Exploring Randomly Wired Neural Networks for Climate Model Emulation

William Yik
Harvey Mudd College
Claremont, CA
wyik@hmc.edu

Sam Silva
Dept. of Earth Sciences
Dept. of Civil and Environmental Engr.
University of Southern California
Los Angeles, CA
samsilva@usc.edu

Andrew Geiss
Pacific Northwest National Laboratory
Richland, WA
andrew.geiss@pnnl.gov

Duncan Watson-Parris
Atmospheric, Oceanic and Planetary Physics
Department of Physics
University of Oxford Oxford, UK
duncan.watson-parris@physics.ox.ac.uk

Abstract

Exploring the climate impacts of various anthropogenic emissions scenarios is key to making informed decisions for climate change mitigation and adaptation. State-of-the-art Earth system models can provide detailed insight into these impacts, but have a large associated computational cost on a per-scenario basis. This large computational burden has driven recent interest in developing cheap machine learning models for the task of climate model emulation. In this manuscript, we explore the efficacy of *randomly wired neural networks* for this task. We describe how they can be constructed and compare them to their standard feedforward counterparts using the ClimateBench dataset. Specifically, we replace the dense layers in multilayer perceptrons, convolutional neural networks, and convolutional long short-term memory networks with randomly wired ones and assess the impact on model performance for models with 1 million and 10 million parameters. We find average performance improvements of 4.2% across model complexities and prediction tasks, with substantial performance improvements of up to 16.4% in some cases. Furthermore, we find no significant difference in prediction speed between networks with standard feedforward dense layers and those with randomly wired layers. These findings indicate that randomly wired neural networks may be suitable direct replacements for traditional dense layers in many standard models.

1 Introduction

Characterizing the response of the Earth system to future emissions scenarios is key to informing climate change mitigation and adaptation strategies. Modern Earth system models (ESMs) are incredibly useful tools for this task, providing detailed climate projections far into the future for scenario-based analysis. However, as the process complexity and spatial resolution of such ESMs increases, so does their computational cost [1]. Consequently, it becomes impractical to explore a wide range of possible emissions scenarios with modern ESMs, limiting their applicability to a restricted set of future scenarios [2]. To address this large computational cost, the climate modeling research community frequently makes use of emulators, a set of computational tools that aim to approximate the complex relationships in a full ESM at a fraction of the computational cost. Recent developments have demonstrated the considerable promise of machine learning (ML) techniques such

Tackling Climate Change with Machine Learning: workshop at NeurIPS 2022.

as linear models [3, 4, 5], tree based methods [6, 7, 8], and various implementations of neural networks [9, 10, 11, 12] in these emulation tasks for both individual model components [13, 8, 14, 15, 16] and entire ESM predictions [17, 18, 5, 19]. ML models often enable accurate predictions at greatly reduced computational cost relative to full ESMs. This is broadly indicative of a very large potential design space for ML model architectures. Here, we investigate so-called "randomly wired neural networks" in an effort to further explore this architecture design space for climate model emulation. Randomly wired neural networks are a special class of neural network where components are connected in a random, rather than structured, manner. This is in direct contrast to widely-used fully connected artificial neural networks, where each component is connected only to the preceding and following components in the neural network. Randomly wired neural networks have demonstrated skill in a variety of domains, including handwriting recognition [20], internet network attack detection [21], and aerosol optics prediction [22]. Random wiring is a form of neural architecture search (NAS) [23] that searches a more complete space of connectivity patterns than other common NAS strategies [24] to identify high-performing model architectures.

In this work, we evaluate the suitability of randomly wired neural networks for climate emulation using the ClimateBench benchmarking dataset. We specifically investigate the use of random wiring to predict temperature and precipitation statistics. To that end, we compare the performance of random wiring against fully connected counterparts within three types of neural network models covering a wide range of complexities: multilayer perceptrons (MLPs), convolutional neural networks (CNNs), and convolutional long short-term memory networks (CNN-LSTMs). We find that random network wiring shows competitive results across model architectures and prediction tasks with average performance improvements of 4.2%, up to 16.4% in some cases. Furthermore, we find that randomly wired neural networks take no longer to make predictions than their standard fully connected counterparts. Our work indicates that in many cases, random wiring can serve as direct replacements for traditional feedforward neural networks to improve predictive skill.

2 Randomly Wired Neural Networks

In our random networks, as well as those of Xie et al. [23] and Geiss et al. [22], data tensors flow through the network in a feedforward manner similar to standard multilayer perceptrons (MLPs). However, unlike the dense layers in MLPs, dense layers in our random networks receive inputs from any number of preceding layers and pass their outputs to any number of subsequent layers. That is, there may be “skip connections" between dense layers. An example of this class of random neural network, which we will henceforth refer to as RandDense networks, is illustrated in Figure 1. Here, each layer is represented as a node in a graph and edges connect nodes following the flow of tensors in a given connectivity pattern.

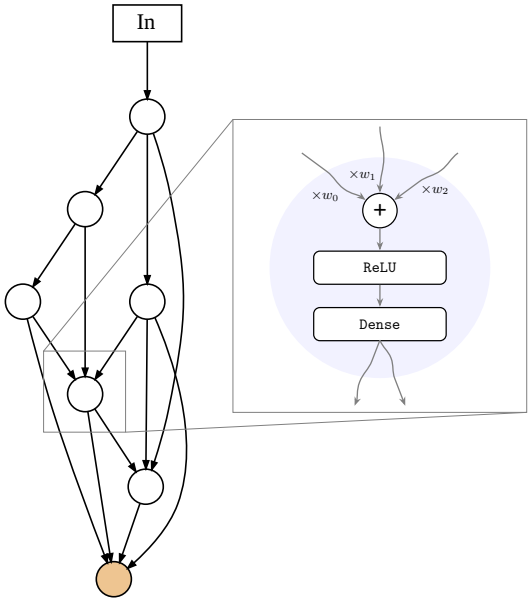


Figure 1: Graph representation of a six layer RandDense network along with its node operation in the inset. White circles represent hidden dense layers and their activation functions. The brown circle represents the output layer. Lastly, the upper rectangular block represents neural network layers preceding the dense layers. In this case, it is a simple input layer which performs no operation, but other choices such as a convolutional block are possible (see Section 4).

In the style of Xie et al. [23], our RandDense networks have several key differences with standard networks beyond their connectivity patterns. First, since RandDense nodes/layers may have multiple inbound edges, unlike the nodes of an MLP which only have one, every node of a RandDense network begins with an aggregation of incoming tensors via a weighted sum with learnable, positive weights. Similarly, our RandDense nodes may have multiple outbound edges, which send out copies of their output,

while MLPs only have one. Lastly, our random networks apply a ReLU activation function before the dense layer so that the outbound tensor may have both positive and negative values, preventing weighted sums at nodes with high input degree from becoming too large [23]. Additional implementation details of our RandDense networks is included in Appendix A.1, including the network connectivity generation process and the input/output nodes.

3 Data

Standard benchmarks are a valuable tool for the intercomparison of ML methods. Here, we use the ClimateBench dataset, along with its associated evalution criteria and metrics as a standardized framework for objectively evaluating ML-driven climate model emulators [18]. The dataset contains four main anthropogenic forcing agents as model inputs: carbon dioxide (CO_2), sulfur dioxide (SO_2), black carbon (BC), and methane (CH_4). While SO_2 and BC are provided as annual mean spatial distributions across a 96×144 global grid, the longer lived and well mixed CO_2 and CH_4 inputs are provided as annual global total concentration and global average emissions, respectively. These four inputs are a subset of the input data used drive the Norwegian Earth System Model version 2 (NorESM2) [25].

The ClimateBench task is to predict the annual mean surface air temperature (TAS), annual mean diurnal temperature range (DTR), annual mean total precipitation (PR), and 90th percentile of daily precipitation (PR90) as predicted by NorESM2. For the years 2015-2100, each of these variables are provided as spatial distributions across the same 96×144 grid as BC and SO_2. Our models are each trained to predict one of TAS, DTR, PR, or PR90. We use the first two years of every decade from every training set experiment as validation data, train for 100 epochs with an early stopping patience of 10, and otherwise follow the ClimateBench model training framework [18].

4 Experiments

We perform experiments three baseline neural network models: a standard MLP, convolutional neural network (CNN), and a convolutional long short-term memory (CNN-LSTM). These architectures span a range of model complexity and are meant to represent realistic deep learning architectures appropriate for climate emulation tasks. We compare each model's performance with sequentially connected dense layers to performance using a RandDense network on the ClimateBench task (see Section 3). We evaluate standard and RandDense networks with either 1- or 10-million parameters, and in each case train 50 models with layer counts ranging from 2-10, and report maximum and mean performance across the ensemble. All networks are evaluated using the "total RMSE" metric defined in the ClimateBench framework [18]. Additional implementation details for each model architecture type can be found in Appendix A.2.

Table 1: Best RMSE performance for each model class across all generated models. Lower is better, and the better RMSE between the standard and RandDense models is bolded.

		TAS	DTR	PR	PR90
MLP	Standard	1.928	15.62	4.663	5.651
	RandDense	**1.612**	**14.67**	**4.472**	**5.206**
CNN	Standard	**3.350**	23.15	9.235	10.30
	RandDense	3.353	**22.92**	**8.681**	**9.964**
CNN-LSTM	Standard	**0.262**	11.85	2.861	3.880
	RandDense	0.263	**11.66**	**2.775**	**3.810**

While mean performance comparisons varied for random models and standard models, the best performing RandDense models were almost always better at predicting their respective ClimateBench variable than the best performing standard models. A summary of these best performance results is shown in Table 1 with additional results in Appendix A.3. The best standard dense networks only outperformed the best RandDense models in two instances, both of which show very minor performance differences: the TAS prediction task for the CNN and CNN-LSTM architectures. In

general, the performance gains for randomization within each model architecture are more stark for the precipitation variables (PR and PR90). The skip connections in our RandDense networks have previously been shown to be helpful for non-linear problems [26], and therefore may provide more benefit in modeling precipitation, which scales non-linearly with the emulator inputs [27]. On average, the best RandDense networks performed 4.2% better than the best standard networks, with performance improvements of up to 16.4% in the MLP baseline TAS prediction task. Furthermore, we found no statistically significant difference between the run times of standard and RandDense models, indicating that they are suitable replacements for standard models in climate model emulation tasks.

The CNN-LSTM RandDense model performed best overall. Figure 2 shows the global distribution of the mean deviations of its best predictions from the ground truth over the test set. In most locations the mean deviations are statistically insignificant ($p > 0.05$ using a two-sided independent sample t-test) and are shown as white in the plot. Furthermore, the significant errors are relatively small. For example, the mean prediction errors for TAS are almost all within than 0.3 K (generally less than 12%). Global distribution plots of mean deviation for the other model architectures may be found in Appendix A.3.

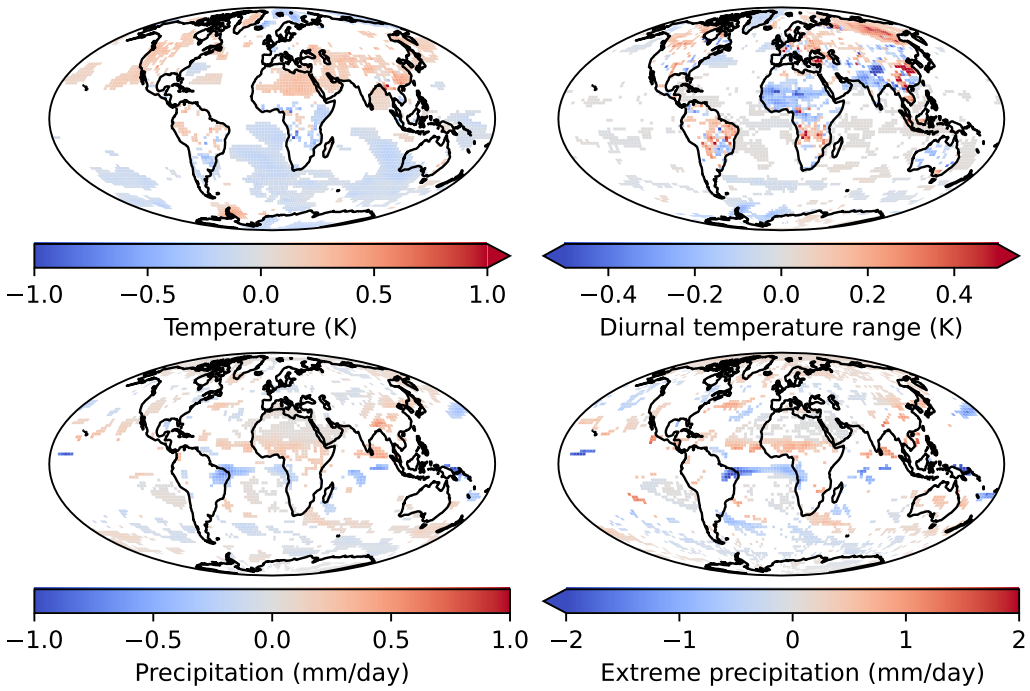


Figure 2: Mean differences between ClimateBench NorESM2 simulated target variables and the best performing CNN-LSTM emulators averaged over the test scenario between 2080-2100. Statistically insignificant differences $(p > 0.05)$ are masked.

5 Conclusion

Motivated by the promise of randomly wired neural networks in related applications, we explored random wirings between dense layers of neural networks for the task of climate model emulation. We replaced the traditional feedforward dense layers in MLPs, CNNs, and CNN-LSTMs with our randomly wired networks, coined "RandDense" networks, and conducted performance experiments using the ClimateBench dataset. Across several architectures, model complexities, and predictands, we found performance benefits for models containing randomly wired layers. Our work indicates that in many cases, standard feedforward networks may be effectively replaced with RandDense networks to achieve better performance at no additional computational cost.

References

[1] Matthew Collins, Richard E Chandler, Peter M Cox, John M Huthnance, Jonathan Rougier, and David B Stephenson. Quantifying future climate change. *Nature Climate Change*, 2(6):403–409, 2012.

[2] Brian C O'Neill, Claudia Tebaldi, Detlef P Van Vuuren, Veronika Eyring, Pierre Friedlingstein, George Hurtt, Reto Knutti, Elmar Kriegler, Jean-Francois Lamarque, Jason Lowe, et al. The scenario model intercomparison project (scenariomip) for cmip6. *Geoscientific Model Development*, 9(9):3461–3482, 2016.

[3] Stephan Rasp. Coupled online learning as a way to tackle instabilities and biases in neural network parameterizations: general algorithms and lorenz 96 case study (v1. 0). *Geoscientific Model Development*, 13(5):2185–2196, 2020.

[4] Sam J Silva, Colette L Heald, and Alex B Guenther. Development of a reduced-complexity plant canopy physics surrogate model for use in chemical transport models: A case study with geos-chem v12. 3.0. *Geoscientific Model Development*, 13(6):2569–2585, 2020.

[5] Laura A Mansfield, Peer J Nowack, Matt Kasoar, Richard G Everitt, William J Collins, and Apostolos Voulgarakis. Predicting global patterns of long-term climate change from short-term simulations using machine learning. *npj Climate and Atmospheric Science*, 3(1):1–9, 2020.

[6] 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.

[7] Janni Yuval, Paul A O'Gorman, and Chris N Hill. Use of neural networks for stable, accurate and physically consistent parameterization of subgrid atmospheric processes with good performance at reduced precision. *Geophysical Research Letters*, 48(6):e2020GL091363, 2021.

[8] Sam J Silva, Po-Lun Ma, Joseph C Hardin, and Daniel Rothenberg. Physically regularized machine learning emulators of aerosol activation. *Geoscientific Model Development*, 14(5):3067–3077, 2021.

[9] Ilan Price and Stephan Rasp. Increasing the accuracy and resolution of precipitation forecasts using deep generative models. In *International Conference on Artificial Intelligence and Statistics*, pages 10555–10571. PMLR, 2022.

[10] Oliver Watt-Meyer, Noah D Brenowitz, Spencer K Clark, Brian Henn, Anna Kwa, Jeremy McGibbon, W Andre Perkins, and Christopher S Bretherton. Correcting weather and climate models by machine learning nudged historical simulations. *Geophysical Research Letters*, 48(15):e2021GL092555, 2021.

[11] 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.

[12] Stephan Rasp and Nils Thuerey. Data-driven medium-range weather prediction with a resnet pretrained on climate simulations: A new model for weatherbench. *Journal of Advances in Modeling Earth Systems*, 13 (2):e2020MS002405, 2021.

[13] Axel Seifert and Stephan Rasp. Potential and limitations of machine learning for modeling warm-rain cloud microphysical processes. *Journal of Advances in Modeling Earth Systems*, 12(12):e2020MS002301, 2020.

[14] Griffin Mooers, Michael Pritchard, Tom Beucler, Jordan Ott, Galen Yacalis, Pierre Baldi, and Pierre Gentine. Assessing the potential of deep learning for emulating cloud superparameterization in climate models with real-geography boundary conditions. *Journal of Advances in Modeling Earth Systems*, 13(5): e2020MS002385, 2021.

[15] Matthew Chantry, Sam Hatfield, Peter Dueben, Inna Polichtchouk, and Tim Palmer. Machine learning emulation of gravity wave drag in numerical weather forecasting. *Journal of Advances in Modeling Earth Systems*, 13(7):e2021MS002477, 2021.

[16] Chang Suk Lee, Eunha Sohn, Jun Dong Park, and Jae-Dong Jang. Estimation of soil moisture using deep learning based on satellite data: A case study of south korea. *GIScience & Remote Sensing*, 56(1):43–67, 2019.

[17] Stephan Rasp, Peter D Dueben, Sebastian Scher, Jonathan A Weyn, Soukayna Mouatadid, and Nils Thuerey. Weatherbench: a benchmark data set for data-driven weather forecasting. *Journal of Advances in Modeling Earth Systems*, 12(11):e2020MS002203, 2020.

[18] Duncan Watson-Parris, Yuhan Rao, Dirk Olivié, Øyvind Seland, Peer J Nowack, Gustau Camps-Valls, Philip Stier, Shahine Bouabid, Maura Dewey, Emilie Fons, and et al. Climatebench: A benchmark dataset for data-driven climate projections. *Earth and Space Science Open Archive*, page 36, 2021. doi: 10.1002/essoar.10509765.1. URL `https://doi.org/10.1002/essoar.10509765.1`.

[19] Lea Beusch, Lukas Gudmundsson, and Sonia I Seneviratne. Emulating earth system model temperatures with mesmer: from global mean temperature trajectories to grid-point-level realizations on land. *Earth System Dynamics*, 11(1):139–159, 2020.

[20] Erol Gelenbe and Yongha Yin. Deep learning with random neural networks. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 1633–1638. IEEE, 2016.

[21] Olivier Brun, Yonghua Yin, Erol Gelenbe, Y Murat Kadioglu, Javier Augusto-Gonzalez, and Manuel Ramos. Deep learning with dense random neural networks for detecting attacks against iot-connected home environments. In *International ISCIS Security Workshop*, pages 79–89. Springer, Cham, 2018.

[22] Andrew Geiss, Po-Lun Ma, Balwinder Singh, and Joseph C Hardin. Emulating aerosol optics with randomly generated neural networks. *EGUsphere*, pages 1–21, 2022.

[23] Saining Xie, Alexander Kirillov, Ross Girshick, and Kaiming He. Exploring randomly wired neural networks for image recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1284–1293, 2019.

[24] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *The Journal of Machine Learning Research*, 20(1):1997–2017, 2019.

[25] Øyvind Seland, Mats Bentsen, Dirk Olivié, Thomas Toniazzo, Ada Gjermundsen, Lise Seland Graff, Jens Boldingh Debernard, Alok Kumar Gupta, Yan-Chun He, Alf Kirkevåg, et al. Overview of the norwegian earth system model (noresm2) and key climate response of cmip6 deck, historical, and scenario simulations. *Geoscientific Model Development*, 13(12):6165–6200, 2020.

[26] A Emin Orhan and Xaq Pitkow. Skip connections eliminate singularities. *arXiv preprint arXiv:1701.09175*, 2017.

[27] Maria Fernanda Cabré, SA Solman, and MN Nuñez. Creating regional climate change scenarios over southern south america for the 2020's and 2050's using the pattern scaling technique: validity and limitations. *Climatic Change*, 98(3):449–469, 2010.

[28] 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.

[29] Duong Tran Anh, Song P Van, Thanh D Dang, and Long P Hoang. Downscaling rainfall using deep learning long short-term memory and feedforward neural network. *International Journal of Climatology*, 39(10):4170–4188, 2019.

[30] MA Ghorbani, Ravinesh C Deo, Zaher Mundher Yaseen, Mahsa H Kashani, and Babak Mohammadi. Pan evaporation prediction using a hybrid multilayer perceptron-firefly algorithm (mlp-ffa) model: case study in north iran. *Theoretical and applied climatology*, 133(3):1119–1131, 2018.

[31] Evangelos Rozos, Panayiotis Dimitriadis, Katerina Mazi, and Antonis D Koussis. A multilayer perceptron model for stochastic synthesis. *Hydrology*, 8(2):67, 2021.

[32] Zewen Li, Fan Liu, Wenjie Yang, Shouheng Peng, and Jun Zhou. A survey of convolutional neural networks: analysis, applications, and prospects. *IEEE transactions on neural networks and learning systems*, 2021.

[33] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*, 2015.

[34] Kevin Trebing, Tomasz Stańczyk, and Siamak Mehrkanoon. Smaat-unet: Precipitation nowcasting using a small attention-unet architecture. *Pattern Recognition Letters*, 145:178–186, 2021.

[35] Arnaud Mounier, Laure Raynaud, Lucie Rottner, Matthieu Plu, Philippe Arbogast, Michaël Kreitz, Léo Mignan, and Benoît Touzé. Detection of bow echoes in kilometer-scale forecasts using a convolutional neural network. *Artificial Intelligence for the Earth Systems*, pages 1–66, 2022.

[36] Cheolhee Yoo, Daehyeon Han, Jungho Im, and Benjamin Bechtel. Comparison between convolutional neural networks and random forest for local climate zone classification in mega urban areas using landsat images. *ISPRS Journal of Photogrammetry and Remote Sensing*, 157:155–170, 2019.

[37] Anton Nikolaev, Ingo Richter, and Peter Sadowski. Deep learning for climate models of the atlantic ocean. In *AAAI Spring Symposium: MLPS, 2020*, 2020.

[38] Corinna Cortes, Mehryar Mohri, and Afshin Rostamizadeh. L2 regularization for learning kernels. *arXiv preprint arXiv:1205.2653*, 2012.

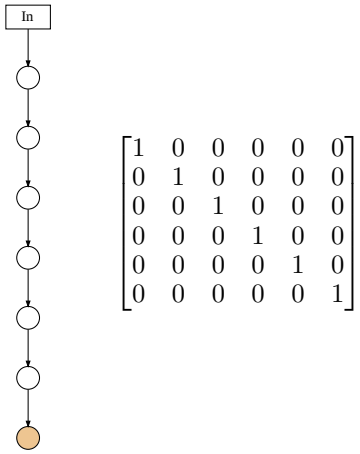
[39] AA Ahmed, Ravinesh C Deo, Afshin Ghahramani, Nawin Raj, Qi Feng, Zhenliang Yin, and Linshan Yang. Lstm integrated with boruta-random forest optimiser for soil moisture estimation under rcp4. 5 and rcp8. 5 global warming scenarios. *Stochastic Environmental Research and Risk Assessment*, 35(9):1851–1881, 2021.

[40] Moyang Liu, Yingchun Huang, Zhijia Li, Bingxing Tong, Zhentao Liu, Mingkun Sun, Feiqing Jiang, and Hanchen Zhang. The applicability of lstm-knn model for real-time flood forecasting in different climate zones in china. *Water*, 12(2):440, 2020.

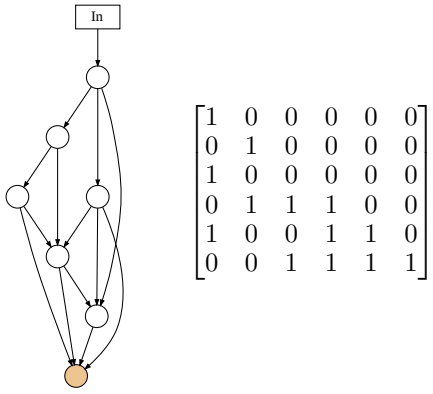
A Supplemental Material

A.1 RandDense Implementation Details

Network connectivity. Since the layers within our RandDense networks are randomly wired, the first step in generating such a network is determining which layers are connected to which, or the network connectivity. Following Geiss et al. [22], given a fixed number of layers n, we randomly select the number of neurons per dense layer and generate an adjacency matrix representing the connections between layers in our RandDense network. Since our RandDense networks are still feedforward in nature, several constraints may be placed on the adjacency matrix. If we let each row represent a layer, and column values represent connections (inbound edges) from previous layers, then the adjacency matrix must be lower triangular. Thus, there are $n(n+1)/2$ possible connections for an n layer random network. The corresponding adjacency matrices for the network in Figure 1 and a standard MLP are shown in Figure 3. Some of these connectivity patterns may have layers without an inbound or outbound tensor. Thus, to ensure the network is valid, we follow Geiss et al. [22] and iterate through each row and column, randomly activating an edge in each row/column if it has no active edges.



(a) A six layer MLP along with its corresponding adjacency matrix.



(b) The six layer RandDense network from 1 along with its corresponding adjacency matrix.

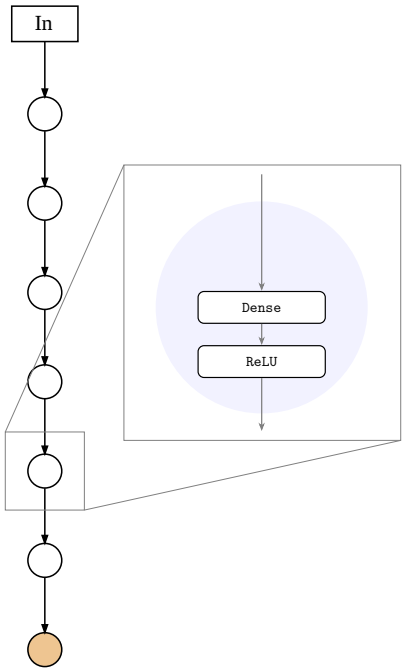
Figure 3: Graph representations of a six layer MLP and six layer RandDense network shown side-by-side with their respective adjacency matrices.

Node operations. Nodes in a RandDense network may have multiple inbound or outbound edges, unlike the nodes of an MLP which only have one of each (see Figure 3). To treat this difference, we must define a new operation that occurs at every node.

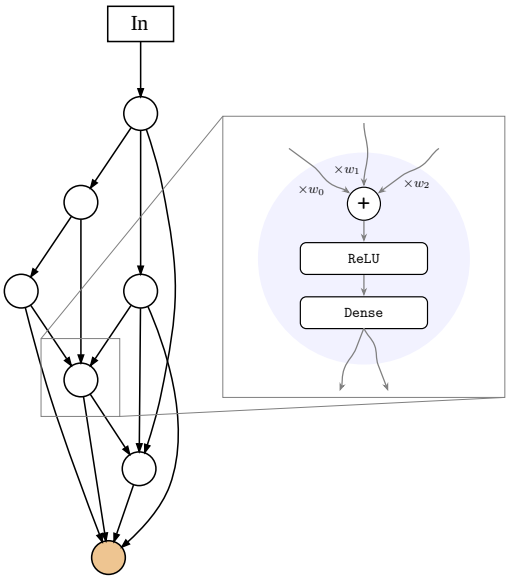
The node operation of our RandDense networks begins with an aggregation of the incoming tensors. If a node has more than one inbound edge, we follow Xie et al. [23] and aggregate the incoming tensors via a weighted sum with learnable, positive weights. The weights are kept positive by applying the sigmoid function. Exclusively summing inbound tensors instead of concatenating them [22] requires a consistent number of neurons per layer throughout the network, but avoids extremely large input tensors that might occur using concatenation. Additionally, in the style of Xie et al. [23], our random networks have the ReLU activation function *before* the dense layer so that the outbound tensor can contain both positive and negative values. This avoids extremely large weighted sums when the number of inputs to a given layer is high. Figure 4 illustrates the differences between a node, which contains the dense layer and its activation, in a standard MLP and our random networks. Notice that a node in the graph representation of an MLP in Figure 4a contains a dense layer followed by ReLU activation, while a node of the RandDense network in Figure 4b contains an weighted summation followed by ReLU activation then the dense layer.

Input and output nodes. Like the hidden layers of a RandDense network, the first and last layers of such a network are different than those of a traditional MLP. In particular, the first (input) node may be connected with multiple other nodes downstream instead of just one, and the last (output) node may have multiple inbound edges instead of just one. As such, we make a few minor changes to the network to account for these differences.

We start with the input node, the first node immediately following the input layer. If it is connected to multiple downstream nodes, it will simply send out a identical copy of its outbound tensor along each outbound edge. We must also carefully define the input node's internal node operation. Similarly to Xie et al. [23] our general node



(a) An MLP network with six hidden layers, along with its node operation. As is typical in MLP networks, the activation function, ReLU in this case, follows the dense layer. Furthermore, each node only has one input and output edge. Thus, unlike the RandDense network, there is no aggregation by weighted sum.



(b) A RandDense network with six hidden layers, along with its node operation. Aggregation from three previous input nodes is done via weighted sum with weights w_0, w_1, and w_2. The summation is followed by ReLU activation and the dense layer. Lastly, two identical copies of the output are sent to two separate nodes downstream.

Figure 4: Graph representations of a six layer MLP and six layer RandDense network shown side-by-side with their respective node operations. White circles represent hidden dense layers and their activation functions. Brown circles represent the output layer discussed in Section 2. Lastly, the upper rectangular block represents neural network layers preceding the dense layers. In this case, it is a simple input layer which performs no operation, but other choices such as a convolutional block are possible (see Section A.2).

operation discussed previously (dense layer followed by ReLU) is the same for every node in the graph. That is, each dense layer will have the same number of neurons, which is randomly chosen. However, this number of neurons is likely not the same as the required input and output shape of the network. As such, we make a few minor changes to the network so that it is valid. First, the node immediately following the input layer will contain a number of neurons equal to the difference between the randomly selected layer size and the input size. For example, if the randomly selected layer size is 100 and the input size is 12, the first randomly wired node following the input layer will contain a dense layer of size 88. Then, this dense layer is concatenated with the input to create a layer of the correct size. This is done so that any nodes downstream of the first node may still have direct access to the input [22].

For the output node which may have multiple inbound edges, we simply take the average of all inbound tensors and send this value to a final dense layer of the desired output size. This final output node, which is colored brown in Figure 4b, has linear activation instead of ReLU so that both positive and negative values may be output.

A.2 Experimental Setup

In this section we provide additional implementation details for each model architecture (MLP, CNN, and CNN-LSTM) and further discuss our experimental setup.

MLP. A standard MLP model is the most common and basic type of neural network which only contains dense layers. They have been extensively used in climate modelling tasks such as subgrid process representation [28],

rainfall downscaling [29], longwave raditaion emulation [11], and evaporation prediction [30], and stochastic synthesis in hydrology simulations [31].

We begin with the first parameter limit of 1 million. We generate MLP models for a given number of hidden layers by randomly selecting a fixed layer size for all of the hidden layers such that the network's parameter count is 1 million $\pm 10\%$. We generate 50 such MLP models with 2 hidden dense layers, 50 MLPs with 3 hidden dense layers, and so on up to 10 hidden dense layers. This gives 450 MLP models. The process is repeated for a parameter limit of 10 million, yielding a total of 900 MLPs. For the RandDense comparison networks, we repeat a similar process. Using the network generation method described in Section 2, we generate 50 RandDense networks for 2-10 hidden layers at both the 1 million and 10 million parameter count for a total of 900 RandDense networks.

As discussed in Section 3, the inputs to the models are global CO_2 and CH_4, as well as 96×144 grids of SO_2 and BC. However, in MLPs each input is handled by one neuron in the input layer. As such, if we directly fed the anthropogenic forcer inputs to the model, the input layer would be over 26,000 neurons wide. This is a extremely high layer size, and so we follow Watson-Parris et al. [18] and perform dimensionality reduction on the SO_2 and BC inputs. Specifically, we take the first five emperical orthogonal functions (EOFs) of each to replace their 96×144 grids. Thus, the total input size to the MLP is 12: the first five EOFs for SO_2 and BC, plus global CO_2 and CH_4.

CNN. Convolutional neural networks (CNNs) have been widely adopted in object tracking and image recognition tasks because they are able to model spacial dependencies [32, 33]. CNNs have also been recently adopted for various weather and climate tasks such as precipitation nowcasting [34], bow echo detection [35], climate zone classification [36] and ocean modelling [37].

Following a similar process as with the MLPs, we generate 50 MLPs and 50 RandDense networks with 2-10 hidden dense layers for both 1 and 10 million parameter counts. However, rather than being standalone networks, these models are appended to a convolutional block. This block begins with a convolutional layer with 20 filters, a kernal size of 3, ReLU activation, and L_2 regularization [38]. The convolutional layer is followed by average pooling with a stride of 2 and global average pooling.

Since CNNs are designed to handle images with multiple channels, we preserve the original dimensionality of the input variables, unlike the transformation we conducted for the standalone MLP and RandDense networks in the previous subsection. Specifically, we are able to feed the 96×144 maps of SO_2 and BC to the convolutional network, as well as global CO_2 and CH_4. In order to treat the four input variables as four channels of one 96×144 "image" of the globe, CO_2 and CH_4 are transformed to 96×144 grids where each grid cell has the same value, the global concentration of CO_2 or emissions of CH_4.

CNN-LSTM. Long short-term memory (LSTM) networks are a type of recurrent neural network (RNN) which model temporal dependencies. This time-aware property makes them useful for a range of forecasting tasks such as precipitation downscaling [29], soil moisture estimation [39], and flood forecasting [40]. For the specific task of climate model emulation, a combined CNN-LSTM model has been shown to outperform CNN and LSTM models in isolation [18].

As in the previous two architecture setups, for this architecture we generate 50 MLPs and 50 RandDense networks with 2-10 hidden layers for 1 and 10 million parameter counts. These models are then appended to a CNN-LSTM block. This block first consists of the same convolutional, average pooling, and global average pooling layers as in the previous subsection, but with each of them time distributed (applied in the same way) across every 10 year time window within the training samples. These time distributed layers are followed by an LSTM with 25 units.

In order to enable the CNN-LSTM block to make time-aware predictions, the training data is transformed slightly by slicing it into 10 year time windows. For example, two such windows might be 2015-2024 and 2016-2025. Like the CNN model of the previous subsection, we are able to preserve the original high dimensionality (96×144) of the input variables.

A.3 Additional Results

In this section we present the full results of our experimental testing, including both mean and best performance graphics. Figure 5 shows a comparison of mean RMSE performance across both parameter counts and all layer counts we tested in our experiments. Points below the $y = x$ line in black indicate that the RandDense networks outperformed the MLP networks, and vice versa for points above the $y = x$ line. The farther a point is from the $y = x$ line, the more drastic the performance difference. Figure 6 is similar but shows best RMSE performances. Figures 7 and 8 show these same mean and best performance metrics for the CNN and CNN RandDense models, with Figures 9 and 10 showing these metrics for the CNN-LSTM and CNN-LSTM RandDense models. While mean performance for temperature variables show mixed results, in almost model type the RandDense variations show better best performance across the temperature variables and better mean and best performance across both precipitation variables.

Figure 11 shows the global distribution of the mean deviations of the best MLP/RandDense predictions from the ground truth over the test set. Figures 12 and 13 show the same mean deviation plots for the best CNN/CNN RandDense and CNN-LSTM/CNN-LSTM RandDense emulators, respectively.

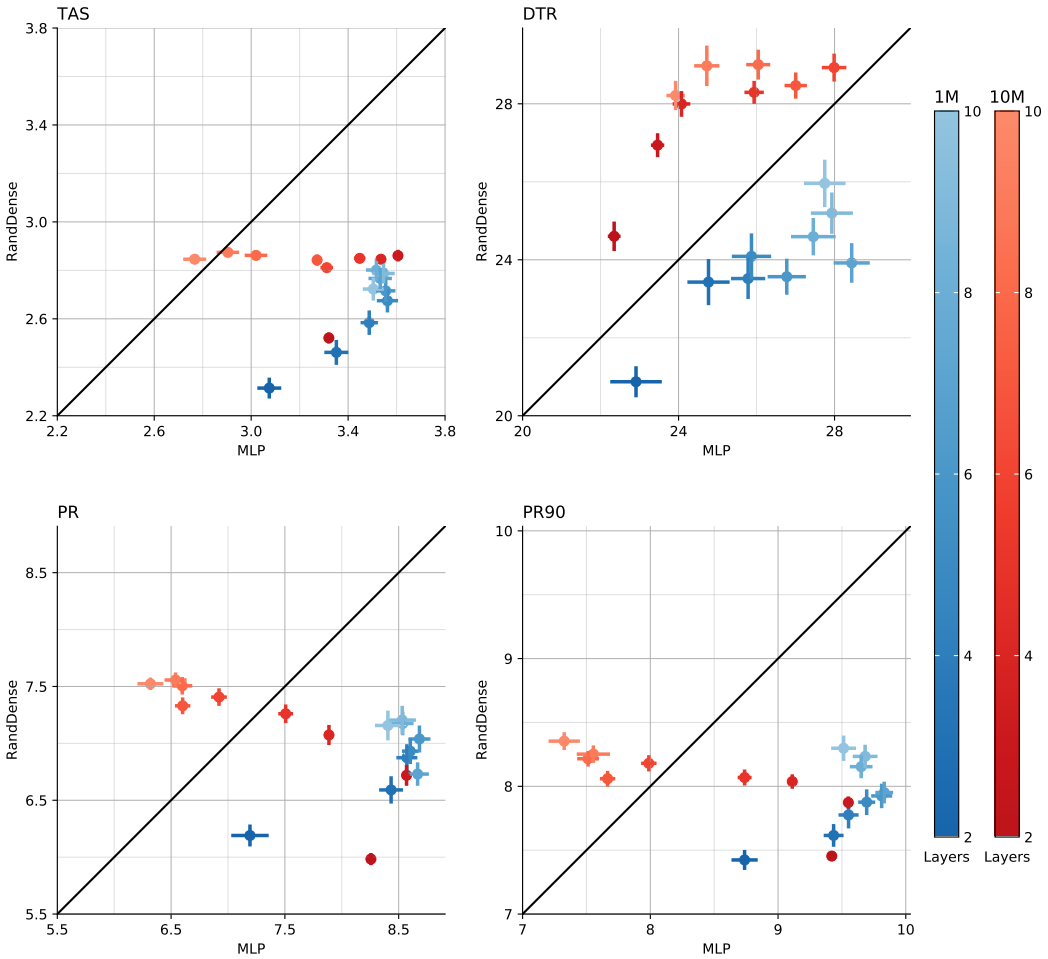


Figure 5: Mean RMSE of 50 MLP models vs. mean RMSE of 50 RandDense models for both TAS, DTR, PR, and PR90. The color heatmaps to the right indicate the number of hidden layers. Errorbars show $\pm$ standard error of the mean.

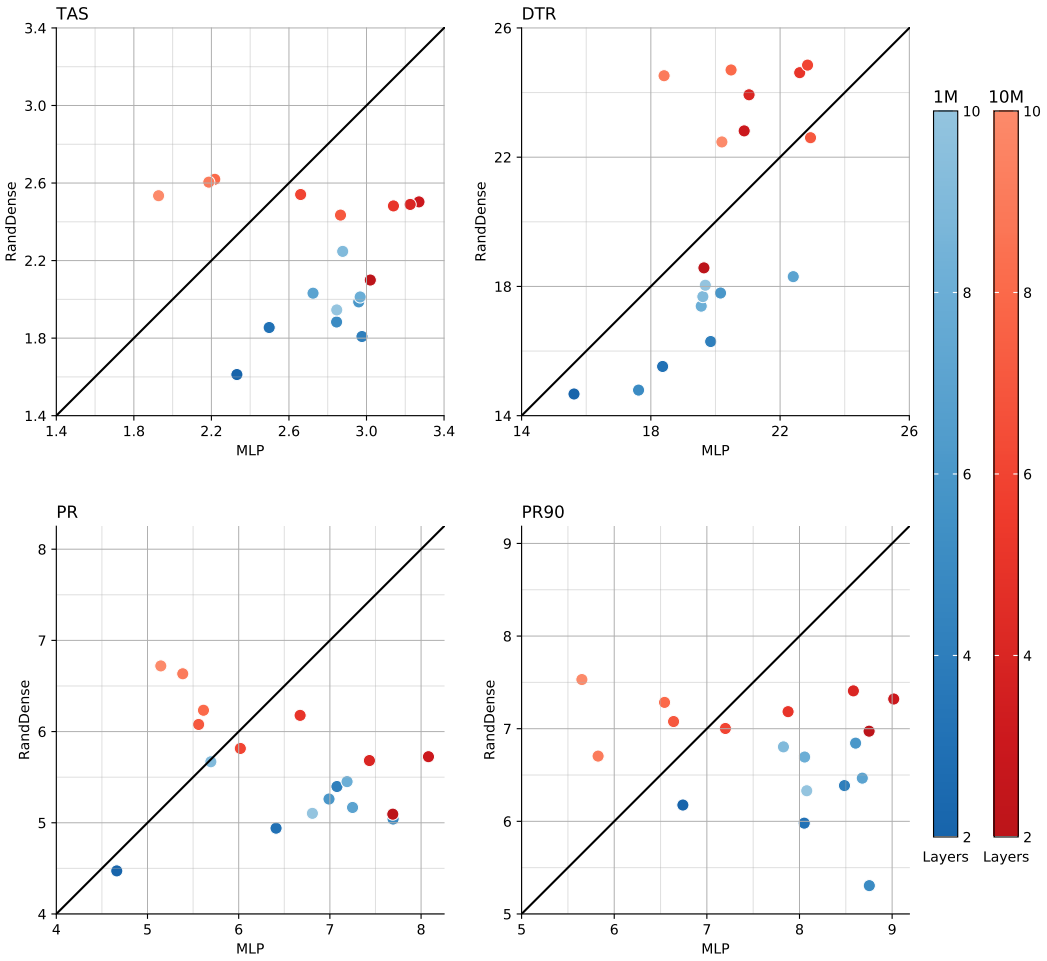


Figure 6: Best RMSE of 50 MLP models vs. best RMSE of 50 RandDense models for both TAS, DTR, PR, and PR90. The color heatmaps to the right indicate the number of hidden layers.

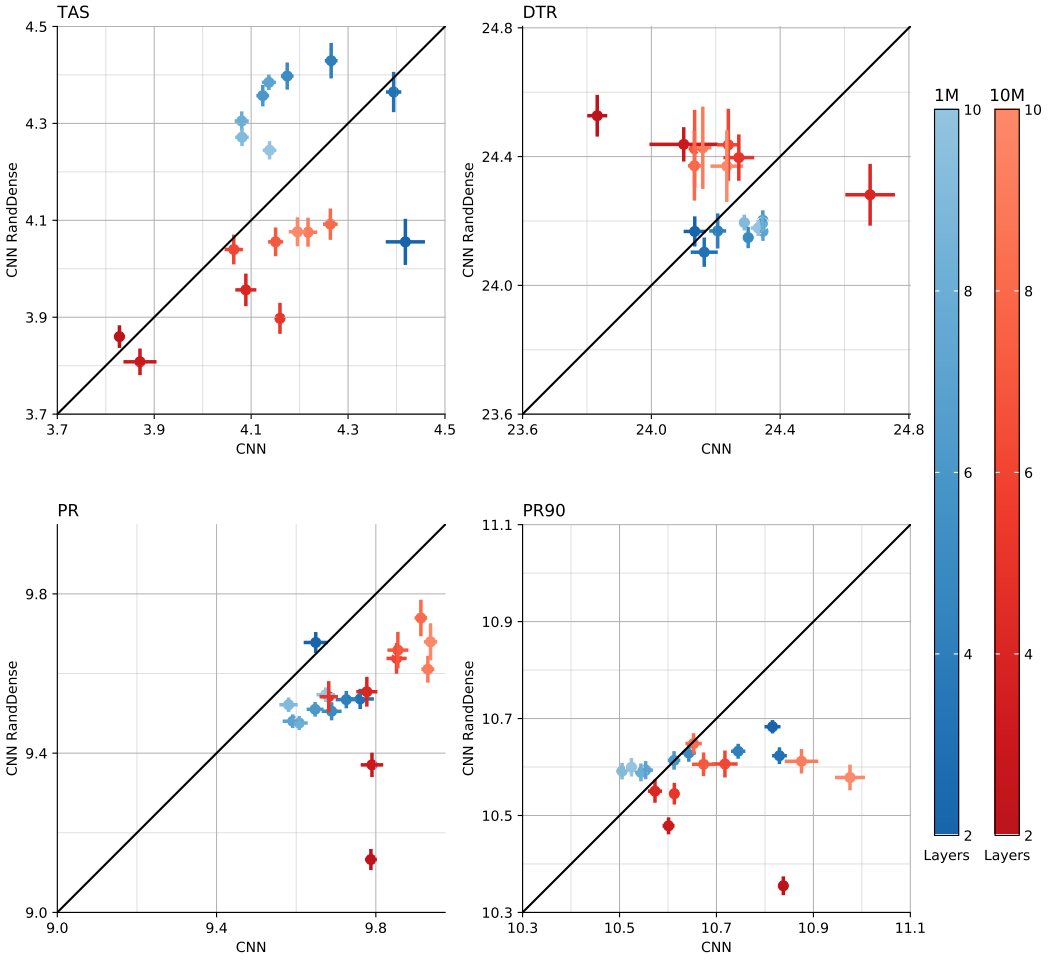


Figure 7: Mean RMSE of 50 CNN models vs. mean RMSE of 50 CNN RandDense models for both TAS, DTR, PR, and PR90. The color heatmaps to the right indicate the number of hidden layers. Errorbars show $\pm$ standard error of the mean.

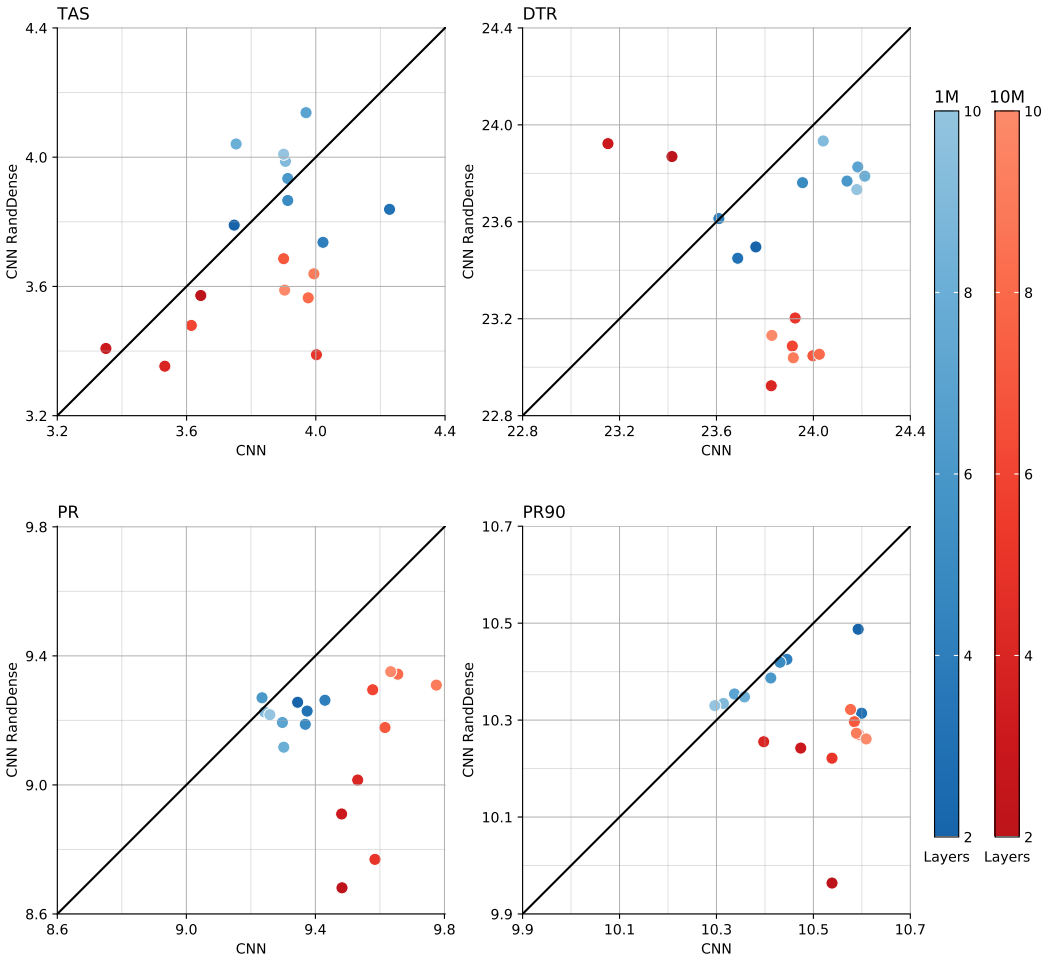


Figure 8: Best RMSE of 50 CNN models vs. best RMSE of 50 CNN RandDense models for both TAS, DTR, PR, and PR90. The color heatmaps to the right indicate the number of hidden layers.

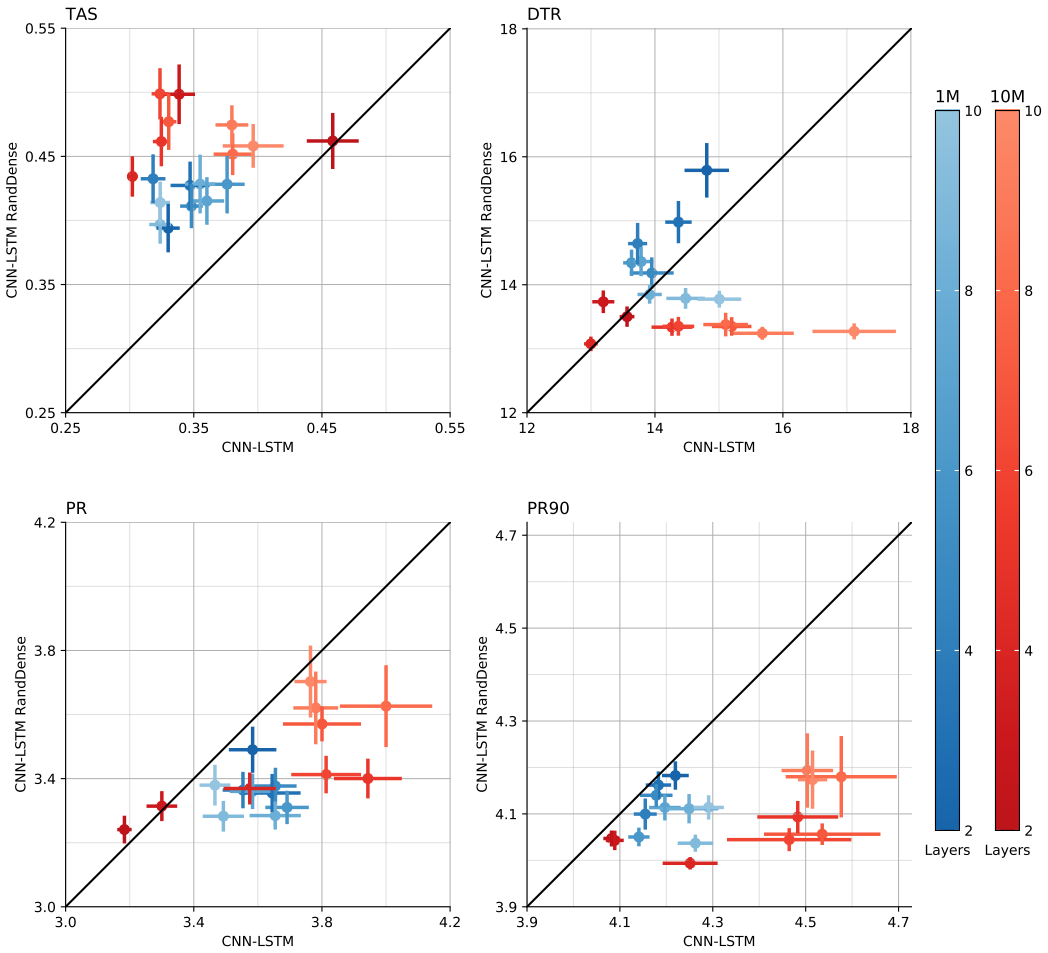


Figure 9: Mean RMSE of 50 CNN-LSTM models vs. mean RMSE of 50 CNN-LSTM RandDense models for both TAS, DTR, PR, and PR90. The color heatmaps to the right indicate the number of hidden layers. Errorbars show $\pm$ standard error of the mean.

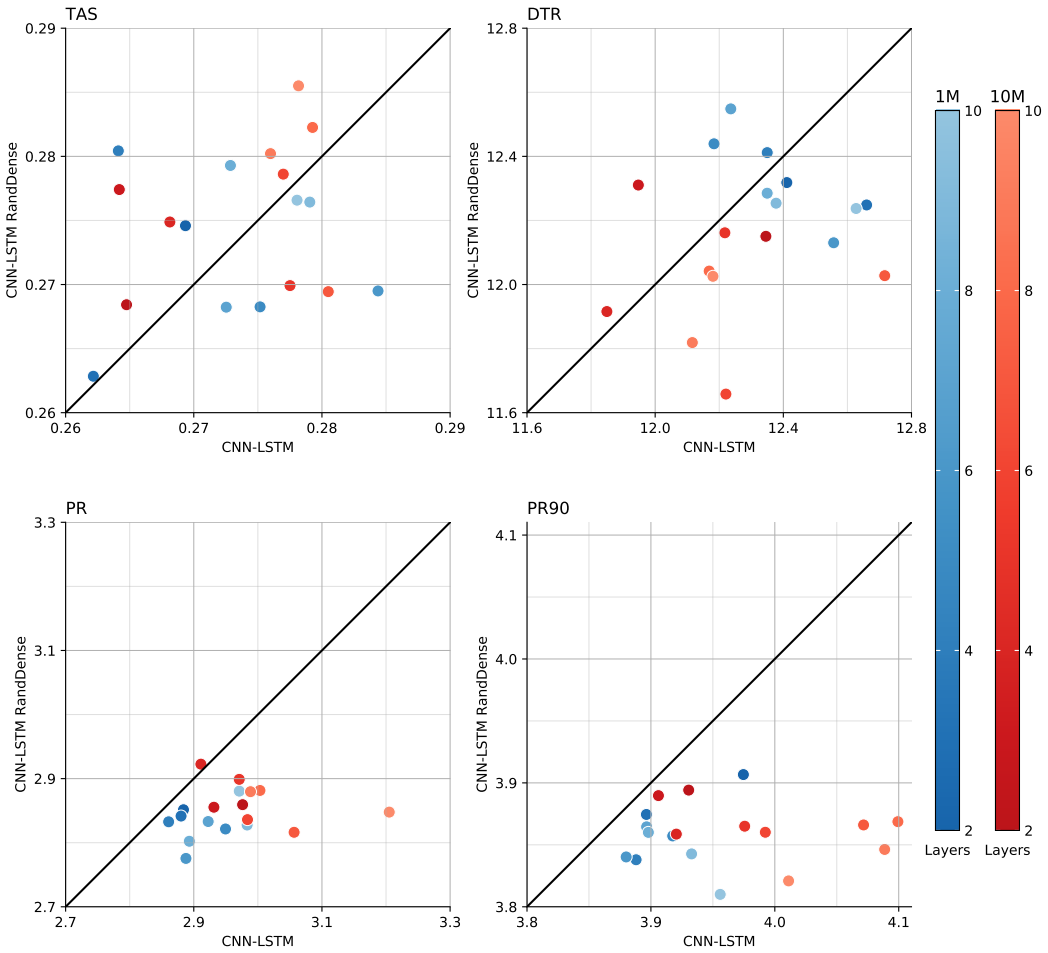


Figure 10: Best RMSE of 50 CNN-LSTM models vs. best RMSE of 50 CNN-LSTM RandDense models for both TAS, DTR, PR, and PR90. The color heatmaps to the right indicate the number of hidden layers.

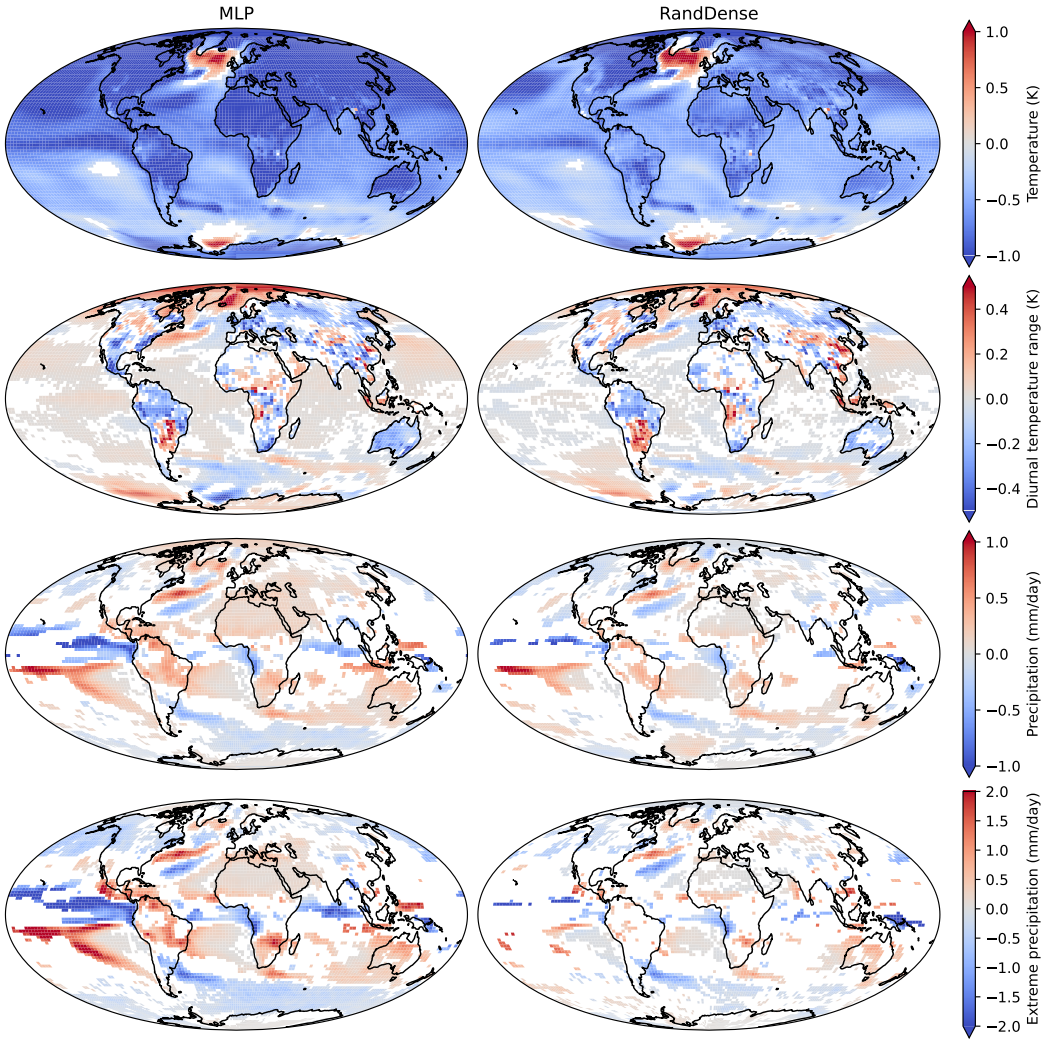


Figure 11: Mean differences between ClimateBench NorESM2 simulated target variables and the best performing MLP/RandDense emulators averaged over the test scenario between 2080-2100. Statistically insignificant differences ($p > 0.05$) are masked.

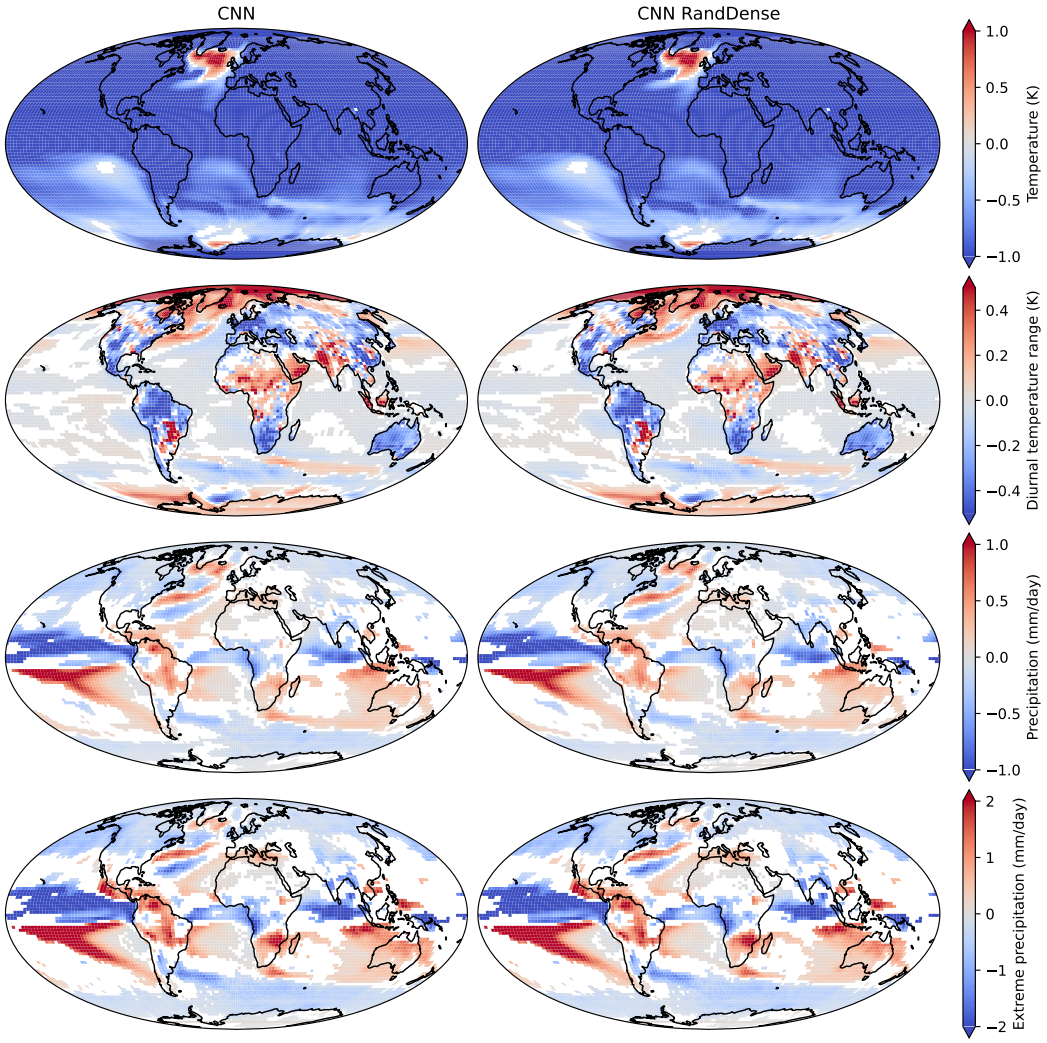


Figure 12: Mean differences between ClimateBench NorESM2 simulated target variables and the best performing CNN/CNN RandDense emulators averaged over the test scenario between 2080-2100. Statistically insignificant differences ($p > 0.05$) are masked.

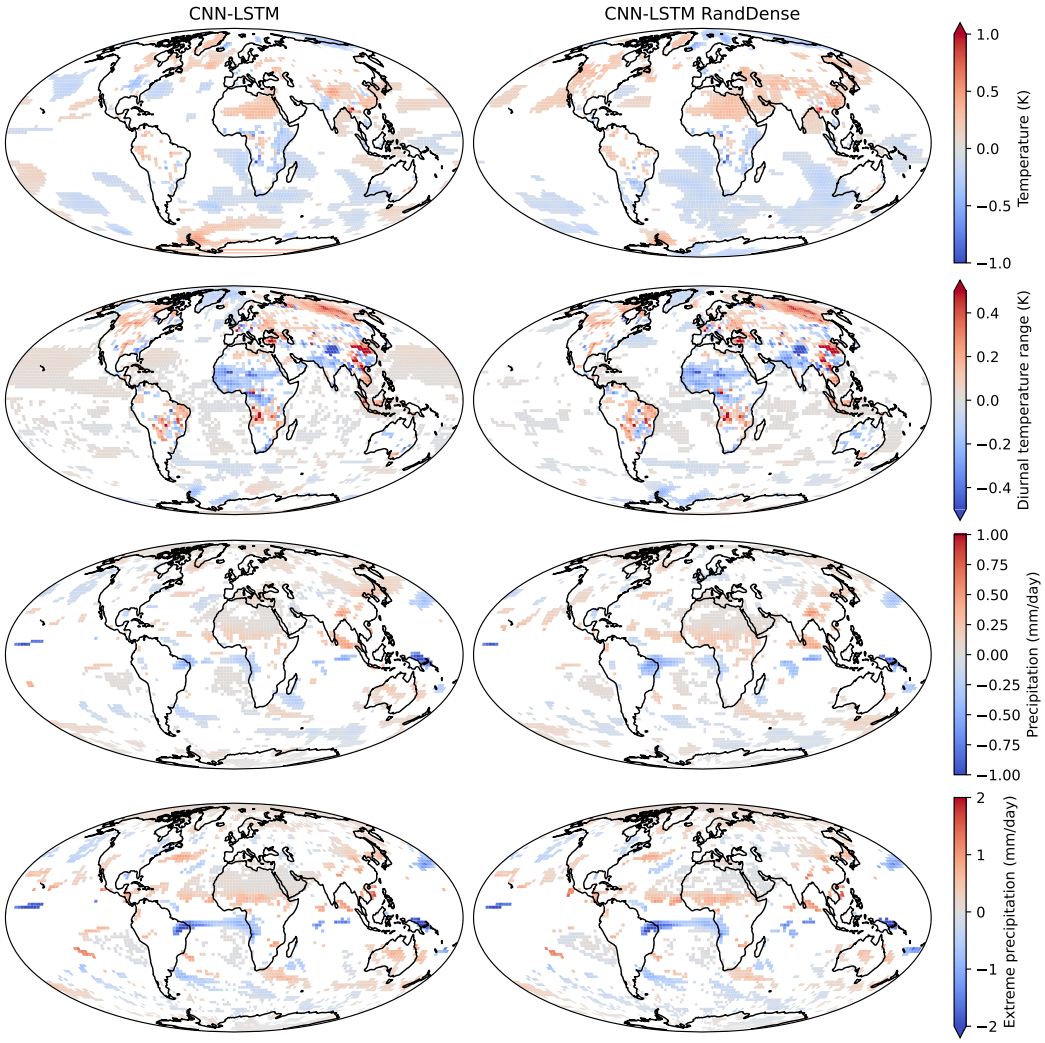


Figure 13: Mean differences between ClimateBench NorESM2 simulated target variables and the best performing CNN-LSTM/CNN-LSTM RandDense emulators averaged over the test scenario between 2080-2100. Statistically insignificant differences ($p > 0.05$) are masked.