Online model error correction with neural networks in the incremental 4D-Var framework

Alban Farchi[1], Marcin Chrust[2], Marc Bocquet[1], Patrick Laloyaux[2], and Massimo Bonavita[2]

[1]CEREA
[2]ECMWF

January 20, 2023

Abstract

Recent studies have demonstrated that it is possible to combine machine learning with data assimilation to reconstruct the dynamics of a physical model partially and imperfectly observed. The surrogate model can be defined as an hybrid combination where a physical model based on prior knowledge is enhanced with a statistical model estimated by a neural network. The training of the neural network is typically done offline, once a large enough dataset of model state estimates is available. By contrast, with online approaches the surrogate model is improved each time a new system state estimate is computed. Online approaches naturally fit the sequential framework encountered in geosciences where new observations become available with time. In a recent methodology paper, we have developed a new weak-constraint 4D-Var formulation which can be used to train a neural network for online model error correction. In the present article, we develop a simplified version of that method, in the incremental 4D-Var framework adopted by most operational weather centres. The simplified method is implemented in the ECMWF Object-Oriented Prediction System, with the help of a newly developed Fortran neural network library, and tested with a two-layer two-dimensional quasi geostrophic model. The results confirm that online learning is effective and yields a more accurate model error correction than offline learning. Finally, the simplified method is compatible with future applications to state-of-the-art models such as the ECMWF Integrated Forecasting System.

Online model error correction with neural networks in the incremental 4D-Var framework

Alban Farchi[1], Marcin Chrust[2], Marc Bocquet[1], Patrick Laloyaux[2], and Massimo Bonavita[2]

[1]CEREA, École des Ponts and EDF R&D, Île–de–France, France
[2]ECMWF, Shinfield Park, Reading, United Kingdom

Key Points:

- Weak-constraint 4D-Var variants can be used to train neural networks for online model error correction.
- Online learning yields a more accurate model error correction than offline learning.
- The new, simplified method, developed in the incremental 4D-Var framework, can be easily applied in operational weather models.

Corresponding author: Alban Farchi, `alban.farchi@enpc.fr`

Abstract

Recent studies have demonstrated that it is possible to combine machine learning with data assimilation to reconstruct the dynamics of a physical model partially and imperfectly observed. The surrogate model can be defined as an hybrid combination where a physical model based on prior knowledge is enhanced with a statistical model estimated by a neural network. The training of the neural network is typically done offline, once a large enough dataset of model state estimates is available. By contrast, with online approaches the surrogate model is improved each time a new system state estimate is computed. Online approaches naturally fit the sequential framework encountered in geosciences where new observations become available with time. In a recent methodology paper, we have developed a new weak-constraint 4D-Var formulation which can be used to train a neural network for online model error correction. In the present article, we develop a simplified version of that method, in the incremental 4D-Var framework adopted by most operational weather centres. The simplified method is implemented in the ECMWF Object-Oriented Prediction System, with the help of a newly developed Fortran Neural Network library, and tested with a two-layer two-dimensional quasi geostrophic model. The results confirm that online learning is effective and yields a more accurate model error correction than offline learning. Finally, the simplified method is compatible with future applications to state-of-the-art models such as the ECMWF Integrated Forecasting System.

Plain Language Summary

We have recently proposed a general framework for combining data assimilation and machine learning techniques to train a neural network for online model error correction. In the present article, we develop a simplified version of this online training method, compatible with future applications to more realistic models. Using numerical illustrations, we show that the new method is effective and yields a more accurate model error correction than the usual offline learning approach. The results show the potential of incorporating data assimilation and machine learning tightly, and pave the way towards an application to the Integrated Forecasting System used for operational numerical weather prediction at the European Centre for Medium-Range Weather Forecasts.

1 Introduction: machine learning for model error correction

In the geosciences, data assimilation (DA) is used to increase the quality of forecasts by providing accurate initial conditions (Kalnay, 2003; Reich & Cotter, 2015; Law et al., 2015; Asch et al., 2016; Carrassi et al., 2018; Evensen et al., 2022). The initial conditions are obtained by combining all sources of information in a mathematically optimal way, in particular information from the dynamical model and information from sparse and noisy observations. There are two main classes of DA methods. In variational DA, the core of the methods is to minimise a cost function, usually using gradient-based optimisation techniques, to estimate the system state. Examples include 3D- and 4D-Var. In statistical DA, the methods relies on the sampled error statistics to perform sequential updates to the state estimation. The most popular examples are the ensemble Kalman filter (EnKF) and the particle filter.

Most of the time, DA methods are applied with the perfect model assumption: this is called strong-constraint DA. However, despite the significant effort provided by the modellers, geoscientific models remain affected by errors (Dee, 2005), for example due to unresolved small-scale processes. This is why there is a growing interest of the DA community in weak-constraint (WC) methods, i.e. DA methods relaxing the perfect model assumption (Trémolet, 2006). This has led, for example, to the iterative ensemble Kalman filter in the presence of additive noise (Sakov et al., 2018) in statistical DA, and to the forcing formulation of WC 4D-Var (Laloyaux, Bonavita, Chrust, & Gürol, 2020) in variational DA. In practice, the DA control vector has to be extended to include the model error in addition to the system state. The downside of this approach is the potentially significant increase of the problem's dimension since the model trajectory is not anymore described uniquely by the initial condition. By construction, WC 4D-Var is an online model error correction method, meaning that the model error is estimated during the assimilation process and only valid for the states in the current assimilation window.

In parallel, following the renewed impetus of machine learning (ML) applications (LeCun et al., 2015; Goodfellow et al., 2016; Chollet, 2018), data-driven approaches are more and more frequent in the geosciences. The goal of these approaches (*e.g.,* Brunton et al., 2016; Hamilton et al., 2016; Lguensat et al., 2017; Pathak, Hunt, et al., 2018; Dueben & Bauer, 2018; Fablet et al., 2018; Scher & Messori, 2019; Weyn et al., 2019; Arcomano et al., 2020, among many others) is to learn a surrogate of the dynamical model using supervised learning, i.e. by minimising a loss function which measures the discrepancy between the surrogate model predictions and an observation dataset. In order to take into account sparse and noisy observations, ML techniques can be combined with DA (Abarbanel et al., 2018; Bocquet et al., 2019; Brajard et al., 2020; Bocquet et al., 2020; Arcucci et al., 2021). The idea is to take the best of both worlds: DA techniques are used to estimate the state of the system from the observations, and ML techniques are used to estimate the surrogate model from the estimated state. In practice, the hybrid DA and ML methods can be used both for full model emulation and model error correction (Rasp et al., 2018; Pathak, Wikner, et al., 2018; Bolton & Zanna, 2019; Jia et al., 2019; Watson, 2019; Bonavita & Laloyaux, 2020; Brajard et al., 2021; Gagne et al., 2020; Wikner et al., 2020; Farchi, Bocquet, et al., 2021; Farchi, Laloyaux, et al., 2021; Chen et al., 2022). In the first case, the surrogate model is entirely learned from observations, while in the latter case, the surrogate model is hybrid: a physical, knowledge-based model is corrected by a statistical model, e.g. a neural network (NN), which is learned from observations. Even though from a technical point of view it can arguably be more difficult to implement, model error correction has many advantages over full model emulation: by leveraging the long history of numerical modelling, one can hope to end up with an easier learning problem (Watson, 2019; Farchi, Laloyaux, et al., 2021).

Most of the current hybrid DA-ML methods use offline learning strategies: the surrogate model (or model error correction) is learned using a large dataset of observations (or analyses) and should be generalisable to other situations (i.e. outside the dataset). There are two main reasons for this choice. First, surrogate modelling requires a large amount of data to provide accurate results – certainly more than what is available in a single assimilation update with online learning. Second, by doing so, it is possible to use the full potential of the ML variational tools. Nevertheless, online learning has on paper several advantages over offline learning.

- Online learning fits the standard sequential DA approach in the geosciences. Each time a new batch of observations becomes available, the surrogate model parameters can be corrected.
- With online learning, the system state and the surrogate model parameters are jointly estimated, which is often not the case with offline learning. Joint estimation is in general more consistent, and hence potentially more accurate, than separate estimation.
- With offline learning, the training only starts once a sufficiently large dataset is available. With online learning, the training begins from the first batch of observations, which means that improvements can be expected before having a large dataset.
- With online learning, since the surrogate model is constantly updated, it can adapt to new (previously unseen) conditions. An example could be, in the case of model error correction, an update of the physical model to correct. Another example could be slowly-varying effects on the dynamics (e.g., seasonality).

Fundamentally, online learning is very similar to parameter estimation in DA: the goal is to estimate at the same time the system state and some parameters – in this case the surrogate model parameters. Several example of online learning methods have recently emerged. Bocquet et al. (2021) and Malartic et al. (2022) have developed several variants of the EnKF to perform a joint estimation of the state and the parameters of surrogate model which fully emulates the dynamics. Gottwald and Reich (2021) have used a very similar approach for the parameters of an echo state network used as surrogate model. Finally, Farchi, Bocquet, et al. (2021) have derived a variant of WC 4D-Var to perform a joint estimation of the state and the parameters of a NN which correct the tendencies of a physical model. In this article, we revisit the method of Farchi, Bocquet, et al. (2021). A new simplified method is derived, compatible with future applications to more realistic models. The method is implemented in the Object-Oriented Prediction System (OOPS) framework developed at the European Center for Medium-Range Weather Forecasts (ECMWF), and tested using the two-layer quasi-geostrophic channel model developed in OOPS. To us, this is a final step before considering an application with the Integrated Forecasting System (IFS, Bonavita et al., 2017), since the IFS will soon rely on OOPS for its DA part.

The article is organised as follows. Section 2 presents the methodology. The quasi-geostrophic (QG) model is described in section 3. The experimental results are then presented in section 4 for offline learning, and in section 5 for online learning. Finally, conclusions are given in section 6.

2 A simplified neural network variant of weak-constraint 4D-Var

2.1 Strong-constraint 4D-Var

Suppose that we follow the evolution of a system using a series of observations taken at discrete times. In the classical 4D-Var, the observations are gathered into time windows $(\mathbf{y}_0, \ldots, \mathbf{y}_L)$. The integer $L \geq 0$ is the window length, and $\mathbf{y}_k \in \mathbb{R}^{N_\mathsf{y}}$, the k-th batch of observations, contains all the observations taken at time t_k, for $k = 0, \ldots, L$. For convenience, we assume that the time interval between consecutive observation batches $t_{k+1} - t_k = \Delta t$ is constant. This assumption is not fundamental; it just makes the presentation much easier. Within the window, the system state $\mathbf{x}_k \in \mathbb{R}^{N_\mathsf{x}}$ at time t_k is obtained by integrating the model in time from t_0 to t_k:

$$\mathbf{x}_k = \boldsymbol{\mathcal{M}}_{k:0}\left(\mathbf{x}_0\right), \tag{1}$$

where $\boldsymbol{\mathcal{M}}_{k:l} : \mathbb{R}^{N_\mathsf{x}} \to \mathbb{R}^{N_\mathsf{x}}$ is the resolvent of the dynamical (or physical) model from t_l to t_k. Moreover, the observations are related to the state by the observation operator $\boldsymbol{\mathcal{H}}_k : \mathbb{R}^{N_\mathsf{x}} \to \mathbb{R}^{N_\mathsf{y}}$ via

$$\mathbf{y}_k = \boldsymbol{\mathcal{H}}_k\left(\mathbf{x}_k\right) + \mathbf{v}_k, \tag{2}$$

where $\mathbf{v}_k$ is the observation error at time t_k, which could be a random vector. Let us make the assumption that the observation errors are independent from each other.

The 4D-Var cost function is defined as the negative log-likelihood:

$$\begin{aligned}
\mathcal{J}^{\mathsf{sc}}\left(\mathbf{x}_0\right) &\triangleq -\ln p\left(\mathbf{x}_0|\mathbf{y}_0, \ldots, \mathbf{y}_L\right), && \text{(3a)}\\
&\propto -\ln p\left(\mathbf{x}_0\right) - \ln p\left(\mathbf{y}_0, \ldots, \mathbf{y}_L|\mathbf{x}_0\right), && \text{(3b)}\\
&\propto -\ln p\left(\mathbf{x}_0\right) - \sum_{k=0}^{L} \ln p\left(\mathbf{y}_k|\mathbf{x}_0\right), && \text{(3c)}
\end{aligned}$$

where conditional independence of the observation vectors on $\mathbf{x}_0$ was used. The background $p\left(\mathbf{x}_0\right)$ is Gaussian with mean $\mathbf{x}_0^\mathsf{b}$ and covariance matrix $\mathbf{B}$, and the observation errors $\mathbf{v}_k$ are also Gaussian distributed with mean $\mathbf{0}$ and covariance matrices $\mathbf{R}_k$, in such a way that $\mathcal{J}^{\mathsf{sc}}$ becomes:

$$\mathcal{J}^{\mathsf{sc}}\left(\mathbf{x}_0\right) = \frac{1}{2}\left\|\mathbf{x}_0 - \mathbf{x}_0^{\mathsf{b}}\right\|^2_{\mathbf{B}^{-1}} + \frac{1}{2}\sum_{k=0}^{L}\left\|\mathbf{y}_k - \boldsymbol{\mathcal{H}}_k \circ \boldsymbol{\mathcal{M}}_{k:0}\left(\mathbf{x}_0\right)\right\|^2_{\mathbf{R}_k^{-1}}, \tag{4}$$

where we have dropped the constant terms and where the notation $\|\mathbf{v}\|^2_{\mathbf{M}}$ stands for the squared Mahalanobis norm $\mathbf{v}^\top\mathbf{M}\mathbf{v}$.

This formulation is called *strong-constraint* 4D-Var because it relies on the perfect model assumption eq. (1). In practice, eq. (4) is minimised using scalable gradient-based optimisation methods to provide the analysis $\mathbf{x}_0^{\mathsf{a}}$. In cycled DA, the model is then used to propagate $\mathbf{x}_0^{\mathsf{a}}$ till the start of the next window, yielding thus a value for the background state $\mathbf{x}_0^{\mathsf{b}}$.

2.2 Weak-constraint 4D-Var

Recognising that the model is not perfect, we can replace the strong constraint eq. (1) by the more general model evolution

$$\mathbf{x}_{k+1} = \boldsymbol{\mathcal{M}}_{k+1:k}\left(\mathbf{x}_k\right) + \mathbf{w}_k, \tag{5}$$

where $\mathbf{w}_k \in \mathbb{R}^{N_{\mathsf{x}}}$ is the model error from t_k to t_{k+1}, potentially random. Let us make the assumption that the model errors are independent from each other and independent from the background errors. This implies that the model evolution satisfies the Markov property.

The updated cost function now depends on all states inside the window:

$$\begin{aligned}
\mathcal{J}^{\mathsf{wc}}\left(\mathbf{x}_0, \ldots, \mathbf{x}_L\right) &\triangleq -\ln p\left(\mathbf{x}_0, \ldots, \mathbf{x}_L | \mathbf{y}_0, \ldots, \mathbf{y}_L\right), && \text{(6a)}\\
&\propto -\ln p\left(\mathbf{x}_0, \ldots, \mathbf{x}_L\right) - \ln p\left(\mathbf{y}_0, \ldots, \mathbf{y}_L | \mathbf{x}_0, \ldots, \mathbf{x}_L\right), && \text{(6b)}\\
&\propto -\ln p\left(\mathbf{x}_0\right) - \sum_{k=0}^{L-1}\ln p\left(\mathbf{x}_{k+1}|\mathbf{x}_k\right) - \sum_{k=0}^{L}\ln p\left(\mathbf{y}_k|\mathbf{x}_k\right). && \text{(6c)}
\end{aligned}$$

With the Gaussian assumptions of section 2.1 and the additional hypothesis that the model errors $\mathbf{w}_k$ also follow a Gaussian distribution with mean $\mathbf{w}_k^{\mathsf{b}}$ and covariance matrices $\mathbf{Q}_k$, $\mathcal{J}^{\mathsf{wc}}$ becomes

$$\begin{aligned}
\mathcal{J}^{\mathsf{wc}}\left(\mathbf{x}_0, \ldots, \mathbf{x}_L\right) = \frac{1}{2}\left\|\mathbf{x}_0 - \mathbf{x}_0^{\mathsf{b}}\right\|^2_{\mathbf{B}^{-1}} &+ \frac{1}{2}\sum_{k=0}^{L-1}\left\|\mathbf{x}_{k+1} - \boldsymbol{\mathcal{M}}_{k+1:k}\left(\mathbf{x}_k\right) - \mathbf{w}_k^{\mathsf{b}}\right\|^2_{\mathbf{Q}_k^{-1}}\\
&+ \frac{1}{2}\sum_{k=0}^{L}\left\|\mathbf{y}_k - \boldsymbol{\mathcal{H}}_k\left(\mathbf{x}_k\right)\right\|^2_{\mathbf{R}_k^{-1}},
\end{aligned} \tag{7}$$

where we have once again dropped the constant terms. This formulation is called *weak-constraint* 4D-Var (Trémolet, 2006) because it relaxes the perfect model assumption eq. (1), which means that the analysis $\left(\mathbf{x}_0^{\mathsf{a}}, \ldots, \mathbf{x}_{L-1}^{\mathsf{a}}\right)$ is not any more a trajectory of the model. However, this comes at a price: the dimension of the problem has increased from N_{x} to LN_{x}.

This dimensionality increase can be mitigated by making additional assumptions. For example, one can assume that the model error is constant throughout the window, i.e.

$$\begin{aligned}
\mathbf{w}_0 = \ldots = \mathbf{w}_{L-1} &\triangleq \mathbf{w}, && \text{(8a)}\\
\mathbf{w}_0^{\mathsf{b}} = \ldots = \mathbf{w}_{L-1}^{\mathsf{b}} &\triangleq \mathbf{w}^{\mathsf{b}}, && \text{(8b)}\\
\mathbf{Q}_0 = \ldots = \mathbf{Q}_{L-1} &\triangleq L\mathbf{Q}. && \text{(8c)}
\end{aligned}$$

In this case, the trajectory $(\mathbf{x}_0, \dots, \mathbf{x}_L)$ is fully determined by $(\mathbf{w}, \mathbf{x}_0)$:

$$\mathbf{x}_k = \boldsymbol{\mathcal{M}}_{k+1:k}(\mathbf{x}_k) + \mathbf{w} = \boldsymbol{\mathcal{M}}_{k+1:k}(\boldsymbol{\mathcal{M}}_{k:k-1}(\mathbf{x}_{k-1}) + \mathbf{w}) + \mathbf{w} = \ldots \triangleq \boldsymbol{\mathcal{M}}^{\mathsf{wc}}_{k+1:0}(\mathbf{w}, \mathbf{x}_0), \tag{9}$$

with $\mathbf{x} \mapsto \boldsymbol{\mathcal{M}}^{\mathsf{wc}}_{k+1:0}(\mathbf{w}, \mathbf{x})$ being the *resolvent* of the $\mathbf{w}$-*debiased* model from t_0 to t_{k+1}. The Gaussian cost function $\mathcal{J}^{\mathsf{wc}}$ eq. (7) can hence be written

$$\mathcal{J}^{\mathsf{wc}}(\mathbf{w}, \mathbf{x}_0) = \frac{1}{2}\left\|\mathbf{x}_0 - \mathbf{x}^{\mathsf{b}}_0\right\|^2_{\mathbf{B}^{-1}} + \frac{1}{2}\left\|\mathbf{w} - \mathbf{w}^{\mathsf{b}}\right\|^2_{\mathbf{Q}^{-1}} + \frac{1}{2}\sum_{k=0}^{L}\left\|\mathbf{y}_k - \boldsymbol{\mathcal{H}}_k \circ \boldsymbol{\mathcal{M}}^{\mathrm{wc}}_{k:0}(\mathbf{w}, \mathbf{x}_0)\right\|^2_{\mathbf{R}_k^{-1}}. \tag{10}$$

This approach is called *forcing* formulation of WC 4D-Var (Trémolet, 2006; Fisher et al., 2011; Laloyaux, Bonavita, Chrust, & Gürol, 2020) and is the one that is implemented at ECMWF (Laloyaux, Bonavita, Dahoui, et al., 2020). By construction, the perfect model assumption eq. (1) is relaxed, but the analysis $(\mathbf{w}^{\mathsf{a}}, \mathbf{x}^{\mathsf{a}}_0)$ yields a trajectory of the $\mathbf{w}^{\mathsf{a}}$-debiased model. In cycled DA, this $\mathbf{w}^{\mathsf{a}}$-debiased model is used to propagate $\mathbf{x}^{\mathsf{a}}_0$ until the start of the next window to provide the background state $\mathbf{x}^{\mathsf{b}}_0$. However this time, a background is also needed for model error $\mathbf{w}^{\mathsf{b}}$. The simplest option is to use $\mathbf{w}^{\mathsf{a}}$ as is, in other words make the assumption that the dynamical model for model error is persistence.

Hereafter, the forcing formulation of WC 4D-Var is simply called WC 4D-Var.

2.3 A neural network formulation of weak-constraint 4D-Var

Following the approach of Farchi, Bocquet, et al. (2021), we now assume that the dynamical model is parametrised by a set of parameters $\mathbf{p} \in \mathbb{R}^{N_{\mathsf{p}}}$ constant over the window, in such a way that the model integration eq. (1) becomes

$$\mathbf{x}_k = \boldsymbol{\mathcal{M}}^{\mathsf{nn}}_{k:0}(\mathbf{p}, \mathbf{x}_0), \tag{11}$$

where $\mathbf{x} \mapsto \boldsymbol{\mathcal{M}}^{\mathsf{nn}}_{k:0}(\mathbf{p}, \mathbf{x})$ is the resolvent of the $\mathbf{p}$-*parametrised* model from t_0 to t_k. Using the state augmentation principle (Jazwinski, 1970), the model parameters $\mathbf{p}$ can be included in the control variables and hence be estimated in DA. If we further assume that the background for model parameters and system state are independent, and that the background for model parameters is Gaussian with mean $\mathbf{p}^{\mathsf{b}}$ and covariance matrix $\mathbf{P}$, then the Gaussian cost function eq. (4) becomes

$$\mathcal{J}^{\mathsf{nn}}(\mathbf{p}, \mathbf{x}_0) = \frac{1}{2}\left\|\mathbf{x}_0 - \mathbf{x}^{\mathsf{b}}_0\right\|^2_{\mathbf{B}^{-1}} + \frac{1}{2}\left\|\mathbf{p} - \mathbf{p}^{\mathsf{b}}\right\|^2_{\mathbf{P}^{-1}} + \frac{1}{2}\sum_{k=0}^{L}\left\|\mathbf{y}_k - \boldsymbol{\mathcal{H}}_k \circ \boldsymbol{\mathcal{M}}^{\mathsf{nn}}_{k:0}(\mathbf{p}, \mathbf{x}_0)\right\|^2_{\mathbf{R}_k^{-1}}. \tag{12}$$

This formulation is called *neural network* 4D-Var because in the present article, the set of parameters $\mathbf{p}$ are typically the weights and biases of a NN. Nevertheless, we would like to emphasise the fact that this formulation is not restricted only to NNs and can be used to estimate any parameters. The similarity between eqs. (10) and (12) is clear, which is why NN 4D-Var should be seen as another formulation of WC 4D-Var. By construction, the perfect model assumption eq. (1) is once again relaxed, but this time the analysis $(\mathbf{p}^{\mathsf{a}}, \mathbf{x}^{\mathsf{a}}_0)$ yields a trajectory of the $\mathbf{p}^{\mathsf{a}}$-parametrised model. In cycled DA, this $\mathbf{p}^{\mathsf{a}}$-parametrised model is used to propagate the analysis state $\mathbf{x}^{\mathsf{a}}_0$ until the start of the next window to provide the background state $\mathbf{x}^{\mathsf{b}}_0$. Once again, a background is also needed for model parameters $\mathbf{p}^{\mathsf{b}}$. The simplest option is to use $\mathbf{p}^{\mathsf{a}}$ as is, in other words make the assumption that the evolution model for model parameters is persistence.

Even though there are a lot of similarities between NN 4D-Var and the WC 4D-Var, two essential differences should be highlighted:

1. The model error $\mathbf{w}$ lies in the state space $\mathbb{R}^{N_{\mathsf{x}}}$ while the model parameters lies in the parameter space $\mathbb{R}^{N_{\mathsf{p}}}$, which has consequences on the design of the covariance matrices $\mathbf{Q} \in \mathbb{R}^{N_{\mathsf{x}} \times N_{\mathsf{x}}}$ and $\mathbf{P} \in \mathbb{R}^{N_{\mathsf{p}} \times N_{\mathsf{p}}}$.

2. More importantly, $\boldsymbol{\mathcal{M}}^{\mathsf{wc}}_{k:0}$ and $\boldsymbol{\mathcal{M}}^{\mathsf{nn}}_{k:0}$ may have different functional forms. In particular, in the first case the model error $\mathbf{w}$ is constant while in the second case, it is the model parameters $\mathbf{p}$ which are constant.

2.4 A simplified NN 4D-Var for model error correction

In the present article, we want to use NN 4D-Var for model error correction. Let us consider the case where the parametrised model is written

$$\mathbf{x}_{k+1} = \boldsymbol{\mathcal{M}}^{\mathsf{nn}}_{k+1:k}\left(\mathbf{p}, \mathbf{x}_k\right) = \boldsymbol{\mathcal{M}}_{k+1:k}\left(\mathbf{x}_k\right) + \boldsymbol{\mathcal{F}}\left(\mathbf{p}, \mathbf{x}_k\right), \tag{13}$$

where $\boldsymbol{\mathcal{F}}$ is a NN correction added to $\boldsymbol{\mathcal{M}}_{k+1:k}$, the resolvent of the (non-corrected) physical model from t_k to t_{k+1}, and $\mathbf{p}$ are the parameters of this NN. Following the approach of section 2.2, we assume that the NN is autonomous, i.e. the NN correction is constant throughout the window. The model evolution eq. (13) can hence be written

$$\boldsymbol{\mathcal{M}}^{\mathsf{nn}}_{k+1:k}\left(\mathbf{p}, \mathbf{x}_k\right) = \boldsymbol{\mathcal{M}}_{k+1:k}\left(\mathbf{x}_k\right) + \mathbf{w}, \quad \mathbf{w} = \boldsymbol{\mathcal{F}}\left(\mathbf{p}, \mathbf{x}_0\right). \tag{14}$$

This evolution model can then be plugged into the cost function $\mathcal{J}^{\mathsf{nn}}$ eq. (12), which yields a simplified variant of NN 4D-Var where the NN is used only once per cycle. Furthermore, comparing this to eq. (9), we conclude that

$$\boldsymbol{\mathcal{M}}^{\mathsf{nn}}_{k:0}\left(\mathbf{p}, \mathbf{x}_0\right) = \boldsymbol{\mathcal{M}}^{\mathsf{wc}}_{k:0}\left(\boldsymbol{\mathcal{F}}\left(\mathbf{p}, \mathbf{x}_0\right), \mathbf{x}_0\right). \tag{15}$$

This means that it will be possible to build this new method on top of the currently implemented WC 4D-Var framework, which is a major practical advantage.

In practice, the minimisation method implemented at ECMWF relies on an incremental approach with *outer* and *inner* loops (Courtier et al., 1994). In each outer loop, the cost function is linearised about the first-guess, and the linearised cost function is then minimised in the inner loop, typically using the conjugate gradient algorithm. Let us see how this works for our simplified NN 4D-Var. Using the change of variables $(\delta\mathbf{p}, \delta\mathbf{x}_0) \triangleq \left(\mathbf{p} - \mathbf{p}^{\mathsf{i}}, \mathbf{x}_0 - \mathbf{x}^{\mathsf{i}}_0\right)$, where $\left(\mathbf{p}^{\mathsf{i}}, \mathbf{x}^{\mathsf{i}}_0\right)$ is the first guess, we have

$$\begin{aligned}
\mathcal{J}^{\mathsf{nn}}\left(\mathbf{p}, \mathbf{x}_0\right) &= \mathcal{J}^{\mathsf{nn}}\left(\mathbf{p}^{\mathsf{i}} + \delta\mathbf{p}, \mathbf{x}^{\mathsf{i}}_0 + \delta\mathbf{x}_0\right), && \text{(16a)}\\
&= \frac{1}{2}\left\|\mathbf{x}^{\mathsf{i}}_0 - \mathbf{x}^{\mathsf{b}}_0 + \delta\mathbf{x}_0\right\|^2_{\mathbf{B}^{-1}} + \frac{1}{2}\left\|\mathbf{p}^{\mathsf{i}} - \mathbf{p}^{\mathsf{b}} + \delta\mathbf{p}\right\|^2_{\mathbf{P}^{-1}}\\
&\quad + \frac{1}{2}\sum_{k=0}^{L}\left\|\mathbf{y}_k - \boldsymbol{\mathcal{H}}_k \circ \boldsymbol{\mathcal{M}}^{\mathsf{nn}}_{k:0}\left(\mathbf{p}^{\mathsf{i}} + \delta\mathbf{p}, \mathbf{x}^{\mathsf{i}}_0 + \delta\mathbf{x}_0\right)\right\|^2_{\mathbf{R}^{-1}_k}, && \text{(16b)}\\
&\approx \frac{1}{2}\left\|\mathbf{x}^{\mathsf{i}}_0 - \mathbf{x}^{\mathsf{b}}_0 + \delta\mathbf{x}_0\right\|^2_{\mathbf{B}^{-1}} + \frac{1}{2}\left\|\mathbf{p}^{\mathsf{i}} - \mathbf{p}^{\mathsf{b}} + \delta\mathbf{p}\right\|^2_{\mathbf{P}^{-1}}\\
&\quad + \frac{1}{2}\sum_{k=0}^{L}\left\|\mathbf{d}_k - \mathbf{H}_k\mathbf{M}^{\mathsf{nn}}_{k:0}\left(\delta\mathbf{p}, \delta\mathbf{x}_0\right)^\top\right\|^2_{\mathbf{R}^{-1}_k}, && \text{(16c)}\\
&\triangleq \widehat{\mathcal{J}}^{\mathsf{nn}}\left(\delta\mathbf{p}, \delta\mathbf{x}_0\right). && \text{(16d)}
\end{aligned}$$

where $\mathbf{d}_k \triangleq \mathbf{y}_k - \boldsymbol{\mathcal{H}}_k \circ \boldsymbol{\mathcal{M}}^{\mathsf{nn}}_{k:0}\left(\mathbf{p}^{\mathsf{i}}, \mathbf{x}^{\mathsf{i}}_0\right)$, $\mathbf{H}_k$ is the tangent linear (TL) operator of $\boldsymbol{\mathcal{H}}_k$ taken at $\boldsymbol{\mathcal{M}}^{\mathsf{nn}}_{k:0}\left(\mathbf{p}^{\mathsf{i}}, \mathbf{x}^{\mathsf{i}}_0\right)$, and $\mathbf{M}^{\mathsf{nn}}_{k:0}$ is the TL operator of $\boldsymbol{\mathcal{M}}^{\mathsf{nn}}_{k:0}$ taken at $\left(\mathbf{p}^{\mathsf{i}}, \mathbf{x}^{\mathsf{i}}_0\right)$. The linearised or *incremental* cost function $\widehat{\mathcal{J}}^{\mathsf{nn}}$ is sometimes also called the *quadratic* cost function because it has the advantage of being quadratic in $\delta\mathbf{p}$ and $\delta\mathbf{x}_0$, where the conjugate gradient algorithm could be very efficient. Its gradient can be computed using algorithm 1, in which the following notation has been used: $\mathbf{F}^{\mathsf{p}}$ and $\mathbf{F}^{\mathsf{x}}$ are the TL operators of $\boldsymbol{\mathcal{F}}$ with respect to $\mathbf{p}$ and $\mathbf{x}$, respectively, both taken at $\left(\mathbf{p}^{\mathsf{i}}, \mathbf{x}^{\mathsf{i}}_0\right)$, and $\mathbf{M}_{k+1:k}$ is the TL operator of $\boldsymbol{\mathcal{M}}_{k+1:k}$ taken at $\boldsymbol{\mathcal{M}}_{k:0}\left(\mathbf{p}^{\mathsf{i}}, \mathbf{x}^{\mathsf{i}}_0\right)$. In this algorithm, lines 2 to 14 corresponds to the gradient of the incremental cost function of the WC 4D-Var cost function (without the background terms).

Algorithm 1 Gradient of the incremental cost function $\widehat{\mathcal{J}}^{\mathsf{nn}}$ eq. (16d).

Input: $\delta\mathbf{p}$ and $\delta\mathbf{x}_0$

1: $\delta\mathbf{w} \leftarrow \mathbf{F}^{\mathsf{p}}\delta\mathbf{p} + \mathbf{F}^{\mathsf{x}}\delta\mathbf{x}_0$ ▷ TL of the NN $\boldsymbol{\mathcal{F}}$
2: $\mathbf{z}_0 \leftarrow \mathbf{R}_0^{-1}\left(\mathbf{H}_0\delta\mathbf{x}_0 - \mathbf{d}_0\right)$
3: **for** $k = 1$ **to** $L-1$ **do**
4: $\quad\delta\mathbf{x}_k \leftarrow \mathbf{M}_{k:k-1}\delta\mathbf{x}_{k-1} + \delta\mathbf{w}$ ▷ TL of the dynamical model $\boldsymbol{\mathcal{M}}_{k:k-1}$
5: $\quad\mathbf{z}_k \leftarrow \mathbf{R}_k^{-1}\left(\mathbf{H}_k\delta\mathbf{x}_k - \mathbf{d}_k\right)$
6: **end for**
7: $\delta\tilde{\mathbf{x}}_{L-1} \leftarrow \mathbf{0}$ ▷ AD variable for system state
8: $\delta\tilde{\mathbf{w}}_{L-1} \leftarrow \mathbf{0}$ ▷ AD variable for model error
9: **for** $k = L-1$ **to** 1 **do**
10: $\quad\delta\tilde{\mathbf{x}}_k \leftarrow \mathbf{H}_k^{\top}\mathbf{z}_k + \delta\tilde{\mathbf{x}}_k$
11: $\quad\delta\tilde{\mathbf{w}}_{k-1} \leftarrow \delta\tilde{\mathbf{x}}_k + \delta\tilde{\mathbf{w}}_k$
12: $\quad\delta\tilde{\mathbf{x}}_{k-1} \leftarrow \mathbf{M}_{k:k-1}^{\top}\delta\tilde{\mathbf{x}}_k$ ▷ AD of the dynamical model $\boldsymbol{\mathcal{M}}_{k:k-1}$
13: **end for**
14: $\delta\tilde{\mathbf{x}}_0 \leftarrow \mathbf{H}_0^{\top}\mathbf{z}_0 + \delta\tilde{\mathbf{x}}_0$
15: $\delta\tilde{\mathbf{x}}_0 \leftarrow \left[\mathbf{F}^{\mathsf{x}}\right]^{\top}\delta\tilde{\mathbf{x}}_0$ ▷ AD of the NN $\boldsymbol{\mathcal{F}}$
16: $\delta\tilde{\mathbf{p}} \leftarrow \left[\mathbf{F}^{\mathsf{p}}\right]^{\top}\delta\tilde{\mathbf{w}}_0$ ▷ AD of the NN $\boldsymbol{\mathcal{F}}$
17: $\delta\tilde{\mathbf{x}}_0 \leftarrow \mathbf{B}^{-1}\left(\mathbf{x}_0^{\mathsf{i}} - \mathbf{x}_0^{\mathsf{b}} + \delta\mathbf{x}_0\right) + \delta\tilde{\mathbf{x}}_0$
18: $\delta\tilde{\mathbf{p}} \leftarrow \mathbf{P}^{-1}\left(\mathbf{p}^{\mathsf{i}} - \mathbf{p}^{\mathsf{b}} + \delta\mathbf{p}\right) + \delta\tilde{\mathbf{p}}$

Output: $\nabla_{\delta\mathbf{p}}\widehat{\mathcal{J}}^{\mathsf{nn}} = \delta\tilde{\mathbf{p}}$ and $\nabla_{\delta\mathbf{x}_0}\widehat{\mathcal{J}}^{\mathsf{nn}} = \delta\tilde{\mathbf{x}}_0$

In the end, in order to implement the simplified NN 4D-Var we can reuse most of the framework already in place for WC 4D-Var and we need to provide:

- the forward operator $\boldsymbol{\mathcal{F}}$ of the NN to compute the nonlinear trajectory at the start of each outer iteration;
- the TL operators $\mathbf{F}^{\mathsf{x}}$ and $\mathbf{F}^{\mathsf{p}}$ of the NN for line 1 of algorithm 1;
- the adjoint (AD) operators $\left[\mathbf{F}^{\mathsf{x}}\right]^{\top}$ and $\left[\mathbf{F}^{\mathsf{p}}\right]^{\top}$ of the NN for lines 15 and 16 of algorithm 1.

From a technical perspective, all these operators have to be computed in the model core, where the components of the system state are available. In OOPS, the model core is implemented in Fortran, which implies that we need a ML library in Fortran. The only one that we could find, namely the Fortran–Keras Bridge (FKB, Ott et al., 2020), does not provide all the required operators. For this reason, we have implemented our own NN library in Fortran, called Fortran neural networks (FNN, Farchi et al., 2022). In this library, we have manually implemented, for each layer that we need, functions for the forward, but also the TL and adjoint operators with respect to both NN parameters and NN input. We have then included the FNN library in OOPS and added the interface between OOPS and FNN for two forecast models, OOPS-QG and OOPS-IFS. Finally, we have included the NN parameters in the control variables in OOPS, in such a way that they can be estimated using the simplified NN 4D-Var method.

3 The quasi-geostrophic model

The simplified NN 4D-Var formulation provides a convenient alternative to the original NN 4D-Var. It has the advantage of being much easier to implement because it is built on top of WC 4D-Var, which is already implemented in OOPS. We first test and validate the method using OOPS-QG. In particular, we want to confirm that the simplified NN 4D-Var method is able to make an accurate online estimation of model error.

Table 1. Set of parameters for the reference setup (middle row) and the perturbed setup (right row).

Parameter	Reference setup	Perturbed setup
Top layer depth	6000 m	5750 m
Bottom layer depth	4000 m	4250 m
Integration time step	10 min	20 min

3.1 Brief model description

The quasi-geostrophic (QG) model in the present article is the same as the one used by Fisher and Gürol (2017), Laloyaux, Bonavita, Chrust, and Gürol (2020) and later by Farchi, Laloyaux, et al. (2021). In the following, we only outline the model description. More details about this model can be found in Fisher and Gürol (2017); Laloyaux, Bonavita, Chrust, and Gürol (2020).

The QG model's equations express the conservation of the (non-dimensional) potential vorticity q for two layers of constant potential temperature in the $x - y$ plane. The potential vorticity is related to the stream function ψ through a specific variant of Poisson's equation. The domain is periodic in the x direction, and with fixed boundary conditions for q in the y direction. We use a horizontal discretisation of 40 grid points in the x direction and 20 in the y direction. In OOPS, the control vector $\mathbf{x}$ contains all values of the stream function ψ for both levels, i.e. a total of $N_{\mathsf{x}} = 1600$ variables.

3.2 The reference and perturbed setups

In the test series reported in sections 4 and 5, we rely on twin experiments. The synthetic truth is generated using the reference setup described by Farchi, Laloyaux, et al. (2021). Model error is then introduced by using a perturbed setup, in which the values of both layer depths and the integration time steps have been modified, as reported in table 1. Note that, by contrast with the perturbed setup of Farchi, Laloyaux, et al. (2021), the orography term has not been changed, because we have found that the model error setup is sufficiently challenging as is and an orography perturbation does not add meaningful complexity here.

3.3 Neural network architecture for model error correction

By construction, NN 4D-Var (both the original and simplified formulations) is very similar to parameter estimation, which is very challenging when the number of parameters is high. For this reason, it is important to use smart NN architectures to be *parameter efficient*, i.e. reduce as much as possible the number of parameters. This typically involves applying prior knowledge about the system under study to the choice of the NN architecture. A typical smart architecture is the monomial architecture introduced by Bocquet et al. (2019), in which the model tendencies are parametrised by a set of regressors (the monomials) and then integrated in time to build the resolvent between two time steps. In the present article, we follow another approach, introduced by Bonavita and Laloyaux (2020) for the IFS. In this case, the NN is applied independently for each atmospheric column and for several groups of variables: mass (temperature and surface pressure), wind (vorticity and divergence), and humidity. Horizontal and temporal variations are taken into account by adding latitude, longitude, time of the day, and month of the year to the set of predictors. This choice is imposed by operational constraints – variables in different columns may come from different processes when using parallelism. It also makes sense because a significant amount of the model error in the IFS comes from

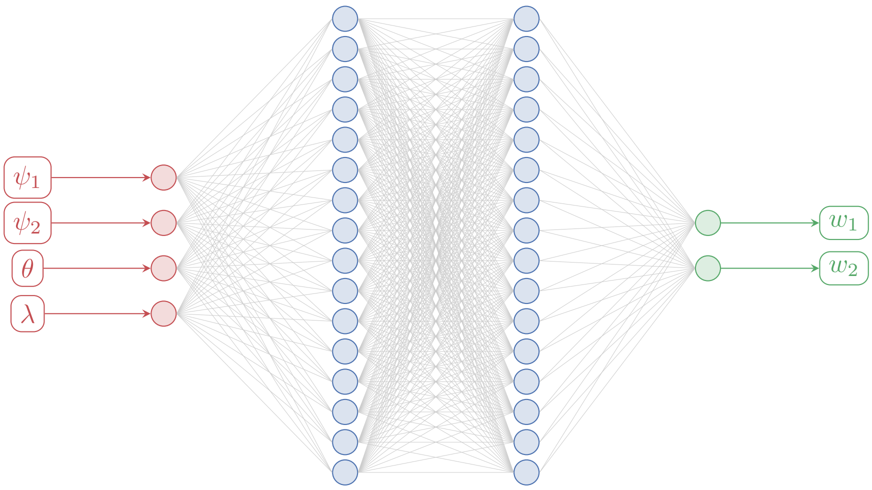

Figure 1. Illustration of the NN architecture. On the left in red, the input layer. In the centre in blue, the two hidden layers, with tanh activation. On the right in green, the output layer.

the parametrisation of physical processes, which is applied in vertical model columns (Polichtchouk et al., 2022), and because in this configuration, the amount of samples is multiplied by the number of vertical columns in the data, which is highly beneficial to the training. Furthermore, it has been shown that the performance of simple vertical NNs is roughly similar to that of non-vertical convolutional neural networks in a realistic model error correction problem (Laloyaux et al., 2022).

The QG model has only two vertical layers and one variable, the stream function ψ, and it is autonomous, i.e. the model does not explicitly depend on time. This means that our NN for model error correction, independently applied to all 40×20 columns, has four predictors:

1. ψ_1 the bottom layer stream function;
2. ψ_2 the top layer stream function;
3. $\sin\left[2\pi\left(\theta - 1/2\right)/40\right]$, where θ is the longitude index between 1 and 40;
4. $\sin\left[\pi\left(\lambda - 1/2 - 10\right)/20\right]$ where λ is the latitude index between 1 and 20;

and two predictands:

1. w_1 the model error estimate for the bottom layer stream function;
2. w_2 the model error estimate for the top layer stream function.

Note that the sinus function is used here to make the NN aware of the periodicity. We have tested several NNs, and ended up with the following sequential architecture, illustrated in fig. 1: (i) a first internal dense layers with 16 neurons and with the tanh activation function; (ii) a second internal dense layers with 16 units and with the tanh activation function as well; (iii) one output dense layer with 2 units and no activation function. This NN has a total of $(2 \times 4 + 4) + (4 \times 4 + 4) + (4 \times 2 + 2) = 386$ parameters, which is significantly less than the number of variables (1600).

To stay within the scope of the simplified NN 4D-Var defined in section 2.4, we assume that the NN correction is constant throughout the window, and that it is added after every model time step (i.e. every 20 min in our case) as it is enforced in the current implementation of WC 4D-Var. According to the classification of Farchi, Laloyaux, et al. (2021) and Farchi, Bocquet, et al. (2021), this approach is a *resolvent* correction, because it is added after the integration scheme. However, a classical resolvent correction would add the correction after every window, in other words much less frequently than after every model time step. Hence, the spirit of the present correction is closer to that of a *tendency* correction.

4 Offline learning results

We begin the numerical experiments by using offline learning to train the NN. Offline learning here serves two purposes: it provides a baseline for comparison as well as a pre-trained NN for online learning.

4.1 Observation and data assimilation setup

In the present test series, we use for the QG model the same initial condition as Farchi, Laloyaux, et al. (2021). After a first relaxation run of 256 d, the state is perturbed and a second relaxation run of 256 d is performed to provide the initial state for the DA experiment. At this point, observations are available every 2 h, starting at 01:00 every day, at 30 fixed locations, whose distribution mimics the coverage provided by (polar-orbiting) satellite soundings. The observation operator is simply a bilinear interpolation of the stream function at the observation locations. The observations are independently perturbed using a Gaussian noise with zero mean and standard deviation equal to 0.2 (about 4 % of the model variability).

We start by assimilating the observations using cycled strong-constraint 4D-Var, with consecutive windows of 1 d starting at 00:00 each. Hence, there are 12 batches of observations, for a total of 360 observations per window. The observation error covariance matrix is set to $\mathbf{R} = 0.2^2\mathbf{I}$ to be consistent with how the synthetic observations are produced. For the first cycle, the background state $\mathbf{x}_0^{\mathsf{b}}$ is set to be the initial condition before the two relaxation runs. For the following cycles, the background state is obtained by forecasting the previous analysis state. Finally, the background error covariance matrix is set to $\mathbf{B} = b^2\mathbf{C}$, where $\mathbf{C}$ is a short-range correlation matrix, the same as the one used by Farchi, Laloyaux, et al. (2021), and where b is the standard deviation, a free parameter. The accuracy of the estimations is measured with the instantaneous root-mean-squared error (RMSE) between the estimate and the truth for all 1600 state variables, possibly averaged over time. In particular, the first-guess (respectively analysis) RMSE is defined in this article as the instantaneous RMSE between the first-guess (or analysis) trajectory, the trajectory originated from the first-guess (or analysis) at the start of the window, and the true trajectory, averaged over the entire DA window. The time-averaged first-guess (respectively analysis) RMSE is then defined as this first-guess (or analysis) RMSE averaged over a sufficiently large number of cycles.

In order to be close to operational conditions, we tune the value of b to minimise the time-averaged first-guess RMSE. Preliminary experiments (not detailed here) have shown that, for the present DA setup, the optimal value is $b = 0.4$. With this value, we run a cycled DA experiment of $N_{\mathsf{t}}^{\text{total}} = 2100$ cycles. The results of the first $N_{\mathsf{t}}^{\text{spinup}} = 51$ cycles are dropped as spin-up process of the experiment. Then, for each remaining cycles $t = 1, \ldots, N_{\mathsf{t}}^{\text{data}} = 2049$, we keep $\mathbf{x}_0^{\mathsf{b}}(t)$ and $\mathbf{x}_0^{\mathsf{a}}(t)$, respectively the first-guess and the analysis at the start of the t-th window.

4.2 Neural network training

As shown by Farchi, Laloyaux, et al. (2021), the analysis increment $\mathbf{x}_0^{\mathsf{a}}(t) - \mathbf{x}_0^{\mathsf{b}}(t)$ can be chosen as a proxy of the model error for a 1-window-long integration, provided that the analysis is a reasonably accurate estimation of the true state:

$$\mathbf{x}_0^{\mathsf{a}}(t+1) - \mathbf{x}_0^{\mathsf{b}}(t+1) = \mathbf{x}_0^{\mathsf{a}}(t+1) - \boldsymbol{\mathcal{M}}_t\left(\mathbf{x}_0^{\mathsf{a}}(t)\right) \approx \mathbf{x}_0^{\mathsf{t}}(t+1) - \boldsymbol{\mathcal{M}}_t\left(\mathbf{x}_0^{\mathsf{t}}(t)\right), \quad (17)$$

where $\boldsymbol{\mathcal{M}}_t$ corresponds to the resolvent of the model between the start of the t-th window and the start of the $(t+1)$-th window, and where $\mathbf{x}_0^{\mathsf{t}}(t)$ is the true state of the system at the start of the t-th window. However, as explained in section 3.3, the NN correction is added after every model time step, which means that we need a proxy of the model error for a 1-step integration. Without further knowledge on the model error dynamics, we assume a uniform linear growth of model error in time and hence we rescale the analysis increments by a factor $\delta t/\Delta T = 1/72$, where $\delta t = 20\,\mathrm{min}$ is the model time step and $\Delta T = 1\,\mathrm{d}$ is the window length. Note that, even if the analysis was available at a 1 model step frequency, we would not use it because the accuracy of the analysis would most probably be insufficient to detect a model error signal in the analysis increments.

To summarise, we use the following dataset for the training of the NN:

$$\left\{\mathbf{x}_0^{\mathsf{a}}(t) \mapsto \frac{\delta t}{\Delta t}\left[\mathbf{x}_0^{\mathsf{a}}(t+1) - \mathbf{x}_0^{\mathsf{b}}(t+1)\right], \quad t = 1, \ldots, N_{\mathsf{t}}^{\mathrm{data}} - 1 = 2048\right\}. \quad (18)$$

Note the time lag between the input $\mathbf{x}_0^{\mathsf{a}}(t)$ and the output $\delta t/\Delta T\left(\mathbf{x}_0^{\mathsf{a}}(t+1) - \mathbf{x}_0^{\mathsf{b}}(t+1)\right)$. Indeed, the analysis increment $\mathbf{x}_0^{\mathsf{a}}(t+1) - \mathbf{x}_0^{\mathsf{b}}(t+1)$ of the $(t+1)$-th window does inform about the model error during the t-th window, which is exactly what we need according to the model formulation described in section 2.4. Also note that we have chosen to use the analysis $\mathbf{x}_0^{\mathsf{a}}(t)$ as predictor, but we could have equivalently chosen the first-guess $\mathbf{x}_0^{\mathsf{b}}(t)$. Preliminary experiments (not illustrated here) have shown that both choices yield similar results. Since the NN is applied independently to each atmospheric column, there are actually $40 \times 20 = 800$ samples per pair (analysis $\mapsto$ analysis increment). Finally, in order to accelerate the convergence, the input and output of the training dataset are standardised before the training, using independent normalisation coefficients for each variable.

In order to evaluate the sensitivity to the length of the dataset, we train the NN using only the last $N_{\mathsf{t}}^{\mathrm{train}}$ pairs (analysis $\mapsto$ analysis increment) for several values of N_{t}. Among all these $N_{\mathsf{t}}^{\mathrm{train}}$ pairs, the first $7/8^{\mathrm{th}}$ form the training dataset and the last $1/8^{\mathrm{th}}$ the validation dataset. The test dataset is formed by $N_{\mathsf{t}}^{\mathrm{test}} = 2048$ pairs (truth $\mapsto$ true model error) originated from a different trajectory of the model. With this setup, the NN is trained for a maximum of 1024 epochs using Adam algorithm (Kingma & Ba, 2015), a variant of the stochastic gradient descent, with the typical learning rate 1×10^{-3}. The loss function is the mean-squared error (MSE). To accelerate the training, we use a relatively large batch size (1024) as well as an early stopping callback on the validation MSE with a patience of 256 epochs. After the training, we compute the test MSE. This experiment is repeated 16 times with different sets of trajectories for training and testing and different random seeds for Adam. For comparison, we have also performed the exact same set of experiments but with dense and perfect observations, i.e. when the analysis is equal to the true state. This second set of experiments illustrates the full predictive power of the NN representation of the model error.

Figure 2 shows the evolution of the test MSE as a function of the length of the training dataset $N_{\mathsf{t}}^{\mathrm{train}}$. The score is normalised by the averaged squared norm of the model error, in such a way that it is equal to 1 when the NN predicts a zero model error. In all experiments, the normalised test MSE is lower than 1. This means that, on average, the model error prediction is useful. When using the truth, both training and test datasets are statistically equivalent. The normalised test MSE decreases with the size of the training dataset $N_{\mathsf{t}}^{\mathrm{train}}$. The final value is 0.334 for $N_{\mathsf{t}}^{\mathrm{train}} = 2048$, but the score is already quite good (0.351) for $N_{\mathsf{t}}^{\mathrm{train}} = 128$. The residual error for a large training dataset comes from the limited predictive power of the NN. We have checked that better scores

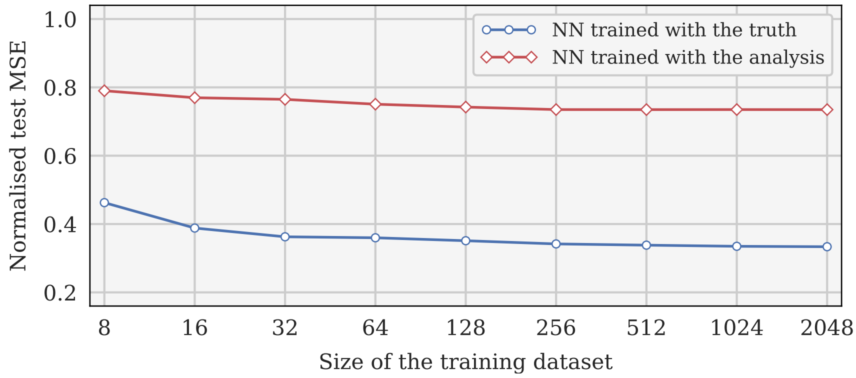

Figure 2. Offline NN training. Evolution of the normalised test MSE as a function of the length of the training dataset $N_{\mathsf{t}}^{\text{train}}$ for the NN trained with the truth (in blue) and the NN trained with the analysis (in red).

can easily be obtained when using larger, non column-wise NNs. Unsurprisingly, when using the analysis the normalised test MSE is significantly higher (0.735 at best) and stops improving for $N_{\mathsf{t}}^{\text{train}} \geq 256$. The primary reason for these discrepancies is the fact that the statistical moments (e.g. the time average and time standard deviation) are not the same between the analysis increments and the true model error. In particular, the average analysis increment norm is lower than the average model error norm. This means that the NN trained with the analysis generally underestimates the model error. This is consistent with what has been found by Crawford et al. (2020) and Farchi, Laloyaux, et al. (2021).

4.3 Corrected data assimilation

Now that the NN has been trained, we would like to test the hybrid model in forecast and DA experiments. We start with DA using the exact same setup as in section 4.1, but with a true state taken from a different trajectory of the model. Four 4D-Var variants are compared:

1. `SC`: strong-constraint with the physical model (no model error correction).
2. `WC`: weak-constraint with the physical model – in this case the model error correction comes from the constant, online estimated forcing.
3. `SC-NNt`: strong-constraint with the hybrid model, where the NN correction has been trained with the truth using the largest dataset ($N_{\mathsf{t}}^{\text{train}} = 2048$).
4. `SC-NNa`: strong-constraint with the hybrid model, where the NN correction has been trained with the analysis using the largest dataset ($N_{\mathsf{t}}^{\text{train}} = 2048$).

In all cases, we use the same background error covariance matrix $\mathbf{B}$ as in section 4.1, because we want to highlight the benefit of each approach without the need to re-tune $\mathbf{B}$. The initial background state $\mathbf{x}_0^{\mathsf{b}}$ corresponds to the background obtained after a spin-up of 32 DA cycles with strong-constraint 4D-Var. For weak-constraint 4D-Var, we need to provide in addition (i) the initial background for model error $\mathbf{w}^{\mathsf{b}}(0)$, and (ii) the background error covariance matrix for model error $\mathbf{Q}$. We choose to use $\mathbf{w}^{\mathsf{b}}(0) = \mathbf{0}$ and $\mathbf{Q} = q^2\widehat{\mathbf{C}}$, where $\widehat{\mathbf{C}}$ is a long-range correlation matrix, the same as the one used by Laloyaux, Bonavita, Chrust, and Gürol (2020), and where q is the standard deviation, another free parameter. We choose $q = 0.004$ in order to minimise the time-averaged

Table 2. Offline DA results. Time-averaged first-guess and analysis RMSE for the four 4D-Var variants presented in section 4.3. For each variant, we report the mean (main numbers) and standard deviation (in parentheses) values over the 128 experiments.

Variant	4D-Var constraint	Model error correction	First-guess RMSE	Analysis RMSE
`SC`	strong	—	0.350 (0.020)	0.157 (0.003)
`WC`	weak	constant, online estimated	0.271 (0.016)	0.128 (0.003)
`SC-NNt`	strong	NN trained offline with the truth	0.263 (0.018)	0.133 (0.003)
`SC-NNa`	strong	NN trained offline with the analysis	0.265 (0.020)	0.144 (0.003)

first-guess RMSE. In each case, we run a cycled DA experiment of $N_{\mathrm{t}}^{\mathrm{assim}} = 257$ cycles, which we empirically consider to be sufficiently long. The results of the first 33 cycles are dropped as spin-up. For the remaining 224 cycles, we compute the first-guess and analysis RMSE. Each experiment is repeated 128 times with different trajectories for the synthetic truth. Note that in the second and third case, the 128 repetitions are equally spread over the 16 trained NN obtained in section 4.2: experiments 1 to 8 use the first trained NN, experiments 9 to 16 use the second, experiments 17 to 24 use the third, etc.

The time-averaged first-guess and analysis RMSE are reported in table 2. The results show the efficiency of model error corrections: in all cases, the first-guess and the analysis are more accurate with model error correction (`WC`/`SC-NNt`/`SC-NNa`) than without (`SC`). As expected, the model error correction provided by the NN is more efficient when the NN has been trained with the truth (`SC-NNt`) than when it has been trained with the analysis (`SC-NNa`). Furthermore, using the offline correction provided by the NN (`SC-NNt`/`SC-NNa`) yields in both cases a more accurate first-guess but a less accurate analysis than using the online correction computed with weak-constraint 4D-Var (`WC`).

4.4 Corrected forecast

To conclude this first test series, we evaluate the accuracy of the model in the four cases described in section 4.3. To this end, we extend the previous set of experiments. After each analysis cycle, we compute a 32-day forecast starting from the DA analysis using the same model as in the 4D-Var cost function. In the case of weak-constraint 4D-Var (`WC`), the constant, online estimated forcing is used throughout the entire forecast. In the case of strong-constraint 4D-Var with the hybrid model (`SC-NNt`/`SC-NNa`), the NN correction is also used throughout the entire forecast, but in a flow-dependent way: the correction values are updated at a 1-day frequency using the forecasted state. With these specifications, the error in the first day of forecast corresponds to the analysis error and the error in the second day of forecast corresponds to the first-guess error. Figure 3 shows the evolution of the forecast RMSE, averaged over the last 32 DA cycles and over the 128 repetitions of the experiments, as a function of the forecast lead time.

With weak-constraint 4D-Var (`WC`), the model error correction is calibrated over the DA window, i.e. over the first day. Overall, the correction is efficient and yields a more accurate forecast than with the non-corrected model (`SC`). After several days, the true model error has significantly evolved and this initial error estimate gets less accurate. This is why the reduction of the forecast error vanishes after several days. Also note that the model has a periodic behaviour, with a period around 16 days. This means that, after 16 days, the model state (and hence the model error) is roughly the same as at the beginning, which explains the forecast error reduction around day 16 and around day 32.

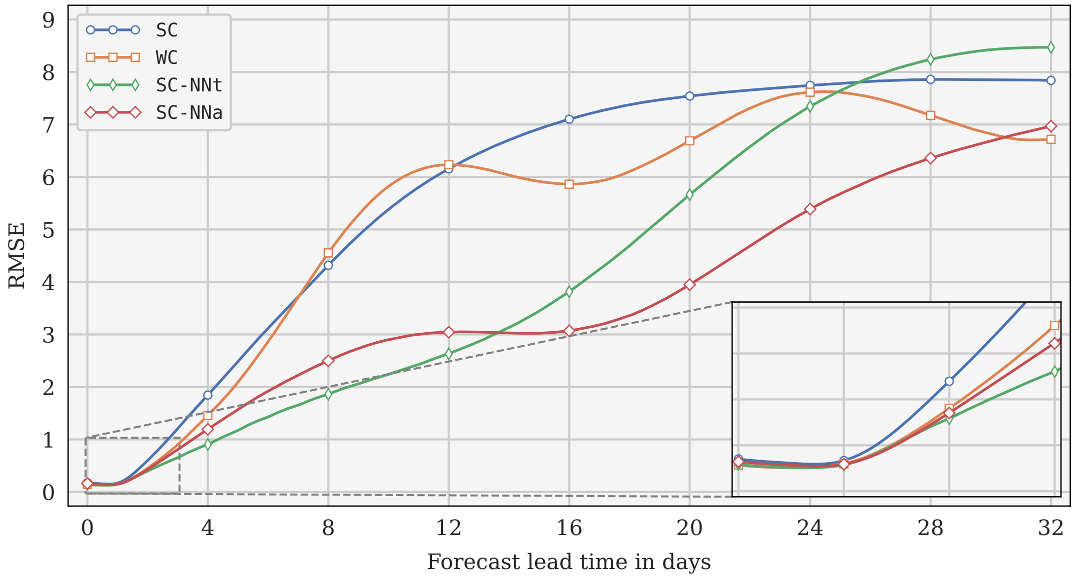

Figure 3. Offline forecast results. Evolution of the forecast RMSE, averaged over the last 32 cycles and over the 128 experiments, as a function of the forecast lead time for the four 4D-Var variants: `SC` in blue, `WC` in orange, `SC-NNt` in green, and `SC-NNa` in red. The insert zooms in the short forecast lead times.

By contrast, when using the hybrid model (`SC-NNt`/`SC-NNa`), the model error correction is flow-dependent (updated every day). This yields overall an even more accurate forecast than with weak-constraint 4D-Var (`WC`). In the first few days, the correction accumulates and positively interacts with the physical model, which is why the forecast error reduction increases over time. After several days however, the model error correction becomes less efficient, because the forecasted state – the most important predictor of the NN – has become significantly different from the true state. At this point, the model error correction does not any more yield a forecast error reduction. Worse, it even increases the forecast errors. This explains the quick increase of the forecast errors after 10 days when the NN is trained with the truth (`SC-NNt`) and after 15 days when the NN is trained with the analysis (`SC-NNa`). In an operational perspective, it would be interesting to progressively mitigate the model error correction over time, but this is beyond the scope of the present study. Surprisingly, the validity period of the model error correction is longer for `SC-NNa` (NN trained with the analysis) than for `SC-NNt` (NN trained with the truth). This could be due to the fact that a NN trained with the analysis underestimates the model error: if the model error estimate is pointing in the wrong direction, it is better to have an underestimated model error (Crawford et al., 2020). Finally, after about 13 days, the forecast is more accurate with `SC-NNa`. We believe that this result is related to the limited predictive power of the chosen NN. Indeed, we have checked that with larger NNs, the accuracy of the forecast is always more accurate with `SC-NNt` than with `SC-NNa`.

5 Online learning results

In the present section, we test the simplified online NN 4D-Var presented in section 2.4 using the same QG model as in the offline experiments.

5.1 Data assimilation setup

In this last test series, we use the same DA setup as in sections 4.1 and 4.3. Once again, the true state stems from a different trajectory. We keep the same initial background state $\mathbf{x}_0^{\mathsf{b}}$ and background error covariance matrix $\mathbf{B}$ as in section 4.3, once again to highlight the benefit of each approach without the need to re-tune $\mathbf{B}$. In addition, we need to provide (i) the initial background for model parameters $\mathbf{p}_0^{\mathsf{b}}$ and (ii) the background error covariance matrix for model parameters $\mathbf{P}$. For $\mathbf{p}_0^{\mathsf{b}}$, we choose to use the parameters of the NN that has been trained offline with the analysis, in other words we use offline learning as a pre-training step for online learning. Hence we hope to immediately see the potential benefits of online learning. Finally, without any prior knowledge on the model parameters, we use $\mathbf{P} = p^2\mathbf{I}$, where p is the standard deviation, a free parameter. After several preliminary tests, we have chosen $p = 0.02$. Following the approach of section 4.4, at each DA cycle, we compute a 32-day forecast starting from the DA analysis using the hybrid model with the updated parameters. Finally, once again, each experiment is repeated 128 times with as many different trajectories for the synthetic truth. In the following paragraphs, we use the label `NN` to refer to this fifth 4D-Var variant.

5.2 Temporal evolution of the forecast errors

Figure 4 shows the temporal evolution of the errors in the first day of forecast (the analysis), in the second day of forecast (the first-guess), and in the eighth day of forecast (which corresponds to a medium-range forecast). The evolution in all three cases is very similar. At the start of the experiment, the forecast errors with `NN` (NN trained online) are close to those with `SC-NNa` (NN trained offline with the analysis). This was expected because in the `NN` variant, we have initialised the parameters of the NN using the parameters obtained by offline training with the analysis. The added positive effect of the online NN training is then rapidly visible. After a few cycles, the forecast errors have decreased. This improvement is quicker for shorter forecast horizons. For the medium-range errors, we even see an increase at the start of the experiments before they eventually decrease, after several dozens of cycles. At the end of the experiments, the forecast is significantly more accurate with `NN` than with `SC-NNa`, which is what we hoped for. In some cases (first-guess and medium range), the forecast is even better with `NN` than `SC-NNt` (NN trained offline with the truth). This results may seem at first somewhat surprising because, unless there has been some optimisation issues, the NN trained offline with the truth should provide the most accurate model error predictions. However, one must keep in mind that two essential simplifications have been made:

1. the model error growth is linear in time (section 4.2);
2. the model error correction is constant over the DA window (section 2.4).

This explains why the NN trained offline with the truth is suboptimal in the DA and forecast experiments considered here. The first assumption could be circumvented by using samples of the true model error for a $\delta t = 20\,\text{min}$ forecast (obviously, this would not be possible when training with the truth) but the second assumption is intrinsic to the simplified NN 4D-Var formulation. This second assumption allows us to build NN 4D-Var as a relatively simple extension of the currently implemented weak-constraint 4D-Var, but it has a negative impact on the forecast that we will illustrate in the following section.

5.3 Focus on the first day of forecast

Figure 5 shows the temporal evolution of the errors in the second day of forecast in two cases: (i) the NN correction is updated every day (as has been done previously – this corresponds to the first-guess errors) or (ii) it is kept constant throughout the entire forecast. The forecast errors with the NN (`SC-NNt`/`SC-NNa`/`NN`) are systematically

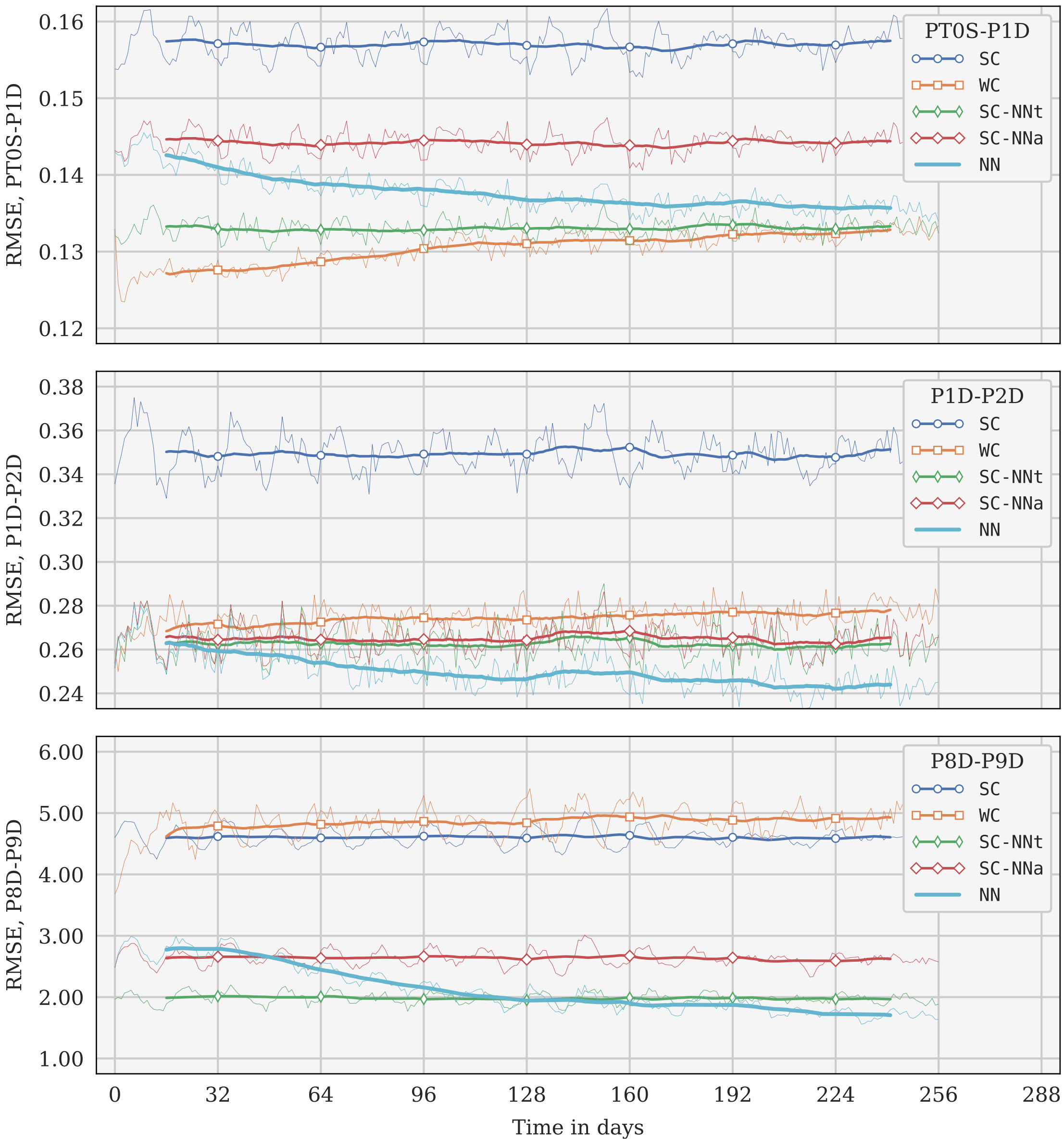

Figure 4. Forecast scores for the online experiments. Evolution of the forecast RMSE, averaged over the 128 experiments and over PT0S-P1D (top panel), over P1D-P2D (middle panel), or over P8D-P10D (bottom panel), as a function of time for the five 4D-Var variants: `SC` in blue, `WC` in orange, `SC-NNt` in green, `SC-NNa` in red, and `NN` in teal. The thin lines report the instantaneous values and the thick lines report the running-average over 32 cycles.

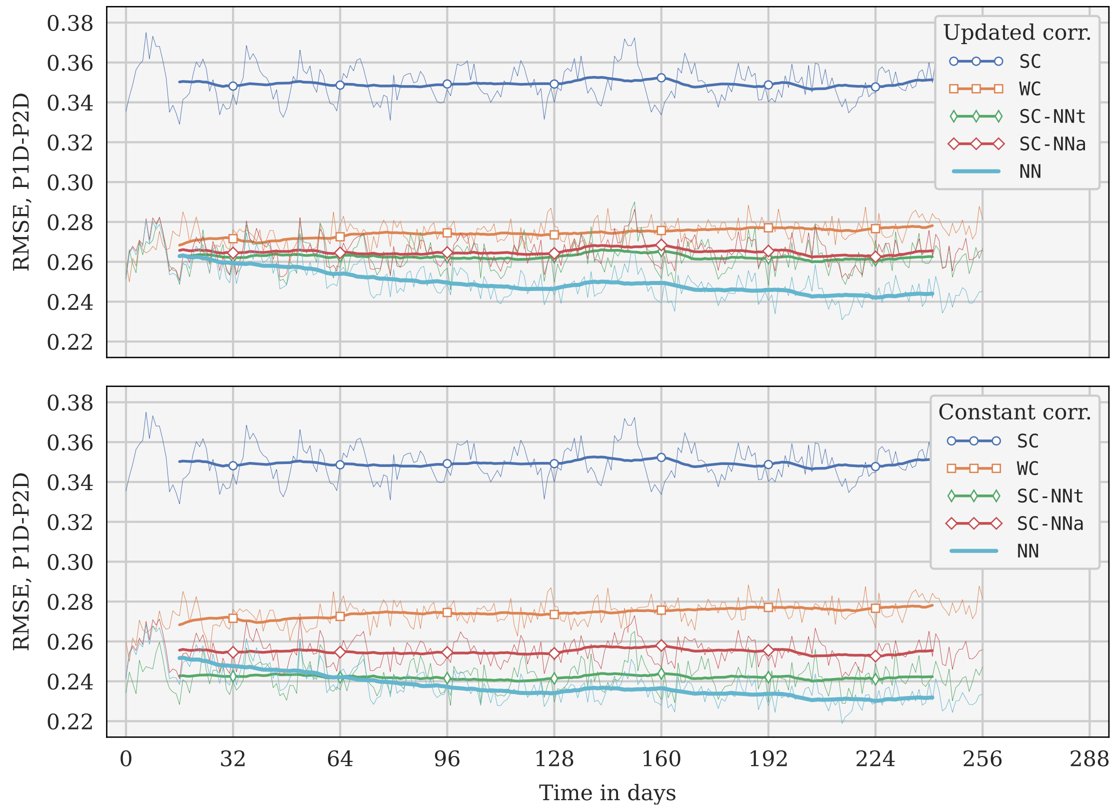

Figure 5. Forecast scores for the online experiments. Evolution of the forecast RMSE, averaged over the 128 experiments and over P1D-P2D, as a function of time for the five 4D-Var variants: `SC` in blue, `WC` in orange, `SC-NNt` in green, `SC-NNa` in red, and `NN` in teal. The NN correction is either updated every day (top panel, same as the middle panel of fig. 4) or kept constant throughout the entire forecast (bottom panel). The thin lines report the instantaneous values and the thick lines report the running-average over 32 cycles.

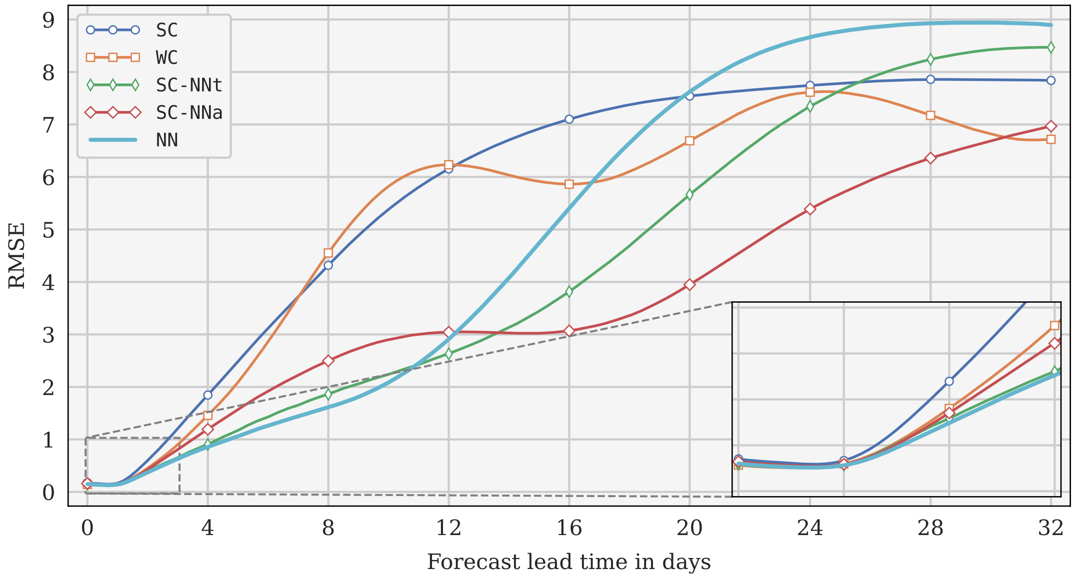

Figure 6. Online forecast results. Evolution of the time-averaged forecast RMSE, averaged over the last 32 cycles and over the 128 experiments, as a function of the forecast horizon for the five 4D-Var variants: `SC` in blue, `WC` in orange, `SC-NNt` in green, `SC-NNa` in red, and `NN` in teal.The insert zooms in the short forecast lead times.

lower in the second case than in the first. Indeed, in the 4D-Var variants considered here, the NN correction is constant over the DA window, hence the forecast model is more consistent with the 4D-Var analysis when the NN correction is not updated. Of course, there is a limit to this logic because the model error evolves over time – see the discussion on the accuracy of the forecast with `WC` in section 4.4 – which is why it is important to update the NN correction for the forecast accuracy. Therefore, we believe that implementing NN 4D-Var without the assumption of a constant model error over the window should have a positive impact on the analysis, but also in the forecast. Rge implementation of such a formulation would not be trivial, as it could not be built directly on top of the existing WC 4D-Var. Although we have not attempted it in this study, we envisage considering it in further studies.

5.4 Forecast errors at the end of the experiments

Finally, fig. 6 shows the evolution of the forecast RMSE, averaged over the last 32 cycles and over the 128 repetitions of the experiments, as a function of the forecast lead time. The errors are the same as the ones in sections 5.2 and 5.3, but aggregated and shown in a different way. For the `NN` variant, the forecast errors up to day 10 are consistent with the description in section 5.2. After day 10, the forecast errors increase accelerate, which indicates that the NN correction is not any more valid. This is the same phenomenon as what has been described in section 4.4 for `SC-NNt` and `SC-NNa`, but this time, the error increase is earlier and quicker. Once again, we believe that this result is related to the limited predictive power of the chosen NN. However, using a larger and deeper NN (i.e. with more parameters) is not necessarily a good strategy with online learning. Indeed, based on preliminary experiments, we conclude that if the number of parameters is large, the background error covariance matrix for parameters (called $\mathbf{P}$ in section 2.3) must be small to avoid a quick divergence of the method. The downside of this choice is that it naturally slows down the learning process. This is why, with online learning, it is important to keep the number of parameters as small as possible, as explained by (Farchi, Bocquet, et al., 2021). Hence, the use of online learning could initially be limited to the correction of short-term forecasts.

6 Conclusions

In this article, we have developed a new variant of weak-constraint 4D-Var, in which a set of parameters can be jointly estimated alongside the system state. The new method is called NN 4D-Var to emphasise the fact that it is used in this article to estimate the coefficients (weights and biases) of a NN. It can be seen as a simplified variant of the original NN 4D-Var method introduced by Farchi, Bocquet, et al. (2021), dedicated to model error correction. It is assumed that the NN provides a correction to a physical model, added after each integration, and constant over the DA window. These simplifications make the method very similar to the forcing formulation of weak-constraint 4D-Var, and hence easier to implement on top of an existing implementation of weak-constraint 4D-Var, such as the one available in the OOPS framework.

In the second part of the article, we have provided a numerical illustration of the new, simplified NN 4D-Var algorithm in conditions which are as close as possible to operational. The illustrations use twin experiments with OOPS-QG, a two-layer two-dimensional QG model. A simple yet non-trivial model error setup is introduced, where the layer depths and integration step of the model are perturbed. The model error correction is computed using a small, dense NN acting on vertical columns, like the one used for an operational model by Bonavita and Laloyaux (2020). The NN is first trained offline, using the analyses and analysis increments of a DA experiment with the non-corrected model, following the method originally introduced by Brajard et al. (2020). The corrected model is then used in forecast and DA experiments, and provides in both cases significant improvements in the scores as already shown by Farchi, Laloyaux, et al. (2021). Then, the NN is trained online using the new, simplified NN 4D-Var algorithm. The results confirm the findings of Farchi, Bocquet, et al. (2021) for the original NN 4D-Var algorithm. With proper tuning of the background error covariance matrices, an online, joint estimation of the system state and the NN parameters is possible. As new observations become available, the model error correction becomes more accurate, which translates into lower analysis, first-guess, and short- to mid-term forecast errors than in the offline training case.

The results also illustrate two limitations of the simplified NN 4D-Var method. The first is related to the assumption of a constant model error throughout the window. This is necessary to build the new method on top of an existing weak-constraint 4D-Var implementation, but we believe that relaxing this simplification could improve the analysis and short-term forecast errors. This could be the topic of further studies on the the method. The other limitation is somewhat more fundamental: the online training process is slower as the number of parameters to estimate is larger, as already highlighted by Farchi, Bocquet, et al. (2021). This underlines the importance of choosing smart, parameter-efficient NNs.

At this point, we estimate that the simplified NN 4D-Var method is mature enough for more realistic applications, for example with the IFS. Implementing the new formulation in this operational model will only require developing an interface to the NN library with all the algorithmic developments already in place in the OOPS framework. For such application, we would typically use the vertical NN architecture of Bonavita and Laloyaux (2020), for which the number of parameters is much lower than the number of system state variables. In this case however, the main difficulty would come from the fact that the true state of the system is unknown, which makes the evaluation much harder because the diagnostics should be based on observations. Nevertheless, we should be able to rely on the test suite developed by ECMWF to evaluate the potential benefits of proposed upgrades to the operational assimilation and forecast systems.

Finally, the current implementation of the simplified NN 4D-Var method in OOPS is dedicated to model error correction only, i.e. the NN is trained for model error correction only. Nevertheless, there is no obstacle to use this method to train the NN for other tasks (e.g. observation bias correction) provided that we are able to model their effect on the 4D-Var cost function.

Open Research

The numerical experiments in this article rely on OOPS and the FNN library (version 1.0.0). The source code of OOPS is property of ECMWF and is not publicy available. The source code of FNN (Farchi et al., 2022) is preserved at `10.5281/zenodo.7245291`, available via the MIT licence and developed openly at `https://github.com/cerea-daml/fnn`.

Acknowledgments

A. Farchi has benefited from a visiting grant of the ECMWF. CEREA is a member of Institut Pierre–Simon Laplace.

References

Abarbanel, H. D. I., Rozdeba, P. J., & Shirman, S. (2018, August). Machine learning: Deepest learning as statistical data assimilation problems. *Neural Computation*, *30*(8), 2025–2055. doi: {10.1162/neco_a_01094}

Arcomano, T., Szunyogh, I., Pathak, J., Wikner, A., Hunt, B. R., & Ott, E. (2020, May). A machine learning-based global atmospheric forecast model. *Geophysical Research Letters*, *47*(9). doi: 10.1029/2020GL087776

Arcucci, R., Zhu, J., Hu, S., & Guo, Y.-K. (2021, January). Deep data assimilation: Integrating deep learning with data assimilation. *Applied Sciences*, *11*(3), 1114. doi: 10.3390/app11031114

Asch, M., Bocquet, M., & Nodet, M. (2016). *Data assimilation: methods, algorithms, and applications* (No. 11). Philadelphia: SIAM, Society for Industrial and Applied Mathematics.

Bocquet, M., Brajard, J., Carrassi, A., & Bertino, L. (2019, July). Data assimilation as a learning tool to infer ordinary differential equation representations of dynamical models. *Nonlinear Processes in Geophysics*, *26*(3), 143–162. doi: 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. doi: 10.3934/fods.2020004

Bocquet, M., Farchi, A., & Malartic, Q. (2021). Online learning of both state and dynamics using ensemble kalman filters. *Foundations of Data Science*, *3*(2639-8001_2019_0_24), 305–330. doi: 10.3934/fods.2020015

Bolton, T., & Zanna, L. (2019, January). Applications of deep learning to ocean data inference and subgrid parameterization. *Journal of Advances in Modeling Earth Systems*, *11*(1), 376–399. doi: 10.1029/2018MS001472

Bonavita, M., & Laloyaux, P. (2020, December). Machine learning for model error inference and correction. *Journal of Advances in Modeling Earth Systems*, *12*(12). doi: 10.1029/2020MS002232

Bonavita, M., Trémolet, Y., Hólm, E., Lang, S., Chrust, M., Janiskova, M., … English, S. (2017). A strategy for data assimilation. , *800*. doi:

10.21957/TX1EPJD2P
Brajard, J., Carrassi, A., Bocquet, M., & Bertino, L. (2020, July). Combining data assimilation and machine learning to emulate a dynamical model from sparse and noisy observations: A case study with the lorenz 96 model. *Journal of Computational Science*, *44*, 101171. doi: 10.1016/j.jocs.2020.101171
Brajard, J., Carrassi, A., Bocquet, M., & Bertino, L. (2021, April). Combining data assimilation and machine learning to infer unresolved scale parametrization. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *379*(2194), 20200086. doi: 10.1098/rsta.2020.0086
Brunton, S. L., Proctor, J. L., & Kutz, J. N. (2016, April). Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, *113*(15), 3932–3937. doi: 10.1073/pnas.1517384113
Carrassi, A., Bocquet, M., Bertino, L., & Evensen, G. (2018, September). Data assimilation in the geosciences: An overview of methods, issues, and perspectives. *Wiley Interdisciplinary Reviews: Climate Change*, *9*(5), e535. doi: 10.1002/wcc.535
Chen, T.-C., Penny, S. G., Whitaker, J. S., Frolov, S., Pincus, R., & Tulich, S. N. (2022). Correcting systematic and state-dependent errors in the noaa fv3-gfs using neural networks. *Earth and Space Science Open Archive*, 22. Retrieved from `https://doi.org/10.1002/essoar.10511972.1` doi: 10.1002/essoar.10511972.1
Chollet, F. (2018). *Deep learning with python*. Shelter Island, New York: Manning Publications Co.
Courtier, P., Thépaut, J.-N., & Hollingsworth, A. (1994). A strategy for operational implementation of 4d-var using an incremental approach. *Quarterly Journal of the Royal Meteorological Society*, *120*, 1367–1388.
Crawford, W., Frolov, S., McLay, J., Reynolds, C. A., Barton, N., Ruston, B., & Bishop, C. H. (2020, September). Using analysis corrections to address model error in atmospheric forecasts. *Monthly Weather Review*, *148*(9), 3729–3745. doi: 10.1175/MWR-D-20-0008.1
Dee, D. P. (2005, October). Bias and data assimilation. *Quarterly Journal of the Royal Meteorological Society*, *131*(613), 3323–3343. doi: 10.1256/qj.05.137
Dueben, P. D., & Bauer, P. (2018, October). Challenges and design choices for global weather and climate models based on machine learning. *Geoscientific Model Development*, *11*(10), 3999–4009. doi: 10.5194/gmd-11-3999-2018
Evensen, G., Vossepoel, F. C., & van Leeuwen, P. J. (2022). *Data assimilation fundamentals: A unified formulation of the state and parameter estimation problem.* Springer Nature.
Fablet, R., Ouala, S., & Herzet, C. (2018, September). Bilinear residual neural network for the identification and forecasting of geophysical dynamics. In *2018 26th european signal processing conference (eusipco)* (pp. 1477–1481). Rome: IEEE. doi: 10.23919/EUSIPCO.2018.8553492
Farchi, A., Bocquet, M., Laloyaux, P., Bonavita, M., & Malartic, Q. (2021, October). A comparison of combined data assimilation and machine learning methods for offline and online model error correction. *Journal of Computational Science*, *55*, 101468. doi: 10.1016/j.jocs.2021.101468
Farchi, A., Chrust, M., Bocquet, M., Laloyaux, P., & Bonavita, M. (2022, 10). *The Fortran Neural Network (FNN) library.* Retrieved from `https://github.com/cerea-daml/fnn` doi: 10.5281/zenodo.7245291
Farchi, A., Laloyaux, P., Bonavita, M., & Bocquet, M. (2021, July). Using machine learning to correct model error in data assimilation and forecast applications. *Quarterly Journal of the Royal Meteorological Society*, *147*(739), 3067–3084. doi: 10.1002/qj.4116
Fisher, M., & Gürol, S. (2017, January). Parallelization in the time dimension of four-

dimensional variational data assimilation: Parallelization of 4d-var. *Quarterly Journal of the Royal Meteorological Society*, *143*(703), 1136–1147. doi: 10.1002/qj.2997

Fisher, M., Trémolet, Y., Auvinen, H., Tan, D., & Poli, P. (2011). Weak-constraint and long-window 4d-var. , *655*. doi: 10.21957/9ii4d4dsq

Gagne, D. J., Christensen, H. M., Subramanian, A. C., & Monahan, A. H. (2020, March). Machine learning for stochastic parameterization: Generative adversarial networks in the lorenz '96 model. *Journal of Advances in Modeling Earth Systems*, *12*(3). doi: 10.1029/2019MS001896

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning.* Cambridge, Massachusetts: The MIT Press.

Gottwald, G. A., & Reich, S. (2021, September). Supervised learning from noisy observations: Combining machine-learning techniques with data assimilation. *Physica D: Nonlinear Phenomena*, *423*, 132911. doi: 10.1016/j.physd.2021.132911

Hamilton, F., Berry, T., & Sauer, T. (2016, March). Ensemble kalman filtering without a model. *Physical Review X*, *6*(1), 011021. doi: 10.1103/PhysRevX.6.011021

Jazwinski, A. H. (1970). *Stochastic processes and filtering theory* (No. 64). San Diego: Acad. Press.

Jia, X., Willard, J., Karpatne, A., Read, J., Zwart, J., Steinbach, M., & Kumar, V. (2019, May). Physics guided RNNs for modeling dynamical systems: A case study in simulating lake temperature profiles. In *Proceedings of the 2019 siam international conference on data mining* (pp. 558, 566). Philadelphia, PA: Society for Industrial and Applied Mathematics. doi: 10.1137/1.9781611975673

Kalnay, E. (2003). *Atmospheric modeling, data assimilation and predictability.* Cambridge University Press, Cambridge.

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. In Y. Bengio & Y. LeCun (Eds.), *3rd international conference on learning representations, iclr 2015, san diego, ca, usa, may 7-9, 2015, conference track proceedings.* San Diego, CA, USA.

Laloyaux, P., Bonavita, M., Chrust, M., & Gürol, S. (2020, October). Exploring the potential and limitations of weak-constraint 4d-var. *Quarterly Journal of the Royal Meteorological Society*, *146*(733), 4067–4082. doi: 10.1002/qj.3891

Laloyaux, P., Bonavita, M., Dahoui, M., Farnan, J., Healy, S., Hólm, E., & Lang, S. T. K. (2020, July). Towards an unbiased stratospheric analysis. *Quarterly Journal of the Royal Meteorological Society*, *146*(730), 2392–2409. doi: 10.1002/qj.3798

Laloyaux, P., Kurth, T., Dueben, P. D., & Hall, D. (2022, June). Deep learning to estimate model biases in an operational nwp assimilation system. *Journal of Advances in Modeling Earth Systems*, *14*(6). doi: 10.1029/2022MS003016

Law, K., Stuart, A., & Zygalakis, K. (2015). *Data assimilation* (Vol. 62). Cham: Springer International Publishing. doi: 10.1007/978-3-319-20325-6

LeCun, Y., Bengio, Y., & Hinton, G. (2015, May). Deep learning. *Nature*, *521*(7553), 436–444. doi: 10.1038/nature14539

Lguensat, R., Tandeo, P., Ailliot, P., Pulido, M., & Fablet, R. (2017, October). The analog data assimilation. *Monthly Weather Review*, *145*(10), 4093–4107. doi: 10.1175/MWR-D-16-0441.1

Malartic, Q., Farchi, A., & Bocquet, M. (2022, June). State, global, and local parameter estimation using local ensemble kalman filters: Applications to online machine learning of chaotic dynamics. *Quarterly Journal of the Royal Meteorological Society*, qj.4297. doi: 10.1002/qj.4297

Ott, J., Pritchard, M., Best, N., Linstead, E., Curcic, M., & Baldi, P. (2020, August). A fortran-keras deep learning bridge for scientific computing. *Scientific Programming*, *2020*, 1–13. doi: 10.1155/2020/8888811

Pathak, J., Hunt, B., Girvan, M., Lu, Z., & Ott, E. (2018, January). Model-free prediction of large spatiotemporally chaotic systems from data: A reservoir computing approach. *Physical Review Letters*, *120*(2), 024102. doi: 10.1103/PhysRevLett.120.024102

Pathak, J., Wikner, A., Fussell, R., Chandra, S., Hunt, B. R., Girvan, M., & Ott, E. (2018, April). Hybrid forecasting of chaotic processes: Using machine learning in conjunction with a knowledge-based model. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, *28*(4), 041101. doi: 10.1063/1.5028373

Polichtchouk, I., Wedi, N., & Kim, Y.-H. (2022, January). Resolved gravity waves in the tropical stratosphere: Impact of horizontal resolution and deep convection parametrization. *Quarterly Journal of the Royal Meteorological Society*, *148*(742), 233–251. doi: 10.1002/qj.4202

Rasp, S., Pritchard, M. S., & Gentine, P. (2018, September). Deep learning to represent subgrid processes in climate models. *Proceedings of the National Academy of Sciences*, *115*(39), 9684–9689. doi: 10.1073/pnas.1810286115

Reich, S., & Cotter, C. (2015). *Probabilistic forecasting and bayesian data assimilation.* Cambridge: Cambridge University Press. doi: 10.1017/CBO9781107706804

Sakov, P., Haussaire, J.-M., & Bocquet, M. (2018, April). An iterative ensemble kalman filter in the presence of additive model error. *Quarterly Journal of the Royal Meteorological Society*, *144*(713), 1297–1309. doi: 10.1002/qj.3213

Scher, S., & Messori, G. (2019, November). Generalization properties of feed-forward neural networks trained on lorenz systems. *Nonlinear Processes in Geophysics*, *26*(4), 381–399. doi: 10.5194/npg-26-381-2019

Trémolet, Y. (2006, October). Accounting for an imperfect model in 4d-var. *Quarterly Journal of the Royal Meteorological Society*, *132*(621), 2483–2504. doi: 10.1256/qj.05.224

Watson, P. A. G. (2019, May). Applying machine learning to improve simulations of a chaotic dynamical system using empirical error correction. *Journal of Advances in Modeling Earth Systems*, *11*(5), 1402–1417. doi: 10.1029/2018MS001597

Weyn, J. A., Durran, D. R., & Caruana, R. (2019, August). Can machines learn to predict weather? using deep learning to predict gridded 500-hpa geopotential height from historical weather data. *Journal of Advances in Modeling Earth Systems*, *11*(8), 2680–2693. doi: 10.1029/2019MS001705

Wikner, A., Pathak, J., Hunt, B., Girvan, M., Arcomano, T., Szunyogh, I., … Ott, E. (2020, May). Combining machine learning with knowledge-based modeling for scalable forecasting and subgrid-scale closure of large, complex, spatiotemporal systems. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, *30*(5), 053111. doi: 10.1063/5.0005541