Data Assimilation Networks

Pierre Boudier, Anthony Fillion, Serge Gratton, Selime Gürol, Sixin Zhang

AGU ADVANCING EARTH AND SPACE SCIENCE

JAMES | Journal of Advances in Modeling Earth Systems®

RESEARCH ARTICLE
10.1029/2022MS003353

Data Assimilation Networks

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

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

Key Points:

- We propose a general framework Data Assimilation Networks (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 functions
- Numerically DAN achieve comparable performance to EnKF methods on Lorenz-95 system, without using explicit regularization such as localization or inflation

Correspondence to:
S. Zhang,
sixin.zhang@irit.fr

Citation:
Boudier, P., Fillion, A., Gratton, S., Gürol, S., & Zhang, S. (2023). Data assimilation networks. *Journal of Advances in Modeling Earth Systems*, *15*, e2022MS003353. https://doi.org/10.1029/2022MS003353

Received 19 AUG 2022
Accepted 31 MAR 2023

Abstract Data Assimilation aims at estimating the posterior conditional probability density functions based on error statistics of the noisy observations and the dynamical system. State of the art methods are sub-optimal due to the common use of Gaussian error statistics and the linearization of the non-linear dynamics. To achieve a good performance, these methods often require case-by-case fine-tuning by using explicit regularization techniques such as inflation and localization. In this paper, we propose a *fully data driven deep learning framework* generalizing recurrent Elman networks and data assimilation algorithms. Our approach approximates a sequence of prior and posterior densities conditioned on noisy observations using a *log-likelihood cost function*. By construction our approach can then be used for general nonlinear dynamics and non-Gaussian densities. As a first step, we evaluate the performance of the proposed approach by using fully and partially observed Lorenz-95 system in which the outputs of the recurrent network are fitted to Gaussian densities. We numerically show that our approach, *without using any explicit regularization technique*, achieves comparable performance to the state-of-the-art methods, IEnKF-Q and LETKF, across various ensemble size.

Plain Language Summary Data Assimilation aims at forecasting the state of a dynamical system by combining information coming from the dynamics and noisy observations. Bayesian data assimilation uses the random nature of a system to predict its states in terms of probability density functions. The calculation of these densities is difficult for non-linear dynamical systems. Practical algorithms compute limited statistics due to computational cost, but this results in sub-optimal DA algorithms which requires then the use of explicit regularization techniques to increase the performance of the algorithm. 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 given observations. Our learning is based on a set of trajectories of the model and observations. It aims to correct the pdfs by optimizing likelihood-based loss functions in the sense of the Kullback-Leibler (KL) divergence. Numerical experiments show that we can obtain similar performance when compared with the IEnKF-Q and LETKF methods, without the need of localization and inflation techniques. These numerical results shows the potential advantage of ML based algorithms when the used practical algorithms are sub-optimal.

1. Introduction

1.1. Context

In Data Assimilation (DA), 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 (Asch et al., 2016). 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) approach, statistical models are assumed to be Gaussian and the physical dynamics are assumed to be linear (Kalman, 1960). Hence, the propagation and analysis steps consist in updating *mean and covariance matrix* of Gaussian densities. A correct estimation of the *covariance matrices* is crucial since they determine to what extent the predictive density will be corrected to

match observations. In the *Ensemble Kalman Filter* (EnKF) approach, the covariance matrices are represented by a *set of sampling vectors* to reduce the computational cost of the filter (Evensen, 2009). When EnKF is used with a small number of ensembles, the covariance matrix estimation becomes low-rank. This causes some spurious correlations in the covariance matrix which are filtered by using regularization techniques such as localization and inflation (Asch et al., 2016; Hamill et al., 2001; Houtekamer & Mitchell, 2001). 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 evaluate the performance of DAN by using fully and partially observed Lorenz-95 system with Gaussian prior and posterior densities in Section 5. The Lorenz-95 system is non-linear and it is often used as a first-step in meteorology to investigate potential applications of the proposed method to high-dimensional chaotic systems. We compare the performance of DAN with state-of-the-art EnKFs methods, IEnKF-Q and LETKF, in terms of root mean square errors, and we also provide the stability analysis with respect to the initial condition and the forecast time-interval beyond the training range. 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; Jia et al., 2021; Li et al., 2020; Raissi et al., 2017a, 2017b, 2019; Rudy et al., 2017). Recently, this research has its significant impact on the design 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 learned 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 albeit within a traditional variational data assimilation framework. A related work to learn a surrogate 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. Penny et al. (2022) learns also a surrogate model, based on recurrent neural networks, by using only state sequence (x_t) which is then used in a deterministic EnKF framework.

All these approaches consider improving the DA methodologies which are based on an *existing DA algorithm*. 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 $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t$ 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 $p(x_{1:t}, y_{1:t}) = p_{\boldsymbol{x}_{1:t},\boldsymbol{y}_{1:t}}(x_{1:t}, y_{1:t})$. We denote the value (realization) of a random variable $\boldsymbol{x}$ as x. The set of pdfs over $\mathbb{X}$ is denoted by $\mathbb{P}_{\mathbb{X}}$. A conditional pdf for $\boldsymbol{x}_t$ conditioned on the value of $\boldsymbol{y}_t$, that is, $\boldsymbol{y}_t = y_t$ is written as $p_{\boldsymbol{x}_t|\boldsymbol{y}_t}(\cdot|y_t) \in \mathbb{P}_{\mathbb{X}}$. Given a function $f(x)$ on a measurable space of $\mathbb{X}$ with measure p, we say $f(x) = 0$ p-almost everywhere x (p-a.e. x in short), when there exists a measurable set A with $p(A) = 0$ such that $f(x) = 0$ for all $x \notin A$.

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:

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

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

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 involve errors and they are represented by an *additive model error*, $\boldsymbol{\eta}_t$, and an *additive observation error*, $\boldsymbol{\varepsilon}_t$. For example, the observation operator may involve spatial interpolations, physical unit transformations and so on, resulting in measurement errors. We assume that these stochastic errors are distributed according to the pdf $p_{\boldsymbol{\eta}}$ and $p_{\boldsymbol{\varepsilon}}$ and they 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 Equation 1a, and $p_{\boldsymbol{y}_t|\boldsymbol{x}_t}(\cdot|x_t) \in \mathbb{P}_{\mathbb{Y}}$ which represents Equation 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 prior 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,

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

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

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

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 Equations 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 operators 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 remedies (see Section 3.3 for further discussions).

2.2. Time-Invariant Structure in the Bayesian Data Assimilation

We review the *invariant structure* of the Bayesian Data Assimilation (BDA) for the Hidden Markov Model (HMM) 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 Equations 1a and 1b, the conditional pdfs $p_{\mathbf{x}_{t+1}|\mathbf{x}_t}$ and $p_{\mathbf{y}_t|\mathbf{x}_t}$ are time invariant, in the sense that for $t = 1, 2, \ldots$

$$p_{\mathbf{x}_{t+1}|\mathbf{x}_t}(u|v) = p_{\mathbf{x}_2|\mathbf{x}_1}(u|v)$$
$$p_{\mathbf{y}_t|\mathbf{x}_t}(y|v) = p_{\mathbf{y}_1|\mathbf{x}_1}(y|v)$$

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

As a result, the conditional pdfs representing the HMM 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 $p_t^a(\cdot|y_{1:t}) \in \mathbb{P}_{\mathbb{X}}$:

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

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

$$p_{t+1}^b(\cdot|y_{1:t}) = b^{BDA}\left[p_t^a(\cdot|y_{1:t})\right].$$

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}, that is, 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 Sections 3 and 4.

3. Data Assimilation Networks (DAN)

In Section 3.1 we present DAN, a general framework for DA, which generalizes traditional data assimilation algorithms such as the Kalman filter and the EnKF detailed in Sections 3.2 and 3.3. Thanks to the repetitive structure of BDA, we propose in Section 3.4, a log-likelihood cost function based on the information loss to approximate conditional pdfs. Instead of calculating the posterior pdfs analytically, DAN aims to learn these pdfs by using sequences of (x_t, y_t) generated from the HMM. We show theoretically that this framework allows one to

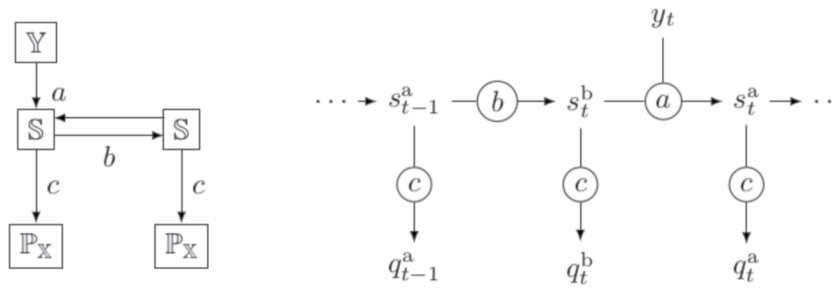

Figure 1. Representation of DAN. Left: scheme of DAN. Right: unrolled DAN along time interval.

handle nonlinear model dynamics and non-Gaussian error distributions where the Bayesian conditional pdfs are not necessarily Gaussian.

3.1. DAN Framework

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

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

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

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

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 1. 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 Sections 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 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 $1 \leq t \leq T$,

$$s^b_t := b\left(s^{\mathrm{a}}_{t-1}\right), \qquad s^{\mathrm{a}}_t := a\left(s^b_t, y_t\right)$$
$$q^b_t := c\left(s^b_t\right), \qquad q^{\mathrm{a}}_t := c\left(s^{\mathrm{a}}_t\right).$$

This recursive application is represented in Figure 1. Note that $\left\{q^b_t\right\}_{t=1}^T$ and $\left\{q^{\mathrm{a}}_t\right\}_{t=1}^T$ are *candidate conditional densities*. This means that for a given sequence of observations $y_{1:t} = (y_1, \ldots, y_t)$, we have $q^b_t(\cdot|y_{1:t-1}) \in \mathbb{P}_{\mathbb{X}}$ and $q^{\mathrm{a}}_t(\cdot|y_{1:t}) \in \mathbb{P}_{\mathbb{X}}$. 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), both the propagation operator M and the observation operator H are assumed to be affine. 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.

Figure 2. Kalman filter mapping diagram. Left: commuting diagram for the KF analysis. Right: commuting diagram for the KF propagation.

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 $s := (\mu, \Sigma)$, 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 $\mathbb{Z}_{\mathbb{X}}$, that is, $a^{\mathrm{KF}} : \mathbb{Z}_{\mathbb{X}} \times \mathbb{Y} \to \mathbb{Z}_{\mathbb{X}}$, given by

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

with $\mu^{\mathrm{a}} = \mu^{\mathrm{b}} + \Sigma^{\mathrm{a}} H^{\mathrm{T}} R^{-1}\left(y - H\left(\mu^{\mathrm{b}}\right)\right)$. When the dimension of the observation y_t is less or equal to the dimension of the state x_t, as an alternative we can obtain $\mu^{\mathrm{a}} = \mu^{\mathrm{b}} + K\left(y - H\left(\mu^{\mathrm{b}}\right)\right)$ and $\Sigma^{\mathrm{a}} = (I - KH)\Sigma^{\mathrm{b}}$ with $K = \Sigma^{\mathrm{b}} H^{\mathrm{T}}\left(H\Sigma^{\mathrm{b}} H^{\mathrm{T}} + R\right)^{-1}$. The mapping diagram for the analysis step of the KF is given by the diagram in Figure 2, which is a commutative diagram. We remind that a diagram is said to commute if any two paths between the same nodes compose to give the same map (Barr & Wells, 1991).

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 $\mathbb{Z}_{\mathbb{X}}$, that is, $b^{\mathrm{KF}} : \mathbb{Z}_{\mathbb{X}} \to \mathbb{Z}_{\mathbb{X}}$, given by

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

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

Unfortunately, the linearity of M and H 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 reduced rank 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 $u = \left(\frac{1}{m}, \ldots, \frac{1}{m}\right)^{\mathrm{T}} \in \mathbb{R}^m, U = \frac{I_m - m \times uu^{\mathrm{T}}}{m-1} \in \mathbb{R}^{m \times m}$ and $I_m \in \mathbb{R}^{m \times m}$ is the identity matrix (Fillion et al., 2020). 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 $m \geq n$. In the case when $m < n$, the error covariance matrix become rank deficient resulting in spurious correlations. In this rank-deficient case, we must select a different base measure where the Gaussian distribution is supported, by using generalized inverse of XUX^{T} (Rao, 1973).

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

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

where $K = X_{\mathrm{b}} U Y_{\mathrm{b}}^{\mathrm{T}} \left(Y_{\mathrm{b}} U Y_{\mathrm{b}}^{\mathrm{T}} + R\right)^{-1} \in \mathbb{R}^{n \times d}$ is the ensemble Kalman gain, $Y_{\mathrm{b}} = \mathcal{H}(X_{\mathrm{b}}) \in \mathbb{Y}^m$ and $Y \in \mathbb{Y}^m \left(= \mathbb{R}^{d \times m}\right)$ 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

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

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

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 (Gharamti, 2018; Hamill et al., 2001; Houtekamer & Mitchell, 2001). 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. 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, that is, 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* $q = \left(q_t^b, q_t^{\mathrm{a}}\right)_{t=1}^T \in \mathbb{P} = \left(\Pi_{t=1}^T \mathbb{Y}^{t-1} \to \mathbb{P}_{\mathbb{X}}\right) \times \left(\Pi_{t=1}^T \mathbb{Y}^t \to \mathbb{P}_{\mathbb{X}}\right)$ *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)*

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

The total log-likelihood cost function is defined as

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

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

Theorem 1. *Let* $\bar{q} \in \arg\min_{q \in \mathbb{P}} \mathcal{J}(q)$, *then* $\forall t \in \{1, \ldots, T\}$, $\bar{q}_t^b(x|y_{1:t-1}) = p_t^b(x|y_{1:t-1})$ *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. See Appendix A.

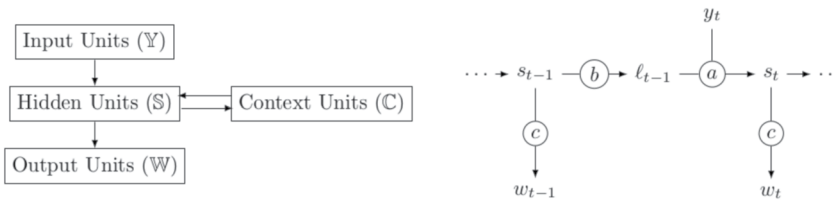

Figure 3. Representation of an Elman Network. Left: scheme of an EN. Right: unrolled EN along time interval.

Theorem 1 shows that the objective function of DAN can approximate the Bayesian prior and posterior cpdf when the candidate pdfs belong to a general functional class (i.e., $q \in \mathbb{P}$). However, the loss function $\mathcal{J}(q)$ can not be numerically computed without making the functional class more specific. As a common specific case, we next consider candidate conditional pdfs as the Gaussian pdfs.

Let $\mathbb{G}_{\mathbb{X}}$ be the set of Gaussian pdfs over $\mathbb{X}$, and $q \in \mathbb{G} = \left(\Pi_{t=1}^{T} \mathbb{Y}^{t-1} \to \mathbb{G}_{\mathbb{X}}\right) \times \left(\Pi_{t=1}^{T} \mathbb{Y}^{t} \to \mathbb{G}_{\mathbb{X}}\right)$. For each $J_t\left(q_t^b, q_t^a\right)$ in Definition 1 to be well-defined, it is necessary to assume that the target prior and posterior distributions $p_t^b(\cdot|y_{1:t-1})$ and $p_t^a(\cdot|y_{1:t})$ have first-order and second-order moments. Under these assumptions, Theorem 2 shows that using Gaussian pdfs, one can match the correct mean and covariance of the target prior and posterior cpdf.

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

Proof. See Appendix B.

Theorem 2 indicates that DAN has the capacity to optimally capture non-linear dynamics in terms of first and second-order statistics. Note that here the optimality is defined with respect to the cost function Equation 12. Contrary to KF-based approaches, DAN never uses Gaussian approximations in its internal computations. DAN fits the output of the recurrent neural network with Gaussian pdfs.

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 a DAN using recurrent neural networks in Section 4.2. Sections 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 an EN is given in Figure 3.

The context units make the Elman network able to process variable length sequences of inputs to produce sequences of outputs as shown in Figure 3. 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 $s_t = a(\ell_{t-1}, y_t) \in \mathbb{S}$. And the function c decodes the hidden state memory into an output $w_t = c(s_t) \in \mathbb{W}$ in the output sequence. The updated hidden state memory is transferred to the context unit via a 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 propagator b (see Figures 1 and 3). DAN can also produce strict future predictions without observations by applying the propagator b multiple times before applying the procoder c. Second, the DAN provides a probabilistic representation of the state that is, 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:

$$a_\theta : \underbrace{\mathbb{S}\times\mathbb{Y}\to\cdots\to\mathbb{S}\times\mathbb{Y}}_{L\ times}\to\mathbb{S}, \tag{13a}$$

$$b_\theta : \underbrace{\mathbb{S}\to\cdots\to\mathbb{S}}_{L\ times}, \tag{13b}$$

The construction of a_θ is built upon L fully connected layers with residual connections. It is based on the LeakyReLU activation function (Bing et al., 2015) 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

$$v_\ell = v_{\ell-1} + \alpha_\ell \mathrm{LeakyReLU}(W_\ell v_{\ell-1} + \beta_\ell). \tag{14}$$

Taking a vector v as its input, the LeakyReLU function outputs a vector w of the same size. For the ith element of w, $w_i = v_i$ if $v_i \geq 0$; $w_i = av_i$ if $v_i < 0$, where a is set to 0.01 by default in our implementation based on Pytorch (Paszke et al., 2019).

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 1, the input a_θ at time t is a concatenation of s_t^b and y_t, that is, $v_0 = \left(s_t^b, y_t\right)$. Similarly, b_θ is constructed from the same L fully connected layers as in Equation 14 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:

$$c_\theta : \mathbb{S}\to\mathbb{R}^{n+\frac{n(n+1)}{2}}\to\mathbb{G}_{\mathbb{X}} \tag{15}$$

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

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 loss computed from I i.i.d trajectories of $(\boldsymbol{x}_{1:T}, \boldsymbol{y}_{1:T})$. We also define the test loss to evaluate the performance of training.

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 $s_{t|t-1}^{b,\theta}$. Then we can rewrite s_t^b and s_t^{a} more explicitly as:

$$s_{t|t-1}^{b,\theta} = b_\theta\left(s_{t-1|t-1}^{\mathrm{a},\theta}\right), \quad \textit{and} \quad s_{t|t}^{\mathrm{a},\theta} = a_\theta\left(s_{t|t-1}^{b,\theta}, y_t\right), \tag{16}$$

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

$$q_{t|t-1}^{b,\theta}(\cdot|y_{1:t-1}) = c_\theta\left(s_{t|t-1}^{b,\theta}\right), \quad \textit{and} \quad q_{t|t}^{\mathrm{a},\theta}(\cdot|y_{1:t}) = c_\theta\left(s_{t|t}^{\mathrm{a},\theta}\right). \tag{17}$$

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 $\left(x^{(i)}_{1:T}, y^{(i)}_{1:T}\right)$ be the ith trajectory, we write the loss function for the ith trajectory as:

$$J^{(i)}_t\left(q^{b,\theta}_{t|t\text{-}1}, q^{\mathrm{a},\theta}_{t|t}\right) = -\log q^{b,\theta}_{t|t\text{-}1}\left(x^{(i)}_t | y^{(i)}_{1:t-1}\right) - \log q^{\mathrm{a},\theta}_{t|t}\left(x^{(i)}_t | y^{(i)}_{1:t}\right).$$

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

$$\frac{1}{TI}\sum_{t=1}^{T}\sum_{i=1}^{I} J^{(i)}_t\left(q^{b,\theta}_{t|t\text{-}1}, q^{\mathrm{a},\theta}_{t|t}\right) \tag{18}$$

We define the test loss $J(\theta)$, as in Equation 18, by using another I independent trajectories of $(\boldsymbol{x}_{1:T}, \boldsymbol{y}_{1:T})$. It allows one to evaluate how well a DAN learns the underlying dynamics of HMM beyond the training trajectories.

4.4. Online Training Algorithm: TBPTT

Direct optimization of the training loss in Equation 18 is impractical for large-scale problems since to compute the gradient of the loss, with back-propagation through time, it requires a large computational graph that consumes a lot of memory (Jaeger, 2002). This limits the training data size TI 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 (Aicher et al., 2020; Tang & Glass, 2018).

Starting from θ_0, TBPTT is an online method which generates a sequence of model parameters θ_k for $k = 1, 2, \ldots, T$. Instead of computing the gradient of the loss Equation 18 with respect θ which depends on time from 1 to T, the idea of TBPTT is to truncate the computation at each iteration k by considering only a part of the gradient from time $k-1$ to k. Each θ_k is obtained from θ_{k-1} based on the information of I training trajectories $\left\{\left(x^{(i)}_k, y^{(i)}_k\right)\right\}_{i\le I}$ on-the-fly.

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

$$\bar{s}^{(i)}_k = a_{\theta_{k-1}}\left(b_{\theta_{k-1}}\left(\bar{s}^{(i)}_{k-1}\right), y^{(i)}_k\right), \quad k \ge 1$$

and then we perform the following gradient update,

$$\theta_{k+1} = \theta_k - \eta_k \frac{1}{I}\sum_{i=1}^{I} \nabla_\theta J^{(i)}_{k+1}\left(c_\theta \cdot b_\theta\left(\bar{s}^{(i)}_k\right), c_\theta \cdot a_\theta\left(b_\theta\left(\bar{s}^{(i)}_k\right), y^{(i)}_{k+1}\right)\right)|_{\theta=\theta_k} \tag{19}$$

where η_k is the learning rate. The learning rate is also called the step size in optimization. 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 Equation 19. 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. Section 5.4 compares standard rmses performance of DAN to state-of-the-art DA methods IEnKF-Q and LETKF 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 each trajectory.

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:

$$\frac{dx_i}{dt} = -x_{i-2}x_{i-1} + x_{i-1}x_{i+1} - x_i + F. \tag{20}$$

In Equation 20 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 behavior. The boundary conditions are set to be periodic, that is, $x_0 = x_{40}$, $x_{-1} = x_{39}$, and $x_{41} = x_1$. The equations are solved using the fourth-order Runge-Kutta scheme, with $\Delta t = 0.05$ (a 6 hr time step).

5.2. Experiment Setup

We study the performance of DAN when trained to map to Gaussian posteriors, that is, the procoder c function is given by Equation 15. This is compared to two state-of-art baseline methods of EnKF: Iterative EnKF with additive model error (IEnKF-Q) (Sakov et al., 2018) and Local Ensemble Transform Kalman filter (LETKF) (Hunt et al., 2007).

A batch of I trajectories of $x \in \mathbb{R}^{40}$ is simulated from the resolvent (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 $\left\{x_{\text{init}}^{(i)}\right\}_{i \le I}$ 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 $x_1^{(i)}$.

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

$$x_{t+1}^{(i)} = \mathcal{M}\left(x_t^{(i)}\right) + \eta_t^{(i)},$$

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

$$y_{t+1}^{(i)} = \mathcal{H}\left(x_{t+1}^{(i)}\right) + \varepsilon_{t+1}^{(i)}.$$

In the numerical experiments we consider two cases for the observation network: (a) fully observed, that is, $\mathcal{H}$ is taken to be the identity operator I, and (b) partially observed, that is, $\mathcal{H}$ is taken as a uniform selection operator H_0. For any 40-dimensional vector x, the vector $H_0 x$ preserves half of the grid of x, by removing even-indexed elements of x. It is left as a future work to study cases where H is a nonlinear operator.

5.2.1. Setup of Baseline

The baseline methods, IEnKF-Q and LETKF, are implemented with explicit inflation or localization regularization in order to obtain a good estimation of the covariance matrix of Gaussian densities. Such regularization is often critical to the final performance of EnKF methods, and it often requires the tuning of hyper-parameters whenever the ensemble size m is changed (Asch et al., 2016).

To illustrate the sensitivity to the hyper-parameter tuning, we provide two set of experiments for LETKF: (a) (with case-by-case turning) The filter for each ensemble size is run with the best performance values provided in Table 1, found by a 2D grid search for each m, named as LETKF*. (b) (without case-by-case tuning) The filter for each ensemble is run with the best performance obtained at $m = 20$, that is, the grid search is only run for this m and the obtained optimal hyper-parameters are used for all m. We name these experiments simply as LETKF.

We implemented the IEnKF-Q which uses only inflation regularization. This allows one to measure the effect of using both inflation and localization regularization in LETKF. We present results without case-by-case tuning

Table 1
Optimal Hyper-Parameter Values of LETKF Across Various Ensemble Size m Found by 2D Grid Search

m	5	10	20	30	m	5	10	20	30
Inflation	1.1	1.07	1.04	1.03	Inflation	1.1	1.04	1.03	1.02
Local. Radius	1	2	4	4	Local. Radius	2	2	4	4

Note. Left: fully observed case ($H = I$). Right: partially observed case ($H = H_0$).

across different number of ensembles for EnKF, that is, $m \in \{5, 10, 20, 30\}$. As we do not have localization in IEnKF-Q, we fine-tune the inflation hyper-parameter of this method at $m = 20$ using grid-search. We find that on both fully observed and partially observed cases, a common inflation parameter 1.1 is close to be optimal among $\{1.0, 1.02, 1.03, 1.04, 1.07, 1.08, 1.09, 1.1, 1.2, 1.3, 2.0\}$, according to the time-averaged posterior (filtering) rmses (see the definition of the rmses in Section 5.4).

Experiments with the LETKF are performed by using an open source code: DAPPER (Raanes et al., 2022, version 1.2.1). For each ensemble, we have performed 2D grid search. Localization radius is chosen from the set $\{1, 2, 4\}$ and the inflation hyper-parameter is chosen from the set $\{1.02, 1.03, 1.04, 1.07, 1.1\}$. We also use rotation after the analysis step which is shown to provide better performance for LETKF (Sakov & Oke, 2008). The inflation and localization radius hyper-parameter values that provide the best performance according to the time-averaged posterior (filtering) rmses are given in Table 1.

5.2.2. Setup of DAN

To make DAN comparable to EnKF in terms of the used memory, we set the memory space $\mathbb{S} = \mathbb{R}^{m \times n}$. Similar to the results without case-by-case turning in LETKF and IEnKF-Q, hyper-parameters of DAN are only tuned at $m = 20$, and then fixed across all m.

Across $m \in \{5, 10, 20, 30\}$, DAN is trained with a batch size of $I = 1{,}024$ 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. To train a neural network with a large number of layers L, we use the ReZero trick (Bachlechner et al., 2020) which sets the initial weight α_ℓ in Equation 14 to be zero for each ℓ. 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).

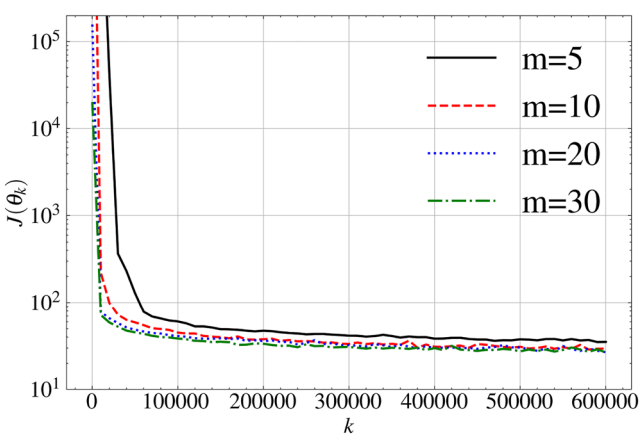

Figure 4. The test loss evaluated at training iterations θ_k of TBPTT, using various memory size m in DAN.

5.3. Training Performance of TBPTT

To show the effectiveness of the training method TBPTT specified in Equation 19, we evaluate the test loss $J(\theta)$ using $I = 1{,}024$ i.i.d samples (defined in Section 4.3), 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 Equation 12. The training time of DAN grows with T but it is not sensitive to the choice of I. This is because our current implementation runs on GPU graphics cards, which allows the computation over I training samples to be in parallel. However, the sequential computation of TBPTT can not be done in parallel. One potential improvement of the running time is to use a modified version of TBPTT to improve the convergence rate, as suggested in (Chen et al. (2022), Algorithm 4.2).

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 the training process, which shows that TBPTT implicitly minimizes the test loss $J(\theta)$. In theory, we expect 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

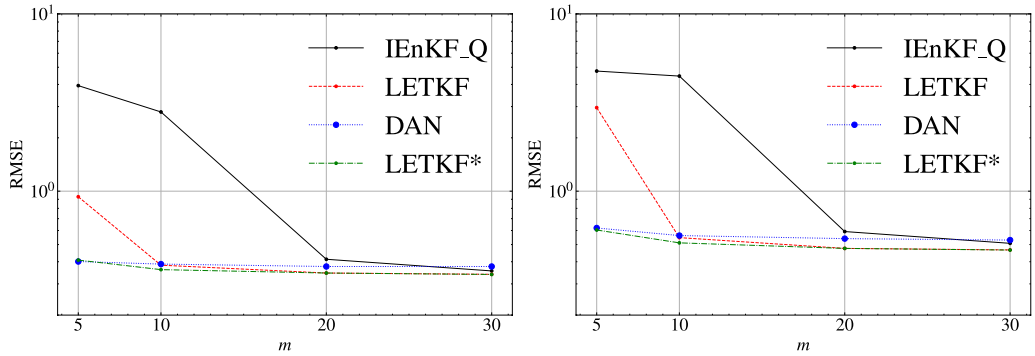

Figure 5. Time averaged posterior (filtering) rmses $\frac{1}{T}\sum_{t=1}^{T} R_t^a$ using various ensemble size m. Left: fully observed case ($H = I$). Right: partially observed case ($H = H_0$). DAN, IEnKF-Q, and LETKF are tuned at $m = 20$; LETKF* is tuned at each m.

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 (i.e., a large gap between the training loss and test loss), 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 y_t are generated from a new unknown state trajectory x_t. 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 x_t.

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, that is, for $1 \le t \le T$, we compute the following normalized posterior and prior rmses,

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

In Figures 5 and 6, we compare the averaged rmses of DAN with IEnKF-Q and LETKF when the ensemble size m is smaller than the dimension n of the state x_t. 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 two baseline methods, IEnKF-Q and LETKF, over the same range of t. Recall that we use the same size m to define the memory space $\mathbb{S} = \mathbb{R}^{m\times n}$ in DAN.

Let us first analyze the numerical results for the fully observed case. When m is small, IEnKF-Q performs worse than DAN, due to sampling errors. Note that with the choice $F = 8$ in the Lorenz-95 dynamics (Equation 20),

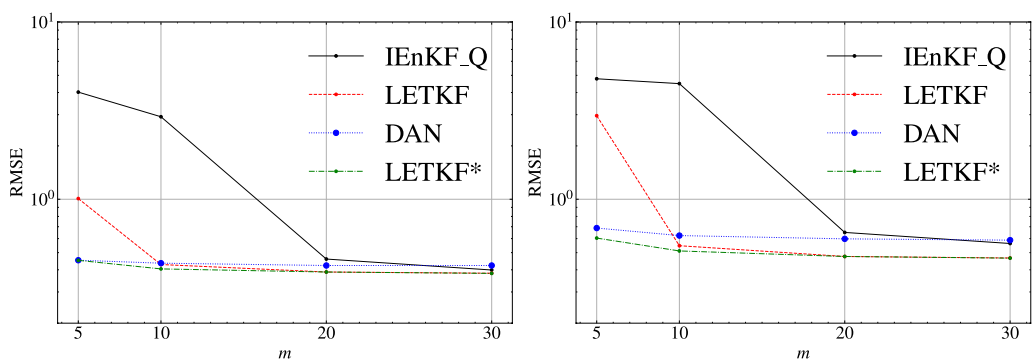

Figure 6. Time averaged prior (prediction) rmses $\frac{1}{T}\sum_{t=1}^{T} R_t^b$ using various ensemble size m. Left: fully observed case ($H = I$). Right: partially observed case ($H = H_0$). DAN, IEnKF-Q, and LETKF are tuned at $m = 20$; LETKF* is tuned at each m.

Table 2
Time Averaged Posterior (Filtering) rmses $\frac{1}{T}\sum_{t=T+1}^{2T} R_t^a$ *With Various Ensemble Size*

m	5	10	20	30
DAN	0.400	0.388	0.377	0.376

the model has 13 positive and one neutral Lypapunov exponents, that is, the dimension of the unstable-neutral subspace is 14 (Bocquet & Carrassi, 2017; Carrassi et al., 2022; Sakov et al., 2018; Trevisan et al., 2010). Therefore, when the model is propagated through time, small perturbations grow along these directions (Carrassi et al., 2022). This explains why IEnKF-Q does not perform well when $m \leq 14$, as a result we need to apply localisation and inflation techniques to reduce these sampling errors. As expected, LETKF and LETKF*, in which localization and inflation techniques are applied with turned parameter values, performs much better than the IEnKF-Q. DAN performs similarly. When $m = 5$, it is slightly better than LETKF* in the fully observed case. LETKF performs much worse than LETKF* and DAN, showing how sensitive the method is to the tuning of the inflation and localization hyper-parameters. When m becomes closer to n (e.g., $m = 20, 30$), we find that the posterior and prior rmses of DAN, IEnKF-Q, LETKF, and LETKF* are similar, with better results for LETKF*. 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. We observe that for the partially observed case, conclusions are similar as well. These experiments clearly show that DAN can achieve a comparable performance without using EnKF-type regularization techniques.

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 posterior rmses for fully observed case are provided in Table 2. We find that the rmses over $T + 1 \leq t \leq 2T$ are close to those over $1 \leq t \leq T$. This suggests that DAN has learned 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 HMM are learned. The results of the ensemble size $m = 20$ are given in Table 3. It shows that the performance of DAN is 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., $m = 20$, burning = 1,000, 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.

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. As a result, this general framework can be used for nonlinear dynamics and non-Gaussian error statistics.

In practice, we need to define the pdfs for the calculation of the loss function. As a first step and to be able to compare performance of DAN with the state-of-the-art ensemble methods, we perform numerical experiments with a procoder c which outputs a Gaussian pdf. Our numerical results on a 40-dimensional chaotic Lorenz-95 system show that when the ensemble size is small, DAN performs similarly compared to LETKF which includes regularization techniques such as localization and inflation. For large ensemble size, DAN has similar performance compared to IEnKF-Q and LETKF.

Table 3
Time Averaged Posterior (Filtering) rmses $\frac{1}{T}\sum_{t=1}^{T} R_t^a$ *With Various Burning Time at Ensemble Size m = 20*

Burning time	10^1	10^3	10^5	10^7
DAN	0.376	0.376	0.377	0.377

It indicates that the DAN framework has the advantage of avoiding some problem-dependent numerical-tuning techniques. We also find that DAN is robust in terms of its predictive performance and its initialization.

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) is left as a future study. From a practical point of view, DAN in its current form is not scalable to perform DA when the dimensionality n is very large (e.g., in the order of 10^9). To make DAN scalable, different training strategies (Chen et al., 2022; Penny et al., 2022) will be considered in the future.

Appendix A: Proof of Theorem 1

Proof. According to Equation 12, it is sufficient to derive the optimal solution of $\mathcal{J}_t\left(q_t^b, q_t^a\right)$ for each t independently. The proof is an application of the KL-divergence (Kullback & Leibler, 1951) to conditional probability densities. We re-write $\mathcal{J}_t\left(q_t^b, q_t^a\right)$ as

$$-\int \ln q_t^b(x_t|y_{1:t-1}) p_t^b(x_t|y_{1:t-1}) p(y_{1:t-1}) dx_t dy_{1:t-1} - \int \ln q_t^a(x_t|y_{1:t}) p_t^a(x_t|y_{1:t}) p(y_{1:t}) dx_t dy_{1:t}, \tag{A1}$$

using the property $p(x_t, y_{1:t-1}) = p_t^b(x_t|y_{1:t-1})p(y_{1:t-1})$ and $p(x_t, y_{1:t}) = p_t^a(x_t|y_{1:t})p(y_{1:t})$. The first term in Equation A1 can be written as a non-negative conditional relative entropy by including a constant conditional entropy term:

$$\int \left(\int \ln \frac{p_t^b(x_t|y_{1:t-1})}{q_t^b(x_t|y_{1:t-1})} p_t^b(x_t|y_{1:t-1}) dx_t \right) p(y_{1:t-1}) dy_{1:t-1} \geq 0. \tag{A2}$$

We have equality in Equation A2 if and only if $q_t^b(x|y_{1:t-1}) = p_t^b(x|y_{1:t-1})$ for $p_t^b(\cdot|y_{1:t-1})$-a.e x, and p-a.e. $y_{1:t-1}$ (see a proof in (Kullback & Leibler (1951), Lemma 3.1) and (Bogachev, 2007, Corollary 2.5.4)). Thus, the minimal solution is given by $\bar{q}_t^b$ as stated in the theorem. Similarly, the minimal solution of the second term Equation A1 is given by the $\bar{q}_t^a$ in the statement.

Appendix B: Proof of Theorem 2

Proof. We shall only provide a proof for $\bar{q}_t^b(\cdot|y_{1:t-1})$ as the proof is similar for $\bar{q}_t^a(\cdot|y_{1:t})$. Let $\bar{p}_t^b(\cdot|y_{1:t-1})$ be the Gaussian distribution which has the mean and covariance of $p_t^b(\cdot|y_{1:t-1})$. Following the proof of Theorem 1, we can rewrite the first term, up to a constant, in Equation A1 into

$$\int \left(\int \ln \frac{\bar{p}_t^b(x_t|y_{1:t-1})}{q_t^b(x_t|y_{1:t-1})} p_t^b(x_t|y_{1:t-1}) dx_t \right) p(y_{1:t-1}) dy_{1:t-1} \tag{B1}$$

This is an equivalent minimization problem because we have added a term of $\bar{p}_t^b$ which does not depend on q_t^b. By definition, $q_t^b(\cdot|y_{1:t-1}) \in \mathbb{G}_{\mathbb{X}}$, $\bar{q}_t^b(\cdot|y_{1:t-1}) \in \mathbb{G}_{\mathbb{X}}$, the logarithm term in Equation B1 is a quadratic function of x_t. As a consequence, we can rewrite Equation B1 as

$$\int \left(\int \ln \frac{\bar{p}_t^b(x_t|y_{1:t-1})}{q_t^b(x_t|y_{1:t-1})} \bar{p}_t^b(x_t|y_{1:t-1}) dx_t \right) p(y_{1:t-1}) dy_{1:t-1} \geq 0. \tag{B2}$$

where we have replaced the density p_t^b by $\bar{p}_t^b$ because they have the same first and second order moments. Note that the inner integral in Equation B2 is the KL divergence between $\bar{p}_t^b$ and q_t^b, so its minimal solution $\bar{q}_t^b(\cdot|y_{1:t-1})$ equals almost surely to $\bar{p}_t^b(\cdot|y_{1:t-1})$. Therefore the mean and covariance of $\bar{q}_t^b(\cdot|y_{1:t-1})$ and $p_t^b(\cdot|y_{1:t-1})$ match for p-a.e. $y_{1:t-1}$.

Appendix C: 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 Equation 15 into a Gaussian distribution $\mathcal{N}\left(\mu, \Lambda\Lambda^T\right)$. Let $v = (v_0, \ldots, v_{n+n(n+1)/2-1})$, we set

$$\mu = \begin{pmatrix} v_0 \\ \vdots \\ v_{n-1} \end{pmatrix} \in \mathbb{R}^n, \tag{C1a}$$

$$\Lambda = \begin{pmatrix} e^{v_n} & 0 & \cdots & 0 \\ v_{2n} & e^{v_{n+1}} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ v_{n+\frac{n(n+1)}{2}-1} & \cdots & v_{3n-2} & e^{v_{2n-1}} \end{pmatrix} \in \mathbb{R}^{\frac{n(n+1)}{2}}. \tag{C1b}$$

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)`.

Data Availability Statement

All the results and data in this paper can be reproduced from a software which is available at https://gitlab.com/aniti-data-assimilation/dan_james. It can be cited at https://doi.org/10.5281/zenodo.7656199.

Acknowledgments
This work is 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. The authors wish to thank the two anonymous referees for their constructive comments which helped to improve the manuscript.

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.
Barr, M., & Wells, C. (1991). Toposes, triples and theories. grundlehren der mathematischen wissenschaften, no. 278. *The Journal of Symbolic Logic*, *56*(1), 340–341. https://doi.org/10.2307/2274934
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. https://doi.org/10.5194/npg-18-735-2011
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*(3), 143–162. https://doi.org/10.5194/npg-26-143-2019
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. https://doi.org/10.3934/fods.2020004
Bocquet, M., & Carrassi, A. (2017). Four-dimensional ensemble variational data assimilation and the unstable subspace. *Tellus A: Dynamic Meteorology and Oceanography*, *69*(1), 1304504. https://doi.org/10.1080/16000870.2017.1304504
Bogachev, V. I. (2007). *Measure theory* (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 of the United States of America*, *113*(15), 3932–3937. https://doi.org/10.1073/pnas.1517384113
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. https://doi.org/10.1002/wcc.535
Carrassi, A., Bocquet, M., Demaeyer, J., Grudzien, C., Raanes, P., & Vannitsem, S. (2022). Data assimilation for chaotic dynamics. In S. K. Park & L. Xu (Eds.), *Data assimilation for atmospheric, oceanic and hydrologic applications* (Vol. iv, pp. 1–42). Springer International Publishing.
Chen, Y., Sanz-Alonso, D., & Willett, R. (2022). Autodifferentiable ensemble Kalman filters. *SIAM Journal on Mathematics of Data Science*, *4*(2), 801–833. https://doi.org/10.1137/21m1434477
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. https://doi.org/10.1207/s15516709cog1402_1
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. https://doi.org/10.1029/2021ms002572
Fillion, A., Bocquet, M., Gratton, S., Gürol, S., & Sakov, P. (2020). An iterative ensemble Kalman smoother in presence of additive model error. *SIAM/ASA Journal on Uncertainty Quantification*, *8*(1), 198–228. https://doi.org/10.1137/19m1244147
Gharamti, M. E. (2018). Enhanced adaptive inflation algorithm for ensemble filters. *Monthly Weather Review*, *146*(2), 623–640. https://doi.org/10.1175/mwr-d-17-0187.1
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. https://doi.org/10.1561/2200000089
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. https://doi.org/10.1175/1520-0493(2001)129<2776:ddfobe>2.0.co;2
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. https://doi.org/10.1080/19942060.2012.11015417
Houtekamer, P. L., & Mitchell, H. L. (2001). A sequential ensemble Kalman filter for atmospheric data assimilation. *Monthly Weather Review*, *129*(1), 123–137. https://doi.org/10.1175/1520-0493(2001)129<0123:asekff>2.0.co;2
Hunt, B. R., Kostelich, E. J., & Szunyogh, I. (2007). Efficient data assimilation for spatiotemporal chaos: A local ensemble transform Kalman filter. *Physica D: Nonlinear Phenomena*, *230*(1), 112–126. (Data Assimilation). https://doi.org/10.1016/j.physd.2006.11.008

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

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 Transactions on Data Science*, *2*(3), 1–26. https://doi.org/10.1145/3447814

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, *82*(1), 35–45. https://doi.org/10.1115/1.3662552

Kingma, D., & Ba, J. (2014). Adam: A method for stochastic optimization. In *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. https://doi.org/10.1214/aoms/1177729694

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* (Vol. 1, pp. 1–18). ECMWF.

McCabe, J., & Brown, J. (2021). Learning to assimilate in chaotic dynamical systems. *Advances in Neural Information Processing Systems*, *34*, 12237–12250.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., et al. (2019). PyTorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, *32*, 8024–8035.

Penny, S. G., Smith, T. A., Chen, T.-C., Platt, J. A., Lin, H.-Y., Goodliff, M., & Abarbanel, H. D. I. (2022). Integrating recurrent neural networks with data assimilation for scalable data-driven state estimation. *Journal of Advances in Modeling Earth Systems*, *14*(3), e2021MS002843. https://doi.org/10.1029/2021ms002843

Raanes, P. N., Chen, Y., Grudzien, C., Tondeur, M., & Dubois, R. (2022). Dapper. Retrieved from https://github.com/nansencenter/DAPPER

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

Rao, C. R. (1973). *Linear statistical inference and its applications: Second edition*. John Wiley and Sons, inc.

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. https://doi.org/10.1109/tsp.2022.3158588

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. https://doi.org/10.1126/sciadv.1602614

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. https://doi.org/10.1002/qj.3213

Sakov, P., & Oke, P. R. (2008). Implications of the form of the ensemble transformation in the ensemble square root filters. *Monthly Weather Review*, *136*(3), 1042–1053. https://doi.org/10.1175/2007mwr2021.1

Tang, H., & Glass, J. (2018). On training recurrent networks with truncated backpropagation through time in speech recognition. arXiv preprint arXiv:1807.03396.

Trevisan, A., D'Isidoro, M., & Talagrand, O. (2010). Four-dimensional variational assimilation in the unstable subspace and the optimal subspace dimension. *Quarterly Journal of the Royal Meteorological Society*, *136*(647), 487–496. https://doi.org/10.1002/qj.571

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. https://doi.org/10.1162/neco.1990.2.4.490

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.