Assessment of ARPEGE-Climat using a neural network convection parameterization based upon data from SPCAM 5

Blanka Balogh[1], David Saint-Martin[1], Olivier Geoffroy[1], Mohamed Aziz Bhouri[2], Pierre Gentine[2,3]

[1]CNRM, Université de Toulouse, Météo-France, CNRS, Toulouse, France
[2]Department of Earth and Environmental Engineering, Columbia University, New York, NY, USA
[3]Center for Learning the Earth with Artificial intelligence and Physics (LEAP), Columbia University, New York, NY, USA

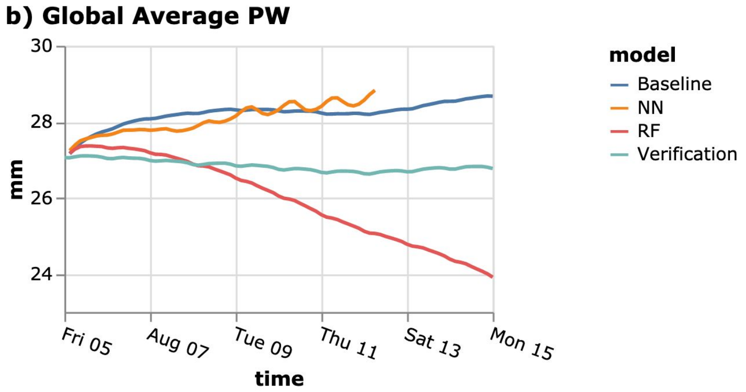
Data-driven (DD) parameterizations

Current data-driven parameterizations:

- Great *offline* score (eg., Rasp et al., 2018, Yuval et al., 2020)
- Only a few *online* test, frequent coupling instabilities (eg., Brenowitz, Henn et al., 2020, Rasp, 2020)

Why are there so few online tests available?

- Climate models (often) in fortran, DD parameterizations in python.
- a few solutions available (eg., Infero, FKB), but often tied to pytorch or TF/Keras.



Divergence of precipitable water during an online test of a DD parameterization. (Brenowitz, Henn et al., 2020)

Objectives

Implementation and *online* evaluation of a neural network (NN) based parameterization in ARPEGE-Climat (Roehrig et al., 2016)

NN parameterization of deep+shallow convection : Bhouri et al., 2023 (preprint).

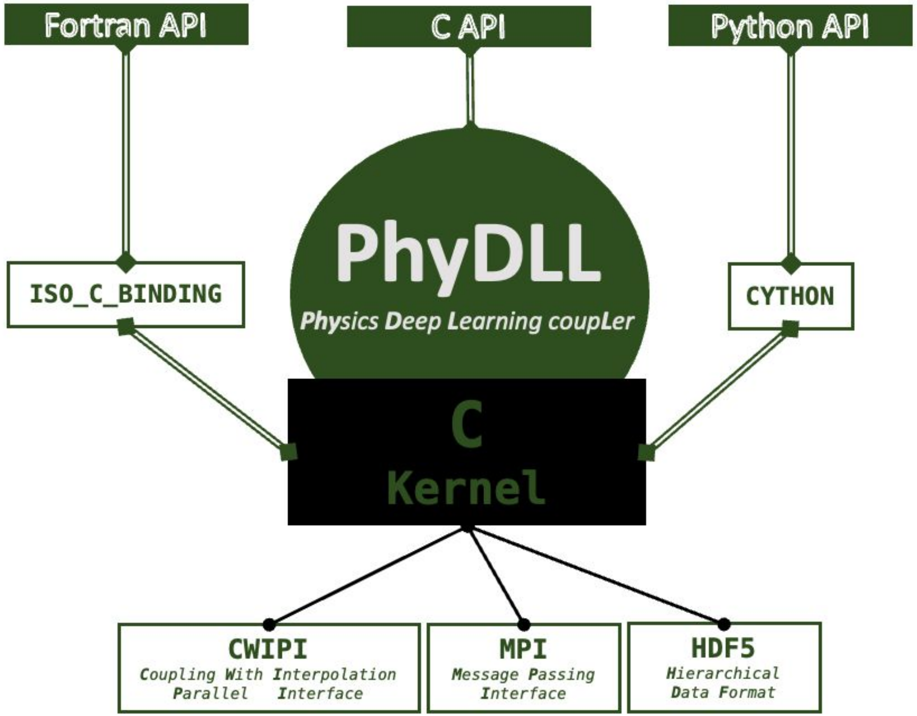
- ensemble of 128 feedforward neural networks (stochastic)
- trained on data from a CAM & SPCAM-5 simulation, in a multi-fidelity setting
- jax-based

Additional challenge: which adaptations are needed to use the neural network trained on CAM & SPCAM-5 data in ARPEGE-Climat?

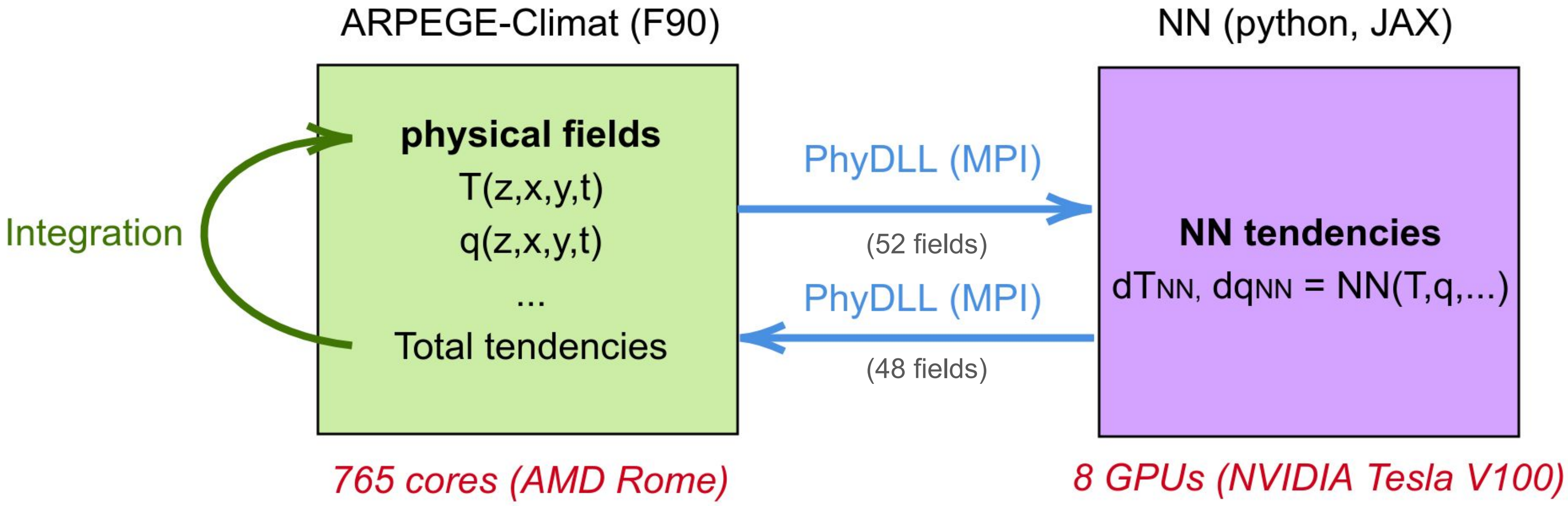
Fortran/python coupling – 1/3

Coupling package : **PhyDLL-0.2** (Serhani et al., under review ; developed by CERFACS).

- designed for physical solvers coupled to DL frameworks, both running on a GPU partition
- MPI to communicate
- C kernel: communication speedup
- User friendly python API, a bit more complicated in fortran.



Fortran/python coupling – 2/3



Different use of PhyDLL: ARPEGE-Climat and the NN run on separate partitions.

Fortran/python coupling – 3/3

How many gridpoints?

- ARPEGE-Climat runs at a resolution of : ~50km or **136k gridpoints**
- 178 gridpoints/CPU (ARPEGE)
- ~17k gridpoints/GPU (NN)

How fast/slow is it?

- **0.85s/timestep: expensive**
- inference time on GPUs ~3 ms (vs. ~0.5s on CPUs)
- fortran -> python MPI communication: ~12ms
- python -> fortran MPI communication: ~0.7s

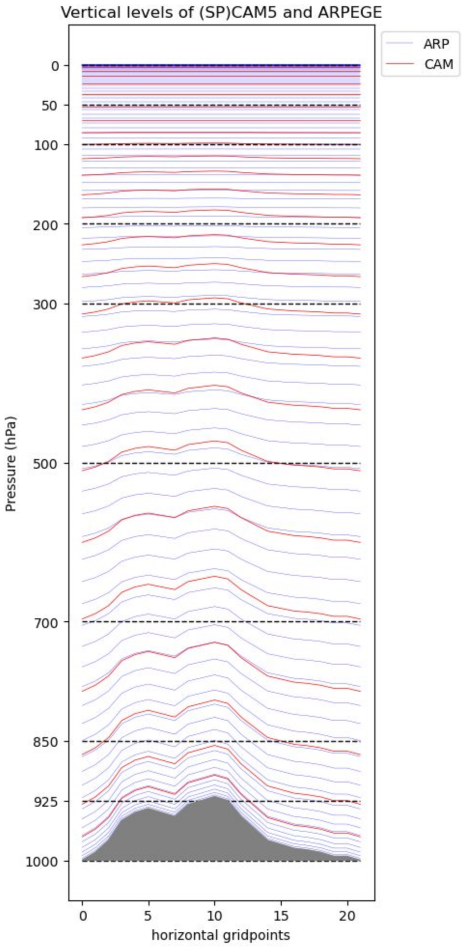
ARPEGE-Climat vs. SPCAM-5

Adaptations in ARPEGE-Climat:

- **New ARPEGE-Climat configuration with 26 vertical levels, matching CAM & SPCAM-5's**
  (This version of ARPEGE-Climat has 50 vertical levels by default)

- Transformation of variables (eg., TOA incoming solar radiation * cos(μ), dT/dt to dh/dt conversion)

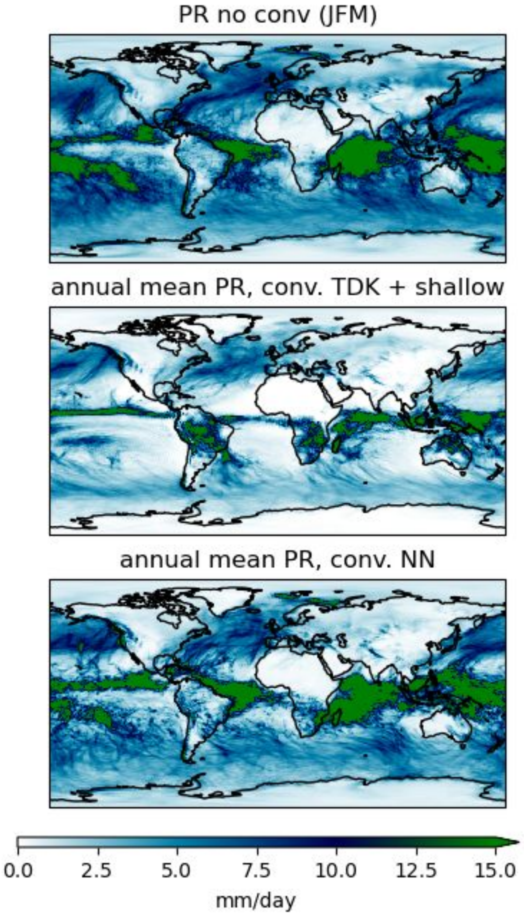
- NN only outputs dT/dt and dq/dt: **precipitations need to be diagnosed in ARPEGE-Climat from NN dq/dt**



Online run

The NN replaces deep + shallow convection:

- **Stability: 1 year without explosion**.
- Great *offline* performance of the NN when using ARPEGE-Climat data as input.
- Convection is too weak when the NN replaces (*online*) deep + shallow convection: **requires further calibration.**



Conclusion

- We present a version of ARPEGE-Climat that interacts with a NN-based parameterization for convection.

- The fortran/python interfacing is achieved using **PhyDLL**.

- We have performed a first test of a NN parameterization in ARPEGE. The NN parameterization was trained using data from SPCAM-5.

- The implementation of the NN parameterization required careful preparation in ARPEGE. We have performed a 1-year-long, stable simulation.

- Performance can be improved by calibration of ARPEGE and/or the NN ensemble.

Thank you for your attention!