable regime, we use a burning phase which propagates an initial batch of states $\{x^{(i)}_{\text{init}}\}_{i \leq I}$ for a fixed number of cycles. The initial states are drawn independently from $\mathcal{N}(3 \times 1_{40}, I_{40})$. The operator $\mathcal{M}$ is then applied 10^3 times (burning time) to the given initial batch of states (Sakov et al., 2018). The resulting states are taken as the initial state $x^{(i)}_1$.

After the burning phase, the Gaussian propagation errors $\{\eta^i_t\}$, sampled independently from $\mathcal{N}(0_{40}, 0.01 \times I_{40})$, are added each subsequent propagation to get the state trajectories

$$x^{(i)}_{t+1} = \mathcal{M}\left(x^{(i)}_t\right) + \eta^{(i)}_t,$$

Then the Gaussian errors $\varepsilon^{(i)}_{t+1}$, sampled independently from $\mathcal{N}(0_{40}, I_{40})$, are added to the observation operator evaluations to get a training batch of observation trajectories

$$y^{(i)}_{t+1} = \mathcal{H}\left(x^{(i)}_{t+1}\right) + \varepsilon^{(i)}_{t+1}.$$

In the numerical experiments we assume that the system is fully observed, i.e. $\mathcal{H}$ is taken to be an identity operator.

The functions a and b in the cost function of DAN are constructed by $L = 20$ fully connected layers with residual connections (as detailed in Section 4). We consider different number of ensembles for EnKF, i.e. $m \in \{5, 10, 20, 30\}$, which requires m-by-n memory size. To make DAN comparable to EnKF we chose the memory space, i.e. $\mathbb{S} = \mathbb{R}^{m \times n}$.

Across all these m, DAN is trained with a batch size of $I = 1024$ of training samples for $T = 6 \times 10^5$ cycles. The initial learning rate η_0 for the TBPTT is set to be 10^{-4}. The initial memory s_0 of the RNN is set to be zero, while the initial parameter θ_0 of the RNN is mostly set to be random. More precisely, we use the standard random initialization for the weights (W, b) of each linear layer implemented in the Pytorch software. The initial weight α_ℓ in (18) is set to zero for each ℓ.

5.3 Training performance of TBPTT

To show the effectiveness of the training method TBPTT specified in (23), we evaluate the test loss $J(\theta)$ using $I = 1024$ i.i.d samples, on a sub-sequence of θ_k. This allows one to access whether the online method is effective to minimize the total loss $\mathcal{J}(q)$ in (12).

The test loss $J(\theta_k)$ changes over iteration k are displayed in Figure 4. We observe that the minimal loss decreases as m increases, suggesting that the performance of DAN is improved with the memory size. Moreover, we find that the test loss decreases during

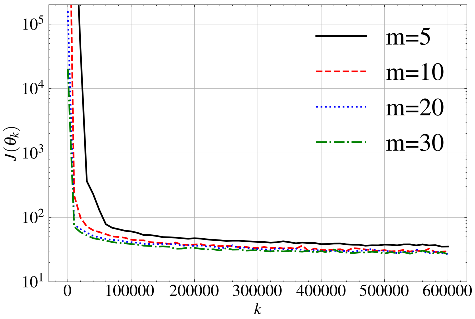


Figure 4: The test loss evaluated at training iterations θ_k of TBPTT, using various memory size m of $\mathbb{S}$ in DAN.

the training process, which shows that TBPTT implicitly minimizes the test loss $J(\theta)$. In theory, we expect that this to happen for a suitable large memory size m because it is proportional to the capacity of the neural networks used in DAN: a larger m implies a better approximation of the posterior distributions due to the universal approximation property of neural networks. The trade-off is that a too large m may lead to over-fitting in machine learning, as we use only I finite trajectories of (x_t, y_t) in the training algorithm.

5.4 Performance of DAN

After DAN is trained, new observation trajectories are generated from a new unknown state trajectory. These testing observations together with a null initial memory vector are then given as input of the trained DAN in a test phase and its outputs are compared with the unknown state.

To evaluate the accuracy of the trained DAN ($k = T$), we compute the accuracy of the mean μ_t^{a} (resp. μ_t^b) of $q_{t|t}^{\mathrm{a},\theta_T}(\cdot|y_{1:t})$ (resp. $q_{t|t-1}^{b,\theta_T}(\cdot|y_{1:t-1})$), evaluated on a test sequence $(x_{1:T}, y_{1:T})$. A standard evaluation in DA is to compute rmses, i.e. for $1 \leq t \leq T$, we compute the following normalized posterior and prior rmses,

$$R_t^{\mathrm{a}} = \frac{1}{\sqrt{n}}\|x_t - \mu_t^{\mathrm{a}}\|, \quad R_t^b = \frac{1}{\sqrt{n}}\|x_t - \mu_t^b\|$$

In Table 1 and 2, we compare the averaged rmses of DAN over t with IEnKF-Q of various ensemble size smaller than the dimension n of x_t.. Recall that we use the same size m to define the memory space $\mathbb{S} = \mathbb{R}^{m\times n}$ in DAN.

For DAN, we report an averaged rmses over t, computed at the parameter θ_T at the last step of training. These rmses are compared to the baseline method IEnKF-Q over the same range of t. When m is small, IEnKF-Q performs worse than DAN, due to sampling errors when m is too small. Note that with the choice $F = 8$ in the Lorenz-95 dynamics (Eq. (24), the model has 13 positive Lypapunov exponents, i.e. the dimension of the unstable subspace is 13 (Asch et al., 2016; Sakov et al., 2018). Therefore, when the model is propagated through time small perturbations grow along these directions. This explains why IEnKF-Q does not perform well when $m \leq 13$, as a result we need to apply localisation and inflation techniques to reduce these sampling errors. When m becomes closer to n (e.g. $m = 20, 30$), we find that the posterior and prior rmses of DAN and IEnKF-Q are similar. This tendency of rmses as a function the ensemble size m is strongly correlated with the smallest test loss achieved by DAN in Figure 4. Note that we use IEnKF-Q with inflation but without any localisation. When IEnKF-Q is used with localisation we expect to get similar performance compared with DAN for each value of m (Asch et al., 2016). With these experiments we

m	5	10	20	30
DAN	**0.401**	**0.388**	**0.376**	0.376
IEnKF-Q	3.939	2.798	0.413	0.355

Table 1: Time averaged posterior (filtering) rmses $\frac{1}{T}\sum_{t=1}^{T} R_t^a$ using various ensemble size m. Bold numbers indicate the performance of DAN better than the baseline.

m	5	10	20	30
DAN	**0.453**	**0.436**	**0.423**	0.423
IEnKF-Q	4.021	2.920	0.460	0.399

Table 2: Time averaged prior (prediction) rmses $\frac{1}{T}\sum_{t=1}^{T} R_t^b$ with various ensemble size m. Bold numbers indicate the performance of DAN better than the baseline.

m	5	10	20	30
DAN	0.400	0.388	0.377	0.376
IEnKF-Q	3.941	2.785	0.412	0.356

Table 3: Time averaged posterior (filtering) rmses $\frac{1}{T}\sum_{t=T+1}^{2T} R_t^a$ with various ensemble size.

want to show that DAN does not require regularization technique when less information is used unlike EnKF.

5.5 Predictive performance and sensitivity to initialization

As DAN is trained on the time interval $t \leq T$, it remains important to evaluate its predictive performance by considering how well it performs for $t > T$. Such performance can be measured by the average rmses over $T+1 \leq t \leq 2T$ instead of over $1 \leq t \leq T$, evaluated using the trained model parameter $\theta = \theta_T$). The comparison with the baseline in terms of the posterior rmses is given in Table 3. We find that the rmses over $T+1 \leq t \leq 2T$ are close to those over $1 \leq t \leq T$ (c.f. Table 1 and 2). This suggests that DAN has learnt the dynamics of the Lorenz system in order to perform well on future trajectories.

All the earlier results are concerned of the performance of DAN under a fixed burning time. Using this burning time for the training of DAN, we further evaluate the rmses on test sequences which have a different burning time. It allows us to indirectly access how well recurrent structures inherited from the ODS are learnt. The results of the ensemble size $m = 20$ are given in Table 4. It shows that the performance of IEnKF-Q and DAN are not sensitive to the distribution of the test sample x_1 initialized over a wide range of burning time.

We remark that among all the simulations, there is always a relatively large error in R_t^a and R_t^b for small t then it decreases very quickly (e.g. $m = 20, \text{burning} = 1000$, both R_t^a and R_t^b get close to a constant level when $t \geq 20$). This transition is needed for DAN to enter a stable regime because the initial memory of the RNN is set to zero.

burning time	10^1	10^3	10^5	10^7
DAN	0.376	0.376	0.377	0.377
IEnKF-Q	0.414	0.413	0.414	0.413

Table 4: Time averaged posterior (filtering) rmses $\frac{1}{T}\sum_{t=1}^{T} R_t^a$ with various burning time.

6 Conclusions

Based on the key observation that the analysis and propagation steps of DA consist in applying time-invariant transformations a and b that update the pdfs using incoming observations, we propose a general framework DAN which encompasses well-known state-of the art methods as special cases. We have shown that by optimizing suitable likelihood-based objective functions, the underlying posterior densities represented by these transformations have the capacity to approximate the optimal posterior densities of BDA. By representing a and b as neural networks, the estimation problem takes the form of the minimization of a loss with respect to the parameters of an extended Elman recurrent neural network.

Our numerical results on a 40-dimensional chaotic Lorenz-95 system show comparable performance compared to a state-of-the-art ensemble technique. We also find that DAN is robust in terms of its predictive performance and its initialization. It indicates that the DAN framework has the advantage of avoiding some problem-dependent numerical-tuning techniques. Although we use a Gaussian approximation of the posterior densities in the procoder c, it can still happen that the memory space $\mathbb{S}$ may encode non-Gaussian information of the posterior distributions. To analyze why DAN can handle problems with nonlinear dynamics (even in other nonlinear dynamical systems, or with partially observed system) is left as a future study.

References

Aicher, C., Foti, N. J., & Fox, E. B. (2020). Adaptively Truncating Backpropagation Through Time to Control Gradient Bias. In *Proceedings of the 35th uncertainty in artificial intelligence conference* (Vol. 115, pp. 799–808).

Asch, M., Bocquet, M., & Nodet, M. (2016). *Data assimilation: methods, algorithms, and applications.* SIAM.

Bachlechner, T., Majumder, B. P., Mao, H. H., Cottrell, G. W., & McAuley, J. (2020). ReZero is All You Need: Fast Convergence at Large Depth. *arXiv preprint arXiv:2003.04887*.

Bing, X., Naiyan, W., Tianqi, C., & Mu, L. (2015). Empirical Evaluation of Rectified Activations in Convolutional Network. *arXiv preprint arXiv:1505.00853*.

Bocquet, M. (2011). Ensemble Kalman filtering without the intrinsic need for inflation. *Nonlinear Processes in Geophysics*, *18*(5), 735–750.

Bocquet, M., Brajard, J., Carrassi, A., & Bertino, L. (2019). Data assimilation as a learning tool to infer ordinary differential equation representations of dynamical models. *Nonlinear Processes in Geophysics*, *26*, 143–162.

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*(1), 55–80.

Bogachev, V. I. (2007). *Measure Theory* (No. vol.∼1). Springer Berlin Heidelberg.

Brajard, J., Carassi, A., Bocquet, M., & Bertino, L. (2020). Combining data assimilation and machine learning to emulate a dynamical model from sparse and noisy observations: a case study with the Lorenz 96 model. *Arxiv preprint arXiv:2001.01520*.

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

Carrassi, A., Bocquet, M., Bertino, L., & Evensen, G. (2018). Data assimilation in the geosciences: An overview of methods, issues, and perspectives. *WIREs Climate Change*, *9*(5), e535.

Cover, T. M., & Thomas, J. A. (2005). *Elements of Information Theory.*

Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, *14*(2), 179–211.

Evensen, G. (2009). *Data Assimilation : The Ensemble Kalman Filter.* Springer-Verlag Berlin Heidelberg.

Fablet, R., Chapron, B., Drumetz, L., Mémin, E., Pannekoucke, O., & Rousseau, F. (2021). Learning Variational Data Assimilation Models and Solvers. *Journal of Advances in Modeling Earth Systems*, *13*(10), e2021MS002572.

Gharamti, M. E. (2018). Enhanced Adaptive Inflation Algorithm for Ensemble Filters. *Monthly Weather Review*, *146*(2), 623–640.

Girin, L., Leglaive, S., Bie, X., Diard, J., Hueber, T., & Alameda-Pineda, X. (2021). Dynamical Variational Autoencoders: A Comprehensive Review. *Foundations and Trends® in Machine Learning*, *15*(1-2), 1–175.

Hamill, T. M., Whitaker, J. S., & Snyder, C. (2001). Distance-Dependent Filtering of Background Error Covariance Estimates in an Ensemble Kalman Filter. *Monthly Weather Review*, *129*(11), 2776–2790.

Harter, F. P., & de Campos Velho, H. F. (2012). Data Assimilation Procedure by Recurrent Neural Network. *Engineering Applications of Computational Fluid Mechanics*, *6*(2), 224–233.

Houtekamer, P. L., & Mitchell, H. L. (2001). A Sequential Ensemble Kalman Filter for Atmospheric Data Assimilation. *Monthly Weather Review*, *129*(1), 123–137.

Jaeger, H. (2002). Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the echo state network approach. *GMD-Forschungszentrum Informationstechnik*, *5*.

Jia, X., Willard, J., Karpatne, A., Read, J. S., Zwart, J. A., Steinbach, M., & Kumar, V. (2021). Physics-Guided Machine Learning for Scientific Discovery: An Application in Simulating Lake Temperature Profiles. *ACM/IMS Trans. Data Sci.*, *2*(3).

Kalman, R. E. (1960). A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*, *82*(1), 35.

Kingma, D., & Ba, J. (2014). Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations*.

Krishnan, R. G., Shalit, U., & Sontag, D. (2015). Deep Kalman Filters. *arXiv preprint arXiv:1511.05121*.

Kullback, S., & Leibler, R. A. (1951). On Information and Sufficiency. *The Annals of Mathematical Statistics*, *22*(1), 79–86.

Le Gland, F., Monbet, V., & Tran, V.-D. (2011). Large sample asymptotics for the ensemble Kalman filter. In D. Crisan & B. Rosovskii (Eds.), *The Oxford handbook of nonlinear filtering* (pp. 598–631). Oxford University Press.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., & Anandkumar, A. (2020). Fourier Neural Operator for Parametric Partial Differential Equations. *arXiv preprint arXiv:2010.08895*.

Lorenz, E. N. (1995). Predictability: a problem partly solved. In *Seminar on predictability, 4-8 september 1995* (Vol. 1, pp. 1–18). Shinfield Park, Reading: ECMWF.

McCabe, J., & Brown, J. (2021). Learning to Assimilate in Chaotic Dynamical Systems. In *Advances in neural information processing systems* (Vol. 34, pp. 12237–12250).

Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2017a). Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations. *arXiv preprint arXiv:1711.10561*.

Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2017b). Physics Informed Deep Learning (Part II): Data-driven Discovery of Nonlinear Partial Differential Equations. *arXiv* *preprint arXiv:1711.10566*.

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.

Revach, G., Shlezinger, N., Ni, X., Escoriza, A. L., van Sloun, R. J. G., & Eldar, Y. C. (2022). KalmanNet: Neural Network Aided Kalman Filtering for Partially Known Dynamics. *IEEE Transactions on Signal Processing*, *70*, 1532–1547.

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

Sakov, P., Haussaire, J.-M., & Bocquet, M. (2018). An iterative ensemble Kalman filter in the presence of additive model error. *Quarterly Journal of the Royal Meteorological Society*, *144*(713), 1297–1309.

Tang, H., & Glass, J. (2018). On Training Recurrent Networks with Truncated Backpropagation Through Time in Speech Recognition. *arXiv preprint arXiv:1807.03396*.

Williams, R. J., & Peng, J. (1990). An Efficient Gradient-Based Algorithm for On-Line Training of Recurrent Network Trajectories. *Neural Computation*, *2*(4), 490–501.

Williams, R. J., & Zipser, D. (1995). Gradient-Based Learning Algorithms for Recurrent Networks and Their Computational Complexity. In *Backpropagation: Theory, architectures, and applications* (pp. 433–486). L. Erlbaum Associates Inc.

Acknowledgments

Appendix A Parameterization of DAN

We use the following parameterization of μ and Λ to convert the vector $v \in \mathbb{R}^{n+\frac{n(n+1)}{2}}$ in (19) into a Gaussian distribution $\mathcal{N}(\mu, \Lambda\Lambda^T)$. Let $v = (v_0, \cdots, v_{n+n(n+1)/2-1})$, we set

$$\mu = \begin{pmatrix} v_0 \\ \vdots \\ v_{n-1} \end{pmatrix} \in \mathbb{R}^n, \tag{A1a}$$

$$\Lambda = \begin{pmatrix} e^{v_n} & 0 & \cdots & 0 \\ v_{2n} & e^{v_{n+1}} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ v_{n+\frac{n(n+1)}{2}-1} & \cdots & v_{3n-2} & e^{v_{2n-1}} \end{pmatrix} \in \mathbb{R}^{\frac{n(n+1)}{2}}. \tag{A1b}$$

The exponential terms in Λ ensure the positive definiteness of $\Lambda\Lambda^T$. This can be easily implemented in Pytorch by using the module `torch.distributions.multivariate_normal:` `MultivariateNormal(loc=`μ`,scale_tril=`Λ`)`.