JAMES | Journal of Advances in Modeling Earth Systems®

RESEARCH ARTICLE
10.1029/2024MS004833

Vertically Recurrent Neural Networks for Sub-Grid Parameterization

P. Ukkonen[1] and **M. Chantry**[2]

[1]Atmospheric, Oceanic and Planetary Physics, University of Oxford, Oxford, UK, [2]Research Department, European Centre for Medium-Range Weather Forecasts, Reading, UK

Key Points:

- We test a sequence learning approach for parameterization of sub-grid processes in a column
- Recurrent neural networks (RNNs) emulate radiation skillfully in the Integrated Forecasting System, but optimized radiation code is generally faster
- RNNs emulate moist physics with similar offline accuracy to ResNets, but are faster and perform better in semi-prognostic tests

Supporting Information:

Supporting Information may be found in the online version of this article.

Correspondence to:

P. Ukkonen,
peter.ukkonen@physics.ox.ac.uk

Citation:

Ukkonen, P., & Chantry, M. (2025). Vertically recurrent neural networks for sub-grid parameterization. *Journal of Advances in Modeling Earth Systems*, *17*, e2024MS004833. https://doi.org/10.1029/2024MS004833

Received 7 NOV 2024
Accepted 18 MAY 2025

Abstract Machine learning has the potential to improve the physical realism and/or computational efficiency of parameterizations. A typical approach has been to feed concatenated vertical profiles to a dense neural network. However, feed-forward networks lack the connections to propagate information sequentially through the vertical column. Here we examine if predictions can be improved by instead traversing the column with recurrent neural networks (RNNs) such as Long Short-Term Memory (LSTMs). This method encodes physical priors (locality) and uses parameters more efficiently. Firstly, we test RNN-based radiation emulators in the Integrated Forecasting System. We achieve near-perfect offline accuracy, and the forecast skill of a suite of global weather simulations using the emulator are for the most part statistically indistinguishable from reference runs. But can radiation emulators provide both high accuracy and a speed-up? We find optimized, state-of-the-art radiation code on CPU generally faster than RNN-based emulators on GPU, although the latter can be more energy efficient. To test the method more broadly, and explore recent challenges in parameterization, we also adapt it to data sets from other studies. RNNs outperform reference feed-forward networks in emulating gravity waves, and when combined with horizontal convolutions, for non-local unified parameterization. In emulation of moist physics with memory, the RNNs have similar offline accuracy as ResNets, the previous state-of-the-art. However, the RNNs are more efficient, and more stable in autoregressive semi-prognostic tests. Multi-step autoregressive training improves performance in these tests and enables a latent representation of convective memory. Recently proposed linearly recurrent models achieve similar performance to LSTMs.

Plain Language Summary Weather and climate models represent small-scale atmospheric processes such as clouds using approximations that are a large source of uncertainty in climate projections. Machine learning has emerged as a promising solution to improve realism, but traditional approaches based on fully-connected networks are not necessarily ideal for modeling vertically non-local processes such as clouds and convection. We show that treating the atmospheric column as a sequence which is traversed using recurrent neural networks leads to improved prediction of sub-grid processes while also using fewer trainable parameters.

1. Introduction

Climate and weather models simulate the dynamics of the atmosphere by solving the differential equations for fluid flow numerically on a 3D grid. Fluid-dynamical processes occurring at sub-grid scales, such as turbulence and convection, and non-fluid-dynamical processes, such as radiation, are not resolved and are instead represented via physical approximations known as parameterizations. Often relying on heuristics and empiricism, parameterizations are a major source of systematic errors in these models (Rasp et al., 2018). Clouds and their interaction with other processes are poorly represented in particular (Zadra et al., 2018), which is unsurprising considering the complexities and multitude of scales involved, from cloud microphysical processes to kilometer-scale convective cells. As a result, the response of clouds to warming remains the greatest source of uncertainty in climate projections (Schneider et al., 2017). Progress on reducing this uncertainty has been limited: the latest generation of climate models show a larger spread in equilibrium climate sensitivity (the global surface temperature response to a doubling of CO_2) than the previous one (Zelinka et al., 2020). Convection-permitting simulations at a kilometer-scale horizontal resolution alleviate some deficiencies but not all, and remain computationally unfeasible for global multidecadal simulations (Gentine et al., 2018). Parameterizations are still needed, so how can they be improved?

Machine learning (ML) has emerged as a potential solution. One approach is to emulate an existing parameterization in order to reduce computational cost. The perceived expense of radiation computations has motivated radiation emulation studies for more than two decades (e.g., Chevallier et al., 1998; V. Krasnopolsky et al., 2010; Belochitski & Krasnopolsky, 2021; Roh & Song, 2020; Song & Roh, 2021; Ukkonen, 2022). High speed-ups have been reported in many studies, but these have typically pitted simple ML architectures, which may not be accurate enough for global models, against out-of-date radiation codes. To our knowledge, no operational Numerical Weather Prediction (NWP) or climate model today uses radiation emulators, nor has it been clearly demonstrated that ML approaches can provide better speed-accuracy trade-offs than physical schemes.

One may also try to learn new parameterizations. While fully ML-based weather models trained on reanalysis data have recently been shown to match or even exceeded NWP models in deterministic skill, Google's NeuralGCM model achieves better stability (up to multiple years) by using a hybrid approach where a dynamical solver is coupled to a learned parameterization module (Kochkov et al., 2024). Attempts to learn new (convective or unified) parameterizations from convection-permitting simulations have been numerous (Brenowitz & Bretherton, 2018; Brenowitz & Bretherton, 2019; Brenowitz et al., 2020; Bretherton et al., 2022; Gentine et al., 2018; Han et al., 2020; Mooers et al., 2021; Rasp et al., 2018; Yuval et al., 2021; Wang Han et al., 2022; Wang and Yuval et al., 2022; O'Gorman & Dwyer, 2018; Yuval & O'Gorman, 2020). Major features of such simulations have been reproduced with ML-parameterized coarse-grid simulations, but despite using simplified aquaplanet setups, challenges with stability have been widely reported (Brenowitz & Bretherton, 2018). Full emulation of moist convection under realistic orography has so far been tackled in only a few published studies (Han et al., 2020; Mooers et al., 2021; Wang Han et al., 2022; Han et al., 2023; Watt-Meyer et al., 2024), of which the last three performed prognostic testing in a GCM. Stable online integration was ultimately achieved (only for a month in Watt-Meyer et al., 2024) but with substantial biases in for example, temperature. The issue of stability is linked to generalization, a key challenge for ML parameterizations. Learning from data is inherently problematic for climate prediction because it requires the ability to generalize to conditions that have potentially not been seen (Gentine et al., 2018), such as higher temperatures and greenhouse gas concentrations.

The commonly used approach in ML parameterization is to take the vertical profiles of atmospheric variables such as temperature and humidity and concatenate them into input and output vectors of a feed-forward neural network (NN); typically a fully-connected, that is, dense NN (DNN). But is this the best approach? By stacking variables in such a way, the vertical dimension is not explicitly represented and parameters are wasted by assigning separate weights to each level. The network also lacks the connections to allow sequential flow of information through the vertical (Figure 1), which would reflect the underlying physics, and instead directly connects the top-of-atmosphere to the surface, which is unphysical. Using vertical convolutions has been shown to improve the emulation of moist physics (Han et al., 2020) and does enforce locality, but also requires deep networks to capture a larger receptive field (higher non-locality).

Therefore, we propose to instead treat the atmospheric column as a sequence and traverse it vertically with recurrent neural networks (RNNs). This enforces physical priors (locality) while allowing full vertical connectivity using the minimum number of parameters. RNNs have shown great success in sequence learning tasks such as machine translation and were previously shown to be very skillful in emulating atmospheric radiative transfer (Ukkonen, 2022; Yao et al., 2023). While these studies were limited to offline evaluation, more recently Tahseen et al. (2024) used the RNN architecture from Ukkonen (2022) to emulate radiative transfer in GCM simulations of Venus, and achieved promising results, with time-averaged differences in temperature profiles staying below 4K in simulations over 117 Earth Days. Perkins et al. (2024) also used RNNs to predict precipitation within an ML emulation of cloud microphysics in a climate model, with good success. Like us, they were also motivated to employ physically meaningful priors in ML architecture (condensation is local, rain falls down), but used vanilla RNNs, and only for one component of the microphysics. RNNs are known to be more difficult to train due to vanishing gradients over longer sequences, which has motivated RNN variants with gating mechanisms such as Long Short-Term Memory (LSTM, Hochreiter & Schmidhuber, 1997) and Gated Recurrent Unit (GRU, Cho et al., 2014), now widely used. Yet, a priori it is unclear if they can outperform previously employed methods when applied to emulation of sub-grid processes more generally.

We study these and other questions by testing RNNs on several data sets. Firstly, we train RNN-based models to emulate ecRad, the radiation scheme developed at the European Centre of Medium-Range Weather Forecasts (ECMWF) and perform global simulations with a leading NWP model to determine their impact on medium-

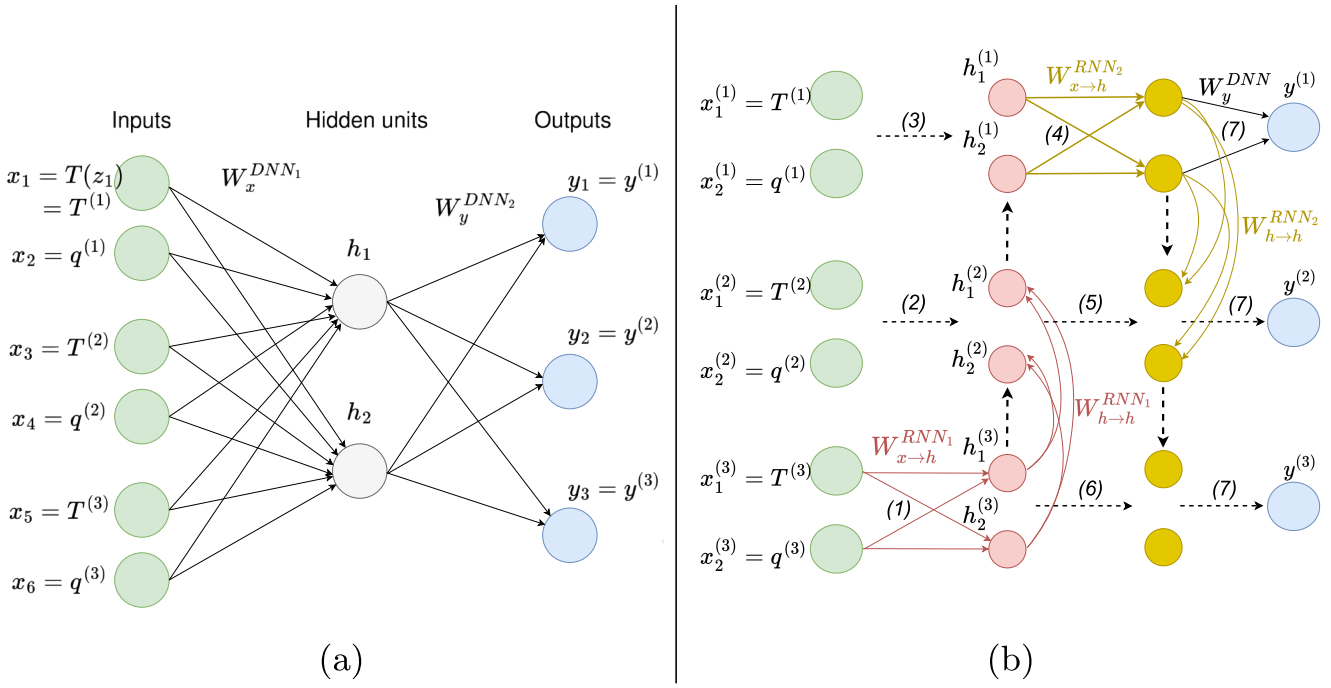

Figure 1. The structure of (a) a dense NN (DNN) and (b) a bidirectional recurrent NN predicting the vertical profile of a variable y from the profiles of two variables, T and q. This toy example has only three vertical levels (superscript) and one hidden DNN layer with two neurons ($N_h = 2$). Atmospheric models use an order of magnitude more vertical levels and more variables. In the DNN approach (a) the input and output columns thus become large and N_h needs to be increased correspondingly. This results in a high model complexity, measured by the number of trainable parameters W_x and W_y, whose size is given by $(N_x \times N_h)$ and $(N_y \times N_h)$ respectively. In addition, the DNN cannot propagate information sequentially through each vertical level. By instead traversing the column with two recurrent neural networks (RNNs) (b), whereby a hidden memory (here two neurons) is updated at each level, weights can be reused and information is propagated sequentially (each calculation step is the number in brackets) through the atmosphere as in for example, radiative transfer equations. The depicted architecture is used throughout the paper, typically consisting of just an upward RNN, a downward RNN and a local DNN for the final outputs. If surface-level variables exist, they are used to predict the initial state of RNN_1 using another DNN.

range forecast skill (Section 2). To assess if emulation is worthwhile we compare the runtime and energy use against a recently optimized version of ecRad. In the second part of the paper, we take data from other studies to test the method on a wide range of atmospheric processes: non-orographic gravity waves (Chantry et al., 2021) in Section 3, horizontally non-local unified physics (Wang and Yuval et al., 2022) in Section 4 and moist physics (Han et al., 2020, 2023) in Section 5. For this important final problem we explore autoregressive training and different methods of representing convective memory, and furthermore test various recently proposed space state models which are similar to RNNs but enable parallel computation. The stability and generalization aspect is explored via offline autoregressive tests (coupling to an ESM will be carried out in a future paper).

Our basic modeling approach, introduced in Ukkonen (2022), is illustrated in Figure 1. It differs from a traditional BiRNN where the second RNN is connected to the inputs. As shown in the moist physics section, connecting the second RNN to the first improves performance. The methodologies for different emulation problems are summarized in Table 1. Unless stated otherwise, when replicating existing studies (Sections 3–5), the inputs and outputs, their normalization and other aspects are identical to the original studies. The Adam optimizer with a learning rate of 0.001 was used for training RNNs, whose architecture was tuned by hand using separate validation data.

2. Radiative Transfer

2.1. Data and Methods

We trained emulators of the ECMWF radiation scheme "ecRad" (Hogan & Bozzo, 2018) by using data from the ML for Scalable meTeoROlogy and climate (MAELSTROM) project which ran from 2021 to 2024. One of the applications in MAELSTROM was the emulation of model components to speed-up weather forecasts. IFS

Table 1
Summary of the Different Parameterization Problems and Their Methodologies

Emulated process	Full-complexity radiation scheme with aerosols, gases and high vertical resolution	Non-orographic gravity waves	Unified parameterization (coarse-graining setup)	Super-parameterization (2D CRM)
Original study	–	Chantry et al., 2021	Wang and Yuval et al., 2022, Wang Han et al., 2022	Han et al., 2020, 2023
Unique aspects	–	–	Horizontal non-locality	Memory
Reference model	–	DNNs with 4–6 layers	DNNs with 3 layers	ResNet with 32 layers
Our model (neurons)	LW: 2xLSTM; SW: 3x LSTM (64 in each)	2xGRU (16–512)	Conv2D + 2xLSTM (160,192,160)	2xLSTM (160,96)/2xLSTM (128,128)
Batch size	512	512	512	1,024–2048
Other architectures tested	RNN, GRU; convolutions with/without attention	LSTM	ConvLSTM, Conv2D + 2xGRU, LSTM, GRU	RNN, GRU; SSMs and linearly recurrent NNs (Table 2); TCN, ConvLSTM
Prognostic testing	Suite of 10-day global NWP simulations	No	No	Semi-prognostic autoregressive tests
Training profiles	21×10^6	28.2×10^6	3.71×10^6	21×10^6
Training/ validation/ testing split	2020/2020/2021	2015/2016/2017	330 days (first 50%/middle 10%/last 40%)	3 years (first 37.5%/middle 10.25%/ last 50%)
Number of levels (sequence length)	137	63	48	30
Level inputs	23 variables: 7 gases (q, CO_2, O_3, CH_4, N_2O, CFC11-eq, CFC12), 12 aerosols, cf, $q_{ice}, q_{liq}, re_{ice}, re_{liq}$	T, p, u, v	T, q, u, v, w	$\frac{dT}{dt}^{l.s.}$, $\frac{dq}{dt}^{l.s.}$, T, q, $\frac{dT}{dt}$, $\frac{dq}{dt}$, q_{ice}, q_{liq}
Other inputs	13 scalars (α, ϵ, T_{skin}, μ_0, ssa); T, p on 138 half-levels; cop on 136 half-levels	p_{sfc}, z_{sfc}		SSHF, SLHF, p_{sfc}
Model outputs	$F_\uparrow, F_\downarrow$ on half-levels	$\frac{du}{dt}$, $\frac{dv}{dt}$ on levels	Fluxes and tendencies, see Figure 7	$\frac{dT}{dt}$, $\frac{dq}{dt}$, q_{ice}, q_{liq} on levels

Note. SW = shortwave; LW = longwave; SSM = Space-State Model; TCM = Temporal Convolutional Networks, T = temperature; p = pressure; q = mixing ratio of water vapor; CO_2 = (mixing ratio of) carbon dioxide; O_3 = ozone; CH_4 = methane; N_2O = nitrous oxide; cf = cloud fraction; q_{ice}; q_{liq} = mixing ratio of cloud ice and liquid water; re_{ice}; re_{liq} = effective radius of cloud ice and liquid water; α = shortwave surface albedos in 6 bands; *epsilon* = LW emissivity in 2 bands; μ_0 = cosine of solar zenith angle; *ssa* = single-scattering albedo; *cop* = cloud overlap parameter, z_{sfc} = surface geopotential, $F_\uparrow, F_\downarrow$ = up- and downwelling flux.

simulations were carried out using ecRad configured either with the TripleClouds or SPARTACUS solvers (coupled to the RRTMG gas optics scheme). TripleClouds represents cloud inhomogeneity efficiently via two cloudy regions (Shonk & Hogan, 2008) and is planned to replace McICA in a future IFS cycle, while SPARTACUS is its more expensive variant which additionally represents intra-column cloud 3D radiative effects (Schäfer et al., 2016). Radiation was computed on a 40 km grid (TL511), resulting in 271,360 samples of grid columns for each timestep. IFS model was run for the year 2020 every 30 days, for 30 days, saving the radiation inputs and outputs every 25 hr. While more than 100 million columns were saved, we trained TripleClouds emulators using roughly 21 million. Testing data comes from similar simulations performed for 2019. Because the task is the emulation of an operational scheme in a leading NWP model in all its complexity, the inputs are expansive, with 23 layer-wise inputs defined on 137 levels (Table 1).

For learning emulators of existing schemes, the communication pattern of the algorithm can motivate the NN architecture. Separate models were developed for the shortwave (SW) and longwave (LW) components of ecRad due to them having different flavors of underlying physics: emission only occurs in the LW, while scattering is more important in the SW (in our simulations and the operational IFS model, LW scattering is only computed for clouds). The LW and SW solvers in ecRad contain only two vertical loops with loop dependencies. The first starts from the surface and computes albedos and source terms (LW only), while the second loop starts from the top-of-atmosphere and computes fluxes from these variables. Our emulators closely mimic this structure (Figure S1 in Supporting Information S1). However, we found it useful to include a third (downward) RNN in the SW model

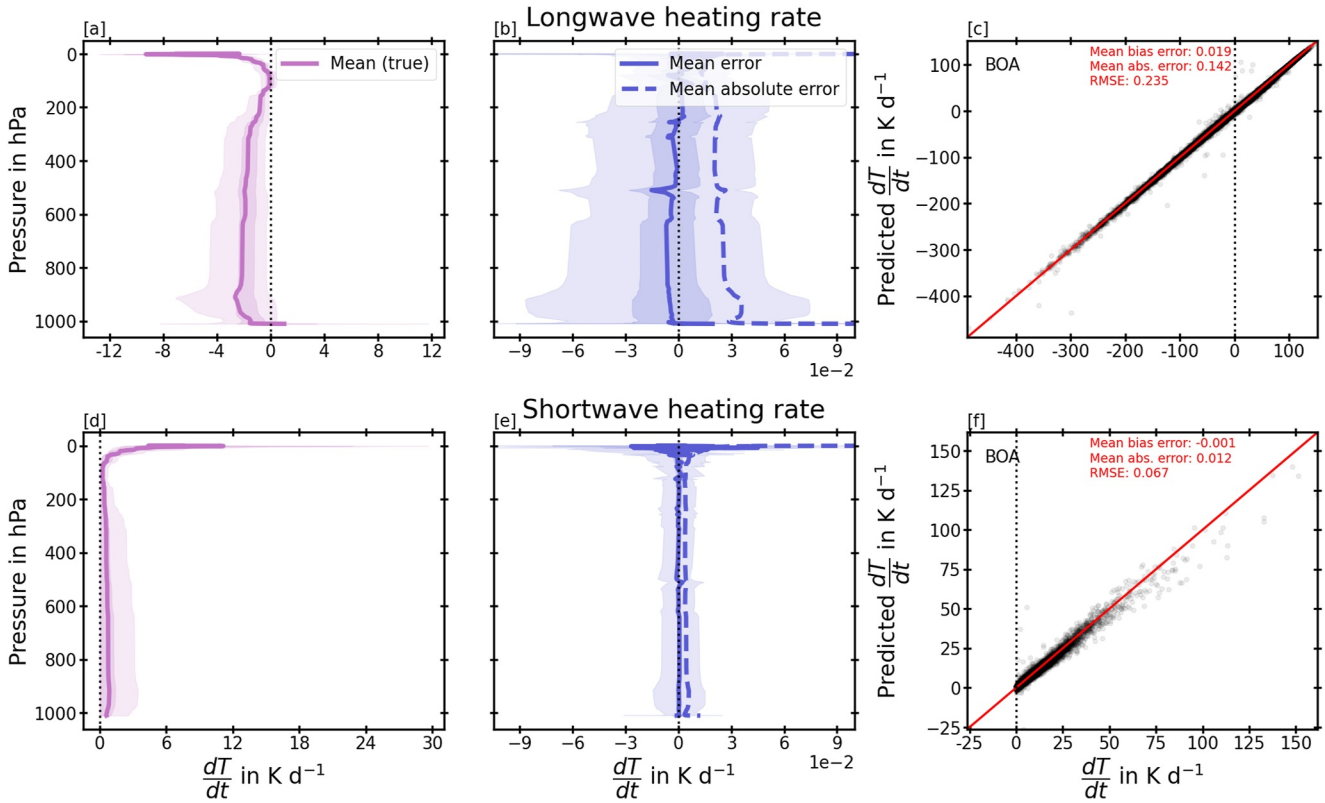

Figure 2. Vertical profile of mean heating rate (a, d), its mean error and mean absolute error (b, e) and a scatter plot of predicted versus true heating rate at bottom-of-atmosphere (BOA) in the longwave (LW) (a, b, c) and shortwave (SW) (d, e, f). In the first two columns the shaded colors correspond to 50% (darker) and 90% (lighter) quantiles. The *x*-axis in (b, e) has been cropped. LW mean absolute errors (MAE) reaches 0.8 K d^{-1} at TOA (0.02 hPa, not shown) and 0.14 K d^{-1} at BOA, SW MAE reached 0.2 K d^{-1} at TOA (not shown).

where it improved validation loss by 20%–25%, with the drawback of increasing its inference time by roughly a third. We use LSTM layers with 64 hidden units. As in Ukkonen (2022); Yao et al. (2023), heating rates are computed from fluxes using a physical equation and a hybrid loss is used to optimize both. The SW fluxes are scaled by incoming solar flux (removing the dependency on a boundary condition). A similar approach is used in the LW, but here the outputs from the final DNN are multiplied by a scaling factor which is predicted from skin temperature using another DNN.

2.2. Offline Accuracy

Firstly, we evaluate the TripleCloud emulators offline using 11.8 million profiles which correspond to three global snapshots of the 2019 IFS runs (00z 31.5, 29.8, and 28.10). The profiles of per-level mean error and mean absolute errors (MAE) are shown in Figure 2 for heating rates and in Figure S2 in Supporting Information S1 for fluxes. Mean absolute errors in LW fluxes is below 0.6 W m^{-2} across the atmosphere while MAE in SW fluxes does not exceed 0.3 W m^{-2}. These errors are small compared to earlier studies, for instance, the RMSE in bottom-of-atmosphere downwelling SW flux was 1.024 W m^{-2}, much lower than the 5.85 W m^{-2} reported in Lagerquist et al. (2023) who used U-nets, and which in turn was much lower than the errors of tens of W m^{-2} obtained in Song and Roh (2021); Song et al. (2022) using DNNs. This was despite the use here of the Huber loss function due to it offering a balance between RMSE and MAE, leading also to reduced bias errors compared to RMSE. R^2 values were 0.999,98 in the SW and 0.999,993 in the LW.

Heating rates are also are excellent, with MAE typically less than 0.03 K d^{-1} (LW) and 0.01 K d^{-1} (SW) although higher at the surface (0.14 K d^{-1} for LW) and upper atmosphere. While these are slightly larger than those reported for the best model in Yao et al. (2023), they used much more complex LSTMs (1.12 million parameters compared to our ∼60,000 in the LW plus ∼125,000 in the SW), did not include aerosols, and used a much coarser

vertical grid. RMSE in near-surface heating rate was 0.83 K d^{-1} in the LW and 0.2 K d^{-1} in the SW in Lagerquist et al. (2023), while the respective values here are 0.235 and 0.067. The state-of-the-art offline results demonstrate the effectiveness of a handcrafted emulation approach to mimic ecRad's information flow. The LSTMs are also fast to train as optimized CUDA kernels are available, the training on 21 million columns using TensorFlow and one A100 GPU taking less than 24 hr to converge.

2.3. Prognostic Runs

The LSTM-based emulators were implemented in cycle 47r3b of the IFS using Infero (https://github.com/ecmwf/infero), an ML inference library being developed at ECMWF which supports multiple back-ends such as ONNX and TensorFlow-C. A suite of 10-day coupled ocean-atmosphere simulations spanning the northern hemisphere summer (June–August 2021), initialized at successive days within this period (92 runs in total) were performed at roughly 30 km horizontal resolution with and without emulators. The emulators replace the physical computations at all levels, which did not cause issues as even at the top level of 0.02 hPa the offline MAE in heating rates did not exceed 10%.

Figure 3 shows the impact of using emulators on forecast skill as measured by temperature RMSE. At the end of the 10-day forecasts significant differences are seen only in the northern hemisphere between 10 and 100 hPa. These changes are actually positive (up to 4% reduction in RMSE); however, this is most certainly a lucky accident as the higher skill is associated with a slight warming in this region (Figure 4) which acts to decrease existing biases. These very small differences indicate that the RNNs emulate the radiation scheme very closely. For comparison, changing the radiative transfer solver from TripleClouds to SPARTACUS impacts temperature RMSEs by close to 10% in the troposphere and stratosphere (Figure S5 in Supporting Information S1). Switching to the McICA solver, which is otherwise similar to TripleClouds but has stochastic noise, leads to similar RMSE changes, up to 4% at specific heights, as switching from reference TripleClouds to the emulator (not shown). The emulators' impact on mean temperatures is slightly higher than for forecast skill, but still extremely reasonable, being statistically significant only between 10 and 100 hPa and the mid-latitude troposphere around 30°(Figure S6 in Supporting Information S1).

Overall, the results are highly promising in terms of accuracy: replacing complex physics-based radiation computations across the full 137-level vertical grid with NN emulators did not substantially affect coupled ocean-atmosphere simulations spanning an independent testing year. Especially in terms of forecast skill, the differences were statistically insignificant in most regions. This is a considerable achievement due to especially the stratosphere and mesosphere being sensitive to small biases in radiative heating rates. Previously, V. Krasnopolsky et al. (2012) tested DNN-based radiation emulators in the NCEP Global Forecast System Model configured with 64 vertical levels and found small differences to control runs overall, but the bias in 200 hPa temperature over the Northern Hemisphere reached 0.4 K relative to control after 6 days (their Figure 7). Song and Roh (2021) tested similar emulators of the RRTMG scheme in a 37-level regional NWP model and found that domain-average RMS errors in skin temperature over the Korean peninsula reached 0.6 K in the middle of 24-hr long runs of 48 different days. By comparison, in our simulations the differences in skin temperature after 24 hr did not exceed 0.05 K anywhere in the globe (Figure S6 in Supporting Information S1), and similarly at 200 hPa the domain-average differences to TripleClouds-control were well below 0.05 K (Figure 4).

Although our main focus was on TripleClouds, we also trained SPARTACUS emulators. In an attempt to trade a little accuracy for a bigger gain in speed, the number of hidden neurons was increased from 64 to 96 while the number of LSTM layers in the SW model was decreased from 3 to 2, making the inference faster despite a higher model complexity. This attempt was perhaps not successful, as the SPARTACUS runs exhibit significantly larger temperature differences to their respective control runs (Figure 4). We did not perform further ablation studies to confirm whether this was due to reducing the number of LSTM layers or because of a more difficult emulation task. In offline evaluation SPARTACUS emulator has similar or even smaller heating rate errors than the TripleClouds emulator, but biases in SW fluxes are several times larger (Figure S4 in Supporting Information S1).

We also briefly tested convolutional architectures for SW emulation. Deep convolutional models using self-attention were found to be competitive with RNNs in offline metrics, however, in preliminary online testing they suffered from issues with instability, which RNNs did not. Radiation emulators based on simple DNNs have previously been tested prognostically in climate-like simulations, and were stable in year-long (Pal et al., 2019)

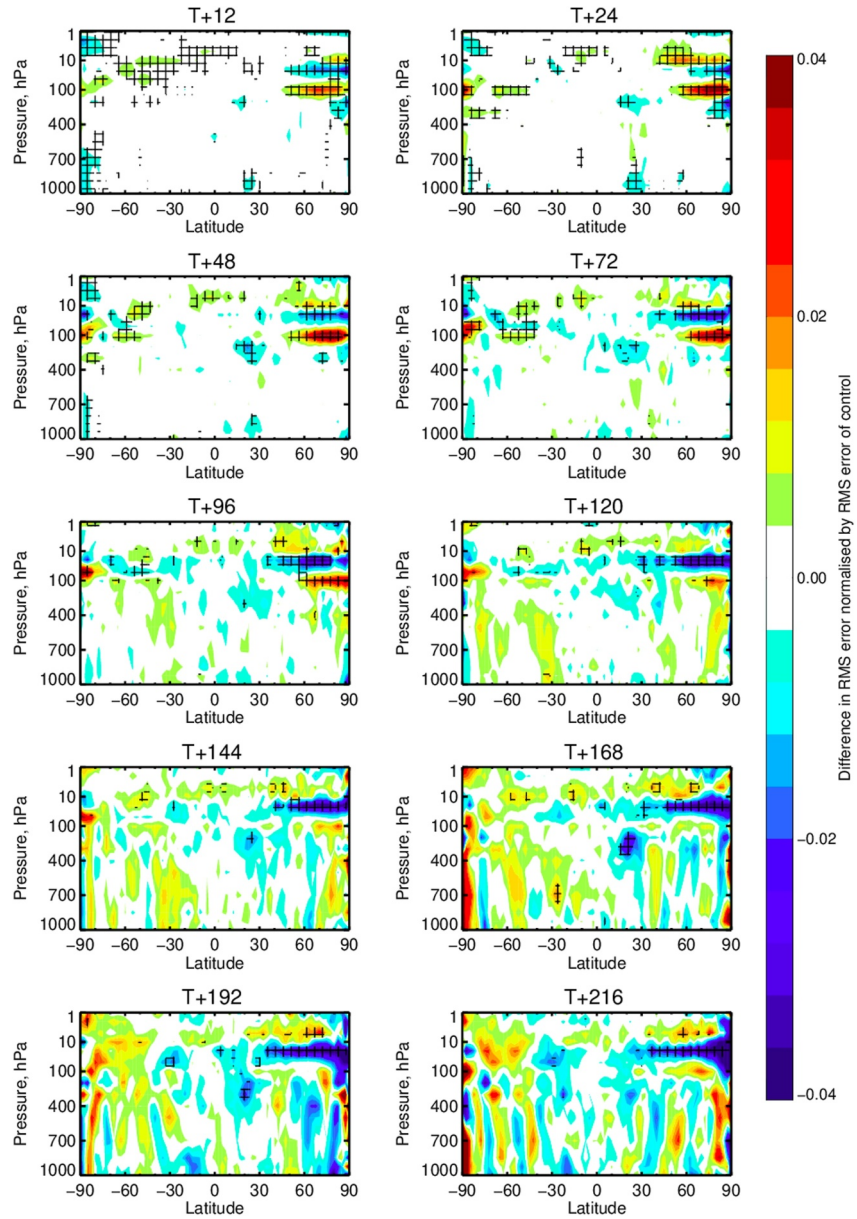

Figure 3. Latitude-pressure plots of the change in forecast skill (TripleClouds-RNN - TripleClouds-Control) as measured by RMS of temperature in a suite of JJA simulations (82–92 samples). Red (blue) indicates degraded (improved) forecast skill by using Long Short-Term Memory emulators. Cross-hatching indicates significant differences with 95% confidence.

and even decadal (V. M. Krasnopolsky et al., 2008) simulations. Emulating a radiation scheme presents smaller challenges for stability than trying to learn a new parameterization from convection-permitting simulations, especially as it lacks aleatoric uncertainty and can easily be formulated in terms of predicting fluxes as opposed to tendencies. Therefore, our focus here has been on accuracy, and we leave it to a future paper to test the stability of RNNs against other architectures in longer simulations.

2.4. Computational Performance

Unlike other atmospheric processes, using ML for radiative transfer, based on emulating existing schemes, has always been purely about acceleration. Especially in coarser-scale climate simulations, radiation has historically been one of the most expensive components (Cotronei & Slawig, 2020). But crucially, the speed/accuracy trade-off of radiation schemes can very greatly. While most schemes used today are based on some variant of the

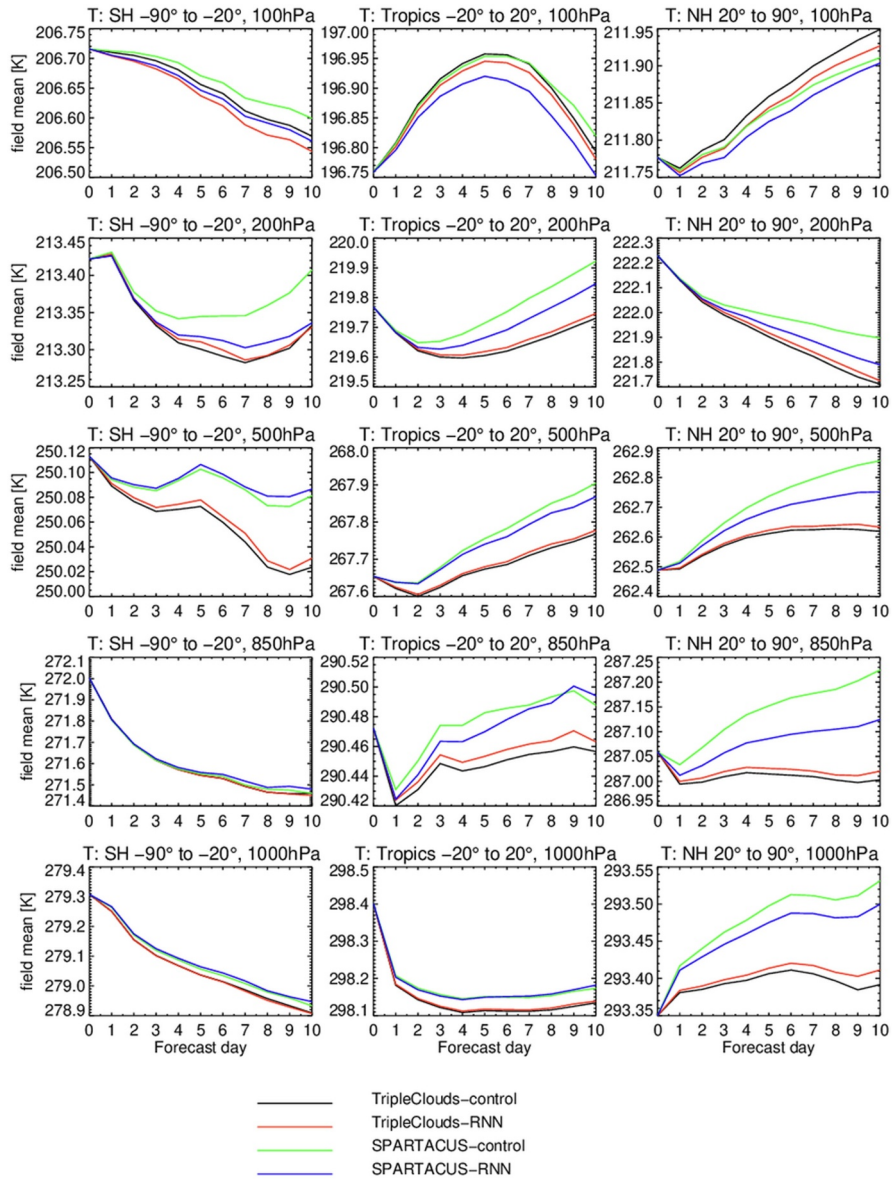

Figure 4. Mean temperature in the suite of JJA IFS simulations (solid lines, each based on 82 samples) at different pressure levels (rows) and different latitudes (columns; SH = Southern Hemisphere, NH = Northern Hemisphere).

correlated-*k*-distribution (CKD), a method which treats gas absorption via *N* pseudo-monochromatic calculations, accuracy for a given spectral resolution can vary substantially (Hogan & Matricardi, 2022). In addition, the computational performance of radiation codes also varies. This makes it important to evaluate the speed/accuracy trade-off of emulators against state-of-the-art schemes, as an outdated one may be one or two magnitudes slower. Therefore we evaluate LSTM-based emulators against a recent version of ecRad which features both kernel-level optimization and high-level refactoring aimed at collapsing spectral and vertical loops (Ukkonen & Hogan, 2024). Combining optimization with a new spectrally reduced gas optics scheme, ecCKD, which offers similar accuracy as operational schemes at a much lower cost (the number of computations being governed by spcetral resolution), ecRad's TripleClouds solver ran 12 times faster than the operational ecRad version in cycle 47r3 of the IFS, which uses RRTMG gas optics and the McICA solver.

Figure 5 shows the time-to-solution and energy-solution of emulators running on a single NVIDIA A100 GPU versus optimized ecRad running on a full AMD EPYC 7H12 node (2 CPUs with 128 cores in total) on the ECMWF HPC. The two systems are roughly equal in purchase cost. While CPU-to-GPU is apples-to-oranges,

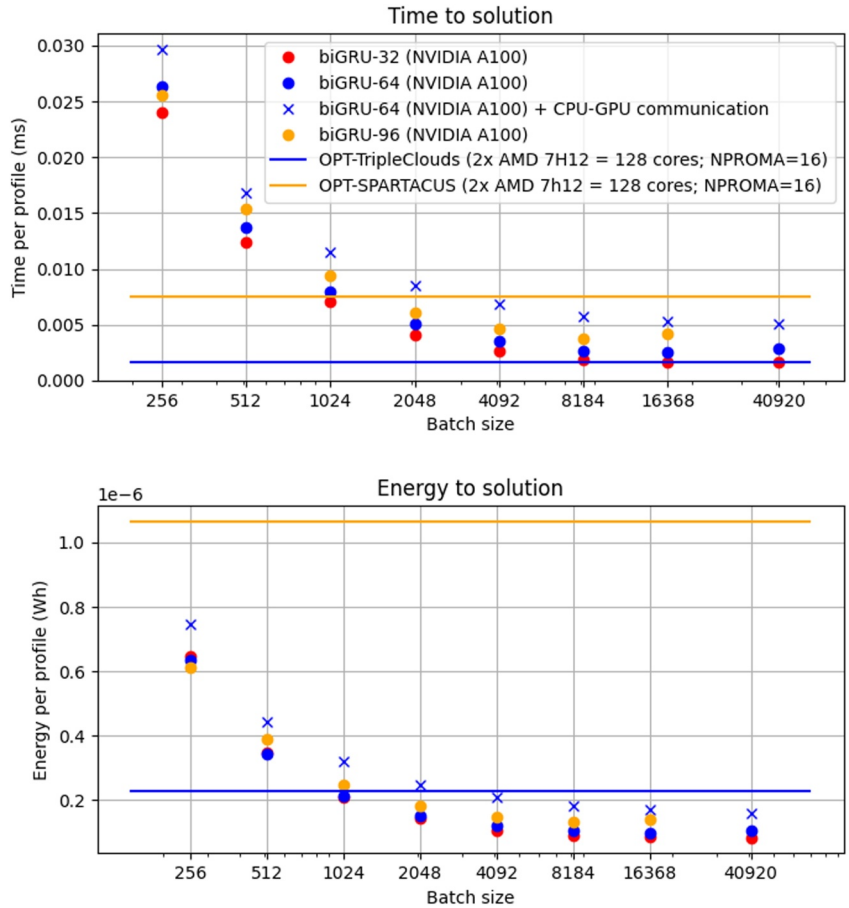

Figure 5. Time-to-solution (top) and energy-to-solution (bottom) for 2x bidirectional Gated Recurrent Unit models of various complexity (circles and crosses) running on an A100 GPU against the optimized TripleClouds and SPARTACUS schemes (horizontal lines) with ecCKD gas optics running on a full CPU node. Batch size is varied for emulators only (*x*-axis) and not ecRad, whose batch size (NPROMA) was fixed to a typical value in a CPU-based Numerical Weather Prediction model (16). The largest batch size tested (40,920) may be infeasible in practice, especially in the IFS where radiation runs on a coarser grid, but optimal performance is already obtained at a batch size of 8,184. Runtime was measured from Python by repeated calls on the same data following a warmup phase. Energy consumption was measured using the Eco2AI Python library.

this comparison is a more meaningful one from the perspective that one of the main motivations for developing radiation emulators is as a "code porting tool" which would enable easy and efficient utilization of powerful GPU hardware without laborious manual porting of existing ESM codes. With the aim of providing a best-case scenario for emulator performance, they are furthermore benchmarked in an uncoupled setting (independent from both IFS and ecRad) with the ONNX inference library, and using GRU-based models instead of LSTMs, since they were 15%–40% faster but only slightly less accurate.

Despite the emulator-friendly setup, emulators struggle to be competitive against TripleClouds (coupled to ecCKD) in runtime. The results are even worse when including CPU-GPU communication within the ECMWF GPU node, after which emulators are always several times slower than optimized TripleClouds. This may be surprising, but we again note the physical code represents a large effort in pushing the speed/accuracy of radiation schemes as far as possible, being able to compute 629 profiles ms^{-1} on the two-CPU-node. Against a more typical radiation scheme, for example, the currently operational IFS scheme, the emulators would typically be faster. Emulators do perform better in terms of energy-to-solution: while on par with or slightly better than TripleClouds depending on batch size, emulators are several times more energy efficient than SPARTACUS running on CPU (assuming batch sizes of at least 256). A major reason is presumably the higher energy efficiency of A100 GPUs compared to AMD Rome CPUs. We also measured floating point operations and found that 2x 64-neuron biGRU models perform roughly 13 times more computations than ecCKD+TripleClouds. While going from 64 to 94

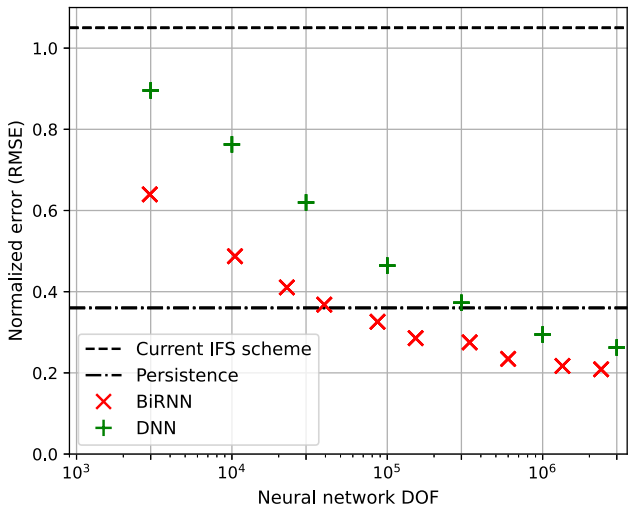

Figure 6. Offline performance of non-orographic gravity wave drag emulation comparing fully-connected (dense) NNs with bidirectional GRUs. The *x*-axis shows the model complexity as measured by the number of trainable parameters. The *y*-axis shows the RMSE error on the test set. See Chantry et al. (2021) for further details on the baselines.

neurons roughly doubles the number of floating point operations, time-to-solution increases by a much smaller factor, indicating that larger models can better utilize the GPU. At the same time, more complex RNNs are not necessarily required to emulate a higher-complexity scheme, as suggested by at least the offline results (Figure S3 and S4 in Supporting Information S1).

This means that emulation of relatively expensive, high-complexity schemes such as SPARTACUS may be more promising. But on the whole, while having established that radiation emulators can be sufficiently accurate for global NWP, our results do not strongly encourage their practical use: if very fast yet accurate radiation schemes are now available, why use emulators which are potentially inaccurate? Particularly in climate models, generalization is a large challenge even for the best ML architectures. Arguably, more resources should instead be spent on further optimizing radiative transfer code and porting them to GPUs. The focus of the rest of the paper is therefore on other processes which are more poorly represented in ESMs, but which could benefit from the sequence learning approach due to also being vertically non-local.

3. Non-Orographic Gravity Wave Drag

Gravity wave drag is typically parametrized as orographic and non-orographic schemes, with the latter capturing the effect of sub-grid gravity waves generated by fronts and convective activity. Non-orographic gravity wave drag (NOGWD) processes have a direct impact on momentum and temperature tendencies, and accurate parametrization is required to maintain accurate zonal-mean winds (Choi et al., 2018) and ensure accurate prediction of the quasi-biennial oscillation (Kang et al., 2020). Learning of gravity wave drag has been explored by several groups, with good success (Chantry et al., 2021; Espinosa et al., 2022; Pahlavan et al., 2023; Sun et al., 2023). In Chantry et al. (2021) the authors generated a NOGWD data set from the parametrization used in ECMWF's IFS model and used a fully connected NN to represent the process. Broadly the NOGWD algorithm has three components: stability calculations local to each vertical layer, followed by upwards propagation of waves which deposit momentum depending on stability and the momentum budget, and finally a correction for critical layer momentum deposition which propagates down through the atmospheric column. This motivates our use of a bidirectional RNN, where the first one is upward.

The original study tested dense NNs and found optimally scoring models across a range of model complexity levels, measured by the number of trainable parameters, which we loosely interpret as degrees of freedom (DOF). We train bidirectional GRUs for a similar range of complexity (Figure 6), having found that they achieve similar performance to LSTMs with fewer parameters. Consistently we find that the GRU models achieve higher accuracy using a smaller number of trainable parameters. In Chantry et al. (2021) a model with 300,000 DOF (achieving a normalized RMSE below 0.4) was tested online and found to be stable and accurate. An equivalent offline score to this model is here achieved with only 40,000 parameters, and the accuracy ceiling seems to be slightly higher.

These results highlight the parameter efficiency of RNNs. The downside of their sequential nature (here across 60 vertical layers) is slower inference compared to simple DNNs. Using the ONNX Runtime library on GPU (NVIDIA A100), prediction of 1.6 million columns took roughly 9 s with the GRU model; by comparison the 300,000 DOF DNN was roughly 4 times faster. Similarly to the radiation example, here we are dealing with emulation of an existing scheme, so any advantage in skill and generalization should be carefully weighed against the computational cost. Unfortunately, popular ML libraries are not necessarily well optimized for small NNs ($\lesssim$128 neurons), with a bespoke CUDA-based library outperforming TensorFlow in inference by up to 10x (Figure 7 in Müller et al., 2021). For the purposes of scope, and because the higher-complexity NOGWD scheme has limited impact on forecast skill, we did not test RNN-based NOGWD emulators in the IFS. In Chantry et al. (2021), DNN emulators manually implemented into an older IFS cycle and running on CPU were not faster than the operational NOGWD scheme. By contrast, the slower inference compared to DNNs is unlikely to be an issue in the next applications, as emulators are almost guaranteed to be faster than running a higher-resolution dynamical model or superparameterization.

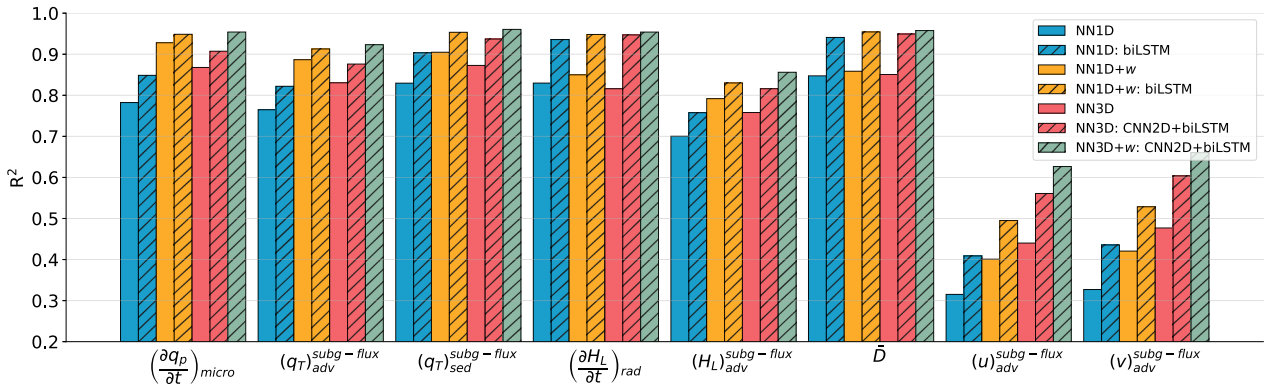

Figure 7. Offline accuracy of NNs emulating all sub-grid processes in a unified parameterization. The figure shows the global coefficient of determination $\left(R^2\right)$ for the test data grouped by output variable, using either a typical single-column (NN1D) or non-local (NN3D) Machine learning parameterization, some of which include vertical velocity ("+w"). Models of the same color use the same inputs. Solid bars are for DNNs trained in the original paper (Wang and Yuval et al., 2022), whose Figure 2 we replicate, while the hashed bars are for biLSTMs operating in the vertical dimension; for non-local parameterization this was combined with 2D horizontal convolutions (hashed red and green bars). From left to right, the outputs are (Wang and Yuval et al., 2022) the tendency of total precipitating water mixing ratio due to microphysics, subgrid flux of total non-precipitating mixing ratio due to vertical advection and sedimentation, subgrid tendency due to radiation, subgrid energy flux due to vertical advection, coarse-grained diffusivity and subgrid zonal and meridional momentum flux due to vertical advection.

4. Non-Local Unified Parameterization

Atmospheric models today use a single-column framework for parameterizations, relying on the idea that sub-grid processes primarily rearrange mass, momentum and energy in the vertical. Wang and Yuval et al., 2022, (hereafter PW22) Wang and Yuval et al., 2022 revisited this assumption by coarse-graining data from convection-permitting aquaplanet simulations and found that the offline prediction of especially subgrid momentum transport was improved when including neighboring columns in their feed-forward NN inputs. This parameterization problem therefore highlights a possible limitation of RNNs; that they operate on 1D sequences. In addition, by considering a unified parameterization where variables associated with different processes are predicted by a single NN, we can explore which sub-grid processes benefit more from a more advanced ML architecture designed to capture non-locality.

Since RNNs operate on 1D sequences, they cannot inherently represent non-locality in three spatial dimensions. On the other hand, the primary direction of sub-grid transport is still typically in the vertical and PW22 found that including only neighboring columns is adequate and helps to avoid a substantial increase in computational cost. Supplementing the vertical RNNs with convolutions in the horizontal therefore seems appropriate. One approach is to use a convolutional LSTM (ConvLSTM) where the input-to-state and state-to-state operations of an LSTM are replaced with a spatial convolution. The architecture was first proposed for a spatiotemporal precipitation nowcasting problem (Shi et al., 2015). However, we find that simply feeding the 3×3 columns to a Conv2D layer without padding (with a kernel size of 3), and connecting the resulting 1D column to a bidirectional LSTM similar to the one depicted in Figure 1, has similar accuracy (Figure S8 in Supporting Information S1) while using much fewer parameters and being much faster to train using Tensorflow. Rather than interleaving the horizontally and vertically non-local operations, they are thus performed in two separate steps.

Figure 7 shows the offline evaluation of fully connected networks alongside RNN-based models, as measured by the global coefficient of determination R^2. Using RNN-based models (dashed bars) instead of DNNs (solid bars) generally improves the prediction of sub-grid processes. In a single-column framework, RNNs are able predict radiative transfer, diffusivity (turbulence) and sub-grid momentum transport with substantially better skill $\left(\Delta R^2 \approx 0.1\right)$. When incorporating neighboring columns in a non-local parameterization, combining RNNs with horizontal convolutions again strongly outperforms $\left(\Delta R^2 \approx 0.1\right)$ fully connected NNs which use flattened 3D inputs. For this problem, we did not strive for parameter efficiency and tuned the models for optimal or near-optimal validation loss. The final model has 160-192-160 neurons in the Conv2D, upward LSTM, and downward LSTM layers respectively. They thus have more trainable parameters (~500,000) than the 3-layer DNNs (~190,000) trained in Wang and Yuval et al. (2022), who also tested more complex DNNs and found they did not

improve performance. We tested simpler non-local Conv + LSTMs and found that they outperformed the reference models even when using fewer than 100,000 parameters.

As noted by PW22, using non-local inputs mainly benefits the prediction of sub-grid momentum transport. The lack of improvement in microphysics and radiation is presumably due to these processes being parameterized under a 1D framework in the high-resolution simulations that were coarse-grained, as they generally are. In the current modeling paradigm, the inclusion of non-local inputs acts to only benefit the prediction of sub-grid momentum transport. For other variables the inclusion of divergence (not tested here) or vertical velocity w (yellow bars in Figure 7) is basically equivalent. PW22 found that the improvement from including neighboring inputs was especially notable in cases characterized by mid-latitude fronts and convective instability. As these are associated with high-impact weather, changing the code infrastructure of the dynamical model to allow 3D parameterization could perhaps still be worth it.

Finally we note that including w as input to non-local models further improves the prediction of sub-grid momentum transport: Conv2D+BiLSTM using (3D) w achieves R^2 exceeding 0.6, slightly better than the equivalent non-local model without w $\left(\Delta R^2 \approx 0.05\right)$, while the most accurate reference model (non-local DNN without w) had R^2 values below 0.5. PW22 did not evaluate the skill of such models which use both 3D inputs and w, perhaps due to concerns the authors raised about the potential for ML schemes to become unstable by learning non-causal relationships between inputs and outputs.

5. Moist Physics

Deep convolutional networks have been shown to outperform fully connected NNs in emulating radiative transfer and moist physics (Han et al., 2020, 2023; Lagerquist et al., 2021), which makes them arguably a more relevant baseline to compare RNNs to. Although better-known for extracting spatial features in image processing applications, models based on 1D convolutions have been successfully applied to sequence learning tasks, even outperforming LSTMs (Gehring et al., 2017) due to easier optimization, the receptive field being adjustable by varying model depth. Dilated convolutions in so-called Temporal Convolutional Networks (Lei et al., 2017) enable exponentially large receptive field by network depth and have been used in for example, time series forecasting.

Therefore, in our final problem we adapt the data and code from Han et al. (2020, hereafter H20) and Han et al. (2023, hereafter H23) who emulated moist physics under realistic orography by training deep convolutional networks with skip connections (ResNets) on SPCAM. SPCAM is a superparameterized version of the CAM5 model which nests a 2D cloud-resolving model (CRM) inside each grid cell of CAM5. H23 evaluated complex ResNets (with 32 batch-normalized layers) prognostically in a 5-year simulation. The simulation was stable and improved the intensity and diurnal cycle of precipitation, but also had large biases in moisture and temperature at high latitudes. To improve prediction these studies incorporate the time dependence (memory) of convection by including variables from prior timesteps as inputs. These convective memory variables include not only past inputs to the CRM - profiles of temperature, moisture, their large-scale tendencies from the host model, and surface sensible and latent heat fluxes - but also outputs (tendencies, cloud water and cloud ice) of the previous time step.

In order to tune the RNNs with respect to independent data, we split the last simulation year (used in H23 for testing) into validation (April–June i.e., 3 months, with 800 grid points randomly sampled each day as in H23) and testing (spanning July-March i.e. 9 months); training data comes from the previous year as in H23. The ResNet architecture and training hyperparameters such as learning rate, batch size (2048) and λ, which controls the contribution of a loss term that penalizes the violation of conservation of column-integrated moist static energy h, and which is added to MSE loss, are identical to H23, but layer reference pressure was added as additional input to all models.

5.1. How to Represent Convective Memory in ML Parameterization?

Since it's unclear what inputs (if any) should be used to represent memory, we first trained RNNs and ResNets with three different setups each: (a) with no convective memory, (b) using both SPCAM inputs and outputs of the previous timestep, and (c) using only the previous inputs (not tested in H20 or H23). Our model consists of an upward 160-neuron LSTM whose initial state is predicted from surface variables using a small DNN, connected to a downward LSTM with 96 neurons. Training and validation curves for the final models are shown in Figure 8. Obvious signs of overfitting are seen only for ResNets without memory, whose energy conservation on the validation data starts to worsen after roughly 100 epochs. The ResNets, which were tuned in Han et al. (2023),

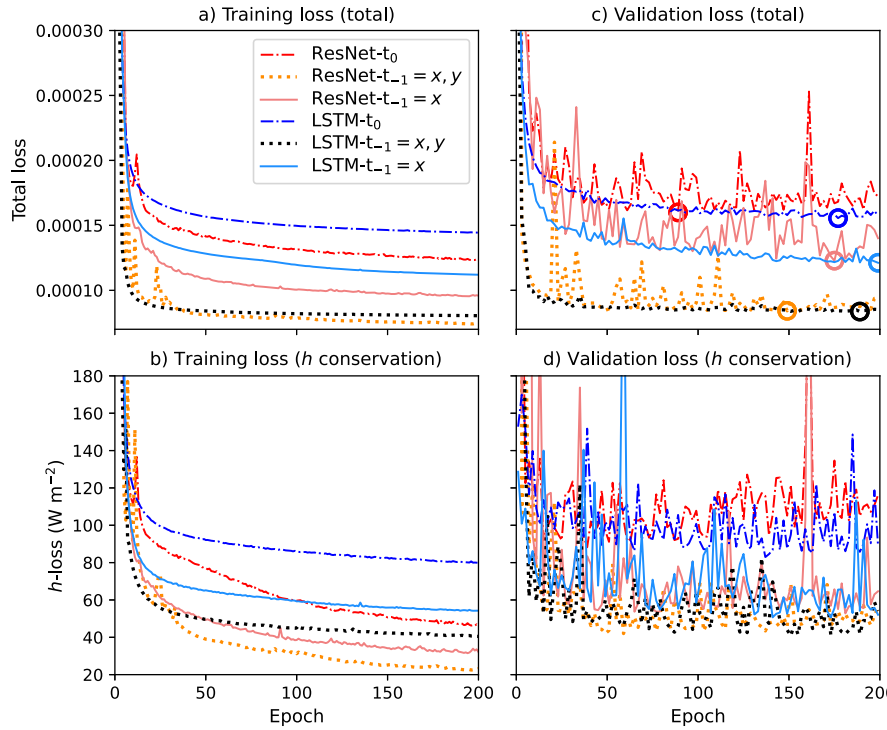

Figure 8. Total loss (a, c) and moist static energy conservation (b, d) with respect to training (a, b) and validation (c, d) data when training ResNets and bidirectional LSTMs on the moist physics data set (Han et al., 2023). The inputs consist of GCM grid-scale state variables and tendencies used to force the cloud-resolving model inside SPCAM, and dotted and solid lines use additional inputs to represent convective memory: both inputs and SPCAM outputs of the previous time step ($t_{-1} = x, y$), only previous inputs ($t_{-1} = x$). The dotted lines correspond to the approach used in Han et al. (2023) except we consider only one previous timestep. Circles indicate the epoch where minimum validation loss was reached.

have 1.5 million parameters while the biLSTM has 220,000 parameters. In general, the ResNets continue to improve their training loss while their validation loss seems to flatten out; by contrast the RNNs' training and validation curves are more similar. Minimum validation losses are very similar for both architectures. Including previous SPCAM outputs as NN inputs (Method 2, used in H20, and H23) gives the best validation performance overall (albeit similar energy conservation as Method 3).

However, in semi-prognostic testing we found that ResNets trained on previous outputs can become unstable, with only shallower ResNets remaining stable (Figure 9a). Here the models are evaluated autoregressively, so that SPCAM outputs are used for memory only for the first time step(s), after which past predictions are used. The results are surprising given that H23 achieved stable online integration in a GCM with a 32-layer ResNet. Testing their learned model, it was stable as expected. Due to randomness in the training procedure, we further tested two more 32-layer and two more 22-layer ResNets, and all were unstable within a few days except one 22-layer ResNet which produced NaNs only at higher latitudes. In total, we therefore tried 6 deeper ResNets (>20 layers), of which only one was (partially) stable, 4 shallower ResNets which were stable, and 7 RNNs which were all stable.

We are unsure how indicative these tests are of stability in online testing, but intuitively regard them as weaker tests of stability, since there is a feedback in this case only on convective memory but not other SPCAM inputs. Our relative difficulty in training stable deep ResNets that are stable (compared with H20, H23 which did not mention such issues) is surprising but could be related to our training code being different: to speed up training and support a large validation set we wrote a TensorFlow data generator instead of calling the fitting function inside a data iteration loop. We also used only one past time step for memory and did not examine the impact of this choice on stability. To summarize, while these results do not show conclusively that ResNets or other deep convolutional models are inherently more unstable, all else being equal, we found RNNs to be more stable in offline autoregressive testing.

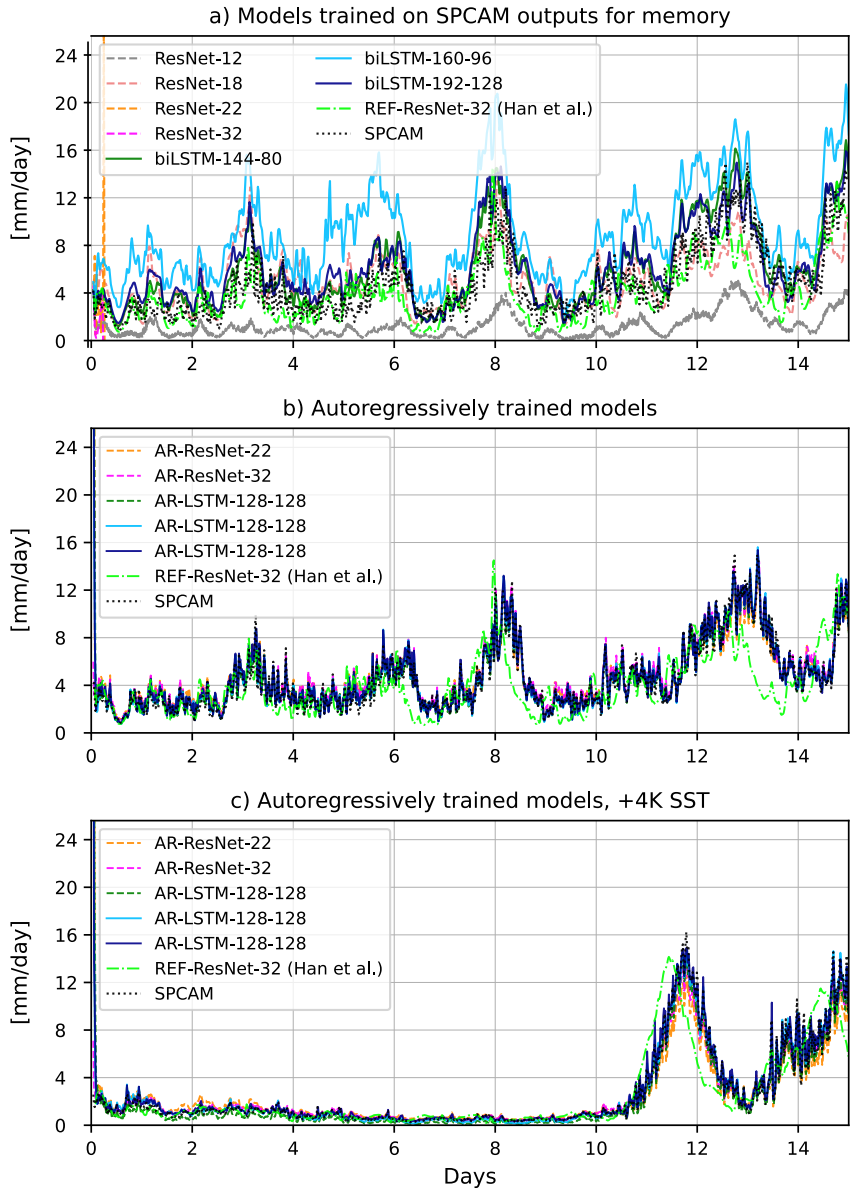

Figure 9. Time series of predicted precipitation over a tropical region (15°S–12°N, 110°E–130°E) by NNs using inputs and outputs from the previous timestep to represent convective memory (a) and for autoregressively trained NNs using a hidden variable for memory (b, c). In this semi-prognostic test using testing data from either a baseline climate (a, b) or a + 4K climate experiment (c), the memory inputs are updated from previous predictions. In a), models were trained using previous SPCAM outputs as memory and these are replaced with neural network predictions during testing. This made deeper ResNets (purple and red dashed lines) unstable within a few timesteps. Autoregressive training (b, c) resulted in also deep ResNets being stable and closely matching SPCAM (note the vertical lines for the first time steps in (b, c) is spin-up due to random initialization of hidden memory). The models were trained for 100 epochs. The ResNets have 0.5–1.5 $\times 10^6$ parameters (depending on number of layers) while the bidirectional LSTMs have 0.22–0.3 $\times 10^6$ (depending on hidden size).

5.2. Autoregressive Training

A possible issue with training on true past labels is that this leads to a discrepancy between training and inference, where past predictions must be used instead. This can lead to instability because the model was not trained on its own errors, which then grow over time. For this reason multi-step rollout during training has been used in many studies (e.g., for training GraphCast in Lam et al., 2022), where the model is fed back its own predictions in an autoregressive fashion and the loss is minimized over multiple steps. Longer rollouts can improve stability and

accuracy over longer time horizons (Kochkov et al., 2024). With this in mind we wrote new training code in PyTorch which supports autoregressive training (such models are hereafter prefixed with "AR") and generated new training data based on fixed locations which were sampled similarly to H23. The new data spans a continuous 9-month period and 1,024 fixed locations (changed from 800), equal to the new batch size which was halved from 2048 (learning rate for ResNet was correspondingly halved). Compared to "online training," which would require a differentiable host GCM interacting with the emulator during training, the training is still fully offline as there is no feedback on other NN inputs besides latent convective memory.

According to the semi-prognostic test, stability and generalization is much improved, with all ResNets tested also being stable and closely matching the SPCAM precipitation timeseries (Figure 9). An excellent match to SPCAM predictions is also obtained on the +4K SST data set from H23. By comparison, the non-autoregressively trained ResNet from H23 predicts the onset of precipitation slightly too soon. To represent convective memory, our new models use hidden variables. This allows more information to be encoded than SPCAM inputs/outputs, which are grid-scale means of high-resolution CRM tendencies and thus by definition cannot represent sub-grid structure which might be important for convective organization. Similarly, Shamekh et al. (2023) learned a low-dimensional latent representation of convective organization within a precipitation modeling task. Here we do not restrict the latent variable to be low-dimensional. After testing a few different approaches, we settled on feeding the final hidden variable as additional input to the first layer of the model. For the RNN-based model this means using the output of the second LSTM as memory (Figure 8). Its size is 128 in the revised model, which uses 128 neurons in both LSTM layers. The models were trained using a window of 5 time steps (of 20 min each, thus 100 min) after a warm up period of 1 step for the first two epochs and then gradually increasing to 5 every epoch.

Multi-step training requires gradients to be propagated over longer sequences (through both the vertical column and time). With this in mind we tested more recent sequence modeling architectures with claimed advantages over transformers and LSTMs for long-range temporal modeling. These are linear RNNs and State Space Models (SSMs) which introduce non-linearities only in local layers sandwiched between linear recurrent layers, enabling parallelization and supposedly being less prone to vanishing gradients. The results are summarized in Table 2. In practice we find that SSMs such as S5 (Smith et al., 2022) do not outperform LSTMs in moist physics emulation. The best validation loss during 100 training epochs is still obtained using LSTMs with hidden convective memory (1.37×10^{-4}), although some SSMs such as Gated State Spaces (Mehta et al., 2022) were only marginally behind in total loss and had slightly better moist static energy conservation (h-loss). Interestingly, we find that autoregressively trained LSTMs using previous NN outputs for memory are significantly worse (1.52×10^{-4}), as are traditional Bi-LSTMs where both RNNs operate on the input sequence (1.66×10^{-4}). The superior performance given the same number of neurons suggests that connecting the second RNN to the first, which mimics for example, radiative transfer equations, is different to regular Bi-RNNs in the transformations it can effectively represent. However, reversing the direction of the LSTMs ("down-then-up") leads to only minimally higher validation loss (1.42×10^{-4}) than our chosen "up-then-down" structure (Figure 10). Vanilla RNNs were also tested; these had very poor performance. We do not have a clear understanding why gating mechanisms are so important here when they were successful in emulating precipitation within a microphysics scheme in Perkins et al. (2024), but note that emulating full moist physics is much more complex compared to the autoconversion of cloud condensate with a unidirectional (downward) dependence.

The loss obtained for AR models is similar to models trained on past inputs, but significantly higher than for those trained and evaluated using past SPCAM outputs, that is, the dotted lines in Figure 9, which reach validation losses below 1.0×10^{-4}. But to what extent can offline scores from non-autoregressive evaluation reflect the performance in a prognostic setting, where true labels are not available? The roughly 50% difference in validation loss between the dotted and solid lines in Figure 9 is likely due to inherent stochasticity in the emulated process (sub-grid convection predicted by a nested CRM). Similarly, the autoregressively trained models cannot either reach such low loss values (a type of irreducible, i.e. aleatoric uncertainty). Temporal autocorrelation of the emulated processes implies that using past true outputs amounts to a type of cheating due to containing stochastic information from the previous time step.

As a measure for generalization, Table 2 lists the ratio of minimum h-con between validation and training data. While vertically recurrent models all have ratios close to 1, the ratio for ResNets is around 1.4, a sign of substantial overfitting. AR-ResNets in particular have poor energy conservation on the independent validation data.

Table 2
Performance Summary for Moist Physics Emulators Which Differ in Architecture, Convective Memory Input and Training Methodology.

	NN backbone	Neurons in RNN/conv. Layers	Para-meters ($\times 10^6$)	Max. Val. R^2	Min. Val. Loss ($\times 10^{-4}$)	Train loss ($\times 10^{-4}$)	Min. *h*-loss (train) Wm^{-2}	Min. *h*-loss (val) Wm^{-2}	Min. *h*-loss, ratio (val/tr)
LSTM	LSTM	(160, 96)	0.21	0.922	1.56	1.44	79.9	84.5	1.06
$LSTM_{mem=x}$	LSTM	(160, 96)	0.22	0.943	**1.27**	1.12	54.2	**50.5**	0.93
$ResNet_{mem=x}$	ResNet, 32 layers	128	1.50	0.938	1.30	1.00	38.5	53.7	*1.39*
$AR\text{-}LSTM_{mem=y}$	LSTM	(128, 128)	0.26	0.925	1.50	1.34	61.3	62.7	1.02
$AR\text{-}LSTM_{mem=h}$	LSTM	(128, 128)	0.27	0.933	**1.37**	1.29	55.5	55.4	1.00
[a]$AR\text{-}LSTM_{mem=h}$	LSTM	(128, 128)	0.27	0.911	1.66	1.47	79.8	66.8	0.84
[b]$AR\text{-}LSTM_{mem=h}$	LSTM	(128, 128)	0.27	0.926	1.42	1.22	57.3	55.5	0.97
$AR\text{-}GRU_{mem=h}$	GRU	(160, 160)	0.31	0.916	1.65	1.50	79.3	72.8	0.92
$AR\text{-}RNN_{mem=h}$	RNN (vanilla)	(128, 128)	0.07	0.833	3.22	3.72	224.0	152.6	0.68
$AR\text{-}ResNet_{mem=h}$	ResNet, 32 layers	128	1.50	0.908	1.79	1.17	46.1	65.7	*1.43*
$AR\text{-}SRU_{mem=h}$	Simple Recurrent Unit (Lei et al., 2017)	(128,64)	0.32	0.930	1.44	1.26	56.9	61.4	1.08
$AR\text{-}S5_{mem=h}$	S5 (Smith et al., 2022)	(224, 112)	0.30	0.931	1.42	1.21	51.3	50.9	0.99
$AR\text{-}GSS_{mem=h}$	GSS (Mehta et al., 2022)	(128, 64)	0.33	0.931	1.39	1.22	48.3	**50.7**	1.05
$AR\text{-}QRNN_{mem=h}$	Quasi-Recurrent NNs (Bradbury et al., 2016)	(192,96)	0.36	0.930	1.43	1.22	51.0	51.4	1.01
$AR\text{-}Gateloop_{mem=h}$	Gateloop (Katsch, 2023)	(256, 128)	0.30	0.927	1.50	1.27	59.9	61.6	1.03

Note. The first three models are those shown in Figure 8 and were trained using TensorFlow on 800 grid points re-sampled daily, whereas subsequent models were trained autoregressively (AR) using PyTorch on the time-continuous data set with 1,024 fixed locations. All except the first AR model use a hidden variable for convective memory ("mem = h") instead of past predictions *y*. The best validation scores in both training categories are bolded. "Max." and "min." refers to best scores during 100 training epochs; due to the training dynamics being stable the results were not sensitive to random seed and only one model of each architecture was trained. However, for vanilla RNNs and GRUs which did have unstable convergence the best of 3 trials is shown. ResNets have the worst generalization with respect to energy conservation (italicized values). SRU, S5, GSS, QRNN, and GateLoop use a linear recurrence (in some cases based on convolution) which enables parallelism along the sequence dimension. In these, the bidirectional RNN architecture was modified slightly (for instance, level inputs go via a dense layer) to ensure conforming sizes of hidden units, inputs and outputs. Different variants of Bi-RNN connectivity were tested. [a]A traditional Bi-LSTM where both LSTMs use the input sequence as input, and whose outputs are concatenated before the final dense layer. [b]Reversed version iterating downwards first, and upwards with the second LSTM, which is initialized using surface information.

Although multi-step training improved the stability and skill in the semi-prognostic tests, AR-ResNets performed poorly against AR-LSTMs trained here and non-AR ResNets shown in H20. Figure 11 shows the latitude-pressure cross sections of mean heating and moistening compared to SPCAM for the 32-layer AR-ResNet and AR-LSTMs which use hidden variables for memory. (To avoid cherry-picking only the first models trained were evaluated on test data. Other models using the same hyperparameters were also trained and had very similar validation loss, with ResNets no longer exhibiting the substantial noise seen in Figure 8c). The AR-ResNet has large errors in zonally averaged moistening of 0.4–0.7 g kg^{-1} d^{-1} almost across the whole troposphere. By contrast, semi-prognostically evaluated ResNets in H20 (their Figure 4) had differences this large only below 800 hPa, and the AR-biLSTM only just above the surface, thus better than ResNets in either study.

Finally, we also evaluated non-autoregressively trained models which represent memory via past inputs only. In this case LSTMs and ResNets had similar offline accuracy. For instance, while LSTMs evaluated on the testing data had smaller biases in moisture tendency (Figure S9 in Supporting Information S1), cloud water and ice were predicted with similar skill (Figure S10 in Supporting Information S1), and LSTMs had a larger precipitation bias than ResNets in similar offline tests using the +4K SST data set (Figure S11 in Supporting Information S1). Due to inherent randomness of the training procedure, some variation was again found between different NNs using the same architecture. This variation was smaller than for autoregressively evaluated models trained on past true outputs, whose test performance varies considerably as seen in Figure 9a.

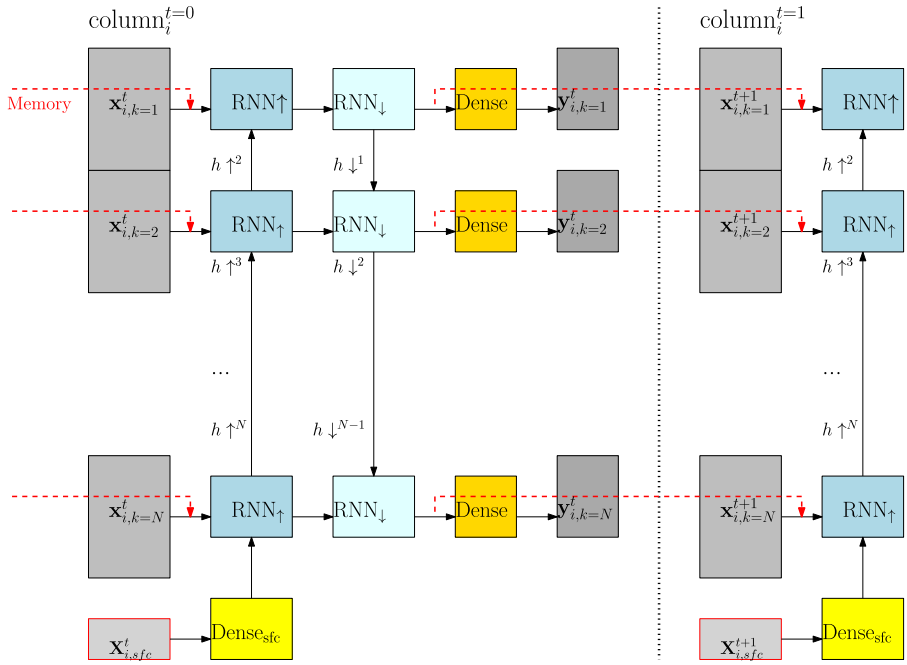

Figure 10. Extension of the vertical recurrent neural network (RNN) method to incorporate convective memory in parameterization of moist physical processes. Memory is represented using a latent variable (the output from the second RNN) instead of outputs y. Subscripts (i, k) are horizontal and vertical indices respectively (time step is given in superscript). The model is a vertically recurrent, temporally autogressive model where gradients were backpropagated through both the vertical column and temporal windows of up to 5 time steps.

6. Discussion and Conclusions

In previous work, vertically recurrent NNs which iterate through a vertical column were shown to emulate radiation schemes accurately in an offline setting. Here we first evaluated such radiation emulators prognostically with the IFS model, and found that they can emulate a full-complexity radiation scheme closely enough not to impact the accuracy of medium-range global NWP forecasts. This is a considerable achievement: emulators trained on one year of training data can fully replace radiation computations accounting for sub-grid cloud structure and involving dozens of aerosol and gas species at 137 model levels, reaching the mesosphere, in simulations for an independent testing year, without degrading accuracy. On the other hand, we may question the whole premise of emulating radiation schemes, as we find that recent advances in the efficiency of physical radiation schemes makes them generally faster than (accurate, RNN-based) emulators, which perform an order of magnitude more floating point operations. This means that future efforts should probably focus on further optimizing physical schemes (and porting them to GPU), or emulating higher-complexity radiation schemes.

If emulation of radiation is no longer promising, a more relevant question is whether vertically recurrent RNNs such as LSTM can emulate other, currently less accurately represented sub-grid processes with similar success. Processes such as convection also involve vertical transport, albeit with less non-locality than radiation, which flows through the entire column during a model time step. In the second part of the paper, we tested the sequence learning method on a wide range of sub-grid processes by using data published in other studies. Consistently we find that vertical RNNs outperform fully connected networks in offline evaluation. Compared to these simple NNs, RNNs trade off some computational speed for parametric sparsity and higher accuracy by encoding the principle of locality. It is only on the moist physics data set that RNNs do not strongly outperform the reference model in offline metrics, but here the baseline was a ResNet and not a fully connected NN. Our interpretation is that deep convolutional networks can also skillfully represent non-locality while respecting locality. However, a potential issue with CNN-based architectures it that they require deeper networks to capture larger receptive fields. This suggests they are particularly inefficient compared to RNNs for emulating radiative transfer, which requires a full receptive field, but we also found them slower for emulating moist physics: biLSTMs were faster to train than the 32-layer ResNets; around 140 versus 800 s per epoch (consisting of 21 million samples) on an

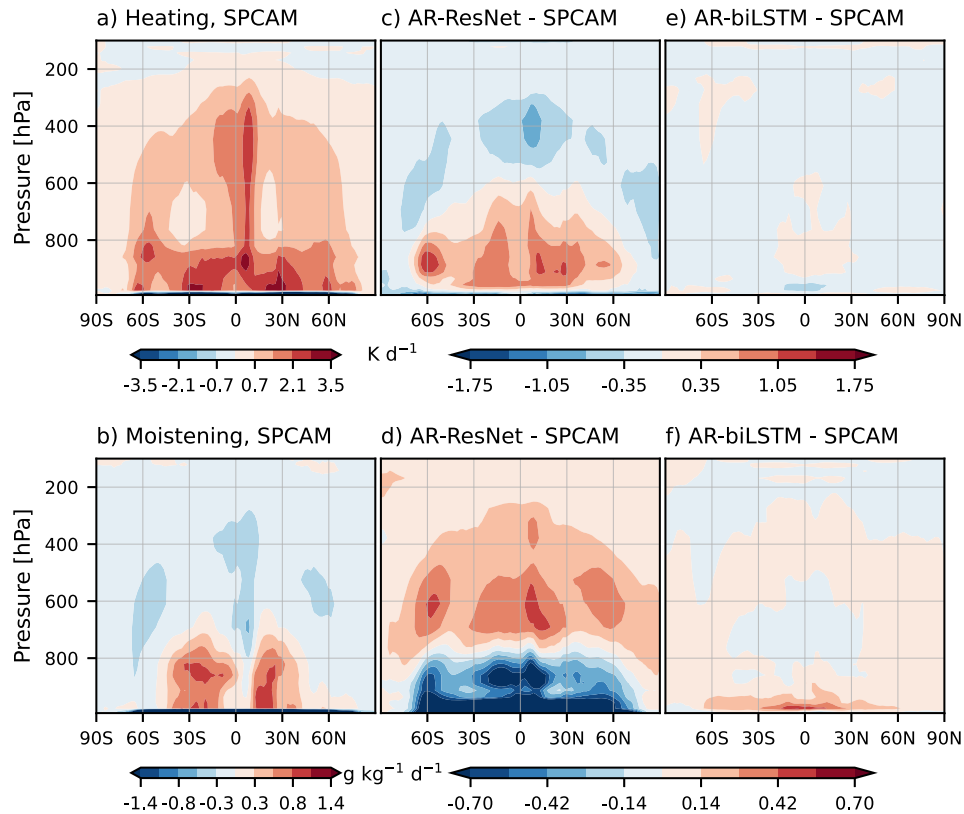

Figure 11. Zonally averaged predictions of convective heating (top row) and moistening (bottom row) tendencies from SPCAM (a, b), and their difference to semiprognostically evaluated ResNets (c, d) and LSTMs (e, f) which use hidden variables for convective memory and were trained autoregressively.

NVIDIA A100 when using a batch size of 2048, a difference of 5.7x. When including communication, the TensorFlow inference of LSTMs was roughly 4.8x faster the ResNet, which itself was reported to be 10x faster than SPCAM (Han et al., 2023).

Furthermore we find that RNNs do not show any sign of overfitting whereas ResNets' energy conservation on training and validation data started to diverge after around 50 epochs. This could be related to the results obtained in semi-prognostic evaluation of emulators trained on true past labels as input, where most of the deep ResNets we tested were unstable. Performance in these tests was improved for both architectures by (offline) autoregressive training using multi-step rollout, suggesting that this approach may be a useful compromise between conventional time-independent training and difficult-to-achieve online training. The best results were obtained with LSTMs using hidden variables for convective memory, which in principle can encode subgrid-scale variability. However, offline scores were similar to non-autoregressively trained models which used SPCAM inputs from the previous time step. In practice, it may be difficult to learn any sub-grid dynamics from data which consists only of grid-scale means. Exploiting sub-grid information when training ML parameterizations with convective memory would be an interesting avenue for future research.

In the fields of natural language processing and computer vision, RNNs have been surpassed by transformers (Vaswani et al., 2017), deep learning models which use a parallelizable attention mechanism to focus on the important parts of a sequence. A downside of transformers is that their complexity is quadratic by sequence length. In an offline evaluation of radiation emulators, transformers were among the best but lagged behind bidirectional LSTMs in accuracy, especially for top-of-atmosphere fluxes (Yao et al., 2023). In brief prognostic testing of convolutional radiation emulators with self-attention, we encountered issues with instability. From a physical point of view, RNNs are more attractive for modeling flows in a 1D column due to offering parametric sparsity and a good prior in that they are sequential, ensuring that a parcel of air or photon does not “skip over” an atmospheric layer. In the ML literature, RNN-type models with linear recurrences have recently made a

resurgence for modeling long sequences. While we did not find any improvement over LSTMs for autoregressive modeling of moist convection, these models were competitive in offline metrics and offer higher parallelization.

To conclude, our results suggest that vertically recurrent NNs are a promising method for parameterizing sub-grid processes, given their ability to efficiently model non-locality within a column. By enforcing a physical information flow, it′s possible that RNNs offer the best of both worlds with good accuracy, efficiency and stability. However, no clear evidence of the latter was presented in the current study, which, due to its wide scope, omitted prognostic evaluation of moist physics emulators. In a future paper we will explore whether the promising offline performance of autoregressively trained RNNs translates to good online performance in emulation of moist physics.

As ESMs and NWP models are being run at increasingly higher resolution, it may be justified to relax the single-column framework in future parameterizations. In such a case, RNNs may still be a useful ML framework as they can, for instance, be combined with convolutions across the horizontal domain in order to incorporate neighboring columns in a relatively parameter-sparse model as shown here. Finally, while using more physically inspired ML architectures can improve accuracy, stability and generalization of ML parameterizations, it is unreasonable to expect that it can completely solve these challenges. Hybrid parameterizations which combine NNs with physical equations or constraints, for example, using NNs only for predicting entrainment rates within a mass-flux scheme (Shin & Baik, 2022) or the optical properties within a radiation scheme (Ukkonen & Hogan, 2023), may be a further step in this direction by requiring even fewer parameters and training data while also being more interpretable than fully ML-based parameterizations.

Data Availability Statement

The Python code used in this paper has been uploaded to Zenodo (Ukkonen, 2024). This repository (https://doi.org/10.5281/zenodo.10462470) also contains code and data for training emulators of the ECMWF non-orographic gravity wave scheme (Chantry et al., 2021) and radiation schemes. Data used in sections 4-5 can be found through the original papers whose data and code we used (Wang and Yuval et al., 2022; Han et al., 2020, 2023).

Acknowledgments
P.U. thanks his former employer the Danish Meteorological Institute for the opportunity to carry out much of this research. P.U. received funding through a Leverhulme Trust Research Leadership Award, "Seamless Uncertainty QuantifiCation for Earth System prediction" (SUQCES). M.C. received funding from the MAELSTROM project funded by the European High-Performance Computing Joint Undertaking (JU; Grant agreement No 955513).

References

Belochitski, A., & Krasnopolsky, V. (2021). Robustness of neural network emulations of radiative transfer parameterizations in a state-of-the-art general circulation model. *Geoscientific Model Development*, *14*(12), 7425–7437. https://doi.org/10.5194/gmd-14-7425-2021

Bradbury, J., Merity, S., Xiong, C., & Socher, R. (2016). Quasi-recurrent neural networks. *arXiv preprint arXiv:1611.01576*. https://doi.org/10.48550/arXiv.1611.01576

Brenowitz, N. D., Beucler, T., Pritchard, M., & Bretherton, C. S. (2020). Interpreting and stabilizing machine-learning parametrizations of convection. *Journal of the Atmospheric Sciences*, *77*(12), 4357–4375. https://doi.org/10.1175/jas-d-20-0082.1

Brenowitz, N. D., & Bretherton, C. S. (2018). Prognostic validation of a neural network unified physics parameterization. *Geophysical Research Letters*, *45*(12), 6289–6298. https://doi.org/10.1029/2018gl078510

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. https://doi.org/10.1029/2019ms001711

Bretherton, C. S., Henn, B., Kwa, A., Brenowitz, N. D., Watt-Meyer, O., McGibbon, J., et al. (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. https://doi.org/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. https://doi.org/10.1029/2021MS002477

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

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*. https://doi.org/10.48550/arXiv.1406.1078

Choi, H.-J., Han, J.-Y., Koo, M.-S., Chun, H.-Y., Kim, Y.-H., & Hong, S.-Y. (2018). Effects of non-orographic gravity wave drag on seasonal and medium-range predictions in a global forecast model. *Asia-Pacific Journal of Atmospheric Sciences*, *54*(S1), 385–402. https://doi.org/10.1007/s13143-018-0023-1

Cotronei, A., & Slawig, T. (2020). Single-precision arithmetic in ECHAM radiation reduces runtime and energy consumption. *Geoscientific Model Development*, *13*(6), 2783–2804. https://doi.org/10.5194/gmd-13-2783-2020

Espinosa, Z. I., Sheshadri, A., Cain, G. R., Gerber, E. P., & DallaSanta, K. J. (2022). Machine learning gravity wave parameterization generalizes to capture the QBO and response to increased CO_2. *Geophysical Research Letters*, *49*(8), e2022GL098174. https://doi.org/10.1029/2022gl098174

Gehring, J., Auli, M., Grangier, D., Yarats, D., & Dauphin, Y. N. (2017). Convolutional sequence to sequence learning. In *International conference on machine learning* (pp. 1243–1252). PMLR. https://arxiv.org/abs/1705.03122

Gentine, P., Pritchard, M., Rasp, S., Reinaudi, G., & Yacalis, G. (2018). Could machine learning break the convection parameterization deadlock? *Geophysical Research Letters*, *45*(11), 5742–5751. https://doi.org/10.1029/2018gl078202

Han, Y., Zhang, G. J., Huang, X., & Wang, Y. (2020). A moist physics parameterization based on deep learning. *Journal of Advances in Modeling Earth Systems*, *12*(9), e2020MS002076. https://doi.org/10.1029/2020ms002076

Han, Y., Zhang, G. J., & Wang, Y. (2023). An ensemble of neural networks for moist physics processes, its generalizability and stable integration. *Journal of Advances in Modeling Earth Systems*, *15*(10), e2022MS003508. https://doi.org/10.1029/2022MS003508

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1007/978-3-642-24797-2_4

Hogan, R. J., & Bozzo, A. (2018). A flexible and efficient radiation scheme for the ECMWF model. *Journal of Advances in Modeling Earth Systems*, *10*(8), 1990–2008. https://doi.org/10.1029/2018MS001364

Hogan, R. J., & Matricardi, M. (2022). A tool for generating fast k-distribution gas-optics models for weather and climate applications. *Journal of Advances in Modeling Earth Systems*, *14*(10), e2022MS003033. https://doi.org/10.1029/2022MS003033

Kang, M.-J., Chun, H.-Y., & Garcia, R. R. (2020). Role of equatorial waves and convective gravity waves in the 2015/16 quasi-biennial oscillation disruption. *Atmospheric Chemistry and Physics*, *20*(23), 14669–14693. https://doi.org/10.5194/acp-20-14669-2020

Katsch, T. (2023). Gateloop: Fully data-controlled linear recurrence for sequence modeling. *arXiv preprint arXiv:2311.01927*. https://doi.org/10.48550/arXiv.2311.01927

Kochkov, D., Yuval, J., Langmore, I., Norgaard, P., Smith, J., Mooers, G., et al. (2024). Neural general circulation models for weather and climate. *Nature*, *632*(8027), 1060–1066. https://doi.org/10.1038/s41586-024-07744-y

Krasnopolsky, V., Belochitski, A. A., Hou, Y., Lord, S. J., & Yang, F. (2012). Accurate and fast neural network emulations of long and short wave radiation for the NCEP global forecast system model. https://repository.library.noaa.gov/view/noaa/6951

Krasnopolsky, V., Fox-Rabinovitz, M., Hou, Y., Lord, S., & Belochitski, A. (2010). 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. https://doi.org/10.1175/2009mwr3149.1

Krasnopolsky, V. M., Fox-Rabinovitz, M. S., & Belochitski, A. A. (2008). Decadal climate simulations using accurate and fast neural network emulation of full longwave and shortwave radiation. *Monthly Weather Review*, *136*(10), 3683–3695. https://doi.org/10.1175/2008mwr2385.1

Lagerquist, R., Turner, D., Ebert-Uphoff, I., Stewart, J., & Hagerty, V. (2021). Using deep learning to emulate and accelerate a radiative transfer model. *Journal of Atmospheric and Oceanic Technology*, *38*(10), 1673–1696. https://doi.org/10.1175/JTECH-D-21-0007.1

Lagerquist, R., Turner, D. D., Ebert-Uphoff, I., & Stewart, J. Q. (2023). Estimating full longwave and shortwave radiative transfer with neural networks of varying complexity. *Journal of Atmospheric and Oceanic Technology*, *40*(11), 1407–1432. https://doi.org/10.1175/jtech-d-23-0012.1

Lam, R., Sanchez-Gonzalez, A., Willson, M., Wirnsberger, P., Fortunato, M., Alet, F., et al. (2022). *Graphcast: Learning skillful medium-range global weather forecasting*. arXiv preprint arXiv:2212.12794. https://doi.org/10.48550/arXiv.2212.12794

Lei, T., Zhang, Y., Wang, S., Dai, H., & Artzi, Y. (2017). Simple recurrent units for highly parallelizable recurrence. *arXiv preprint arXiv: 1709.02755*. arxiv 2017. https://doi.org/10.48550/arXiv.1709.02755

Mehta, H., Gupta, A., Cutkosky, A., & Neyshabur, B. (2022). Long range language modeling via gated state spaces. *arXiv preprint arXiv: 2206.13947*.

Mooers, G., Pritchard, M., Beucler, T., Ott, J., Yacalis, G., Baldi, P., & Gentine, P. (2021). Assessing the potential of deep learning for emulating cloud superparameterization in climate models with real-geography boundary conditions. *Journal of Advances in Modeling Earth Systems*, *13*(5), e2020MS002385. https://doi.org/10.1029/2020ms002385

Müller, T., Rousselle, F., Novák, J., & Keller, A. (2021). Real-time neural radiance caching for path tracing. *arXiv preprint arXiv:2106.12372*. https://doi.org/10.48550/arXiv.2106.12372

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. https://doi.org/10.1029/2018ms001351

Pahlavan, H. A., Hassanzadeh, P., & Alexander, M. J. (2023). Explainable offline-online training of neural networks for parameterizations: A 1D gravity wave-QBO testbed in the small-data regime. *arXiv preprint arXiv:2309.09024*, *51*(2), e2023GL106324. https://doi.org/10.1029/2023GL106324

Pal, A., Mahajan, S., & Norman, M. R. (2019). Using deep neural networks as cost-effective surrogate models for super-parameterized e3sm radiative transfer. *Geophysical Research Letters*, *46*(11), 6069–6079. https://doi.org/10.1029/2018gl081646

Perkins, W. A., Brenowitz, N. D., Bretherton, C. S., & Nugent, J. M. (2024). Emulation of cloud microphysics in a climate model. *Journal of Advances in Modeling Earth Systems*, *16*(4), e2023MS003851. https://doi.org/10.1029/2023ms003851

Rasp, S., Pritchard, M. S., & Gentine, P. (2018). Deep learning to represent subgrid processes in climate models. *Proceedings of the National Academy of Sciences*, Vol. 115(39), 9684–9689. https://doi.org/10.1073/pnas.1810286115

Roh, S., & Song, H.-J. (2020). Evaluation of neural network emulations for radiation parameterization in cloud resolving model. *Geophysical Research Letters*, *47*(21), e2020GL089444. https://doi.org/10.1029/2020gl089444

Schäfer, S. A., Hogan, R. J., Klinger, C., Chiu, J. C., & Mayer, B. (2016). Representing 3-D cloud radiation effects in two-stream schemes: 1. Longwave considerations and effective cloud edge length. *Journal of Geophysical Research: Atmospheres*, *121*(14), 8567–8582. https://doi.org/10.1002/2016jd024876

Schneider, T., Teixeira, J., Bretherton, C. S., Brient, F., Pressel, K. G., Schär, C., & Siebesma, A. P. (2017). Climate goals and computing the future of clouds. *Nature Climate Change*, *7*(1), 3–5. https://doi.org/10.1038/nclimate3190

Shamekh, S., Lamb, K. D., Huang, Y., & Gentine, P. (2023). Implicit learning of convective organization explains precipitation stochasticity. *Proceedings of the National Academy of Sciences of the United States of America*, *120*(20). https://doi.org/10.1073/pnas.2216158120

Shi, X., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W.-K., & Woo, W.-c. (2015). Convolutional LSTM network: A machine learning approach for precipitation nowcasting. *Advances in Neural Information Processing Systems*, *28*. https://doi.org/10.5555/2969239.2969329

Shin, J., & Baik, J.-J. (2022). Parameterization of stochastically entraining convection using machine learning technique. *Journal of Advances in Modeling Earth Systems*, *14*(5), e2021MS002817. https://doi.org/10.1029/2021MS002817

Shonk, J. K., & Hogan, R. J. (2008). Tripleclouds: An efficient method for representing horizontal cloud inhomogeneity in 1D radiation schemes by using three regions at each height. *Journal of Climate*, *21*(11), 2352–2370. https://doi.org/10.1175/2007jcli1940.1

Smith, J. T., Warrington, A., & Linderman, S. W. (2022). Simplified state space layers for sequence modeling. *arXiv preprint arXiv:2208.04933*. https://doi.org/10.48550/arXiv.2208.04933

Song, H.-J., & Roh, S. (2021). Improved weather forecasting using neural network emulation for radiation parameterization. *Journal of Advances in Modeling Earth Systems*, *13*(10), e2021MS002609. https://doi.org/10.1029/2021MS002609

Song, H.-J., Roh, S., Lee, J., Nam, G., Yun, E., Yoon, J., & Kim, P. S. (2022). Benefits of stochastic weight averaging in developing neural network radiation scheme for numerical weather prediction. *Journal of Advances in Modeling Earth Systems*, *14*(10), e2021MS002921. https://doi.org/10.1029/2021MS002921

Sun, Y. Q., Hassanzadeh, P., Alexander, M. J., & Kruse, C. G. (2023). Quantifying 3D gravity wave drag in a library of tropical convection-permitting simulations for data-driven parameterizations. *Journal of Advances in Modeling Earth Systems*, *15*(5), e2022MS003585. https://doi.org/10.1029/2022ms003585

Tahseen, T. P., Mendonça, J. M., Yip, K. H., & Waldmann, I. P. (2024). Enhancing 3D planetary atmosphere simulations with a surrogate radiative transfer model. *Monthly Notices of the Royal Astronomical Society*, *535*(3), 2210–2227. https://doi.org/10.1093/mnras/stae2461

Ukkonen, P. (2022). Exploring pathways to more accurate machine learning emulation of atmospheric radiative transfer. *Journal of Advances in Modeling Earth Systems*, *14*(4), e2021MS002875. https://doi.org/10.1029/2021MS002875

Ukkonen, P. (2024). Code and partial data for "Representing sub-grid processes in weather and climate models via sequence learning". *Zenodo*. https://doi.org/10.5281/zenodo.10462470

Ukkonen, P., & Hogan, R. J. (2023). Implementation of a machine-learned gas optics parameterization in the ECMWF integrated forecasting system: RRTMGP-NN 2.0. *Geoscientific Model Development*, *16*(11), 3241–3261. https://doi.org/10.5194/gmd-16-3241-2023

Ukkonen, P., & Hogan, R. J. (2024). Twelve times faster yet accurate: A new state-of-the-art in radiation schemes via performance and spectral optimization. *Journal of Advances in Modeling Earth Systems*, *16*(1), e2023MS003932. https://doi.org/10.1029/2023ms003932

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., et al. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, *30*.

Wang, P., Yuval, J., & O'Gorman, P. A. (2022). Non-local parameterization of atmospheric subgrid processes with neural networks. *Journal of Advances in Modeling Earth Systems*, *14*(n/a), e2022MS002984. n/a. https://doi.org/10.1029/2022MS002984

Wang, X., Han, Y., Xue, W., Yang, G., & Zhang, G. J. (2022). Stable climate simulations using a realistic general circulation model with neural network parameterizations for atmospheric moist physics and radiation processes. *Geoscientific Model Development*, *15*(9), 3923–3940. https://doi.org/10.5194/gmd-15-3923-2022

Watt-Meyer, O., Brenowitz, N. D., Clark, S. K., Henn, B., Kwa, A., McGibbon, J., et al. (2024). Neural network parameterization of subgrid-scale physics from a realistic geography global storm-resolving simulation. *Journal of Advances in Modeling Earth Systems*, *16*(2), e2023MS003668. https://doi.org/10.1029/2023ms003668

Yao, Y., Zhong, X., Zheng, Y., & Wang, Z. (2023). A physics-incorporated deep learning framework for parameterization of atmospheric radiative transfer. *Journal of Advances in Modeling Earth Systems*, *15*(5), e2022MS003445. https://doi.org/10.1029/2022MS003445

Yuval, J., & O'Gorman, P. A. (2020). Stable machine-learning parameterization of subgrid processes for climate modeling at a range of resolutions. *Nature Communications*, *11*(1), 1–10.

Yuval, J., O'Gorman, P. A., & Hill, C. N. (2021). Use of neural networks for stable, accurate and physically consistent parameterization of subgrid atmospheric processes with good performance at reduced precision. *Geophysical Research Letters*, *48*(6), e2020GL091363. https://doi.org/10.1029/2020gl091363

Zadra, A., Williams, K., Frassoni, A., Rixen, M., Adames, Á. F., Berner, J., et al. (2018). Systematic errors in weather and climate models: Nature, origins, and ways forward. *Bulletin of the American Meteorological Society*, *99*(4), ES67–ES70. https://doi.org/10.1175/bams-d-17-0287.1

Zelinka, M. D., Myers, T. A., McCoy, D. T., Po-Chedley, S., Caldwell, P. M., Ceppi, P., et al. (2020). Causes of higher climate sensitivity in CMIP6 models. *Geophysical Research Letters*, *47*(1), e2019GL085782. https://doi.org/10.1029/2019gl085782