Model error correction with data assimilation and machine learning

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

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

Tuesday, 16 November 2021

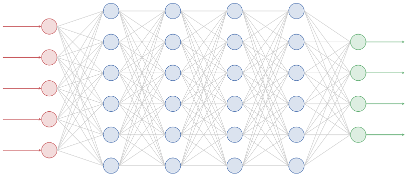
ESA-ECMWF workshop on Machine Learning for Earth System Observation and Prediction

Machine learning for NWP with dense and perfect observations

- A typical (supervised) machine learning problem: given observations $\mathbf{y}_k$ of a system, derive a *surrogate model* of that system.

$$\mathcal{J}(\mathbf{p}) = \sum_{k=1}^{N_\mathrm{t}} \left\| \mathbf{y}_{k+1} - \mathcal{M}(\mathbf{p}, \mathbf{y}_k) \right\|^2.$$

- $\mathcal{M}$ depends on a *set of coefficients* $\mathbf{p}$ (*e.g.*, the weights and biases of a neural network).
- This requires dense and perfect observations of the system. In NWP, observations are usually *sparse* and *noisy*: we need data assimilation!

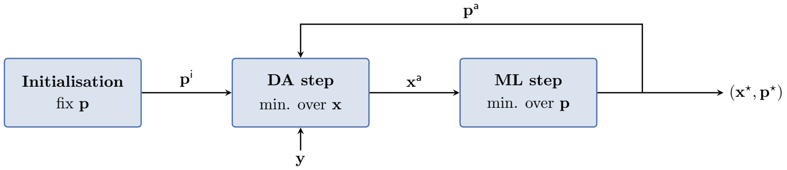


Machine learning for NWP with sparse and noisy observations

- A rigorous Bayesian formalism for this problem[1]:

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

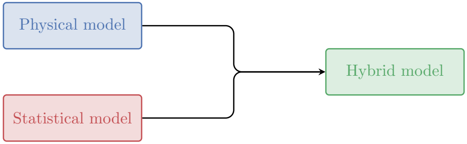
- This resembles a typical *weak-constraint 4D-Var* cost function!
- *DA* is used to estimate the state and then *ML* is used to estimate the model.



[1] Bocquet et al. (2019, 2020), Brajard et al. (2020)

Machine learning for NWP: learning model error

- Even though NWP models are not perfect, they are already quite good!
- Instead of building a surrogate model from scratch, we use the DA-ML framework to build a *hybrid* surrogate model, with a physical part and a statistical part[2].



- In practice, the statistical part is trained to learn the *error* of the physical model.
- In general, it is easier to train a correction model than a full model: we can use *smaller NNs* and *less training data*.

[2]Farchi et al. (2021), Brajard et al. (2021)

Typical architecture of a physical model

- The model is defined by a set of ODEs or PDEs which define the *tendencies*:

$$\frac{\partial \mathbf{x}}{\partial t} = \phi(\mathbf{x}). \tag{1}$$

- A numerical scheme is used to integrate the tendencies from time t to $t + \delta t$ (*e.g.*, a Runge–Kutta method):

$$\mathbf{x}(t + \delta t) = \mathcal{I}\Big(\mathbf{x}(t)\Big). \tag{2}$$

- Several integration steps are composed to define the *resolvent* from one analysis (or window) to the next:

$$\mathcal{M} : \mathbf{x}_k \mapsto \mathbf{x}_{k+1} = \mathcal{I} \circ \cdots \circ \mathcal{I}(\mathbf{x}_k) \tag{3}$$

A correction term can be applied:

1. *as an integrated correction* from one analysis to the next, *i.e.* in the resolvent (3);
2. or directly in the *tendencies* (1).

Resolvent or tendency correction with NNs

Resolvent correction

- The contributions of the physical model and of the NN are *independent*.
- As a consequence, the NN must simply predict the analysis increments.
- The resulting hybrid model is not suited for short-term predictions.

Tendency correction

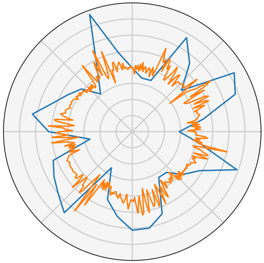
- The NN and the physical model are *entangled*.
- We cannot subtract the contribution of the physical model in the training dataset.
- We need the TL of the physical model to train the NN!
- The resulting hybrid model is suited for any prediction.

Resolvent or tendency correction: numerical comparison

- To compare the two approaches, we use a model implemented in TensorFlow for the *automatic differentiation*.
- The true model is the 2-scale Lorenz system (36 slow variables and 360 fast variables).
- The physical model (to correct) is the 1-scale Lorenz system with 36 variables.

Sources of model error

- the fast variables are not represented;
- the integration step is 0.05 instead of 0.005;
- the forcing constant is 8 instead of 10.



Model error correction: flowchart

- Run a long simulation run with the true model.
- Extract noisy observations from the slow variables.
- Run 4D-Var experiments with the physical (wrong) model (several thousands of windows).
- Use successive analyses to train the correction.

- The choice of the *DAW length* is critical here.
- Increasing the DAW length can help retrieving the model error signal in the analysis, but at the same time longer forecasts are usually harder to predict.
- We chose to use the DAW length which minimises the analysis error in the non-corrected experiment, in order to make the training data *as accurate as possible*.

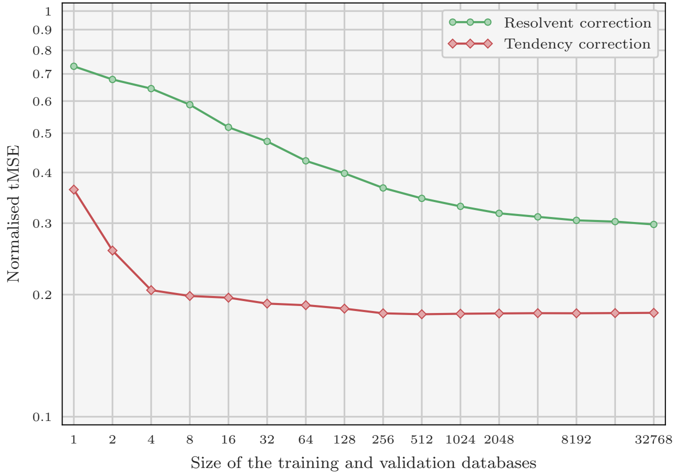
Resolvent correction (RC)

- Input layer (36 variables).
- 4 Conv1D layers, tanh activation.
- Output layer (36 variables).
- 4001 parameters in total.

Tendency correction (TC)

- Input layer (36 variables).
- 1 Conv1D layer, linear activation.
- Output layer (36 variables).
- 113 parameters in total.

Training results



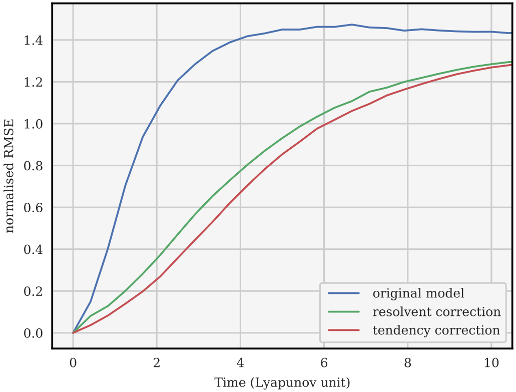
- In both cases, the correction is efficient and indeed reduces the prediction error.
- The TC is *more accurate* than the RC, even with a smaller NN and less training data.

Model error correction: flowchart

- The test MSE measures the accuracy of the forecast, but only for a forecast of one DAW.
- What about longer and shorter forecasts?

- Test the hybrid model in forecast mode.
- Test the hybrid model in assimilation mode.
  - For the resolvent correction, we need to assume a *linear growth of errors* with time.

Forecast skill and corrected analysis



Model	Analysis RMSE
Original model	0.31
Resolvent correction	0.28
Tendency correction	0.24
True model	0.22

- The TC benefits from the *interaction* with the physical model.
- The RC is highly penalised (in DA) by the assumption of linear growth of errors.

Online model error correction

- So far, the model error has been learnt *offline*: the ML (or training) step first requires a long analysis trajectory.
- We now investigate the possibility to make *online* learning, *i.e.* improving the correction as new observations become available.
- To do this, we use the formalism of DA to estimate both the state and the NN parameters:

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

- Information is flowing from one window to the next using the prior for the state $\mathbf{x}^{\mathrm{b}}$ and for the NN parameters $\mathbf{p}^{\mathrm{b}}$.
- For simplicity, we have neglected potential cross-covariance between state and NN parameters in the prior.

- This approach is very similar to classical *parameter estimation* in DA, and it can be seen as a NN formulation of weak-constraint 4D-Var.
- This has been already done in an EnKF context[3].

[3] Bocquet et al. (2020)

Online or offline model error correction: numerical comparison

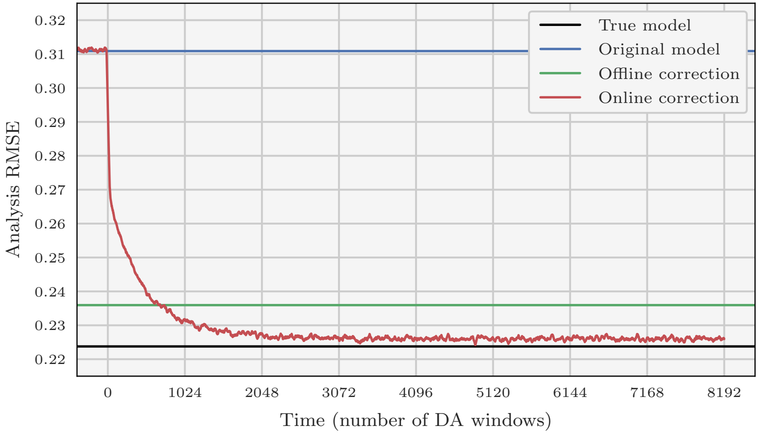
- Still with the 2-scale Lorenz model.
- To compare online and offline model error correction, we use the *tendency correction approach* (it does not require the assumption of linear growth of errors) and the same NN.

Tendency correction (TC)

- Input layer (36 variables).
- 1 Conv1D layer, lin. activation.
- Output layer (36 variables).
- 113 parameters in total.

- We start the experiment by using the (non-corrected) physical model.
- At some point, we switch on the online correction.

Online or offline model error correction: numerical comparison



- The online correction steadily improves the model.
- At some point, the online correction *gets more accurate* than the offline correction.
- Eventually, the improvement saturates. The analysis error is similar to that obtained with the true model!

Online learning: tuning the background error covariance

- Tuning the background error covariance matrix $\mathbf{B}_\mathsf{p}$ is critical for the success of this experiment.
- If $\mathbf{B}_\mathsf{p}$ is too small, the algorithm gives too much weight to the background, which makes the convergence *slower*.
- On the other hand, if $\mathbf{B}_\mathsf{p}$ is too large, the algorithm overfits the model parameters to the observation window, which can yield *divergence*.

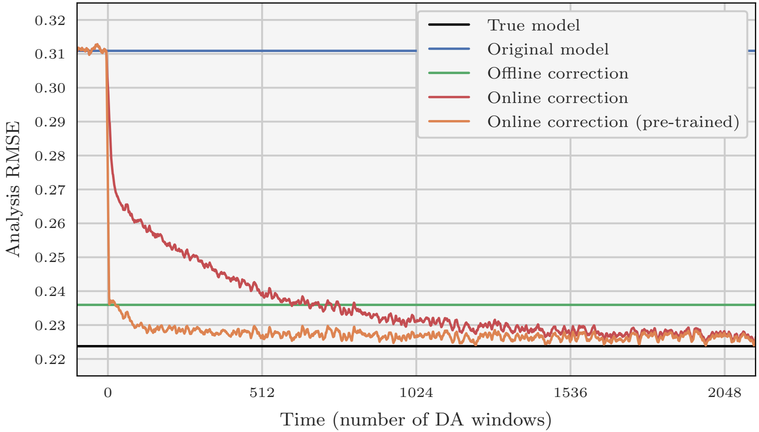
- In addition, $\mathbf{B}_\mathsf{p}$ should be larger at the start of the experiment than at the end, when the corrected model is more accurate.

- In the previous experiment, we used the following values:

$$\begin{aligned}\mathbf{B}_\mathsf{p} &= b_\mathsf{p}^2\mathbf{I},\\ b_\mathsf{p} &= \min\left[0.05, 0.001 + 0.1 \times \exp(-t/1024)\right],\end{aligned}$$

which we have empirically found to yield good performances.

Online learning: warm start



- Optionally, the parameters can be pre-trained, using offline learning.
- In this experiment, this results in a much faster convergence!

Conclusions

- Two frameworks for model error correction: resolvent or tendency correction.
- The resolvent correction is *easier to implement and train* because the physical and statistical models are independent.
- The tendency correction is *more technical* because the physical and statistical models are entangled, but the resulting hybrid models are potentially more accurate, in particular for data assimilation.

- Furthermore, the tendency correction framework opens the possibility to make *online learning*.
- This is similar to classical parameter estimation in data assimilation.
- As new observations become available, the model error correction improves and eventually gets *more accurate* than with the offline approach.

Paper published in JoCS: 10.1016/j.jocs.2021.101468