Efficient and stable coupling of the SuperdropNet deep-learning-based cloud microphysics (v0.1.0) with the ICON climate and weather model (v2.6.5)

Caroline Arnold[1,2,3], **Shivani Sharma**[2,3,4], **Tobias Weigel**[1,2,3], **and David S. Greenberg**[2,3]

[1]German Climate Computing Center (DKRZ), Hamburg, Germany
[2]Helmholtz-Zentrum Hereon, Geesthacht, Germany
[3]Helmholtz AI, Munich, Germany
[4]International Max Planck Research School on Earth System Modelling, Hamburg, Germany

Correspondence: Caroline Arnold (arnold@dkrz.de) and Shivani Sharma (shivani.sharma@hereon.de)

Abstract. Machine learning (ML) algorithms can be used in Earth system models (ESMs) to emulate sub-grid-scale processes. Due to the statistical nature of ML algorithms and the high complexity of ESMs, these hybrid ML ESMs require careful validation. Simulation stability needs to be monitored in fully coupled simulations, and the plausibility of results needs to be evaluated in suitable experiments.

We present the coupling of SuperdropNet, a machine learning model for emulating warm-rain processes in cloud microphysics, with ICON (Icosahedral Nonhydrostatic) model v2.6.5. SuperdropNet is trained on computationally expensive droplet-based simulations and can serve as an inexpensive proxy within weather prediction models. SuperdropNet emulates the collision–coalescence of rain and cloud droplets in a warm-rain scenario and replaces the collision–coalescence process in the two-moment cloud microphysics scheme.

We address the technical challenge of integrating SuperdropNet, developed in Python and PyTorch, into ICON, written in Fortran, by implementing three different coupling strategies: embedded Python via the C foreign function interface (CFFI), pipes, and coupling of program components via Yet Another Coupler (YAC).

We validate the emulator in the warm-bubble scenario and find that SuperdropNet runs stably within the experiment. By comparing experiment outcomes of the two-moment bulk scheme with SuperdropNet, we find that the results are physically consistent and discuss differences that are observed in several diagnostic variables.

In addition, we provide a quantitative and qualitative computational benchmark for three different coupling strategies – embedded Python, coupler YAC, and pipes – and find that embedded Python is a useful software tool for validating hybrid ML ESMs.

1 Introduction

Machine learning (ML) is increasingly used in Earth system models (ESMs) to emulate sub-grid-scale processes that are typically parameterized or neglected due to their high computational cost (Christensen and Zanna, 2022; Dueben et al., 2021; Irrgang et al., 2021; Gentine et al., 2018). ML algorithms are statistical algorithms that are trained on data. Neural networks are a widely used class of ML algorithms. They contain trainable parameters (i.e., the weights and biases) that are learned from data by minimizing a cost function. The trained algorithm can then be used for inference (i.e., application on unseen data of the same kind). When sub-grid-scale processes are replaced by ML algorithms, the improvement can aim at speeding up the overall simulation by emulating the existing parameterization. This was first established using neural networks to emulate long-wave radiative transfer (Chevallier et al., 2000; Krasnopolsky et al., 2005). Recent examples include the emulation of the gravity wave drag (Chantry et al., 2021), the cloud microphysics (Brenowitz et al., 2022), the ocean in a coupled climate model (Son-

newald et al., 2021), and the cloud radiative effects (Meyer et al., 2022).

Other studies aim to improve the overall description of the Earth system by providing a better parameterization. ML algorithms can be trained on high-resolution ESM output or even on separately simulated processes to emulate resolved processes in a low-resolution simulation, for example, for gravity waves (Dong et al., 2023), cloud cover parameterizations (Grundner et al., 2022), general parameterizations (Brenowitz and Bretherton, 2018), sub-grid-scale momentum transport (Yuval and O'Gorman, 2023), effects of cloud-resolving simulations (Rasp et al., 2018), ozone distributions (Nowack et al., 2018), and radiative transfer (Belochitski and Krasnopolsky, 2021).

Many parameterizations in ESMs, such as the convective parameterizations, can be removed at higher resolutions if the process can be completely resolved. On the other hand, some others would need to be parameterized even for 1 km scale weather models. Cloud microphysical processes fall in this category. Processes dealing with the droplet interactions that lead to precipitation are lumped together and referred to as cloud microphysical processes. Due to high particle counts even at small grid sizes and our incomplete understanding of processes that occur at a molecular level in clouds (Morrison et al., 2020), we cannot expect cloud microphysical parameterizations to become obsolete in the near future for high-resolution models.

The parameterization of these processes suffers from a unique accuracy–speed trade-off. The most accurate droplet-based Lagrangian schemes such as the super-droplet method (Shima et al., 2009) are computationally expensive. The commonly used two-moment bulk schemes represent the complex particle size distributions as only the first two moments, referring to the total droplet concentration and the total water content of the hydrometeors. For modeling the droplet collisions in a warm-rain scenario, ICON uses the well-studied two-moment bulk scheme developed in Seifert and Beheng (2001). To bridge this gap and to make the use of more complex microphysical schemes feasible within operational models, a data-driven approach can be employed. Here, we present the integration of SuperdropNet (Sharma and Greenberg, 2024), an ML algorithm for emulating warm-rain processes in cloud microphysics, into ICON v2.6.5. SuperdropNet is trained on zero-dimensional box model super-droplet simulations from McSnow v1.1.0 (Brdar and Seifert, 2018), a super-droplet-based cloud microphysics model, in a warm-rain scenario and replaces the warm-rain processes in the two-moment scheme available in ICON v2.6.5 (Seifert and Beheng, 2006).

Due to the statistical nature of ML algorithms and the complex nonlinear interactions in ESMs, hybrid systems of numerical ESMs and ML algorithms require careful validation and verification (Dueben et al., 2022; Brenowitz and Bretherton, 2019). Stand-alone ML algorithms are first trained on a dataset and then validated on a holdout test dataset that is not seen during training. This test set is within the distribution of the training data. When an ML algorithm is coupled with an ESM, it may encounter conditions outside of the range of the training data, and the required extrapolation could lead to instabilities (Yuval et al., 2021). Thus, the so-called *offline* performance of an ML algorithm is often not a good indicator of its *online* performance (Brenowitz et al., 2020b; Rasp, 2020). Stability is a major concern when introducing ML emulators into ESMs. It can be improved by adapting the training procedure (Qu and Shi, 2023; Brenowitz et al., 2020a; Rasp, 2020; Brenowitz and Bretherton, 2018) or by fulfilling physical constraints in the network architecture (Beucler et al., 2021; Yuval et al., 2021). Careful validation setups can help the scientific community to build trust in so-called black box ML algorithms (McGovern et al., 2019).

To avoid devoting resources to the development of ML algorithms that fail in contact with reality, we encourage incorporating online testing at an early stage. ML algorithms are developed iteratively, and new versions should be tested quickly in their final place of application in the Earth system model.

The popular software libraries for ML algorithm development, such as PyTorch (Paszke et al., 2019), Keras (Chollet et al., 2023), or TensorFlow (Abadi et al., 2016), are based on the Python language. On the other hand, ICON is written in Fortran. Online testing requires either rewriting the ML emulator in Fortran or integrating the two programming languages with one another (Brenowitz and Bretherton, 2019). Since ML algorithm development is an iterative process, frequent rewrites of the ML algorithm would be required in the former case. In order to save developer resources, we recommend coupling Python and Fortran at least during the stage of algorithm development.

In Sect. 2.1, we introduce the warm-bubble scenario, which serves as a test case for SuperdropNet. The ML algorithm itself is described in Sect. 2.3. Different strategies for integrating SuperdropNet into ICON are discussed in Sect. 3. The results and the impact of SuperdropNet on atmospheric processes and prognostic variables are presented in Sect. 4.2. A computational and qualitative benchmark of three different strategies is included in Sect. 4.3.

2 Methods

2.1 Warm-bubble scenario

We validate SuperdropNet in the warm-bubble scenario, a test case for cloud microphysics available in ICON v2.6.5. It describes an atmosphere temperature profile with a warm air bubble at the bottom that rises vertically. The test case operates on a torus grid. This grid is created by a domain of 22×20 cells where periodic boundary conditions are applied in x and y directions. The horizontal resolution is 5 km, and there are 70 vertical levels in the z direction. The simula-

tion time step is 20 s, with a total simulation time of 120 min. The experiment is computationally lightweight and runs on a single compute node. We test SuperdropNet in a warm atmosphere with no ice particle formation, as well as in a mixed-phase and a cold atmosphere that both allow for ice formation. All simulation parameters are summarized in Table 1. We transport the tracers required for two-moment cloud microphysics (i.e., the first and second moment of the hydrometeor cloud water, cloud ice, rain, snow, graupel, and hail).

2.2 Two-moment bulk scheme for cloud microphysics

In our test case, a two-moment bulk scheme is employed to compute the number concentration and total mass for all hydrometeors involved. In ICON, the two-moment bulk scheme used for warm-rain cloud microphysics is based on Seifert and Beheng (2001). To account for collision–coalescence, the number concentration and total mass for both cloud and rain are determined by calculating the rates of collision–coalescence processes, including autoconversion, accretion, and self-collection. Here, autoconversion refers to the process by which cloud droplets coalesce to form rain droplets, while accretion accounts for collisions between rain and cloud droplets. Self-collection rates for cloud and rain droplets account for collisions that do not convert cloud droplets into rain. These process rates rely solely on the droplets themselves and are subsequently utilized to update the bulk moments for the following time step using a set of ordinary differential equations.

2.3 SuperdropNet cloud microphysics model

SuperdropNet is a machine learning emulator for super-droplet simulations in a warm-rain scenario. It is a neural network consisting of fully connected layers and is trained to predict updates of the bulk moments for cloud and rain over different droplet size distributions. SuperdropNet is detailed in Sharma and Greenberg (2024); therefore, we will provide only a brief summary of the training procedure here.

The super-droplet simulations used for training are generated with McSnow (Brdar and Seifert, 2018). In Brdar and Seifert (2018), McSnow is used for simulating ice particles, while in Seifert and Rasp (2020), it simulated a warm-rain scenario. Similarly to Seifert and Rasp (2020), the training data for SuperdropNet are generated in a warm-rain scenario that describes only the conversion of cloud droplets into rain in a dimensionless control volume. As super-droplet simulations are stochastic in nature, we use multiple realizations of simulations to train SuperdropNet. Hence, given a set of initial conditions, SuperdropNet is completely deterministic in nature, and the bulk moments estimated by it are the equivalent of averaged super-droplet simulations (Sharma and Greenberg, 2024). The microphysical processes accounted for are accretion, autoconversion, and self-collection of rain and cloud droplets. In ICON, the droplet collisions corre-
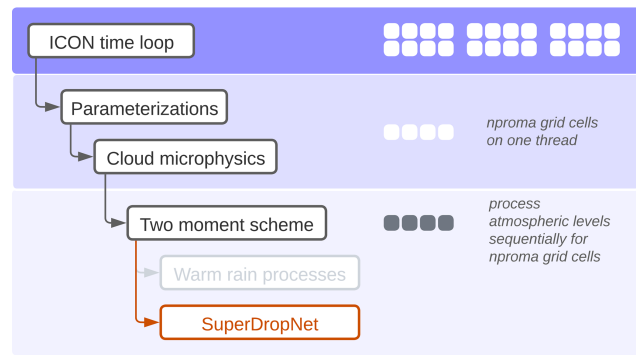

Figure 1. We replace the warm-rain processes (gray) with a call to SuperdropNet (orange). At this point, each thread has access to an *ik* slice, a specific representation in the cloud microphysics parameterization that corresponds to one atmospheric level for one block of grid cells.

sponding to warm-rain processes are treated in a separate module where the process rates for accretion, autoconversion, and self-collection of rain and cloud droplets are calculated. The parameterization scheme is localized; i.e., the process rates calculated for a grid cell depend only on the rain and cloud moments which correspond to that grid cell. Other microphysical processes and the vertical transport are accounted for in separate modules, which implies that the parameterization in ICON is structured in such a way that all individual grid points can be considered zero-dimensional boxes. Thus, the parameterization setup for droplet collisions in ICON mimics the training data for SuperdropNet. This justifies the choice of using a test scenario in ICON for the online coupling and testing of SuperdropNet.

Note that only the warm-rain processes are replaced with SuperdropNet. In a cold atmosphere, SuperdropNet can still be coupled with ICON, but since warm-rain processes are not relevant there, including SuperdropNet is expected not to change the experiment results.

2.4 ICON program flow

To illustrate at which point of program execution ML ESM coupling becomes necessary, we show the flowchart for a single ICON time step in Fig. 1, focusing only on the steps relevant to our application. Starting from the general ICON time loop, where the full grid information is available, we enter the cloud microphysics parameterization. At this point, a given thread has access to one block of grid cells with block length nproma, and all threads work in parallel. The two-moment scheme has its own grid representation, called *ik* slices, where the block of grid cells is again divided by atmospheric levels. In our experiment, we simply replace the warm-rain processes with a call to SuperdropNet, which provides updated moments for cloud and water droplets.

Since the call to the ML component is not at the grid level but operates on *ik* slices far down in the nested structure of

Table 1. Experiment parameters for the warm-bubble, mixed-phase bubble, and cold-bubble test cases. Note that t_{dyn} and t_{2mom} reflect the time step used for training SuperdropNet.

Parameter	Description	Warm bubble	Mixed-phase bubble	Cold bubble
L_D	Torus domain length		5000 m	
t_{dyn}	Dynamical time step		20 s	
t_{2mom}	Two-moment scheme time step		20 s	
z_{lev}	Atmospheric levels		70	
p_{srfc}	Surface pressure		1013.25 hPa	
T_0	Cold point of the atmosphere	303.15 K	273.15 K	268.15 K
γ_0	Vertical temperature lapse rate		0.006 K m^{-1}	0.009 K m^{-1}
z_0	Altitude up to which γ_0 applies		3000 m	4000 m
γ_1	Lapse rate above z_0		0.00001 K m^{-1}	0.0001 K m^{-1}
$T_{perturb}$	Temperature perturbation		10 K	5 K
ϕ_{bg}	Background relative humidity		0.7	
ϕ_{mx}	Maximum relative humidity		0.9	0.95
ξ	Half-width of temperature perturbation in x		12 500 m	
ζ	Half-width of temperature perturbation in z		200 m	250 m
x_0	Center of temperature perturbation in x		0 m	

the ICON program flow, we need to call SuperdropNet several times per time step – once for each block of grid cells and once for each atmospheric level. Note that saturation adjustments and evaporation are handled outside of the parts of the ICON code replaced by SuperdropNet.

3 Integrating SuperdropNet in ICON

There are several ways to integrate Python machine learning components into Fortran code (Partee et al., 2022). Based on a pre-selection of suitable methods, we have implemented three strategies, so-called Fortran–Python bridges. For convenience, we add a name list to ICON that allows the selection of the coupling strategy. We perform the experiment with all three methods on the German Climate Computing Center (DKRZ) Levante system. Levante is a BullSequana XH2000 supercomputer with 3042 compute nodes using the third generation of AMD EPYC CPUs (Milan) with 128 cores per node, NVIDIA A100 GPUs, and a 130-petabyte DDN filesystem. The nodes are connected to a Mellanox InfiniBand HDR100 fabric.

3.1 Embedding Python as a dynamic library

Using the techniques in Brenowitz (2023), we develop a dynamic library based on Python code. The library is generated using the C foreign function interface (CFFI) (Rigo and Fijalkowski, 2018) and is linked to ICON at compile time. At runtime, Python code is executed from the library. Employment of CFFI results in Python and Fortran sharing their address space and hence passing memory pointers is sufficient for accessing the same data. Jobs are run in a homogeneous setting, with Python code executed on the same CPU compute node as ICON.

3.2 Using the coupling software YAC

Yet Another Coupler (YAC) (Hanke et al., 2023, 2016) is commonly used to couple different ICON components, e.g., atmosphere, ocean, and I/O. YAC provides Python bindings so that external Python programs can be coupled with little effort with ICON.

YAC requires a definition of fields that are to be exchanged and an exchange schedule that cannot be below the time step of ICON. For the warm-bubble scenario, we set the block length to the number of grid cells (880) and define two exchange fields per atmospheric level, one for the ICON-to-Python exchange and one for the reverse exchange. This yields a total of 140 fields that are exchanged at each time step. A smaller block length would require the developer to define more exchange fields such that bulk moments in each grid cell can be exchanged at every time step.

Data transfer is building on Message Passing Interface (MPI) routines that are integrated in YAC. This offers the flexibility to use heterogeneous jobs – i.e., running ICON on CPU nodes and ML inference on GPU nodes. Due to the current limitations of the scheduling software employed in the DKRZ Levante system, it was not possible to schedule simulations that span the CPU and the GPU partition of the system. Thus, we were not able to test the performance in a heterogeneous setting. With ICON shifting to GPUs, we foresee that in the future homogeneous jobs will be run on GPU nodes.

3.3 Pipes

We implemented a coupling between n ICON processes and 1 Python process running on the same node using FIFO (first-in–first-out) pipes. The first ICON MPI rank on the node will spawn a separate Python process that runs a worker script. Each rank also creates two pipes, one for each direction of communication (input and output to the Python worker). The worker iterates over all input pipes, performs the warm-rain calculation on data that are available, and writes results back to the corresponding ICON process via its output pipe.

While this solution does not incur the potential overhead of using MPI to communicate locally, it is not a full shared-memory solution exclusively relying on pointers. The corresponding extensions to ICON and the Python worker script are optimized to do as few memory copies as possible, though naturally some copying cannot be avoided when interacting with the pipes. As FIFO pipes only work on a local node, no cross-node setups are possible, such as running ICON and Python on different types of nodes (CPU, GPU). As the Python worker runs as a separate process on a dedicated core, the number of cores available to ICON is also marginally reduced by one.

3.4 Other methods

We note that the selection of methods in Sect. 3.1–3.3 by no means encompasses all the available tools, and here we summarize the alternatives to the best of our knowledge.

Four software libraries developed at the European Centre for Medium-Range Weather Forecasts (ECMWF; Bonanni et al., 2022), the Cambridge Institute for Computing in Climate Science (Elafrou et al., 2023), NVIDIA (Alexeev, D., 2023), and Tongji University (Mu et al., 2023) address ML inference directly by exposing the TensorFlow and PyTorch APIs for Fortran, respectively. This adds the benefit of not requiring a Python runtime environment at the time of execution. Since we require flexibility to use Python code beyond ML inference and data exchange is done here via RAM comparable to the approach described in Sect. 3.1, we did not investigate these libraries further.

During development, we noted that integrating SmartSim (Partee et al., 2022) would require a rewrite of the ICON startup routine that is beyond the scope of this project. On a similar note, the coupling routines developed for the open-source Weather Research and Forecasting (WRF) model, WRF–ML, cannot easily be adjusted to work with ICON (Zhong et al., 2023).

The Fortran–Keras bridge (Ott et al., 2020) allows for ML inference in Fortran based on ML algorithms developed in the Keras framework. This limits flexibility, since only those network layers and functionalities supported by the library can be used. On a similar note, the implementation of the ML algorithm in neural-fortran (Curcic, 2019) is contingent on the library, and the Fortran Inference-Engine (Berkeley Lab, 2023) is restricted to feed-forward neural networks. We chose to forego these methods since we desire the flexibility to use any novel PyTorch development without depending on their integration into an external library.

4 Results

4.1 Experiment description

Using the three coupling techniques described in Sect. 3.1–3.3, we integrate SuperdropNet in ICON. The experiment results are the same since the same network is called, but the impact on computational performance is different. We run the warm-bubble scenario and the cold-bubble scenario, both with a representation of warm-rain processes using SuperdropNet and the existing two-moment bulk scheme in the two-moment cloud microphysics module.

We compare the effect of replacing warm-rain processes with SuperdropNet with the experiment outcome in Sect. 4.2. In Sect. 4.3, we compare the impact on computational performance that is incurred by integrating SuperdropNet for all three coupling techniques.

4.2 Comparison of the two-moment bulk scheme and SuperdropNet

4.2.1 Rain rates

Figure 2a shows the grid-averaged rain rate in the warm-bubble scenario which is derived from warm-rain processes using ICON's two-moment bulk cloud microphysics, with a comparison to SuperdropNet microphysics. Since SuperdropNet was trained on particle-based simulations that avoid certain statistical approximations of two-moment bulk schemes, we do not expect the rain rates in both scenarios to match. Due to the experimental setup, it is not possible to identify with certainty which model produces the more accurate rain rates. We do note, however, that SuperdropNet yields physically plausible rain rates. The rain rate obtained using SuperdropNet evolves in a predictable way; that is, there is no rain at the beginning of the simulation, and then it eventually builds up to a peak and then slowly rescinds. At the end of the simulation, the rain rate is 0 for both simulations. No negative values are observed, and the coupling with SuperdropNet does not result in significant divergence from the simulation. This emphasizes that SuperdropNet is stable over longer simulation runs and overall behaves like a realistic ML-based emulator for droplet collisions. One of the key differences in the evolution of the rain rate with the two different parameterizations is that the onset of rain is slightly delayed in the case of SuperdropNet coupling, which indicates a slower conversion of cloud droplets to rain droplets.

As a sanity check, we perform the cold-bubble experiment using both the two-moment bulk scheme and SuperdropNet for the warm-rain processes. In this scenario, warm-
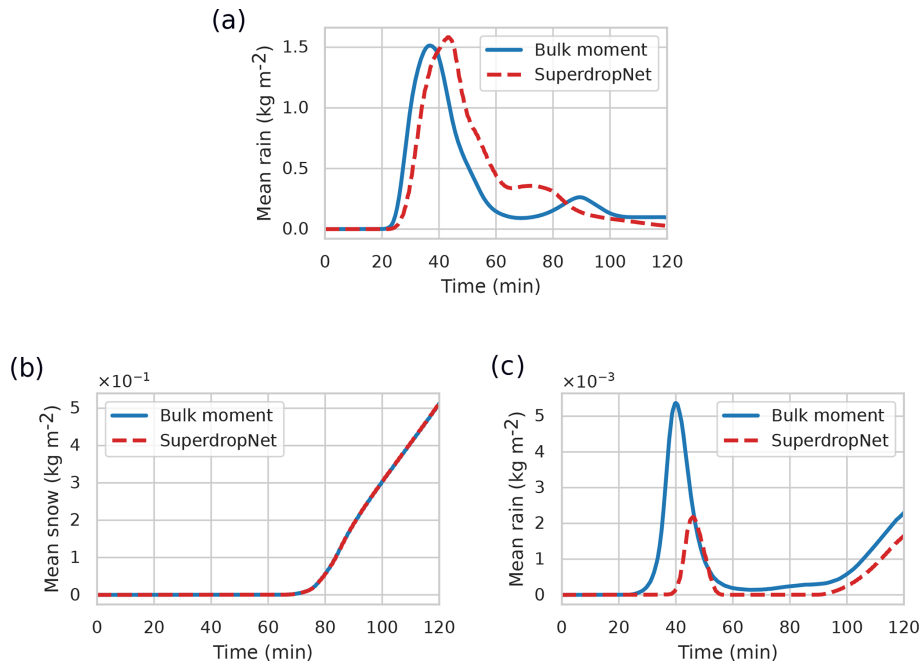
Figure 2. Grid-averaged quantities for the two-moment bulk scheme and SuperdropNet for the **(a)** warm-bubble scenario, **(b)** cold-bubble scenario, and **(c)** mixed-phase scenario.

rain processes are not relevant for the cloud microphysics, and we expect that including SuperdropNet does not affect processes with frozen particles. Figure 2b shows the grid-averaged snow rate.

Both schemes show identical snow rates, which confirms that there are no undesired side effects from coupling SuperdropNet when the conditions in the atmosphere do not allow for warm-rain processes.

We also perform a mixed-phase experiment with the same setup. In this scenario, both frozen and non-frozen particles occur in the atmosphere. Figure 2c shows the grid-averaged rain rate. The grid-averaged values for all hydrometeors are included in the Appendix. In this case, coupling to SuperdropNet significantly drops the total rain rate. Since the total water mass remains conserved in ICON, the suppression of rain formation leads to increased ice, cloud, and snow formation (Fig. A1). In ICON, the warm-rain processes are simulated before other processes such as ice nucleation, ice self-collection, and snow melting. Hence, SuperdropNet's effect on decreasing rain formation is subsequently reflected in the excess of other hydrometeors.

4.2.2 Heat transport fluxes

Figure 3 shows the grid-averaged evaporative fluxes as it evolves with time during the coupled warm-bubble simulation in ICON. While in the beginning both the two-moment bulk scheme and SuperdropNet produce similar fluxes, the values diverge approximately after 30 min, which corresponds to the onset of rain. This difference between the magnitude of fluxes is also reflected in the evolution of winds during the simulation. Winds are the primary source of energy transport, and Fig. 4 shows the evolution of meridional winds in the simulation. After approximately 40 min, which roughly corresponds to the end of the first rainfall with both parameterizations, the wind patterns are markedly different for the two-moment bulk and the SuperdropNet parameterizations. The winds appear much stronger in case of the two-moment bulk parameterization across the vertical column. The reduced magnitude of winds in SuperdropNet coupling corresponds to reduced heat fluxes in Fig. 3.

Figure 5 shows the vertical profile of specific humidity at different time steps during the simulation. For the first 40 min of the experiment, both parameterization schemes produce similar specific humidity profiles, but this changes during the later part of the simulation. Close to the surface, it can be observed that the two-moment bulk parameterization produces a stronger humidity gradient in comparison to SuperdropNet. This difference in the specific humidity gradient possibly results in a higher evaporative flux for the two-moment bulk coupling than the SuperdropNet coupled simulation.

Similarly, in Fig. 6, the evolution of mean rain droplet mass ($\overline{X}_r$) is shown. The differences in $\overline{X}_r$ close to the surface, as calculated using the two-moment bulk scheme compared to SuperdropNet, become more visible after 40 min. In general, with the two-moment bulk parameterization $\overline{X}_r$, values are higher than those with the super-droplet parameterization close to the surface. Since the evaporative flux is proportional to the mean rain mass, higher $\overline{X}_r$ in two-
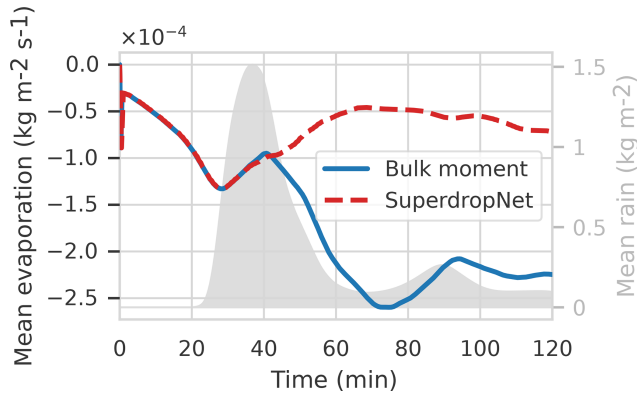
Figure 3. Grid-averaged evaporative heat fluxes for the two-moment bulk scheme used in ICON two-moment cloud microphysics and for SuperdropNet. The gray area shows the grid-averaged rain obtained using the two-moment bulk scheme (see Fig. 2a). High negative values indicate a larger amount of heat transfer.

moment bulk coupling results in higher heat fluxes. Throughout the vertical column, the SuperdropNet parameterization usually corresponds to lower $\overline{X}_r$, except at the 40 min time step, where the high $\overline{X}_r$ value near the 3000 m height also corresponds to a higher amount of the vertically integrated rain rate, as seen in Fig. 2a.

Note that the warm-bubble scenario in ICON is highly sensitive to the tiniest fluctuations in the assumptions made for cloud microphysics parameterization. Since many other complex phenomena are simplified and the focus is only on the formation and dissipation of a single cloud, small deviations in the approximation of the cloud and rain moments lead to changes in other diagnostic variables that can accumulate over time.

4.3 Computational performance upon including SuperdropNet

4.3.1 Benchmark

We ran the experiments on the Levante computing system at the German Climate Computing Center on compute nodes equipped with two AMD 7763 CPUs with a total of 128 cores and 256 GB of main memory. The nodes are connected with a Mellanox InfiniBand HDR100 fabric.

SuperdropNet provides a significant speedup by emulating processes that would otherwise be computationally infeasible to include in ICON, but when adding a Python component to the existing highly optimized Fortran code, we expect an impact on computational performance. Table 2 summarizes the total time spent in the calculation of the two-moment scheme in the ICON warm-bubble scenario, using the two-moment bulk scheme and SuperdropNet coupled with ICON using three different coupling strategies. The fastest time-to-solution is provided by including SuperdropNet via embed-

Table 2. Time spent in the two-moment scheme in the ICON warm-bubble scenario, using the two-moment bulk scheme (Fortran), and SuperdropNet (PyTorch) coupled with ICON. Note that by coupling SuperdropNet with ICON, we introduce a scheme that would be computationally intractable for cloud microphysics in standard numerical simulations. A direct comparison of runtimes is therefore not possible.

Experiment		t_{2mom} (s)	Nodes
Two-moment bulk scheme (Fortran)		1.25	1
SuperdropNet (PyTorch)	CFFI	24.1	1
	Pipes	62.6	1
	YAC	49.5	2

ded Python – i.e., the C foreign function interface (CFFI) (Sect. 3.1). Coupling SuperdropNet via YAC (Sect. 3.2) increases the relative runtime by a factor of 2 compared to embedded Python. Note that when coupling with YAC, the ICON and the Python main program run on two different computational nodes, which doubles the amount of computational resources required for the experiment. In the current configuration, YAC can only be used when the block length is equal to the grid size, which limits us to small experiments like the bubble scenarios. Coupling SuperdropNet and ICON using pipes is almost 3 times slower than embedded Python. On a qualitative note, implementing the coupling via pipes requires changes to core components of ICON beyond the cloud microphysics parameterization and may be an additional challenge for ML developers.

We note that coupling a super-droplet model directly with our test case in ICON is extremely challenging. ICON represents the warm-rain processes as bulk moments, while Mc-Snow represents them as droplet distributions. For an ideal benchmark simulation, we would need to completely overhaul the current representation of cloud microphysics processes in ICON and represent them as super-droplets for a two-way coupling. At the time of conducting this research, ICON did not allow for the representation of cloud microphysical processes as super-droplets, mainly because doing so would be computationally expensive. This is an active area of research, but as of now, it remains a work in progress, which makes SuperdropNet a cheaper data-driven alternative to the super-droplet simulations.

4.3.2 Detailed evaluation of coupling with embedded Python

We now turn to the fastest coupling scheme, embedded Python, and investigate the contribution of the individual steps to the total runtime. By including SuperdropNet, we incur the computational cost of data exchange and of machine learning inference. Table 3 summarizes the contribution of the individual parts, measured with a block length of nproma = 44 grid cells using the ICON timer module. ICON

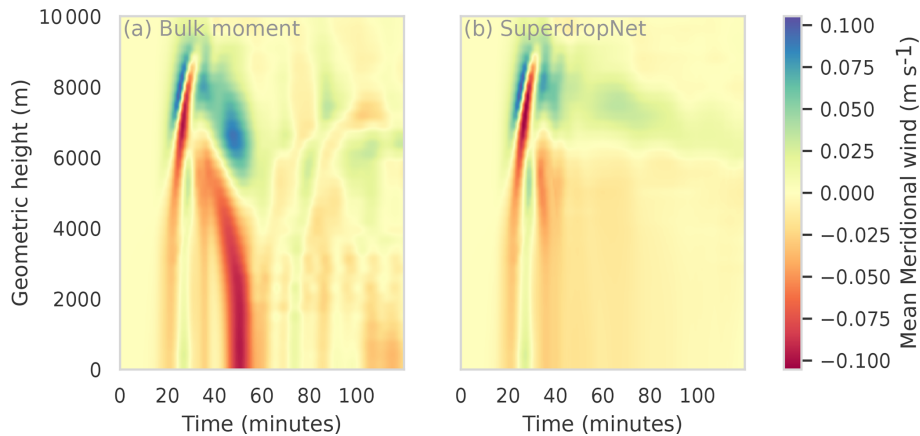**Figure 4.** Averaged meridional winds for the two-moment bulk scheme used in ICON two-moment cloud microphysics **(a)** and for SuperdropNet **(b)**.
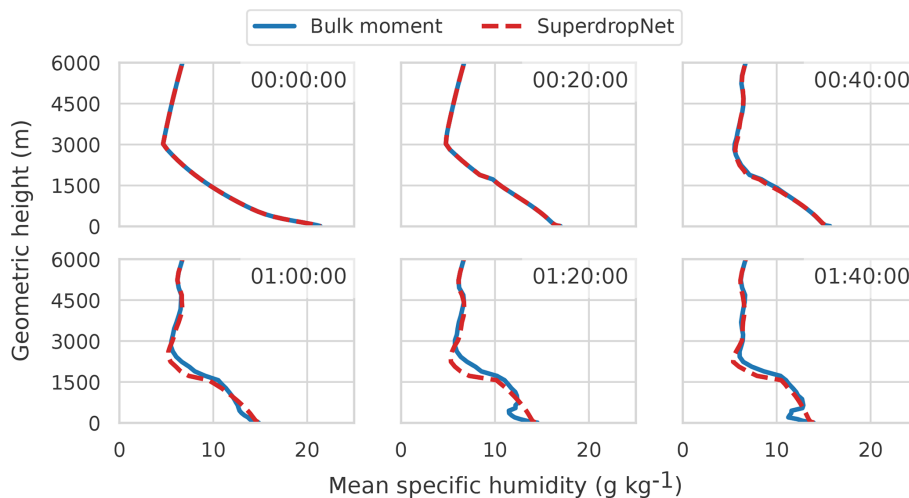

Figure 5. Vertical profile of the specific humidity at different times for the two-moment bulk scheme and for SuperdropNet.

averages the execution time across a total of 496 800 calls to SuperdropNet. Most of those times can be attributed to model inference, while the actual data transfer is less significant. This could be attributed to the fact that ML inference has to be done on a CPU. On a node equipped with an NVIDIA A100 GPU, we measure an inference time of 267 μs. This corresponds to 33 % of the inference time reported on a CPU (see Table 3).

Note, however, that a heterogeneous setup, where moments are transferred to and from the GPU nodes via the Mellanox InfiniBand network, would likely lead to a larger overall wall time. Given the successful efforts to port ICON to GPU, a future experiment could be run exclusively on GPUs. By only applying SuperdropNet when at least one input moment is non-zero, we are already reducing the number of calls to the ML inference to improve performance.

5 Conclusions

We have coupled SuperdropNet, a machine learning algorithm emulating warm-rain processes in a two-moment cloud microphysics scheme, with ICON. In the warm-bubble experiment, the ML emulator is stable, and the results are physically sound.

The strategies to connect ICON and Python provide flexibility to the development of the ML component and account for the fact that ML development is done iteratively. Both embedded Python and YAC can be integrated, with little programming overhead, into ICON. For a later ML emulator, which replaces a full parameterization at the grid level, YAC can be used regardless of the block length. Coupling via pipes is comparatively slow and does not scale well. Since it requires an extensive rewrite of the core components of ICON, we do not recommend it for implementation. Out of the three coupling strategies we tested, embedded Python provided
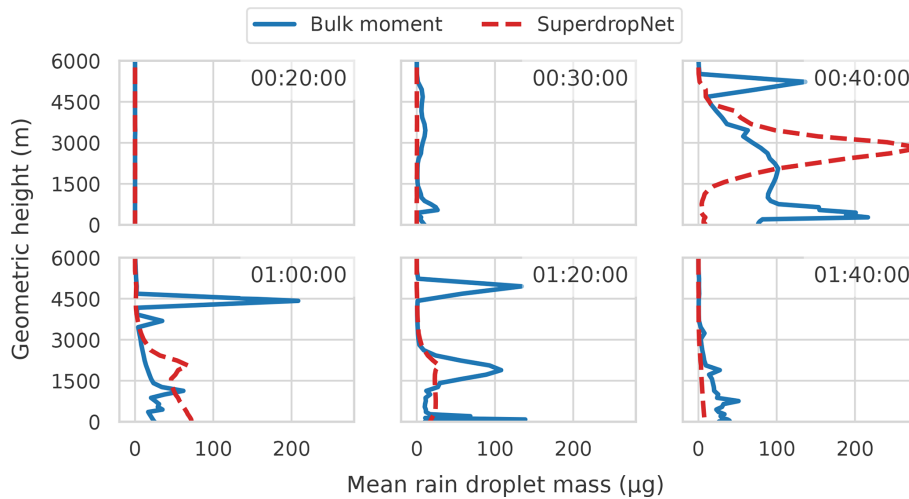
Figure 6. Vertical profile of the rain droplet mass, calculated as the ratio of the specific rain content and the number concentration of rain droplets at different times for the two-moment bulk scheme and for SuperdropNet.

Table 3. Processes when coupling SuperdropNet to ICON via embedded Python and their associated duration. Machine learning inference is executed on a CPU node of the Levante computing system at the German Climate Computing Center (DKRZ).

Process	Time (µs)	Fraction
Time reported by ICON	5.0×10^2	100 %
Time reported by Python	4.8×10^2	96 %
↪ of which time reported for inference	4.4×10^2	87 %
↪ of which time reported for data transfer	4.2×10^1	8.5 %

the fastest performance. It can be used independently of the ICON grid to execute any Python code at any level of the ICON time loop.

We note that by coupling SuperdropNet to ICON, we introduce a scheme that would otherwise be computationally intractable for cloud microphysics in standard numerical simulations. A direct comparison of runtimes is therefore not possible. Note, however, that integrating a Python component will slow down the overall time to solution due to the incurred cost in network inference and data transfer. For applications that are more demanding than our warm-bubble scenario test case and if the ML component is thoroughly tested, a reimplementation in Fortran would likely increase performance at the expense of losing the flexibility in development.

A natural extension of this work is more complex modeling scenarios. This would involve training machine-learning-based emulators for other cloud microphysical processes and/or the introduction of hydrometeors other than clouds and rain. Apart from droplet collisions, processes such as the sedimentation of droplets and deep convection can be challenging to represent with two-moment bulk parameterization schemes. Hence, in the future, we want to explore the possibility of creating ML-based proxies for these processes, while continuing to use hybrid ML ESMs for continuous online testing.

Appendix A: Evaluation of SuperdropNet

A1   Mixed-phase bubble

We include the grid-averaged cloud ice, cloud water, graupel, snow, and ice for the mixed-phase experiment described in Sect. 4.2.1. The results are shown in Fig. A1.
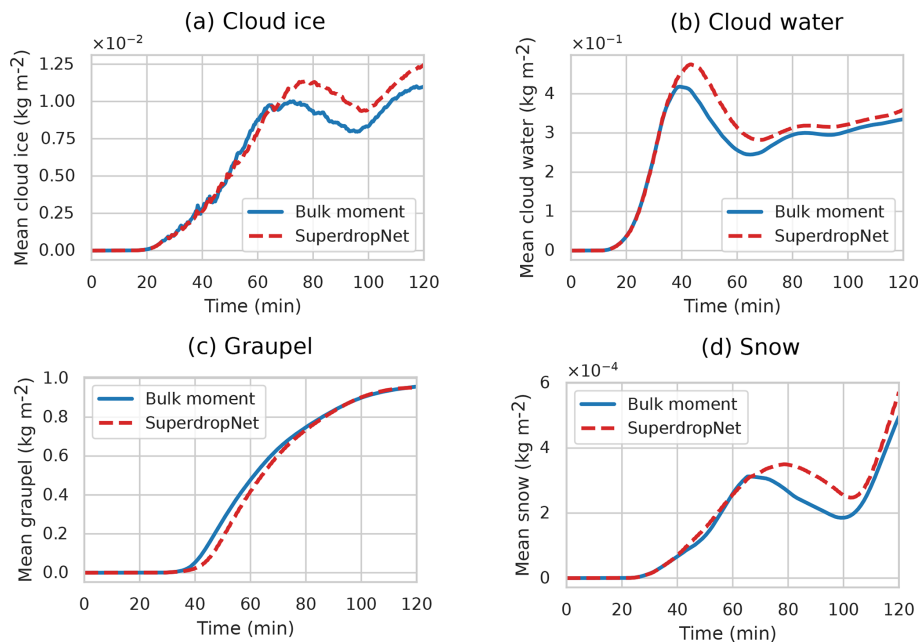

Figure A1. Grid-averaged quantities for the two-moment bulk scheme and SuperdropNet under **(a)** warm-bubble scenario, **(b)** cold-bubble scenario, and **(c)** mixed-phase scenario.

Author contributions. CA developed the embedded Python and YAC coupling software, performed the experiments, provided the visualizations, and led the writing of the paper as a whole. SS developed SuperdropNet. CA and SS defined and evaluated the experiments, curated the software and data, and wrote the original draft. TW developed the pipe coupling software and contributed to the original draft. DG helped conceive the project and define the coupling task and contributed to the original draft. CA, SS, TW, and DG reviewed and edited the final paper.

Competing interests. The contact author has declared that none of the authors has any competing interests.

Disclaimer. Publisher's note: Copernicus Publications remains neutral with regard to jurisdictional claims made in the text, published maps, institutional affiliations, or any other geographical representation in this paper. While Copernicus Publications makes every effort to include appropriate place names, the final responsibility lies with the authors.

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mane, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viegas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X.: Tensorflow: A system for large-scale machine learning, in: 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16), Savannah, GA, US, 2–4 November 2016, 265–283, https://arxiv.org/abs/1603.04467 (last access: 6 September 2023), 2016.

Alexeev, D.: PyTorch bindings for Fortran (v0.4), GitHub [code], https://github.com/alexeedm/pytorch-fortran (last access: 6 September 2023), 2023.

Arnold, C., Sharma, S., and Weigel, T.: DKRZ-AIM/dkrz-hereon-icon-superdropnet: Integrating SuperdropNet (v0.1.0), Zenodo [code], https://doi.org/10.5281/zenodo.10069121, 2023a.

Arnold, C., Sharma, S., and Weigel, T.: ICON Code v 2.6.5 including coupling schemes for integrating SuperdropNet, Zenodo [code], https://doi.org/10.5281/zenodo.8348256, 2023b.

Arnold, C., Sharma, S., and Weigel, T.: Data set for: Efficient and Stable Coupling of the SuperdropNet Deep Learning-based Cloud Microphysics (v0.1.0) to the ICON Climate and Weather Model (v2.6.5), Zenodo [data set], https://doi.org/10.5281/zenodo.8348266, 2023c.

Belochitski, A. and Krasnopolsky, V.: Robustness of neural network emulations of radiative transfer parameterizations in a state-of-the-art general circulation model, Geosci. Model Dev., 14, 7425–7437, https://doi.org/10.5194/gmd-14-7425-2021, 2021.

Beucler, T., Pritchard, M., Rasp, S., Ott, J., Baldi, P., and Gentine, P.: Enforcing Analytic Constraints in Neural Networks Emulating Physical Systems, Phys. Rev. Lett., 126, 098302, https://doi.org/10.1103/PhysRevLett.126.098302, 2021.

Bonanni, A., Hawkes, J., and Quintino, T.: infero: A lower-level API for Machine Learning inference in operations (version 0.2.0), GitHub [code], https://github.com/ecmwf/infero (last access: 6 September 2023), 2022.

Brdar, S. and Seifert, A.: McSnow: A Monte-Carlo Particle Model for Riming and Aggregation of Ice Particles in a Multidimensional Microphysical Phase Space, J. Adv. Model. Earth Sy., 10, 187–206, https://doi.org/10.1002/2017MS001167, 2018.

Brenowitz, N.: Call Python from Fortran (v0.2.1), Zenodo [code], https://doi.org/10.5281/zenodo.7779572, 2023.

Brenowitz, N. D. and Bretherton, C. S.: Prognostic Validation of a Neural Network Unified Physics Parameterization, Geophys. Res. Lett., 45, 6289–6298, https://doi.org/10.1029/2018GL078510, 2018.

Brenowitz, N. D. and Bretherton, C. S.: Spatially Extended Tests of a Neural Network Parametrization Trained by Coarse-Graining, J. Adv. Model. Earth Sy., 11, 2728–2744, https://doi.org/10.1029/2019MS001711, 2019.

Brenowitz, N. D., Beucler, T., Pritchard, M., and Bretherton, C. S.: Interpreting and Stabilizing Machine-Learning Parametrizations of Convection, J. Atmos. Sci., 77, 4357–4375, https://doi.org/10.1175/JAS-D-20-0082.1, 2020a.

Brenowitz, N. D., Henn, B., McGibbon, J., Clark, S. K., Kwa, A., Perkins, W. A., Watt-Meyer, O., and Bretherton, C. S.: Machine Learning Climate Model Dynamics: Offline versus Online Performance, arXiv [preprint], https://doi.org/10.48550/arXiv.2011.03081, 2020b.

Brenowitz, N. D., Perkins, W. A., Nugent, J. M., Watt-Meyer, O., Clark, S. K., Kwa, A., Henn, B., McGibbon, J., and Bretherton, C. S.: Emulating Fast Processes in Climate Models, in: Machine Learning for the Physical Sciences, NEURIPS Workshop, arXiv [preprint], https://doi.org/10.48550/arXiv.2211.10774, 2022.

Chantry, M., Hatfield, S., Dueben, P., Polichtchouk, I., and Palmer, T.: Machine learning emulation of gravity wave drag in numerical weather forecasting, J. Adv. Model. Earth Sy., 13, e2021MS002477, https://doi.org/10.1029/2021MS002477, 2021.

Chevallier, F., Morcrette, J.-J., Chéruy, F., and Scott, N.: Use of a neural-network-based long-wave radiative-transfer scheme in the ECMWF atmospheric model, Q. J. Roy. Meteor. Soc., 126, 761–776, https://doi.org/10.1002/qj.49712656318, 2000.

Chollet, F. et al.: Keras (v2.14.0), GitHub [code], https://github.com/fchollet/keras, last access: 6 September 2023.

Christensen, H. and Zanna, L.: Parametrization in Weather and Climate Models, in: Oxford Research Encyclopedia of Climate Science, Oxford University Press, ISBN 978-0-19-022862-0, https://doi.org/10.1093/acrefore/9780190228620.013.826, 2022.

Curcic, M.: A parallel Fortran framework for neural networks and deep learning, arXiv [preprint], https://doi.org/10.48550/arXiv.1902.06714, 2019.

Dong, W., Fritts, D. C., Liu, A. Z., Lund, T. S., Liu, H.-L., and Snively, J.: Accelerating Atmospheric Gravity Wave Simulations Using Machine Learning: Kelvin-Helmholtz Instability and Mountain Wave Sources Driving Gravity Wave Breaking and Secondary Gravity Wave Generation, Geophys. Res. Lett., 50, e2023GL104668, https://doi.org/10.1029/2023GL104668, 2023.

Dueben, P., Modigliani, U., Geer, A., Siemen, S., Pappenberger, F., Bauer, P., Brown, A., Palkovic, M., Raoult, B., Wedi, N., and Baousis, V.: Machine learning at ECMWF: A roadmap for the next 10 years, ECMWF Technical Memoranda, ECMWF, https://doi.org/10.21957/ge7ckgm, 2021.

Dueben, P. D., Schultz, M. G., Chantry, M., Gagne, D. J., Hall, D. M., and McGovern, A.: Challenges and Benchmark Datasets

for Machine Learning in the Atmospheric Sciences: Definition, Status, and Outlook, Artificial Intelligence for the Earth Systems, 1, e210002, https://doi.org/10.1175/AIES-D-21-0002.1, 2022.

Elafrou, A., Orchard, D., and Clifford, S.: fortran-pytorch-lib (commit: ffe833b66a6e1ce1c6cf023708d1f351a3a11f8b), GitHub [code], https://github.com/Cambridge-ICCS/fortran-pytorch-lib, last access: 6 September 2023.

Gentine, P., Pritchard, M., Rasp, S., Reinaudi, G., and Yacalis, G.: Could Machine Learning Break the Convection Parameterization Deadlock?, Geophys. Res. Lett., 45, 5742–5751, https://doi.org/10.1029/2018GL078202, 2018.

Grundner, A., Beucler, T., Gentine, P., Iglesias-Suarez, F., Giorgetta, M. A., and Eyring, V.: Deep Learning Based Cloud Cover Parameterization for ICON, J. Adv. Model. Earth Sy., 14, e2021MS002959, https://doi.org/10.1029/2021MS002959, 2022.

Hanke, M., Redler, R., Holfeld, T., and Yastremsky, M.: YAC 1.2.0: new aspects for coupling software in Earth system modelling, Geosci. Model Dev., 9, 2755–2769, https://doi.org/10.5194/gmd-9-2755-2016, 2016.

Hanke, M., Dreier, N.-A., and Redler, R.: YetAnotherCoupler (YAC) (version 2.6.1) [code], https://dkrz-sw.gitlab-pages.dkrz.de/yac/, last access: 6 September 2023.

Irrgang, C., Boers, N., Sonnewald, M., Barnes, E. A., Kadow, C., Staneva, J., and Saynisch-Wagner, J.: Towards neural Earth system modelling by integrating artificial intelligence in Earth system science, Nature Machine Intelligence, 3, 667–674, https://doi.org/10.1038/s42256-021-00374-3, 2021.

Krasnopolsky, V. M., Fox-Rabinovitz, M. S., and Chalikov, D. V.: New approach to calculation of atmospheric model physics: Accurate and fast neural network emulation of longwave radiation in a climate model, Mon. Weather Rev., 133, 1370–1383, https://doi.org/10.1175/MWR2923.1, 2005.

McGovern, A., Lagerquist, R., Gagne, D. J., Jergensen, G. E., Elmore, K. L., Homeyer, C. R., and Smith, T.: Making the Black Box More Transparent: Understanding the Physical Implications of Machine Learning, B. Am. Meteorol. Soc., 100, 2175–2199, https://doi.org/10.1175/BAMS-D-18-0195.1, 2019.

Meyer, D., Hogan, R. J., Dueben, P. D., and Mason, S. L.: Machine Learning Emulation of 3D Cloud Radiative Effects, J. Adv. Model. Earth Sy., 14, e2021MS002550, https://doi.org/10.1029/2021MS002550, 2022.

Morrison, H., van Lier-Walqui, M., Fridlind, A. M., Grabowski, W. W., Harrington, J. Y., Hoose, C., Korolev, A., Kumjian, M. R., Milbrandt, J. A., Pawlowska, H., Posselt, D. J., Prat, O. P., Reimel, K. J., Shima, S.-I., van Diedenhoven, B., and Xue, L.: Confronting the Challenge of Modeling Cloud and Precipitation Microphysics, J. Adv. Model. Earth Sy., 12, e2019MS001689, https://doi.org/10.1029/2019MS001689, 2020.

Mu, B., Chen, L., Yuan, S., and Qin, B.: A radiative transfer deep learning model coupled into WRF with a generic fortran torch adaptor, Frontiers in Earth Science, 11, https://doi.org/10.3389/feart.2023.1149566, 2023.

Nowack, P., Braesicke, P., Haigh, J., Abraham, N. L., Pyle, J., and Voulgarakis, A.: Using machine learning to build temperature-based ozone parameterizations for climate sensitivity simulations, Environ. Res. Lett., 13, 104016, https://doi.org/10.1088/1748-9326/aae2be, 2018.

Ott, J., Pritchard, M., Best, N., Linstead, E., Curcic, M., and Baldi, P.: A Fortran-Keras Deep Learning Bridge for Scientific Computing, Scientific Programming, 2020, 8888811, https://doi.org/10.1155/2020/8888811, 2020.

Partee, S., Ellis, M., Rigazzi, A., Shao, A. E., Bachman, S., Marques, G., and Robbins, B.: Using Machine Learning at scale in numerical simulations with SmartSim: An application to ocean climate modeling, J. Comput. Sci.-Neth., 62, 101707, https://doi.org/10.1016/j.jocs.2022.101707, 2022.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S.: PyTorch: An Imperative Style, High-Performance Deep Learning Library, in: Advances in Neural Information Processing Systems 32, Vancouver, Canada, 8–14 December 2019, Curran Associates, Inc., 8024–8035, http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf (last access: 6 September 2023), 2019.

Qu, Y. and Shi, X.: Can a Machine Learning–Enabled Numerical Model Help Extend Effective Forecast Range through Consistently Trained Subgrid-Scale Models?, Artif. Intell. Earth Syst., 2, e220050, https://doi.org/10.1175/AIES-D-22-0050.1, 2023.

Rasp, S.: Coupled online learning as a way to tackle instabilities and biases in neural network parameterizations: general algorithms and Lorenz 96 case study (v1.0), Geosci. Model Dev., 13, 2185–2196, https://doi.org/10.5194/gmd-13-2185-2020, 2020.

Rasp, S., Pritchard, M. S., and Gentine, P.: Deep learning to represent subgrid processes in climate models, P. Natl. Acad. Sci. USA, 115, 9684–9689, https://doi.org/10.1073/pnas.1810286115, 2018.

Rigo, A. and Fijalkowski, M.: C Foreign Function Interface for Python, CFFI [code], https://cffi.readthedocs.io/en/release-1.14/ (last access: 6 September 2023), 2018.

Berkeley Lab: Inference Engine (v0.10.0), GitHub [code], https://github.com/BerkeleyLab/inference-engine/, 2023.

Seifert, A. and Beheng, K.: A two-moment cloud microphysics parameterization for mixed-phase clouds. Part 1: Model description, Meteorol. Atmos. Phys., 92, 45–66, https://doi.org/10.1007/s00703-005-0112-4, 2006.

Seifert, A. and Beheng, K. D.: A double-moment parameterization for simulating autoconversion, accretion and selfcollection, Atmos. Res., 59–60, 265–281, https://doi.org/10.1016/s0169-8095(01)00126-0, 2001.

Seifert, A. and Rasp, S.: Potential and Limitations of Machine Learning for Modeling Warm-Rain Cloud Microphysical Processes, J. Adv. Model. Earth Sy., 12, e2020MS002301, https://doi.org/10.1029/2020MS002301, 2020.

Sharma, S. and Greenberg, D.: SuperdropNet: a Stable and Accurate Machine Learning Proxy for Droplet-based Cloud Microphysics, arXiv [preprint], https://doi.org/10.48550/arXiv.2402.18354, 2024.

Shima, S., Kusano, K., Kawano, A., Sugiyama, T., and Kawahara, S.: The super-droplet method for the numerical simulation of clouds and precipitation: a particle-based and probabilistic microphysics model coupled with a non-hydrostatic model, Q. J. Roy. Meteor. Soc., 135, 1307–1320, https://doi.org/10.1002/qj.441, 2009.

Sonnewald, M., Lguensat, R., Jones, D. C., Dueben, P. D., Brajard, J., and Balaji, V.: Bridging observations, theory and numerical simulation of the ocean using machine learning, Environ. Res. Lett., 16, 073008, https://doi.org/10.1088/1748-9326/ac0eb0, 2021.

Yuval, J. and O'Gorman, P. A.: Neural-Network Parameterization of Subgrid Momentum Transport in the Atmosphere, J. Adv. Model. Earth Sy., 15, e2023MS003606, https://doi.org/10.1029/2023MS003606, 2023.

Yuval, J., O'Gorman, P. A., and Hill, C. N.: Use of Neural Networks for Stable, Accurate and Physically Consistent Parameterization of Subgrid Atmospheric Processes With Good Performance at Reduced Precision, Geophys. Res. Lett., 48, e2020GL091363, https://doi.org/10.1029/2020GL091363, 2021.

Zhong, X., Ma, Z., Yao, Y., Xu, L., Wu, Y., and Wang, Z.: WRF–ML v1.0: a bridge between WRF v4.3 and machine learning parameterizations and its application to atmospheric radiative transfer, Geosci. Model Dev., 16, 199–209, https://doi.org/10.5194/gmd-16-199-2023, 2023.