Emulation of cloud microphysics in a climate model

W. Andre Perkins[1], Noah D. Brenowitz[2],
Christopher S. Bretherton[1], Jacqueline M. Nugent[3]

[1]Allen Institute for Artificial Intelligence
[2]NVIDIA
[3]University of Washington

Key Points:

- We build an emulator to replace the Zhao-Carr Fortran microphysics scheme in FV3GFS
- The integrated emulator sustains high skill throughout a 1-year simulation
- Tailoring the ML architecture to the structure of the underlying scheme greatly improves the online behavior of the emulator

Corresponding author: W. Andre Perkins, `andrep@allenai.org`

Abstract

We present a machine learning based emulator of a microphysics scheme for condensation and precipitation processes (Zhao-Carr) used operationally in a global atmospheric forecast model (FV3GFS). Our tailored emulator architecture achieves high skill ($\geq$94%) in predicting condensate and precipitation amounts and maintains low global-average bias ($\leq$4%) for 1 year of continuous simulation when replacing the Fortran scheme. The stability and success of this emulator stems from key design decisions. By separating the emulation of condensation and precipitation processes, we can better enforce physical priors such as mass conservation and locality of condensation, and the vertical dependence of precipitation falling downward, using specific network architectures. An activity classifier for condensation imitates the discrete-continuous nature of the Fortran microphysics outputs (i.e., tendencies are identically zero where the scheme is inactive, and condensate is zero where clouds are fully evaporated). A temperature-scaled conditional loss function ensures accurate condensate adjustments for a high dynamic range of cloud types (e.g., cold, low-condensate cirrus clouds or warm, condensate-rich clouds). Despite excellent overall performance, the emulator exhibits some deficiencies in the uppermost model levels, leading to biases in the stratosphere. The emulator also has short episodic skill dropouts in isolated grid columns and is computationally slower than the original Fortran scheme. Nonetheless, our challenges and strategies should be applicable to the emulation of other microphysical schemes. More broadly, our work demonstrates that with suitable physically motivated architectural choices, ML techniques can accurately emulate complex human-designed parameterizations of fast physical processes central to weather and climate models.

Plain Language Summary

In this study, we create computer code that uses machine learning to mimic a weather model's algorithm for handling how clouds form and rain falls. When used in the weather model to replace this algorithm, our machine learning code is highly accurate in simulations for a whole year. We achieve this by making smart code design choices. We split the code into two parts: one for cloud formation and one for rain and snow. This allows us to better build important aspects of these processes into the machine learning approach. For instance, clouds form where it is moist and evaporate when it gets dry, and rain and snow fall downward. Our code learns cloud behavior based on temperature to ensure it works both for cold, thin clouds high up in the sky and warm, thick clouds closer to the ground. Our work shows a path for suitably-designed machine learning code to eventually replace important parts of weather and climate models, but also that this path still requires careful human design respecting known physical principles.

1 Introduction

Atmospheric models combine fluid dynamics integrated on a discrete global grid with parameterizations of unresolved physical processes for weather and climate prediction. These parameterizations, encompassing phenomena such as cloud formation, precipitation, and radiative transfer, are crafted by experts and typically blend theoretical foundations with empirical relationships to capture interactions between various atmospheric processes. The ongoing development and refinement of these components require a careful balance between accuracy and efficiency to achieve high-fidelity simulations using limited computational resources.

Over the past few decades, advances in machine learning have led to substantial investments in computing facilities that combine more traditional CPU-based computing resources with accelerators such as GPUs. This shift in computational infrastructure has motivated the atmospheric modeling community to explore ways to capitalize on these newer resources to speed up simulations. The fluid dynamics algorithms implemented in atmospheric models can often be recoded for more efficient GPU computation using compiler directives or domain-specific language extensions (Dahm et al., 2023). However, the column-based physics parameterizations often involve more complex logic and data dependences that do not naturally fit into this paradigm.

An alternative approach to accelerating the physical components of atmospheric models is the creation of machine-learned emulators. Emulators are machine learning (ML) models trained directly on the inputs and outputs of a specific component, aiming to provide a seamless replacement of the original scheme. This strategy offers a natural path to speed up model operation on accelerator-based compute resources, which are optimized to run ML workloads. Consequently, most emulation studies have focused on radiative transfer (Chevallier et al., 1998; Krasnopolsky et al., 2005, 2010; Veerman et al., 2021; Ukkonen et al., 2020), the most expensive subcomponent in the typical atmospheric physics suite. However, recent studies have also emulated deep convection (O'Gorman & Dwyer, 2018), gravity wave drag (Chantry et al., 2021), atmospheric chemistry (Keller & Evans, 2019; Kelp et al., 2022; Schreck et al., 2022), and details of the warm rain process (Gettelman et al., 2021).

Emulation also serves as an excellent test bed for ML approaches that aim to improve on existing physical parameterizations, such as those using fine-resolution data to train corrective ML models (e.g., Brenowitz & Bretherton, 2019; Rasp et al., 2018; Yuval & O'Gorman, 2020; Bretherton et al., 2022). Typically, these learn improvements to the combined suite of physical parameterizations, e.g. radiation, microphysics, turbulence and surface exchange, cumulus convection and orographic drag. Emulation of individual component physical processes is clearly posed as a supervised learning task, so it can be used to explore skill bounds, quirks, and optimal architectural choices for emulating an entire parameterization suite.

The cloud microphysics scheme plays a central role in atmospheric modeling, managing rapid phase changes such as condensation, evaporation, and precipitation. It is tightly coupled to the model dynamics through latent heat release. We are not aware of past studies using ML to emulate an entire microphysics scheme, perhaps due to its lower computational cost compared to radiation. Nevertheless, it is a key part of emulating the combined physical parameterization suite and exposes a variety of ML challenges that are relevant to that broader problem. It is also a fast-acting process, producing localized atmosphere heating and drying tendencies that are much larger than for radiation. Thus, emulation of a representative microphysics scheme is a worthy complement to emulation of radiation parameterizations. It can provide valuable insights into the potential and challenges of ML emulators of atmospheric physical processes.

In this work, we train an ML model to emulate the Zhao and Carr (1997, ZC) microphysics scheme. This scheme was used for many years in the Global Forecast System (GFS) model by the U. S. National Centers for Environmental Prediction (NCEP). Here, it is included in a recent version of GFS that uses the FV3 dynamical core (Harris & Lin, 2013), which we call the FV3GFS global atmospheric model. The ZC scheme, with only one prognostic condensate variable, seemed to be a simple machine learning target. However, for a variety of reasons, developing a successful emulator of this scheme proved more challenging than anticipated, and required several architectural choices relevant to emulating other more complex microphysical parameterizations with many more prognostic hydrometeor types.

In Section 2, we describe the emulator architecture, training data, and integration into the FV3GFS model. In Section 3, we demonstrate that the emulator serves as a stable, skillful replacement to the original Fortran Zhao-Carr microphysics scheme, with low global average bias for at least 1 year of simulation. Despite impressive overall performance, the emulator induces regional biases in the uppermost model levels— in our experience, a relatively common online issue with ML integrated as one component in conventional atmospheric models (e.g., Brenowitz & Bretherton, 2019; Clark et al., 2022). In Section 4, we discuss the major decisions that influenced the emulator's performance and address some remaining challenges and limitations of our approach.

In accordance with AGU's AI tool policy, the authors acknowledge the use of OpenAI's ChatGPT-4 tool to help edit the manuscript draft for clarity, conciseness, and grammatical correctness. All suggestions provided by the AI tool were reviewed and edited by the authors for correctness and consistency. The plain language summary was generated by prompting the tool for a generally accessible version of our written abstract and then edited by the authors.

2 Methods

In this work, we utilize the FV3GFS global atmospheric model (Harris & Lin, 2013), which is currently used by NOAA for operational weather forecasting. FV3GFS combines the FV3 nonhydrostatic finite-volume dynamical core with a suite of physical parameterizations developed for the Global Forecast System (GFS). For the simulations presented here, the FV3GFS model is run on a C48 cubed-sphere grid (approximately 200 km horizontal grid spacing) with 79 vertical levels.

Within FV3GFS, we target the emulation of the Zhao-Carr (ZC) microphysics (Zhao & Carr, 1997), which was used in the operational version of GFS until 2018. The ZC microphysics scheme predicts changes in cloud condensate, precipitation, and the associated heating and moistening rates at each grid point in a vertical column, based on state inputs. The scheme divides the prediction into two subroutines: one calculating the local condensate change via grid-scale condensation (gscond) and the other calculating column precipitation and associated condensate adjustments (precpd). Figure 1 shows a graphical depiction of the information flow through the ZC microphysics subroutines. The scheme diagnoses the phase partitioning of cloud condensate into liquid and ice at each step based on temperature and the presence of overlying ice cloud. Furthermore, it diagnoses the downward precipitation flux and its phase partitioning into rain and snow at each grid level during each time step. Appendix A gives further details.

The ZC scheme initially seemed appealing for ML emulation due to its simplicity, featuring only a single prognostic hydrometeor type: the cloud water mixing ratio. Despite the initial appearance of simplicity, the schematic (Fig. 1) illustrates that the ZC scheme is architecturally more complex than we anticipated due to the implicit dependence on the column thermodynamic state sampled within the previous time step. Furthermore, vertically and temporally nonlocal phase partitioning of condensate does not appear as an explicit output of the scheme, despite its use by other parameterizations. These subtleties add considerable time-consuming challenge to the accurate ML emulation of the ZC scheme.

To emulate the ZC scheme, we employ hooks to interact with the Fortran model via the package `call_py_fort` (https://github.com/nbren12/call_py_fort). This package enables users to call functions and interact with selected Fortran state fields within an initialized Python environment, giving access to the comprehensive suite of ML and data tools available in Python and accelerating ML prototyping and testing.

2.1 Training Data

We generate the training dataset by initializing 30-day simulations from GFS analysis on the first day of each month in 2016, saving fields every 5 hours to sample the diurnal cycle. A list of all stored fields is shown in Table S1. We reserve three months of data for validation during training (February, June, and September). The training dataset

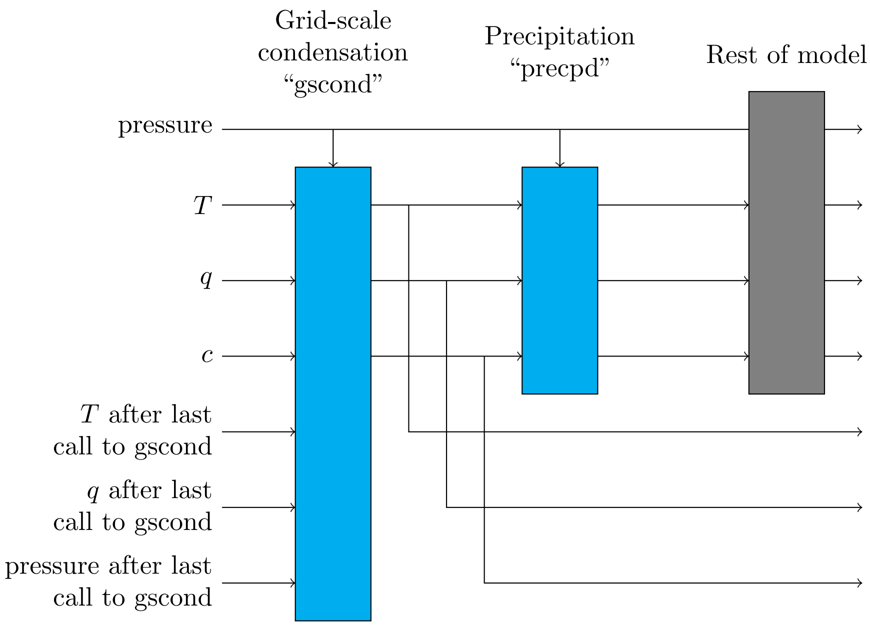

Figure 1. Information flow of the Zhao-Carr microphysics within FV3GFS for a single time step. Inputs of a given scheme are represented as inward arrows. The "after last call to gscond" inputs are used to compute a relative humidity tendency that encompasses the rest of the model and prepcd. This approach to computing the tendency effectively adds three new state variables to the model.

includes 1080 global snapshots consisting of $48^2 \times 6 = 13824$ atmospheric columns, totaling nearly 15 million samples.

From the saved training data, we derive the target increments for the ZC microphysics that we seek to emulate. The total change, denoted as $\Delta = \Delta_g + \Delta_p$, is the sum of the two subroutine updates from gscond (Δ_g) and precpd (Δ_p). Both gscond and precpd calculate updates for temperature (T), specific humidity (q), and the cloud water mixing ratio (c); precpd also diagnoses the amount of surface precipitation (P) during the time step. We note that the use of tendencies in this manuscript refers to the subroutine increment divided by the model time step (15 minutes).

Figure 2 displays an example transect of tendencies of the target data for clouds and humidity along the 100°W meridian. The gscond cloud water tendency (Fig. 2a; $\Delta_g c$) can be positive (condensation) or negative (evaporation), depending on local thermodynamic state. Active regions in this snapshot include the boundary layer of the subtropical Pacific and free-tropospheric weather features (e.g., convection or frontal zones) over land. Because cloud water tendency involves a phase change between water vapor and condensate, the corresponding tendencies of temperature ($\Delta_g T$) and specific humidity ($\Delta_g q$) exhibit similar patterns to the cloud water tendency. The gscond tendencies for these three fields are fully determined by grid-local thermodynamic state, with the exception of one vertically non-local flag, which influences the diagnostic decomposition between liquid and ice clouds and the resulting latent heating tendency. That flag indicates whether mixed-phase clouds with temperatures between 0° and -15°C are overlaid by contiguous ice cloud colder than -15°C.

The corresponding precpd condensate tendency transect (Fig. 2b; $\Delta_p c$) shows losses due to autoconversion of thicker clouds to precipitation. Regions of positive precpd va-

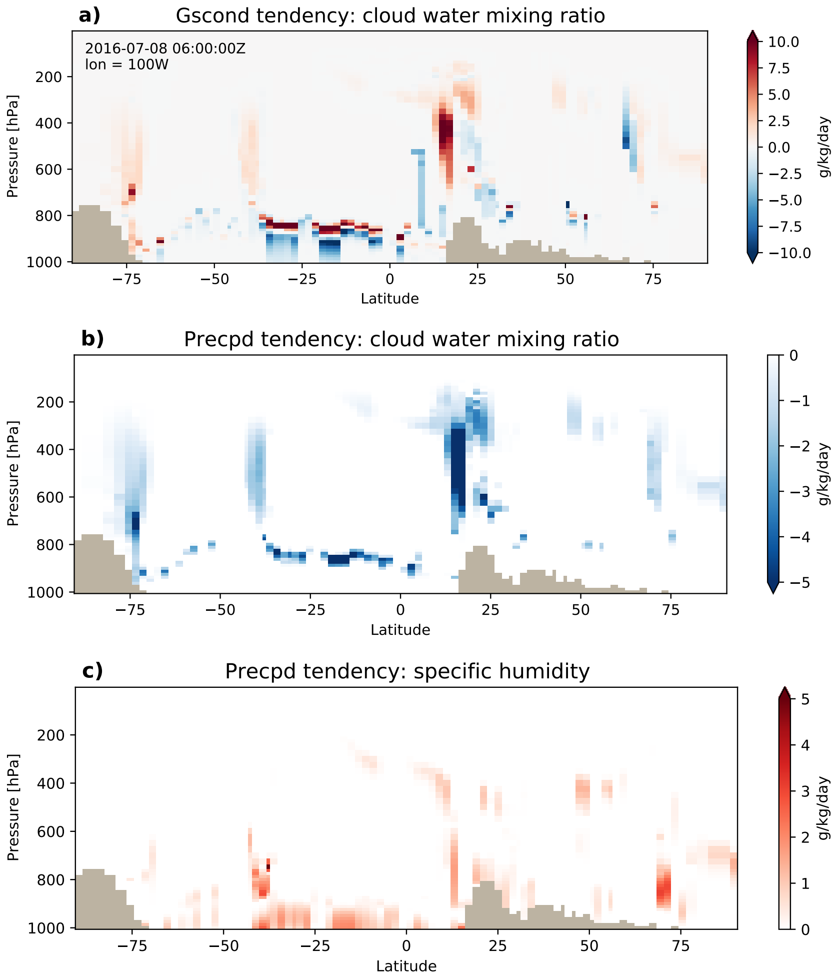

Figure 2. Latitude–pressure transects along longitude 100°W for a sample Zhao-Carr microphysics step on July 8th, 2016 at 06Z showing: (a) the condensation rate from gscond, (b) the conversion rate of cloud to precipitation in precpd, and (c) the precipitation evaporation rate in precpd. Transect data has been interpolated to pressure levels from model levels for presentation.

por tendency (Fig. 2c; $\Delta_p q$) are due to the evaporation of precipitation falling from overlying grid layers.

These transects highlight two general challenges for emulating microphysics. First, the microphysics scheme is not active at the majority of grid points. It produces a range of adjustments to the state fields where clouds or precipitation are present, but elsewhere, the tendencies should be exactly zero. Second, the condensation scheme can generate large condensate increments throughout the troposphere despite the humidity being orders of magnitude smaller in the upper troposphere than near the surface.

Some other general considerations are also important for ML microphysics emulation. For instance, clouds are very sensitive to relative humidity. A small error in predicted water vapor or temperature can significantly impact clouds and precipitation. Second, cirrus clouds with small condensate mixing ratios can be as radiatively important as liquid water clouds with hundred-fold higher condensate mixing ratios. Thus, an ML emulator must accurately predict a large range of condensate tendencies to skillfully re-

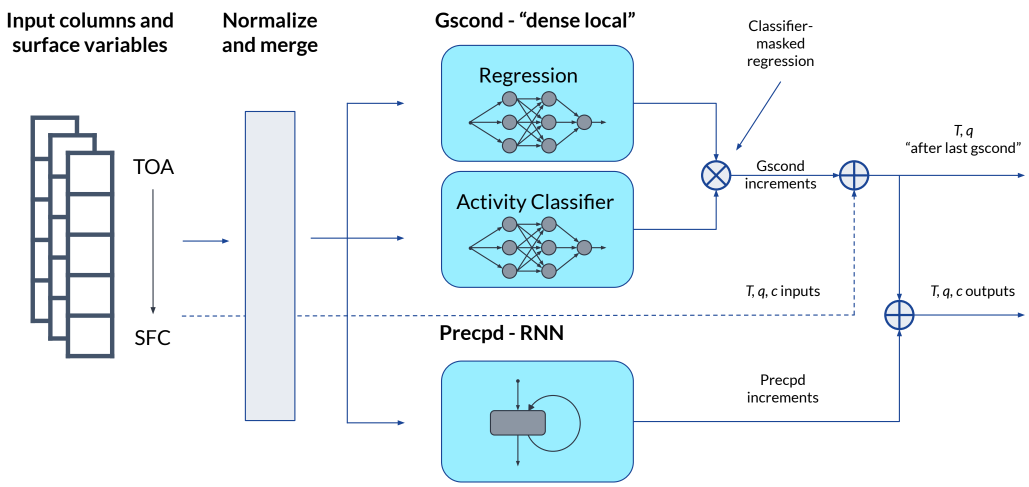

Figure 3. A schematic of the ZC microphysics emulation architecture.

produce the original model's climate. Third, complete cloud evaporation/sublimation is common; to obtain this outcome in a model time step requires the condensate tendency to exactly remove all cloud condensate in a grid box. Lastly, microphysical tendencies are a combination of local (e.g., condensation) and non-local (e.g., precipitation) processes and constraints. An emulation scheme must replicate these dependencies to yield accurate and physically consistent results.

These factors heavily influenced the final design of our emulation methodology, which we detail in the following section. We elaborate on the sensitivity of results to these choices and discuss remaining challenges in Section 4.

2.2 Emulator Architecture

The emulation model architecture is shown in Figure 3. Separate emulators for gscond and precpd take a total of 13 input variables, including the same set of inputs as the Fortran ZC scheme: T, q, c, and surface pressure as well as the "after last gscond" values of T, q, and surface pressure. We provide additional inputs of air pressure and pressure thickness of the atmospheric layer, as well as derived inputs of relative humidity (RH), and log-scaled q, c, and q after last gscond. Each input is normalized:

$$x'_j = (x_j - \mu_j)/\sigma \tag{1}$$

and combined to form input channels for the emulation models. The mean, μ_j, is a sample mean at level j using 150,000 random columns from the training data. The scaling factor, σ, is calculated using the standard deviation over all per-level centered $(x_j - \mu_j)$ values in the same sample. This scaling enhances training stability and conveniently downweights inputs from the upper levels, where the microphysics scheme is less active. Surface variables are normalized as a single level and then broadcast to 79 levels when merged into model inputs to simplify general usage. The same input data are passed to all three of the emulator subcomponents.

2.2.1 Condensation emulator

In the condensation subroutine (gscond), net condensation $\Delta_g c$ at a given point in an atmospheric column is physically determined by the thermodynamic inputs at that same level, a property we refer to as grid-point locality. The gscond emulator takes advantage of this property by applying a single MLP to each grid point, which we refer to as a dense-local model. The MLP is 2 layers of 256 channels, each with ReLU activation. It takes in 79-level $\times$ 13-channel inputs, applies the model to each level, and outputs a single column (79×1) through a linear readout layer. We train the gscond dense-local regressor for 50 epochs using the Adam optimizer with a learning rate of 0.0001. We use a mean squared error (MSE; Eq. 2) based loss (Eq. 3).

$$\mathrm{MSE}(a,\, b) = \frac{1}{N}\sum_{i=1}^{N}(a_i - b_i)^2 \tag{2}$$

$$L = \mathrm{MSE}(\widetilde{y},\, \hat{y}) + \lambda \cdot \mathrm{MSE}(c_g',\, \hat{c}_g') \tag{3}$$

$$\widetilde{y} = \frac{\Delta_g c - \widetilde{\mu}(T_{in})}{\widetilde{\sigma}(T_{in})} \tag{4}$$

$$\hat{y} = f(x) \tag{5}$$

$$c_g = \Delta_g c + c_{in} \tag{6}$$

$$\hat{c_g} = \hat{y}\sigma(T_{in}) + \mu(T_{in}) + c_{in} \tag{7}$$

The target increment in the loss ($\widetilde{y}$, Eq. 4) is conditionally scaled due to a physical expectation that cloud properties depend strongly on temperature (Fig. S1). To accurately emulate cold cirrus clouds, which typically have little condensate and correspondingly small condensate increments, and also emulate warm liquid clouds, which can have hundred-fold larger condensate increments, the loss function normalizes to be sensitive in both cases. The scaling terms for the mean $\widetilde{\mu}(T_{in})$ and standard deviation $\widetilde{\sigma}(T_{in})$ represent a piecewise interpolation based on the input temperature T_{in}. We compute the underlying interpolation function by calculating binned mean and standard deviation values after grouping samples of $\Delta_g c$ into 50 linearly-spaced bins between the minimum and maximum input temperature. We optimize the gscond emulator $\hat{y} = f(x)$ to predict temperature-scaled increments ($\widetilde{y}$) as functions of the grid point features x. These increments are descaled into a predicted post-gscond condensate amount ($\hat{c}_g$, Eq. 7) by adding the de-scaled increment to the input condensate amount. We include a post-gscond condensate MSE in the loss (Eq. 3) using the normalized condensate amounts ($c_g', \hat{c}_g'$) scaled by $\lambda = 50000$ to make the loss contribution $O(1)$. The addition of the final condensate value to the loss function improves validation MSE for the unscaled condensate increment by over 80%. This likely happens because the final condensate term gives additional weight to warm-cloud condensation. The remaining state increments for T and q are determined at runtime from the predicted $\Delta_g c$ value (see Section 2.3).

We train an activity classifier to handle the mixed discrete-continuous nature of the condensation scheme, i.e., the need to force the emulator prediction to either (i) zero tendency when there should be no cloud change during the time step, or (ii) the exact tendency to fully evaporate cloud condensate present at the beginning of the time step. The classifier model employs the same dense-local architecture as the regressor, but predicts four target variables to identify the following classes:

- $\Delta_g c = 0$,
- $c_g = 0$ and $\Delta_g c \neq 0$,
- $c_g \neq 0$ and $\Delta_g c > 0$, and
- $c_g \neq 0$ and $\Delta_g c < 0$.

The first two cases, corresponding to situations (i) and (ii) above, together usually account for 80% or more of the outcomes depending on the level (Fig. S2). During inference, the model constrains $\Delta_g c$ when the classifier identifies either of the first two cases. Otherwise, the regressor makes the condensate prediction. We train the classifier using categorical cross-entropy loss with the same hyperparameters as the regressor, except for an increased learning rate of 0.001. After training, the classifier is approximately 98% accurate over all classes and levels (Table S2). Please refer to Section 4.1 for a more in-depth discussion on the impacts of the conditional loss function and activity classifier.

2.2.2 Precipitation emulator

The diagnostic precipitation scheme (precpd) generates precipitation through autoconversion of cloud condensate in upper levels. The precipitation falls and can either evaporate in lower layers or reach the surface. To enforce this downward dependence in the precpd emulator by construction, we use a recurrent neural network (RNN) that recurses over vertical layers starting at the top of atmosphere (see schematic in Fig. S3). A single RNN layer,

$$h_{j+1} = (W_h h_j + W_x x_j + b)^+, \tag{8}$$

uses the same normalized inputs, x'_j, as the gscond emulator where $j \in [0, 79)$ and $j = 0$ is the top of the atmosphere. In this form, h_j is the RNN hidden state at level j, W_h represents trainable weights for the recursion on hidden state, W_x are the trainable weights for inputs, b is the bias, and $(\cdot)^+$ represents a ReLU activation function. We stack two hidden 256-channel layers followed by a level-independent linear readout layer ($\hat{y}_j = Ah_j + b$) to predict the increments $\Delta_p T$, $\Delta_p q$, and $\Delta_p c$. This construction ensures that only inputs x_i from levels at and above level j ($i \leq j$) can affect RNN predictions at level j. We embed additional constraints within the precpd emulator such that it converts clouds to precipitation ($\Delta_p c \leq 0$), that it evaporates precipitation ($\Delta_p q \geq 0$ and $\Delta_p T \leq 0$), and that the final cloud is non-negative ($c_p \geq 0$). The RNN loss includes the MSE for the normalized increments (using Eq. 1 instead of conditional normalization) and the MSE of the normalized post-precpd output for each variable scaled such that the individual contributions are $O(1)$. The surface precipitation rate (P) is diagnosed from the net loss in total column water at runtime using:

$$P = -\sum_{j=0}^{78} (\Delta_p c_j + \Delta_p q_j) \cdot \Delta p_j \, / \, g, \tag{9}$$

where for each level j, $(\Delta_p c_j + \Delta_p q_j)$ is the local water change due to autoconversion and evaporation, Δp_j is the input pressure thickness of the atmospheric layer, and g is gravity.

2.3 Prognostic runs

The utility of a microphysics emulator ultimately depends on its performance when used within the atmospheric model as a substitute for the human-designed parameterization it is trained to replace. Specifically, the emulator should not cause catastrophic model failures, it should consistently provide a skillful representation of the original microphysics behavior, and it should have a minimal impact on the integrated statistics (i.e., the climate) of the underlying model. To test this, we embed the ZC microphysics emulator in FV3GFS and run a series of prognostic tests using two model configurations: one with the emulator as the active microphysics scheme (online) and a baseline with the Fortran microphysics active (offline). In each case, we run the inactive component in a diagnostic mode ("piggybacked"; Grabowski, 2019) and save the resulting tendencies for comparison.

To evaluate the skill and climate impact of the emulated microphysics, we initialize 30-day simulations in each calendar month from February 2016 to January 2017. The initializations are taken from the end of the training data simulations, testing both model configurations on atmospheric states independent of the training data. We compute skill scores for all microphysics tendencies (ΔT, Δq, Δc; converted to a tendency by dividing increments by $\Delta t = 900$) and P using a modified R^2 score 1 - $\sum(\hat{y} - y)^2 / \sum y^2$. A score of 1 indicates a perfect emulation, while a value of 0 or lower indicates an emulator worse than a no-information prediction. We also compute the bias of the microphysics outputs and the atmospheric state over all levels and times of the 12 simulations. To assess long-term stability, we simulate a full year using the emulator in place of the ZC microphysics and check the global averages and bias for evidence of any climate drifts.

The last step in applying the emulator as part of an online simulation is to apply final physical limiters and constraints and generate the full set of outputs for the emulated ZC microphysics. For the gscond emulator, we compute the increments $\Delta_g T$ and $\Delta_g q$ through local conservation of the net condensation. First, we limit the net condensation based on moisture availability using:

$$\Delta_g c = \begin{cases} \max(-c_{in},\ \Delta_g c), & \text{if } \Delta_g c < 0 \\ \min(q_{in},\ \Delta_g c), & \text{if } \Delta_g c > 0 \end{cases}. \tag{10}$$

Then, the change in water vapor mirrors the change in condensate ($\Delta_g q = -\Delta_g c$) and the temperature change is determined via latent heating ($\Delta_g T = (L_v/c_p)\Delta_g c$), where L_v is the latent heat of vaporization and c_p is the specific heat of air at constant pressure. This is an approximation, as some phase changes in ZC occur between ice and vapor, releasing additional latent heat; however, these phase changes are not fully locally determined and our efforts to use a posthoc determination of ice cloud latent heating effects slightly degraded online emulator skill. For online application, we set the top 5 levels of gscond increments to zero since the ZC microphysics scheme is never active in those stratospheric levels and noise issues in ML-predicted condensate increments arise in these levels (see Section 4.2 for further discussion). Finally, we add the increments to the corresponding input state variable to obtain fields after gscond ($T_g = T_{in} + \Delta_g T$, $q_g = q_{in} + \Delta_g q$, and $c_g = c_{in} + \Delta_g c$).

The precpd increment constraints are directly integrated into the ML model as described earlier. We derive the surface precipitation (Eq. 9), and then add the precpd increments to the post-gscond values to generate the final scheme outputs ($T_p = T_g + \Delta_p T$, $q_p = q_g + \Delta_p q$, and $c_p = c_g + \Delta_p c$).

3 Results

We begin with the top-level results of our ZC emulation 30-day runs in Table 1. The offline skill scores for all emulated quantities are nearly perfect at ∼99%, with low root mean-square error (RMSE) values and biases that are 1–2 orders of magnitude smaller than the RMSE (i.e., a small component of the error).

Online skill is a strict test where deviations from a realistic physical state can cause the diagnostic Fortran microphysics to output large state adjustments or even crash. Nevertheless, when the emulator is used online, it maintains high skill scores with only a ∼1–5% reduction compared to the offline case. Predicted cloud water tendencies show the lowest average performance at 94%, which is still quite high for a sparse and highly sensitive tendency field. The corresponding tendency RMSEs of emulator tendencies vs. piggybacked Fortran tendencies are roughly double those of the offline configuration, ex-

	Offline			Online		
ZC Output	Skill score	RMSE	Bias	Skill Score	RMSE	Bias
ΔT [K/day]	0.99	0.42	-0.03	0.98	0.58	-0.02
Δq [mg/kg/day]	0.995	110	3.0	0.99	200	-1.1
Δc [mg/kg/day]	0.99	140	-1.0	0.94	330	-0.7
P [mm/day]	0.998	0.21	-0.02	0.97	0.77	0.02

Table 1. Skill metrics for the ZC microphysics emulator outputs compared to the Fortran microphysics outputs for the offline (Fortran driving) and online (emulator driving) configurations. All table metrics are calculated for twelve 30-day runs initialized at the start of each calendar month and then averaged together.

cept for P, where the tendency RMSE is nearly four times larger. The larger online error result is an expected outcome due to detrimental feedbacks between the model and the ML emulator that cannot be accounted for when using offline training. The biases remain small in the online case, suggesting no systematic breakdown of the emulator behavior from the diagnostic Fortran microphysics.

We compare the time-averaged atmospheric state averaged across the twelve 30-day online simulations with identically initialized baseline simulations to show that the emulator produces little mean-state drift when used in FV3GFS in place of the original ZC microphysics. Figure 4 depicts zonal averages of the online bias of the emulator-based simulation compared to the baseline simulation, which have been interpolated from model level to pressure coordinates to display biases at a true relative height. Table 2 gives global average area- and mass-weighted bias for selected output fields.

Cloud water is a key output of the microphysics scheme. Its zonal average mixing ratio (Fig. 4a, b) has the largest absolute bias near the surface in Antarctica, $\sim$6 mg/kg. This bias is relatively large for the characteristically cold,dry air there. Outside of the Antarctic, the cloud water biases are $\sim$3 mg/kg or less— a much smaller relative change from the baseline— and are generally positive, except for a negative bias in the tropical upper troposphere. The global-mean cloud water bias is small— 0.2 mg/kg, an approximately 2% increase compared to the baseline state (Table 2). These cloud changes result in $O(1\%)$ changes to the outgoing top-of-atmosphere longwave (-1.4 W/m^2) and shortwave radiation (+1.3 W/m^2), but in total the changes largely cancel out.

Figure 4d depicts the online bias in RH, which displays a small shift towards saturation in the middle-to-lower troposphere. The largest biases in RH (>10%) occur in the Antarctic upper atmosphere near the large gradient in drying near the tropopause. There are also similar albeit smaller positive RH biases in the tropics and Arctic tropopause regions. Overall, the global-mean RH shows a small positive bias of 0.8% (Table 2), congruent with the small positive cloud water bias.

The zonal average temperature has a small cold bias of up to -1.5 K in the high latitudes. Between 50°S–50°N, this bias is weakened or even slightly reversed at some pressures, but there is a thin layer of warm bias up to 1 K near the tropopause. The zonal temperature biases largely cancel out when averaged globally over the 30-day runs (Table 2).

Lastly, the total surface precipitation (emulated ZC microphysics + convective sources) has a slight positive bias of 0.03 mm/day, a 1% increase from the baseline simulation (Table 2). Fig. 5a depicts the online zonal average surface precipitation just from the ZC microphysics component. The emulated ZC precipitation production is nearly identical to the baseline simulation owing to the high emulation skill of Δq and Δc, but produces

Field	Bias	Baseline mean
Air temperature [K]	-0.1	251
Specific humidity [mg/kg]	-0.7	2590
Relative humidity [%]	0.8	45.5
Cloud water [mg/kg]	0.2	9.6
Total surface precipitation [mm/day]	0.03	3.04
Upward shortwave at TOA [W/m^2]	1.3	91.9
Upward longwave at TOA [W/m^2]	-1.4	237
Total outgoing radiation at TOA [W/m^2]	-0.06	329

Table 2. Global average online biases and baseline means for selected state fields averaged over all 30-day simulations.

	Online skill	
ZC Output	1-year run	30-day runs avg.
ΔT	0.98	0.98
Δq	0.98	0.99
Δc	0.94	0.94
P	0.97	0.97

Table 3. Online skill score for 1-year online simulation compared against the skill scores averaged across the twelve 30-day runs initialized across the calendar year.

0.02 mm/day less global precipitation than the baseline ZC scheme. This bias must mostly be associated with state drift rather than offline emulator errors, because the piggybacked Fortran ZC scheme, which is applied to the online emulator state, diagnoses slightly less precipitation than the online emulator, especially in the Northern Hemisphere storm track. The Fortran convection parameterization also responds to the slight emulator-induced state changes by producing a global mean convective precipitation increase of 0.05 mm/day.

The instantaneous precipitation-rate distribution based on all grid columns and sampling times (Fig. 5b) corroborates this analysis. It shows that the emulator overproduces light precipitation ($<$ 0.1 mm/day) compared to the piggybacked Fortran scheme, but these two schemes agree well at most higher precipitation rates, and their small discrepancies don't explain the online emulator differences from the baseline simulation. Instead, the largest precipitation rate bins ($\sim$100 mm/day) suggest that the online emulator-driven simulation shifts to fewer states that support heavy precipitation events compared to the baseline simulation.

3.1 1-year continuous simulation

The monthly-initialized runs show the embedded ZC emulator is stable for at least 30 days during all calendar months of the year, with low biases. To further explore the long-term fidelity of emulator-based simulations, we present results from a continuous 1-year integration starting in July 2016. We ran two simulations, one masking only the top 5 levels of the gscond increments (i.e., setting the increments to 0) and the other masking the top 5 levels of both gscond and precpd increments. We found adding the mask to the top 5 levels of the precpd scheme reduced the number and severity of transient tendency skill dropouts (Fig. B1) for the 1-year simulation. Both online simulations ran with online emulation for the full year. We present results for the top 5 gscond and precpd increment configuration due to better performance. We discuss the unresolved sensitivity of the emulator to the upper levels in Section 4.2.

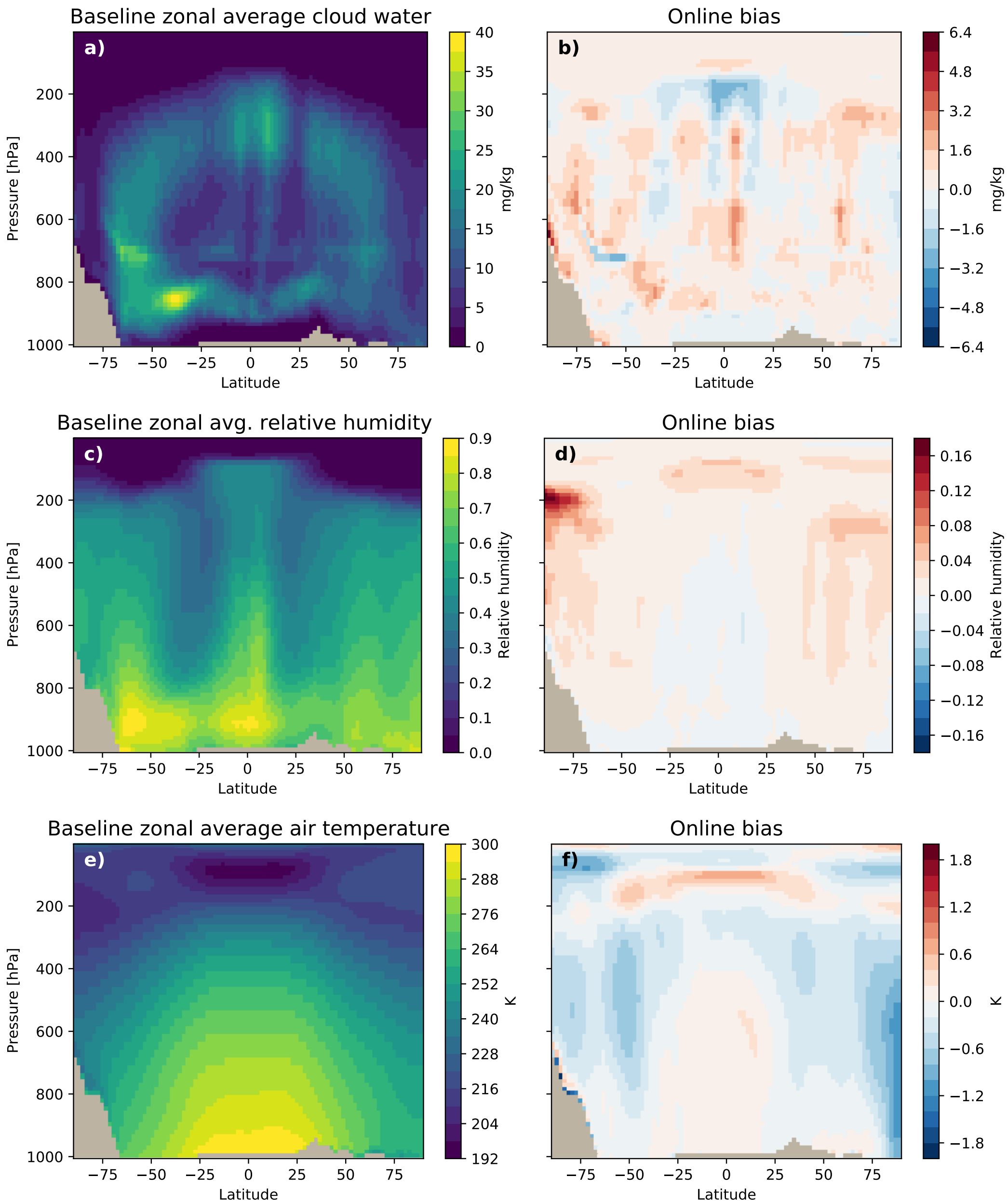

Figure 4. Latitude–pressure sections of zonal and time average state from baseline Fortran simulations (left) and online bias of simulation using the emulator (right) for cloud water mixing ratio (a, b), relative humidity (c, d), and air temperature (e, f). Averages are over twelve 30-day simulations initialized in each month of the calendar year, using values vertically interpolated from model levels.

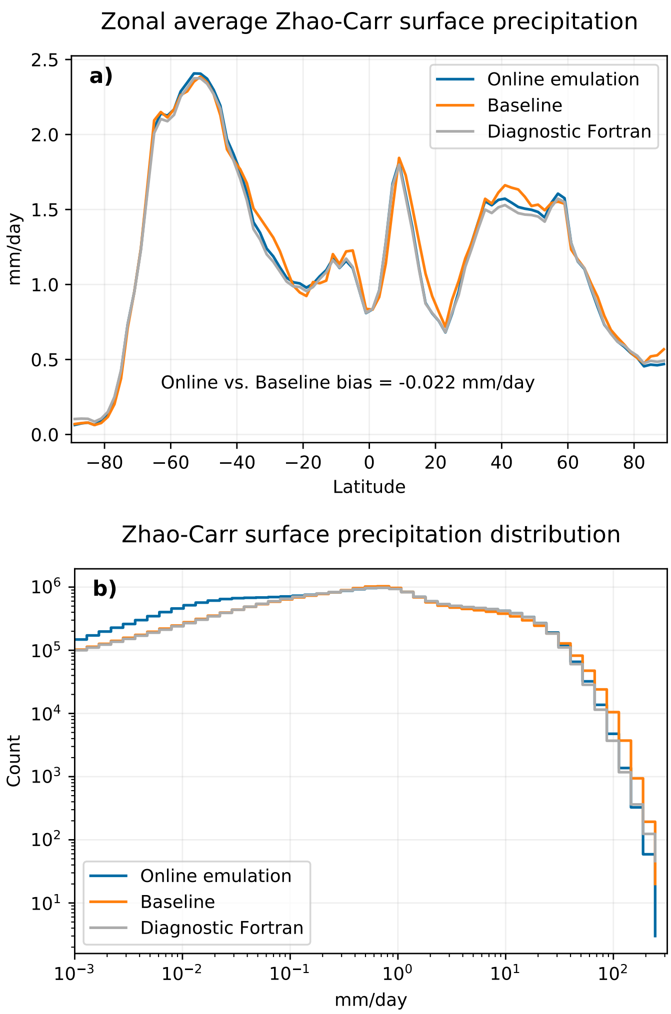

Figure 5. (a) Zonal average surface precipitation rate from ZC microphysics compared between the online emulator (blue) baseline Fortran (orange) and diagnostic Fortran microphysics (grey), which is generated diagnostically using inputs from the online emulation state. (b) Surface precipitation rate distribution compared between the same schemes. Shown quantities are calculated from twelve 30-day simulations initialized at the beginning of each calendar month.

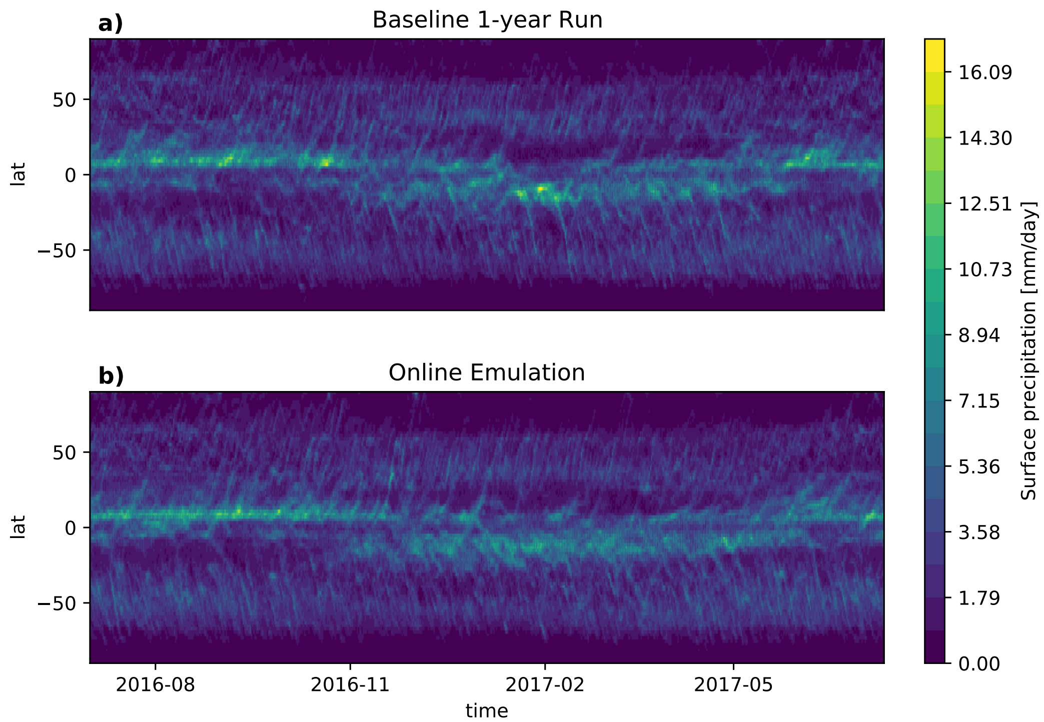

Figure 6. Time–latitude plots of the instantanous surface precipitation rate saved every 3 hours from the 1-year (a) baseline and (b) online emulation simulations.

The online skill metrics for the 1-year continuous run are, reassuringly, almost identical to the average of the 30-day runs (Table 3). A time–latitude plot of total surface precipitation (Fig. 6) compares the baseline and online emulation runs, demonstrating the emulation retains the spatiotemporal character of the baseline precipitation (and precipitating clouds by proxy) throughout the seasonal cycle. A slight reduction in the largest precipitation events for the online emulation is apparent in the tropics; we already noted this issue for the month-long simulations in Fig. 5b. Some global-annual-average biases (Table 4) are somewhat larger than in the 30-day runs: T (-0.3 K), RH (1.9%), and net TOA outgoing radiation (-0.4 W/m^2; the difference of a -2.1 W/m^2 outgoing longwave bias and a 1.6 W/m^2 reflected shortwave bias). Absolute cloud water and surface precipitation biases remain similar to those of the 30-day runs. Cloud water and RH have the largest relative bias from the baseline simulation at $\sim$4%, respectively.

The zonal average biases of T and RH from the 1-year emulator-based simulation are very small in the troposphere but become more significant in the polar stratosphere (Fig. 7). In this region, large negative cold biases (as low as -8 K) are co-located with positive RH biases up to 30%. The temperature bias appears within the first few months of the simulation and stabilizes for the rest of the simulation. We further investigated these biases and found that both the gscond and precpd emulators have deficiencies in the dry, cold polar stratosphere. Within a few hours after the start of the simulation, the gscond emulator produces too much condensate because the emulator predicts condensation for what the Fortran piggybacked microphysics diagnoses should mostly be evaporation at marginal relative humidities (40–50%; Fig. S4). We have confirmed that the gscond bias drift is unrelated to precpd or the classifier. We hypothesize that the tendency drift is likely related to a subtle online shift in some characteristics of the input distribution specific to this region.

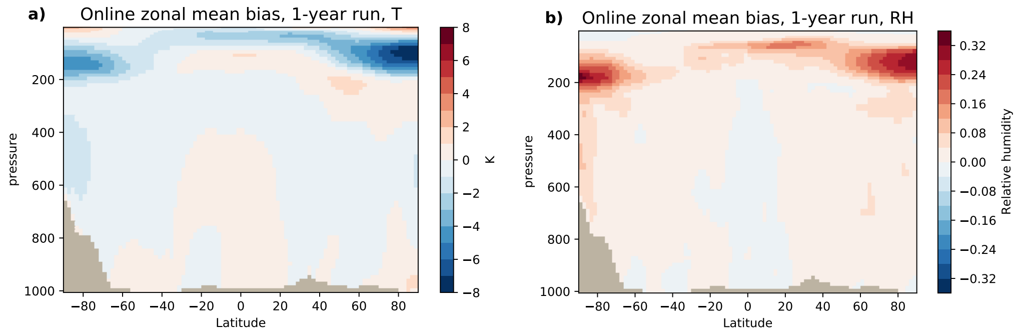

Figure 7. Zonal mean bias of the 1-year online emulation simulation for (a) temperature and (b) relative humidity.

Field	Bias	Baseline mean
Air temperature [K]	-0.3	247
Specific humidity [mg/kg]	17.2	2680
Relative humidity [%]	1.9	45.6
Cloud water [mg/kg]	0.2	7.6
Surface precipitation [mm/day]	0.03	3.03
Upward shortwave at TOA [W/m^2]	1.6	92.1
Upward longwave at TOA [W/m^2]	-2.1	237
Total outgoing radiation at TOA [W/m^2]	-0.44	329

Table 4. As in Table 2 but for the 1-year simulation.

The precpd emulator's shortcomings in the polar stratosphere are evident from offline diagnosis. Specifically, errors from the emulator's noise floor produce evaporation despite no falling precipitation (Fig. S5) in this region. This is a particular failing of the the single-scaling loss normalization (Eq. 1), where optimization fails to minimize the large relative errors in the polar stratosphere. The errors produce a directional bias due to constraints imposed in the model architecture ($\Delta_p q > 0$ and $\Delta_p T < 0$) and a lack of enforced conservation. As they grow, these biases in the high-latitude stratosphere likely feed back with radiation and the atmospheric circulation before ultimately equilibrating.

4 Challenges and choices

In this section, we highlight key decisions that led to a skillful, stable, and low-bias emulation, as well as some remaining challenges. From the outset, our goal was to use simpler ML models with the potential for general applicability in emulating atmospheric physics parameterizations. However, the path to the final emulator necessitated several problem-specific choices to successfully emulate the ZC microphysics scheme.

4.1 Key decisions

One of the most influential decisions was to target subcomponents of the microphysics scheme, specifically grid-scale condensation (gscond) and precipitation (precpd). Initial attempts to encapsulate the total ZC scheme tendency increments in a single model yielded high offline skill, but the online integration often resulted in difficult-to-interpret failures that crashed the simulation. This is a common failure mode when training models outside of the environment in which they are deployed (e.g., Brenowitz & Brether-

run type	gscond arch.	precpd arch.	ΔT	Δq	Δc	P
offline	dense-local	RNN	0.99	0.995	0.99	0.998
	dense-local	dense-column	0.99	0.99	0.97	0.99
	dense-column	dense-column	0.97	0.98	0.95	0.99
online	dense-local	RNN	0.98	0.98	0.95	0.98
	dense-local	dense-column	0.74	0.76	0.01	0.01
	dense-column	dense-column	-0.39	-0.46	-0.07	0.17

Table 5. Sensitivity of emulation skill to the use of general vs. prior-informed model architectures. “Dense-column” refers to a fully connected MLP with 2 hidden layers of 256 width and a linear readout layer. “Dense-local” and “RNN” refer to the architectures described in the methods section.

ton, 2019). Separating the subcomponents simplifies the enforcement of physical priors through model architecture design or output postprocessing.

Following component separation, we observed substantial improvements in online emulation skill by incorporating physically informed architectures. For the gscond emulator, we enforce grid point locality (i.e., dependence only on the grid point-local thermodynamic state) by using a dense-local MLP that does not mix any vertical information. For the precpd emulator, we enforce the downward dependence (i.e., rain falls downward) using an RNN that recurses downward over a vertical column. Table 5 displays the offline and online skill for a single 30-day run initialized in July, comparing the performance of the informed architectures to a reasonable uninformed default for atmospheric model process parameterization— a dense MLP combining features over the entire grid column to predict the full column increments. While these dense-column models exhibit high skill offline (always >95%), they fail online when continuously integrating on the atmospheric state. Replacing the RNN used for precpd emulation with a dense-column architecture that does not enforce downward dependence reduces cloud and precipitation skill to nearly 0%, even when using the physically informed gscond architecture. Using dense-column models for both subroutines results in negative skill (i.e., worse than zero-increment predictions) for all variables except surface precipitation.

The discrete-continuous nature of outputs from some atmospheric physics parameterizations (e.g., for microphysics) poses a unique challenge for emulation. Neural network regressors have difficulty producing exact zeros, since they are trained to a certain degree of precision and will produce noise below that threshold. This can complicate online integration, particularly for a microphysical scheme, where the local thermodynamic state may be quite sensitive to small changes in condensate or humidity, especially in very cold regions (e.g., Antarctica or the upper troposphere). For this reason, we introduced the activity classifier described in Sect. 2.2.1 into the gscond emulator. Figure 8 illustrates the need for such a classifier by comparing cloud distributions from simulations with and without a classifier to a baseline run. By day 15 after initialization, the condensate histogram shows that the emulation scheme without an activity classifier accumulates small values of cloud water ($\leq$2 mg/kg) at many grid points. Including a classifier within the gscond emulator to constrain the microphysical activity resolves this issue. Based on the good performance of the 30-day online simulations and non-locality of the precipitation scheme, we decided not to pursue an activity mask for the precpd emulator. However, the erroneous T and q precpd increments in the polar stratosphere contributing to biases in the 1-year run suggest a classifier might be helpful overall.

The final choice important to the success of the ZC emulator involved optimizing the model to predict condensate increments that span many orders of magnitude. As de-

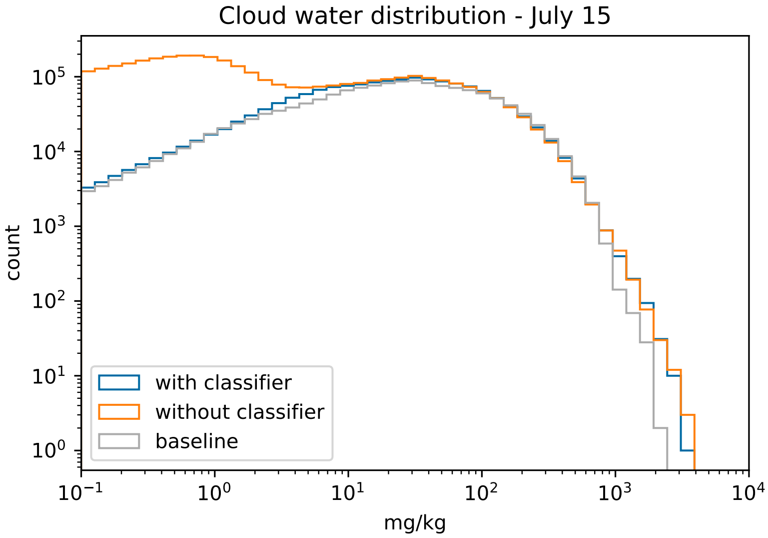

Figure 8. Cloud water mixing ratio distributions compared between three configurations: online emulation with a gscond activity classifier (blue), online emulation without an activity classifier (orange), and a baseline simulation (grey). Samples are taken from 8 3-hourly snapshots across day 15 of a 30-day simulation initialized on July 1.

scribed in Sect. 2.2.1, we used a temperature-dependent scaling in the gscond loss function, ensuring proportionate errors across a large range of local microphysical states. Model-level scaling is insufficient to handle this because a given model level may span a broad range of temperatures (e.g., the tropical boundary layer vs. the Antarctic plateau).

In addition to the conditional scaling, we added select rescaled input values (RH, log-scaled q and c) into the emulator inputs. Removing log-scaled inputs negatively impacts offline skill in polar and upper-level model regions (not shown). Including RH as an input increased skill and reduced condensate biases, particularly in the Antarctic region. For example, by day 5 of a July 1 initialized simulation, the emulator using RH as an input has an Antarctic average column-integrated condensate of 87 g/m^2 compared to a baseline value of 79 g/m^2. When not including RH, the average Antarctic column-integrated condensate value is 154 g/m^2 by July 5, roughly double the baseline value. Despite the overlap of the additional inputs, we believe they help reduce errors in cold-cloud regions by allowing the emulator discern vertical position, which is removed by per-level demeaning in the input normalization (Eq. 1). We conducted an experiment to reintroduce the vertical information by adjusting the input normalization for air pressure to remove the column mean instead of the per-level mean from each level. This configuration also increased offline skill and largely removed the Antarctic condensate bias without the need for RH, but was generally more sensitive to skill dropouts when used online.

4.2 Remaining challenges

In developing our emulation scheme, online simulations commonly presented unexpected challenges that needed to be addressed. Certain months, primarily October and November, tended to have lower online skill (∼85–90% compared to ∼93–96%) for clouds and precipitation compared to other months. The lower aggregate skill in these months was mainly due to significant precpd autoconversion misses (“skill dropouts”) during convective events for a few low-latitude columns (see Appendix B for an example). These skill dropouts start in the mid-troposphere near the freezing level and quickly affect the entire upper troposphere. The emulator recovers in the affected grid columns within a few hours or, at worst, a few days.

To minimize such dropouts, we employed a strategy of training an ensemble of emulators initialized with varying random seeds (e.g., as in Clark et al., 2022) and then select combinations of gscond and precpd emulators with the best online skill during the most problematic months of October and November. While this approach does not guarantee prevention of severe skill dropouts during other months or in a year-long simulation, it consistently produces stable, low-bias emulators with high skill.

We still do not have a foolproof approach for designing emulators without occasional skill dropouts. For instance, the emulator configuration that gave the most skillful 1-year online simulation (masking the top 5 levels of increments from both gscond and precpd) produces a substantial skill dropout in a 30-day simulation initialized at the start of December, leading to a December Δc skill = 54%, while the original gscond-only top 5 mask configuration has no issues (December Δc skill = 94%).

Altogether, this suggests the need for further refinement of the architectural design and training choices, such as whether recursion from the top model level is necessary, whether additional measures should be adopted to reduce sensitivity to the upper levels, or whether more training data are needed to handle the few convective events on the edges of the data distribution.

To handle the large dynamic range of condensate increments, we use temperature scaling in the gscond loss function. While this is generally very beneficial, especially in tandem with the gscond classifier, it does not prevent the emulator from occasionally creating spurious cloud in the uppermost model levels. These levels lie in the stratosphere, where temperature increases with height. Warmer temperatures lead to larger-amplitude condensate "noise", which the emulator later struggles to remove. Because there should never be any cloud in the top-most levels, we pragmatically resolved this by masking gscond increments in the top 5 model levels. However, as seen in the 1-year simulation polar stratospheric biases, a few issues remain related to emulator deficiencies in the upper levels.

While the current manuscript focuses on the development and evaluation of a robust, accurate ZC emulator, we recognize that speed of execution is a paramount consideration for emulator adoption, especially in operational settings. The current code infrastructure was designed for flexibility and ease of testing new ideas, rather than for optimal speed. In its current unoptimized state, the model with online emulation runs approximately 30% slower ($\sim$5.8 s/time step) than to the original C48 simulation ($\sim$4.8 s/time step) even when using available GPUs (4x Nvidia T4). Variable transfer between Python and Fortran adds around 7% to the run time. The remaining slowdown is likely related to choices in model architecture, such as shallow depth and sequential RNN steps, which lead to low GPU utilization (<10%). We believe that it will be possible to design ML emulators of more complex microphysical schemes that are more speed-competitive with the Fortran code which they aim to replace.

5 Conclusions

We have successfully developed an emulator to replace a simple Fortran microphysics scheme (Zhao-Carr) in FV3GFS, which controls grid-scale condensation (gscond) and precipitation (precpd) processes. Our findings demonstrate that when used online as a replacement for the Fortran scheme, the emulator maintains high skill ($\geq$94%) with low global-average bias (on the order of 1% or less) and remains stable for at least one year of continuous simulation. To our knowledge, this is the first successful emulation of a bulk microphysics scheme, and the first successful online emulation of a fast-timescale atmospheric parameterization central to global atmospheric forecasting.

A key contributor to the success of our emulator was tailoring its architecture to the underlying physical processes. By creating separate emulators for gscond and precpd, we enforce grid point locality and conservation for the condensation scheme, and we use an RNN to impose downward dependence in the atmosphere associated with falling precipition. This greatly improves the emulator's skill, especially when used online. Adding an activity classifier to the condensation emulator alleviated issues of excess condensate related to the discrete-continuous nature of the tendencies and field outputs. Using a temperature-scaled conditional loss function for the gscond emulator and providing re-scaled inputs to all emulators helped maintain skill across the high dynamic range of condensate and humidity tendencies that must be accurately predicted to simulate cloud processes throughout the global atmosphere.

As with any ML-based emulation problem, achieving perfection is difficult, and the current scheme is no exception. In 1-year online integrations, biases develop in the polar stratospheric temperature and humidity fields. These regions challenge the ML training because they have distinctly different local environments than the rest of the atmosphere and comprise a small fraction of the emulator's training data. Further improvements could clearly be made, but are beyond the scope of this paper, which was to demonstrate the feasibility of a skillful ML microphysics emulator for online use. For instance, a natural possibility that we did not have time to implement would be to explicitly predict precipitation flux at every model interface, which carries all the nonlocality in the microphysics. The hidden state of the precpd RNN is a skillful but imprecise proxy for this design, causing potential biases and drifts because physical constraints are imperfectly respected (e.g., that the evaporation of precipitation in any model level cannot exceed the downward flux of precipitation into that model level).

A compounding difficulty in the present work and generally for physics emulation is the inability to train emulation schemes directly in the context of their deployment within an atmospheric model. Fortran tooling for ML applications is challenging compared to the Python, but is still required for current atmospheric models. We utilize a Python package (`call_py_fort`) that provides an exceptional solution for interactive prototyping, but is not optimized for computational efficiency. Modeling frameworks on the horizon may simplify this process of ML integration and speed the development path to emulators that perform well online (Schneider et al., 2017; Dahm et al., 2023).

Our results stress the importance of evaluating the online performance for any proposed emulator, as it is straightforward to produce skillful offline models that may not perform well when integrated back into the model. It is also important to recognize that the development of emulators that perform well online is a challenging and time-consuming endeavor. If efficiency is the only goal, it may sometimes be more practical to invest in porting existing codes to run on GPUs, for example, as emulation requires significant human effort and problem-specific tuning.

Despite the challenges, our method and results are a proof-of-concept that machine learning techniques can effectively emulate fast physical processes central to the dynamics in weather and climate models. While our focus has been on a specific microphysics parameterization, we hope that the illustration of our problem-specific decisions will inform the application to similar or more complex physical schemes. With further research and development, emulation techniques can continue to contribute to improved skill and efficiency of weather and climate models.

Appendix A Zhao-Carr Microphysics

This scheme handles both phase changes—condensation and evaporation—and precipitation processes. Tendencies due to the former are typically 10x larger in magnitude.

The prognostic variables used by the scheme are the temperature T, specific humidity q, and a combined cloud water/ice mixing ratio c.

The gscond scheme handles evaporation of cloud and condensation. Evaporation of cloud is given by $E_c = \frac{1}{\Delta t} \max[\min[q_s(f_0 - f), c], 0]$. f is relative humidity. f_0 is a critical relative humidity threshold which Zhao and Carr (1997) describe as "empirically set to 0.75 over land and 0.90 over ocean." q_s is the saturation specific humidity.

Condensation C_g on the other hand is given by a more complex formula involving a relative humidity tendency. See Eq. (8) of Zhao and Carr (1997). Both formulas depend only on the thermodynamic state of a single (x, y, z) location, but there is some non-local dependence on the assumed phase of the cloud and the corresponding latent heating rate.

The precpd scheme handles the conversion of cloud into rain/snow and the evaporation of the latter as it falls through the atmosphere. Broadly speaking, it can be written as the following:

$$\begin{aligned}
E_{rr} &= E_r(T, f, P_r) \\
E_{rs} &= E_r(T, f, P_s) \\
P &= P(T, f, c, P_r, P_s) \\
P_{sm} &= P_{sm}(T, f, c, P_r, P_s) \\
P_r &= \int_{p_t}^{p} (P - E_{rr}) dp/g \\
P_s &= \int_{p_t}^{p} (P_{sm} - E_{rs}) dp/g.
\end{aligned}$$

Most of the formulas are proportional to rainfall P_r and snowfall P_r rates at a given level, though are some rate constants that depend exponentially on temperature. p_t is the pressure at the top of the atmosphere.

Appendix B Precpd emulator skill dropouts

Over the course of refining the emulation methodology, we observed larger variability in the online skill scores of cloud and precipitation predictions, despite minimal-or-no changes in emulator training or runtime configuration. In this section, we discuss the primary source of that variance, which we refer to as skill dropouts. As an example, Figure B1 displays the surface precipitation skill over time for two 1-year simulations. When the top 5 layer increment mask is adjusted from application to only gscond to both gscond and precpd, the severity of skill dropouts decreases markedly.

Upon closer examination of the skill dropouts, the precpd emulator appears to be the source of the issue. We focus on the dropout about 6 months into the gscond-only masking experiment to illustrate this point. In this case, a cluster of columns near the Maritime Continent is responsible for most of skill reduction. By removing the five grid columns with the largest tendency errors, the overall snapshot skill goes from approximately 0% to over 70%. When examining the tendency profiles from the column with the largest errors (Fig. B2), the gscond emulator largely matches the diagnostic Fortran, while the precpd emulator completely misses the autoconversion of condensate to precipitation in middle-and-upper levels. Leading up to this time step, we have confirmed that gscond remains skillfull, while precpd skill degrades (not shown). The gscond emulator retaining skill throughout this event suggests that a process outside of the ZC scheme, such as deep convection, adds condensate throughout the column. The precpd emulator then fails to precipitate the added condensate.

Overall, we hypothesize that the skill dropouts are associated with training data insufficiency related to intense convection and/or unconstrained sensitivities of the RNN

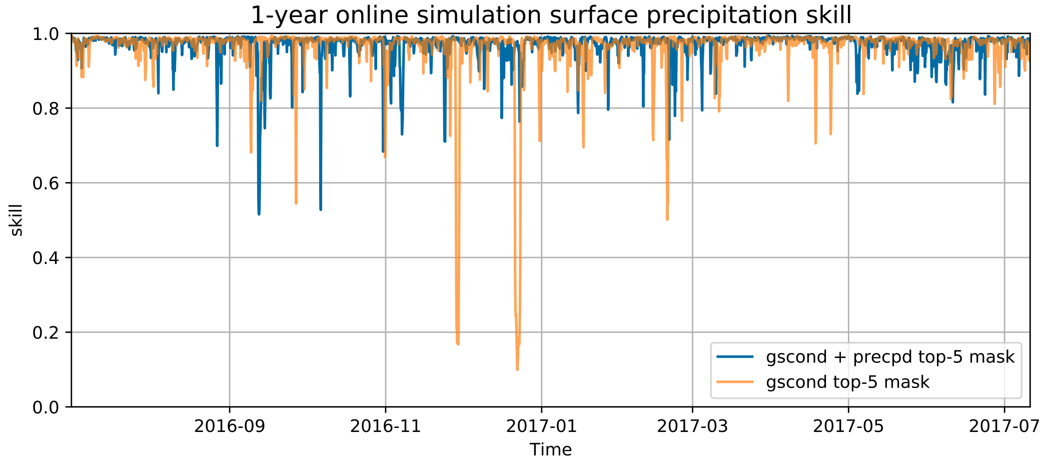

Figure B1. Surface precipitation skill over the 1-year online simulation for two increment masking configurations: (orange) gscond-only top 5 layer masking and (blue) gscond and precpd top 5 layer masking.

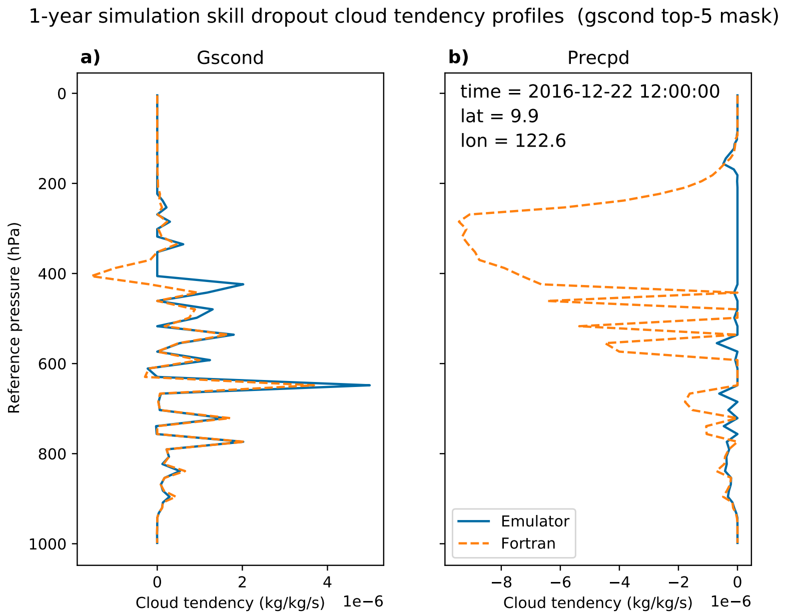

Figure B2. Vertical tendency profiles from the (a) gscond and (b) precpd schemes during the December 22nd 12 UTC skill dropout event in the gscond top 5 layer increment mask 1-year simulation. Each subcomponent panel shows the condensate tendencies predicted from the emulator (blue) and the diagnostic Fortran (orange dashed) for the selected column with the largest errors.

to upper-level inputs. It is encouraging that despite the magnitude of the misses, the ZC emulators resolve the issue in a few days or less for all cases observed (e.g., see Fig. S6). We also note that the dropouts tend to be confined to only a few grid columns, typically occurring in the tropics or subtropics. The isolated spatial extent of the skill dropout sources highlights the challenge in achieving consistently high skill in our chosen metrics throughout the simulations. It also demonstrates how quickly the skill can deteriorate if even a few predictions degrade.

Glossary

dense-local An MLP that takes in a single vertical level of inputs from a column and produces outputs for that same level. The vertical independence makes it "local".

gscond The gridscale condensation component of Zhao-Carr microphysics

precpd The precipitation component of Zhao-Carr microphysics

skill dropout A temporary reduction in the online skill metric calculated between the emulator tendencies and the diagnostic Fortran tendencies

Acronyms

ML machine learning

MLP multi-layer perceptron (feed-forward neural net)

RNN recurrent neural net

ZC Zhao-Carr

Appendix C Open Research

The code and configurations used to produce training data, train ML models, and run FV3GFS simulations are available on Github (`https://github.com/ai2cm/zc-emulation-manuscript`) and archived on Zenodo (`https://doi.org/10.5281/zenodo.7976184`). The data and docker images to reproduce results with the code are available on Zenodo (`https://doi.org/10.5281/zenodo.79`

Acknowledgments

We thank the Allen Institute for Artificial Intelligence for their support of this work and for hosting Jacqueline Nugent as a summer intern during a portion of this project. We would also like to acknowledge NOAA-GFDL, NOAA-EMC, and the UFS community for providing publicly available code and data to initialize and run the FV3GFS atmospheric model.

References

Brenowitz, N. D., & Bretherton, C. S. (2019). Spatially Extended Tests of a Neural Network Parametrization Trained by Coarse-Graining. *Journal of Advances in Modeling Earth Systems*, *11*(8), 2728–2744. Retrieved 2023-05-25, from `https://onlinelibrary.wiley.com/doi/abs/10.1029/2019MS001711` (_eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1029/2019MS001711) doi: 10.1029/2019MS001711

Bretherton, C. S., Henn, B., Kwa, A., Brenowitz, N. D., Watt-Meyer, O., McGibbon, J., . . . Harris, L. (2022). Correcting Coarse-Grid Weather and Climate Models by Machine Learning From Global Storm-Resolving Simulations. *Journal of Advances in Modeling Earth Systems*, *14*(2), e2021MS002794. Retrieved 2023-05-25, from `https://onlinelibrary.wiley.com/doi/abs/10.1029/2021MS002794` (_eprint:

https://onlinelibrary.wiley.com/doi/pdf/10.1029/2021MS002794) doi: 10.1029/2021MS002794

Chantry, M., Hatfield, S., Dueben, P., Polichtchouk, I., & Palmer, T. (2021). Machine Learning Emulation of Gravity Wave Drag in Numerical Weather Forecasting. *Journal of Advances in Modeling Earth Systems*, *13*(7), e2021MS002477. Retrieved 2023-05-25, from https://onlinelibrary.wiley.com/doi/abs/10.1029/2021MS002477 (_eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1029/2021MS002477) doi: 10.1029/2021MS002477

Chevallier, F., Chéruy, F., Scott, N. A., & Chédin, A. (1998, November). A Neural Network Approach for a Fast and Accurate Computation of a Longwave Radiative Budget. *Journal of Applied Meteorology and Climatology*, *37*(11), 1385–1397. Retrieved 2023-05-25, from https://journals.ametsoc.org/view/journals/apme/37/11/1520-0450_1998_037_1385_annafa_2.0.co_2.xml (Publisher: American Meteorological Society Section: Journal of Applied Meteorology and Climatology) doi: 10.1175/1520-0450(1998)037⟨1385:ANNAFA⟩2.0.CO;2

Clark, S. K., Brenowitz, N. D., Henn, B., Kwa, A., McGibbon, J., Perkins, W. A., … Harris, L. M. (2022). Correcting a 200 km Resolution Climate Model in Multiple Climates by Machine Learning From 25 km Resolution Simulations. *Journal of Advances in Modeling Earth Systems*, *14*(9), e2022MS003219. Retrieved 2023-05-25, from https://onlinelibrary.wiley.com/doi/abs/10.1029/2022MS003219 (_eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1029/2022MS003219) doi: 10.1029/2022MS003219

Dahm, J., Davis, E., Deconinck, F., Elbert, O., George, R., McGibbon, J., … Fuhrer, O. (2023, May). Pace v0.2: a Python-based performance-portable atmospheric model. *Geoscientific Model Development*, *16*(9), 2719–2736. Retrieved 2023-05-25, from https://gmd.copernicus.org/articles/16/2719/2023/ (Publisher: Copernicus GmbH) doi: 10.5194/gmd-16-2719-2023

Gettelman, A., Gagne, D. J., Chen, C.-C., Christensen, M. W., Lebo, Z. J., Morrison, H., & Gantos, G. (2021). Machine Learning the Warm Rain Process. *Journal of Advances in Modeling Earth Systems*, *13*(2), e2020MS002268. Retrieved 2023-05-25, from https://onlinelibrary.wiley.com/doi/abs/10.1029/2020MS002268 (_eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1029/2020MS002268) doi: 10.1029/2020MS002268

Grabowski, W. W. (2019, September). Separating physical impacts from natural variability using piggybacking technique. *Advances in Geosciences*, *49*, 105–111. Retrieved 2023-05-25, from https://adgeo.copernicus.org/articles/49/105/2019/ doi: 10.5194/adgeo-49-105-2019

Harris, L. M., & Lin, S.-J. (2013, January). A Two-Way Nested Global-Regional Dynamical Core on the Cubed-Sphere Grid. *Monthly Weather Review*, *141*(1), 283–306. Retrieved 2023-05-25, from https://journals.ametsoc.org/view/journals/mwre/141/1/mwr-d-11-00201.1.xml (Publisher: American Meteorological Society Section: Monthly Weather Review) doi: 10.1175/MWR-D-11-00201.1

Keller, C. A., & Evans, M. J. (2019, March). Application of random forest regression to the calculation of gas-phase chemistry within the GEOS-Chem chemistry model v10. *Geoscientific Model Development*, *12*(3), 1209–1225. Retrieved 2023-05-25, from https://gmd.copernicus.org/articles/12/1209/2019/ (Publisher: Copernicus GmbH) doi: 10.5194/gmd-12-1209-2019

Kelp, M. M., Jacob, D. J., Lin, H., & Sulprizio, M. P. (2022). An Online-Learned Neural Network Chemical Solver for Stable Long-Term Global Simulations of Atmospheric Chemistry. *Journal of Advances in Model-*

ing Earth Systems, *14*(6), e2021MS002926. Retrieved 2023-05-25, from https://onlinelibrary.wiley.com/doi/abs/10.1029/2021MS002926 (_eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1029/2021MS002926) doi: 10.1029/2021MS002926

Krasnopolsky, V. M., Fox-Rabinovitz, M. S., & Chalikov, D. V. (2005, May). New Approach to Calculation of Atmospheric Model Physics: Accurate and Fast Neural Network Emulation of Longwave Radiation in a Climate Model. *Monthly Weather Review*, *133*(5), 1370–1383. Retrieved 2023-05-25, from https://journals.ametsoc.org/view/journals/mwre/133/5/mwr2923.1.xml (Publisher: American Meteorological Society Section: Monthly Weather Review) doi: 10.1175/MWR2923.1

Krasnopolsky, V. M., Fox-Rabinovitz, M. S., Hou, Y. T., Lord, S. J., & Belochitski, A. A. (2010, May). Accurate and Fast Neural Network Emulations of Model Radiation for the NCEP Coupled Climate Forecast System: Climate Simulations and Seasonal Predictions. *Monthly Weather Review*, *138*(5), 1822–1842. Retrieved 2023-05-25, from https://journals.ametsoc.org/view/journals/mwre/138/5/2009mwr3149.1.xml (Publisher: American Meteorological Society Section: Monthly Weather Review) doi: 10.1175/2009MWR3149.1

O'Gorman, P. A., & Dwyer, J. G. (2018). Using Machine Learning to Parameterize Moist Convection: Potential for Modeling of Climate, Climate Change, and Extreme Events. *Journal of Advances in Modeling Earth Systems*, *10*(10), 2548–2563. Retrieved 2023-05-25, from https://onlinelibrary.wiley.com/doi/abs/10.1029/2018MS001351 (_eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1029/2018MS001351) doi: 10.1029/2018MS001351

Rasp, S., Pritchard, M. S., & Gentine, P. (2018, September). Deep learning to represent subgrid processes in climate models. *Proceedings of the National Academy of Sciences*, *115*(39), 9684–9689. Retrieved 2023-05-25, from https://www.pnas.org/doi/full/10.1073/pnas.1810286115 (Publisher: Proceedings of the National Academy of Sciences) doi: 10.1073/pnas.1810286115

Schneider, T., Lan, S., Stuart, A., & Teixeira, J. (2017). Earth System Modeling 2.0: A Blueprint for Models That Learn From Observations and Targeted High-Resolution Simulations. *Geophysical Research Letters*, *44*(24), 12,396–12,417. Retrieved 2023-05-25, from https://onlinelibrary.wiley.com/doi/abs/10.1002/2017GL076101 (_eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/2017GL076101) doi: 10.1002/2017GL076101

Schreck, J. S., Becker, C., Gagne, D. J., Lawrence, K., Wang, S., Mouchel-Vallon, C., … Hodzic, A. (2022). Neural Network Emulation of the Formation of Organic Aerosols Based on the Explicit GECKO-A Chemistry Model. *Journal of Advances in Modeling Earth Systems*, *14*(10), e2021MS002974. Retrieved 2023-05-25, from https://onlinelibrary.wiley.com/doi/abs/10.1029/2021MS002974 (_eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1029/2021MS002974) doi: 10.1029/2021MS002974

Ukkonen, P., Pincus, R., Hogan, R. J., Pagh Nielsen, K., & Kaas, E. (2020). Accelerating Radiation Computations for Dynamical Models With Targeted Machine Learning and Code Optimization. *Journal of Advances in Modeling Earth Systems*, *12*(12), e2020MS002226. Retrieved 2023-05-25, from https://onlinelibrary.wiley.com/doi/abs/10.1029/2020MS002226 (_eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1029/2020MS002226) doi: 10.1029/2020MS002226

Veerman, M. A., Pincus, R., Stoffer, R., van Leeuwen, C. M., Podareanu, D., & van Heerwaarden, C. C. (2021, February). Predicting atmospheric optical properties for radiative transfer computations using neural networks. *Philo-*

sophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, *379*(2194), 20200095. Retrieved 2023-05-25, from https://royalsocietypublishing.org/doi/10.1098/rsta.2020.0095 (Publisher: Royal Society) doi: 10.1098/rsta.2020.0095

Yuval, J., & O'Gorman, P. A. (2020, July). Stable machine-learning parameterization of subgrid processes for climate modeling at a range of resolutions. *Nature Communications*, *11*(1), 3295. Retrieved 2023-05-25, from https://www.nature.com/articles/s41467-020-17142-3 (Number: 1 Publisher: Nature Publishing Group) doi: 10.1038/s41467-020-17142-3

Zhao, Q., & Carr, F. H. (1997, August). A Prognostic Cloud Scheme for Operational NWP Models. *Monthly Weather Review*, *125*(8), 1931–1953. Retrieved 2023-01-30, from https://journals.ametsoc.org/view/journals/mwre/125/8/1520-0493_1997_125_1931_apcsfo_2.0.co_2.xml (Publisher: American Meteorological Society Section: Monthly Weather Review) doi: 10.1175/1520-0493(1997)125⟨1931:APCSFO⟩2.0.CO;2