ClimART: A Benchmark Dataset for Emulating Atmospheric Radiative Transfer in Weather and Climate Models

Salva Rühling Cachay[*]
TU Darmstadt & Mila
salvaruehling@gmail.com

Venkatesh Ramesh[*]
Université de Montréal & Mila
venkatesh.ramesh@umontreal.ca

Jason N. S. Cole **Howard Barker**
Environment and Climate Change Canada
{jason.cole,howard.barker}@canada.ca

David Rolnick
McGill University & Mila
drolnick@cs.mcgill.ca

Abstract

Numerical simulations of Earth's weather and climate require substantial amounts of computation. This has led to a growing interest in replacing subroutines that explicitly compute physical processes with approximate machine learning (ML) methods that are fast at inference time. Within weather and climate models, atmospheric radiative transfer (RT) calculations are especially expensive. This has made them a popular target for neural network-based emulators. However, prior work is hard to compare due to the lack of a comprehensive dataset and standardized best practices for ML benchmarking. To fill this gap, we build a large dataset, ClimART, with more than *10 million samples from present, pre-industrial, and future climate conditions*, based on the Canadian Earth System Model. ClimART poses several methodological challenges for the ML community, such as multiple out-of-distribution test sets, underlying domain physics, and a trade-off between accuracy and inference speed. We also present several novel baselines that indicate shortcomings of datasets and network architectures used in prior work.[2]

1 Introduction

Numerical weather prediction (NWP) models have become essential tools for numerous sectors of society. Their close relatives, global and regional climate models (GRCM) provide crucial information to policymakers and the public about Earth's changing climate and its various impacts on the biosphere. These models attempt to simulate many complicated physical processes that interact over wide ranges of space and time and seamlessly link Earth's atmosphere, ocean, land, and ice. However, due to the complexity and number of physical processes that have to be addressed, various simplifications must in practice be made, involving mathematical and numerical approximations that often have substantial statistical bias and computational cost.

One of these approximations is the *sub-grid scale parametrization* that is routinely used to approximate atmospheric radiative transfer (RT). The RT routine has traditionally represented the largest computational bottleneck in most weather and climate simulation. To speed up the computation, this routine is run only every few iterations and the results for the intermediate iterations have to be interpolated. Needless to say, this approximation of intermediate steps introduces errors in climate

[*]Equal contribution

[2]Download instructions, baselines, and code are available at: https://github.com/RolnickLab/climart

35th Conference on Neural Information Processing Systems (NeurIPS 2021) Track on Datasets and Benchmarks.

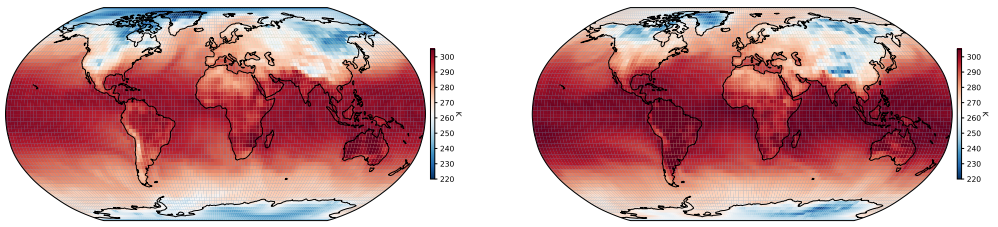
and weather predictions. The computational cost also means that climate models must be run at extremely coarse spatial resolutions that leave most processes unresolved and reduce the utility of information in predicting and responding to the effects of climate change.

Thanks to their better inference speed, neural network-based *surrogates* are a promising alternative to computationally slow physics parametrizations. Such hybrid modelling approaches, however, present several challenges, including accurate emulation of complex physical processes, as well as the (out-of-distribution) *generalization power* of ML models to handle environmental conditions not present in their training datasets (e.g., weather states that are not realized under current conditions).

To date, different datasets, setups, and evaluation procedures have made results in this space hard to compare. This can be attributed to the fact that no comprehensive public dataset exists, and the creation of it requires access to, and knowledge of, the relevant climate model. To address these issues and catalyze further work, we introduce a new and comprehensive dataset for the RT problem and open-source it under the Creative Commons license.

Our key contributions include:

- **ClimART: Climate Atmospheric Radiative Transfer**, is the *most comprehensive* publicly available dataset for ML emulation of weather and climate model parameterizations. It comes with *more than 10 million samples*, including three subsets of data for evaluating *out-of-distribution (OOD) generalization*.
- **Applying New Models to ClimART**. We propose multiple new models not studied in the related work, which thanks to the comprehensiveness of ClimART, allow us to *analyze the limitations of previously used models and datasets*.
- **Towards Advancing the State-of-the-Art**. ClimART's scale, unique properties, and ease of access, together with the accompanying code interface and baselines, will lower barriers for the ML community to tackle impactful challenges in climate science. ClimART also presents opportunities for spurring methodological innovation in ML via multiple *out-of-distribution test sets*, the scope for building *physics-informed ML models*, and the *accuracy versus inference speed trade-off* inherent in the problem setting.



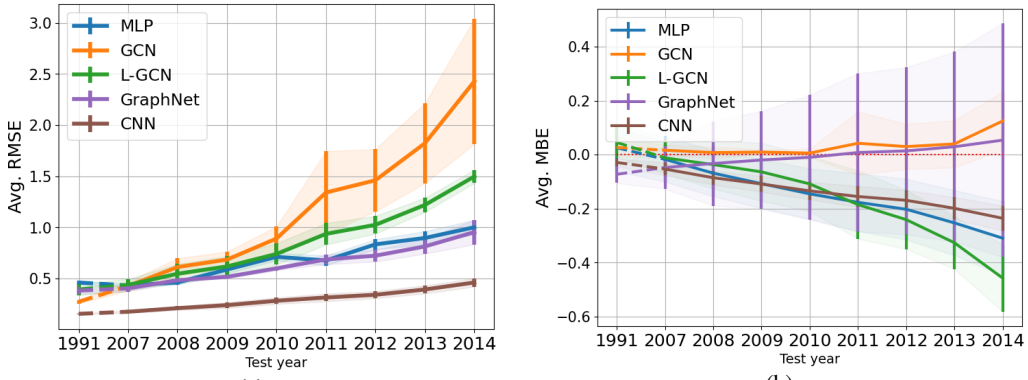
(a) Surface Temperature on Jan. 1, 1979 (b) Projected Surface Temperature on Jan. 1, 2070

Figure 1: Surface temperatures in atmospheric snapshots from 1979 and 2070.

2 Related Work

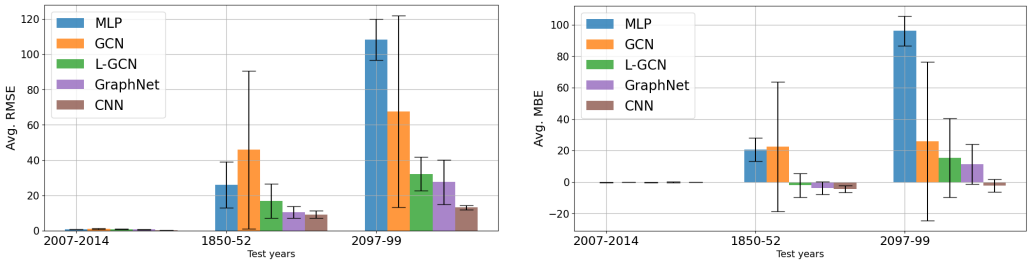
There have been various recent efforts to emulate sub-grid scale parameterizations by neural networks, which, thanks to their computational efficiencies, are expected to significantly speed up large-scale model simulations [6, 25, 30].

The computational burden of the RT physics motivated early pioneering work to seek out its emulation with shallow multi-layer perceptron (MLP) networks [8, 9, 15, 16], including decadal climate model simulations [17]. More recent work still focuses on using MLPs to emulate (a part of) the RT physics [19, 20, 22, 27, 28]. 2D CNNs have been also used in [19], which however treat the different input variables within the second spatial dimension instead of in the channel dimension. Prior work on such ML emulators, however, employed datasets that simplify Earth (e.g., with Aqua-planet conditions [6, 25, 30]), use a limited subset of climate model variables as predictors [15, 19, 25, 28], use manually perturbed test sets [19, 28], and generally fail to accurately probe the generalization power of ML models [19, 20, 22, 28]. The latter is particularly important, randomly-split test sets [20, 22, 28], and/or test data coming from at most two different years [19, 20, 22] can overestimate the actual skill



(a) (b)

Figure 2: Present-day test performance



(a) Test RMSE (b) Test MBE

Figure 3: Performance as a function of the main test set year (Fig. 2) and OOD test set (Fig. 3) for our different baseline models. Metrics are in W/m^2 and shown as the average over the vertical and over the up- and down-welling flux errors. The leftmost x-tick for the OOD plots corresponds to the average metric of the main test years 2007-2014, for which the metrics are shown on a yearly basis in Fig. 2. Generalization to pre-industrial and future pristine-sky conditions is a particularly challenging task because of the changes in gas concentrations More structured models like a CNN or GraphNet perform significantly better than an MLP.

of the ML emulators on real world unseen data. This can be fatal when the ML emulator is to be used in long-term simulations (e.g., future climate projections).

In physics-based RT models as used in large-scale environmental models, computation of radiative flux profiles is a two-step process. The first step involves the calculation of optical properties for gases, aerosols, and clouds. The second, more computationally intensive, and arguably most erroneous step is the application of a solution of the RT equation, using the optical properties from the first step, leading to vertical profiles of radiative fluxes. Work by [27, 28] focused on a hybrid approach in which gaseous optical properties are predicted by an ML model and then passed off to a physics-based RT model for calculation of fluxes. However, [28] generates training data by just using perturbations on input variables of RFMIP[24]. [27] make a more comprehensive effort via generation of training data by combining multiple sources such as RFMIP[24], CKDMIP[12] and CAMS[1], and perturbing them to generate sufficient training data. Such perturbations, however, might lead to unrealistic input values.

The present work is closer to [19] that used a small fraction of ERA-Interim data from 1979-85 and 2015-16, Their concern was however, limited, to longwave radiative flux profiles for simplified clear-sky atmospheric conditions (without greenhouse gases like methane). Similarly to [28], they employ a test set that includes manually perturbed atmospheric states. We believe that our dataset that includes data of pre-industrial and future climate drawn from an actual climate model, is more realistic and the better choice.

3 Background

In the following, we introduce background and terminology that is helpful in order to better understand the problem setup and the dataset we present.

Earth System Models Earth System Models (ESMs) simulate Earth's climate, including interactions between the atmosphere, ocean, sea-ice and carbon cycle. Within the atmosphere, the ESM computes the current state of the modelled variables which includes temperature, water vapor, aerosols, and clouds. This is done using models that discretize the Earth's atmosphere into a 3D spatial grid along latitude, longitude, and vertical dimensions, as well as a discretized timestep. At each discretized point in the horizontal, one can envision processes occurring in the vertical within a *column* that provides a *profile* of information about the state of the atmosphere, e.g., temperature. These profiles can be provided with respect to n *layers* in the atmosphere, in addition to $n+1$ *levels* (interfaces between neighboring layers), where the bottom-most level corresponds to the surface and the top-most level corresponds to the top-of-atmosphere (TOA).

Atmospheric radiative transfer Radiative transfer describes the propagation of radiation. In the atmosphere, radiation transmits energy between atmospheric layers. It is classified into *shortwave* (solar), and *longwave* (thermal) radiation, which are emitted by the sun and Earth, respectively. In general, shortwave radiation is absorbed by the Earth and then re-emitted as longwave radiation; some of this longwave radiation is then re-absorbed by the atmosphere. At any given time and level of the atmosphere, *up- and down-welling fluxes* refer to the amount of radiation that is traveling up and down between the adjacent atmospheric layers. As a result of emitting and absorbing radiation, the layers of the atmosphere change temperature, which is captured in *heating rate* profiles (also called cooling rates when negative). The heating rate of any given layer can be directly computed based on the up- and down-welling fluxes of the two adjacent levels (see Appendix B.4). Ultimately, the difference in radiative flux into and out of the atmosphere is responsible for changes in the Earth's overall temperature, and the dependence of radiative fluxes on gas concentrations is the source of climate change, since human activity has increased the concentration of various greenhouse gases such as carbon dioxide, methane, and nitrous oxide.

To predict radiative flux and heating rate profiles in the atmosphere, climate scientists work with three types of models for the sky, which differ in the kinds of information they factor in: Pristine, clear, and clouds. *Pristine-sky* is the simplest, meaning that only the concentrations of gases are factored in. *Clear-sky* also includes the concentrations of aerosols, which are particles present in the air such as sulfur-containing compounds. The most general case also includes clouds.

CanESM and its radiative transfer parameterization The Canadian Earth System Model (CanESM) is a comprehensive global model used to simulate Earth's past climate and the present results of climate change, as well as to make future climate projections. Figure 1 shows an example of CanESM's simulation of current (1979) and future (2070) surface temperatures. Its most recent version, CanESM5 [26], simulates the atmosphere, ocean, sea-ice, land and carbon cycle, including the coupling between each of these components. For the atmosphere it uses parameterizations to represent unresolved sub-grid scale processes like radiation, convection, aerosols, and clouds.
The radiative transfer parameterization in CanESM5, is representative of the approach used in most modern ESMs. The optical properties of a number of components are accounted for, including the surfaces, aerosols, clouds and gases (represented using a correlated k-distribution model). The parameterization follows an independent column approximation. This intuitively means that for a given latitude and longitude, the RT physics model takes as input only information from the 1D vertical profile of the atmospheric state at the corresponding geographical location. More details on CanESM and its RT parametrization can be found in Appendix A.

4 ClimART Dataset

4.1 Dataset collection

For our main dataset, global snapshots of the current atmospheric state were sampled from CanESM5 simulations every 205 hours from 1979 to 2014.[3] CanESM5's horizontal grid discretizes longitude into 128 columns with equal size and latitude into 64 columns using a Gaussian grid ($8192 = 128 \times 64$ columns in total). This results in 43 global snapshots per year for a total of more than 12 million columns for the period 1979-2014 and a raw dataset size of 1.5TB. Each column of atmospheric-

[3]The choice of 205 hours provides a manageable amount of equally spaced data while also ensuring that every hour of the day is covered since 205 is relatively prime to 24.

surface properties was then passed through CanESM5's RT physics model in order to collect the corresponding RT output: Shortwave and longwave (up- and down-welling) flux and heating rate profiles for pristine- and clear-sky conditions. The resulting NetCDF4 datasets were then processed to NumPy arrays stored in Hdf5 format (one per year), with three distinct input arrays as described in the following subsection, and one output array per potential target variable. We proceeded analogously for the pre-industrial and future climate years, 1850-52 and 2097-99 respectively (see section 4.3.1). More details are available in the Appendix B.1.

4.2 Dataset interface

Inputs We saved the exact same inputs used by the CanESM5 radiation code and augmented them by auxiliary variables such as geographical information (see Appendix B.3 for full details and description). Each input corresponds to a column of CanESM5. Its variables can be divided into three distinct types: i) layer variables, ii) level variables, iii) variables not tied to the height/vertical discretization. Examples for the two first 1D variables are pressure (occurring at both, levels and layers) and water vapour (only present at the layers). The third type of variables are comprised of optical properties of the surface, boundary conditions, and geographical information related data. We refer to this set as the *global* variables. ***The data thus has a unique structure with heterogenous data types, where 1D vertical data is complemented by non-spatial information.*** We also note that the RT problem is non-local, since the spectral composition, and thus the heating rate, at one level can depend much on attenuation, or production, of radiation at a far-removed layer (e.g., reduced absorption of solar radiation by water vapour near the surface due to the presence of reflective high-altitude cirrus clouds).

Outputs We provide the full radiation output of CanESM5's as a potential target for pristine- and clear-sky conditions. That is, ClimART ***comes with two levels of complexity***: pristine-sky (no aerosols and no clouds) being the simpler one compared to clear-sky, which also reflects the impacts on RT due to aerosols. It consists, for both shortwave and longwave radiation, of the up-and down-welling flux profiles and corresponding heating rate profiles (i.e. $6 = 2 \times 3$ distinct variables for each sky condition).

In our experiments we focus on pristine shortwave radiation. Our dataset, however, allows the user to choose the desired target variables based on their needs.

4.3 Dataset split

In the following we describe the data split that we recommend to follow for benchmarking purposes.

Training and Validation sets ClimART provides the complete data extracted from CanESM5, as described above, from 1979 to 2006, excluding 1991-93, as suggested data for training and validating ML models. In our experiments we used 1990, 1999, and 2003 for training, while keeping 2005 for validation.

Main Test set We suggest to use the data from the years 2007 to 2014 as main test set. This relatively long interval allows to test the ML model on a very diverse set of present-day conditions. To make evaluations feasible regardless of the available compute resources, we chose to subsample 15 random snapshots for each year. This results in almost 1 million testing samples.

4.3.1 Out-of-distribution (OOD) test sets

In order to evaluate how well a ML model generalizes beyond the present-day conditions found in the main dataset, we provide *three distinct OOD test sets* that cover an anomaly in the atmospheric state, as well as two temporal distributional shifts.

Mount Pinatubo eruption This test set includes conditions from the year 1991, when the Mount Pinatubo volcano erupted, and probes how well the ML model can cope with sudden atmospheric changes. The challenges arise via a sudden increase in atmospheric opacity due to high-altitude volcanic aerosol (see Appendix B.2 for more details). While CanESM5's solar RT model deals with these aerosols well, it is the specification of changes in aerosol mass and distribution (i.e., inputs to the RT model) that pose the largest challenges. Since remnants of the emitted stratospheric aerosols

remained in the atmosphere for years after the eruption, we chose to exclude the two subsequent years, 1992 and 1993, from ClimART in order to avoid data leakage during training.

Pre-industrial and future This test set probes how well the ML emulator generalizes under challenging distributional shifts. For this purpose, ClimART provides historic data from the years 1850-52 and future data from the years 2097-99. In both cases, the primary challenge for ML models is that they can be expected to encounter surface-atmosphere conditions that are not present in the training dataset. The primary differences between current and *pre-industrial* conditions involve: reduced atmospheric trace (greenhouse) gas concentrations for pre-industrial times; changes in aerosol emission; and some land surface properties that arise through changes in land usage. The *future climate* data can test how well ML models extrapolate to conditions that differ from the current climatic state as a result of radiative forcing through increases in greenhouse gas concentrations. Future climatic conditions were simulated by CanESM5 based on increases in atmospheric greenhouse gas concentrations and changes in aerosol emissions that follow well-defined scenarios (see [26]) laid out for the Sixth Coupled Model Intercomparison Project (CMIP6; cf. [10]).

4.4 Usage

It is important to note that a ML model trained on our dataset be both *verified* and *validated* before it can be employed "operational" in a global climate or NWP model. The *verification* phase is characterized by "simple" quantitative assessment of a ML model's bias and random (conditional) errors. Once one feels that the ML model is ready to go into the dynamical model, its computation-saving aspect can be assessed against any ramifications it has on the overall forecast of the model. Ideally, a successful ML model (i.e., one that is *validated*) will simultaneously reduce computation time and have incur only statistically insignificant impacts on the overall forecast.

Furthermore, we note that the level of "success" of an ML model at estimation of radiative flux profiles can differ for weather and climate applications, for it is expected that these areas of application will tolerate bias and random errors differently. For instance, an ML model with minor, but non-negligible, bias error might be acceptable for short-range weather predictions, but could have untenable affects on longer-term climate projections. Likewise, an ML model's random errors might tend to wash-out in long, low-resolution climate simulations, but might initiate spurious extreme events in high-resolution weather forecasts.

4.5 Limitations

Firstly, while advances on this problem would directly benefit the whole community, the inconsistent interfaces between different climate models and their parameterizations would likely require re-training the models for those specific input-output interfaces. We note that one motivation for proposing fully convolutional and graph-based networks in our experiments, is their applicability regardless of the vertical discretization of the columns (depending on the climate model, a column might be divided into different layers). The shortcoming of MLPs, which do not enjoy this property, was also identified by [28]. Applying fully convolutional and graph-based networks to emulate parameterizations with different vertical discretization than the one trained on is an interesting direction for future work and could present a way to have ML emulators that are more generally applicable.

Secondly, our targets do not include radiation output under all-sky conditions (which, besides aerosols, includes clouds). We believe however that our otherwise comprehensive dataset will serve well as a test-bed for ML emulators under the more simple (yet complex) pristine- and clear-sky conditions. Moreover, we note that pristine- and clear-sky are routinely used in diagnostic analyses of climate and weather model results. That is, while such conditions do not often occur in the atmosphere (mostly above the troposphere), they are nevertheless computed for the entire globe in order to assess a model's cloud dynamics and compare to satellite data.

	Vertical Avg.		Surface		TOA	
Model	RMSE	MBE	RMSE	MBE	RMSE	MBE
MLP	0.701 ± 0.04	-0.160 ± 0.10	0.684 ± 0.07	-0.282 ± 0.09	0.573 ± 0.06	-0.208 ± 0.10
GCN	1.209 ± 0.25	0.034 ± 0.04	1.260 ± 0.70	0.244 ± 0.47	0.815 ± 0.12	0.071 ± 0.15
L-GCN	0.878 ± 0.09	-0.179 ± 0.10	0.714 ± 0.21	-0.241 ± 0.29	0.440 ± 0.09	-0.048 ± 0.21
GraphNet	0.648 ± 0.04	$\mathbf{-0.001 \pm 0.25}$	0.620 ± 0.08	$\mathbf{0.017 \pm 0.29}$	0.434 ± 0.08	$\mathbf{-0.033 \pm 0.21}$
CNN	$\mathbf{0.303 \pm 0.03}$	-0.142 ± 0.03	$\mathbf{0.265 \pm 0.03}$	-0.138 ± 0.03	$\mathbf{0.284 \pm 0.05}$	-0.177 ± 0.04

Table 1: We run several neural network architectures on ClimART to emulate the shortwave down- and up-welling radiative fluxes. The reported metrics (in W/m^2) are averaged out over all test samples (years 2007-2014), three random seeds, as well as over the two errors for up- and down-welling fluxes. Vertical Avg. also averages the metrics over all levels (heights), see 5.1 for more.

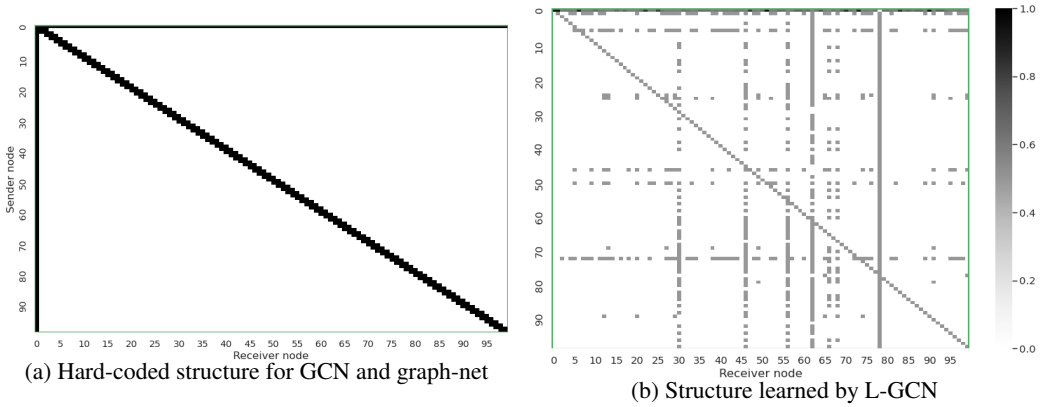
5 Experiments

5.1 Benchmarking neural network architectures

Note that prior work usually restricted the ML model to be a multi-layer perceptron (MLP). In light of the structured data in ClimART, we aim to 1) propose more structured neural network architectures that we believe are more suitable to the task and on which we hope follow-up work can build upon; 2) study how these more structured neural network architectures compare to the unstructured MLP. Thus, we benchmark an MLP against a 1-D convolutional neural network (CNN) [18], a graph convolutional network (GCN) [14], and a graph network [4]. We now give a high-level overview over each of the architectures, which all are relatively lightweight as it is important to keep the inherent *inference speed versus accuracy trade-off* in mind. More details on it and used hyperparameters can be found in Appendix C.

- **MLP:** The MLP used for our experiments is a simple three layer MLP with the following hidden-layer dimensions: $\langle 512, 256, 256 \rangle$. As an MLP takes unstructured 1D data as input, all the input variables need to be flattened into a single vector for the MLP.
- **GCNs**, take graph-structured data as input. To map the columns to a graph, we use a straighforward line-graph structure where each node is a level or layer and is connected to the two layers or levels spatially adjacent to it above and below. To take into account the *global* information, we add it as an additional node to the graph with connections to all other nodes. The resulting graph structure, in form of an adjacency matrix, is shown in Fig.4a, where the global node has index 0, and the other nodes are spatially indexed for plotting purposes, where 1 corresponds to the TOA level and the last node corresponds to the surface. A more sophisticated graph structure is studied in section 5.2 (**L-GCN**).
  The used GCN has three layers of dimension 128.
- **Graph networks**, take graph-structured data (with node and edge features), complemented by a so-called global feature vector, as input. Thus, it is the most natural model for our task, since we can map the levels to be the nodes, layers to be the edges connecting the adjacent levels (i.e. a line-graph), and the non-spatial variables to the global feature vector. Essentially, a graph network [4] consists of multiple MLP modules, for which we use 1-layer MLPs with a hidden dimension of 128. We use a three-layered graph network.
- **1D CNN:** For the CNN model, we use a 3-layer network with kernel sizes $\langle 20, 10, 5 \rangle$ and the corresponding strides set as $\langle 2, 2, 1 \rangle$. The channels parameter is given by $\langle 200, 400, 100 \rangle$, with the last channels setting it equal to the input size. We then apply a global average over the resulting tensor to get the output. To preprocess the data for CNN, we pad the surface and layers variable to match the dimensions of levels variable. Then the result is concatenated and fed to the model.

For all the models, we use a learning rate of 2e-4 with an exponential decay learning rate scheduler and Adam [13] as optimizer. All the models are trained for 100 epochs with the mean squared error loss. We report root mean squared error (RMSE) and mean bias error (MBE) statistics over three random seeds. The results for our baseline models on pristine-sky conditions are shown in Fig. 2



(a) Hard-coded structure for GCN and graph-net

(b) Structure learned by L-GCN

Figure 4: L-GCN, a GCN with learnable adjacency matrix, learns an edge structure very different to the diagonal structure of a line-graph (a) that is used is static adjacency matrix for the GCN and graph network baselines. This indicates that the problem benefits from *non-local information*. Notably, the many outgoing edges from *global* node (index 0, top row) indicates its importance. Node indices are sorted by height in descending order, i.e index 1 corresponds to the top-of-atmosphere node/level, and 100 to the surface level.

(more detailed in Fig. 5) and reported in Table 1. Notably all models can be seen to significantly deteriorate as a function of the test year (that becomes temporally farther away from the training years). We note that prior work could not observe this phenomenon since the test data did not cover as many years or was randomly split from the training set. We also find that the CNN architecture provides the most skillful emulation in terms of RMSE, while the GraphNet provides the least biased errors. This holds for the respective metrics computed at the TOA and surface level only, as well as when averaged out vertically over all levels. We note that the surface and TOA flux predictions are especially important, as they are directly used by the host climate or weather model. We also find that the L-GCN, which extends the GCN by a learnable edge structure module, is able to significantly outperform it, especially for TOA predictions, see next subsection.

5.2 Exploiting non-locality

Heating and cooling at one layer of the atmosphere depends on attenuation of radiation in all other layers. This non-locality complicates numerical simulation of the process greatly, and takes sizable amounts of computer resources to handle properly (see Appendix A for more details). Therefore, we expect ML models that can take non-locality into account to be a promising research direction. In the following we support this hypothesis with a GCN that *learns the edge structure* (i.e. the adjacency matrix of the underlying graph) as proposed in [7], denoted L-GCN. The model architecture and all other parameters are identical to the GCN. Making the connections between arbitrary layers and levels learnable relaxes the hard-coded inductive bias imposed by the highly local line-graph structure used in the standard GCN model. Indeed, not only does L-GCN outperform the GCN (Table 1), but our post-analysis also reveals that it 1) ***learns a graph structure very different to the line-graph*** used by the GCN and GraphNet (see Fig. 4), 2) gives high importance to the *global* node, which is expected given the importance of the boundary conditions and surface type of a column (see Appendix C.2 for an analysis based on the eigenvector centrality).

5.3 Speed

To assess the speed of our models at inference time, we speed-test the ML models for pristine-sky/clear-sky conditions on CPU and GPU, and speed-test the physics-based models for pristine-sky/clear-sky conditions for different numbers of CPUs. Note that we did not optimize the forward pass of the ML models for efficiency; thus, greater speed should be readily attainable.

For the evaluation of ML models, we make use of an instance with 4 CPUs (*2x AMD EPYC Zen 2 "Rome" 7742*), 12GB Memory and Nvidia v100 GPU. The results for different ML models for pristine-sky conditions is shown in 2 excluding the time for data loading. These results are averaged

Model	Hardware		Time (s)
	CPU	GPU	
Physics-RT	2	N/A	*3.3919*
	4	N/A	3.3666
	16	N/A	2.0988
	64	N/A	1.9817
MLP	4	×	**0.1643**
	4	✓	**0.0016**
CNN	4	×	3.1870
	4	✓	0.0218
GCN	4	×	4.6846
	4	✓	1.6818
GraphNet	4	×	28.1253
	4	✓	0.1659

Table 2: *Pristine-sky speed benchmark* of physics-based model and ML models on different hardware configurations. The physics-based model runs serially in offline mode and is not GPU compatible yet. The ML models are all evaluated with a batch size of 8192 in CPU only and with GPU. The fastest models with and without GPU are in **boldened** and the second fastest in blue.

over 10 forward passes for an entire snapshot (8192 columns) excluding the first two warm-up passes. As expected, the MLP is fastest for both clear-sky and and pristine-sky inputs. Especially on a GPU the MLP provides, together with the CNN and GraphNet. a considerable speed-up over the RT physics. This is promising since GPUs are starting to be natively supported within the compute environments in which NWPs and GRCMs run [5], including CanESM's. When evaluated with a batch-size of 8192, the model performs **3.5x** better than with a batch-size of 512.

The physics-based RT parameterization is not GPU compatible yet so we run it by increasing the number of CPUs in the instance and average them over three runs. The RT parameterization can be significantly sped-up by increasing the number of CPUs from 4 to 16. However, the going from 16 to 64 CPUs has diminishing returns. It should be noted that this physics-based model was run in *offline* mode, where the computation is done serially. When run together with its host weather or climate model, the predictions occur in parallel.

5.4 OOD generalization

For evaluating the generalization of our models in OOD data, we run it on historic data (1850-1852) and future data (2097-2099). These experiments are extremely challenging given the limited size of of training set use for baseline models. Apart from this, in historic (pre-industrial) and future conditions, the values of input variables, especially those relating to the concentrations of gases vary quite a lot. For a model to be able to perform well on this data, it has to have understood the role of gas concentrations in prediction of the flux properties. As seen from the results in Fig. 3, all models degrade significantly in performance, especially for future climate conditions. However, it is notable how the models that better account for the structure of atmospheric data perform considerably better compared to MLPs: While both, the MLP and GraphNet, perform comparably well for present-day conditions with an RMSE of less than 1 W/m^2 (Fig. 5a), the MLP's RMSE for future conditions is above 100 while the GraphNet's stays at around 30 (Fig. 6a). Similarly, the CNN degrades "only" from less than 0.5 RMSE on the main test set, to below 18 W/m^2 for future-day climate conditions-

6 Conclusion & Future Work

We introduce a novel dataset ClimART which aims to provide a comprehensive dataset for parameterization of radiative transfer using ML models. We conduct a series of experiments to demonstrate which models are able to perform well under the inherent structure of atmospheric data in Experiments. Future work for improving upon the current baselines could include:

- Improving the model's inference speed via methods like ***weight pruning, model compression, or weight quantization***.
- Using a ***physics-informed neural network*** or loss function to predict realistic values in-line with the equations governing radiative transfer.
- ***Multi-task learning*** can be explored to emulate both shortwave and longwave fluxes or heating-rates simultaneously.
- Using Transformer-based architectures for their ability to perform well with arbitrary sequence lengths and incorporation of an ***attention mechanism.***

On the dataset side, we plan to extend ClimART to include all-sky data that includes the complexity due to clouds. We hope that ClimART will advance both fundamental ML methodology and climate science, and catalyze greater involvement of ML researchers in problems relevant to climate change.

References

[1] Inness A., Ades M., Agustí-Panareda A., Barré J., Benedictow A., Blechschmidt A.-M., Dominguez J. J., Engelen R., Eskes H., Flemming J., Huijnen V., Jones L., Kipling Z., Massart S., Parrington M., Peuch V.-H., Razinger M., Remy S., Schulz M., and Suttie M. The CAMS reanalysis of atmospheric composition. *Atmospheric Chemistry and Physics*, 19, 2019.

[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[3] H. W. Barker, J. N. S. Cole, J.-J. Morcrette, R. Pincus, P. Räisänen, K. von Salzen, and P. A. Vaillancourt. The monte carlo independent column approximation: an assessment using several global atmospheric models. *Quarterly Journal of the Royal Meteorological Society*, 134(635): 1463–1478, 2008.

[4] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

[5] Peter Bauer, Peter D Dueben, Torsten Hoefler, Tiago Quintino, Thomas C Schulthess, and Nils P Wedi. The digital revolution of earth-system science. *Nature Computational Science*, 1 (2):104–113, 2021.

[6] N. D. Brenowitz and C. S. Bretherton. Prognostic validation of a neural network unified physics parameterization. *Geophysical Research Letters*, 45(12):6289–6298, 2018.

[7] Salva Rühling Cachay, Emma Erickson, Arthur Fender C Bucker, Ernest Pokropek, Willa Potosnak, Suyash Bire, Salomey Osei, and Björn Lütjens. The world as a graph: Improving El Niño forecasts with graph neural networks. *arXiv preprint arXiv:2104.05089*, 2021.

[8] Frédérique Cheruy, Frederic Chevallier, Jean-Jacques Morcrette, Noëlle A Scott, and Alain Chédin. Une méthode utilisant les techniques neuronales pour le calcul rapide de la distribution verticale du bilan radiatif thermique terrestre. *Comptes Rendus de l'Academie des Sciences Serie II*, 322:665–672, 1996.

[9] F Chevallier, F Chéruy, NA Scott, and A Chédin. A neural network approach for a fast and accurate computation of a longwave radiative budget. *Journal of applied meteorology*, 37(11): 1385–1397, 1998.

[10] V. Eyring, S. Bony, G. A. Meehl, C. A. Senior, B. Stevens, R. J. Stouffer, and K. E. Taylor. Overview of the coupled model intercomparison project phase 6 (CMIP6) experimental design and organization. *Geoscientific Model Development*, 9(5):1937–1958, 2016.

[11] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (GELUs). *arXiv preprint arXiv:1606.08415*, 2020.

[12] R. J. Hogan and M. Matricardi. Evaluating and improving the treatment of gases in radiation schemes: The correlated k-distribution model intercomparison project (CKDMIP). *Geoscientific Model Development Discussions*, 13, 2020.

[13] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2017.

[14] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

[15] V. M. Krasnopolsky, M. S. Fox-Rabinovitz, Y. T. Hou, S. J. Lord, and A. A. Belochitski. 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, 2010.

[16] Vladimir M Krasnopolsky, Michael S Fox-Rabinovitz, and Dmitry V Chalikov. 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, 2005.

[17] Vladimir M. Krasnopolsky, Michael S. Fox-Rabinovitz, and Alexei A. Belochitski. Decadal climate simulations using accurate and fast neural network emulation of full, longwave and shortwave, radiation. *Monthly Weather Review*, 136(10):3683 – 3695, 2008.

[18] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4): 541–551, 1989.

[19] Y. Liu, R. Caballero, and J. M. Monteiro. RadNet 1.0: exploring deep learning architectures for longwave radiative transfer. *Geoscientific Model Development*, 13(9):4399–4412, 2020.

[20] David Meyer, Robin J Hogan, Peter D Dueben, and Shannon L Mason. Machine learning emulation of 3D cloud radiative effects. *arXiv preprint arXiv:2103.11919*, 2021.

[21] Lazaros Oreopoulos, Eli Mlawer, Jennifer Delamere, Timothy Shippert, Jason Cole, Boris Fomin, Michael Iacono, Zhonghai Jin, Jiangnan Li, James Manners, Petri Räisänen, Fred Rose, Yuanchong Zhang, Michael J. Wilson, and William B. Rossow. The continual intercomparison of radiation codes: Results from phase I. *Journal of Geophysical Research: Atmospheres*, 117 (D6), 2012.

[22] Anikesh Pal, Salil Mahajan, and Matthew R. Norman. Using deep neural networks as cost-effective surrogate models for super-parameterized E3SM radiative transfer. *Geophysical Research Letters*, 46(11):6069–6079, 2019.

[23] Robert Pincus, Eli J. Mlawer, Lazaros Oreopoulos, Andrew S. Ackerman, Sunghye Baek, Manfred Brath, Stefan A. Buehler, Karen E. Cady-Pereira, Jason N. S. Cole, Jean-Louis Dufresne, Maxwell Kelley, Jiangnan Li, James Manners, David J. Paynter, Romain Roehrig, Miho Sekiguchi, and Daniel M. Schwarzkopf. Radiative flux and forcing parameterization error in aerosol-free clear skies. *Geophysical Research Letters*, 42(13):5485–5492, 2015.

[24] Forster P. M. Pincus R. and Stevens B. The radiative forcing model intercomparison project (RFMIP): experimental protocol for CMIP6. *Geoscientific Model Development*, 9, 2016.

[25] Stephan Rasp, Michael S. Pritchard, and Pierre Gentine. Deep learning to represent subgrid processes in climate models. *Proceedings of the National Academy of Sciences*, 115(39): 9684–9689, 2018.

[26] N. C. Swart, J. N. S. Cole, V. V. Kharin, M. Lazare, J. F. Scinocca, N. P. Gillett, J. Anstey, V. Arora, J. R. Christian, S. Hanna, Y. Jiao, W. G. Lee, F. Majaess, O. A. Saenko, C. Seiler, C. Seinen, A. Shao, M. Sigmond, L. Solheim, K. von Salzen, D. Yang, and B. Winter. The canadian earth system model version 5 (CanESM5.0.3). *Geoscientific Model Development*, 12 (11):4823–4873, 2019.

[27] Peter Ukkonen, Robert Pincus, Robin J. Hogan, Kristian Pagh Nielsen, and Eigil Kaas. Accelerating radiation computations for dynamical models with targeted machine learning and code optimization. *Journal of Advances in Modeling Earth Systems*, 12(12):e2020MS002226, 2020.

[28] Menno A. Veerman, Robert Pincus, Robin Stoffer, Caspar M. van Leeuwen, Damian Podareanu, and Chiel C. van Heerwaarden. Predicting atmospheric optical properties for radiative transfer computations using neural networks. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 379(2194):20200095, 2021.

[29] Knut von Salzen, John F. Scinocca, Norman A. McFarlane, Jiangnan Li, Jason N. S. Cole, David Plummer, Diana Verseghy, M. Cathy Reader, Xiaoyan Ma, Michael Lazare, and Larry Solheim. The canadian fourth generation atmospheric global climate model (CanAM4). part I: Representation of physical processes. *Atmosphere-Ocean*, 51(1):104–125, 2013.

[30] Janni Yuval and Paul A O'Gorman. Stable machine-learning parameterization of subgrid processes for climate modeling at a range of resolutions. *Nature communications*, 11(1):1–10, 2020.

Appendix

A CanESM and radiative transfer details

The Canadian Earth System Model (CanESM) CanESM is a comprehensive global model used to simulate Earth's climate past and present climate change as well as to make future climate projections. The most recent version of CanESM is version 5 [26]. CanESM5 simulates the atmosphere, ocean, sea-ice, land and carbon cycle, including the interactions between each of these components. The atmospheric component of CanESM5 is version 5 of the Canadian Atmospheric Model (CanAM5), which simulates a range of atmospheric physical processes, including radiation, convection, aerosols and clouds. CanAM5 uses parameterizations to represent these unresolved sub-gridscale processes, which are similar to those in its predecessor, CanAM4 [29].

CanESM5's radiative transfer parameterization The radiative transfer parameterization in CanESM5, is representative of the approach used in most modern ESMs. The optical properties of a number of components are accounted for, including the surfaces, aerosols, clouds and gases (represented using a correlated k-distribution model). The solar and thermal radiative transfer is computed using a 2-stream solution [29]. The unresolved, subgrid-scale variability of clouds is treated using the Monte Carlo Independent Column Approximation (McICA) [3]. The subgrid-scale variability of the surface albedo for solar and emissivity for thermal are accounted for in the radiative transfer calculations [26]. The performance of the CanESM radiative transfer code under pristine (gas-only), clear (gas plus aerosols) and all-sky conditions has been documented relative to line-by-line calculations and other radiative transfer models with similar complexity [21, 23].

Non-locality in radiative transfer While RT models used in large-scale models assume, with some justification, that radiation does not flow laterally between columns, they absolutely have to consider flows of radiation vertically. This means that heating and cooling at one layer depends on attenuation of radiation in all other layers. This non-locality complicates numerical simulation of the process greatly, and takes sizable amounts of computer resources to handle properly. Since the simplifying assumption of horizontally independent columns is expected to be employed for some time still, the hope is that the ML community can successfully apply or develop novel models that adequately handle the vertical non-local aspects of computing atmospheric RT.

B Dataset details

B.1 Dataset collection

Our dataset focuses on pristine-sky (no aerosols and no clouds) as well as clear-sky (no clouds) conditions, i.e. it leaves out the most general all-sky condition that includes clouds. These input conditions, which consist of surface properties and profiles of pressure, temperature, humidity, and trace gases, were simulated by setting input variables corresponding to clouds (and aerosols for pristine-sky) to zero. These input snapshots, for the respective atmospheric conditions, were then forwarded through CanESM5's RT physics code. for each atmospheric condition, the outputs are profiles of up- and down-welling fluxes for both, shortwave (solar) and longwave (thermal) radiation, plus their respective heating rates. These raw inputs and outputs are stored in separate NetCDF4 files for each snapshot. All together (for the main dataset of 1979-2014), they amount to over 1.5Tb of data.

B.2 Extreme volcanic eruption conditions

Occasionally, a volcanic eruption is large enough to inject material and gases well into the stratosphere. When that happens, the resulting aerosol loading spreads over the globe and can remain suspended for periods of time that are long enough to have measurable impacts on remote sensing data and surface-atmosphere climatic conditions. The overwhelming impact is a reduction of solar radiation absorbed by Earth, and hence a slight, but measurable and attributable, reduction in lower-atmospheric and surface temperatures (with other variables responding according which can both amplify or mitigate the initial cooling). To account for these radiative forcings with some confidence in a global model requires reliable input of height-dependent mass loading and aerosol optical properties. If

these can be supplied, simple solar RT models can predict accurate flux perturbations. For an ML model to be able to address them well, however, appropriate inputs and responses must be included in the training dataset. The added challenge is that not all volcanoes are equal and their time and location can result in distinct radiative forcings.

B.3 Complete list and description of variables

A complete list of all input variables can be found in Table 3, and for all potential target variables in Table 4. Within a CanESM grid box, the surface can include multiple types. An example of this is a grid box that includes a coast line which includes both land and water. The fraction of the grid box and its optical properties for a particular surface type is passed into the radiative transfer code where it is accounted for in the subsequent calculations.

The variable aerin holds information about the aerosols passed into the radiative transfer calculations. In the dataset provided these are aerosol mixing ratios. The third index of the arrays are associated with different aerosols simulated in CanESM5 [29],

- 1: SO4
- 2: Accumulation mode sea salt
- 3: Coarse mode sea salt
- 4: Accumulation mode dust
- 5: Coarse mode dust
- 6: Hydrophobic black carbon
- 7: Hydrophyllic black carbon
- 8: Hydrophobic organic carbon
- 9: Hydrophyllic organic carbon

B.4 Computing heating rates based on radiative fluxes

The heating rate h_l of any given layer $l \in \{1, .., S_{lay}\}$ can be directly computed based on the up- and down-welling fluxes of the two adjacent levels as follows:

$$h_l = c \cdot \frac{(F_{l+1}^{\text{up}} - F_{l+1}^{\text{down}}) - (F_l^{\text{up}} - F_l^{\text{down}})}{p_{l+1}^{\text{lev}} - p_l^{\text{lev}}}, \tag{1}$$

where $c \approx 9.76 \times 10^{-5}$, and F_k^{up}, F_k^{down}, and p_k^{lev} are the corresponding up-welling flux, down-welling flux, and pressure, of a level $k \in \{1, .., S_{lay} + 1\}$.

B.5 Dataset interface

B.5.1 Inputs pre-processing

To decrease the dataset size as well as mapping the raw variables into a format that is more amenable for ML models, we chose to concatenate the input variables that share the same spatial dimension across the feature/channel dimension. The information about which channel corresponds to which variable was saved, and is provided in the `META_INFO.json` file included in the dataset root directory. This results in three distinct input types and arrays per sample:

- *globals*: Consists of variables related to boundary conditions (e.g. sun angle), surface type variables, as well as geographical information (as described in B.8), which all do not have a spatial dimension (i.e. one, possibly multi-dimensional, variable per column/data example).
- *levels*: Consists of variables occurring at each level of the column (50 levels in this case). These are only four variables. It is worth recalling that the target radiative flux profiles are level variables.
- *layers*: Consists of variables occurring at each layer of the column (49 layers in this case). It is worth recalling that the target heating rate profiles are layer variables (although they can be computed based on the up- and down-welling fluxes).

Table 3: Definition of all the physical ***input variables*** (var.), and whether they are part of the *globals* (G), *layers* (Lay), or *levels* (Lev) input type. The storing scheme for the input variables is described in B.5.3. A cross in the Clear-sky column indicates that the corresponding variable is only used for experiments with clear-sky conditions.

Var. Name	Definition	G	Lay	Lev	Clear-sky
$shtj$	Eta coordinate at layer interfaces (levels)			✓	
$trow$	Temperature at levels			✓	
shj	Eta coordinate at layer mid-point		✓		
$dShj$	Layer thickness in eta coordinate		✓		
dz	Geometric thickness of the layer		✓		
$height$	Geometric height of a level			✓	
$tlayer$	Temperature at layer mid-point		✓		
$temp\_diff$	Temperature difference between levels		✓		
qc	Water vapour		✓		
$ozphs$	Ozone		✓		
$co2rox$	CO2 concentration		✓		
$ch4rox$	CH4 concentration		✓		
$n2orox$	N2O concentration		✓		
$f11rox$	CFC11 concentration		✓		
$f12rox$	CFC12 concentration		✓		
rhc	Relative humidity		✓		✓
$aerin$	Aerosol mass mixing ratios		✓		✓
sw_ext_sa	Solar extinction coefficient for stratospheric aerosols		✓		✓
sw_ssa_sa	Solar single scattering albedo for stratospheric aerosols		✓		✓
sw_g_sa	Solar asymmetry for stratospheric aerosols		✓		✓
lw_abs_sa	Thermal absorptivity for stratospheric aerosols		✓		✓
$pressg$	Surface pressure	✓			
$level\_pressure$	Level pressure			✓	
$layer\_pressure$	Layer pressure		✓		
$layer\_thickness$	Layer thickness in pressure		✓		
$gtrow$	Grid-mean surface temperature	✓			
$oztop$	Ozone above the top of the model	✓			
$cszrow$	Cosine of the solar zenith angle	✓			
$emisrow$	Grid-mean surface emissivity	✓			
$salbrol$	Grid-mean all-sky surface albedo	✓			
$csalrol$	Grid-mean clear-sky surface albedo	✓			
$emisrot$	Surface emissivity for each surface tile	✓			
$gtrot$	Surface temperature for each surface tile	✓			
$farerot$	Fraction of grid of each surface tile	✓			
$salbrot$	All-sky surface albedo for each surface tile	✓			
$csalrot$	Clear-sky surface albedo for each surface tile	✓			
$x\text{-}cord$	see B.8	✓			
$y\text{-}cord$	see B.8	✓			
$z\text{-}cord$	see B.8	✓			

Table 4: Definition of all the physical ***output variables*** (var.). The naming is the same for both pristine- and clear-sky, but are stored in different subdirectiories: *outputs_pristine/* and *outputs_clear_sky/* respectively. The profile type column indicates whether the variable profile is across the levels or layers of the column.

Var. Name	Definition	Profile type	Unit
$rsuc$	Up-welling shortwave (solar) flux	levels	W/m^2
$rsdc$	Down-welling shortwave (solar) flux	levels	W/m^2
$hrlc$	Solar heating rate profile	layers	K/s
$rluc$	Up-welling longwave (thermal) flux	levels	W/m^2
$rldc$	Down-welling longwave (thermal) flux	levels	W/m^2
$hrlc$	Thermal heating rate profile	layers	K/s

There are 82 global features per column, 4 level features per level, and 14 (45 for clear-sky) layer features per layer. Thus, all together there are a total of $2487 = 82 + 4 \times 50 + 45 \times 49$ (968 for pristine-sky) potential features.

B.5.2 Data normalization

For convenience, we provide pre-computed dataset statistics (mean, standard deviation, minimum and maximum) in the `statistics.npz` file that can be found in the root directory of the dataset. All statistics were computed on 1979-1990 + 1994-2004, i.e. on the years that we propose to use for training. Given the large sample size, it is important to use float64 precision for the mean and standard deviation in order to avoid numerical overflows. The statistics are provided for each input type, `in-type` $\in \{$*layers*, *levels*, *globals*$\}$, separately and the corresponding arrays have the same feature/channel dimensionality so that they can be directly used for normalization. The statistics that follow the naming `<statistic>_<in-type>` are concatenated scalar statistics for each variable. The statistics that follow the naming `spatial_<statistic>_<in-type>` were, additionally, computed for each level or layer separately (and are thus 2D arrays). In our experiments we used these statistics to scale the input data to have zero mean and unit standard deviation (*"z-scaling"*), as is common.

B.5.3 Storing scheme

Inputs Recall that each example in ClimART consists of three distinct input arrays that correspond to the *globals*, *layers*, and *levels* data subset. All three arrays are stored together in a single Hdf5 file for each year, which can all be found in the `inputs/` sub-directory.
The *layers* array is concatenated along the channel dimension in such a way, that the 14 first features are the ones needed for pristine-sky experiments, while the whole array would be used for clear-sky experiments. This avoids storage redundancy, and allows it to access the pristine-sky data by simple slicing of the *layers* array (see B.5.4 for the exact shape of the input arrays).

Outputs To allow flexible use of the potential target variables, we store one array per output variable together in a single Hdf5 file per year (a list of all possible target variables is given in Table 4). Since the targets differ between pristine- and clear-sky conditions, they are stored into the `outputs_pristine/` and `outputs_clear_sky/` sub-directories, respectively.

Directory structure The dataset is stored as separate Hdf5 files for each year (filenames follow <*year*>*.h5*). From the dataset root directory the structure thus follows:

- `META_INFO.json`
- `statistics.npz`
- `inputs/`
  - 1850.h5
  - 1851.h5
  - 1852.h5
  - 1979.h5

  - ...
  - 2014.h5
  - ...
  - 2097.h5
  - 2098.h5
  - 2099.h5
- `outputs_pristine/`
  - *Same as for inputs*
- `outputs_clear_sky/`
  - *Same as for inputs*

B.5.4 Inputs dimensions

For this reason and to avoid storage redundancy, we store one single input array for both pristine- and clear-sky conditions. The dimensions of ClimART's input arrays are:

- *layers*: $(N, S_{\text{lay}}, D_{\text{lay}})$
- *levels*: $(N, S_{\text{lev}}, D_{\text{lev}})$
- *globals*: (N, D_{glob}),

where N is the data dimensions (i.e. the number of examples of a specific year), S_{lay} and S_{lev} are the number of layers and levels in a column respectively (49 and 50 in this case), and D_{lay}, D_{lev}, D_{glob} is the number of features/channels for *layers*, *levels*, *globals* respectively. For both pristine-sky and clear-sky conditions, we have that $D_{\text{lev}} = 4$ and $D_{\text{glob}} = 82$, while $D_{\text{lay}} = 14$ for pristine-sky, and $D_{\text{lay}} = 45$ for clear-sky conditions (see B.5.1 for details on the nature of this). The array for pristine-sky conditions can be easily accessed by slicing the first 14 features out of the stored array, e.g.:

$$\texttt{pristine\_array} = \texttt{layers\_array}[:, :, :14] \tag{2}$$

B.6 Reading the dataset in Python

Using Python, ClimART's input and target arrays can be accessed as follows (for the example year 2007, and assuming that the user wants to predict longwave heating rates under pristine-sky conditions):

```
# Assume that h5py and numpy are installed and we are in the root data directory.
    import h5py
    import numpy as np
    with h5py.File("inputs/2007.h5", 'r') as h5f:
        X = {
            'layers': np.array(h5f['layers'][..., :14]), # for clear-sky targets no slicing is needed!
            'levels': np.array(h5f['levels']),
            'globals': np.array(h5f['globals'])
        }
    with h5py.File("outputs_pristine/2007.h5", 'r') as h5f:
        Y = np.array(h5py['hrlc']) # or take any other variable from Table 4
```

B.7 Dataset split sizes

Recall that each snapshot (the state of CanESM5 at some timestep) consists of 8192 columns/samples. Further, recall that by sampling every 205 hours, each year contains either 42 or 43 snapshots (344064 or $352,256$ total samples).

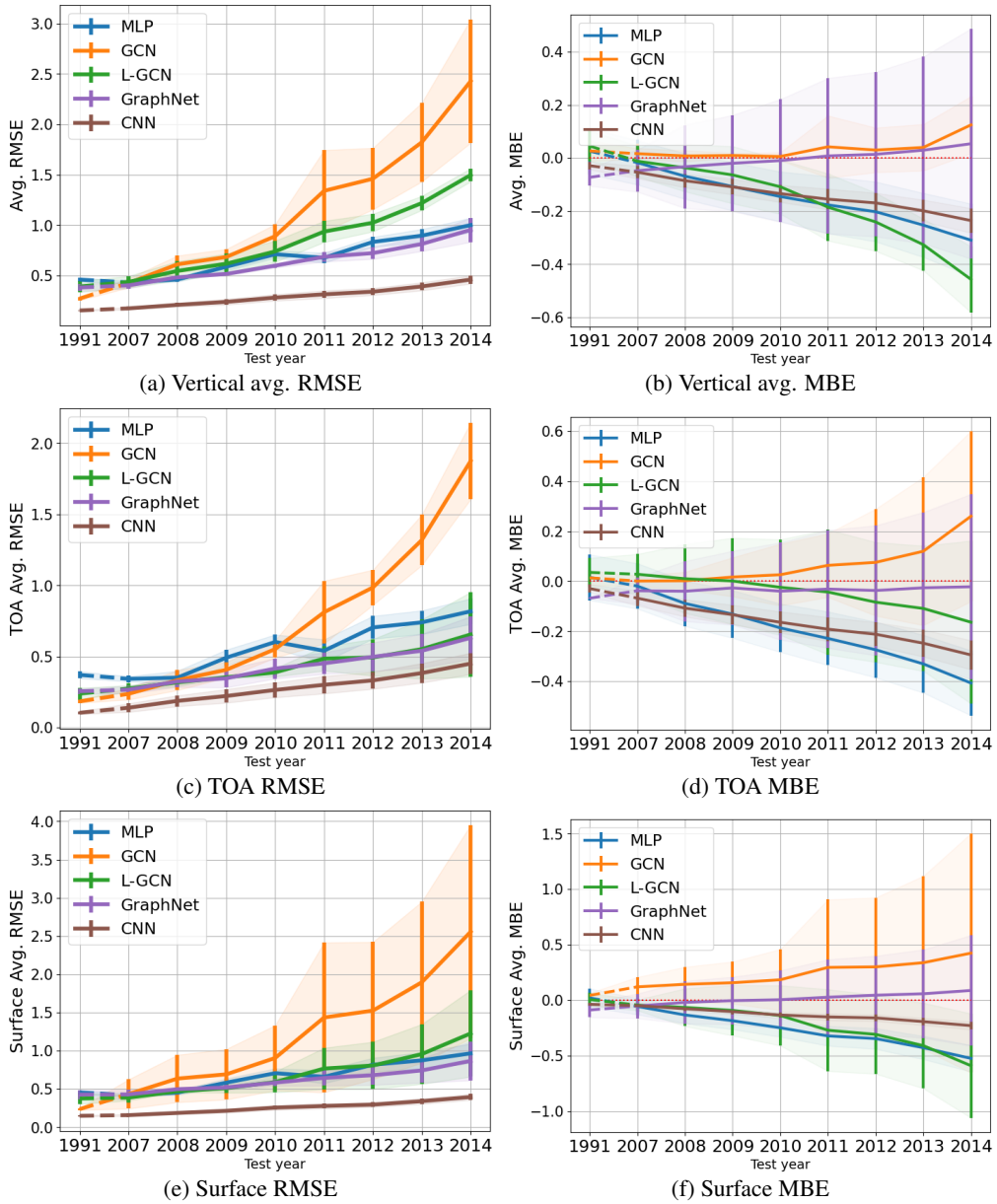
We provide the complete data for the years 1979 to 2006, excluding the years 1992-93 in order to avoid potential data leakage when using 1991 as an out-of-distribution test set. In total there are thus 10,076,160 samples for this period. Thus, minus the held-out year 1991, this results in up to $9,732,096$ potential training samples from present-day conditions.

For the suggested testing period, 2007 to 2014 (inclusive), we randomly subsampled 15 out of the 43 snapshots per year (giving $122,880$ distinct samples per year). In total this results in $983,040$ samples for the eight testing years. Randomly subsampling on a yearly basis ensures a diverse test set (as opposed to sampling the snapshots from the same yearly timesteps), which is further magnified by the yearly variability.

B.8 Adding geographical information

Coordinates in the latitude-longitude system are two features used to represent a 3-D space. Due to this, they are not the optimal choice for a ML model to get informed about the three dimensional Earth. To deal with this issue, we map them to x, y, and z coordinates on a unit sphere. This ensures that the extreme longitudes are close by in the new coordinates. Concretely, we set for each columns with latitude lat and longitude lon as follows:

$$x\text{-}cord = \cos(lat) * \cos(lon) \qquad y\text{-}cord = \cos(lat) * \sin(lon) \qquad z\text{-}cord = \sin(lat)$$

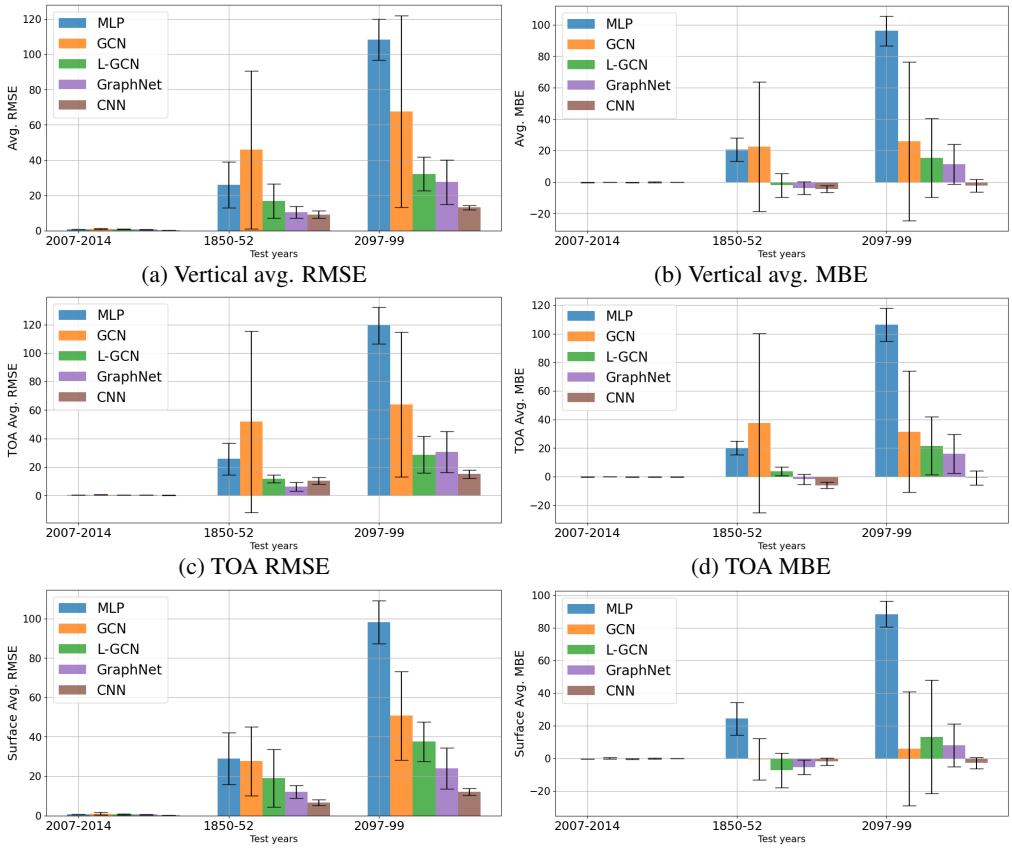


(a) Vertical avg. RMSE (b) Vertical avg. MBE

(c) TOA RMSE (d) TOA MBE

(e) Surface RMSE (f) Surface MBE

Figure 5: Performance as a function of the test year at different levels for our baseline models. (Fig. 5a) and (Fig. 5b) show the errors vertically averaged over all levels of a column (profile). The TOA errors are shown in (Fig. 5c) & (Fig. 5d) and the error at the surface is presented in (Fig. 3a) & (Fig. 3b). Apart from the superior performance of CNN, it's interesting to note the miniscule mean bias error (MBE) of the GraphNet. which is an important property for climate simulations.



(a) Vertical avg. RMSE (b) Vertical avg. MBE

(c) TOA RMSE (d) TOA MBE

(e) Surface RMSE (f) Surface MBE

Figure 6: Performance as a function of the test year at different levels for our baseline models. (Fig. 6a) and (Fig. 6b) show the errors vertically averaged over all levels of a column (profile). Apart from the vertically averaged errors, it's important to calculate the errors in top of the atmosphere (TOA) and surface levels as they're used for the calculation of heating rates from the predicted radiative flux. The TOA errors are shown in (Fig. 6c) & (Fig. 6d) and the error at the surface is presented in (Fig. 6e) & (Fig. 6f). As expected, CNNs and Graph-based models (L-GCN & GraphNet) are far more superior in all the levels compared to the MLPs for whom the error in future predictions is higher by and order of a magnitude.

C Experiments

C.1 Implementation details

C.1.1 Model architectures

MLP The MLP used for our experiments is a simple three layer MLP with the following hidden-layer dimensions: $\langle 512, 256, 256 \rangle$. As an MLP takes unstructured 1D data as input, all the input variables need to be flattened into a single vector for the MLP.

CNN For the CNN model, we use a 3-layer network with kernel sizes $\langle 20, 10, 5 \rangle$ and the corresponding strides set as $\langle 2, 2, 1 \rangle$. The channels parameter is given by $\langle 200, 400, 100 \rangle$, with the last channels setting it equal to the input size. We then apply a global average pooling over the resulting tensor to get the output. To preprocess the data for CNN, we pad the surface and layers variable to match the dimensions of levels variable. Then the result is concatenated and fed to the model.

Graph Convolutional Network (GCN) and L-GCN We use a three-layer GCN [14] with hidden dimensionality 128 and residual connections. As nodes of the graph we use all three input types: *layers*, *levels*, and *globals*. The latter is mapped to a global node that is connected to all other nodes,

while for the edge structure we use a simple line-graph that contains connections between adjacent levels and layers only. Thus, the graph has $49 + 50 + 1 = 100$ nodes. As is standard practice, we add self-loops to the adjacency matrix, see Fig. 4a for a visualization of the resulting adjacency matrix. Since *layers*, *levels*, and *globals* are heterogenous data arrays with different numbers of features, we project them to a hidden size of 128 with a separate 1-layer MLP for each of the input types, before passing it to the GCN. The MLP projectors use LayerNorm and GeLU as activation function. The GCN backbone is the same for both GCN and L-GCN, i.e. L-GCN only differs from GCN in its structure learning module, which is identical to the one proposed by [7]. To get predictions we use a 1-layer MLP head that takes as input mean-pooled node embeddings generated by the last GCN layer.

Graph network We use a three-layer graph network (GraphNet)[4], i.e. with three sequential graph network blocks, that do not share weights. As in [4], each GraphNet block consists of three update functions for each of the three graph components: global, node, and edge features. The update functions are modeled by distinct 1-layer MLPs with hidden size of 128. Each block uses residual connections. For the graph structure, we use similarly to the GCN a line-graph with self-loops. However, a GraphNet enables more modeling flexibility, since we can get rid of the global node in the GCN and instead map it to the global feature vector of a GraphNet. A GraphNet also supports edge features, thus we map the layer features to be edge features (and thus layers be treated as edges between adjacent levels). As nodes of the graph we then use the remaining 50 levels. Similarly to the GCN, we stack a 1-layer MLP on top of the last GraphNet block to predict the desired number of outputs. The MLP inputs are the mean-pooled node (i.e. levels) representations of the last layer. We choose to pool from the nodes/levels since the target variables – up- and down-welling flux profiles – are level variables too.

C.1.2 Hyperparameters

Recall from 4.3, that we use the years 1990, 1999, and 2003 for training, while validating on 2005 and testing on the proposed test set years 2007-2014. For all the models, we normalize the input data by subtracting the mean and dividing by the standard deviation that were computed on the potential training years $\{1979 - 90, 1994 - 2004\}$. The targets are *not* normalized in any form, but directly predicted in their raw form by all models. The batch size used for training all the models was fixed at 128. All models use the GeLU activation function [11]. For the optimizer, Adam, we use a weight decay of 1e-6 and an exponential decay learning rate scheduler (with gamma $= 0.98$, and a minimum learning rate of 1e-6). We clip the L2 gradient norm of all our models at 1, which is important due to the unnormalized targets. We use LayerNorm [2] for the MLP, while all other models do not use any network normalization – these configurations were found empirically to be superior for the respective models.

C.2 Eigenvector centrality analysis

Following [7], we analyze the learned adjacency matrix of L-GCN (see Fig. 4b for the explicit structure), via the node eigenvector centrality score method. See [7] for details of the method. A high centrality score for a node translates to the node being important within the graph. In the particular case of a GCN the score reflects into the core message-passing forward-pass [14], since the node propagates its information to a greater extent than other nodes. Our eigenvector centrality analysis shows that L-GCN learns to assign a high importance to the global node, see Fig. 7. The figure shows how the score for the global node converges across differently seeded runs to a very high score of over 0.8 (in the later epochs no other node has a score that surpasses 0.5, and most nodes have scores lower than 0.05). This underlines the importance of using the non-spatial *globals* information that contains important boundary conditions like the sun angle as well as surface type and geographical related information.

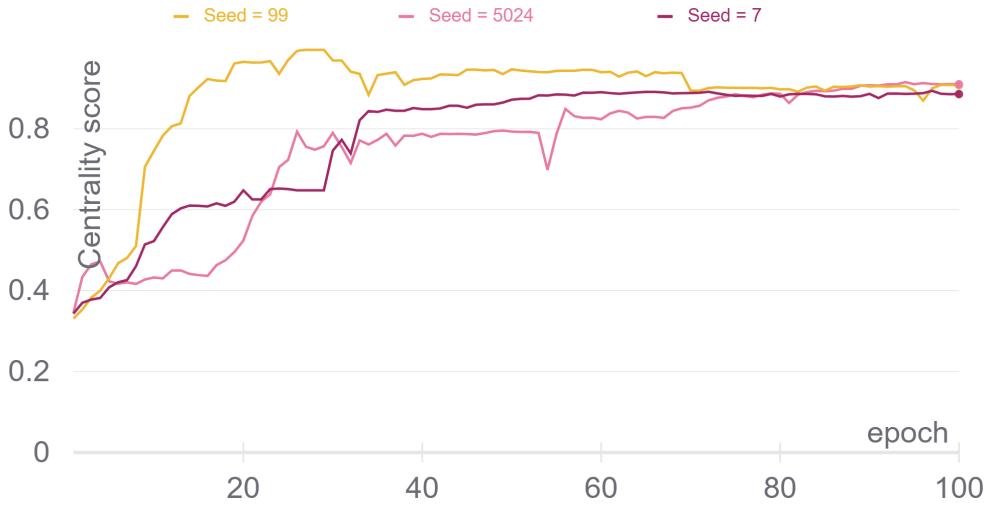


Figure 7: L-GCN, a graph convolutional network with learnable adjacency matrix, learns to give high importance to the global node, which contains boundary conditions information, as measured by its high eigenvector centrality score (for the learned adjacency matrix). We plot this score as a function of the epoch for all three differently seeded runs of L-GCN. See Appendix C.2 for more discussion.