# 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:

$$h_l = c \cdot \frac{(F^{\text{up}}_{l+1} - F^{\text{down}}_{l+1}) - (F^{\text{up}}_{l} - F^{\text{down}}_{l})}{p^{\text{lev}}_{l+1} - p^{\text{lev}}_{l}}, \tag{1}$$

where $c \approx 9.76\text{e}^{-3}$, and $F^{\text{up}}_k, F^{\text{down}}_k$, and $p^{\text{lev}}_k$ 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*: $(N, D_{\text{glob}})$,

where $N$ is the data dimensions (i.e. the number of examples of a specific year), $S_{\text{lay}}$ and $S_{\text{lev}}$ are the number of layers and levels in a column respectively (49 and 50 in this case), and $D_{\text{lay}}$, $D_{\text{lev}}$, $D_{\text{glob}}$ 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).

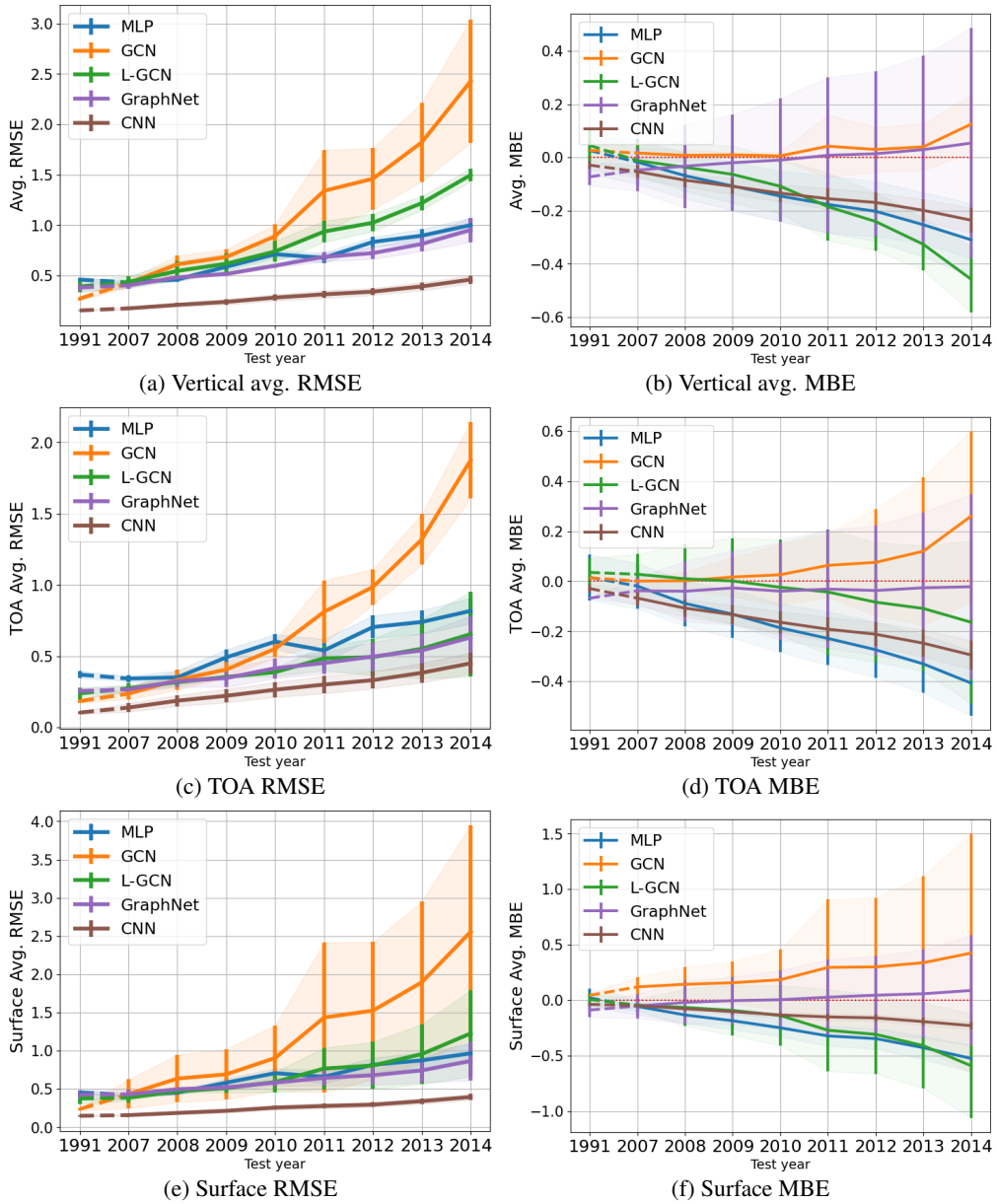
We provide the complete data for the years 1979 to 2006, excluding the years 1992-93 in order to avoid potential data leakage when using 1991 as an out-of-distribution test set. In total there are thus 10,076,160 samples for this period. Thus, minus the held-out year 1991, this results in up to $9,732,096$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

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

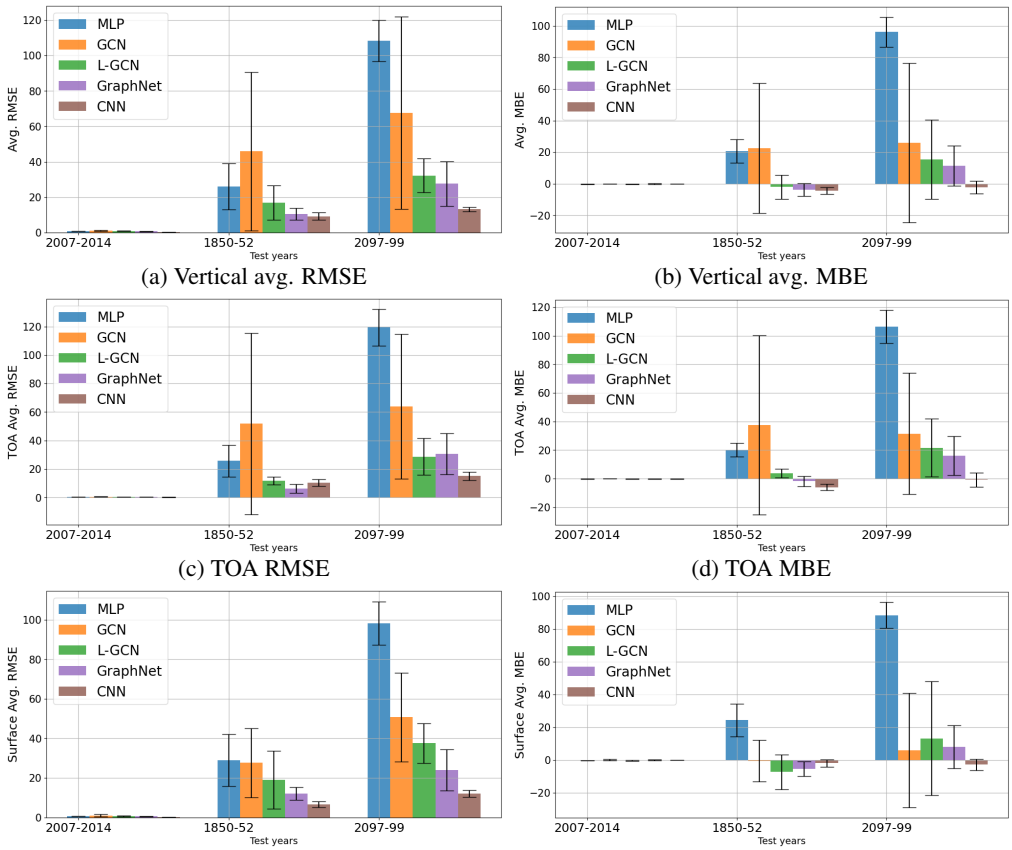



(a) Vertical avg. RMSE (b) Vertical avg. MBE

(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.

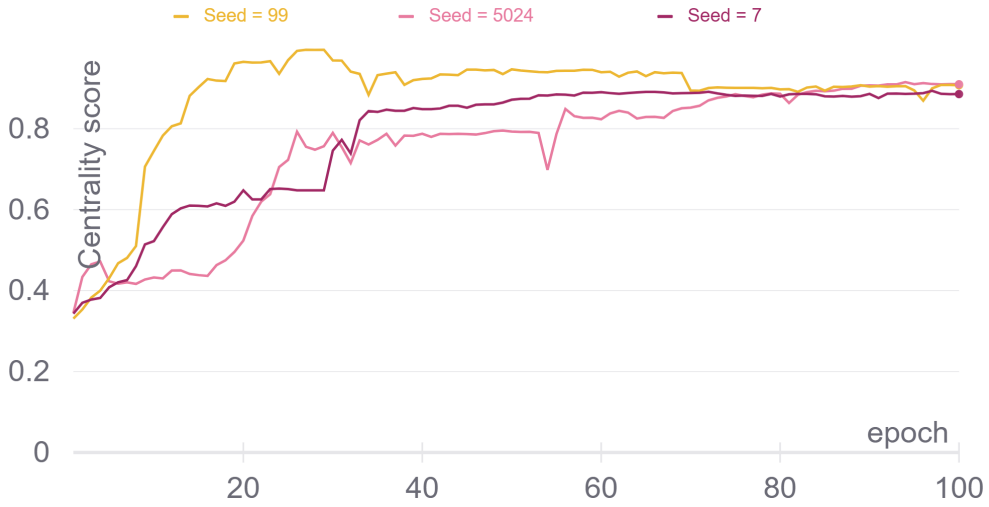



Figure 7: L-GCN, a graph convolutional network with learnable adjacency matrix, learns to give high importance to the global node, which contains boundary conditions information, as measured by its high eigenvector centrality score (for the learned adjacency matrix). We plot this score as a function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