Hybrid models at ECMWF: towards online trained model error corrections

Alban Farchi, Marcin Chrust,
Patrick Laloyaux, Massimo Bonavita, Marc Bocquet (ENPC), Wei Pan

ECMWF

Outline

I. Machine learning for NWP: offline model error correction

II. From offline to online correction

III. Application to the ECMWF forecasting system

IV. Towards a batched online training system

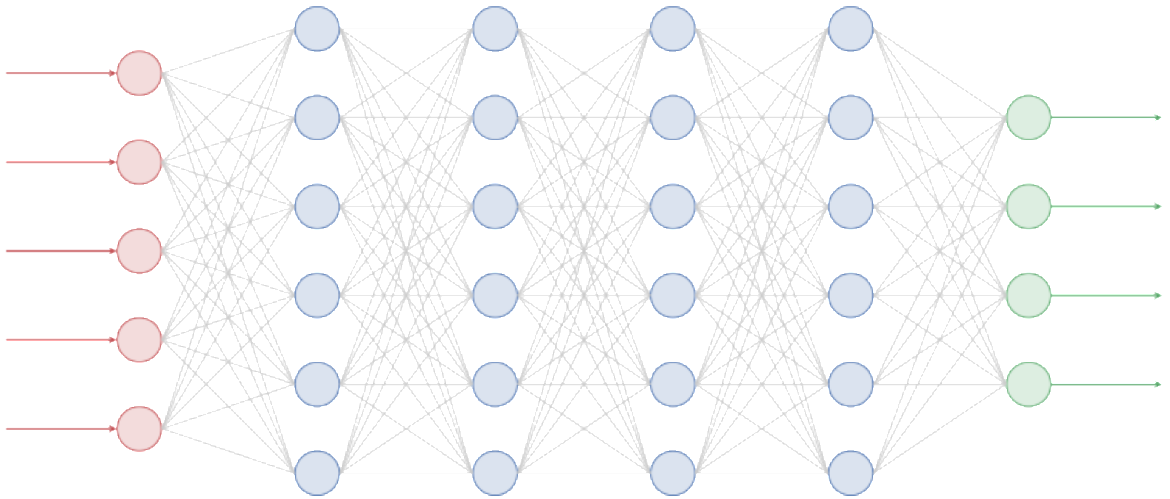
Machine learning for NWP with dense and perfect observations

- A typical (supervised) machine learning problem: given observations y_k of a system, derive a surrogate model of that system

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

- $\mathcal{M}$ depends on a 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 sparse and noisy: we need data assimilation!

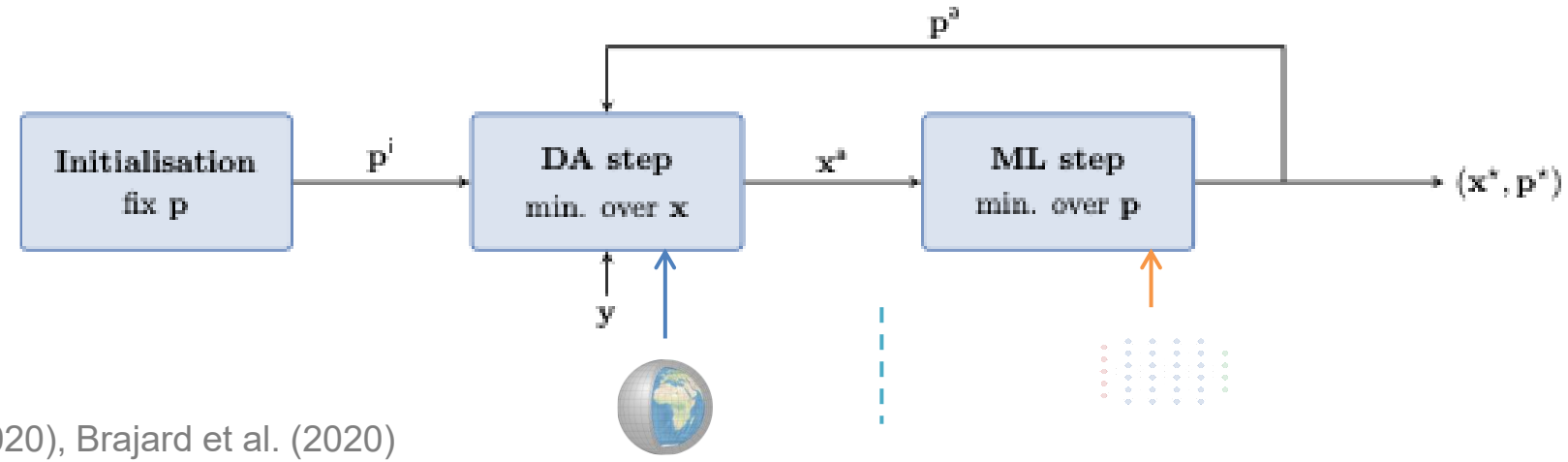


Machine learning for NWP with sparse and noisy observations

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

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

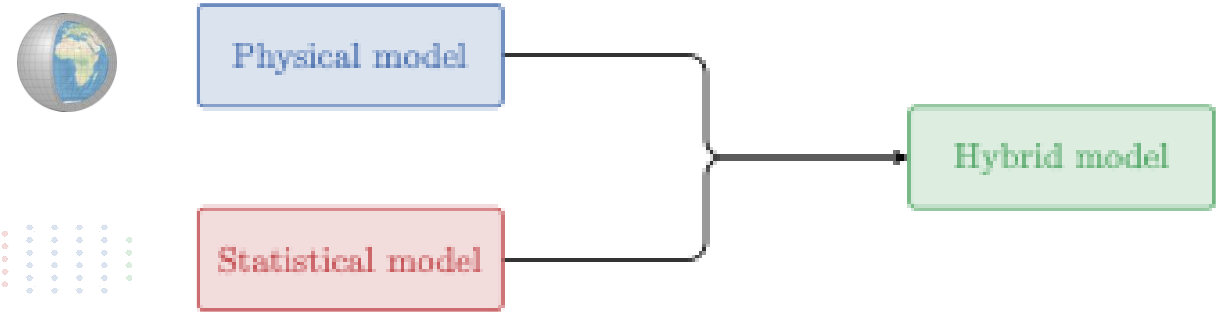
- This resembles a typical weak-constraint 4D-Var cost function!
- DA is used to estimate the state and then ML is used to estimate the model



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

Machine learning for model error correction

- 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 and correct 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.

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

Typical architecture of a physical model

- A typical NWP model rely on a set of ODEs or PDEs which define the tendencies:

$$\frac{\partial \mathrm{x}}{\partial \mathrm{t}} = \phi(\mathrm{x})$$

- A numerical scheme is used to integrate the tendencies from t to $t + \delta t$:

$$x(t + \delta t) = \mathcal{I}\big(x(t)\big)$$

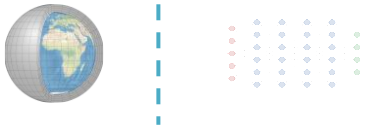
- Several integration steps are composed to build the resolvent from one analysis (or window) to the next:

$$\mathcal{M}: x_k \mapsto x_{k+1} = \mathcal{I} \circ \cdots \circ \mathcal{I}(x_k)$$

Correcting the resolvent or the tendencies?

Resolvent correction

- Physical model and NN are independent
- The NN must predict the analysis increments
- Adding a (potentially large) NN correction to the state can provoke initialisation shocks
- The hybrid model cannot be used for short-term predictions with ad-hoc approximations (e.g. linear growth of errors in time



Tendency correction

- Physical model and NN are entangled
- Need the adjoint of the physical model to train the NN! Requires online training.
- Can be used as is in DA experiments

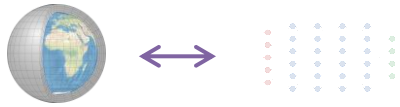


Illustration with the two-scale Lorenz system: setup

- True model: 2-scale Lorenz (2005-III) system with 36 slow variables and 360 fast variables
- Physical model (to correct): 1-scale Lorenz (1996) system with 36 variables

- 3 sources of model error
  - The fast variables are not represented
  - The integration step is 0.05 instead of 0.005
  - (the forcing is 8 instead of 10)

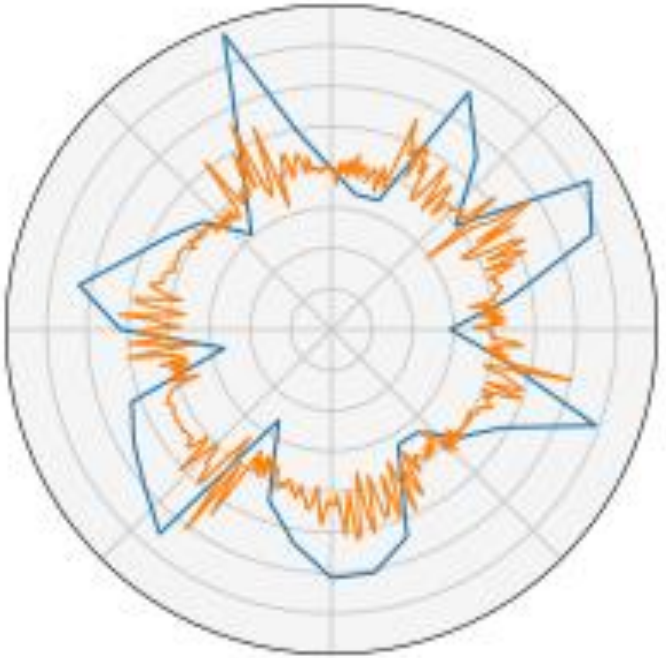


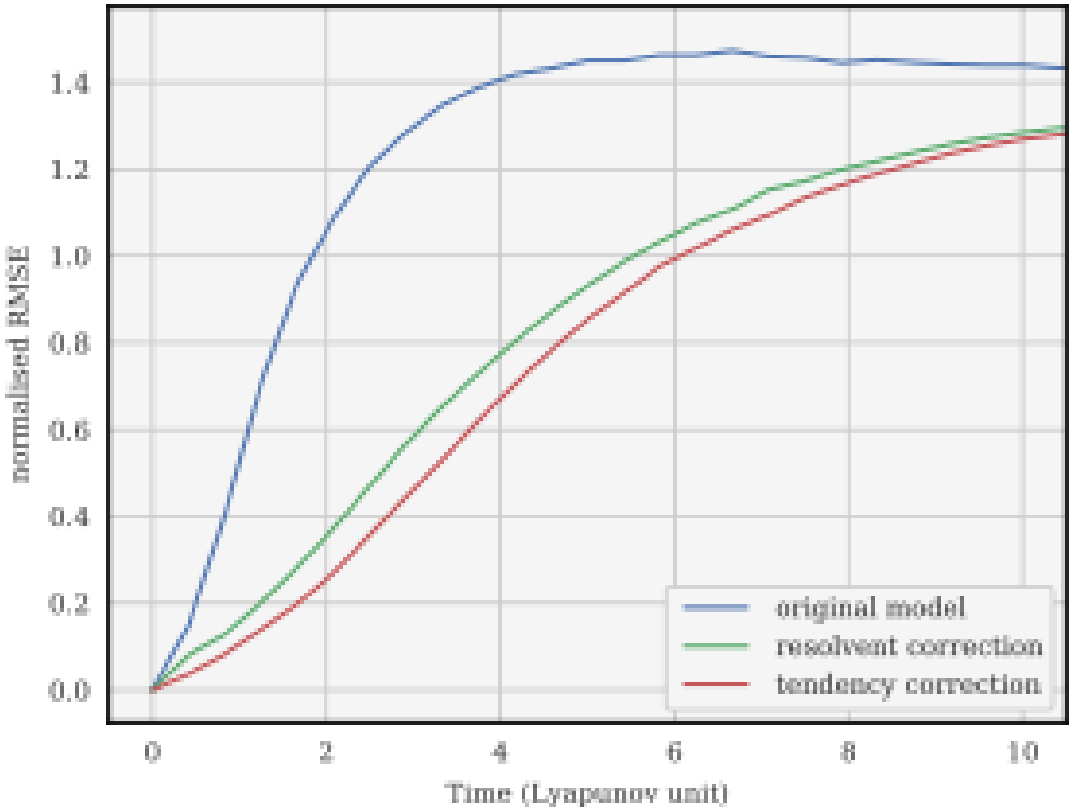
Illustration with the two-scale Lorenz system: results

- Noisy observations are assimilated using strong-constraint 4D-Var
- Simple CNNs are trained using the 4D-Var analysis dataset to correct model errors

Data assimilation experiments

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

Forecast experiments



- The TC is more accurate than the RC, even with smaller NNs and less training data
- The TC benefits from the interaction with the physical model
- The RC is highly penalised (in DA experiments) by the assumption of linear growth of errors

Outline

I. Machine learning for NWP: offline model error correction

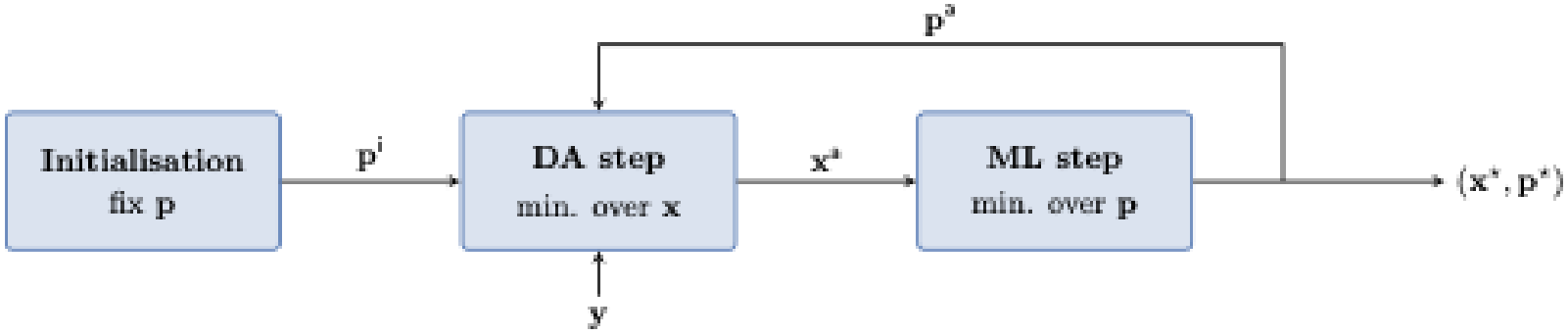
II. From offline to online correction

III. Application to the ECMWF forecasting system

IV. Towards a batched online training system

Merging DA and ML for online model error correction

- So far, the model error correction has been learnt offline: the NN is trained only once the entire analysis dataset is available.



- We now investigate the possibility to make online learning, i.e. improving the NN as new observations become available.
- In practice, we propose to merge the DA and ML steps: we want to use the formalism of DA to estimate both the state and the NN parameters at the same time.

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

- Taking inspiration from weak-constraint 4D-Var, we propose to use the following DA cost function:

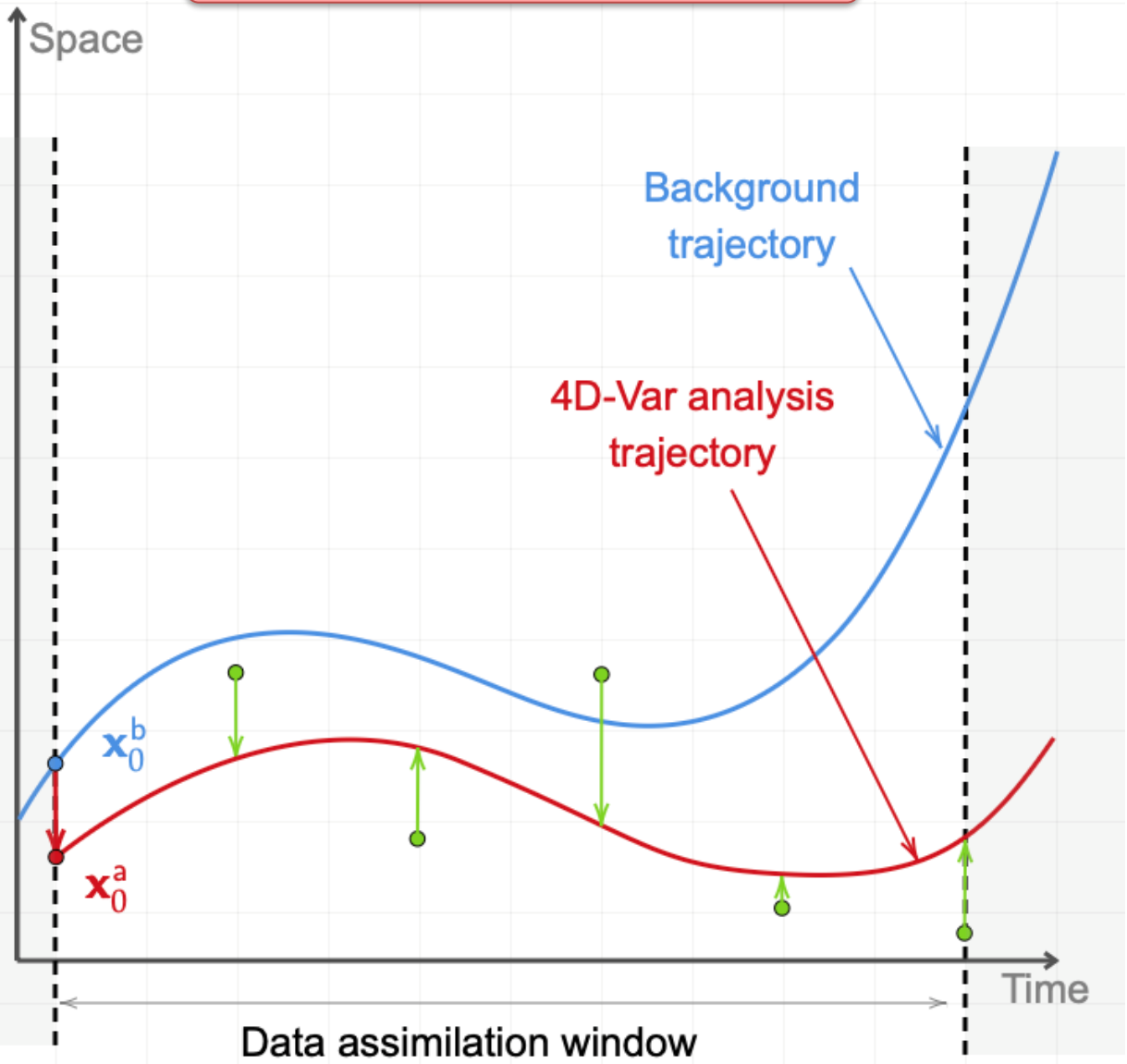
$$\mathcal{J}(\mathbf{p}, \mathbf{x}_0) = \frac{1}{2}\left\|\mathbf{x}_0 - \mathbf{x}_0^b\right\|^2_{\mathbf{B}^{-1}} + \frac{1}{2}\left\|\mathbf{p} - \mathbf{p}^b\right\|^2_{\mathbf{P}^{-1}} + \frac{1}{2}\sum_{k=0}^{L}\left\|\mathbf{y}_k - \mathcal{H}_k \circ \mathcal{M}_{k:0}(\mathbf{p}, \mathbf{x}_0)\right\|^2_{\mathbf{R}_k^{-1}}.$$

- The parameters p (e.g., NN weights and biases) are assumed constant over the DA window.
- Information is flowing from one window to the next using the priors x^b and p^b.
- 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].

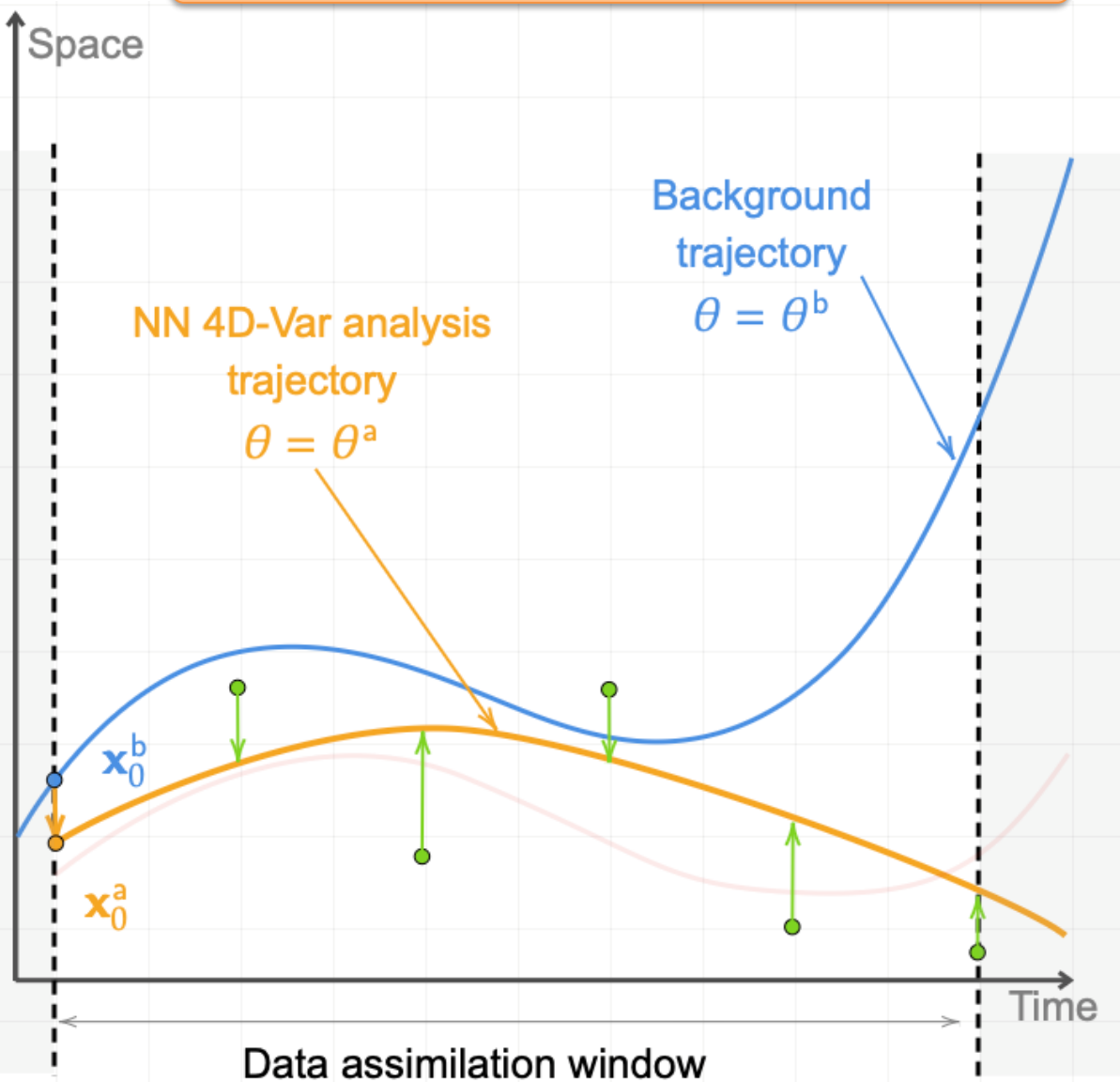
[3] Bocquet et al. (2020)

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

4D-Var: x estimation



NN 4D-Var: x and θ (or p) estimation

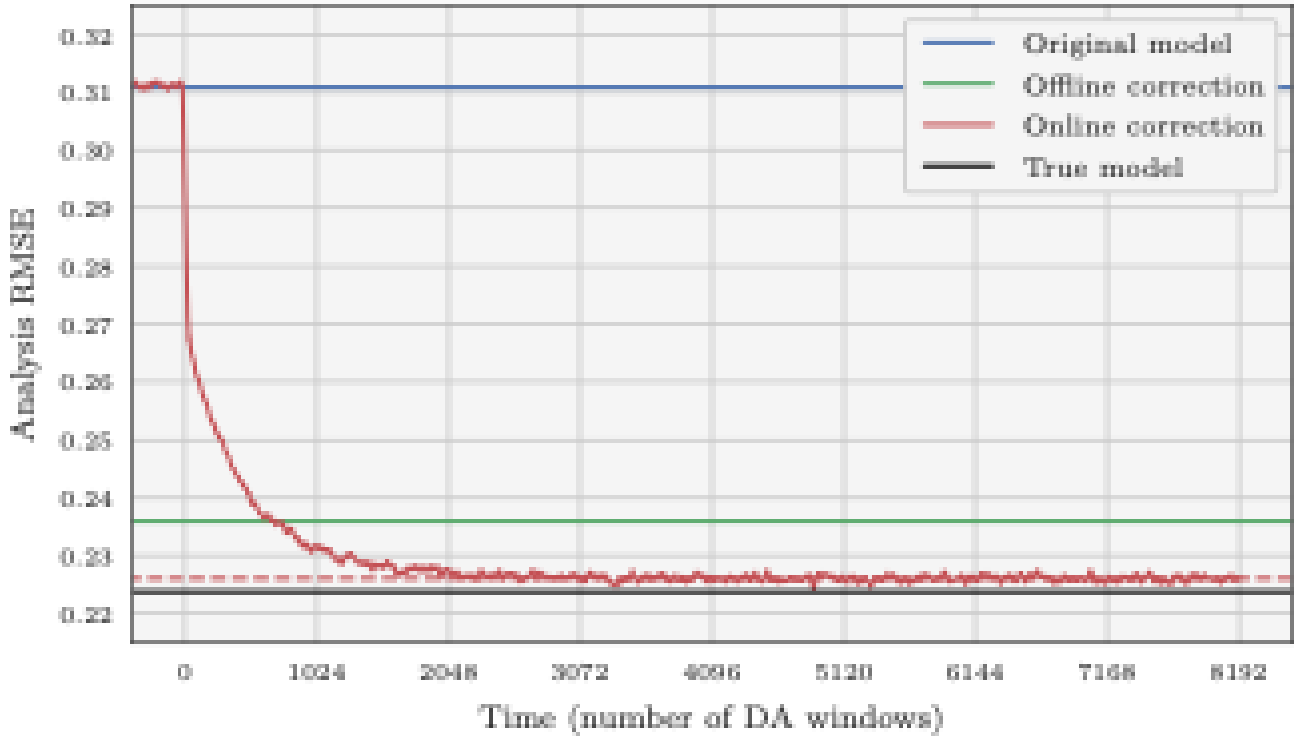


Motivations for online learning

- Offline Training
  - Uses historical data
  - Static correction with fixed parameters
  - Can become outdated as model and model error evolve (numerical model updates, slowly evolving model climate)
  - Requires periodic retraining
- Why train model error online?
  - Continuous adaptation to changing model error
  - Model error diagnosed using latest observations
  - Model error correction constrained by and consistent with TLM dynamics
  - Mitigated model error in data assimilation

Illustration with the two-scale Lorenz system

- We use the tendency correction approach, with the same simple CNN as before.



- 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!

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 assumptions, e.g. the model error w is constant over the DA window:

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

- The DA cost function can hence be written

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

- 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).

A simplified NN 4D-Var built on top of WC 4D-Var

- In order to merge the two approaches, we consider the case where the constant model error w is estimated using a neural network F:

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

- This means that the model evolution can be written

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

- 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 $\delta \mathbf{x}_0$

$\delta \mathbf{w} \leftarrow \mathbf{F}^{\mathbf{p}} \delta \mathbf{p} + \mathbf{F}^{\mathbf{x}} \delta \mathbf{x}_0$ ▷ 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 $\mathcal{M}_{k:k-1}$

  $\mathbf{z}_k \leftarrow \mathbf{R}_k^{-1} \left(\mathbf{H}_k \delta \mathbf{x}_k - \mathbf{d}_k \right)$

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**

  $\delta \tilde{\mathbf{x}}_k \leftarrow \mathbf{H}_k^{\mathrm{T}} \mathbf{z}_k + \delta \tilde{\mathbf{x}}_k$

  $\delta \tilde{\mathbf{w}}_{k-1} \leftarrow \delta \tilde{\mathbf{x}}_k + \delta \tilde{\mathbf{w}}_k$

  $\delta \tilde{\mathbf{x}}_{k-1} \leftarrow \mathbf{M}_{k:k-1}^{\mathrm{T}} \delta \tilde{\mathbf{x}}_k$ ▷ AD of the dynamical model $\mathcal{M}_{k:k-1}$

end for

$\delta \tilde{\mathbf{x}}_0 \leftarrow \mathbf{H}_0^{\mathrm{T}} \mathbf{z}_0 + \delta \tilde{\mathbf{x}}_0$

$\delta \tilde{\mathbf{x}}_0 \leftarrow \left[\mathbf{F}^{\mathbf{x}} \right]^{\mathrm{T}} \delta \tilde{\mathbf{x}}_0$ ▷ AD of the NN $\mathcal{F}$

$\delta \tilde{\mathbf{p}} \leftarrow \left[\mathbf{F}^{\mathbf{p}} \right]^{\mathrm{T}} \delta \tilde{\mathbf{w}}_0$ ▷ AD of the NN $\mathcal{F}$

$\delta \tilde{\mathbf{x}}_0 \leftarrow \mathbf{B}^{-1} \left(\mathbf{x}_0^{\mathrm{i}} - \mathbf{x}_0^{\mathrm{b}} + \delta \mathbf{x}_0 \right) + \delta \tilde{\mathbf{x}}_0$

$\delta \tilde{\mathbf{p}} \leftarrow \mathbf{P}^{-1} \left(\mathbf{p}^{\mathrm{i}} - \mathbf{p}^{\mathrm{b}} + \delta \mathbf{p} \right) + \delta \tilde{\mathbf{p}}$

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

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 F of the NN to compute the nonlinear trajectory at the start of each outer iteration;
  - the tangent linear (TL) operators F^x and F^p of the NN;
  - the adjoint (AD) operators $[F^x]^T$ and $[F^p]^T$ 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[4].
- The FNN library has been interfaced and included in OOPS.

[4] https://github.com/cerea-daml/fnn

Illustration with a quasi-geostrophic model: the model

- Before using it in operational data assimilation, we would like to illustrate the method with a simplified model.
- To do so, we use the QG model implemented in OOPS. This is a two-layer, two-dimensional quasi geostrophic model.
- The control vector contains all values of the stream function ψ for both levels for a total of 1600 variables.
- Model error is introduced by using a perturbed setup, in which layer depths and the integration time steps have been modified.

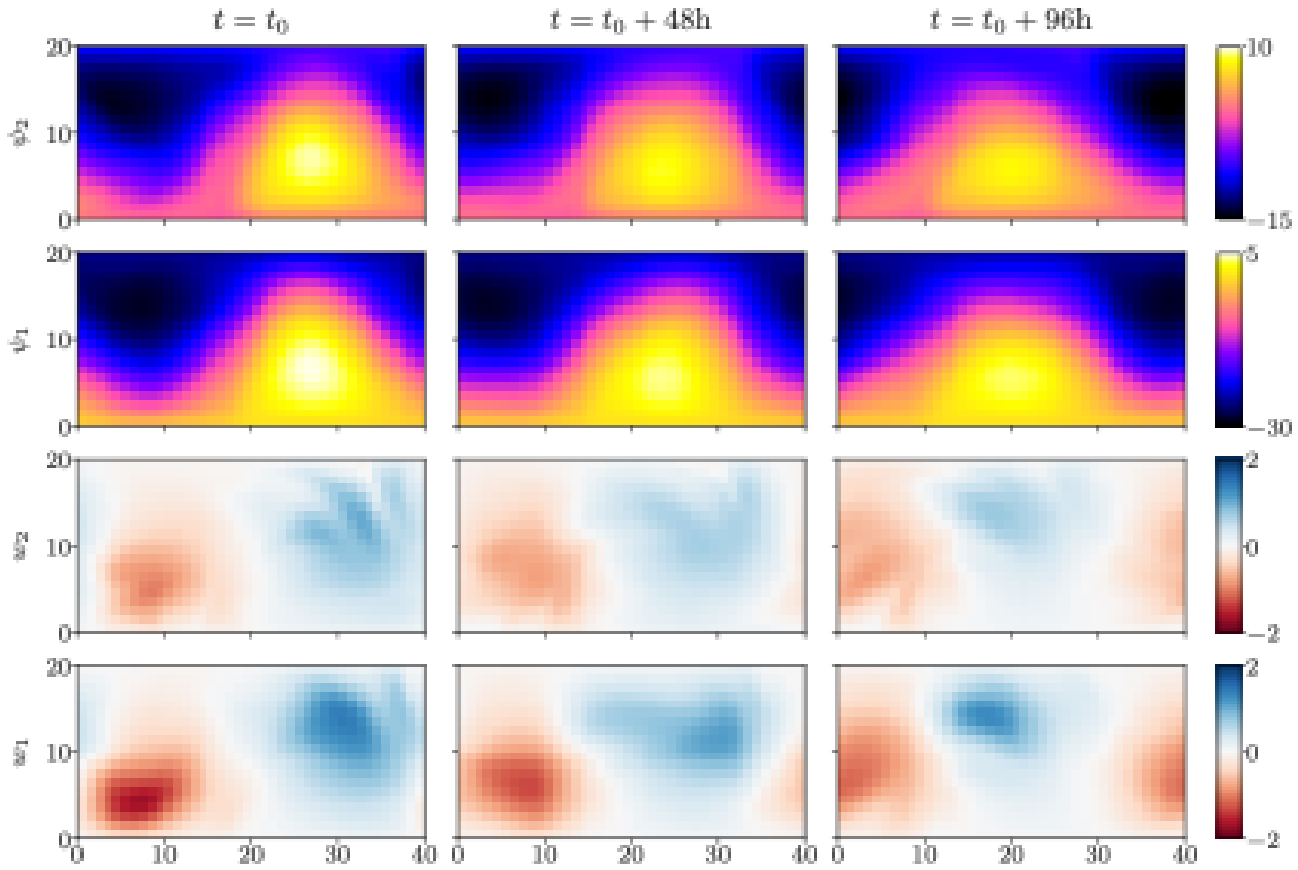


Illustration with a quasi-geostrophic model: NN architecture

- By construction, NN 4D-Var is very similar to parameter estimation, which is challenging when the num. of parameters is high.
- For this reason, it is important to use smart NN architectures to be parameter efficient.
- Taking inspiration from Bonavita & Laloyaux (2020) we use a vertical/column architecture, with only 386 parameters.

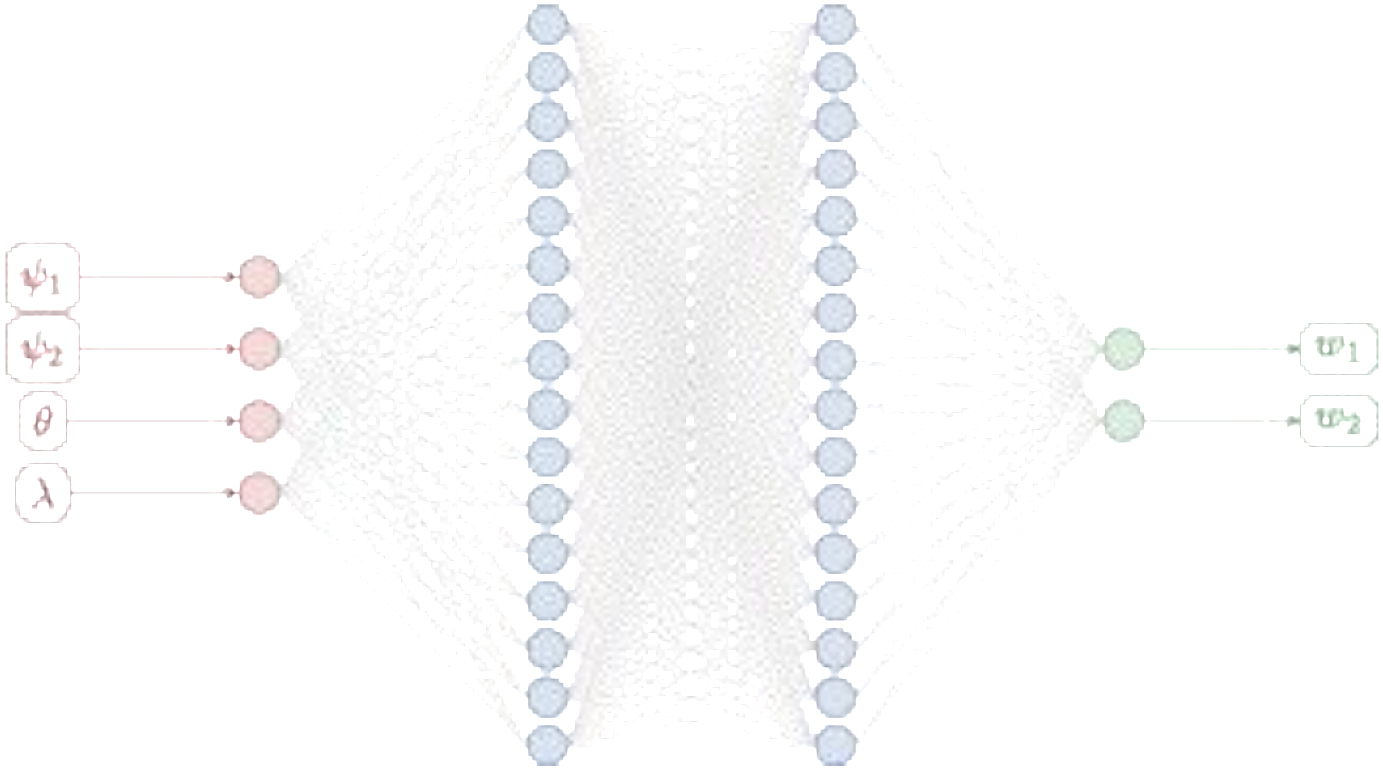
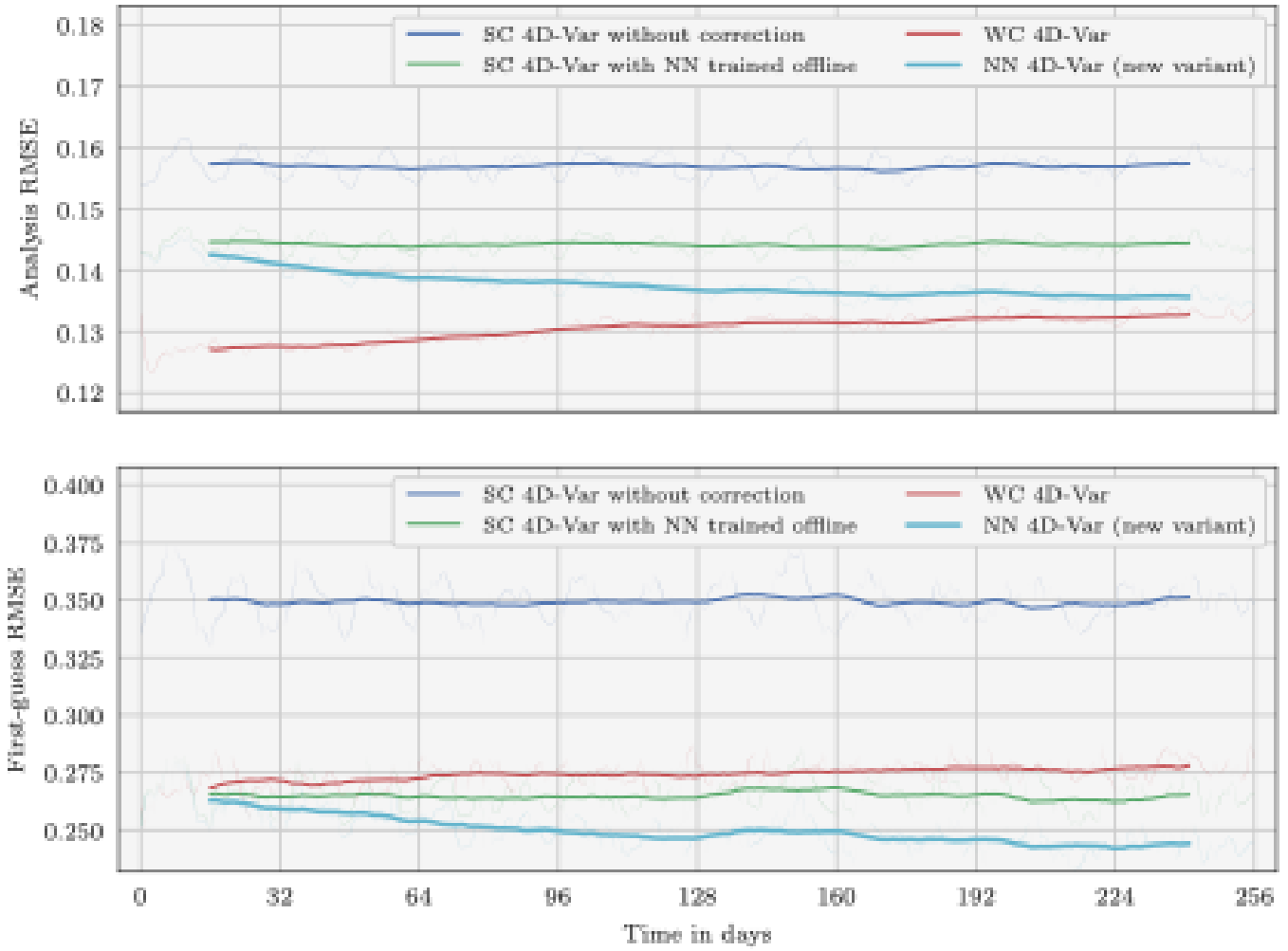


Illustration with a quasi-geostrophic model: results

- Hybrid training approach
  1. Train the NN offline as a resolvent correction
  2. Rescale the correction from 1 DAW to 1 time step
  3. Fine-tune the NN online as a tendency correction

- As new observations become available, online learning steadily improves the model, resulting in more accurate first-guess and analysis.



Outline

I. Machine learning for NWP: offline model error correction

II. From offline to online correction

III. Application to the ECMWF forecasting system

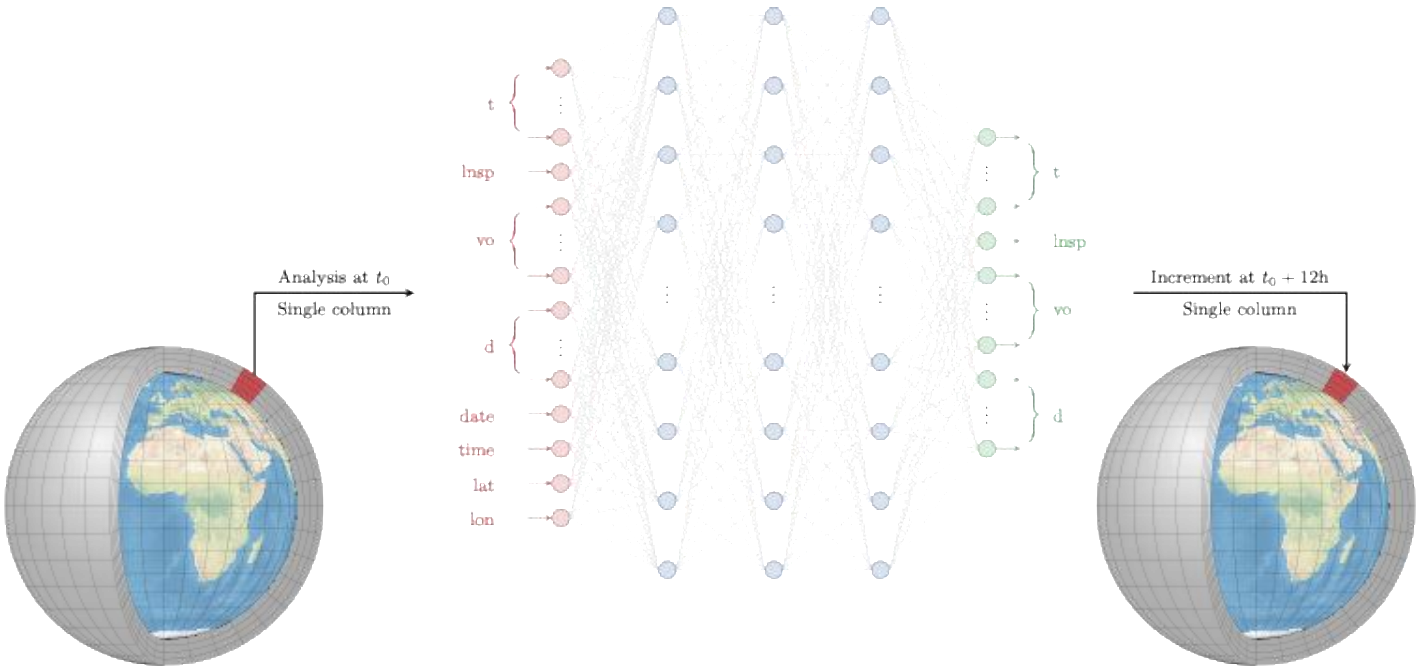
IV. Towards a batched online training system

A NN correction for the IFS

- We extend the preliminary work of Bonavita and Laloyaux (2020)
- We compute a correction for 4 atmospheric variables in the same NN: temperature (t), logarithm of surface pressure (lnsp), vorticity (vo), and divergence (d)
- Hybrid training approach
  1. Train the NN offline as a resolvent correction
  2. Rescale the correction from 1 DAW to 1 time step
  3. Fine-tune the NN online as a tendency correction

The vertical / column architecture

- We use a vertical architecture, where the NN processes independently each atmospheric column
- We include all 137 model levels in both input and output, for a total of 412 channels



- Once trained, the NN can be applied on any grid
- The num. of parameters is relatively small compared to the size of the control vector (<2%) and to the size of the training dataset (<1%)
- Horizontal information is partially lost

Offline training dataset

- We use data from the operational archive between 2021 and 2023:
  - Training set: 2021 and 2022 (cycles 47R1 to 47R3)
  - Validation set: 2023 (cycles 47R3 to 48R1)
- 2024 is set aside for online evaluation
- Pre-2021 data is discarded because the implementation of WC 4D-Var (in June 2020 with cycle 47R1) significantly modified the increments

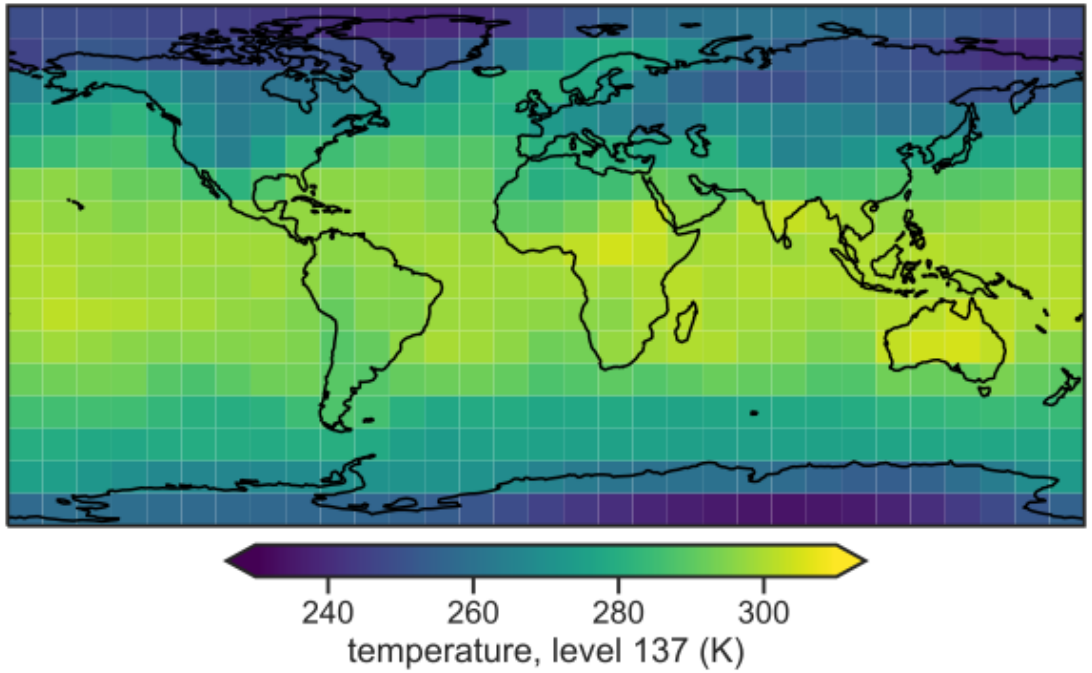
- The NN is trained to predict the sum of the analysis increments and the model error diagnosed by WC 4D-Var (1 sample every 12h)

$$\mathcal{F}_\theta : x_k^a \mapsto x_{k+1}^a - x_{k+1}^f + 12 * w_k^a$$

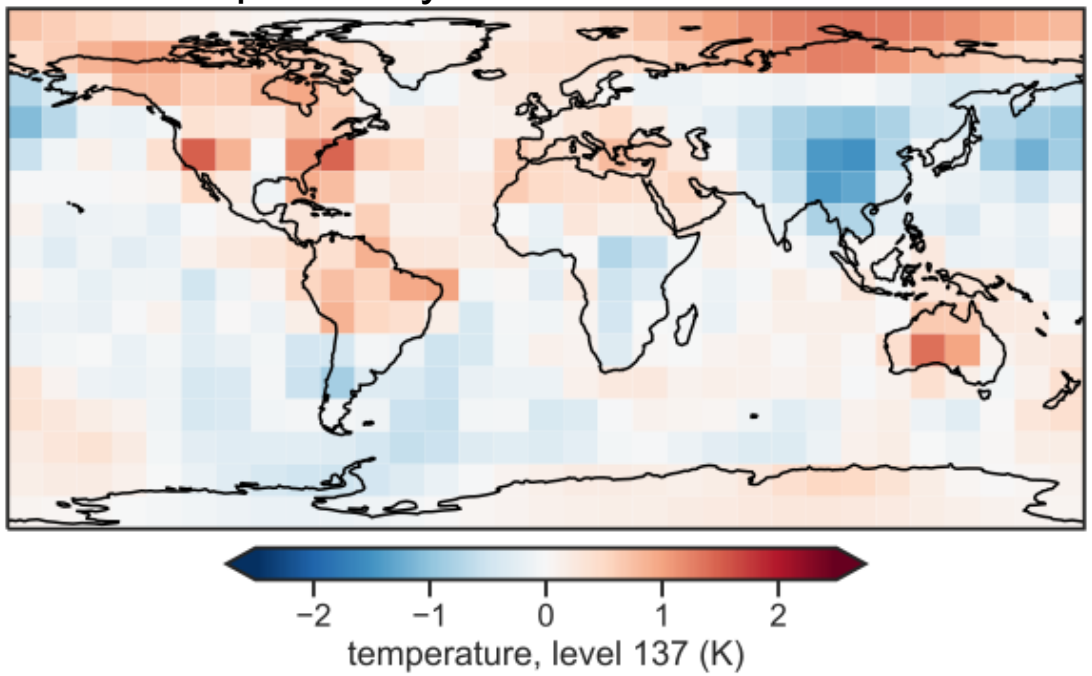
Choice of the spectral truncation

- To focus on large-scale model error, we truncate the data at T15 and interpolated in a 16x32 Gaussian grid

NN input: analysis at time t



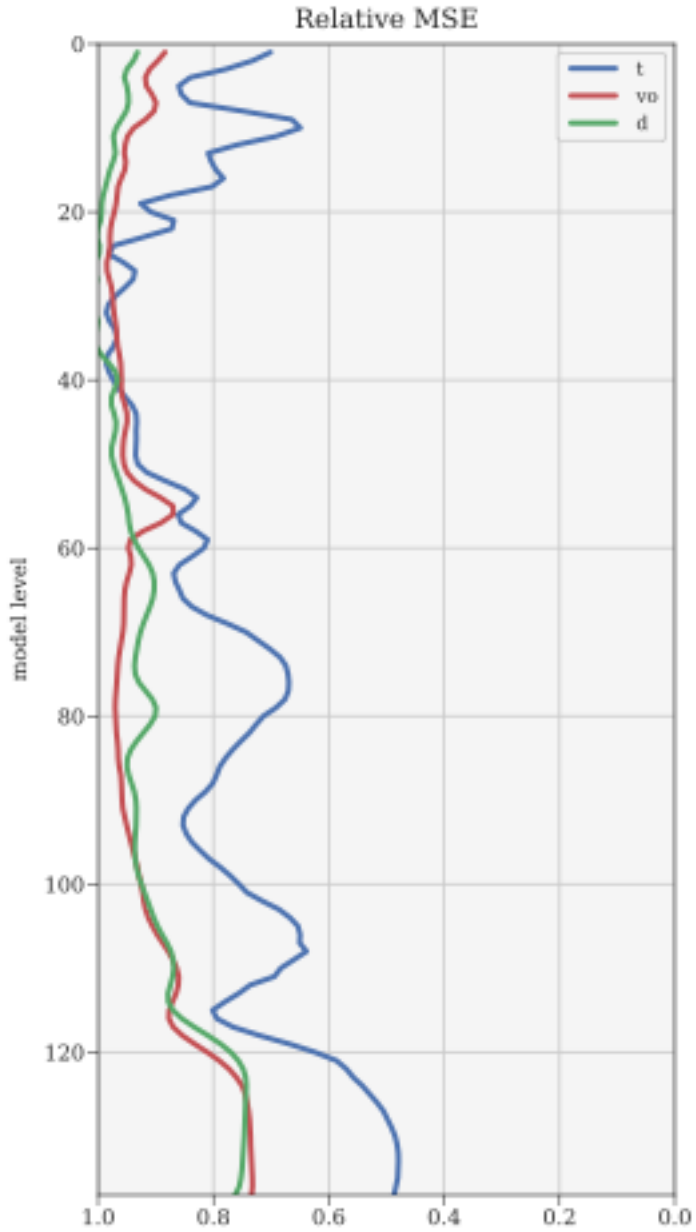
NN output: analysis increment at time t+12h



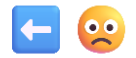
Offline validation at T15

Relative MSE on the validation set

Name	Lnsp	T	Vo	D
No correction	1.000	1.000	1.000	1.000
Mean increment	1.000	0.969	1.000	1.000
Climatological increment	0.769	0.779	0.982	0.959
Trained NN	**0.768**	**0.726**	**0.921**	**0.926**

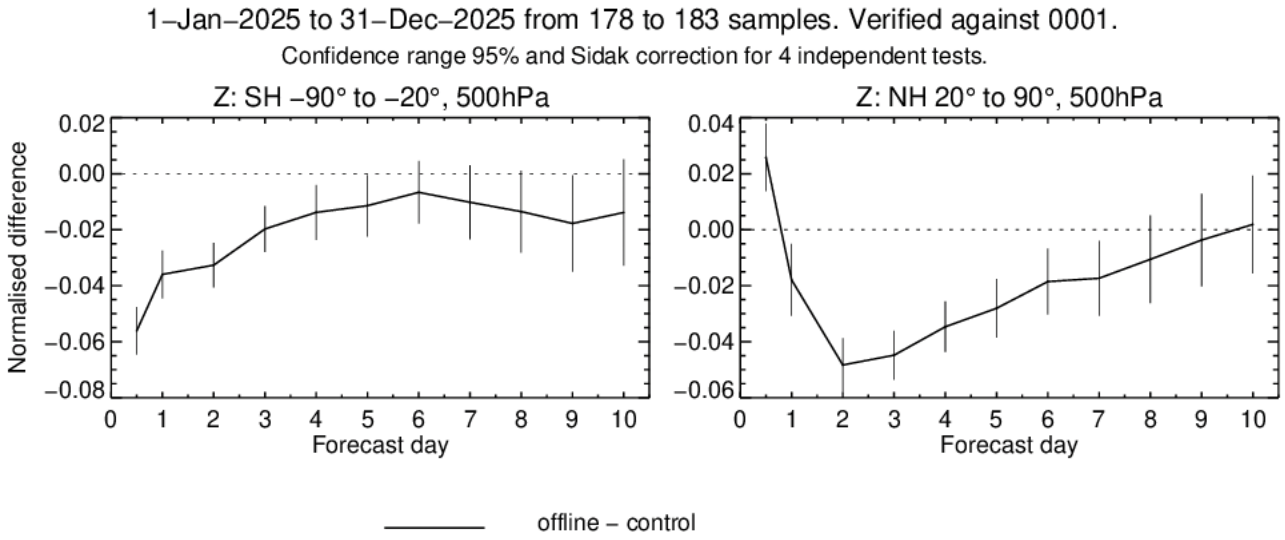


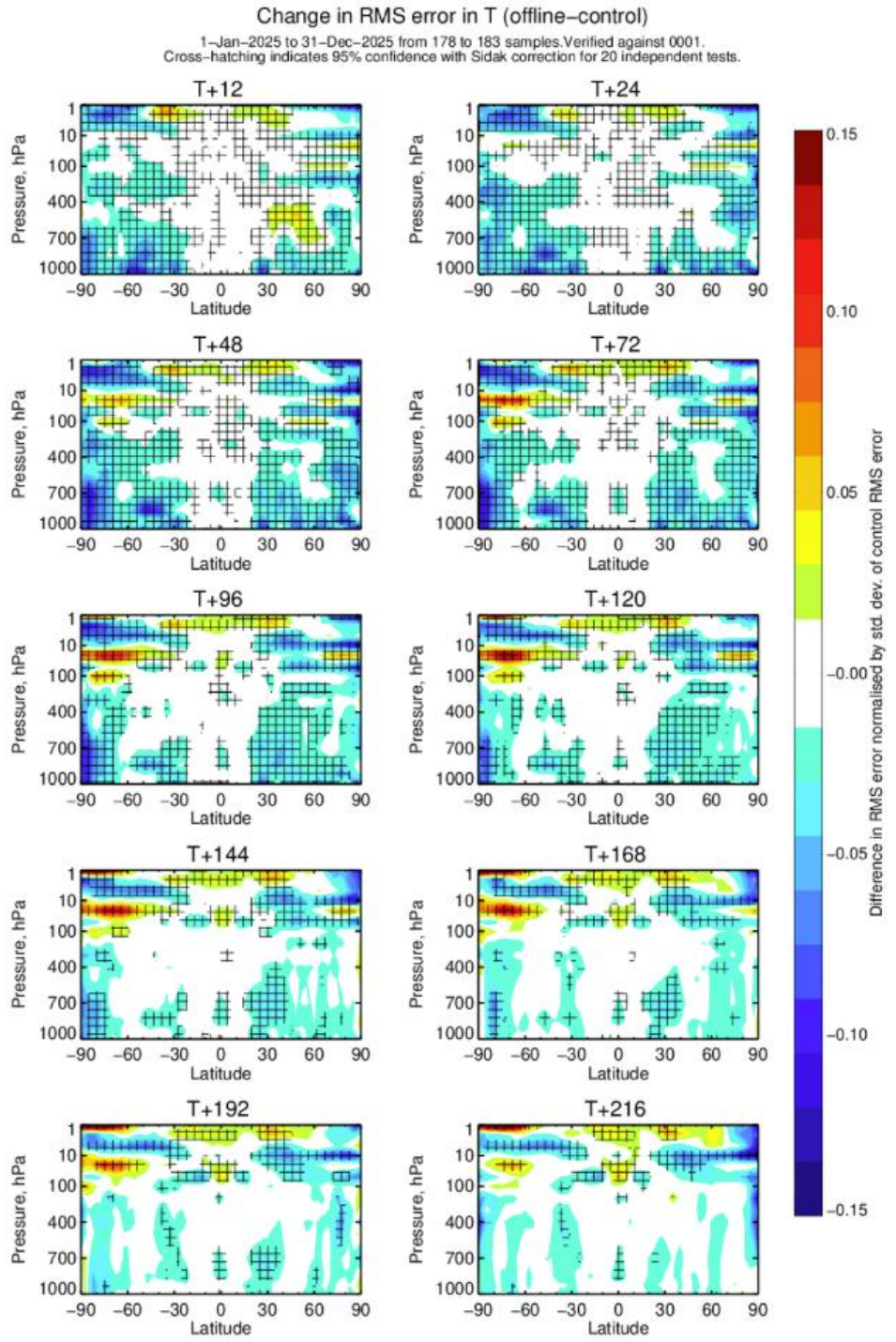
- Overall, the NN can predict 10 to 40% of the increments
- The increments for t, lnsp, and d are more predictable than for vo
- The predictions are in general most accurate close to the surface



Accuracy of the offline hybrid model

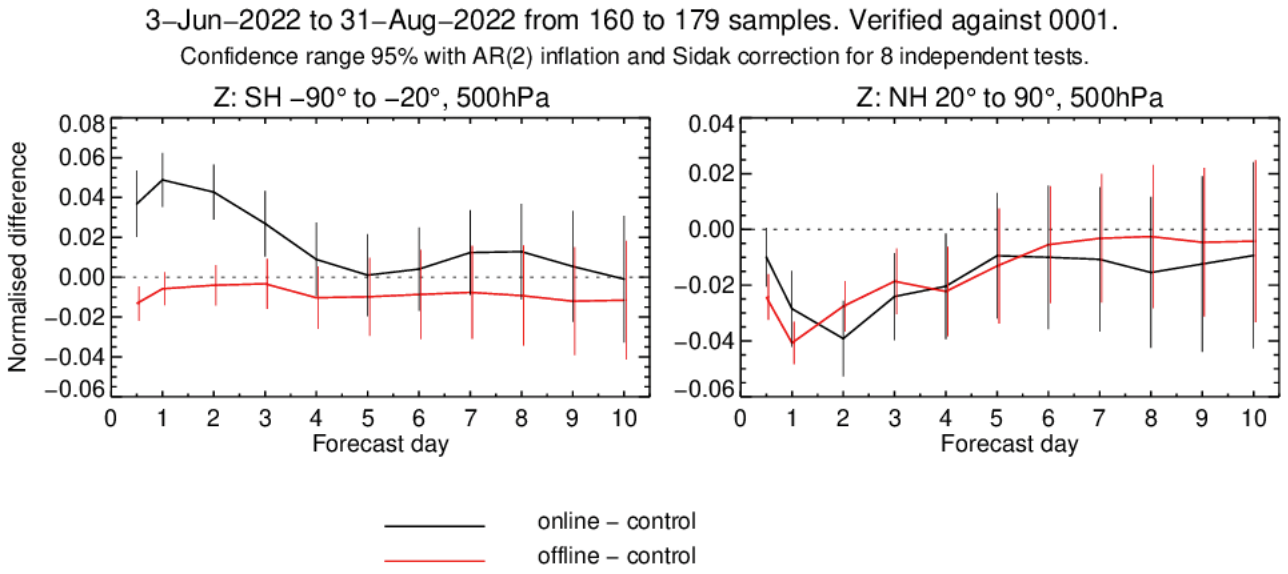
- Forecast-only experiments in 2025 (unseen data) at TCo399
- Correction masked in the tropics



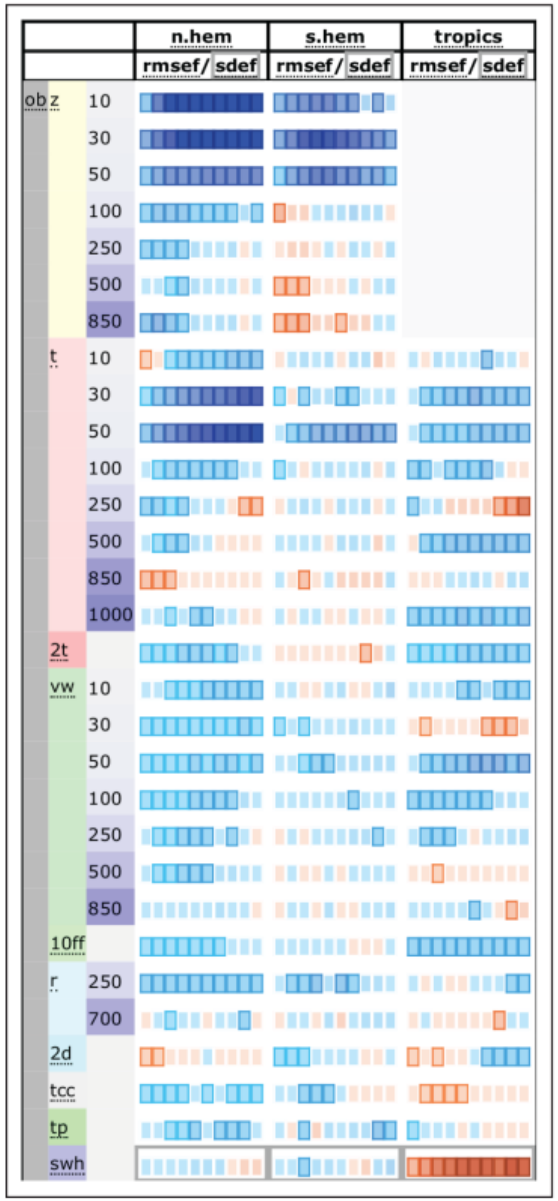


Training online with the standard DA setup (12h window)

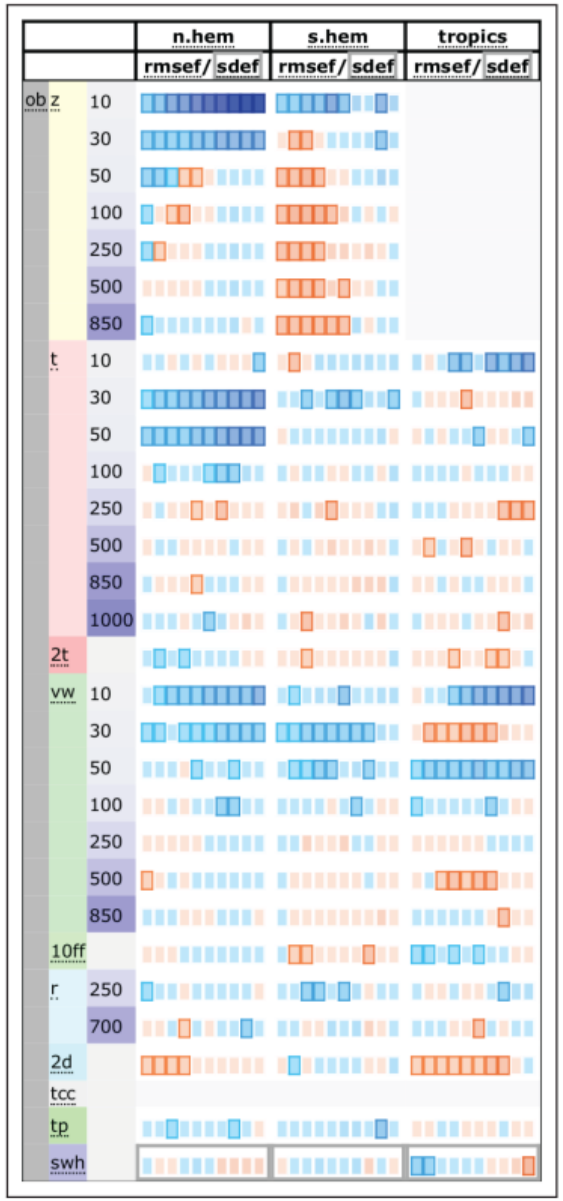
- Online training (fine-tuning) in summer 2022 at TCo399 (from Farchi et al., 2025)
- Largest impact visible in the stratosphere



Online - Reference



Online - Offline



Outline

I. Machine learning for NWP: offline model error correction

II. From offline to online correction

III. Application to the ECMWF forecasting system

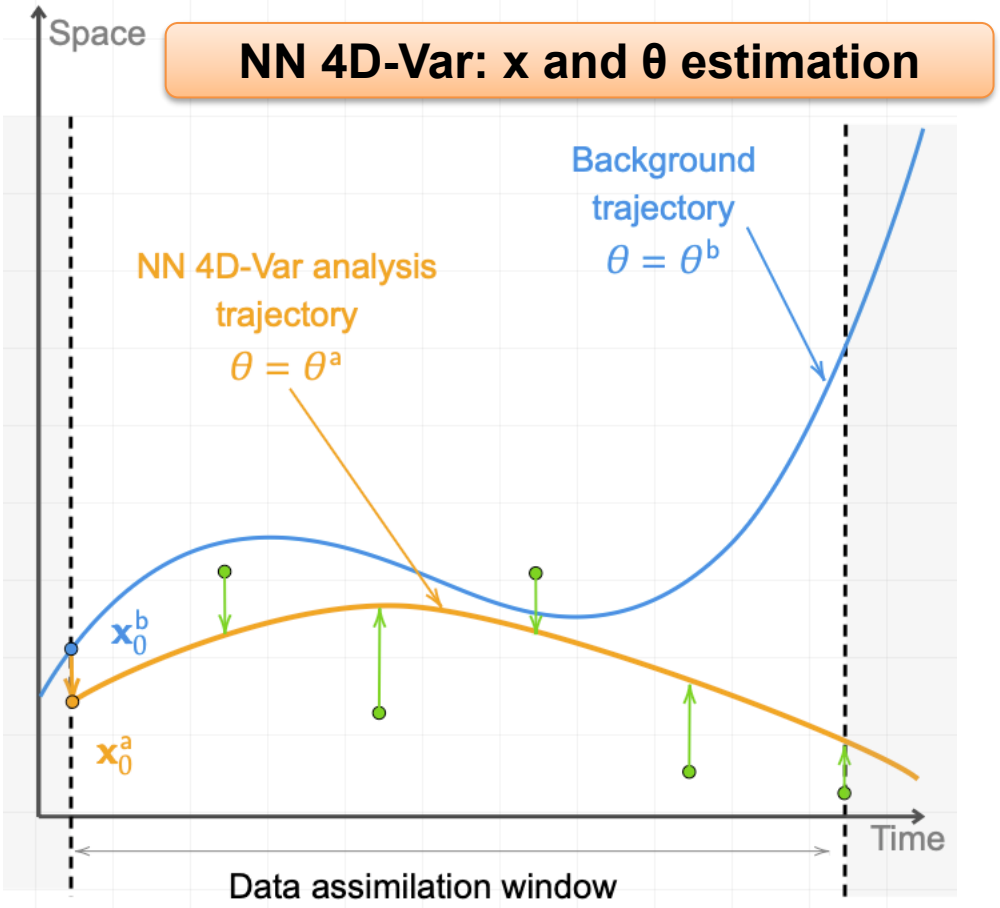
IV. Towards a batched online training system

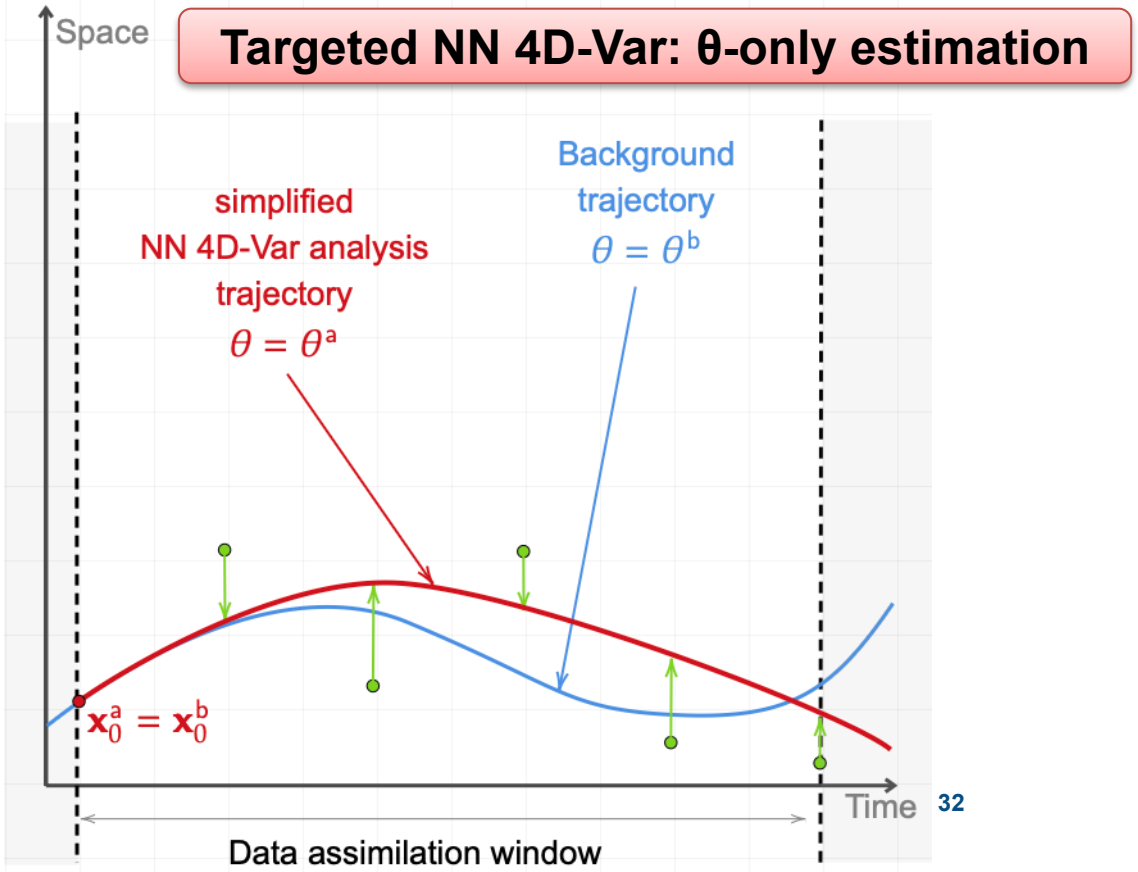
Simplifying NN 4D-Var to focus on the NN parameters

- To further increase the impact of online training, we need 4D-Var to focus on the NN parameters:

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

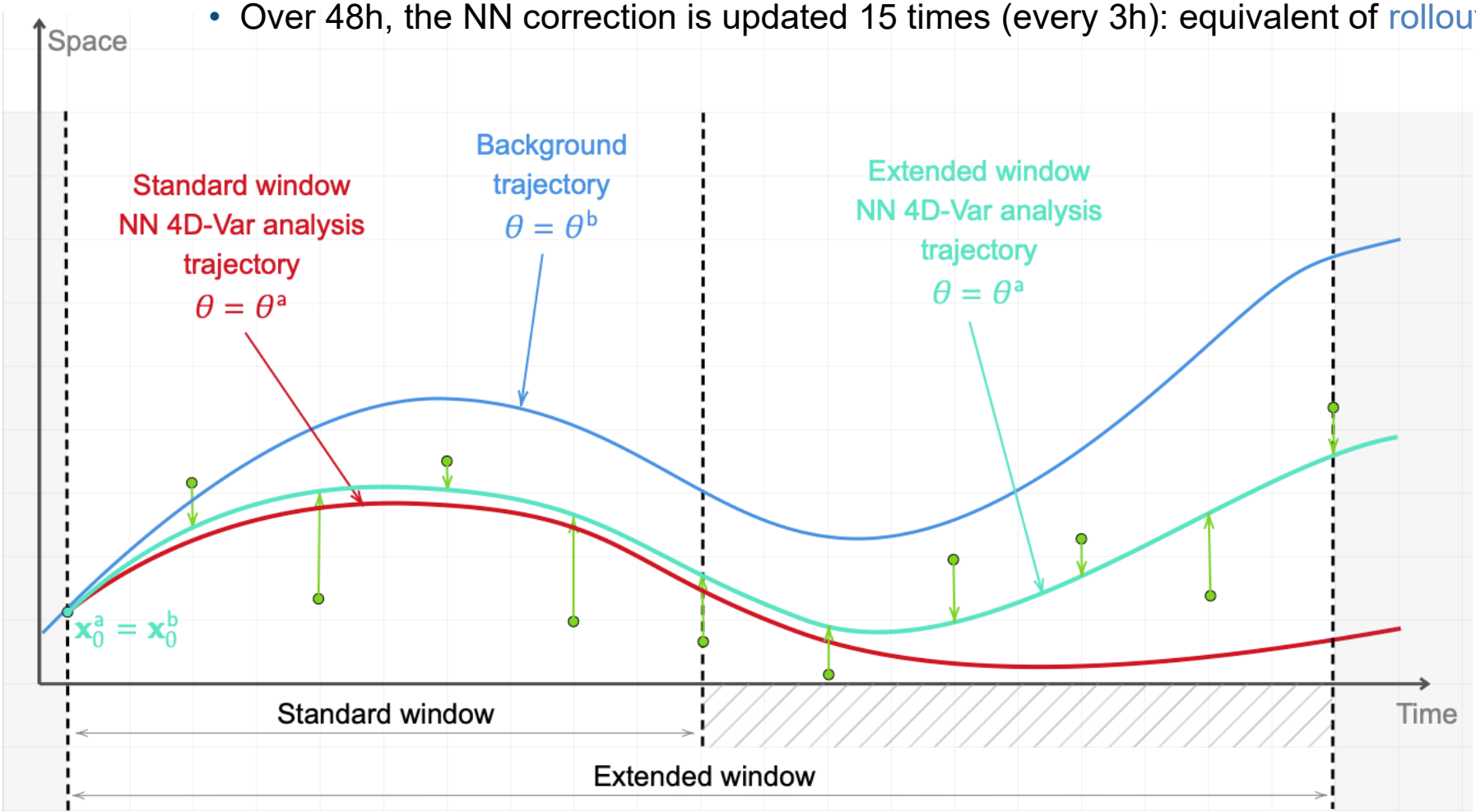
- Where $\mathbf{x}_0^{\mathrm{pa}}$ is a precomputed analysis (e.g. the operational analysis)





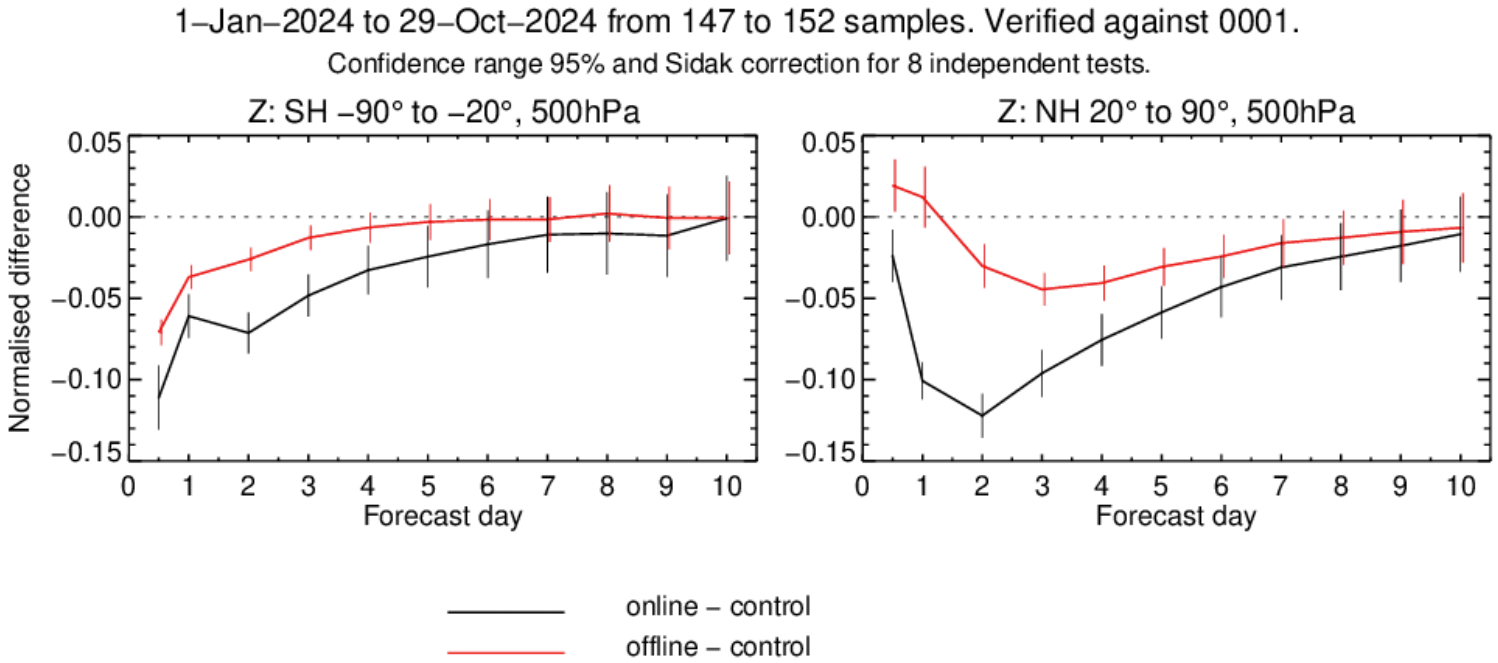
Extending the DA window from 12h to 48h

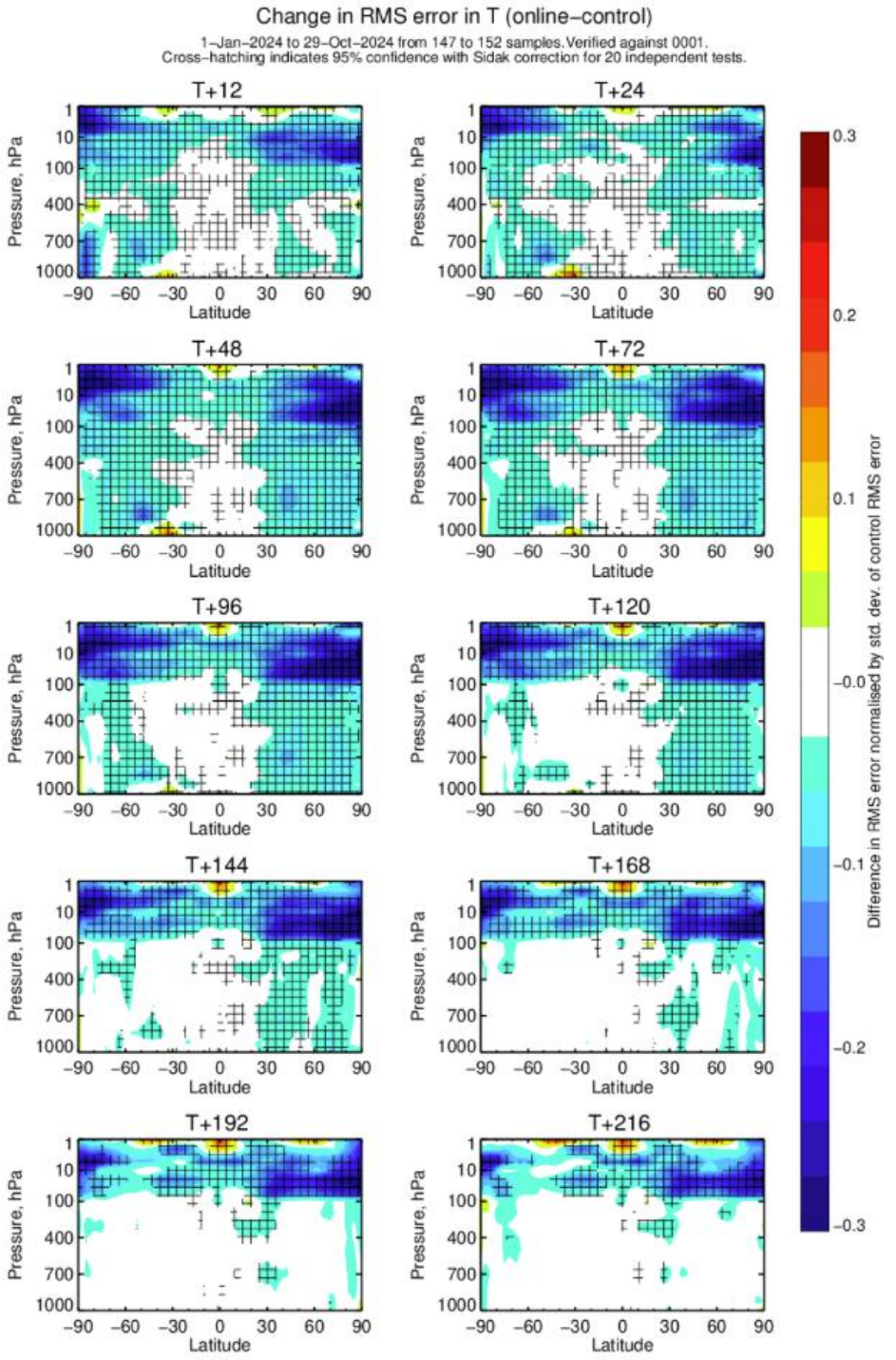
- Furthermore, we decided to extend the length of the DAW from 12h to 48h:
  - More observations are available to constrain the parameters of the NN
  - It helps capture model errors that develop over longer time-scales
- Over 48h, the NN correction is updated 15 times (every 3h): equivalent of rollout for MLWP models



Training online with 48h windows

- Online training (fine-tuning) in 2024 at TCo399
- Much stronger impact than with 12h windows
- Caveats:
  - 48h windows means 36h of future observations
  - Does not correspond to 1 NN, but a collection of NNs

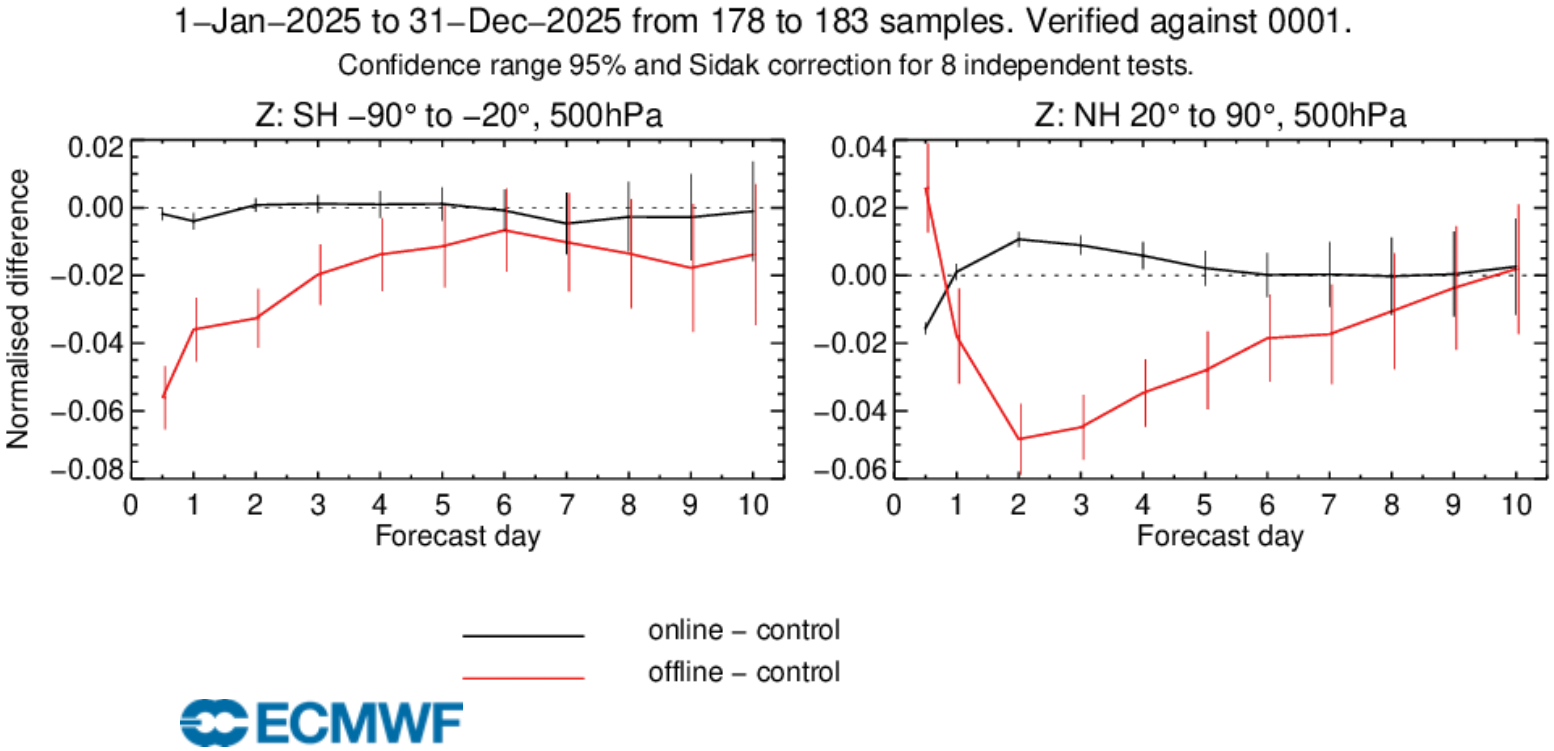


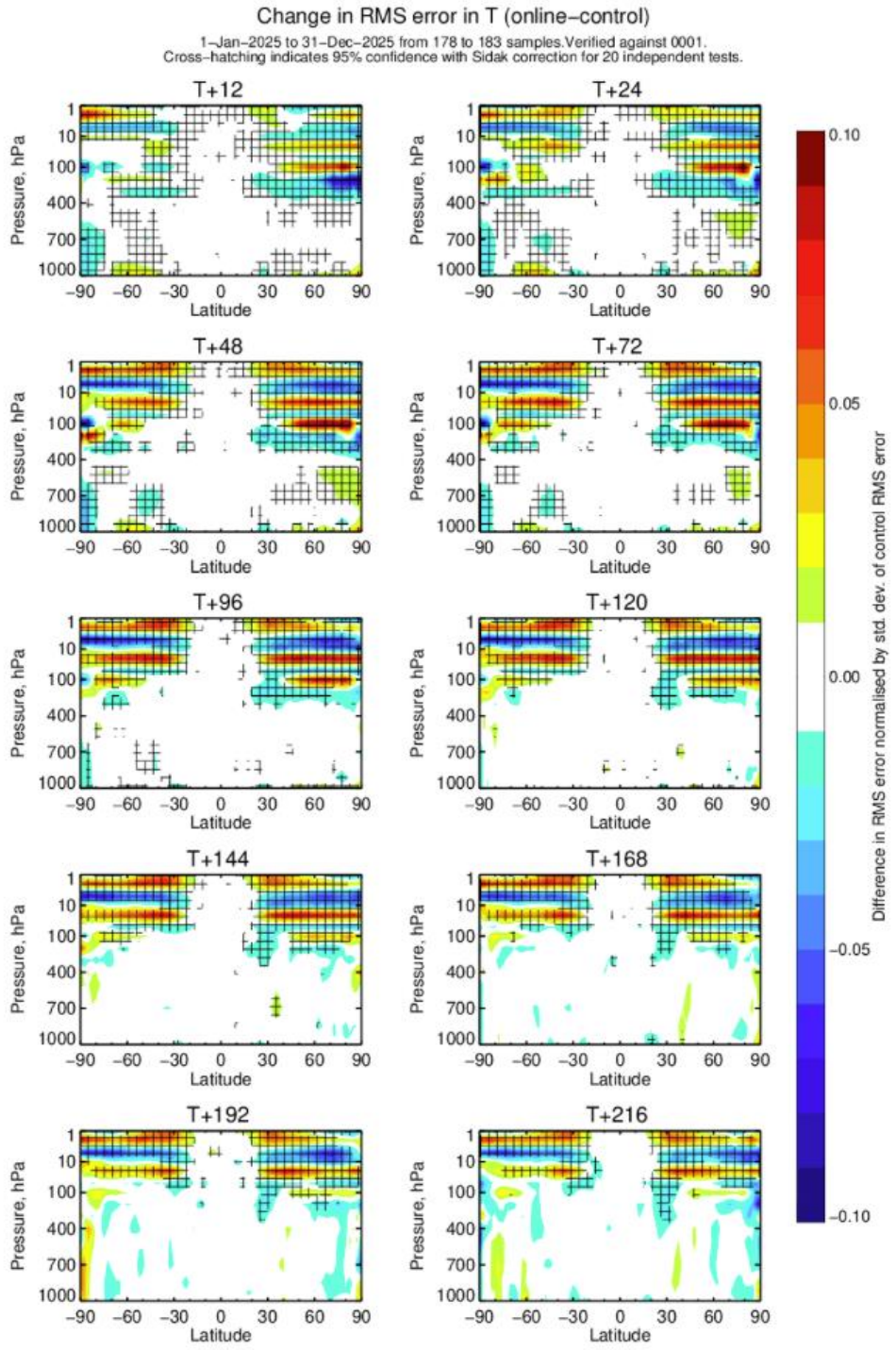


ECMWF

Accuracy of the online hybrid model

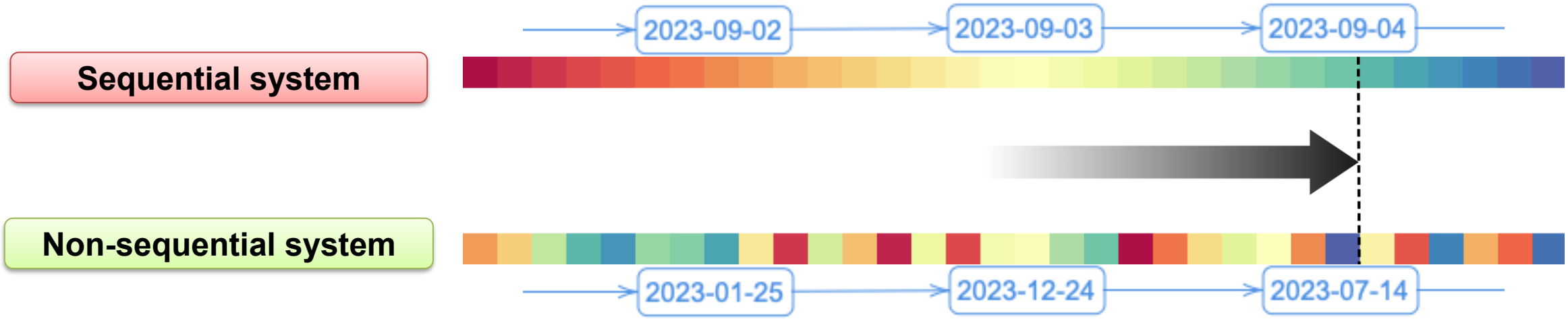
- Forecast-only experiments in 2025 (unseen data) at TCo399
- The online NN correction does not generalise to unseen data





Limitations of a sequential NN 4D-Var system

- NWP models are not autonomous, and model errors have seasonal components
- NN 4D-Var has been tested in a sequential setup
  - NN parameters may only keep in memory information about model errors in the recent past
- We need to relax the sequential assumption of the system



NN 4D-Var – a reimplementation of SGD?

	NN 4D-Var	Targeted NN 4D-Var	SGD
Control Variable	x and θ	Only θ	Only θ
Order of windows / sample	Sequential	Sequential / Random	Random
Parameter updated mechanism	$\theta(t+1) = \theta(t) + \delta\theta^{a}(t)$	$\theta(t+1) = \theta(t) + \delta\theta^{a}(t)$	$\theta(t+1) = \theta(t) - \gamma\nabla_{\theta}\mathcal{L}\,(\theta(t))$

- There is a clear similarity between one cycle of the targeted NN 4D-Var and one iteration in a SGD algorithm, with

$$\delta\theta^{a}(t) \Leftrightarrow -\gamma\nabla_{\theta}\mathcal{L}\,(\theta(t))$$

Batching NN 4D-Var

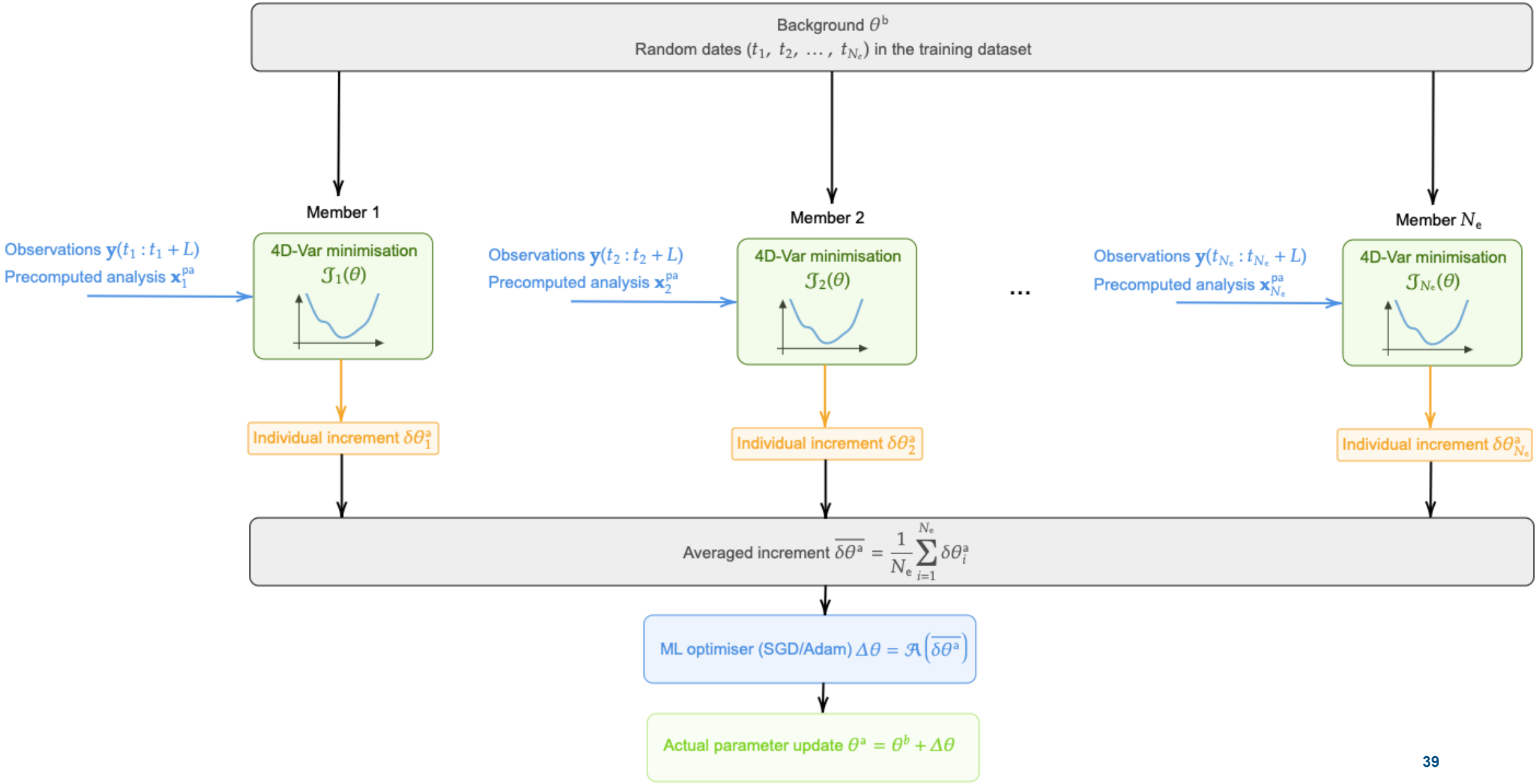
- In a batched gradient descent algorithm, the parameter update rely on the averaged gradient over a (mini) batch:

$$\overline{\nabla_\theta \mathcal{L}\left(\theta(t)\right)} = \frac{1}{N_e}\sum_{i=1}^{N_e} \nabla_\theta \mathcal{L}^i\left(\theta^i(t)\right)$$

- Pushing the analogy further, we define a batched-variant of NN 4D-Var, where several windows are processed in parallel

- Effectively multiplies by N_e the number of observation windows that the NN see in a fixed amount of optimisation steps

Batching NN 4D-Var



Implementation of the batched NN 4D-Var

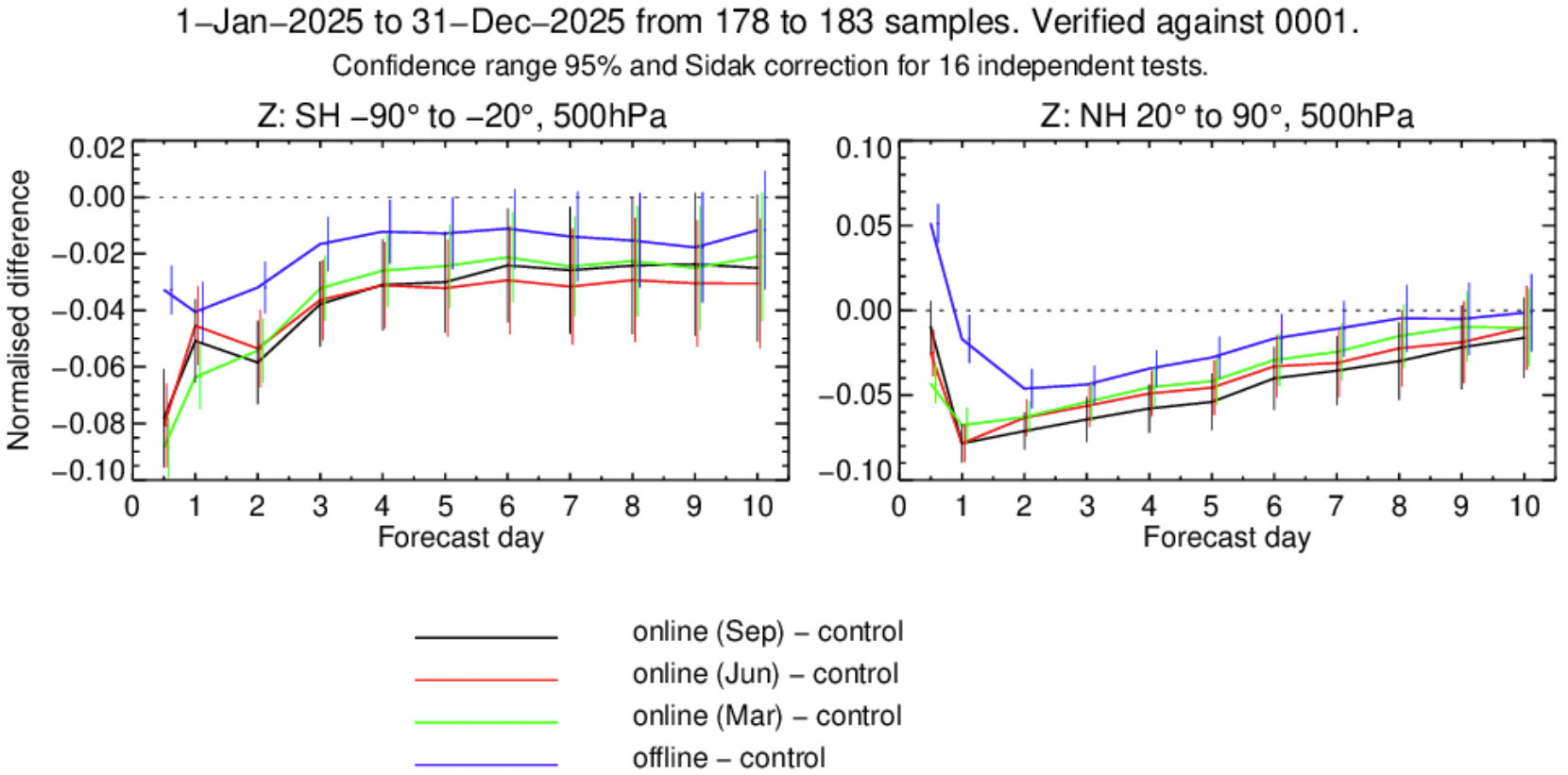
- We built the batched NN 4D-Var on the EDA framework
- Base time → fictitious time to synchronise the ensemble members
- Each member gets its own (random) date from the dataset:
  - Needs to retrieve observations, initial conditions, etc.
  - Needs to clean-up, with additional safeguards to ensure that members do not interfere with each other
- Synchronisation step added to merge individual analyses
- This is where the NN optimisation step takes place!

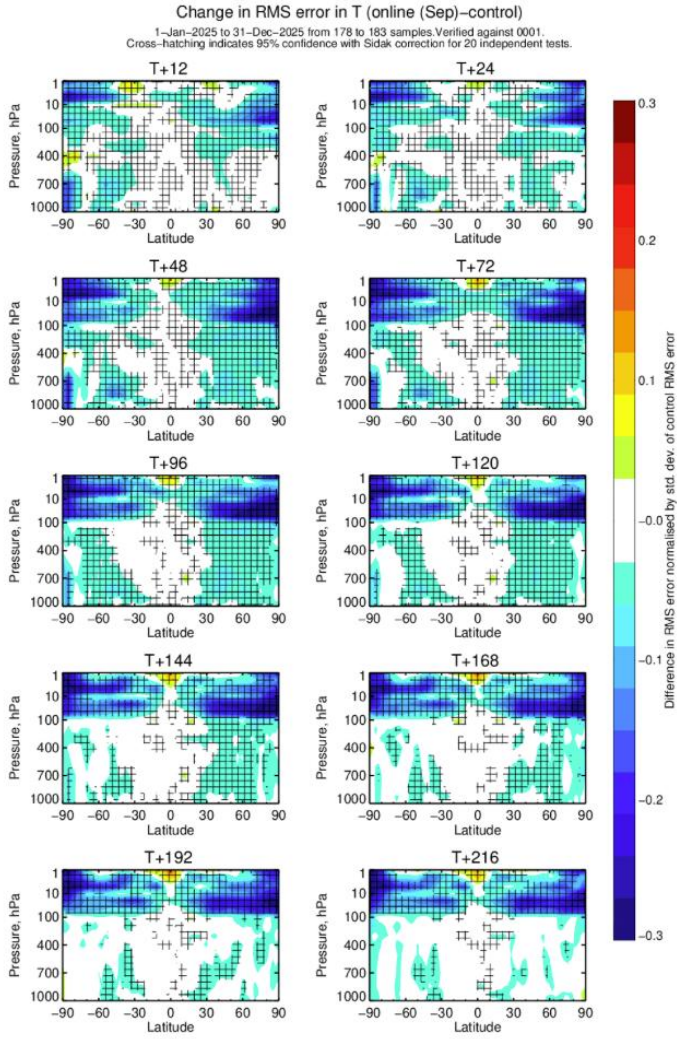
ECMWF

Accuracy of the online hybrid model

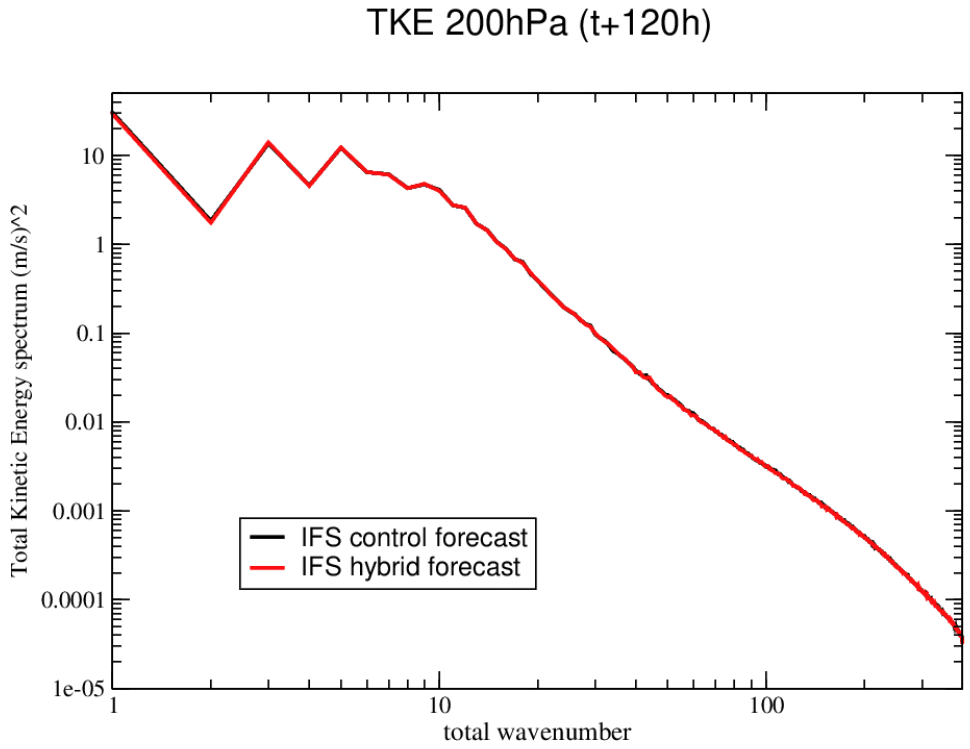
- Online training at TCo399, with 8 members
- Training dataset: 182 48h-windows over 2024
- Hybrid model evaluated in forecast-only experiments in 2025 (unseen data) after approximately 2, 5, and 8 fictitious months of training

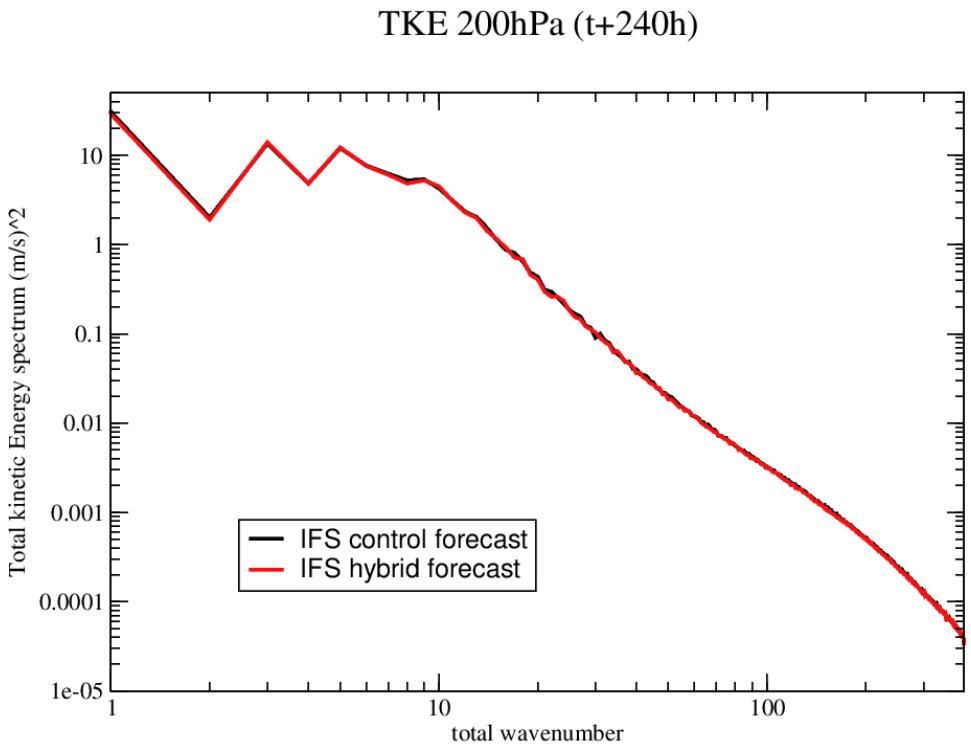
Fictitious training time	Optim. steps	DAWs	Epochs
2024-01-01 to 2024-03-01	31	248	1.36
2024-01-01 to 2024-06-01	77	616	3.38
2024-01-01 to 2024-09-01	123	984	5.41





Physical consistency

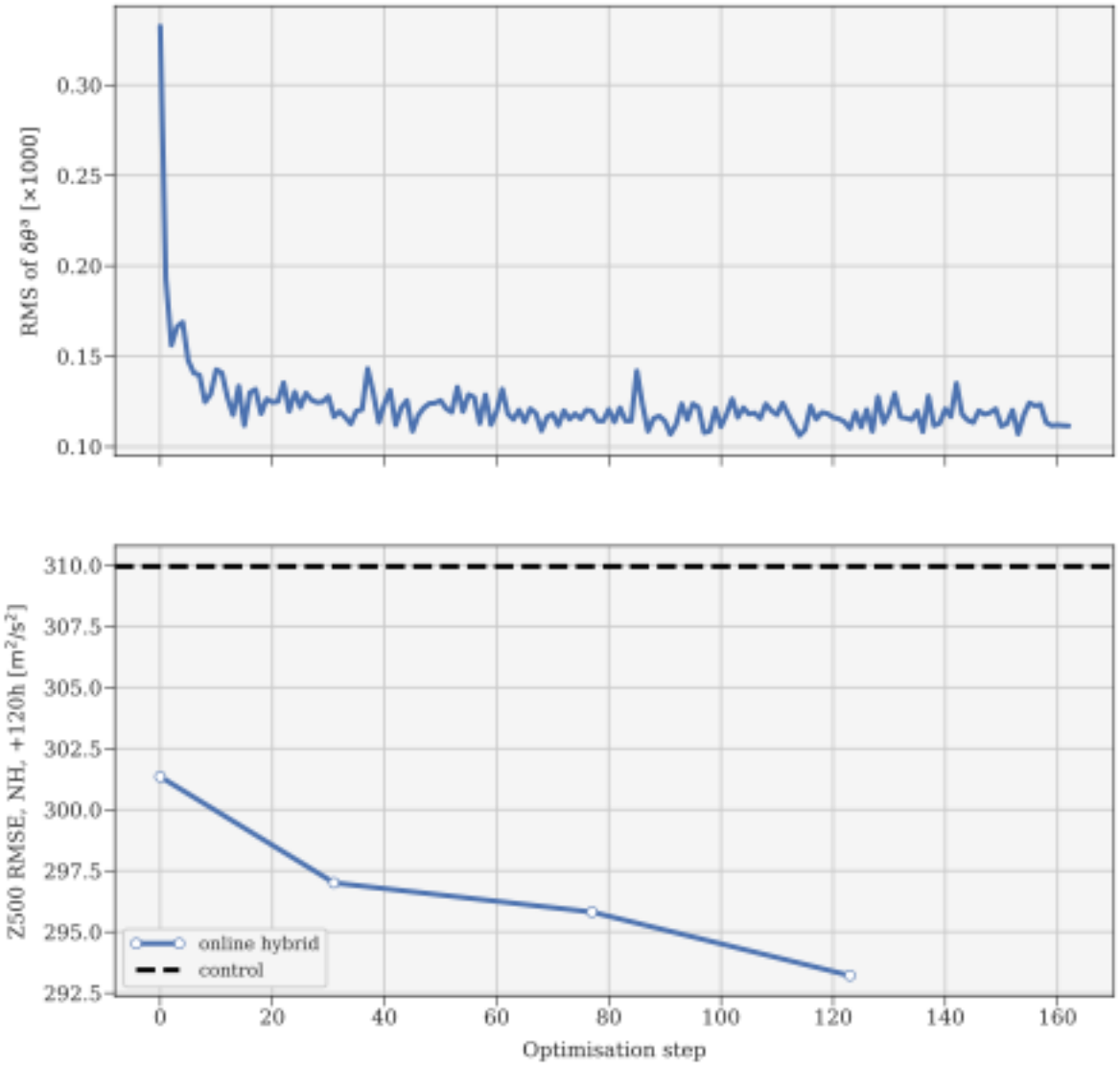




- Dynamically indistinguishable from the IFS forecasts

Convergence of the model parameters

- After an initial adjustment phase, the norm of the increments stabilises
- 120 steps (~ 5 epochs) may not be sufficient to extract the information from the 1-year dataset
- Towards longer training dataset?



Conclusions

- Hybrid DA + ML methods can be used to develop surrogate- and hybrid models
- We have illustrated the progressive development of a NN-based model correction for the IFS, from offline to batched online learning
- Hybrid modelling is still an active area of research
  - There is no clear consensus on whether an approach is clearly better than the others
  - More to come in the following years!