Online model error correction with neural networks
From theory to the ECMWF forecasting system

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

[†] CEREA, joint laboratory École des Ponts ParisTech and EDF R&D, Île-de-France, France
[‡] ECMWF, Shinfield Park, Reading, United Kingdom

Friday, May 10 2024

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

Weak-constraint 4D-Var: the forcing formulation

- The idea of *weak-constraint 4D-Var* is to relax the perfect model assumption.
- The price to pay is a huge increase in problem dimensionality.
- This can be mitigated by making additional assumption, e.g. the model error $\mathbf{w}$ is constant over the DA window:

$$\mathbf{x}_{k+1} = \boldsymbol{\mathcal{M}}_{k+1:k}\left(\mathbf{x}_k\right) + \mathbf{w} \triangleq \boldsymbol{\mathcal{M}}^{\mathsf{wc}}_{k+1:0}\left(\mathbf{w}, \mathbf{x}_0\right).$$

- The cost function can hence be written

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

- This is called *forcing formulation* of weak-constraint 4D-Var. This is the weak-constraint 4D-Var currently implemented in OOPS (the ECMWF data assimilation system).

Weak-constraint 4D-Var: a neural network formulation

- Now suppose that the dynamical model is *parametrised* by a set of parameters $\mathbf{p}$ constant over the window:

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

- Following the same approach, the cost function becomes

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

- This approach can be seen as a *neural network formulation* of weak-constraint 4D-Var when $\mathbf{p}$ is the set of parameters (weights and biases) of a NN.

Neural network 4D-Var for model error correction

- In order to merge the two approaches, we consider the case where the *constant model error* $\mathbf{w}$ is *estimated using a neural network*:

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

- This means that the model evolution becomes

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

- As a consequence, it will be possible to build this simplified method on top of the *currently implemented weak-constraint* 4D-Var, in the *incremental assimilation* framework (with inner and outer loops).

Gradient of the incremental cost function

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

$\delta\mathbf{w} \leftarrow \mathbf{F}^{\mathsf{p}}\delta\mathbf{p} + \mathbf{F}^{\mathsf{x}}\delta\mathbf{x}_0$ ▷ TL of the NN $\mathcal{F}$
$\mathbf{z}_0 \leftarrow \mathbf{R}_0^{-1}\left(\mathbf{H}_0\delta\mathbf{x}_0 - \mathbf{d}_0\right)$
for $k = 1$ **to** $L-1$ **do**
    $\delta\mathbf{x}_k \leftarrow \mathbf{M}_{k:k-1}\delta\mathbf{x}_{k-1} + \delta\mathbf{w}$ ▷ TL of the dynamical model $\boldsymbol{\mathcal{M}}_{k:k-1}$
    $\mathbf{z}_k \leftarrow \mathbf{R}_k^{-1}\left(\mathbf{H}_k\delta\mathbf{x}_k - \mathbf{d}_k\right)$
end for
$\delta\tilde{\mathbf{x}}_{L-1} \leftarrow \mathbf{0}$ ▷ AD variable for system state
$\delta\tilde{\mathbf{w}}_{L-1} \leftarrow \mathbf{0}$ ▷ AD variable for model error
for $k = L-1$ **to** 1 **do**
    $\delta\tilde{\mathbf{x}}_k \leftarrow \mathbf{H}_k^{\top}\mathbf{z}_k + \delta\tilde{\mathbf{x}}_k$
    $\delta\tilde{\mathbf{w}}_{k-1} \leftarrow \delta\tilde{\mathbf{x}}_k + \delta\tilde{\mathbf{w}}_k$
    $\delta\tilde{\mathbf{x}}_{k-1} \leftarrow \mathbf{M}_{k:k-1}^{\top}\delta\tilde{\mathbf{x}}_k$ ▷ AD of the dynamical model $\boldsymbol{\mathcal{M}}_{k:k-1}$
end for
$\delta\tilde{\mathbf{x}}_0 \leftarrow \mathbf{H}_0^{\top}\mathbf{z}_0 + \delta\tilde{\mathbf{x}}_0$
$\delta\tilde{\mathbf{x}}_0 \leftarrow \left[\mathbf{F}^{\mathsf{x}}\right]^{\top}\delta\tilde{\mathbf{x}}_0$ ▷ AD of the NN $\mathcal{F}$
$\delta\tilde{\mathbf{p}} \leftarrow \left[\mathbf{F}^{\mathsf{p}}\right]^{\top}\delta\tilde{\mathbf{w}}_0$ ▷ AD of the NN $\mathcal{F}$
$\delta\tilde{\mathbf{x}}_0 \leftarrow \mathbf{B}^{-1}\left(\mathbf{x}_0^{\mathsf{i}} - \mathbf{x}_0^{\mathsf{b}} + \delta\mathbf{x}_0\right) + \delta\tilde{\mathbf{x}}_0$
$\delta\tilde{\mathbf{p}} \leftarrow \mathbf{P}^{-1}\left(\mathbf{p}^{\mathsf{i}} - \mathbf{p}^{\mathsf{b}} + \delta\mathbf{p}\right) + \delta\tilde{\mathbf{p}}$

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

Gradient of the incremental cost function

- In order to implement the simplified NN 4D-Var we can reuse most of the framework already in place for WC 4D-Var.
- A few *new bricks* need to be implemented:
  - the forward operator $\mathcal{F}$ of the NN to compute the nonlinear trajectory at the start of each outer iteration;
  - the tangent linear (TL) operators $\mathbf{F}^{\mathrm{x}}$ and $\mathbf{F}^{\mathrm{p}}$ of the NN;
  - the adjoint (AD) operators $[\mathbf{F}^{\mathrm{x}}]^{\top}$ and $[\mathbf{F}^{\mathrm{p}}]^{\top}$ of the NN.
- These operators have to be computed in the model core (where the components of the state are available), which is implemented in Fortran.

- To do so, we have implemented our own *NN library in Fortran*.

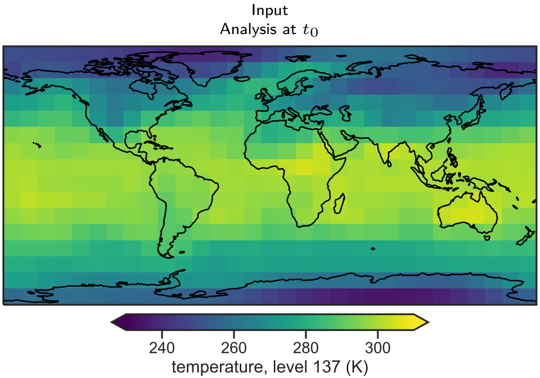
`https://github.com/cerea-daml/fnn`

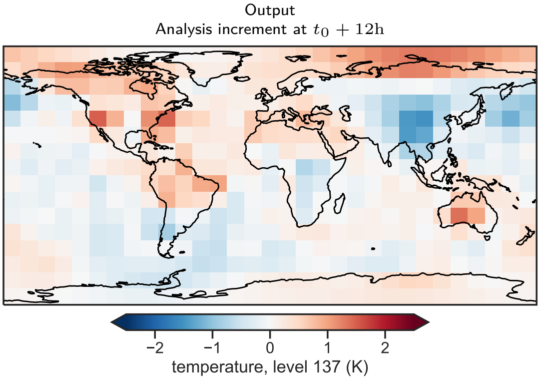
- The FNN library has been interfaced and included in OOPS.

- We want to develop a model error correction for the operational IFS.
- We use a two-step training process:
  - offline learning to *screen potential architectures* and *pre-train the NN*
  - online learning: data assimilation and forecast experiments
- Offline experiments rely on preliminary work by Bonavita & Laloyaux (2020), using the *operational analyses* produced by ECMWF between 2017 and 2021.
- The NN is trained to predict the analysis increments, which are available every 12 hours.
- Training / validation split:
  - training from 2017-01-01 to 2020-10-01 (IFS cycles 43R1 to 47R1);
  - validation from 2020-10-01 to 2021-10-01 (IFS cycles 47R1 to 47R2).

Focus on large-scale model errors

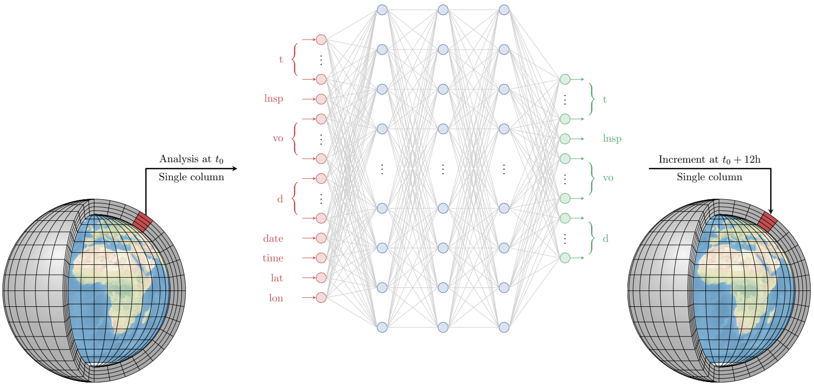
- Focus on *large-scale model errors*: we use the data at a low spectral resolution (T15), interpolated in Gaussian grid with 16×31 nodes.





Neural network architecture

- We compute a correction for 4 variables in the same NN: temperature (t), logarithm of surface pressure (lnsp), vorticity (vo) and divergence (d).
- We keep the same *vertical architecture* as in Bonavita & Laloyaux (2020).



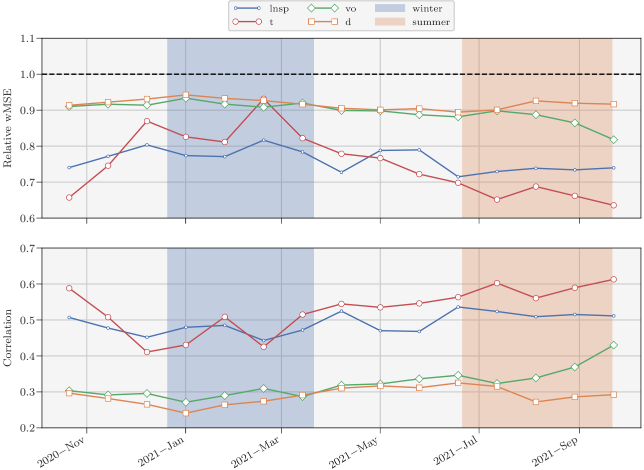
- The NN can be used with any grid.
- The number of parameters is relatively small (approx. 1M) compared to the dimension of the control vector and to the size of the training dataset (approx. 700M).
- Spatial information is partially lost.

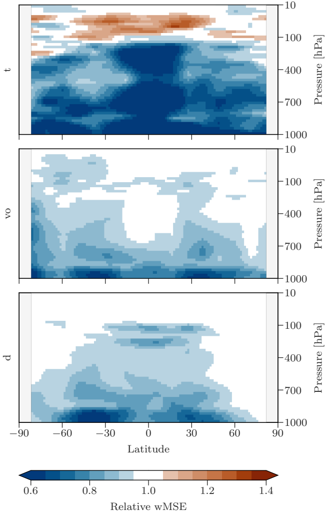
Offline performance of the NN

Test MSE (relative)

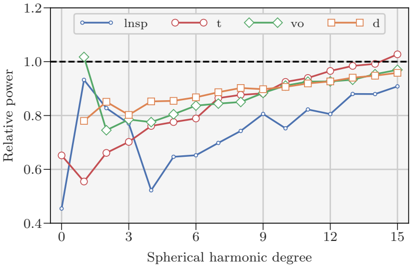
Model	t	lnsp	vo	d
No correction	1.000	1.000	1.000	1.000
Trained NN	0.760	0.759	0.898	0.919

- Overall, the NN predicts approximately *15% of the analysis increments*.
- The increments for *tlnsp* are more predictable than for *vod*.
- The increments are more predictable in summer than in winter.





- The NN is most accurate *close to the surface*.
- The estimations deteriorate between 10 and 100 hPa, where weak constraint 4D-Var is active in the test set.
- The estimations are more accurate *at larger scales*.



Second step: data assimilation experiments

- The trained NN is inserted *into the IFS*, in a standard research configuration:
  - 12h assimilation window;
  - Latest IFS cycle 48R1;
  - Resolution of the nonlinear model: TCo399;
  - Resolution of the inner loops: TL95, TL159, TL255.
- Three-month experiment in *summer 2022* (outside the offline training and test set).
- First test series *without online learning*.

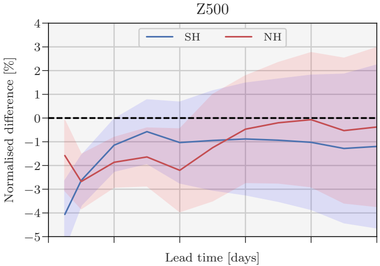
  This is equivalent to using strong-constraint 4D-Var with the corrected model.
- Second test series *with online learning*.

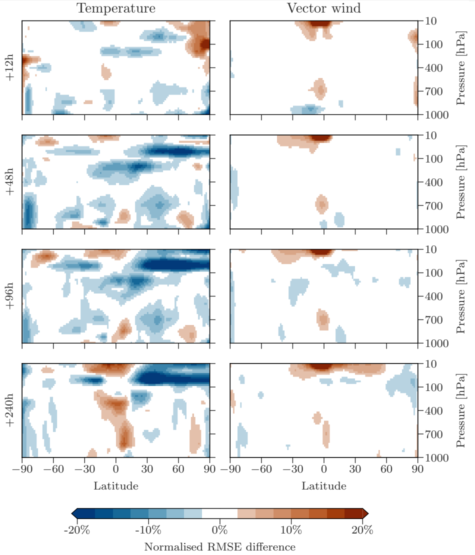
Data assimilation experiments without online training

- Comparison to the *operational analysis*.
- Baseline: standard weak-constraint 4D-Var by Laloyaux et al. (2020).

- Significantly reduced errors above 100 hPa, especially at long lead time.
- Below 100 hPa, the performance in the tropics is degraded.

- For Z500, we see a RMSE reduction of 1 % to 2 %.

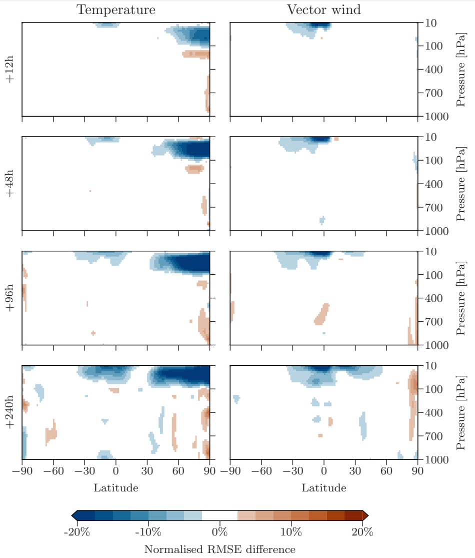




Data assimilation experiments with online training

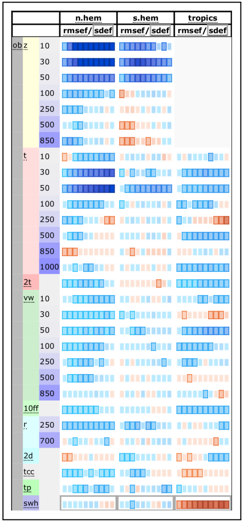
- Comparison to the *operational analysis*.
- Baseline: experiment without online training.

- Significantly reduced the errors in the stratosphere.
- Especially in the northern hemisphere for temperature and in the tropics for vector winds.

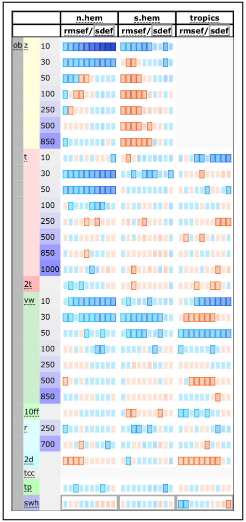


- Comparison to *independent observations*.
- Overall, the impact on forecast RMSE of all variables is positive in the *northern hemisphere* and in the *tropics*.
- Relatively modest impact in the *southern hemisphere* except in the stratosphere.
- On the downside, some score are slightly degraded, e.g. temperature at 850 hPa.

Online - Reference



Online - Offline



Conclusions

- We have developed a *new variant* of weak-constraint 4D-Var to perform an *online, joint estimation* of the system state and NN parameters.
- The new variant is built on top of the existing weak-constraint 4D-Var, in the incremental assimilation framework.
- The new variant is *implemented in OOPS*, using a newly developed NN library in Fortran (FNN).

- We are testing the method with the operational IFS.
- First results are promising.
- Upcoming challenges:
  - training at *higher resolution*;
  - develop a *time-dependent correction* within the window;
  - improve the consistency between offline and online training.

- More details can be found in our preprint:

https://doi.org/10.48550/arXiv.2403.03702