Revisiting Machine Learning Approaches for Short- and Longwave Radiation Inference in Weather and Climate Models, Part II: Online Performance

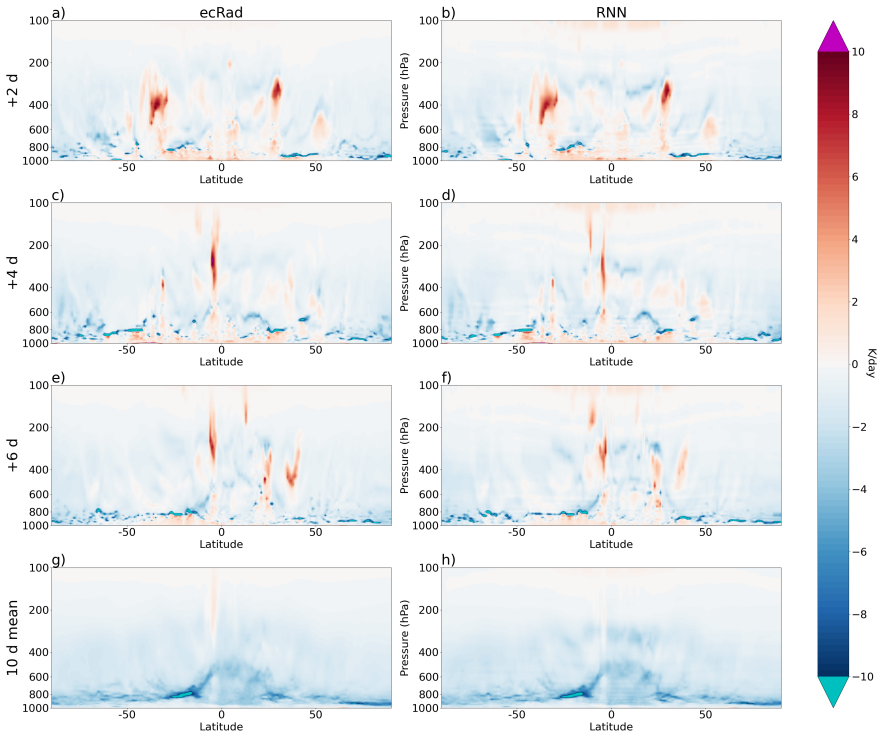
Guillaume Bertoli[1], Salman Mohebi[2], Firat Ozdemir[3], Jonas Jucker[4], Stefan Rüdisühli[3], Fernando Perez-Cruz[2], and Sebastian Schemm[3]

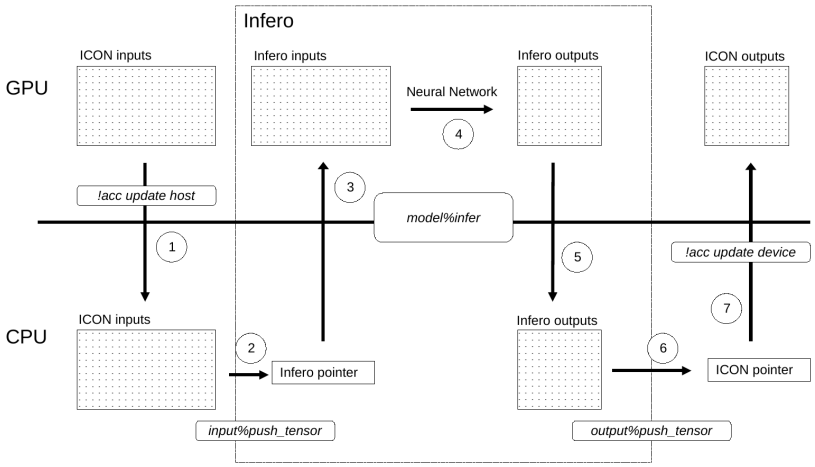
[1]ETHZ
[2]Swiss Data Science Center, ETH Zurich
[3]ETH Zurich
[4]Center for Climate Systems Modeling (C2SM)

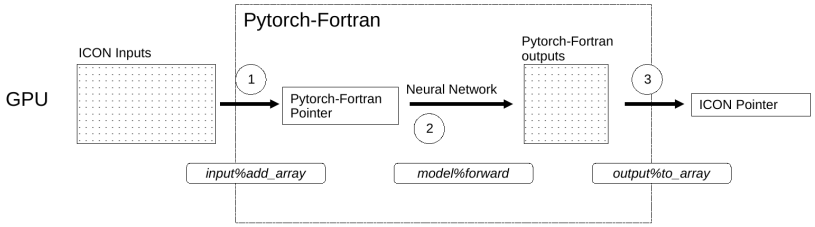
April 16, 2024

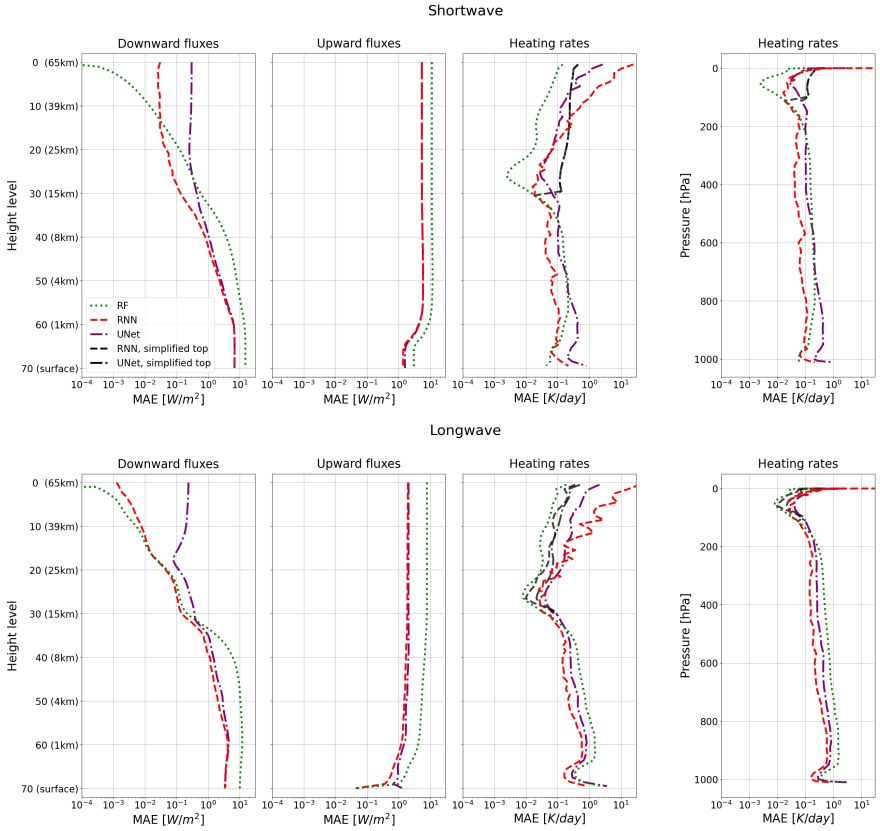
Abstract

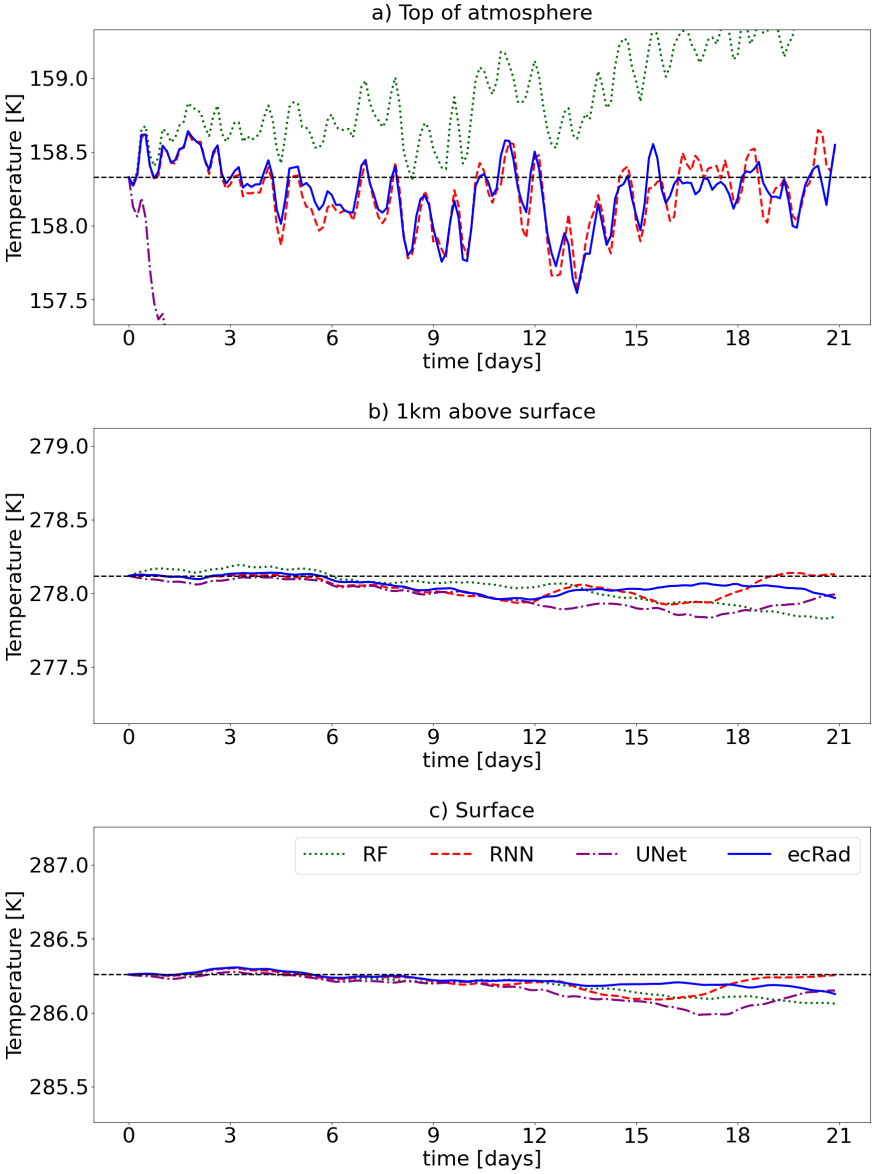
This paper continues the exploration of Machine Learning (ML) parameterization for radiative transfer for the ICOsahedral Nonhydrostatic weather and climate model (ICON). Three ML models, developed in Part I of this study, are coupled to ICON. More specifically, a UNet model and a bidirectional recurrent neural network (RNN) with long short-term memory (LSTM) are compared against a random forest. The ML parameterizations are coupled to the ICON code that includes OpenACC compiler directives to enable GPUs support. The coupling is done through Infero, developed by ECMWF, and PyTorch-Fortran. The most accurate model is the bidirectional RNN with physics-informed normalization strategy and heating rate penalty, but the fluxes above 15 km height are computed with a simplified formula for numerical stability reasons. The presented setup enables stable aquaplanet simulations with ICON for several weeks at a resolution of about 80 km and compare well with the physics-based radiative transfer solver ecRad. However, the achieved speed up when using the emulators and the minimum required memory usage relative to the GPU-enabled ecRad depend strongly on the Neural Network (NN) architecture. Future studies may explore physics-constraint emulators that predict heating rates inside the atmospheric model and fluxes at the top.











Revisiting Machine Learning Approaches for Short- and Longwave Radiation Inference in Weather and Climate Models, Part II: Online Performance

Guillaume Bertoli[1], Salman Mohebi[2], Firat Ozdemir[2], Jonas Jucker[4], Stefan Rüdisühli[1], Fernando Perez-Cruz[2,3], and Sebastian Schemm[1]

[1]Institute for Atmospheric and Climate Science, ETH Zurich, Zurich, Switzerland
[2]Swiss Data Science Center, ETH Zurich and EPFL, Zurich, Switzerland
[3]Computer Science Department, ETH Zurich, Zurich, Switzerland
[4]Center for Climate Systems Modeling, ETH Zurich, Zurich, Switzerland

Key Points:

- The Fortran-based and OpenACC-enhanced ICON weather and climate model is coupled to a neural network radiation solver developed in Python and both run in tandem on graphic processing units (GPUs). The resulting speed-up critically depends on the architecture of the neural network.
- ICON with the radiation emulator runs stable for several weeks with a negligible difference to ecRad, but tuning at the top of the model at very low pressure values is required.
- Future research could explore physics-informed radiation emulators that predict heating rates inside the atmosphere and fluxes at the surface and model top as auxiliary fields.

Corresponding author: Guillaume Bertoli, `guillaume.bertoli@env.ethz.ch`

Abstract

This paper continues the exploration of machine Learning (ML) parameterization for radiative transfer for the ICOsahedral Nonhydrostatic weather and climate model (ICON). Three ML models, developed in Part I of this study, are coupled to ICON. More specifically, a UNet model and a bidirectional recurrent neural network (RNN) with long short-term memory (LSTM) are compared against a random forest. The ML parameterizations are coupled to the ICON code that includes OpenACC compiler directives to enable GPUs support. The coupling is done through Infero, developed by ECMWF, and PyTorch-Fortran. The most accurate model is the bidirectional RNN with physics-informed normalization strategy and heating rate penalty, but the fluxes above 15 km height are computed with a simplified formula for numerical stability reasons. The presented setup enables stable aquaplanet simulations with ICON for several weeks at a resolution of about 80 km and compare well with the physics-based radiative transfer solver ecRad. However, the achieved speed up when using the emulators and the minimum required memory usage relative to the GPU-enabled ecRad depend strongly on the Neural Network (NN) architecture. Future studies may explore physics-constraint emulators that predict heating rates inside the atmospheric model and fluxes at the top.

Plain Language Summary

Machine learning (ML) methods could drastically accelerate existing parts of weather and climate models. This research explores machine learning methods to replace the radiation solver responsible for computing the solar and terrestrial radiative fluxes in the ICON model. Three ML models are trained for this task: a random forest, which combine decision trees to make accurate prediction, and two neural networks, an increasingly popular deep learning model which learn from data to perform tasks using interconnected mathematical function. Because the ML models and ICON are implemented with different programming languages, a auxiliary software is used to couple them. Simulations with the ML models show accurate results for two-weeks simulations but the acceleration depends strongly on the ML method.

1 Introduction

This paper explores machine Learning (ML) parameterizations of simulated radiative transfer in the atmosphere. This second part of our two-part study presents a detailed description of how the offline trained ML models, which are developed using PyTorch (Paszke et al., 2017) and Tensorflow (Abadi et al., 2015), are incorporated into the Fortran code (Kedward et al., 2022) of the ICOsahedral Nonhydrostatic weather and climate model (ICON) (MPI-M et al., 2024; Giorgetta et al., 2022), enabled for graphics processing units by means of OpenACC (*OpenACC, version 3.2*, 2022), and the steps required to make the radiation emulator and ICON performant on graphic processing units (GPUs) on Piz Daint, a Cray XC50 machine at the Swiss National Supercomputing Centre (CSCS).

As discussed in detail in the first part, radiative transfer emulation has a long history in atmospheric modelling. Previous research includes studies that emulate the entire radiative transfer code (Chevallier et al., 1998; Krasnopolsky et al., 2005; Ukkonen, 2022; Roh & Song, 2020; Pal et al., 2019; Lagerquist et al., 2021), while others focus on replacing individual components of existing radiative schemes (Ukkonen et al., 2020; Veerman et al., 2021). Some studies use radiative flux as a training target (Chevallier et al., 1998; Ukkonen, 2022), while others directly predict the resulting heating rates (Krasnopolsky et al., 2005; Roh & Song, 2020; Pal et al., 2019; Lagerquist et al., 2021). The advantages and disadvantages of emulating shortwave and longwave radiative fluxes or directly predicting the resulting heating rates are known: Direct prediction of heating rates allows the calculation of radiative flux convergence to be omitted, thus avoiding numerical sta-

bility problems, especially when the predicted flux profile is not smooth. However, the prediction of fluxes seems to be a necessity for an earth system model (ESM), since the radiative flux serves as an input to other components, such as the land model. It is also a relevant output variable as it serves as an input to impact models (e.g., solar energy production) and can be compared with measurements, which is important for validation. For a more detailed literature review, the reader is referred to Part I of this study.

In Part I of this study (Bertoli et al., 2024), the decision was made to employ a single Neural Network (NN) to predict both short- and longwave up- and downward fluxes, using the resulting heating rates as an additional penalty to the loss function during the training. Additionally, a physics-informed normalization strategy proved beneficial for enhanced training and inference accuracy. Longwave training data was normalized using the Stefan-Boltzmann law and shortwave data was normalized using the cosine of the solar zenith angle multiplied by the solar constant. This additional physics-informed regularization also benefited the random forest (RF) baseline model. In general, the accuracy of the RF scaled with its memory usage, but at fixed memory usage the RF was typically outperformed by most NN architectures. The only exception was at the top of the atmosphere (TOA), where only the most advanced NN, a recurrent neural network (RNN), showed higher accuracy than the RF in predicting fluxes. However, a particular advantage of the RF, besides its simplicity during (re)training, was that the predicted flux profile is relatively smooth, and hence the derived heating rates did not yield spurious peaks as observed with some NNs. This feature make the RF an ideal candidate for low-cost, fast and simple radiation flux emulation.

The structure of Part II is as follows: Section 2 describes the implementation details for the ML radiative transfer parametrizations. Section 3 presents offline and online results. We conclude the paper with a summary in Section 4.

2 Methods

We start with the dataset construction for training the ML models, followed by the architecture choices for the RF and NN emulators, including normalization of inputs and outputs, the loss function for training the NNs, and a modification to the computation of fluxes above 15 km height to ensure ICON's stability at height-levels with very low pressure. Finally, we explain the coupling of the ML emulators with ICON using Infero (Antonino Bonanni, 2022) developed at ECMWF and PyTorch-Fortran (Alexeev, 2022).

2.1 Dataset

A two-years-long ICON aquaplanet simulation is performed with a physics time-step of 3 min and a horizontal resolution of approximately 80 km (ICON grid R02B05). The radiation parameterization ecRad is called every time-step and at full spatial resolution. The simulation uses 70 vertical levels and thus 71 vertical half levels, ranging from index 70 at the surface to index 0 at the TOA (65 km height). The solar zenith angle is held constant at equinox.

The first year of the simulation is considered as a spinup phase during which no data are stored. Starting at the beginning of the second year, the required inputs and outputs are stored every 3 h and 3 min (183 min output interval). This is slightly different from what is done in Part I (Bertoli et al., 2024), where data are stored every 3 h. The new strategy allows for a more complete coverage of different Sun positions over time in the dataset, which proved beneficial for the training. After 61 simulated days (480 time-steps stored every 183 min), the dataset contains all possible angles based on the 3 min time-step interval. All NNs trained with this dataset outperform the old models.

The dataset is split into two parts: the initial 270 days for model training and the last 88 days for testing. A 7-day gap is included between these datasets to allow the test set distribution to vary from the training set. Within the initial segment, two sub-sections are established. The first and last 20 days of the training dataset serve as a validation dataset for the ML models during training, aiding in determining when to halt the training process using an early stopping criterion. The remaining 230 days are designated as the actual model training dataset.

In ICON, the dynamical core, the horizontal diffusion, the tracer advection, the fast physics and then the slow physics processes are solved sequentially. The radiative transfer parameterization is part of the slow physics processes, which are solved in parallel. It is therefore essential to store the states of the inputs after they have been updated by the fast physics but before the slow physics update. For this reason, we modified the ICON code to extract the states of the input variables to ecRad in the middle of the sequential time splitting.

2.2 Radiation emulation: ML architecture

In Part I (Bertoli et al., 2024), we explored different ML architectures as possible emulators of ecRad. First, an RF emulator composed of 10 trees served as a baseline model and was used to assess the performance of the NNs. While RFs could get very accurate, their memory footprints quickly surpassed 100GB and they thus became prohibitively memory-intensive. For this reason, RF size was constrained by imposing a minimum leaf size equal to 0.01% of the training dataset size.

Second, three NN architectures were explored. A baseline feed-forward multilayer perceptron (MLP) served to evaluate the performance of more complex architectures. This MLP architecture, since it was outperformed by more complex architectures in Part I, is not considered further in this paper for the online tests. A convolutional NN was constructed, more precisely a UNet, which allowed us to reduce the number of parameters significantly. Lastly, a RNN was constructed, more precisely a bidirectional RNN with long short-term memory (LSTM), which was the most accurate model. The RNN closely imitates the ecRad parameterization, which solves the effect of each atmospheric layer sequentially. The accuracy gain of the RNN compared to the UNet might result from the fact that the RNN has access to its own prediction in the layers above (for downward fluxes) and below (for upward fluxes). However, as discussed in Section 3.2, the RNN emulator is significantly slower than the UNet emulator and requires more memory. In Tab. 1, the exact number of layers and number of neurons for each of the two NNs are listed. To improve the accuracy of the emulators, normalization strategies and custom loss functions are employed as detailed in Bertoli et al. (2024) and briefly reiterated here.

The ML model outputs normalizes short- and longwave up- and downward fluxes. The normalization is chosen such that for each atmospheric column, the model returns values that are approximately in the range [0–1] as in Ukkonen (2022). An exception are columns without incoming solar radiation, where the model returns zero shortwave fluxes at each height by definition. Each atmospheric column's shortwave fluxes are divided by the cosine of the solar zenith angle multiplied by the solar constant (approx. $1400\,\mathrm{kW\,m^{-2}}$). Atmospheric columns whose cosine of solar zenith angle is smaller than 10^{-4} are not normalized in the shortwave because these are truncated in ICON for numerical stability reasons at the day-night interface. Following the Stefan-Boltzmann law (Petty, 2006)[Chapter 6.1.3] for the emission of a black body, the longwave fluxes are divided by the fourth power of the surface temperature multiplied by the Stefan-Boltzmann constant. This normalization strategy improves the results of all NNs and of the RF. Each input field is also normalized by subtracting its mean and dividing by its standard deviation computed from the training dataset. Note that the input features are normalized across all heights,

Table 1: *NN architectures.*

Model	Architecture Overview
UNet	The 2D features are broadcasted and concatenated with the 3D features (along height axis with size 70). A UNet with convolutional units [128, 256, 512, 1024] and pooling layers [1, 2, 5, 7] are then applied. Finally, a 1D convolutional layer with 4 layers along the height axis and a final dense layer maps to the outputs to size 71x4.
RNN	The 2D features are broadcasted and concatenated with the 3D input (along height axis with size 70). The features are then concatenated with a constant vector (all ones) such that the height is equal to the output height (i.e., 71). The feature vectors at each height level are then passed through an MLP with units [128, 256]. A Bidirectional LSTM layers with units [128, 256, 512] is applied. Finally, a dense layer is applied which maps the 71x512 hidden units to the 71x4 output features.

which means that for each field (temperature, cloud cover, etc.), only a single mean and standard deviation are computed. This approach is adopted because, at higher atmospheric levels, certain fields that should be zero end up numerically near, but not equal to zero. Normalizing across all heights prevents the unnatural scaling that would occur if they were normalized independently at each height.

For the custom loss function, the radiative heating rates for each atmospheric layer are computed by an ICON routine, using a finite-difference approximation of the vertical flux derivative. Obtaining accurate heating rates from the predicted fluxes is essential for updating the thermodynamic equation in ICON. Therefore, the mean squared error of the heating rates calculated from the emulated flux is added to the loss function of the NNs. The loss function thus consists of two parts: the mean squared error of the flux prediction and the mean squared error of the heating rates.

A challenge of the ML emulation of ecRad found during online inference is obtaining accurate heating rates at atmospheric levels above 35 km height. At those heights, the air mass between two model levels is extremely small and small errors in the height profile of the fluxes can result in large heating rate errors which can then break ICON simulations. The proposed loss function already ensures improved heating rate estimations, but it turns out not to be enough to ensure the stability of the ICON model at its top. For this reason, the computation of the fluxes above 15 km height is modified[1]. At those heights, a simplified formula is used to compute the heating rates. The fluxes above the chosen level $H = 30$ (15 km height) are constructed by multiplying the fluxes at level H by a set of constants $\beta_{H-1}, \ldots, \beta_0$, which are optimized based on the training set:

$$f_{l,k} = \beta_k f_{l,H}, \tag{1}$$

where $f_{l,k}$ is the chosen flux for the atmospheric column l at height level k and β_k are different for short- and longwave up- and downward fluxes. See Appendix A for more details.

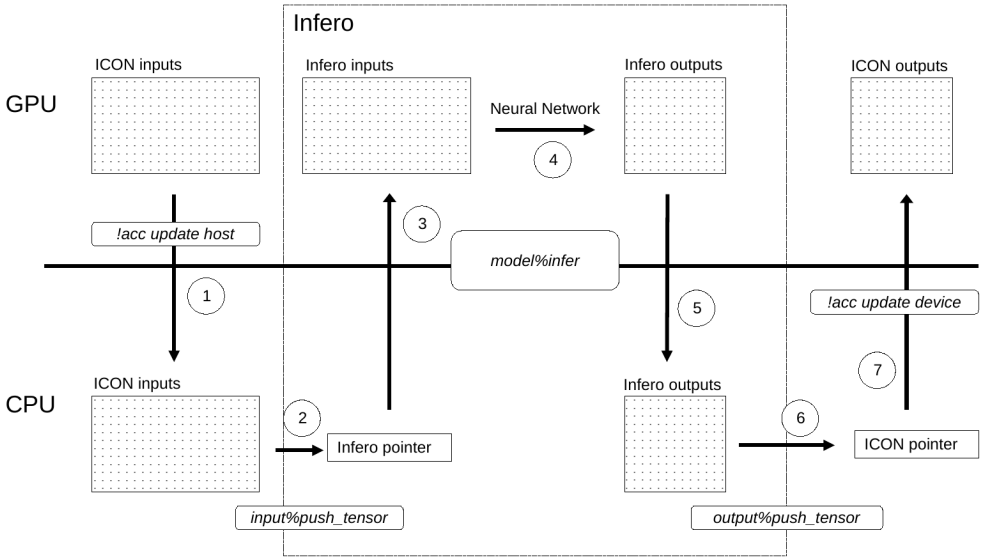


Figure 1: *Schematic illustrating the integration of the ML emulator into ICON with Infero. Infero requires both the input and output data to be copied back and forth between GPU and CPU memory (steps 1, 3, 5 and 7), which substantially slows down the ICON simulation. First, the input data from ICON are copied from GPU to CPU memory (step 1). Infero obtains a CPU pointer to these ICON inputs (step 2). The "Tensorflow for C" backend then copies the inputs to the GPU memory (step 3), calls the NNs (or RF) for the fluxes predictions on GPUs (step 4) and finally copies back the outputs to the CPU memory (step 5). Finally, a CPU pointer to the Infero outputs is created (step 6) and the output data are copied back to GPU memory (step 7).*

2.3 Implementation of the ML emulators into ICON

The implementation of the ML models with Python into the Fortran code of ICON using Infero (Antonino Bonanni, 2022) is discussed here. Infero works with models built with the Tensorflow Python library (Abadi et al., 2015) or with models in the `ONNX` format (Bai et al., 2019). The RF is trained with the Scikit-learn (Pedregosa et al., 2011) Python library and then ported to the `ONNX` format. The NNs are implemented with Tensorflow. In Figure 1, we show a schematic of how Infero integrates into ICON running on GPUs. Infero requires the inputs to be in CPU (referred to as host) memory and after an ML prediction on the GPUs (referred to as device), it returns the outputs there. This is a limitation for a weather or climate model running on GPUs as it leads to back-and-forth copies of data between the separate memory spaces. Note however that for most models, which are running solely on the central processing unit (CPU), this is not an issue and Infero then becomes a suitable coupler. Furthermore, Infero is developed by ECMWF, which are increasing their efforts in using ML methods to improve current weather models (ECMWF, 2023). Infero may thus see improvements in the future, allowing for an optimized coupling on the GPUs. Hence, we still report next how Infero calls the NNs when coupled to ICON, although an alternative coupler allowing direct access of the data on the GPUs could be preferred for GPU-enabled weather and climate models.

[1] Note that the method we explain here does not appear in Bertoli et al. (2024) since the problem became apparent only after the emulator was integrated into ICON.

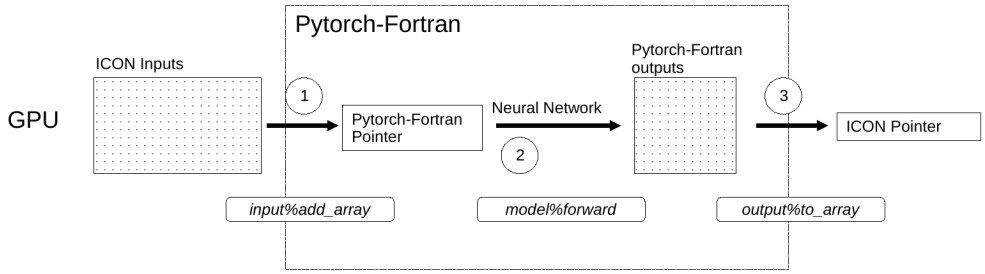


Figure 2: *Schematic illustrating the integration of the ML emulator into ICON with PyTorch-Fortran. In contrast to Infero, no copies between CPU and GPU memory are required. First, a pointer to ICON inputs is created (step 1). Then, the NNs is called (step 2). Finally, a pointer to PyTorch-Fortran outputs is created so that ICON can handle the predicted fluxes (step 3).*

In this paper, we experiment with the ICON version running fully on GPUs. The data are first copied from the GPU to the CPU (step 1 in Figure 1). Infero can then access the input data (step 2) and compute the outputs with the NNs or RF (steps 3–5). Internally, Infero uses a "Tensorflow for C" backend, which will copy the input data to GPU memory (step 3), run the ML model on the GPUs (step 4) and then copy the output data back to CPU memory (step 5). Finally, the output data are accessed by ICON (step 6) and copied back to GPU memory (step 7). The input and output data are thus copied twice each. This drastically slows down the computation of the radiative fluxes and diminishes the possible speed-up from using an ML model instead of ecRad.

An alternative to Infero is PyTorch-Fortran (Alexeev, 2022). Figure 2 shows how the ML models are integrated into ICON with PyTorch-Fortran. PyTorch-Fortran requires the models to be built with the Pytorch library instead of Tensorflow. The main advantage of Pytorch-Fortran is that both ML emulator and ICON can run purely on GPUs. PyTorch-Fortran first obtains pointers to the input data, then runs the ML model and return pointers to the output data to ICON. Therefore, it completely avoids copying data between CPU and GPU memory. This is a major advantage compared to Infero. A limitation is, however, that PyTorch-Fortran does not work with models built in the `ONNX` format. This renders it incompatible with the Scikit-learn library which contains a variety of ML models.

3 Results

3.1 Offline performance

Figure 3 shows the mean absolute error (MAE) of the different ML models. The RF is the most accurate above 15 km height for the heating rates and the downward fluxes. It is however the least accurate model below 15 km height, where accuracy is the most important. The RNN outperforms the UNet at all heights for both the fluxes and heating rates. The effect of the simplified formula (1) used to compute the fluxes above level 30 is shown in the heating rates profile. This simplified equation improves the shortwave heating rates accuracy above 35 km height and the longwave heating rates accuracy above 25 km for both NNs. On pressure coordinates, we observe that the increase in the heating rates error from levels 30–25 to level 0 at the TOA, is concentrated in a small pressure interval from around level 100 hPa up to the model top. This compares well with results of the literature shown in pressure coordinates (Chevallier et al., 1998; Ukkonen, 2022; Liu et al., 2020) or height coordinates with logarithmic scale (Lagerquist et al., 2021).

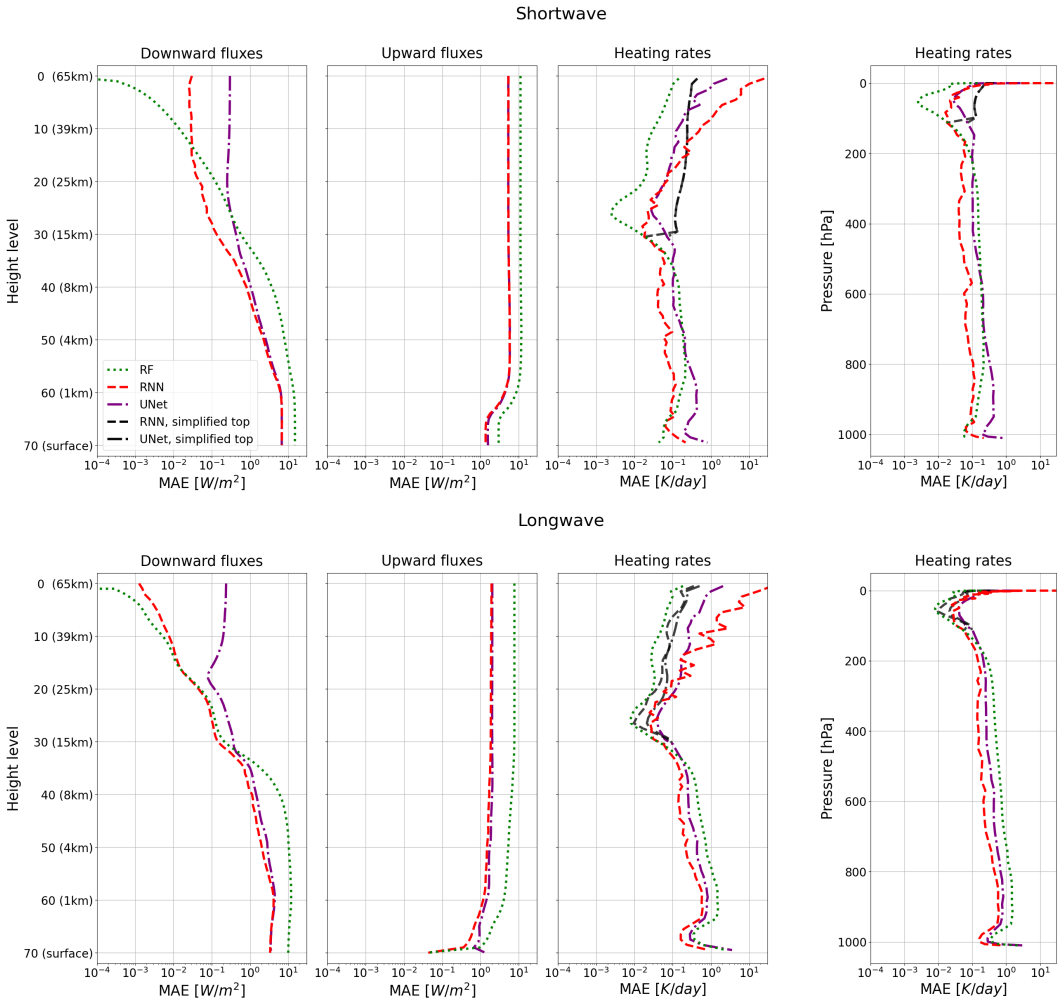


Figure 3: *MAE over the test set for the RF, the UNet and the RNN models. Above level 30, the simplified Equation* (1) *is used to compute the fluxes for the UNet and the RNN. The MAEs of the RNN and UNet without the simplified Equation* (1) *is shown in black. In the last column, the heating rates MAE is shown in pressure coordinates.*

3.2 Online performance

ICON simulations are performed over 3 weeks with the different ML emulators. Each simulation is restarted from the end of the 23$^{\text{rd}}$ month of the two-year simulation that produced the training and testing datasets. Recall that the ML models are trained with months 13 to 21. The experiments related to Figures 4 and 5 are performed on the Piz Daint HPC with NVIDIA Tesla P100 16GB GPUs (NVIDIA, 2016), while the runtime presented in Table 2 are performed on the Balfrin CSCS machine with NVIDIA A100 Tensor Core GPU (NVIDIA, 2022).

In Figure 4, the global mean temperatures at the TOA, at 1 km height and at the surface from ICON simulations with different radiation emulators are compared to the reference simulation with ecRad. At the TOA, the RNN model provides by far the highest accuracy. For the UNet model, the global temperature is dropping fast to below 135 K at the end of the three-weeks-long simulation. We extended the simulation with the UNet to two months and although the global temperature continues to drop at the TOA, this

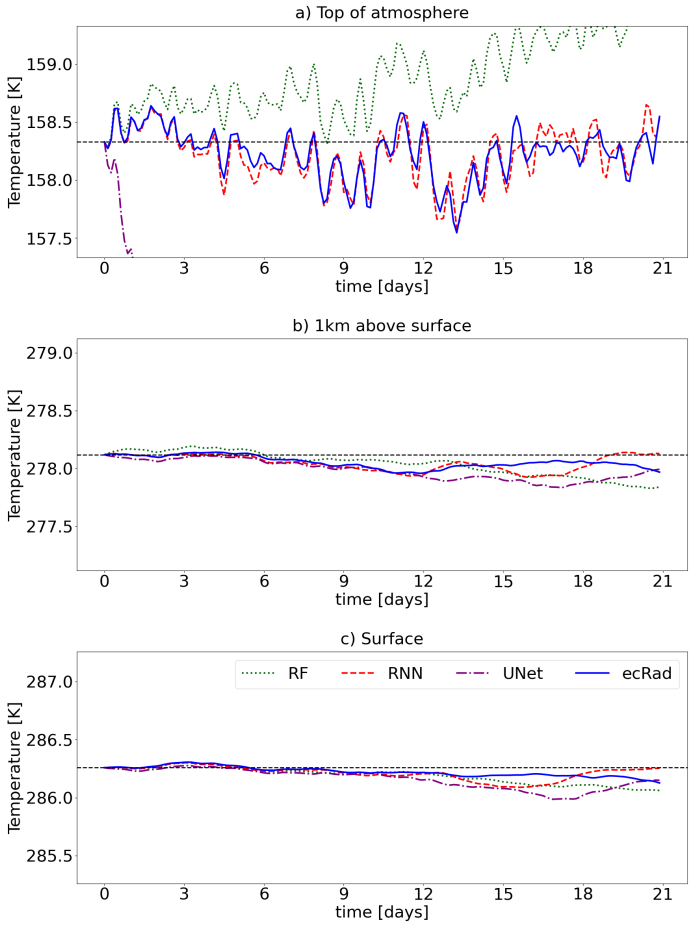


Figure 4: *Global mean temperature in Kelvin at the TOA, at 1 km height and at the surface for three-weeks-long ICON simulations with four different radiation parameterizations: RF, RNN, UNet and ecRad. The horizontal dotted line corresponds to the ecRad global mean temperature at time 0. Each vertical axis is centered on the ecRad global mean temperature at time 0 and has a 2 K range of values. At the TOA, the RNNs is the only ML model that provides an accurate global mean temperature with respect to ecRad.*

does not seem to affect ICON's stability. Furthermore, no perturbations of the lower levels due to the decrease of temperature at the top is observed. The temperature of the RF simulation is significantly more accurate than the UNet simulation although the temperature is increasing over time at the TOA. It is however the least accurate model at 1 km height and at the surface during the first ten days where accuracy matters the most. Extensive online simulation horizons with RNN are still competitive with the ecRad reference. Throughout a time horizon of two months, the temperature difference between RNN and ecRad simulations never exceed 2 K at TOA and 0.3 K at surface and 1 km height levels. The superior stability of the RNN makes it a good candidate for multi-years ICON climate simulations.

Figure 5 presents meridional vertical cross sections of heating rates along the prime meridian 2 d, 4 d, and 6 d into the simulation using ecRad and the RNN model. Both simulations are started from the same restart file after a 23 months of simulation with ecRad. This is possible because our implementation allows us to switch between ecRad and the RNN model during runtime. After two days, both parameterizations exhibit nearly iden-

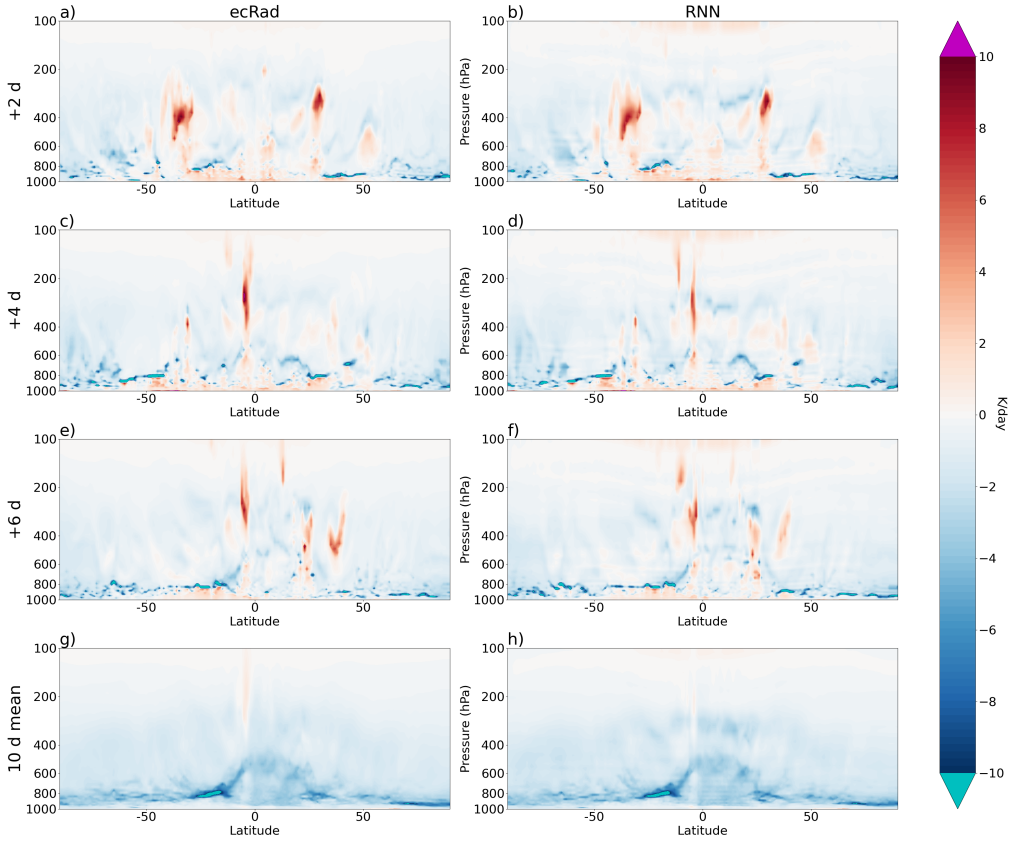


Figure 5: *Meridional vertical cross-sections along the prime meridian of net heating rates in pressure coordinates as computed by ICON with (left) ecRad and (b) the RNN radiation emulator. The rows show (top to bottom) instantaneous fields (a, b) 2 d, (c, d) 4 d and (e, f) 6 d into the simulation, as well as (g, h) the mean over the first 10 d. The bottom row shows a 10-day average. Both simulations have been started from the same restart file saved after 23 months of simulation with ecRad.*

tical heating rate profiles. By day 4, differences emerge near the equator. By day 6, they have become more pronounced above 600 hPa, which is expected as small differences between the simulations start to grow over time, similar as in ensemble runs. While the mean heating rates align in both simulations, a small discrepancy arises at 5°S where positive heating rates between 400 hPa and 100 hPa are underestimated on average in the RNN-based simulation. This is evident in the instantaneous heating rate displayed for 4 days and 6 days into the simulations in Figure 5. At this stage it is unclear whether this indicates a systematic emulation bias, deviating model trajectories due to growing perturbations (akin to different ensemble members) or a combination of both.

In Tab. 2, the run times of the radiation and the whole physics are shown for 400 ICON time-steps, corresponding to 20 hours of simulated time. The RNN model is twice as slow as ecRad and requires more than 8 GPUs (Nvidia A100 tensor core GPU with 80 GB of memory) to run. In comparison, the slim UNet model is as fast as ecRad with 12 GPUs, twice as fast with 8 GPUs and three times as fast with 4 GPUs. It can even fit on 2 GPUs in contrast to ecRad. As an alternative to the RNN model, a smaller model is trained, with similar MAE than the RNN model described in Table 1, with fewer LSTM layers and a single set of weights for the first MLPs of size [128, 256]. This tuning reduces the number of trainable parameters from 15 million to 5 million and the size of

Table 2: *Comparison of run time in seconds for 400 ICON time-steps using ecRad and the RNN and the UNet emulators as a function of the number of GPU nodes (Nvidia A100 tensor core GPU with 80 GB of memory). Out-of-memory (OOM) issues are indicated. Shown in brackets are runtimes of the all physics parameterization combined.*

No. GPUs	2	4	8	12
UNet	92 [116]	50 [65]	35 [49]	46 [57]
RNN	OOM	OOM	OOM	108 [117]
RNN optimized	OOM	143 [156]	77 [78]	48 [59]
ecRad	OOM	144 [184]	73 [99]	51 [73]

the RNN from 64 MB to 22 MB. The smaller model fits on 4 GPUs and the total run time of all physics parameterizations is now below that including ecRad, and the radiation emulation itself is now comparably fast between the two, with variations between the number of GPUs used. Note that, for the ICON simulation corresponding to the runtime given in Table 2, the radiation parameterization is called every time-step and is solved at full resolution. The total runtime of all physics is thus dominated by the radiation runtime. The runtime ratio between ecRad and the machine learning models for the radiative process aligns with a similar ratio for the sum of all physical phenomena as seen in Table 2. The ratio for ecRad and the full physics may differ slightly due to due different workload on the Balfrin system used for these experiments. This experiment shows that ML-based radiation emulators running on GPUs are not per se faster than a highly optimized physics-based GPU-enabled solver like ecRad, which is written in Fortran with OpenACC compiler directives.

ML models size could be reduced further with, for example, automatic mixed precision (Carilli, 2024), where some operations are done with half precision instead of full precision, and dynamic quantization (*Dynamic Quantization*, 2024), which reduces the resolution of the model's weight. ICON and PyTorch are written in two different languages (Fortran with OpenACC and C++ with CUDA) which access the same GPU memory space when coupled together. It is yet unclear how both languages interact regarding data access and further profiling would be required to optimize how PyTorch should share the data access with the rest of the ICON code. This is however beyond the scope of this study. By comparison, ICON and ecRad are both written in Fortran with OpenACC directives. Note also that in (Ukkonen & Hogan, 2024), the authors restructured the ecRad code and improved its runtime performance by a factor of up to 12. The code's restructuring is designed for a CPU usage but the improved parallelism could benefit a GPU implementation of ecRad. Depending on its performance on GPUs, this optimized version of ecRad could outperform the ML models presented here.

4 Summary

In this two-part study, an ML emulation of the ecRad radiative transfer parameterization is built for the GPU-enabled ICON weather and climate model. In Part I, through a series of offline tests the most accurate ML model has been identified as a bidirectional recurrent NNs with long-short memory layers and additional physics-informed normalization of input and output features, as well as an additional heating rate related loss term in the objective function (Bertoli et al., 2024). In this work, Part II, a significant technical advancement is made by integrating the ML radiative transfer parameterization into ICON.

Since the air in the upper atmospheric layers is substantially less dense than lower layers, a small error in the flux profile results in a large heating rate error in the upper levels, which during long integration can cause spurious temperature trends near the model top. Model tops are known to often require additional tuning not seen during offline training. For example, Brenowitz and Bretherton (2019) removed the model top from training to stabilise the online performance of their ML-based convection scheme. To mitigate the aforementioned problem with high heating rates, which is easily overlooked in the offline testing, a simplified formula is used to calculate the fluxes in the damping layer of ICON near the model top, which reduces the heating rate error significantly and keeps the model free from any temperature drifts and very close to the original ecRad simulation for several simulated weeks. However, future research is needed to further improve the emulation at the top of the model, in particular for shortwave radiation, and to increase the reliability of the results at scales beyond weather forecasting. A potential way forward is to train an emulator that infers the TOA and surface level shortwave and longwave fluxes plus the heating rates on all levels within the atmosphere.

To seamlessly connect the Fortran code with the NNs implemented with Python, the Infero coupler from ECMWF was explored initially (Section 2.3). Infero requires that the ML model inputs and outputs are provided in the CPU memory. However, the version of ICON examined in this paper operates entirely on GPUs. Consequently, using Infero leads to needless data transfers between CPU and GPU memory, causing notable delays in the ML parameterization compared to ecRad. Note however that the next generation of hardware, like the Grace Hopper chips (NVIDIA, 2023) chosen for the next CSCS supercomputer Alps (CSCS et al., 2021), reduces the overhead of CPU-GPU copies and can even expose a shared CPU-GPU memory space to the user. This could make Infero more competitive, even in its current form. There appear to be no obstacles preventing the adaptation of Infero for complete GPU utilization, thereby eliminating the need for data transfers between hardware components. As such, the limitation of Infero exposed in this paper could potentially be nonexistent in future versions of this software. In this paper, to avoid CPU-GPU copies, the PyTorch-Fortran library (Alexeev, 2022) is adopted as an alternative solution. This approach enables direct processing of ML inputs and outputs on the GPU. The integrated system of GPU-enabled ICON, by means of OpenACC, and ML emulator implemented with PyTorch is deployed on the Piz Daint and Balfrin systems at CSCS, leveraging GPU capabilities for enhanced performance.

To the best of our knowledge, this is the first time that a full-fledged weather and climate model in combination with an ML-based parameterization developed in PyTorch has been run completely on GPUs. The performance gain compared with simulations using the original ecRad radiation solver depends critically on the complexity of the NNs architecture, and not all tested NNs are *per se* faster than the traditional physics-based code. Performance gains reported in past studies may stem from the fact that the emulated parameterisation was originally run on CPUs while the ML emulator was run on GPUs. Also in terms of memory consumption, we find that the memory footprint of ecRad is smaller compared to the RNN, albeit larger compared to the UNet architecture. We cannot rule out the possibility that a more sophisticated tuning of the NNs architectures would result in a higher speed-up, but this holds also true for the original radiation solver (Ukkonen & Hogan, 2024).

Open Research Section

The data were generated using the ICON climate model described in Prill et al. (2023). The software is available at `https://www.icon-model.org/`. The codes to reproduce the results of this paper will be made available in `https://gitlab.renkulab.io/deepcloud/rfe`. Data to reproduce results of this work will be hosted at ETH Research Collection `https://www.research-collection.ethz.ch/` (with a DOI) together

with the ICON runscript used to generate the full dataset. ETH Zurich's Research-Collection adheres to the FAIR principles and data is stored for at least 10 years.

Acknowledgments

This work was funded through a grant by the Swiss Data Science Center (SDSC grant C20-03). This research was supported by computation resources provided by the EXCLAIM project funded by ETH Zurich (CSCS project number d121). The Center for Climate Systems Modeling (C2SM) at ETH Zurich is acknowledged for providing technical and scientific support. Sebastian Schemm and Stefan Rüdisühli are supported by the European Research Council, H2020 European Research Council (grant no. 848698).

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., ... Zheng, X. (2015). *Tensorflow: Large-scale machine learning on heterogeneous systems.* Retrieved from `https://www.tensorflow.org/` (Software available from tensorflow.org)

Alexeev, D. (2022). *pytorch-fortran v0.4.* `https://github.com/alexeedm/pytorch-fortran`. GitHub.

Antonino Bonanni, T. Q., James Hawkes. (2022). *Infero 0.1.0.* `https://github.com/ecmwf/infero`. GitHub.

Bai, J., Lu, F., Zhang, K., et al. (2019). *ONNX: Open neural network exchange.* `https://github.com/onnx/onnx`. GitHub.

Bertoli, G., Ozdemir, F., Perez-Cruz, F., & Schemm, S. (2024). Revisiting machine learning approaches for short- and longwave radiation inference in weather and climate models, part I: Offline performance. *Journal of Advances in Modeling Earth Systems*.

Brenowitz, N. D., & Bretherton, C. S. (2019). Spatially extended tests of a neural network parametrization trained by coarse-graining. *Journal of Advances in Modeling Earth Systems*, *11*(8), 2728–2744. Retrieved from `https://doi.org/10.1029/2019ms001711` doi: 10.1029/2019ms001711

Carilli, M. (2024). *Automatic mixed precision.* `https://pytorch.org/tutorials/recipes/recipes/amp_recipe.html`. PyTorch.

Chevallier, F., Chéruy, F., Scott, N. A., & Chédin, A. (1998). A neural network approach for a fast and accurate computation of a longwave radiative budget. *Journal of Applied Meteorology*, *37*(11), 1385–1397. Retrieved from `https://doi.org/10.1175/1520-0450(1998)037<1385:annafa>2.0.co;2` doi: 10.1175/1520-0450(1998)037⟨1385:annafa⟩2.0.co;2

CSCS, NVIDIA, & Entreprise, H. P. (2021). *Swiss national supercomputing centre, hewlett packard enterprise and nvidia announce world's most powerful ai-capable supercomputer.* `https://www.cscs.ch/science/computer-science-hpc/2021/cscs-hewlett-packard-enterprise-and-nvidia-announce-worlds-most-powerful-ai-capable-supercomputer`.

Dynamic quantization. (2024). `https://pytorch.org/tutorials/recipes/recipes/dynamic_quantization.html`. PyTorch.

ECMWF. (2023). *Annual report 2022.* `https://www.ecmwf.int/sites/default/files/elibrary/2023/81359-annual-report-2022.pdf`.

Giorgetta, M. A., Sawyer, W., Lapillonne, X., Adamidis, P., Alexeev, D., Clément, V., ... Stevens, B. (2022). The ICON-A model for direct QBO simulations on GPUs (version icon-cscs:baf28a514). *Geoscientific Model Development*, *15*(18), 6985–7016. Retrieved from `https://gmd.copernicus.org/articles/15/6985/2022/` doi: 10.5194/gmd-15-6985-2022

Kedward, L. J., Aradi, B., Certik, O., Curcic, M., Ehlert, S., Engel, P., ... Vandenplas, J. (2022). The state of fortran. *Computing in Science I& En-*

gineering, *24*(2), 63–72. Retrieved from http://dx.doi.org/10.1109/MCSE.2022.3159862 doi: 10.1109/mcse.2022.3159862

Krasnopolsky, V. M., Fox-Rabinovitz, M. S., & Chalikov, D. V. (2005). New approach to calculation of atmospheric model physics: Accurate and fast neural network emulation of longwave radiation in a climate model. *Monthly Weather Review*, *133*(5), 1370–1383. Retrieved from https://doi.org/10.1175/mwr2923.1 doi: 10.1175/mwr2923.1

Lagerquist, R., Turner, D., Ebert-Uphoff, I., Stewart, J., & Hagerty, V. (2021). Using deep learning to emulate and accelerate a radiative-transfer model. *Journal of Atmospheric and Oceanic Technology*. Retrieved from https://doi.org/10.1175/jtech-d-21-0007.1 doi: 10.1175/jtech-d-21-0007.1

Liu, Y., Caballero, R., & Monteiro, J. M. (2020). RadNet 1.0: exploring deep learning architectures for longwave radiative transfer. *Geoscientific Model Development*, *13*(9), 4399–4412. Retrieved from https://doi.org/10.5194/gmd-13-4399-2020 doi: 10.5194/gmd-13-4399-2020

MPI-M, DWD, KIT, DKRZ, & C2SM. (2024). *ICON: Icosahedral nonhydrostatic weather and climate model.* Retrieved from https://icon-model.org/ (Software available at https://icon-model.org)

NVIDIA. (2016). *NVIDIA Tesla P100 GPU Accelerator.* https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/tesla-p100/pdf/nvidia-tesla-p100-datasheet.pdf. Author.

NVIDIA. (2022). *NVIDIA A100 Tensor Core GPU.* https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/a100/pdf/nvidia-a100-datasheet-nvidia-us-2188504-web.pdf. Author.

NVIDIA. (2023). *NVIDIA GH200 Grace Hopper Superchip.* https://resources.nvidia.com/en-us-grace-cpu/grace-hopper-superchip. Author.

OpenACC, version 3.2. (2022). https://www.openacc.org/.

Pal, A., Mahajan, S., & Norman, M. R. (2019). Using deep neural networks as cost-effective surrogate models for super-parameterized E3SM radiative transfer. *Geophysical Research Letters*, *46*(11), 6069–6079. Retrieved from https://doi.org/10.1029/2018gl081646 doi: 10.1029/2018gl081646

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., ... Lerer, A. (2017). Automatic differentiation in pytorch. In *Nips-w*.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Petty, G. W. (2006). *A first course in atmospheric radiation.* Madison, Wisconsin: Sundog Publishing.

Prill, F., Reinert, D., Rieger, D., & Zängl, G. (2023). ICON tutorial 2023: Working with the ICON model. *Deutscher Wetterdienst*. Retrieved from https://www.dwd.de/EN/ourservices/nwv_icon_tutorial/pdf_volume/icon_tutorial2023_en.pdf?__blob=publicationFile&v=3 doi: 10.5676/DWD_PUB/NWV/ICON_TUTORIAL2023

Roh, S., & Song, H.-J. (2020). Evaluation of neural network emulations for radiation parameterization in cloud resolving model. *Geophysical Research Letters*, *47*(21). Retrieved from https://doi.org/10.1029/2020gl089444 doi: 10.1029/2020gl089444

Ukkonen, P. (2022). Exploring pathways to more accurate machine learning emulation of atmospheric radiative transfer. *Journal of Advances in Modeling Earth Systems*, *14*(4). Retrieved from https://doi.org/10.1029/2021ms002875 doi: 10.1029/2021ms002875

Ukkonen, P., & Hogan, R. J. (2024). Twelve times faster yet accurate: A new state-of-the-art in radiation schemes via performance and spectral optimization. *Journal of Advances in Modeling Earth Systems*, *16*(1). Retrieved from http://dx.doi.org/10.1029/2023MS003932 doi: 10.1029/2023ms003932

Ukkonen, P., Pincus, R., Hogan, R. J., Nielsen, K. P., & Kaas, E. (2020). Accelerating radiation computations for dynamical models with targeted machine learning and code optimization. *Journal of Advances in Modeling Earth Systems*, *12*(12). Retrieved from `https://doi.org/10.1029/2020ms002226` doi: 10.1029/2020ms002226

Veerman, M. A., Pincus, R., Stoffer, R., van Leeuwen, C. M., Podareanu, D., & van Heerwaarden, C. C. (2021). Predicting atmospheric optical properties for radiative transfer computations using neural networks. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *379*(2194), 20200095. Retrieved from `https://doi.org/10.1098/rsta.2020.0095` doi: 10.1098/rsta.2020.0095

Appendix A Modification of the fluxes computation at the upper levels

Four different sets of constants β_k corresponding to the shortwave down, shortwave up, longwave down and longwave up fluxes are constructed. The constants β_k, $k = 0, \dots, H-1$, are given by

$$\beta_k = \frac{1}{N} \sum_{l=1}^{N} f_{k,l}/f_{H,l}, \tag{A1}$$

where the mean is taken over all atmospheric columns in the training set. The top atmospheric layers in ICON are Rayleigh damping layers, whose purpose is to attenuate oscillations reaching the top boundary. ICON is hence not designed to be accurate at such heights. In particular the ML emulator does not need to emulate perfectly ecRad in the damping layers. The goal of this strategy is to thus sacrifice flux accuracy at the top height levels for an increase in the stability of ICON. This strategy is not used on the random forest, which already provides accurate heating rates at the top levels. This is to be expected because the random forest prediction is average of flux profiles encountered in the training set.

For the upper levels, where the fluxes are approximated with Formula 1, the heating rates $\partial_t T_k^{rad}$ at level k for k in $0, \dots, 29$, are given by the following formula:

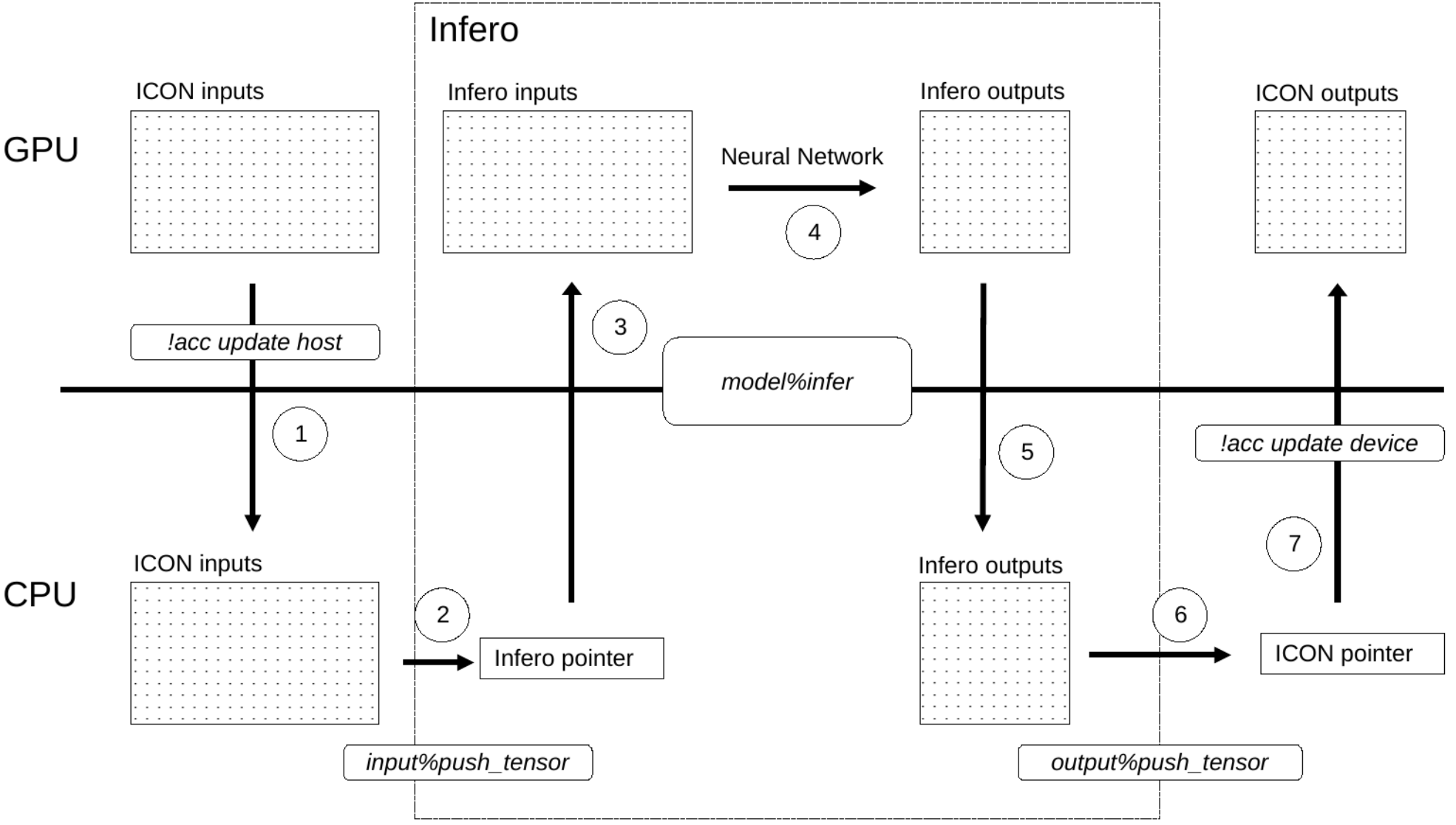
$$\begin{aligned}
\partial_t T_k^{rad} &= C_k(f_{k-1} - f_k) \\
&= C_k f_{30}(\beta_{k-1} - \beta_k) \\
&= C_k f_{30} \left(\frac{1}{N} \sum_{l=1}^{N} \frac{f_{k-1,l}}{f_{30,l}} - \frac{1}{N} \sum_{l=1}^{N} \frac{f_{k,l}}{f_{30,l}} \right) \\
&= \frac{1}{N} \sum_{l=1}^{N} \frac{f_{30}}{f_{30,l}} C_k(f_{k-1,l} - f_{k,l}) \\
&= \frac{1}{N} \sum_{l=1}^{N} \frac{f_{30}}{f_{30,l}} \frac{C_k}{C_{l,k}} \partial_t T_{k,l}^{rad},
\end{aligned} \tag{A2}$$

where C_k represent the effect of the air mass and humidity and is given by

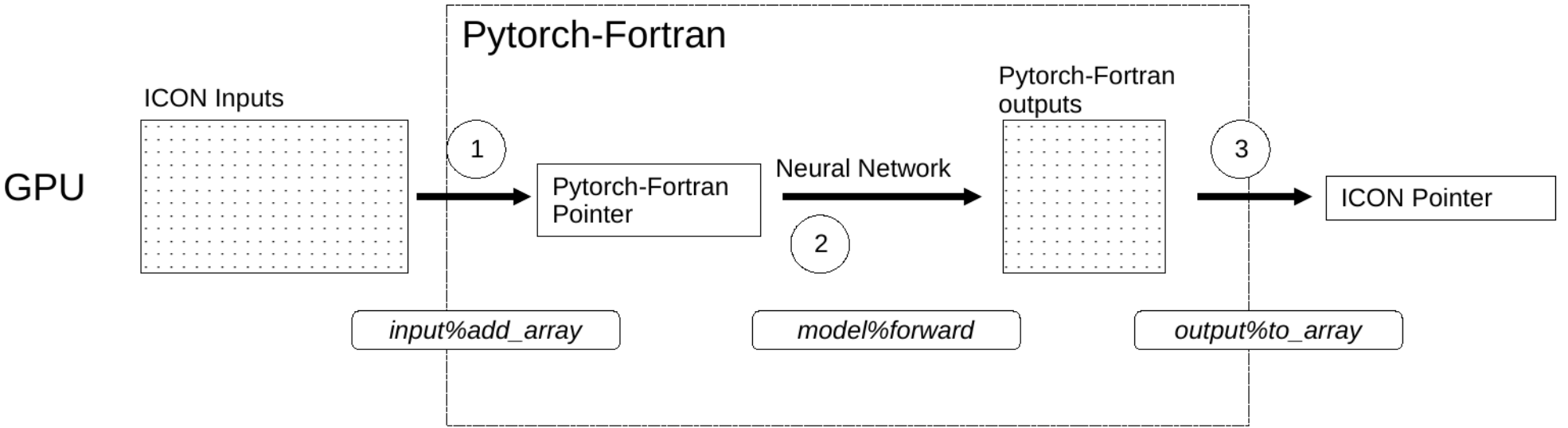
$$C_k = \frac{1}{m_k(c_d + (c_v - c_d)q_k)},$$

and where m_k an q_k are the air mass and specific humidity at height level k and c_v, c_d are the specific heat of water vapor and dry air at constant volume, assumed constant in ICON. Equation A2 shows that the heating rates obtained from the predicted fluxes are then the approximate weighted mean of heating rates observed during training. It is only an approximate weighted mean because $\frac{1}{N} \sum_{l=1}^{N} \frac{f_{30}}{f_{30,l}} \frac{m_{k,l}(c_d+(c_v-c_d)q_{k,l})}{m_k(c_d+(c_v-c_d)q_k)}$ is not equal to 1 in general. Furthermore, no vertical derivative of the predicted fluxes appear in Equation A2. Additionally, the inverse of the air mass $1/m_k$ is multiplied by the air mass $m_{k,l}$ of observed atmospheric columns during training. This reduces the sensitivity of the heating rates to the fluxes prediction.

Infero schematic.

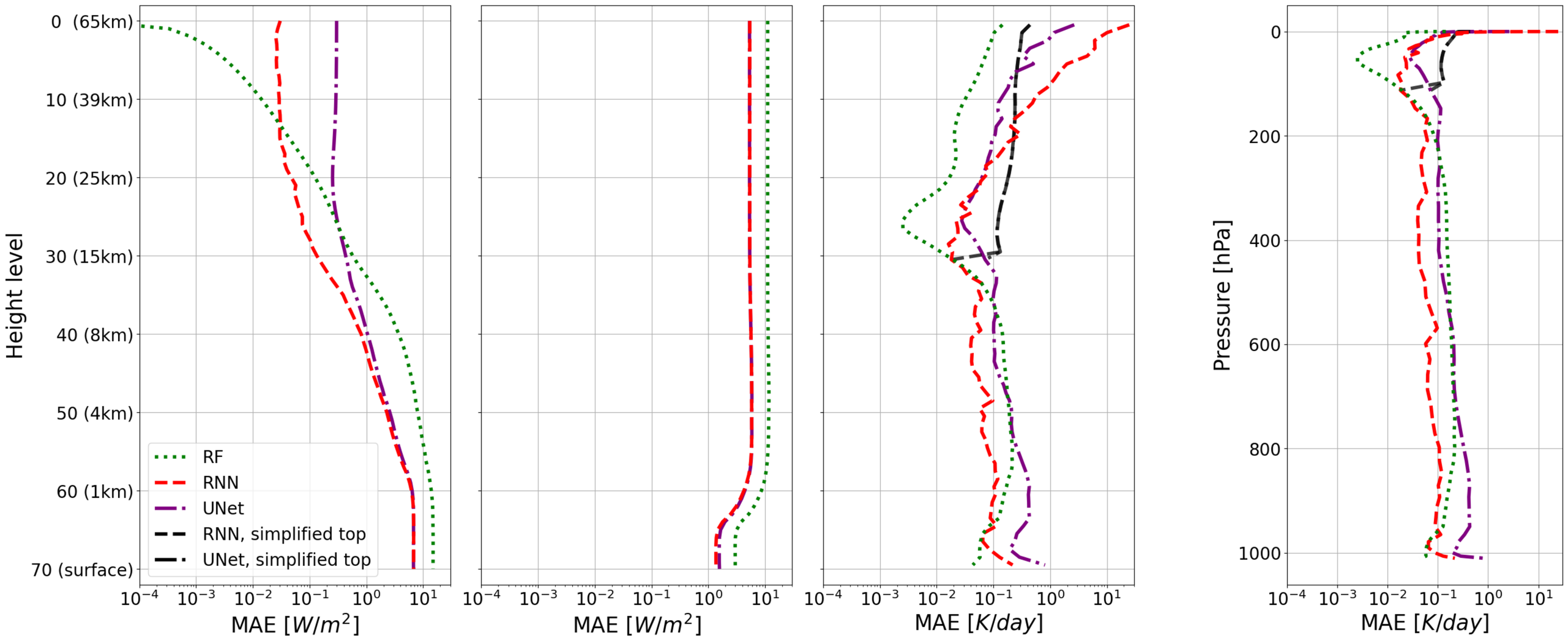

Infero
GPU
CPU
ICON inputs
Infero inputs
Neural Network
Infero outputs
ICON outputs
!acc update host
model%infer
!acc update device
1
2
3
4
5
6
7
ICON inputs
Infero pointer
Infero outputs
ICON pointer
input%push_tensor
output%push_tensor

PyTorch-Fortran schematic.

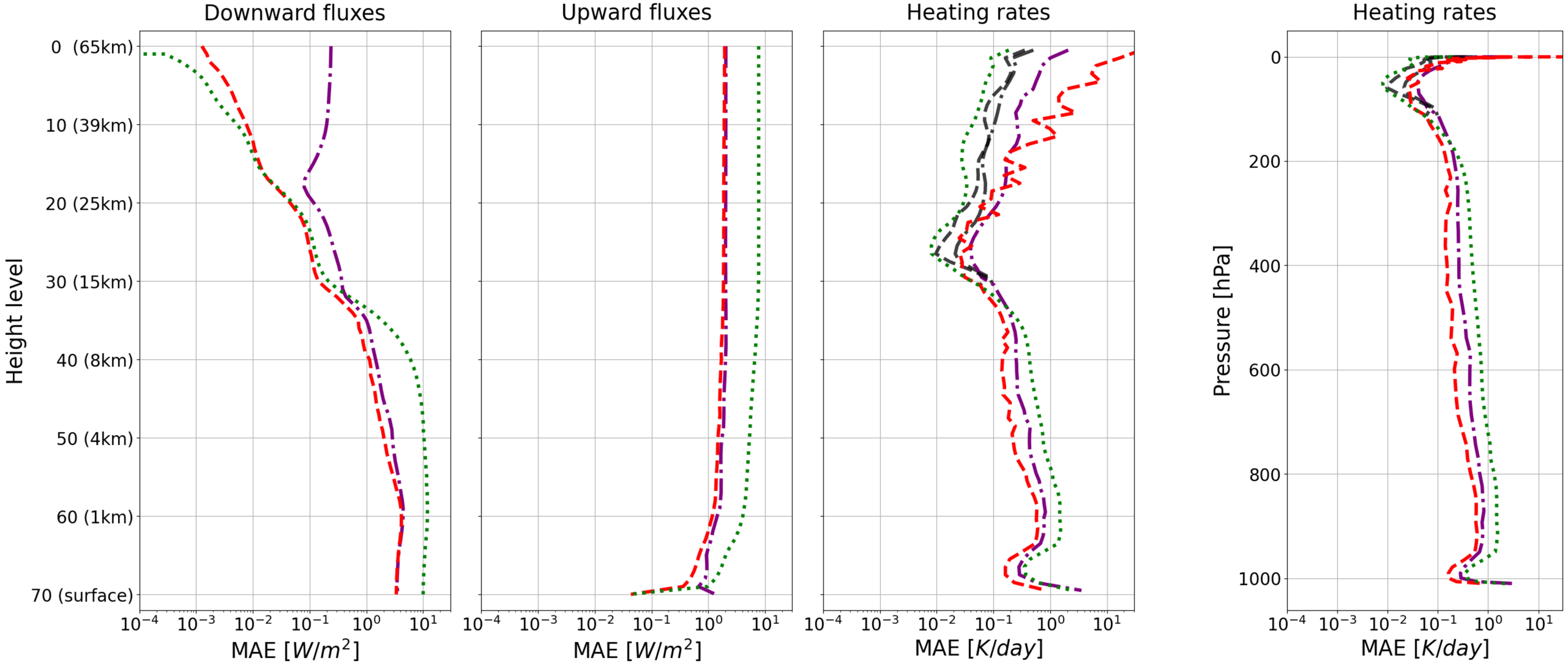

GPU
Pytorch-Fortran
ICON Inputs
1
Pytorch-Fortran
Pointer
Neural Network
2
Pytorch-Fortran
outputs
3
ICON Pointer
input%add_array
model%forward
output%to_array

MAE vs height.

Shortwave



Longwave



Mean temperature.

a) Top of atmosphere

Temperature [K]

159.0
158.5
158.0
157.5

0 3 6 9 12 15 18 21

time [days]

b) 1km above surface

Temperature [K]

279.0
278.5
278.0
277.5

0 3 6 9 12 15 18 21

time [days]

c) Surface

RF RNN UNet ecRad

Temperature [K]

287.0
286.5
286.0
285.5

0 3 6 9 12 15 18 21

time [days]

heating rates prediction.

a) ecRad
b) RNN
c)
d)
e)
f)
g)
h)
+2 d
+4 d
+6 d
10 d mean
Pressure (hPa)
Latitude
100
200
400
600
800
1000
-50
0
50
10
8
6
4
2
0
−2
−4
−6
−8
−10
K/day