Exploring machine learning for data assimilation

Alban Farchi[†], Patrick Laloyaux[‡], Massimo Bonavita[‡], and Marc Bocquet[†]

[†] CEREA, joint laboratory École des Ponts ParisTech and EDF R&D, Université Paris-Est, Champs-sur-Marne, France
[‡] ECMWF, Shinfield Park, Reading, United Kingdom

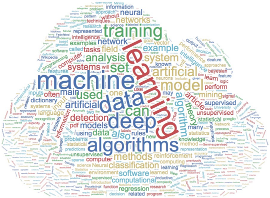
Thursday the 7th of May, 2020

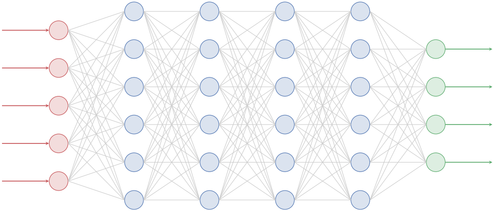
Machine Learning seminar series

Contents

The emergence of machine learning

- *Machine learning* (ML) methods, and in particular *deep learning* (DL), have recently demonstrated impressive skills in reproducing complex spatiotemporal processes.
- The emergence of DL is largely due to:
  - the development of efficient and user-friendly libraries;
  - the increasing computational capabilities (and in particular the use of GPUs);
  - the access to (very) large datasets for training.





Machine learning and optimisation

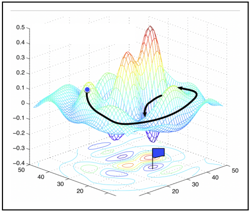
Definition

Machine learning (ML) algorithms build a *mathematical model* based on sample data, known as "training data", in order to make predictions or decisions without being explicitly programmed to perform the task.

- In most cases, the goal is to *minimise a loss function* which expresses the discrepancy between the model prediction and the data:

$$\mathbf{w}^* = \underset{\mathbf{w}\in\mathbb{R}^{N_\mathrm{p}}}{\arg\min} \sum_{i=1}^{N_\mathrm{e}} \left\| \mathbf{y}_i - \mathcal{M}(\mathbf{w}, \mathbf{x}_i) \right\|_2^2. \tag{1}$$

  - The set $\left\{(\mathbf{x}_i, \mathbf{y}_i), i = 1 \ldots N_\mathrm{e}\right\}$ is the *training data*.
  - The *model* $\mathcal{M}$ depends on a set of *parameters* $\mathbf{w} \in \mathbb{R}^{N_\mathrm{p}}$.
- This approach is called *supervised learning*.
- In this sense, ML is not that far from *data assimilation* (DA).



Machine learning for numerical weather prediction

- Suppose that $\psi(t)$ is the trajectory of a physical system and define

$$\mathbf{x}_i = \psi\big(i \times \Delta t\big), \tag{2a}$$

$$\mathbf{y}_i = \psi\big((i+1) \times \Delta t\big). \tag{2b}$$

- Then the ML problem (1) consists in finding the best approximation of the map $\psi(t) \mapsto \psi(t+\Delta t)$, *i.e.*, the *resolvent* of $\psi(t)$, among all models

$$\Big\{\mathcal{M} : \mathbf{x} \mapsto \mathcal{M}(\mathbf{w}, \mathbf{x}),\ \mathbf{w} \in \mathbb{R}^{N_\mathrm{p}}\Big\}. \tag{3}$$

Machine learning for numerical weather prediction

- Such approaches has been used to reconstruct the dynamics of low-order models (Lorenz 1963, Lorenz 1996, Kuramoto–Sivashinski) using different variants:
  - recurrent neural network [Park and Zhu, 1994];
  - reservoir computing [Pathak et al., 2017, 2018];
  - artificial neural networks [Dueben and Bauer, 2018].
- In these examples, the trajectory of the system is *perfectly known*.
- By contrast, observations in NWP are *sparse* and *noisy*: we need DA to recover the full state and to filter the noise!
- A rigorous formalism for this problem is that of a DA system with the model parameters $\mathbf{w}$ inside the control vector [Bocquet et al., 2019, 2020; Brajard et al., 2020].

The data assimilation problem

- In this case, the cost function to minimise is

$$\mathcal{J}(\mathbf{w}, \mathbf{x}_0, \dots, \mathbf{x}_{N_\mathrm{t}}) = \frac{1}{2}\sum_{k=0}^{N_\mathrm{t}} \left\| \mathbf{y}_k - \mathcal{H}_k(\mathbf{x}_k) \right\|^2_{\mathbf{R}_k^{-1}} + \frac{1}{2}\sum_{k=0}^{N_\mathrm{t}-1} \left\| \mathbf{x}_{k+1} - \mathcal{M}_k(\mathbf{w}, \mathbf{x}_k) \right\|^2_{\mathbf{Q}_k^{-1}}, \quad (4)$$

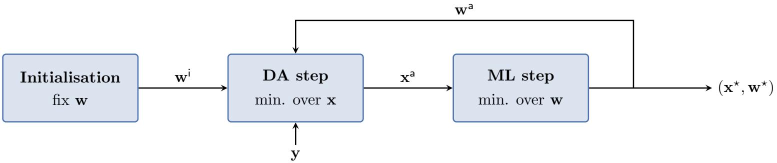
  - $\mathbf{x}_k \in \mathbb{R}^{N_\mathrm{x}}$ is the *state* at time t_k;
  - $\mathbf{y}_k \in \mathbb{R}^{N_\mathrm{y}}$ is the *observation vector* at time t_k;
  - $\mathbf{w} \in \mathbb{R}^{N_\mathrm{p}}$ is the set of *parameters of the surrogate model* $\mathcal{M}_k$ (*e.g.*, the weights of an artificial neural network);
  - N_t is the length of the *assimilation* or *training window*.
- This resemble a typical weak-constraint 4D-Var cost function!
- If $\mathcal{H}_k = \mathrm{Id}$ (full observations) and $\mathbf{R} = \mathbf{0}$ (no observation noise), we recover the standard ML cost function.

Joint minimisation of $\mathbf{w}$ and $\mathbf{x}$

- If possible, one can optimise for the parameters $\mathbf{w}$ and the trajectory $\mathbf{x}_0, \ldots, \mathbf{x}_{N_\mathrm{t}}$ *at the same time*.
- This method has been used by [Bocquet at al., 2019] to reconstruct the dynamics of the Lorenz 1996 and Kuramoto–Sivashinski models, with as few parameters as possible.
- For realistic models, a joint minimisation is very difficult to implement:
  - the state space $\mathbb{R}^{N_\mathrm{x}}$ is already *high-dimensional*;
  - in order to get an accurate description of the dynamics, the *number of parameters* N_p and the length of the *training window* N_t must be large enough.

Coordinate descent: alternate ML and DA

- Because the parameters $\mathbf{w}$ and the trajectory $\mathbf{x}_0, \ldots, \mathbf{x}_{N_\mathrm{t}}$ are of different nature, it could be more efficient to use a *coordinate descent*, in which we alternate
  - a standard *DA step*: minimisation over $(\mathbf{x}_0, \ldots, \mathbf{x}_{N_\mathrm{t}})$;
  - a standard *ML step*: minimisation over $\mathbf{w}$.
- The algorithm is flexible: the DA and ML methods are independent.
- In this framework, the use of ML is more technical than conceptual.



Coordinate descent: alternate ML and DA

- This method has been used by [Bocquet et al., 2020] and [Brajard et al., 2020] to reconstruct the dynamics of the Lorenz 1996 and Lorenz 2005 (two-scale) models using *convolutional neural networks*.
- The extension to realistic models is not immediate:
  - the algorithm initialisation is critical;
  - the convergence is not guaranteed and we cannot afford many DA steps.
- Instead of constructing the model from scratch, we could build a *hybrid* model using an already existent model:

$$\mathcal{M}_k^{\mathsf{h}} : (\mathbf{w}, \mathbf{x}) \mapsto \mathcal{M}_k^{\mathsf{o}}(\mathbf{x}) + \mathcal{M}_k^{\mathsf{ml}}(\mathbf{w}, \mathbf{x}), \tag{5}$$

where $\mathcal{M}^{\mathsf{o}}$ is the *original model* and $\mathcal{M}^{\mathsf{ml}}$ is the *trainable model*.

The hybrid model

- In this case, the ML cost can be rewritten as

$$\mathcal{J}^{\mathsf{ml}}(\mathbf{w}, \mathbf{x}_0, \ldots, \mathbf{x}_{N_\mathrm{t}}) = \frac{1}{2}\sum_{k=0}^{N_\mathrm{t}-1} \left\| \mathbf{x}_{k+1} - \mathcal{M}_k^{\mathsf{h}}(\mathbf{w}, \mathbf{x}_k) \right\|^2_{\mathbf{Q}_k^{-1}}, \tag{6}$$

$$= \frac{1}{2}\sum_{k=0}^{N_\mathrm{t}-1} \left\| \mathbf{x}_{k+1} - \mathcal{M}_k^{\circ}(\mathbf{x}_k) - \mathcal{M}_k^{\mathsf{ml}}(\mathbf{w}, \mathbf{x}_k) \right\|^2_{\mathbf{Q}_k^{-1}}. \tag{7}$$

- Therefore, the trainable model $\mathcal{M}^{\mathsf{ml}}$ has to learn the relationship

$$\mathbf{x}_k \mapsto \mathbf{x}_{k+1} - \mathcal{M}_k^{\circ}(\mathbf{x}_k) = \eta_{k+1}, \tag{8}$$

in other words the *model error* associated to $\mathcal{M}^{\circ}$.

Short summary

How to estimate the full model dynamics using perfect observations (full and noiseless)?

$$\mathcal{J}(\mathbf{w}) = \frac{1}{2}\sum_{k=0}^{N_\mathrm{t}-1}\left\|\mathbf{x}_{k+1} - \mathcal{M}_k^{\mathsf{ml}}(\mathbf{w}, \mathbf{x}_k)\right\|^2_{\mathbf{Q}_k^{-1}}.$$

How to estimate the full model dynamics using sparse and noisy observations?

$$\mathcal{J}(\mathbf{w}, \mathbf{x}_\star) = \frac{1}{2}\sum_{k=0}^{N_\mathrm{t}}\left\|\mathbf{y}_k - \mathcal{H}_k(\mathbf{x}_k)\right\|^2_{\mathbf{R}_k^{-1}} + \frac{1}{2}\sum_{k=0}^{N_\mathrm{t}-1}\left\|\mathbf{x}_{k+1} - \mathcal{M}_k^{\mathsf{ml}}(\mathbf{w}, \mathbf{x}_k)\right\|^2_{\mathbf{Q}_k^{-1}}.$$

How to estimate the model error using sparse and noisy observations?

$$\mathcal{J}(\mathbf{w}, \mathbf{x}_\star) = \frac{1}{2}\sum_{k=0}^{N_\mathrm{t}}\left\|\mathbf{y}_k - \mathcal{H}_k(\mathbf{x}_k)\right\|^2_{\mathbf{R}_k^{-1}} + \frac{1}{2}\sum_{k=0}^{N_\mathrm{t}-1}\left\|\mathbf{x}_{k+1} - \mathcal{M}_k^{\circ}(\mathbf{x}_k) - \mathcal{M}_k^{\mathsf{ml}}(\mathbf{w}, \mathbf{x}_k)\right\|^2_{\mathbf{Q}_k^{-1}}.$$

Learning the model error

- We want to validate this approach using the framework developed at ECMWF (namely OOPS).
- The observations will be generated using the *QG model*:
  - a reasonably complex problem (2D, 2 layers, 1600 variables in total);
  - sufficiently small to perform extensive tests;
  - it has been used to validate the weak-constraint 4D-Var algorithm [Laloyaux et al., 2020].
- The DA step will be performed using the *strong-constraint 4D-Var* algorithm:
  - the original model $\mathcal{M}^{\circ}$ is to be determined (perturbed QG model);
  - the total training window will be divided into smaller sub-windows.
- The ML step will be performed with *standard ML tools*:
  - the trainable model $\mathcal{M}^{\mathrm{ml}}$ will be built with artificial neural networks;
  - the optimisation is left to TensorFlow 2.

Contents

Brief model description

- The model expresses the *conservation of potential vorticity* q for two layers of constant potential temperature *in the $x - y$ plane* (two-dimensional model):

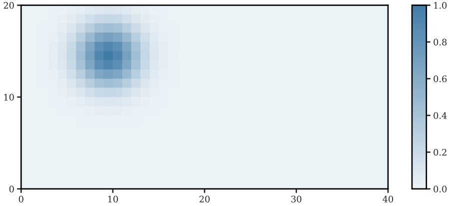
$$\frac{\mathrm{d}q_1}{\mathrm{d}t} = \frac{\mathrm{d}q_2}{\mathrm{d}t} = 0. \tag{9}$$

- The potential vorticity q is related to the *stream function* ψ by

$$q_1 = \Delta\psi_1 - F_1(\psi_1 - \psi_2) + \beta y, \tag{10a}$$

$$q_2 = \Delta\psi_2 - F_2(\psi_2 - \psi_1) + \beta y + R(x, y). \tag{10b}$$

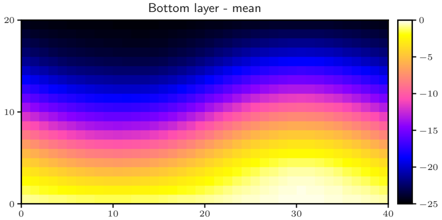
- The domain is *periodic* in the x direction and *fixed boundary conditions* are used for q in the y direction. We use a discretisation of 40×20 points.
- The *orography* R is characterised by a Gaussian hill.



Dynamical behaviour

[show animation here]

- We have first checked that the model is stable and realistic for very long runs.
- The evolution of ψ is characterised by a *slow westward motion*, with a mean period around $16\,\mathrm{d}$.
- The model is *chaotic*, with a doubling time of errors around $250\,\mathrm{h}$.
- For comparison, the doubling time of errors in the IFS is around $2\,\mathrm{d}$.



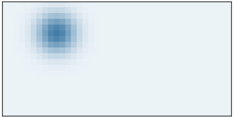
Data assimilation with the QG model

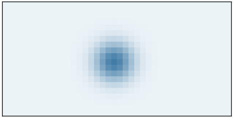
- We use the DA setup of [Laloyaux et al., 2020].
- The *control vector* is ψ: the state dimension is $N_\mathrm{x} = 40 \times 20 \times 2 = 1600$.
- Observations are available every $\Delta t = 2\,\mathrm{h}$ at $N_\mathrm{y} = 50$ *random locations*.
- The observation standard deviation is set to $\sigma = 0.1$, about $2\,\%$ of the model variability;
- The *4D-Var algorithm* is used with consecutive windows of $\Delta T = 1\,\mathrm{d}$.

The perturbed QG model: definition

- We add a model error on top of the exact QG model $\mathcal{M}^{\mathrm{t}}$ to create the original model $\mathcal{M}^{\mathrm{o}}$. The goal will be to recover the exact model $\mathcal{M}^{\mathrm{t}}$.
- In the weak-constraint 4D-Var test series [Laloyaux et al., 2020], the model error is a *random additive noise*, with a given covariance, and constant in time.
- However, such model noise makes the model unstable in the long term: we need another approach.
- For $\mathcal{M}^{\mathrm{o}}$ we use the QG model with *different parameters* (top and bottom layer depth, orography) and *different integration time step*.

Exact QG model $\mathcal{M}^{\mathrm{t}}$	Perturbed QG model $\mathcal{M}^{\mathrm{o}}$
$R_{\text{top}} = 6000\,\mathrm{m}$	$R_{\text{top}} = 5750\,\mathrm{m}$
$R_{\text{bot}} = 4000\,\mathrm{m}$	$R_{\text{bot}} = 4250\,\mathrm{m}$
$\delta t = 10\,\mathrm{min}$	$\delta t = 20\,\mathrm{min}$

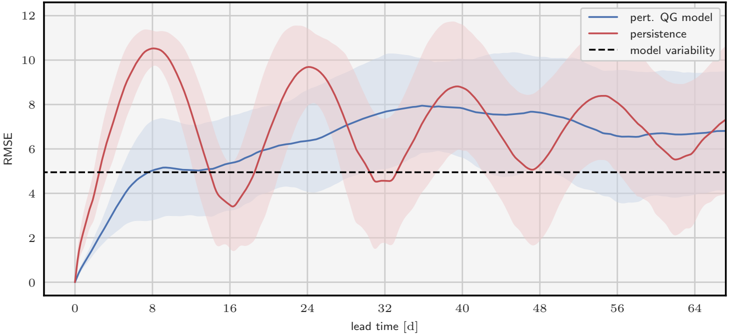




The perturbed QG model: forecast skill

- We compute the *forecast skill* (FS) of $\mathcal{M}^{\circ}$ compared to $\mathcal{M}^{\mathrm{t}}$:

$$\mathrm{FS}(t) = \mathrm{RMSE}\Big(\mathcal{M}^{\mathrm{t}}_{0\to t}(\mathbf{x}_0), \mathcal{M}^{\circ}_{0\to t}(\mathbf{x}_0)\Big) \tag{11}$$

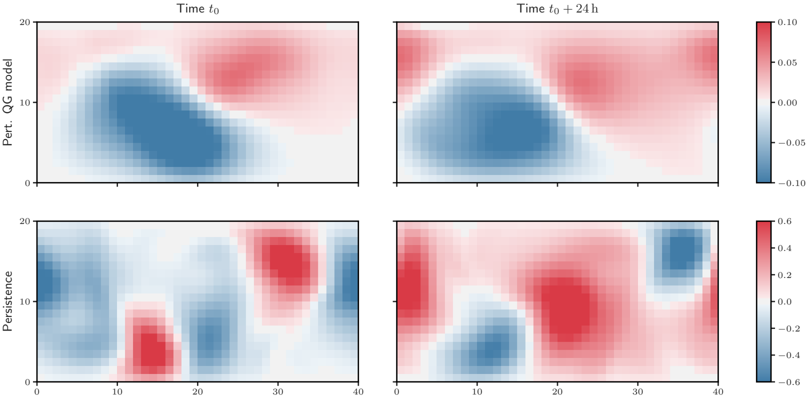


- The quantity is averaged over a large number (100) of initial conditions $\mathbf{x}_0$ to get a reliable estimate.
- NB: learning the model error starting from persistence (*i.e.*, when $\mathcal{M}^{\circ} = \mathrm{Id}$) is roughly equivalent to learning the full dynamics!

The perturbed QG model: model error

[show animation here]

- The resulting model error is dominated by the *orography* error.
- Compared to the error with persistence, it is *large scale* and it has a *slow* time evolution.



Contents

Idealised ML experiments

- Before starting the experiments with sparse and noisy observations, we want to evaluate the potential of ML.
  - What kind of model should be used? How should they be trained?
  - What level of improvement can we expect?
- Therefore, we first try to learn the model error using *perfect observations* (*i.e.*, full and noiseless).

Database creation

- We first make a long run with the exact QG model $\mathcal{M}^{\mathrm{t}}$ and we extract ψ at regular intervals:

$$\psi_k = \psi(k \times \Delta T), \quad k = 0, \dots, N_{\mathrm{t}}. \tag{12}$$

- Then, we compute the model error η for the perturbed QG model $\mathcal{M}^{\circ}$ as

$$\eta_{k+1} = \psi_{k+1} - \mathcal{M}^{\circ}(\psi_k), \quad k = 0, \dots, N_{\mathrm{t}} - 1. \tag{13}$$

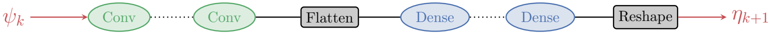
- Finally, the database for ML is

$$\Big\{ \big(\psi_{k-1}, \eta_k\big),\ k = 1, \dots, N_{\mathrm{t}} \Big\}. \tag{14}$$

- The process is repeated 18 times:
  - one trajectory is used for *training*;
  - one trajectory is used for *validation* (when to stop the training);
  - 16 trajectories are used for *testing*.
- This experiment has two hyperparameters:
  - the *sampling period* $\Delta T \quad \rightarrow \quad 1,\ 2,\ 4,\ 8\,\mathrm{d}$;
  - the *size of the database* $N_{\mathrm{t}} \quad \rightarrow \quad 16,\ 32,\ 64,\ \dots,\ 1024.$

Machine learning models

- The trainable model $\mathcal{M}^{\mathrm{ml}}$ is built using artificial neural networks.
- Two classes of architectures are considered:
  - sequential models with only *dense* or *fully-connected* layers;
  - sequential models with *convolutional* layers followed by *dense* layers.



Machine learning models

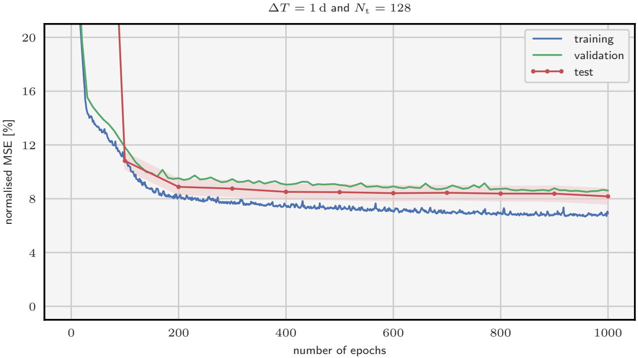
- The models are implemented using TensorFlow (only a few lines of python code).
- For each experiment, 24 models are trained, with variation of
  - the *number of layers* $\rightarrow$ $1, \ldots, 4$;
  - the *number of nodes* or *filters* per layer $\rightarrow$ $4,\ 8,\ 16$;
  - the *activation function* $\rightarrow$ linear or relu $x \mapsto \frac{1}{2}(x + |x|)$.
- The models are designed to use as few parameters as possible (N_p is between 10^4 and 10^5) because the problem is small ($N_\mathrm{x} = 1600$).
- Regularisation is empirically unnecessary in our experiments.

Model training

- The models are trained using *Adam*, a variant of the *stochastic gradient descent* implemented in TensorFlow.
- The model input / output are normalised to accelerate the convergence.
- The loss function is the *mean squared error* (MSE).

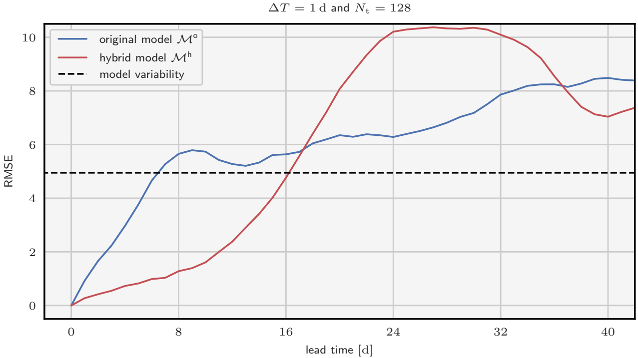
- The training consists of:
  - 10^3 epochs with an initial learning rate of 10^{-3};
  - 10^3 epochs with an initial learning rate of 10^{-4} (fine-tuning);
  - in each case, we keep the model with the lowest validation MSE.

Training example



- Trainable model $\mathcal{M}^{\mathsf{ml}}$: 1 dense layer with 4 nodes, linear activation, 14 404 parameters in total.
- The model learns about $92\,\%$ of the model error variance.

Corrected forecast skill



- The trained model is included in the hybrid model $\mathcal{M}^\mathrm{h} = \mathcal{M}^\mathrm{o} + \mathcal{M}^\mathrm{ml}$ and tested in forecast condition:

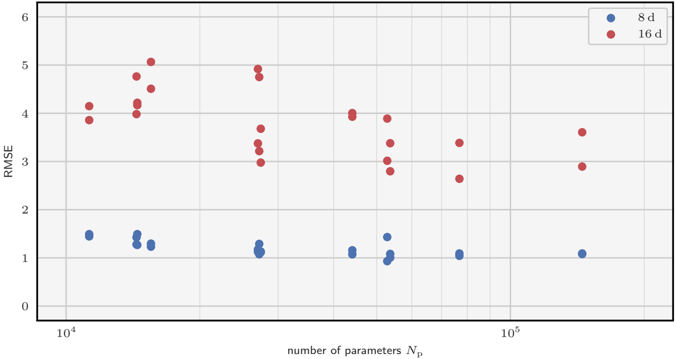
$$\mathrm{FS}(t) = \mathrm{RMSE}\Big(\mathcal{M}^\mathrm{t}_{0\to t}(\mathbf{x}_0), \mathcal{M}^\mathrm{h}_{0\to t}(\mathbf{x}_0)\Big) \tag{15}$$

- The FS is averaged over 16 different initial conditions $\mathbf{x}_0$ to get a reliable estimate.
- The correction is still effective after a $10\,\mathrm{d}$ forecast!

Next steps

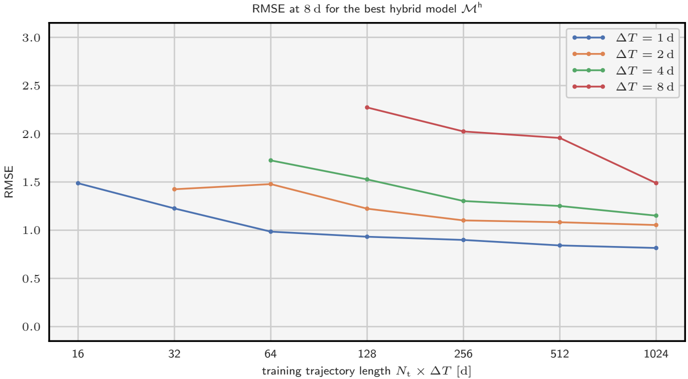
- What are the results for the other models? Which are the best models?
- What happens if we change the sampling period ΔT or the size of the database N_t?
- How does this compare to persistence?

Comparative forecast skill



- There is a clear tendency: the more parameters, the lower the RMSE.
- The spread at $16\,\mathrm{d}$ is much larger than at $8\,\mathrm{d}$.

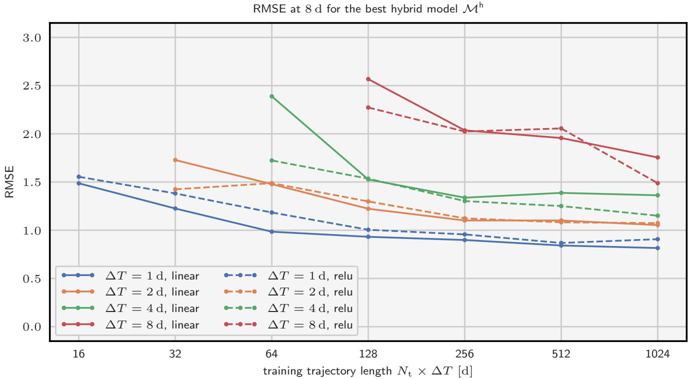
How long must be the training trajectory?



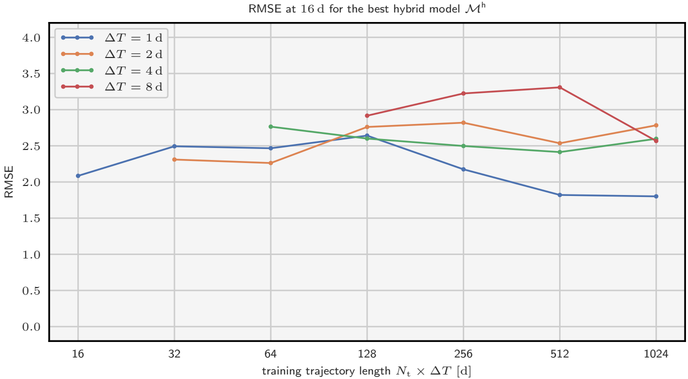
- At fixed sampling period ΔT, the $8\,\mathrm{d}$ forecast improves with the length of the training trajectory.
- Globally the RMSE improves as the sampling period ΔT is decreased, even though this means using more hybrid model cycles.

Which are the best models?

- Increasing the number of parameters (*e.g.*, the *number of nodes*) is a good strategy.
- The *number of layers* and *layer types* (dense or convolutional) has little impact.
- *Nonlinear activation functions* are more efficient for small databases or if the sampling period ΔT is long. This is related to the development of nonlinearity for longer model forecast.

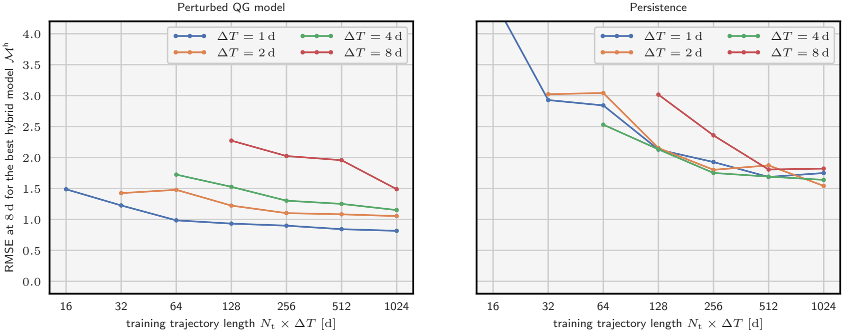


What about longer forecast?



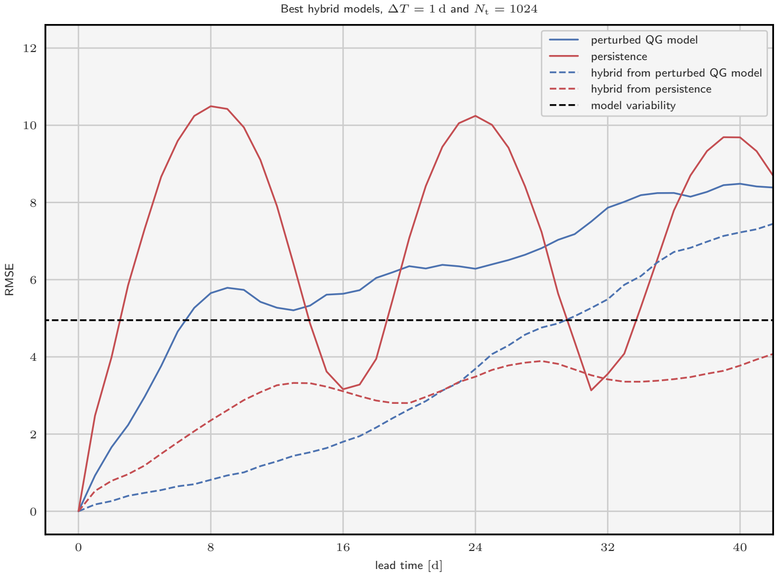
- The hybrid model $\mathcal{M}^{\mathrm{h}}$ clearly improves upon the original model $\mathcal{M}^{\mathrm{o}}$.
- However, increasing the length of the training trajectory N_{t} or decreasing the sampling period ΔT does not improve the RMSE.

Comparison with persistence



- The training is easier when $\mathcal{M}^{\circ}$ is the perturbed QG model than with persistence ($\mathcal{M}^{\circ} = \mathrm{Id}$).
- With the perturbed QG model, the RMSE is globally better and much better for small databases.
- With persistence, decreasing the sampling period ΔT does not significantly improve the RMSE.

Summary


Best hybrid models, $\Delta T = 1\,\mathrm{d}$ and $N_\mathrm{t} = 1024$
perturbed QG model
persistence
hybrid from perturbed QG model
hybrid from persistence
model variability
RMSE
lead time [d]
0
2
4
6
8
10
12
0
8
16
24
32
40

Contents

Coupled DA–ML experiments

- We are now ready to start the coupled DA–ML experiments.
- We first perform a single cycle: one DA step, followed by one ML step.
- We then evaluate our options.

The data assimilation step

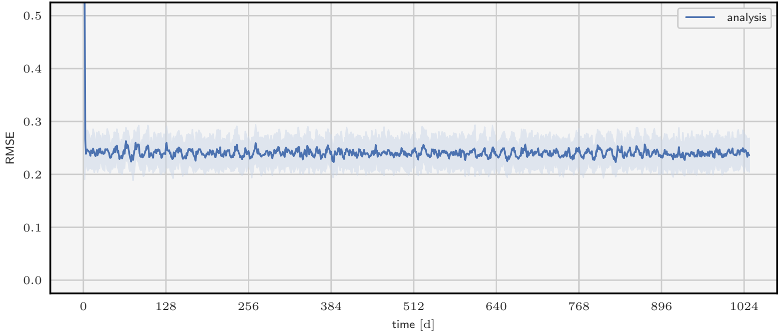
- Observations are available every $\Delta t = 2\,\mathrm{h}$ at $N_{\mathrm{y}} = 50$ *random locations*.
- The data assimilation step is performed with the *strong-constraint 4D-Var* algorithm:
  - we use windows of $\Delta T = 1\,\mathrm{d}$;
  - the algorithm uses the original (perturbed) QG model $\mathcal{M}^{\circ}$;
  - the observation standard deviation is set to $\sigma = 0.1$, about $2\,\%$ of the model variability;
  - the standard deviation of the background covariance matrix $\mathbf{B}$ is optimally tuned to yield the lowest analysis RMSE.

Database creation with data assimilation

- We first make a long run with the exact QG model $\mathcal{M}^{\mathsf{t}}$ and we extract ψ at regular intervals:
$$\psi_k^{\mathsf{t}} = \psi(k \times \Delta T), \quad k = 0, \ldots, N_{\mathrm{t}}. \tag{16}$$
- Then, for each window k, we generate the synthetic observations $\mathbf{y}_k$ (with noise) and we use the 4D-Var algorithm (with $\mathcal{M}^{\mathsf{o}}$) to compute the analysis ψ_k^{a} and the analysis increment $\delta\psi_{k+1}^{\mathsf{a}}$ as
$$\delta\psi_{k+1}^{\mathsf{a}} = \psi_{k+1}^{\mathsf{a}} - \mathcal{M}^{\mathsf{o}}(\psi_k^{\mathsf{a}}). \tag{17}$$
- Finally, the database for ML is
$$\left\{ \left(\psi_{k-1}^{\mathsf{a}}, \delta\psi_k^{\mathsf{a}}\right),\ k = 1, \ldots, N_{\mathrm{t}} \right\}. \tag{18}$$
- The process is repeated 18 times:
  - one trajectory is used for *training* ;
  - one trajectory is used for *validation* (when to stop the training) ;
  - 16 trajectories are used for *testing*.

Data assimilation results

- We have successfully applied the method for 1032 consecutive assimilation windows.
- The analysis RMSE stabilises after about 5 windows: we drop the first 8 windows.
- The time-averaged analysis RMSE, averaged over the 18 trajectories, is about 0.25.

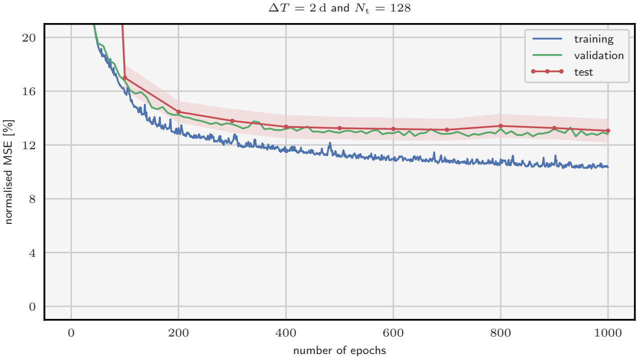


Machine learning models and training

- We can easily play with the *size of the database* by keeping only the first N_t elements.
- We can also play with the *sampling period* ΔT, for example by only keeping every other analysis state.

- We can now start the ML step with the *same models* and *same training method* as for the idealised experiments.

Training example



- Trainable model $\mathcal{M}^{\mathrm{ml}}$: 1 dense layer with 4 nodes, linear activation, 14 404 parameters in total.
- The model learns about $87\,\%$ of the *increments* variance.

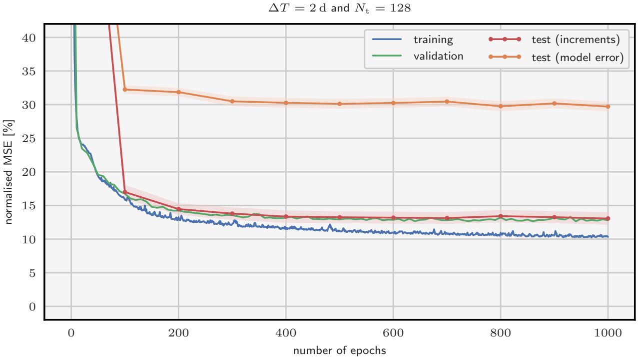
Further model evaluation

- The primary goal is to learn the *model error* and not the analysis increments.
- For each of the 16 test trajectory, we compute the model error using the truth:

$$\eta_{k+1} = \psi^{\mathrm{t}}_{k+1} - \mathcal{M}^{\circ}(\psi^{\mathrm{t}}_{k}), \quad k = 0, \ldots, N_{\mathrm{t}} - 1. \tag{19}$$

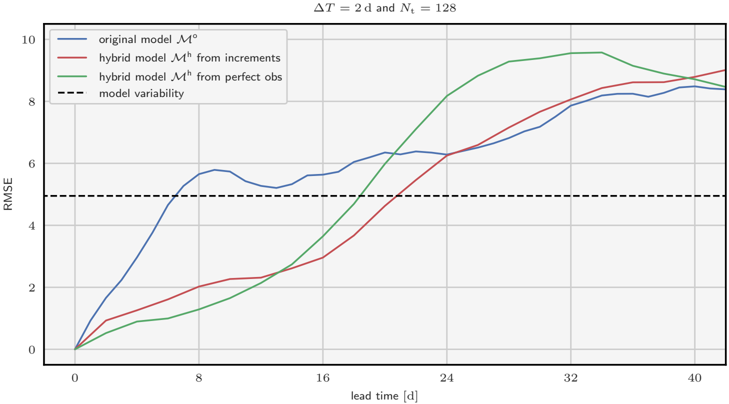
- We use this ideal database to test the trained model $\mathcal{M}^{\mathrm{ml}}$.

Further model evaluation



- Trainable model $\mathcal{M}^{\mathrm{ml}}$: 1 dense layer with 4 nodes, linear activation, 14 404 parameters in total.
- The model learns about $87\,\%$ of the *increments* variance, but only $70\,\%$ of the *model error* variance.

Corrected forecast skill



- The correction is less effective than if trained with perfect observations (full and noiseless), but still quite good!

Next steps

- Confirm these results with other ML models.
- As for the idealised experiments, change the size of the database N_t and the sampling period ΔT.
- *Use the hybrid model $\mathcal{M}^\mathrm{h}$ for data assimilation* and compare the results with other model error estimation methods, *e.g.*, weak-constraint 4D-Var.
  - We need to be able to perform forecasts *shorter* than the assimilation window ΔT.
  - An easy fix could be to assume a linear growth of the error in time.
  - Another option is to cycle the hybrid model $\mathcal{M}^\mathrm{h}$ between consecutive sampling times.

Using the hybrid model for data assimilation

- Suppose that we want to predict the model error for a $1\,\mathrm{d}$ integration using a sampling period of $2\,\mathrm{d}$.
- The *effective model* $\mathcal{M}^{\mathsf{e}}$ to train is

$$\mathcal{M}^{\mathsf{e}} : (\mathbf{w}, \mathbf{x}) \mapsto \mathcal{M}^{\mathsf{h}}\big(\mathbf{w}, \mathcal{M}^{\mathsf{h}}(\mathbf{w}, \mathbf{x})\big), \tag{20}$$

  where $\mathcal{M}^{\mathsf{h}}$ is the hybrid model:

$$\mathcal{M}^{\mathsf{h}} : (\mathbf{w}, \mathbf{x}) \mapsto \mathcal{M}^{\mathsf{o}}(\mathbf{x}) + \mathcal{M}^{\mathsf{ml}}(\mathbf{w}, \mathbf{x}). \tag{21}$$

- For the ML step, we need the gradient of $\mathcal{M}^{\mathsf{e}}$ with respect to $\mathbf{w}$:

$$\frac{\partial \mathcal{M}^{\mathsf{e}}}{\partial \mathbf{w}} = \frac{\partial \mathcal{M}^{\mathsf{ml}}}{\partial \mathbf{w}} + \frac{\partial \mathcal{M}^{\mathsf{ml}}}{\partial \mathbf{w}} \times \left\{ \frac{\partial \mathcal{M}^{\mathsf{o}}}{\partial \mathbf{x}} + \frac{\partial \mathcal{M}^{\mathsf{ml}}}{\partial \mathbf{x}} \right\} \circ \left\{ \mathcal{M}^{\mathsf{o}} + \mathcal{M}^{\mathsf{ml}} \right\}, \tag{22}$$

  which depends on the *tangent linear of the original model* $\mathcal{M}^{\mathsf{o}}$.
- We need to make OOPS and TensorFlow interact!

Contents

Conclusions

- ML tools can be used to learn either the *full model dynamics* or the *model error dynamics* of a system using only observations of the system.
- If the observations are *sparse* and *noisy*, ML must be coupled with DA:
  - DA is used to estimate the state of the system;
  - ML is used to learn the model or model error dynamics.

- With perfect observations of the *QG model*, we have shown that it is possible to learn the model or model error dynamics with only *simple artificial neural networks*.
- The best results are obtained when learning the model error dynamics instead of the full model dynamics.
- The application to sparse and noisy observations is on the way!

References

- Bocquet, M., Brajard, J., Carrassi, A., and Bertino, L.: *Data assimilation as a learning tool to infer ordinary differential equation representations of dynamical models*, Nonlin. Processes Geophys., 26, 2019.
- Bocquet, M., Brajard, J., Carrassi, A., and Bertino, L.: *Bayesian inference of chaotic dynamics by merging data assimilation, machine learning and expectation-maximization*, Foundations of Data Science, 2 (1), 2020.
- Brajard, J., Carrassi, A., Bocquet, M., and Bertino, L.: *Combining data assimilation and machine learning to emulate a dynamical model from sparse and noisy observations: a case study with the Lorenz 96 model*, J. Comput. Sci., 2020, *in revision*.
- Laloyaux, P., Bonavita, M., Chrust, M., and Gürol, S.: *Exploring the potential and limitations of weak-constraint 4D-Var*, Q. J. R. Meteorol. Soc, 2020.