Geosci. Model Dev., 16, 199–209, 2023
https://doi.org/10.5194/gmd-16-199-2023

Geoscientific Model Development
Open Access

WRF–ML v1.0: a bridge between WRF v4.3 and machine learning parameterizations and its application to atmospheric radiative transfer

Xiaohui Zhong, Zhijian Ma, Yichen Yao, Lifei Xu, Yuan Wu, and Zhibin Wang

Damo Academy, Alibaba Group, Hangzhou 311121, China

Correspondence: Zhibin Wang (zhibin.waz@alibaba-inc.com)

Received: 7 September 2022 – Discussion started: 23 September 2022
Revised: 7 December 2022 – Accepted: 7 December 2022 – Published: 6 January 2023

Abstract. In numerical weather prediction (NWP) models, physical parameterization schemes are the most computationally expensive components, despite being greatly simplified. In the past few years, an increasing number of studies have demonstrated that machine learning (ML) parameterizations of subgrid physics have the potential to accelerate and even outperform conventional physics-based schemes. However, as the ML models are commonly implemented using the ML libraries written in Python, very few ML-based parameterizations have been successfully integrated with NWP models due to the difficulty of embedding Python functions into Fortran-based NWP models. To address this issue, we developed a coupler to allow the ML-based parameterizations to be coupled with a widely used NWP model, i.e., the Weather Research and Forecasting (WRF) model. Similar to the WRF I/O methodologies, the coupler provides the options to run the ML model inference with exclusive processors or the same processors for WRF calculations. In addition, to demonstrate the effectiveness of the coupler, the ML-based radiation emulators are trained and coupled with the WRF model successfully.

1 Introduction

Numerical weather prediction (NWP) models have become the most important tools for operational weather forecasting and have many applications in different domains, including energy, traffic, logistics, and planning (Coiffier, 2011; Pu and Kalnay, 2018). The physics-based parameterizations are essential in the NWP models as they approximate some processes that are either too small-scale to be explicitly resolved at the model grid resolution or too complex and not fully understood. Although many simplifications are made to parameterizations to reduce the computational cost, the calculations of physical parameterizations still account for a significant portion of the total computational time of NWP models (Wang et al., 2019). Also, the parameterization schemes often contain uncertain parameters estimated from more faithful high-resolution simulations with statistical models (Stensrud, 2013). For example, the parameters in radiative transfer parameterizations can be fitted to the output of the most accurate line-by-line model (Clough et al., 2005), and parameters in cloud turbulence parameterizations can be inferred from large-eddy simulations (LESs) (Mason, 1989). However, other parameters can only be learned from observations as the related governing equations are unknown (Schneider et al., 2017).

One alternative is to train machine learning (ML) models to replace the traditional physics-based parameterization schemes. The ML-based parameterizations have the potential to outperform traditional parameterizations with higher computational efficiency. For example, the radiative transfer parameterization scheme is one of the most computationally expensive components in NWP models, and it has the longest history of developing ML-based radiation emulators. Chevallier et al. (1998, 2000) developed a neural network (NN) based longwave radiation parameterization (NeuroFlux) and has been used operationally in the European Centre for Medium-Range Weather Forecasts (ECMWF) four-dimensional variational data assimilation system. The NeuroFlux is 7 times faster than the previous scheme with comparable accuracy (Janisková et al., 2002). Recently, Song and Roh (2021) developed and used the neural network based radiation emulators in the operational weather forecasting model in the Korea Meteorological Administration. They demonstrated that using NN-based emulators frequently can improve real-time weather forecasting in terms of accuracy and speed compared to the original method, which infrequently uses the original radiation parameterization.

Similarly, ML-based emulators have been developed for other parameterization schemes in NWP models. Rasp et al. (2018) successfully developed and coupled an NN-based convection parameterization into an aquaplanet general circulation model (GCM). They showed that the NN-based parameterization was able to perform multi-year simulations, of which results were close to that of the original simulations. For planetary boundary layer (PBL) parameterization, Wang et al. (2019) used the inputs and outputs from Yonsei University (YSU) PBL scheme of the Weather Research and Forecasting (WRF) model to develop the NN-based parameterizations. They showed that the NN-based parameterization successfully simulated the vertical profiles of velocities, temperature, and water vapor within the PBL over the entire diurnal cycle. Urban land surface models (ULSMs) are reasonably fast, but none of them is best at predicting all the main surface fluxes (Grimmond et al., 2010, 2011). One solution is to run an ensemble of ULSMs coupled to the NWP models to improve predictions, which is technically difficult to implement and would require more computational time due to running multiple ULSMs simultaneously. Meyer et al. (2022a) developed an urban NN (UNN) to emulate the ensemble mean of 22 ULSMs. They demonstrate that the WRF model coupled with the UNN is more accurate than the WRF with a single well-known USLM. Grundner et al. (2021) demonstrated the potential of an NN-based cloud cover parameterization to accurately reproduce the subgrid scale cloud fraction to improve the low-resolution climate model predictions.

In many cases, ML-based parameterization schemes are only evaluated in offline settings due to the practical challenge of coupling ML-based emulators with NWP models. However, the offline performance of ML-based emulators does not necessarily reflect their online performance when coupled with other components of NWP models. ML-based parameterizations are often trained using high-level programming languages like Python and easy-to-use ML libraries such as PyTorch (Paszke et al., 2017) and Keras (Gulli and Pal, 2017). However, the NWP codes are mainly written in Fortran, making it difficult to integrate directly with ML-based emulators. Researchers used multiple approaches to circumvent these issues. Krasnopolsky (2014) developed a single-layer NN software for developing NN-based radiation emulators in NWP models (Belochitski and Krasnopolsky, 2021; Krasnopolsky et al., 2010; Song and Roh, 2021). Some researchers wrote a Fortran module that reproduces NN architectures using matrix–matrix multiplication with the weights saved from offline training (Chantry et al., 2021; Hatfield et al., 2021; Rasp et al., 2018), which only works well for simple NN architectures such as the fully connected (FC) NNs and becomes increasingly difficult to implement for more complex NN structures. Ott et al. (2020) produced an open source software library, the Fortran–Keras Bridge (FKB), which enables users to access many features of the Keras application programming interface (API) in Fortran and implement NNs in Fortran code. However, the FKB can only support FC or dense layer NNs. Therefore, researchers cannot fully use the most advanced NN structures, such as convolution, self-attention, and variational autoencoder structures, to build powerful ML-based emulators for online applications. Wang et al. (2022) proposed an NN–GCM coupler to allow Python-based NNs to communicate with the Fortran-based GCM. Although using the coupler can make the ML-based emulators be deployed efficiently, its requirement for data transfer on a hard disk makes it hard to achieve speed-up.

To address the abovementioned problems, we developed a WRF–ML coupler that allows any ML-based parameterization schemes to be coupled with the WRF model. The WRF model is selected as it is a popular open source NWP model used by many researchers and operational organizations. Also, to demonstrate the applicability of the coupler, we trained and coupled the ML-based radiation emulators with the WRF model. The ML-based radiation emulators were studied most among all the physical parameterization schemes and coupled with the ECMWF Integrated Forecasting System (IFS), the National Centers for Environmental Prediction (NCEP) Climate Forecast System (CFS), and the WRF model in the previous studies (Song and Roh, 2021; Meyer et al., 2022b). Also, the rapid radiative transfer model for general circulation models (RRTMG; Iacono et al., 2008) is selected for radiation as it is widely used in weather and climate models, such as the NCEP Global Forecast System (GFS) and the China Meteorological Administration (CMA) Global/Regional Assimilation and Prediction System (GRAPES). To train the ML-based radiation emulators, the WRF model is run to generate a dataset for training and evaluation. We also illustrate the performance of the ML-based radiation emulators in the online setting. Lastly, we compare the accuracy and computational costs of the WRF simulations coupled with ML-based radiation emulators with the WRF simulations using original RRTMG schemes.

The remainder of the paper is organized as follows. Section 2 describes the original and ML-based RRTMG module. Section 3 introduces the WRF I/O quilt processors and the WRF–ML coupler. Section 4 briefly presents the WRF model setup and ML-based radiation emulators. Section 5 presents the analysis and results in online settings. Conclusions and discussions are in Sect. 6.

```
subroutine radiation_driver
    defining variables
    initializing data
    call data preprocessing related subroutines, e.g.: call
    radconst, cldfra, ozone or aerosol related interpolation
    lwrad_select:
    select case (config_flag%ra_lw_physics)
        case (RRTMG_LWSCHEME)
            call rrtmg_lwinit
            call rrtmg_lwrad
        case (other lw schemes)
    end select lwrad_select
    swrad_select:
    select case (config_flag%ra_sw_physics)
        case (RRTMG_SWSCHEME)
            call rrtmg_swrad
        case (other sw schemes)
    end select swrad_select
end subroutine radiation_driver
```

Figure 1. Pseudo-code for the original RRTMG module.

2 Description of original and ML-based RRTMG model

2.1 Description of original RRTMG module

The RRTMG is developed by the Atmospheric and Environmental Research (AER) and comprises shortwave (SW) and longwave (LW) schemes that calculate SW and LW radiation fluxes and heating rates, respectively. Figure 1 presents the pseudo-code for the original radiation module in the WRF model. At the beginning of the radiation driver module, the input and output variables are initialized. Then, the corresponding radiation calculations are started according to the parameters (i.e., ra_lw_physics and ra_sw_physics) specified in the configuration file (i.e., namelist.input). When the RRTMG schemes are selected, the subroutine rrtmg_lwinit is called to perform calculations needed to initialize the longwave calculations. Next, the subroutines rrtmg_lwrad and rrtmg_swrad are called to perform the RRTMG SW and LW radiation calculations.

2.1.1 Description of ML-based RRTMG module

The original radiation computations process one atmospheric column at a time and loop over all the horizontal grid points. The three-dimensional variables associated with the grid cell (indexed as (i, j, k) in the WRF model output files) are indexed as (i, k, j) in memory by the WRF to make it more cache friendly and increase the cache hit rate. The index i, j, and k represent the west–east, south–north, and vertical directions, respectively. With the ML-based RRTMG model, inference can be implemented on a batch of data, which requires the input data to be packed into batches. The input features of ML-based radiation emulators have dimensions of $W \times H$, where W and H represent the number of features and vertical levels, respectively. Therefore, the original three-dimensional variables are reshaped to be indexed as (i, j, k) to match the dimensions requirement of ML-based input features (done by preprocess_layout in Fig. 2). The ML model inference is completed within the function infer_run. Unlike physics-based radiation schemes, ML-based schemes do not need to calculate intermediate variables such as optical depth. Instead, the ML-based model accomplishes the mapping from the input features to outputs. After inference is finished, ML model outputs will be post-processed to match the dimensions of the original WRF data array to continue model simulations.

```
subroutine radiation_driver
    defining variables
    initializing data
    call preprocess_layout
    call infer_run
    call postprocess
end subroutine radiation_driver
```

Figure 2. Pseudo-code for the ML-based RRTMG module.

3 WRF–ML coupler

This subsection briefly describes the WRF I/O quilt server techniques as a similar method is adopted in the WRF–ML coupler to have exclusive servers for ML model inference. Then the methodologies of the WRF–ML coupler are described in detail.

3.1 Description of WRF I/O quilt servers

The WRF model is designed to run efficiently on parallel distributed-memory computing systems. To do this, the WRF applies domain decomposition to split the entire domain into nprocs (equal to $n_x \times n_y$) number of subdomains so that the total amount of work is divided into nprocs compute tasks, where n_x and n_y are the number of processors along the west–east and south–north directions. Furthermore, the WRF model contains an I/O layer to gather all the distributed data arrays onto the master message passing interface (MPI) rank 0 while other MPI ranks block until the master completes the write. However, as the domain size increases or resolution increases, leading to an increasing amount of information to write, the increasing time cost of writing output files becomes a bottleneck for the overall performance. Therefore, the WRF model also provides an option to use asynchronous I/O (quilt) servers that deal exclusively with I/O so that the compute processors can continue without waiting for writing output files. More specifically, the WRF allows users to specify the number of groups of I/O servers to allocate (nio_groups) and the number of I/O ranks per group (nio_tasks_per_group). Similar to the I/O quilt servers, the WRF–ML coupler also provides the option to use several processors for ML model inference exclusively, of which more details are described in the following subsection. While GPUs are typically more powerful than CPUs, both CPUs and GPUs are supported for inference. CPUs are cheaper than GPUs and more widely available, and using CPUs for inference can also avoid extra costs due to GPU–CPU data transfer.

3.1.1 Description of the WRF–ML coupler

Figure 3 shows that when the WRF model launches the initialization process, the subroutine infer_init is called to initialize the ML model inference services. Similar to the asynchronous I/O servers, several processes are assigned to exclusively work on ML model inference using either GPUs or CPUs. The number of inference processors and the number of inference ranks per group are specified in the configuration file by setting the newly added variables ninfer_tasks_per_group and ninfer_groups in the namelist.input file. The difference between the inference processors and the asynchronous I/O servers is that the inference processors apply the synchronous method, as WRF calculations run sequentially, and the subsequent calculations of the WRF model rely on the outputs from ML-based parameterizations. Usually, the number of inference processors is much smaller than that of the WRF compute processors, so a single inference processor must process data from multiple processors. The inference processor listens and receives data from the corresponding WRF compute processors through the MPI transmission while blocking the compute processes (see the right part in Fig. 4). After inference is finished, the inference process sends outputs of ML-based schemes back to the compute processors to continue the standard WRF calculations. In addition, the ninfer_tasks_per_group and ninfer_groups can be set to -1 to execute the ML model inference using the same processors as for the WRF compute processes. Then, memory copy is not needed between the WRF compute processors and inference processors.

The functions for implementing the ML-based model inference are all written in Python, commonly used by the machine learning community. To allow the Fortran-based WRF code to call those Python functions, it is necessary first to link the Fortran executables to the system Python library. iso_c_binding is an intrinsic module defined in Fortran 2003 for interoperability with C. As shown in the left part of Fig. 3, the iso_c_binding is imported within the Fortran code. Although no C code is needed, the C foreign function interface (CFFI) for Python is used to generate C-style binding interfaces within the Python script (see the middle part of Fig. 3). The interfaces for C-function are also defined in the Python scripts, which are used to generate a dynamic library that Fortran modules can link.

This WRF–ML coupler allows the ML models to be easily coupled with the WRF model with minimal effort. Also, researchers are able to make full use of the state-of-the-art ML model structures. This methodology is made open source in the hope that more and more machine learning researchers will participate in improving the traditional NWP models.

4 Experiment setup

The ML-based radiation emulator is coupled with the WRF model to demonstrate the coupler's practicality. This section firstly describes the WRF model setup, ML-based radiation emulators' network structures, and offline evaluation metrics.

4.1 WRF model setup

In this work, the WRF model version 4.3 is compiled using the GNU Fortran (gfortran version 6.5.1) compiler with the "dmpar" option. To generate a dataset for model training, the WRF model is run using the domain configuration shown in Fig. 5. The WRF model is set up within the GFS model grid with a single domain at a horizontal grid spacing of 5 km and has 190×170 grid points in the west–east and north–south directions (i.e., 32 300 grid cells). The number of vertical levels is 57, and the model top is 10 hPa. In addition, the WRF model is configured using physics schemes, including the New Thompson et al. scheme (Thompson et al., 2008), Bougeault–Lacarrère (BouLac) planetary boundary layer (PBL) scheme (Bougeault and Lacarrère, 1989), RUC land surface model (Smirnova et al., 1997; Smirnova et al., 2000), and RRTMG for both SW and LW radiation (Iacono et al., 2008). The WRF simulations were run for 3 d per month, and the initialization days were randomly selected. The first 2 d of every 3 d simulations are used for training, and the last day is used for testing. The model generates radiation inputs and outputs every 30 model minutes.

4.2 ML-based radiation emulators and offline evaluations

Many researchers (Chevallier et al., 2000; Krasnopolsky et al., 2010; Song and Roh, 2021) have previously used the FC networks to replace the radiation schemes within the operational NWP models. Additionally, Ukkonen (2022) showed that bidirectional recurrent NNs (RNNs) are more accurate than the feed-forward NNs (FNNs) for emulating atmospheric radiative transfer. Yao et al. (2023) also demonstrated that the bidirectional long short-term memory (Bi-LSTM) achieves the best accuracy in radiative transfer parameterization. Therefore, this study has trained and tested both FC networks and Bi-LSTM models. As an increasing number of trainable parameters of a ML model increases both accuracy and computational cost, finding a balance is

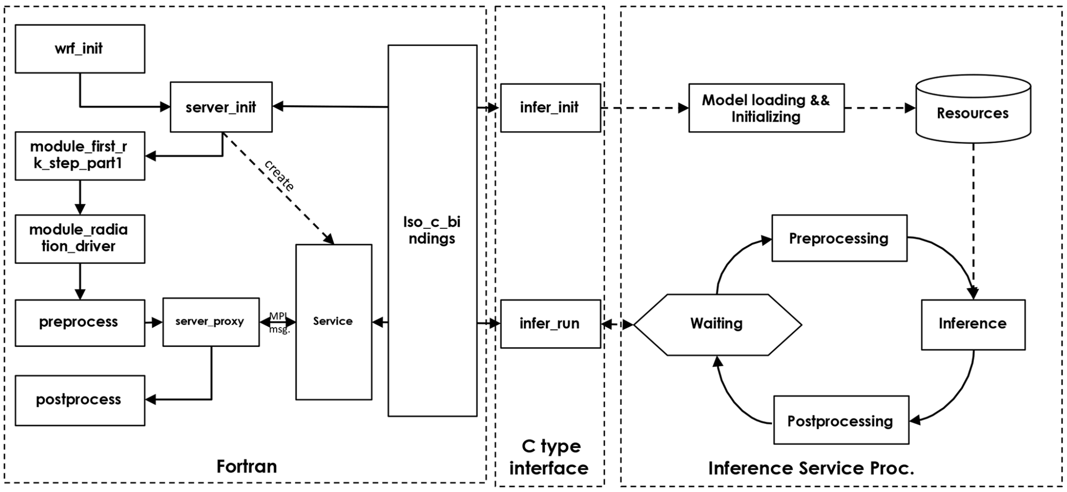

Figure 3. Framework of the WRF–ML coupler.

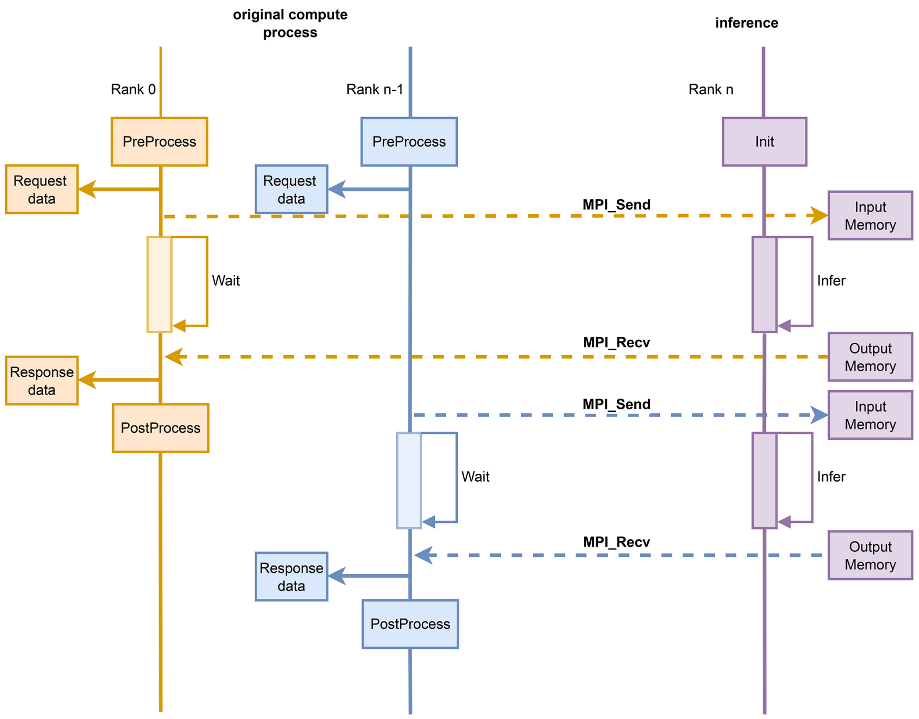

Figure 4. Synchronous sequence implementation between WRF and ML-based parameterization.

crucial for applying ML models in operational NWP models. Table 1 illustrates the structures of five ML models, including two FC networks (models A and B) and three Bi-LSTM models (models C to E). Models A and D, as well as models B and E, have approximately the same number of parameters, respectively. More details about the model training and comparisons can be referenced in Yao et al. (2023). All the ML-based emulators have been converted to ONNX and run using the ONNX Runtime library version 1.7.

Figure 6 shows the inference time of different ML-based radiation emulators on one Intel Xeon Platinum 8269CY CPU and one NVIDIA Tesla T4 GPU, respectively. Model E has the longest inference time and is much longer than model B, even though they have a similar number of parameters. This is because the FC networks have a much lower time complexity than LSTM models. Similarly, although the number of parameters of model A is about 10.3 and 1.03 times that of models C and D (see Table 1), model A is still faster than models C and D.

Table 1. Description of different ML models used for emulating radiative transfer calculations, including description of model structures and total number of parameters.

Model name	Network structure	Number of layers (bidirectional layers for Bi-LSTM models)	Number of nodes	Number of parameters
Model A	FC	5	30	70 248
Model B	FC	10	200	840 028
Model C	Bi-LSTM	1	16	6788
Model D	Bi-LSTM	3	32	67 844
Model E	Bi-LSTM	5	96	993 028

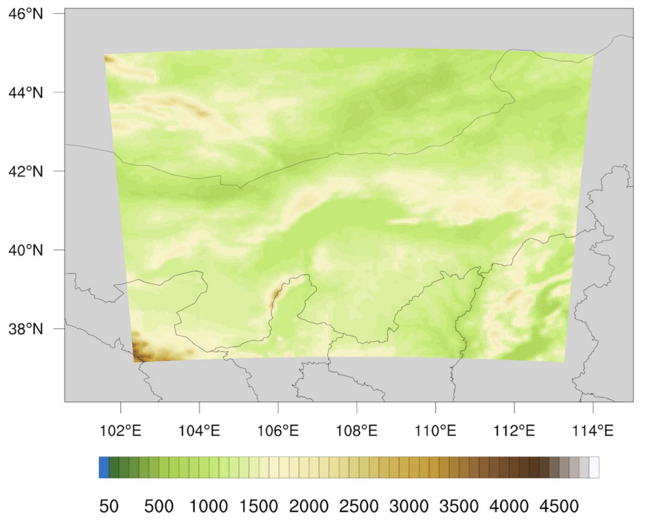

Figure 5. Digital evaluation data of the single WRF domain with horizontal resolution at 5 km.

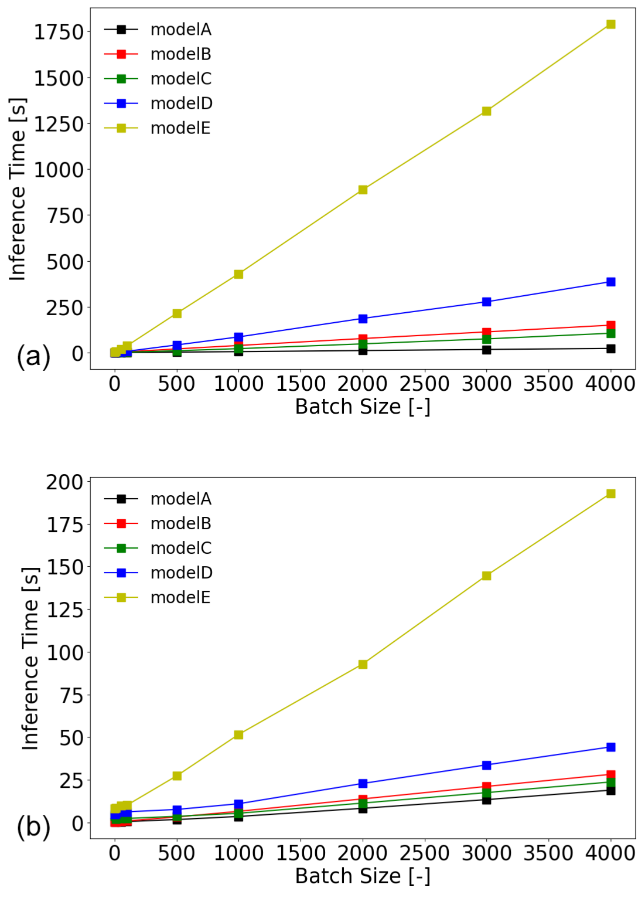

Figure 6. Inference time of different ML-based emulators run on one CPU core **(a)** and GPU **(b)** with different batch sizes. The number of vertical levels is 57.

The performance of different ML-based radiation emulators is evaluated on the testing data in the offline setting. Table 2 summarizes the root mean square error (RMSE) of fluxes and heating rates which are the final outputs of the radiation schemes. Since SW fluxes at the surface and top-of-atmosphere (TOA) radiation are particularly important to climate and weather prediction, their RMSE is also shown in Table 2. It is shown that model E is the most accurate, with RMSE of SW and LW fluxes of about 6.157 and 1.274 $\mathrm{W\,m^{-2}}$, RMSE of SW and LW heating rates of about 0.051 and 0.095 $\mathrm{K\,d^{-1}}$, and SW fluxes at the surface and TOA of about 6.871 and 5.576 $\mathrm{W\,m^{-2}}$. Model D has comparable accuracy to the best-performing model E while having only 1/14 of the parameters and a much shorter inference time. Furthermore, considering that model C only has the smallest number of parameters, model C has relatively good accuracy, while the RMSE of SW heating rate is 5 times higher than model E. On the other hand, models A and B have at least 4 times higher RMSE than model E in terms of heating rates, with RMSE of SW fluxes at the surface and top of the atmosphere at least 20 times higher than model E. In summary, it is demonstrated that the Bi-LSTM model is more accurate than FC networks for radiation emulation, and model D achieves the best balance between accuracy and computing efficiency.

Since it is difficult to know the performance of the ML-based emulators in the online settings based on offline evaluation, the following Section presents the online evaluations of different ML-based emulators coupled with the WRF

Table 2. RMSE of different ML-based radiation emulators for test data in offline evaluation.

	SW flux $\mathrm{W\,m^{-2}}$	LW flux $\mathrm{W\,m^{-2}}$	SW heating rate $\mathrm{K\,d^{-1}}$	LW heating rate $\mathrm{K\,d^{-1}}$	Surface SW flux $\mathrm{W\,m^{-2}}$	TOA SW flux $\mathrm{W\,m^{-2}}$
Model A	247.0	41.46	1.172	2.035	209.2	273.5
Model B	195.2	29.64	0.946	2.172	173.3	216.3
Model C	7.658	2.032	0.277	0.337	8.534	6.628
Model D	5.355	1.305	0.099	0.161	5.854	4.804
Model E	6.157	1.274	0.051	0.095	6.871	5.576

model. In addition, the WRF model is also run with the original RRTMG schemes as a reference to evaluate the performance of the ML-based emulators.

5 Online evaluation results

The ML-based radiation emulators are coupled with the WRF model using the WRF–ML coupler. The WRF simulations coupled with ML-based emulators were run using three configurations: zero exclusive processes for inference, four exclusive inference processes on CPUs, and four exclusive inference processes on GPUs. Also, the total number of WRF processes is kept to 24 to ensure fair comparisons. The calculation time of radiation driver using different ML-based radiation emulators is compared. The WRF model is initialized at 12:00 UTC on a randomly selected day (i.e., 9 November) in the year 2021 and runs for three days, which is not part of the previously used dataset to ensure the ML-based emulators are evaluated on unseen data. Finally, the temperature at 2 m ($T_{2\,\mathrm{m}}$) and wind speed at 10 m ($\mathrm{WS}_{10\,\mathrm{m}}$) output from the WRF model are compared.

5.1 Computational efficiency evaluation of radiation driver

The calculation time of the WRF radiation driver using original RRTMG schemes and ML-based radiation emulators are profiled using the timing function rsl_internal_microclock provided by the WRF model. Table 3 shows that it takes about 1441 ms (ms: milliseconds) to run a radiation driver using the original RRTMG schemes. In terms of ML-based radiation emulators, using GPUs for inference is significantly faster than using CPUs for inference, especially for the Bi-LSTM model, as it is less computationally efficient than the FC networks given a similar number of parameters. When GPUs are used for inference, all ML-based emulators except model E are at least 3 times faster than the original RRTMG schemes. However, whether this speed-up is due to applying ML-based emulators or the upgrade in hardware (using GPUs instead of CPUs) is unclear and is beyond the scope of this paper. On the other hand, when CPUs are used for inference, having four processes exclusively for inference is slower than having inference executed on the WRF compute

Table 3. Calculation time (in units of milliseconds) of radiation driver when using the original RRTMG schemes and the ML-based radiation emulators. The batch size is 32 300, as the domain has 190 × 170 grid points.

Total processes	24		
Original RRTMG	1441.24 ms		
ML inference processes	0	4 CPUs	4 GPUs
Model A	91.73	237.57	176.87
Model B	365.31	808.43	331.34
Model C	666.37	921.56	258.66
Model D	2620.92	3383.42	418.28
Model E	7166.46	10 330.05	1690.76

processors, which is probably due to the more significant increase in time cost due to data copy between CPUs than the decrease in time cost from having CPUs exclusively for inference.

5.2 Prognostic validation

Figures 7 and 8 present the difference in 24, 48, and 72 h forecasts between the WRF simulations coupled with ML-based radiation emulators and the WRF simulation with original RRTMG schemes. On the spatial difference, the red and blue patterns indicate significantly positive and negative biases of ML-based simulations, respectively. On the other hand, green patterns indicate no or small difference from the original WRF simulations. A domain-averaged mean absolute difference (MAD) is also calculated to measure the overall performance of ML-based emulators in online settings. Notably, the difference does not increase with the simulation time, as the difference at 72 forecast hours is similar to that of 24 forecast hours. The WRF simulations coupled with FC networks (models A and B) have the worst performance, with a much larger difference from the original WRF simulation than WRF simulations coupled with Bi-LSTM models (models D and E) over the entire domain. Model B is more accurate than model A in offline settings but performs much worse than model A in the online evaluation. The MAD of $T_{2\,\mathrm{m}}$ and $\mathrm{WS}_{10\,\mathrm{m}}$ is greater than 5.6 and 1.3, respectively, for model B, while the MAD of $T_{2\,\mathrm{m}}$ and $\mathrm{WS}_{10\,\mathrm{m}}$ is slightly less

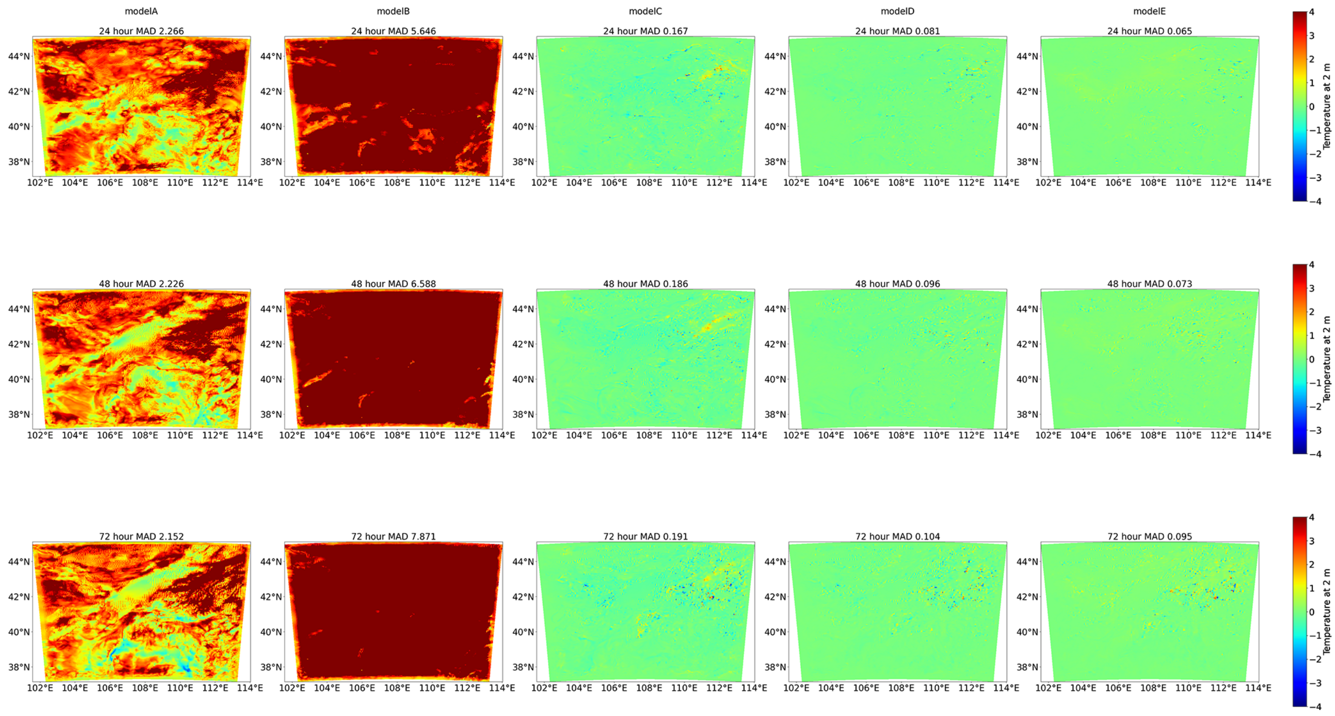

Figure 7. Differences in temperature at 2 m at forecast horizons of 24, 48, and 72 h between WRF simulations coupled with ML models A, B, C, D, and E and the WRF simulation with the original RRTMG schemes for radiation emulation. The title of each figure indicates the mean absolute difference (MAD) of the WRF domain.

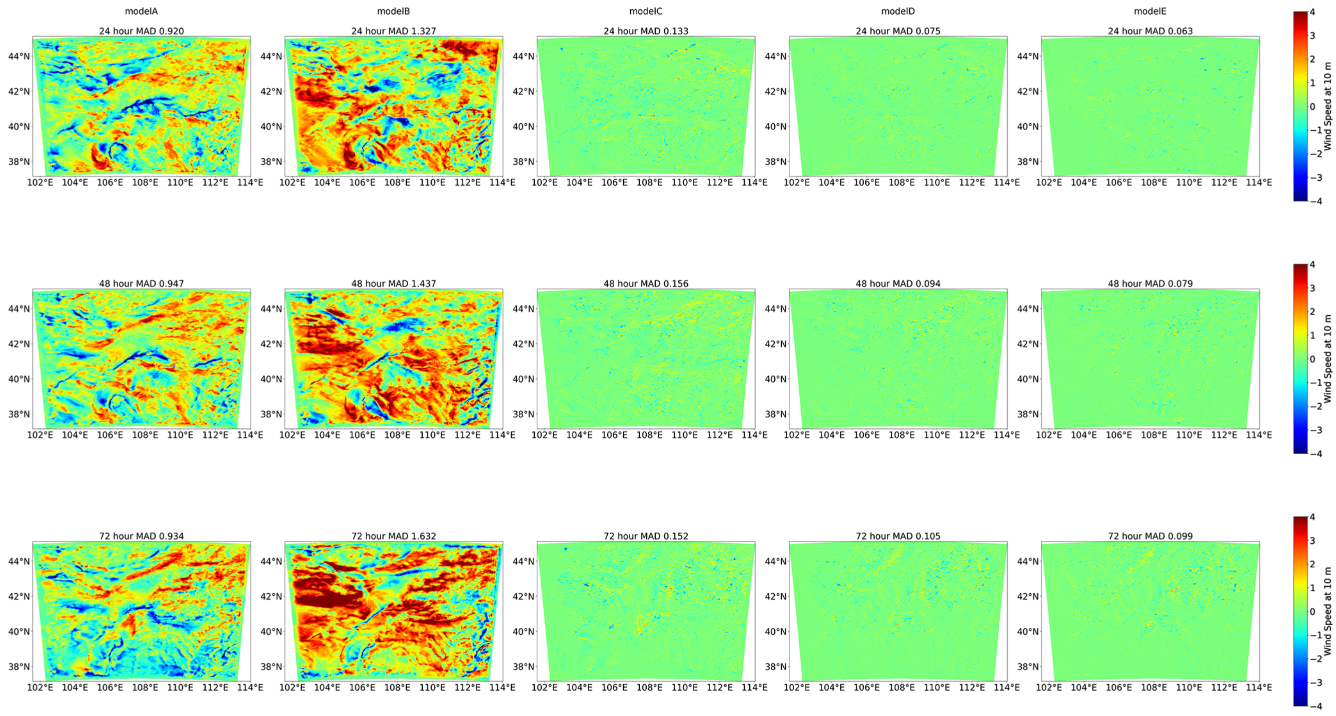

Figure 8. Differences in wind speed at 10 m at forecast horizons of 24, 48, and 72 h between WRF simulations coupled with ML models A, B, C, D, and E and the WRF simulation with the original RRTMG schemes for radiation emulation.

than 2.3 and 1.0, respectively, for models A. Model C performs best among all the three FC networks, with the MAD slightly worse than the MAD of models D and E. Models D and E have a similar spatial distribution of difference, although model E has about 10 times the number of parameters as model D and a slightly higher accuracy in offline validation.

Overall, model E demonstrates the best performance, with the domain-averaged MAD smaller than 0.10 for both temperature at 2 m and wind speed at 10 m. Both model D and model E show a comparable forecast with the original WRF simulation over the entire domain. Overall, model D is more suitable to replace the original RRTMG schemes as it is more computationally efficient than model E. Using model D with GPU as inference processors is about 7 times faster.

In summary, the WRF–ML coupler enables all the ML-based radiation emulators to couple with the WRF models efficiently. Moreover, the WRF simulations coupled with ML models are run successfully using CPU or GPU for inference. Furthermore, the WRF–ML coupler can be a valuable tool for researchers to evaluate their ML-based parameterization in online settings.

6 Summary and conclusions

As demonstrated in many previous studies, emulating the subgrid physics using ML models has the potential to be faster or, by training on observations or detailed high-resolution models, even more accurate than the conventional parameterization schemes. However, the fact that ML models are commonly implemented using Python while the NWP models are coded in Fortran makes it challenging to implement the ML-based parameterization in NWP models for operational applications. Previously, researchers usually hard-coded the operations of NNs into Fortran, which is time-consuming and prone to error as even minor changes to the ML model require rewriting the Fortran code. Some researchers used simple Fortran-based NN libraries, which could convert existing models trained in Python to models usable in the Fortran framework. However, these methods are only feasible for applying simple, dense, layer-based NNs. In this paper, we propose a WRF–ML coupler that provides researchers with the tool to implement ML models into NWP models with minimal effort. The advantage of this coupler is that more complex and advanced ML model structures are supported, as it can directly communicate with Python-based modules. The coupler is made open source in the hope that more and more researchers can integrate the ML-based parametrization to improve the traditional NWP models for more accurate weather or climate forecast. In addition, the methodologies also allow the integration of any Python scripts into the WRF model. An example of coupling the primitive Python subroutines into the WRF code is also provided in the code repository.

Furthermore, the capability of the WRF–ML coupler was demonstrated by coupling the ML-based radiation parameterization into the WRF model. It was also illustrated that ML-based parameterizations' computational efficiency and accuracy should be balanced to achieve the best overall performance for operational applications.

Code and data availability. The source code and data used in this are available at https://doi.org/10.5281/zenodo.7407487 (Zhong et al., 2022).

Author contributions. YY trained the machine learning models and calculated the statistics of models' offline performance. XZ, YW, LX, and ZM modified the WRF source code and developed the WRF–ML coupler. XZ conducted the WRF model simulations coupled with the original RRTMG schemes for ML model training and offline evaluation. XZ and ZM conducted the WRF simulations coupled with ML-based radiation emulators. XZ wrote, reviewed, and edited the original draft; ZW supervised and supported this research and gave valuable opinions. All of the authors have contributed to and agreed to the published version of the paper.

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

Financial support. This research has been supported by the Zhejiang Science and Technology Program under grant no. 2021C01017.

Review statement. This paper was edited by Klaus Klingmüller and reviewed by Peter Ukkonen and Sergey Osipov.

References

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.

Bougeault, P. and Lacarrère, P.: Parameterization of orographic induced turbulence in a mesobeta scale model, Mon. Weather Rev., 117, 1872–1890, https://doi.org/10.1175/1520-0493(1989)117<1872:POOITI>2.0.CO;2, 1989.

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., Chéruy, F., Scott, N., and Chédin, A.: A neural network approach for a fast and accurate computation of a longwave radiative budget, J. Appl. Meteorol., 37, 1385–1397, 1998.

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, 2000.

Clough, S., Shephard, M., Mlawer, E., Delamere, J., Iacono, M., Cady-Pereira, K., Boukabara, S., and Brown, P.: Atmospheric radiative transfer modeling: a summary of the AER codes, J. Quant. Spectrosc. Ra., 91, 233–244, 2005.

Coiffier, J.: Fundamentals of Numerical Weather Prediction, 1st edn., Cambridge University Press, ISBN 9780511734458, https://doi.org/10.1017/CBO9780511734458, 2011.

Grimmond, C. S. B., Blackett, M., Best, M. J., Barlow, J., Baik, J.-J., Belcher, S. E., Bohnenstengel, S. I., Calmet, I., Chen, F., Dandou, A., Fortuniak, K., Gouvea, M. L., Hamdi, R., Hendry, M., Kawai, T., Kawamoto, Y., Kondo, H., Krayenhoff, E. S., Lee, S.-H., Loridan, T., Martilli, A., Masson, V., Miao, S., Oleson, K., Pigeon, G., Porson, A., Ryu, Y.-H., Salamanca, F., Shashua-Bar, L., Steeneveld, G.-J., Tombrou, M., Voogt, J., Young, D., and Zhang, N.: The international urban energy balance models comparison project: First results from phase 1, J. Appl. Meteorol. Clim., 49, 1268–1292, https://doi.org/10.1175/2010JAMC2354.1, 2010.

Grimmond, C. S. B., Blackett, M., Best, M. J., Baik, J.-J., Belcher, S. E., Bohnenstengel, S. I., Calmet, I., Chen, F., Coutts, A., Dandou, A., Fortuniak, K., Gouvea, M. L., Hamdi, R., Hendry, M., Kanda, M., Kawai, T., Kawamoto, Y., Kondo, H., Krayenhoff, E. S., Lee, S.-H., Loridan, T., Martilli, A., Masson, V., Miao, S., Oleson, K., Ooka, R., Pigeon, G., Porson, A., Ryu, Y.-H., Salamanca, F., Steeneveld, G.-J., Tombrou, M., Voogt, J., Young, D., and Zhang, N.: Initial results from Phase 2 of the international urban energy balance model comparison, Int. J. Climatol., 31, 244–272, https://doi.org/10.1002/joc.2227, 2011.

Grundner, A., Beucler, T., Iglesias-suarez, F., Gentine, P., Giorgetta, M. A., and Eyring, V.: Deep Learning Based Cloud Cover Parameterization for ICON, arXiv [preprint], https://doi.org/10.48550/arXiv.2112.11317, 21 December 2021.

Gulli, A. and Pal, S.: Deep learning with Keras, 1st edn., Packt Publishing Ltd, ISBN 9781787128422, 2017.

Hatfield, S., Chantry, M., Dueben, P., Lopez, P., Geer, A., and Palmer, T.: Building Tangent-Linear and Adjoint Models for Data Assimilation With Neural Networks, J. Adv. Model. Earth Sy., 13, e2021MS002521, https://doi.org/10.1029/2021MS002521, 2021.

Iacono, M. J., Delamere, J. S., Mlawer, E. J., Shephard, M. W., Clough, S. A., and Collins, W. D.: Radiative forcing by long-lived greenhouse gases: Calculations with the AER radiative transfer models, J. Geophys. Res.-Atmos., 113, D13103, https://doi.org/10.1029/2008JD009944, 2008.

Janisková, M., Mahfouf, J. F., Morcrette, J. J., and Chevallier, F.: Linearized radiation and cloud schemes in the ECMWF model: Development and evaluation, Q. J. Roy. Meteor. Soc., 128, 1505–1527, 2002.

Krasnopolsky, V., Fox-Rabinovitz, M., Hou, Y., Lord, S., and Belochitski, A.: Accurate and fast neural network emulations of model radiation for the NCEP coupled climate forecast system: Climate simulations and seasonal predictions, Mon. Weather Rev., 138, 1822–1842, 2010.

Krasnopolsky, V. M.: NN-TSV, NCEP neural network training and validation system; brief description of NN background and training software, NOAA, https://doi.org/10.7289/v5qr4v2z, 2014.

Mason, P. J.: Large-Eddy Simulation of the Convective Atmospheric Boundary Layer, J. Atmos. Sci., 46, 1492–1516, https://doi.org/10.1175/1520-0469(1989)046<1492:LESOTC>2.0.CO;2, 1989.

Meyer, D., Grimmond, S., Dueben, P., Hogan, R., and Reeuwijk, M. V.: Machine Learning Emulation of Urban Land Surface Processes, J. Adv. Model. Earth Sy., 14, e2021MS002744, https://doi.org/10.1029/2021MS002744, 2022a.

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, 2022b.

Ott, J., Pritchard, M., Best, N., Linstead, E., Curcic, M., and Baldi, P.: A Fortran-Keras Deep Learning Bridge for Scientific Computing, arXiv [preprint], https://doi.org/10.48550/arXiv.2004.10652, 14 April 2020.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A.: Automatic Differentiation in PyTorch, in: NIPS 2017 Workshop on Autodiff, Long Beach, California, USA and 9 December 2017, https://openreview.net/forum?id=BJJsrmfCZ (last access: 23 December 2022), 2017.

Pu, Z. and Kalnay, E.: Numerical Weather Prediction Basics: Models, Numerical Methods, and Data Assimilation, Springer Berlin Heidelberg, 67–97, https://doi.org/10.1007/978-3-642-40457-3_11-1, 2018.

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.

Schneider, T., Lan, S., Stuart, A., and Teixeira, J.: Earth System Modeling 2.0: A Blueprint for Models That Learn From Observations and Targeted High-Resolution Simulations, Geophys. Res. Lett., 44, 12396–12417, https://doi.org/10.1002/2017GL076101, 2017.

Smirnova, T. G., Brown, J. M., and Benjamin, S. G.: Performance of Different Soil Model Configurations in Simulating Ground Surface Temperature and Surface Fluxes, Mon. Weather Rev., 125, 1870–1884, https://doi.org/10.1175/1520-0493(1997)125<1870:PODSMC>2.0.CO;2, 1997.

Smirnova, T. G., Brown, J. M., Benjamin, S. G., and Kim, D.: Parameterization of cold-season processes in the MAPS land-surface scheme, J. Geophys. Res., 105, 4077–4086, https://doi.org/10.1029/1999JD901047, 2000.

Song, H.-J. and Roh, S.: Improved Weather Forecasting Using Neural Network Emulation for Radiation Parameterization, J. Adv. Model. Earth Sy., 13, e2021MS002609, https://doi.org/10.1029/2021MS002609, 2021.

Stensrud, D. J.: Parameterization schemes: keys to understanding numerical weather prediction models, 2nd edn., Cambridge University Press, ISBN 9780511812590, https://doi.org/10.1017/CBO9780511812590, 2013.

Thompson, G., Field, P. R., Rasmussen, R. M., and Hall, W. D.: Explicit Forecasts of Winter Precipitation Using an Improved Bulk Microphysics Scheme. Part II: Implementation of a New

Snow Parameterization, Mon. Weather Rev., 136, 5095–5115, https://doi.org/10.1175/2008MWR2387.1, 2008.

Ukkonen, P.: Exploring pathways to more accurate machine learning emulation of atmospheric radiative transfer, J. Adv. Model. Earth Sy., 14, e2021MS002875, https://doi.org/10.1029/2021MS002875, 2022.

Wang, J., Balaprakash, P., and Kotamarthi, R.: Fast domain-aware neural network emulation of a planetary boundary layer parameterization in a numerical weather forecast model, Geosci. Model Dev., 12, 4261–4274, https://doi.org/10.5194/gmd-12-4261-2019, 2019.

Wang, X., Han, Y., Xue, W., Yang, G., and Zhang, G. J.: Stable climate simulations using a realistic general circulation model with neural network parameterizations for atmospheric moist physics and radiation processes, Geosci. Model Dev., 15, 3923–3940, https://doi.org/10.5194/gmd-15-3923-2022, 2022.

Yao, Y., Zhong, X., Zheng, Y., and Wang, Z.: A Physics-Incorporated Deep Learning Framework for Parameterization of Atmospheric Radiative Transfer, J. Adv. Model. Earth Sy., in review, 2023.

Zhong, X., Ma, Z., Yao, Y., Xue, L., Wu, Y., and Wang, Z.: Code and Data for "A Bridge between WRF and Deep Learning Parameterization and Application on Deep Learning Parameterization of Atmospheric Radiative Transfer", Zenodo [code and data set], https://doi.org/10.5281/zenodo.7407487, 2022.