Combining data assimilation and machine learning to build data-driven models of chaotic dynamics

DA-based ML & ML-based DA

Marc Bocquet†, Quentin Malartic†, Alban Farchi†, Massimo Bonavita‡, Patrick Laloyaux‡, Marcin Chryst‡, Tobias Finn†, Charlotte Durand†, and Julien Brajard, Alberto Carrassi, Laurent Bertino.

†CEREA, École des Ponts and EDF R&D, Île-De-France, France
‡ECMWF, Reading, United Kingdom.

Exploring the frontiers in Earth system modeling with machine learning and big data
A typical (supervised) machine learning problem: given observations $y_k$ of a system, derive a \textit{surrogate model} of that system.

$$J(p) = \sum_{k=1}^{N_t} \|y_{k+1} - M(p, y_k)\|^2.$$  

$M$ depends on a \textit{set of coefficients} $p$ (e.g., the weights and biases of a neural network).

This requires dense and perfect observations of the system.

In NWP, observations are usually \textit{sparse} and \textit{noisy}: we need \textit{data assimilation}!
A rigorous Bayesian formalism for this problem:¹

\[ \mathcal{J}(p, x_0, \ldots, x_{N_t}) = \frac{1}{2} \sum_{k=0}^{N_t} \left\| y_k - \mathcal{H}_k(x_k) \right\|^2 R_k^{-1} + \frac{1}{2} \sum_{k=0}^{N_t-1} \left\| x_{k+1} - \mathcal{M}(p, x_k) \right\|^2 Q_k^{-1}. \]

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.

The problem can (almost) fully be solved from a Bayesian standpoint using the empirical **Expectation-Maximization** algorithm with an ensemble smoother². But it has a significant numerical cost.

---

¹[Bocquet et al. 2019; Bocquet et al. 2020; Brajard et al. 2020] in the wake of [Hsieh et al. 1998; Abarbanel et al. 2018]

²[Ghahramani et al. 1999; Nguyen et al. 2019; Bocquet et al. 2020]
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:\(^3\)

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.

But prone to initialisation shocks.\(^4\)

---

\(^3\) [Farchi et al. 2021; Brajard et al. 2021].

\(^4\) We learned it the hard way with sea-ice (VESRI/SASIP project).
Typical architecture of a physical model

- The model is defined by a set of ODEs or PDEs which define the tendencies:
  \[
  \frac{\partial \mathbf{x}}{\partial t} = \phi(\mathbf{x}).
  \] (1)

- A numerical scheme is used to integrate the tendencies from time \( t \) to \( t + \delta t \) (e.g., Runge–Kutta):
  \[
  \mathbf{x}(t + \delta t) = \mathcal{F}(\mathbf{x}(t)).
  \] (2)

- Several integration steps are composed to define the resolvent from one analysis (or window) to the next:
  \[
  \mathcal{M} : \mathbf{x}_k \mapsto \mathbf{x}_{k+1} = \mathcal{F} \circ \cdots \circ \mathcal{F}(\mathbf{x}_k).
  \] (3)

### Resolvent correction
- Physical model and of NN are independent.
- NN must predict the analysis increments.
- Resulting hybrid model not suited for short-term predictions.
- For DA, need to assume linear growth of errors in time to rescale correction.

### Tendency correction
- Physical model and NN are entangled.
- Need TL of physical model to train NN!
- Resulting hybrid model suited for any prediction.
- Can be used as is for DA.
Two-scale Lorenz model (L05III)

The two-scale Lorenz model (L05III) model: 36 slow & 360 fast variables, with equations:

\[
\begin{align*}
\frac{dx_n}{dt} &= \psi_n^+(\mathbf{x}) + F - \frac{c}{b} \sum_{m=0}^{9} u_{m+10n}, \\
\frac{du_m}{dt} &= \frac{c}{b} \psi_m^-(b\mathbf{u}) + \frac{c}{b} x_{m/10}, \quad \text{with} \quad \psi_n^\pm(\mathbf{x}) = x_{n\mp1}(x_{n\pm1} - x_{n\mp2}) - x_n,
\end{align*}
\]

Lyapunov time units
Numerical illustration with the two-scale Lorenz system

- The non-corrected model is the one-scale Lorenz system.
- Noisy observations are assimilated using strong-constrained 4D-Var.
- Simple CNNs are trained using the 4D-Var analysis.

![Graph showing model performance](image)

Data assimilation score

<table>
<thead>
<tr>
<th>Model</th>
<th>Analysis RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>No correction</td>
<td>0.31</td>
</tr>
<tr>
<td>Resolvent correction</td>
<td>0.28</td>
</tr>
<tr>
<td>Tendency correction</td>
<td>0.24</td>
</tr>
<tr>
<td>True model</td>
<td>0.22</td>
</tr>
</tbody>
</table>

- The tendencies corr. is more accurate than the resolvent corr., with smaller NNs and less training data.
- The tendencies corr. benefits from the interaction with the physical model.
- The resolvent corr. 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 perform 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:

\[
\mathcal{J}(p, x) = \frac{1}{2} \left\| x - x^b \right\|_{B_x^{-1}}^2 + \frac{1}{2} \left\| p - p^b \right\|_{B_p^{-1}}^2 + \frac{1}{2} \sum_{k=0}^{L} \left\| y_k - \mathcal{H}_k \circ \mathcal{M}_k^k(p, x) \right\|_{R_k^{-1}}^2.
\]

For simplicity, we have neglected potential cross-covariance between state and NN parameters in the prior.

Information is flowing from one window to the next using the prior for the state \(x^b\) and for the NN parameters \(p^b\).

 Already been investigated with an EnKF, with solutions.\(^5\)

\(^5\)[Bocquet et al. 2021; Malartic et al. 2022]
Numerical illustration with the same two-scale Lorenz system

- We use the tendency correction approach, with the same simple CNN as before, and still using 4D-Var.

- 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: towards an operational implementation with OOPS

- **Development of a fortran NN library** to interact with the fortran implementation of the forecast model.
- **Interfacing the NN library with OOPS** to estimate the NN parameters with DA.
- **Simplifications of the NN correction:**
  - the correction is additive, and added after each integration step (close to tendency correction);
  - the correction is computed independently for each atmospheric column\(^6\).
  - the correction is computed at the start of the DA window and not updated during the window;
  - in practice, it requires only *small adjustments* to the current WC 4D-Var already implemented.
- Demonstration with OOPS-QG with promising results, implementation with OOPS-IFS in progress.

\(^6\)[Bonavita et al. 2020]
Conclusions

Main messages:
- Bayesian DA view on joint state and model estimation. DA can address goals assigned to ML but with partial & noisy observations.
- Successful on 1D and 2D low-order models (L96, L05II, L96i, mL96, OOPS QG).

In progress: more ambitious models and datasets
- Application to the Marshall-Molteni 3-layer QG model on the sphere
- Application to the ERA5 and CMIP data (WeatherBench\textsuperscript{7}-like)
- Application to the ECMWF IFS
- Application to sea-ice surrogate modelling\textsuperscript{8}

\textsuperscript{7}[Rasp et al. 2020]
\textsuperscript{8}Schmidt Futures/VESRI/SASIP project


