Emulating The Effects of Climate Change with Deep Learning

Keith Kwong
khkwong@ucsd.edu

Jun Zhang
juz018@ucsd.edu

Jack Kai Lim
jklim@ucsd.edu

Duncan Watson Paris
dwatsonparris@ucsd.edu

Abstract

Climate modeling has long been limited by the amount of resources needed to use legacy built Earth system models. As such, exploration into the many different and possible emission pathways have been relegated to one-dimensional impulse response or simple pattern scaling models, neither of which are capable of accounting for finer details when emulating the Shared Socioeconomic Pathways. This paper introduces ClimateBench-Plus, an advanced emulator that builds on the deep learning models established in Watson-Parris et al. (2022) by utilizing more advanced techniques and hybrid modeling to improve upon the capabilities of their predecessors. These models are capable of emulating the response of full complexity Earth system models to forcers using a fraction of the time and resources. Moreover, these models are not only more complex and advanced than simple pattern scaling models, but also capable of predicting annual mean global distributions of temperature, diurnal temperature range, and precipitation given the emissions and concentrations of carbon dioxide, methane, sulfur dioxide, and black carbon. We will compare ClimateBench-Plus with its predecessors on their accuracy, interpretability, and robustness showcasing the potential of revolutionizing climate modeling with reduced computational demand.

Website: https://jackljk.github.io/DSC180B-website/
Code: https://github.com/jackljk/ClimateBench-Plus

1 Introduction

Climate scientists have already generated different emissions pathways that limit the increase of global temperature to under 2 celsius degrees. However, policymakers need to assess different social and economic impacts under different emission scenarios to reduce the effect of climate change and achieve the goal. The current ClimateBench machine learning models that emulate different emissions scenarios are able to generate predictions of global temperature, diurnal temperature, and precipitation(including extreme precipitation)Watson-Parris et al. (2022). Our models are based on ClimateBench models but are more advanced and efficient in terms of predicting future climate. Therefore, our ClimateBench-Plus machine learning models can help the general public and policymakers assess which emission pathways can achieve the goal.

There are many previous models such as **impulse response** models and **pattern scaling** models which both try to solve a similar problem as **ClimateBench** was trying to solve. But they all come with their disadvantages/demerits, as with the **impulse response** models, they have the ability to capture the general non-linearity of the climate system but are unable to model the regional changes in the climate. While **pattern scaling** models are heavily reliant on the scaling from the spatial distribution of for example *global mean temperature*, causing it to be unable to accurately emulate other variables such as *precipitation*, due to the strong non-linear relationship with the temperature. In addition to that, Statistic emulators have also been built for the purpose of climate modeling, however, they tend to be too regionally specific or are only capable of simple emulations. Furthermore, none of the other approaches besides ClimateBench, account for the influence from other aerosols such as black carbon and sulfur dioxide, which are also capable of affecting both regional temperature and precipitation. Expanding from ClimateBench, we are able to use it as a baseline to delve deeper into more complex modeling and tackle the problem of Climate Emulation to a deeper extent as we now have a benchmark to test and compare more complex and advanced models too.

The training data that is going to be used for the models is training from the Norwegian Earth System Model (NorESM2; Seland et al., 2020) which is generated data from simulations performed by the NorESM2 model. This was done as part of the sixth coupled model intercomparison project(CMIP6 ; Eyring et al.,2016). The data is used by the policymakers when deciding climate policies so that the emissions data that come from the NorESM2 and CMIP allow the improved ClimateBench Plus models to predict outcomes of different possible scenarios that align with the policymakers' want to reduce climate change. The data that is extracted from NorESM2 and CMIP are netcdfs which are multi-dimensional containing data on every latitude and longitude for the emissions and aerosols that we are looking at i.e Carbon Dioxide, Methane, Sulfur Dioxide, and Black Carbon and span from the 1850s to the end of 2100.

The input variables include emission data of **CO_2**, **SO_2**, **CH_4**, and **Black Carbon** by different emission scenarios data and historical data. We took the global average of input variables CO_2 and CH_4 by year. We convert the units of SO_2 and Black Carbon into Tera gram and take the global sum. Through this process, we acquire the patterns of four different input

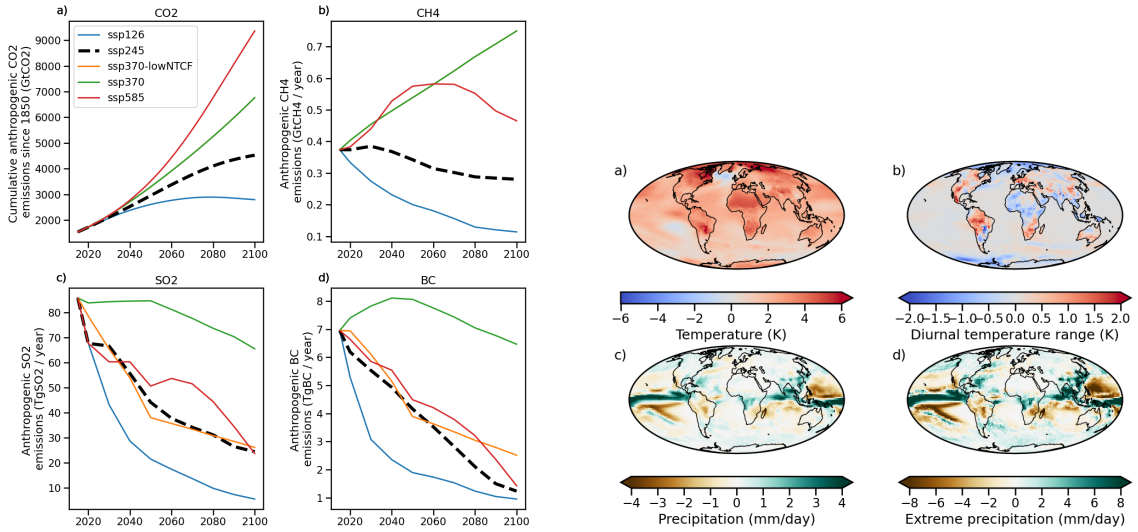

Figure 1: (a) Line Graphs of Input Variables from Multiple SSP (b) Map of Output Variables from SSP245

variables from 2020 to 2100. The output variables are temperature (TAS), Precipitation (PR), 90th percentile precipitation (PR90), and daily diurnal temperature range (DTR).

2 Methods

In ClimateBench, the paper implements 3 Deep Learning models for Climate Modeling. The models used there were a **Gaussian Process**, a **XGBoost**, and a **CNN with LSTM**. So to explore the possibilities of using more complex models, we decided to tackle the challenge of building models of similar types but adding additional layers of complexity to them.

2.1 Deep Kernel Learning Gaussian Process

For the Gaussian Process, the increase in complexity that we are adding to the original Gaussian Process used in ClimateBench, is **Deep Kernel Learning(DKL)**. DKL is a method first introduced in Wilson et al. (2016) paper on "Deep Kernel Learning" with the idea of combining the infinite expressiveness and structure of Deep Learning architectures with the non-parametric flexibility of Kernels in order to learn a feature representation of the data which best fits the probabilistic modeling of a Gaussian Process the best. This method provides the benefit of better expressiveness of the data by the feature representation/kernel for the Gaussian Process in addition to o better scalability of the kernel for larger and more complex and deep modelings.

For a review on the equations and marginal likelihood of Gaussian Processes and an in-depth look into them, see Rasmussen, Williams et al. (2006) for a comprehensive overview. For a more lighter and Visual overview of a Gaussian Process, see Görtler, Kehlbeck and

Deussen (2019).

Following the understanding of a Gaussian Process, we can look into how we can construct a kernel to encapsulate the expressive power of deep architectures and underlying patterns in data. And how to learn their properties by combining them as part of a scale-able probabilistic Gaussian Process.

The general equation for Deep Kernel Learning is given as,

$$k(x_i, x_j|\theta) \rightarrow k(g(x_i, w), g(x_j, w)|\theta, w) \tag{1}$$

$g(x_i, w)$ here is a non-linear mapping given the weights from the feed forward of the Deep Architecture, of something like a Convolutional Neural Network parameterized by the weights of w. Overall, the equation represents the kernel which is to be learnt from training the model.

For the Neural Network to learn, the loss is then back propagated via the **log marginal likelihood** for the network to learn the feature representation, which is given as,

$$\log p(\mathbf{y}|\mathbf{X}) = -\frac{1}{2}\mathbf{y}^\top (K + \sigma_n^2 I)^{-1}\mathbf{y} - \frac{1}{2}\log|K + \sigma_n^2 I| - \frac{n}{2}\log 2\pi \tag{2}$$

and the derivative of the loss is computed using the chain rule, which gives an implicit derivative with respect to the nxn covariance matrix as follows,

$$\frac{\partial \mathscr{L}}{\partial K_\gamma} = \frac{1}{2}(K_{gamma}^{-1} y y^T K_\gamma^{-1} - K_\gamma^{-1}) \tag{3}$$

that is just a very high level quick overview of the general idea of Deep Kernel Learning, for a more in-depth perspective as mentioned earlier, you can find that in Wilson et al. (2016).

Choosing a even in Deep Kernel Learning can be non-trivial due to the extreme variability of the method, if not guided properly, the method could be suboptimal. For example, a population choice for geostatistics and environmental modeling is the the Matern32, k_{M32} kernel which is defined as follows,

$$k_{M32}(x_i, x_j) = \sigma^2(1 + \sqrt{3}||x_i - x_j||)\exp(-\sqrt{3}||x_i - x_j||) \tag{4}$$

Looking at Figure 2.1 which is a simple diagram to depict how Deep Kernel Learning Works. Where a dataset X in D dimensions, as depicted by $x_1, \ldots, x_D$ is mapped through L Hidden layers which learn to describe and express the data, which then is followed by a Hidden **Parameter Layer** which has an infinite number of possible basis functions, which comes from a base kernel hyper-parameter, which in this case is the k_{M32}. In essence, the Deep Kernel produced by the neural network, produces a probabilistic mapping with the infinite number of adapted basis functions parameterized by a $\gamma = \{w, \theta\}$, where the back-propagations learned through the marginal likelihood loss of the Gaussian Processes in the output layer.

Another thing to note on the fitting and training of the Deep Kernel Learning into the Gaussian Process. We notice that the dimensions when fitting $\mathbf{SO}_2$ and **BC** are described in

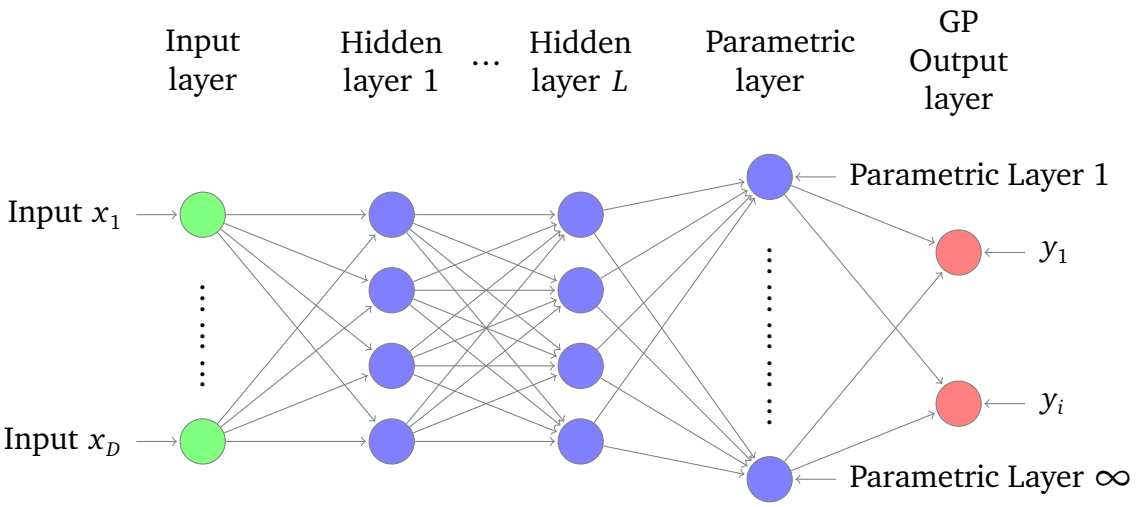

Figure 2: Deep Kernel Learning Diagram

5 dimensions. The reason why this is done can be found 5.1. So to handle this issues, we use Principle Component Analysis which explains around 96% to 98% of the variance, and built a 'pca_solver'. The Gaussian Process will then treat each principle component individually, outputting a value for each dimension, which will be solved via the 'pca_solver' for use in validation.

In initializing the kernels, we follow the approach outlined in (Watson-Parris et al. 2022), assigning a distinct kernel to each input dimension. This strategy aims to guide the Deep Neural Network (DNN) towards more favorable outcomes, preventing convergence to sub-optimal minima. Given the model's complexity, numerous hyperparameters could influence its performance and predictive capabilities. To address this, I conducted a comprehensive grid search, as described in (Liaw et al. 2018), to identify the optimal combination of neural network architecture, kernel initialization, and hyperparameters for each variable. For detailed information on this process, please refer to A.1.1.

2.2 XGBoost

The Random Forest models from ClimateBench integrated predictions from multiple decision trees. To improve the robustness of the model, we applied XGBoost techniques. The XGBoost stands for "Extreme Gradient Boosting" and is a scalable tree-boosting system. This technique combined multiple algorithms and models which is faster and more robust. The XGBoost is the technique that boosts decision trees with gradient-boosting methods. It sums up the first-order gradient and second-order gradient on each leaf so that it can evaluate the quality of the tree structures by scoring functions. The regularization terms that are applied to the regression tree penalize the complexity of models and smooth the weights to reduce overfitting. The XGBoost used "exact greedy algorithms" to find the best split. The "exact greedy algorithms" enumerated all possible splits for the features but sorted data by the feature values first which makes it work more efficiently. Furthermore, the whole sys-

tem applied the block structure. It splits the results of leaves into blocks and collects them in a parallelized style. This process helps the model optimize computation complexity (Chen and Guestrin (2016)).

The XGBoost model can be expressed as the following formula:

$$\hat{y}_i = \phi(X_i) = \sum_{k=1}^{K} f_k(X_i), f_k \in F \tag{5}$$

The $\hat{y}_i$ represents the predicted values, the X_i is the input data, and the $f_k(X_i)$ is the score for i-th data in k-th tree (Ma et al. (2021)). The process of gradient boosting is trained in an additive manner.

$$L^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t) \tag{6}$$

The predicted $\hat{y}_i^{(t)}$ is constantly add to f_t at each t-th iteration to optimize the tree (Chen and Guestrin (2016)).

2.3 Physics Informed Neural Network (PINN)

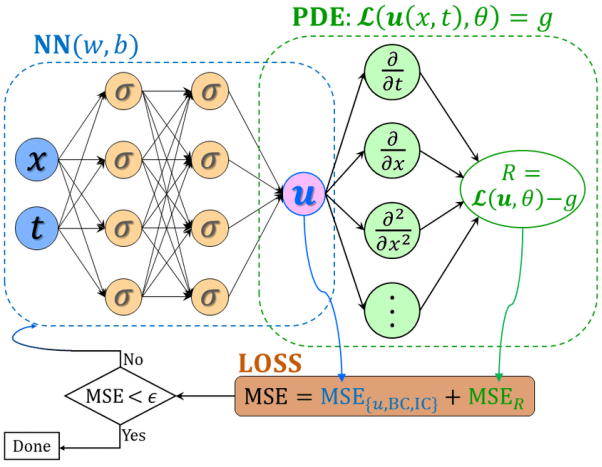

Figure 3: Building a PINN

As mentioned in the original ClimateBench paper, one of the potential methods of improving the robustness and generalizability of ML emulators is through marrying them with physical constraints brought on by known physical equations. To further explore this area, we draw upon the field of scientific machine learning by attempting to improve the original CNN architecture by changing the architecture to one of a physics informed neural network (PINN). Currently, there is no one known set of equations that are used to model the relation between the aerosols being released and the temperature and precipitation, but previous work done on Finite amplitude Impulse Response (FaIR) models (Leach et al. 2021) have proposed a set of ordinary differential equations that they have proven to be adequate in modeling the relation between emissions of greenhouse gases and mean climate

response. By incorporating these equations into the existing loss function of the original CNN architecture outlined in ClimateBench, the resulting PINN will produce results that are more robust and physically constrained.

The differential equation seen below was implemented into a custom loss function for the baseline CNN architecture. In addition the the standard mean squared error used in training, an additional error value that is calculated by comparing the PINN prediction to the expected physical response as given by the FaIRv2.0 equation pictured below in Figure 4. For the constant values q_j and d_j, values of 0.3 and 10 were chosen to represent the aggregate of all the different thermal boxes. Forcing constants of $f_1 = 4.57$, $f_2 = 0$, and $f_3 = 0.086$ were chosen respectively as averages of the values present in Leach et al. (2021) and a starting coefficient of 598 gigatons of greenhouse gases was chosen as the pre-industrial amount.

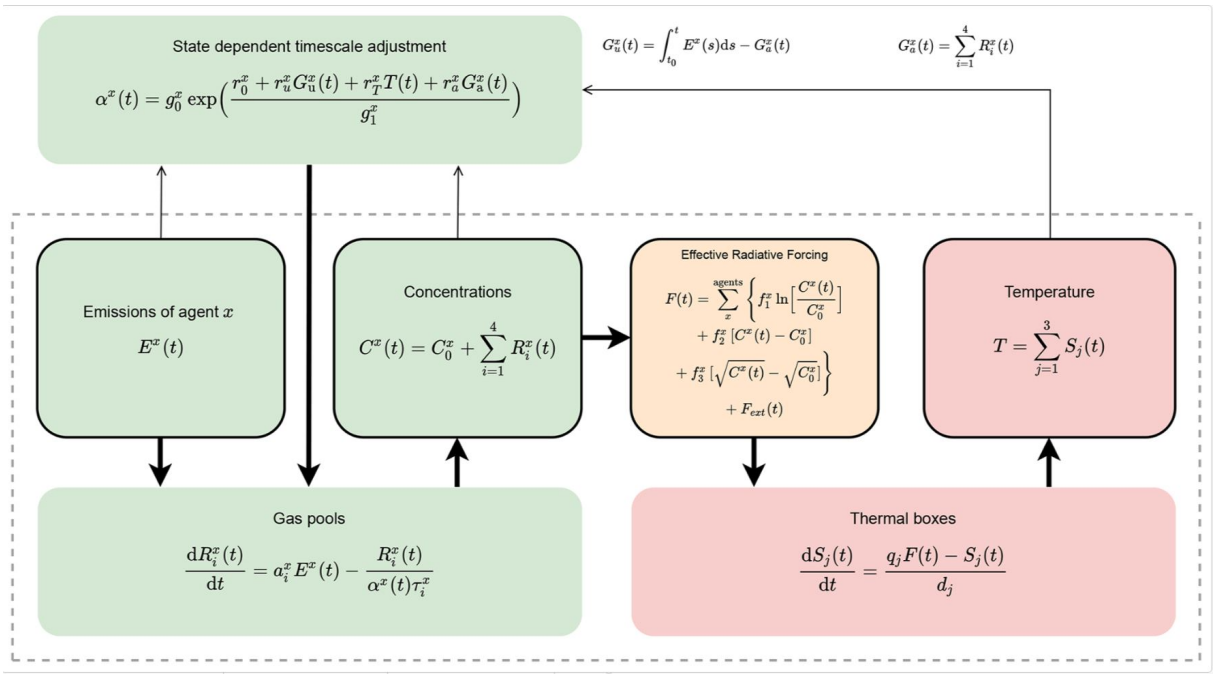

Figure 4: FaIRv2.0 Impulse Response Model

3 Evaluation Metric

Following Watson-Parris et al. (2022) we will also be using the $NRMSE_t$ from the paper which is defined as follows,

$$NRMSE_s = \sqrt{\langle (|x_{i,j,t}|_t - |y_{i,j,t,n}|_{n,t})^2 \rangle} / |\langle y_{i,j} \rangle|_{t,n} \tag{7}$$

$$NRMSE_g = \sqrt{|(\langle x_{i,j,t} \rangle - \langle |y_{i,j,t,n}|_n \rangle)^2|_t} / |\langle y_{i,j} \rangle|_{t,n} \tag{8}$$

$$NRMSE_t = NRMSE_s + \alpha \times NRMSE_g \tag{9}$$

Where $NRMSE_s$ is the global mean root-mean squared error, and $NRMSE_g$ is $NRMSE$ in the global mean. The equation also includes a weighing function to take the decreasing grid-cell area towards the north and south poles which is defined as follows,

$$\langle x_{i,j} \rangle = \frac{1}{N_{lat} N_{lon}} \sum_{i}^{N_{lat}} \sum_{i}^{N_{lon}} \cos(lat(i)) x_{i,j} \tag{10}$$

and the co-efficient α is chosen to be 5 from the ClimateBench paper, in order to provide equal weightage between the measures.

The reason, we are using this evaluation metric is to be able to compare directly the performances of our models against the models from the paper. And using the same evaluation metric will give us the most 1-1 comparison.

4 Results

As a comparison between the different models, we are going to use the evaluation metrics listed in 3 as a way to for direct comparisons between both models.

4.1 Deep Kernel Learning Results

Looking at the NRMSE values in table 1, we can see that comparing the $NRMSE_t$ Deep Kernel Learning performs better overall with a large improvement for the Diurnal Temperature Range. The improvement for **TAS** is however negligible to random chance, the reason for this is that the baseline Gaussian Process was already able to capture its variance well leaving little room for improvement.

In addition, the Deep Kernel Learning Models were also able to capture the spatial variance and distribution which can be seen in figure A 1.

Table 1: NRMSE Comparison Table for Deep Kernel Learning and GP

Model	Deep Kernel Learning			GP from Reproduction			Difference		
Variable	Global	Spatial	Total	Global	Spatial	Total	Global	Spatial	Total
TAS	0.0458	0.0752	**0.3044**	0.0437	0.0906	0.3089	0.0021	-0.0154	-0.0045
DTR	1.5602	8.4860	**16.2872**	2.6495	9.1950	22.4425	-1.0893	-0.7090	-6.1553
PR	0.3169	2.4195	**4.0041**	0.3784	2.3301	4.2223	-0.0615	0.0894	-0.2182
PR90	0.4133	2.4919	**4.5582**	0.3955	2.6048	4.5821	0.0178	-0.1129	-0.0239

4.2 XGBoost Learning Results

According to the table, we can observe that XGBoost overall has better scores than the baseline model of Random Forest. Especially for the precipitation and the 90th percentile

Table 2: NRMSE Comparison Table for XGBoost and Random Forest

Model	XGBoost			Random Forest from Reproduction			Difference		
Variable	Global	Spatial	Total	Global	Spatial	Total	Global	Spatial	Total
TAS	0.1367	0.2273	0.9107	0.3979	0.4525	2.4419	-0.3044	-0.2746	-1.7965
DTR	2.8948	10.5215	24.9953	2.7712	13.1610	27.0172	0.1213	-1.9044	-1.2981
PR	0.3851	4.0115	5.9368	0.9235	5.5991	10.2167	-0.5828	-2.2157	-5.1300
PR90	0.3745	4.7913	6.6639	0.9955	6.7342	11.7122	-0.4744	-0.8491	-3.2214

precipitation, the score is significantly lower than the baseline model. The scores of temperature and diurnal daily temperature also improve from the random forest model, although the Global score for diurnal daily temperature is slightly higher.

4.3 PINN Results

The values in Table 3 compares the performances of the Physically informed neural network to that of the CNN baseline. Overall, there wasn't any significant improvements in any of the four variables being predicted, with the PINN performing marginally better for the 90th percentile precipitation and diurnal temperature range.

Table 3: NRMSE Comparison Table for PINN and CNN

Model	PINN			CNN from Reproduction			Difference		
Variable	Global	Spatial	Total	Global	Spatial	Total	Global	Spatial	Total
TAS	0.0428	0.1021	0.3164	0.0440	0.0966	0.3167	-0.0012	0.0055	-0.0003
DTR	0.9371	8.3310	13.0166	1.2263	8.4313	14.5632	-0.2892	-0.1003	-1.5466
PR	0.1998	2.1588	3.1582	0.1776	2.2642	3.1526	0.0222	-0.1054	0.0056
PR90	0.3159	2.7057	4.2857	0.3726	2.5163	4.3796	-0.0567	0.1894	-0.0939

5 Discussion

5.1 Aerosol Dimension Differences

As we can see from the cleaned data, for $\mathbf{CO_2}$ and $\mathbf{CH_4}$ there is only 1 dimension and for $\mathbf{SO_2}$ and **BC** has 5 dimensions. The reason for this is because $\mathbf{SO_2}$ and **BC** would only stay in the atmosphere for a few days to a week. While $\mathbf{CH_4}$ would stay in the atmosphere for years and millennia for $\mathbf{C0_2}$. This is why they have more dimensions, as they would be out of our atmosphere before they get mixed completely into our environment and have a global effect. Hence, the dimensions give us the distributions of $\mathbf{SO_2}$ and **BC** to account for where they are being emitted.

5.2 DKL vs GP variation

Looking at the $NRMSE_t$ results table 1 between the Deep Kernel Learning Model against the Baseline Gaussian Process, we can see that the Deep Kernel Learning Regressor performed better overall. In general, both models perform extremely well in capturing the spatial variance as compared to the NorESM2 model which is clear in figure A 1. However, in terms of the temporal mean, looking at figure A 2, for the temperature, it is clear that both models were able to capture the trends very well, the variable with a little more noticeable difference and improvements are the diurnal temperature range and Precipitation.

Comparing the Deep Kernel Learning model and the Gaussian Process, we can see for diurnal temperature range that the Gaussian Process was only mildly able to capture the trend as shows a shallow increase overtime but fails to truly follow the overall trend seen in the NorESM2 model. While the Deep Kernel Learning model was able to increase relatively well with the trend, and seems to overshoot the predictions and then dips down drastically, which could be due to the the neural networks lack of ability to store and capture temporal data well, and causing an inconsistency as the trend travels further in the time dimension for the more complicated variable of Diurnal Temperature Range. Looking at precipitation, both the Deep Kernel Learning and the Gaussian Process were able to learn the upward trend well, but it looks like the Deep Kernel Learning was able to capture the mean better compared to the Gaussian Process, as overall the Gaussian Process underestimated through all the predicted values, as opposed to the Deep Kernel Learning which as compared to the Gaussian Process was able to capture the mean and stay relatively centered compared to NorESM2 time series plot.

5.3 XGBoost vs Random Forest

By comparing the results between these models, we can observe that most of the scores are lower than baseline models from reproduction. Due to the regularization and hyperparameters tunning, the XGBoost is capable of reducing the complexity of models and overfitting. Additionally, the gradient boosting greatly improves the accuracy of the model when each iteration compares the residual error with the previous model (Chen and Guestrin (2016)). By looking at the improvement in NRSEM scores, we know that both variables precipitation and 90th percentile precipitation are being predicted more accurately. However, there are still some shortcomings in training the XGBoost model. For example, the process of hyperparameters tunning requires a longer time than the Random Forest model. Despite Chen mentioning that XGBoost applied the ”Cache-aware algorithm”, which puts each leaf into blocks and collects the results parallelized, the training process still requires large memory space (Chen and Guestrin (2016)). This could be due to the complexity and multi-dimensionality of the climate data.

5.4 PINN vs CNN variation

In the results shown in table 3 above, the total difference in the $NRMSE_t$ shows that the overall performance of both the PINN and baseline CNN are pretty much identical. As a whole, however, the PINN can be said to have performed better than the baseline CNN in terms of $NRMSE_t$, with most of the improvements coming from the side of $NRMSE_g$.

The lack of difference between the two models isn't all too surprising of an outcome, given the underlying architecture of the model remains the same for both the baseline CNN and PINN, with the only difference being the loss equation that the two are trained on. The baseline CNN used the standard mean squared error function while the PINN has an extra physical constant added on based on what the impulse model as dictated in FaIRv2.0 (Leach et al. (2021)) would have produced from the same training step. While the lack of improvement from the PINN is slightly disappointing, the baseline CNN was already one of the top performers in Watson-Parris et al. (2022), with its worst variable being the diurnal temperature range, so with the biggest and most notable improvement being in that variable shows that improvements, however marginal, can still be made. The original impulse model put out by FaIRv2.0 (Leach et al. (2021)) also only applied to temperature, so while we still applied it to the precipitation variables, the lack of effect does make sense.

5.5 Future Work

5.5.1 Deep Kernel learning

For future endeavors toward further exploration on the deep kernel learning pathway, one promising direction involves investigating the implementation of Neural Processes (NPs) on climate data. Neural Processes, akin to Deep Kernel Learning, represent a hybrid modeling approach that synergizes the strengths of Neural Networks with Gaussian Processes. However, unlike Deep Kernel Learning, Neural Processes implement a different structure using context points which are input-output pairs that get parameterized and is then modeled to fit onto the targets. More on Neural Processes - Garnelo et al. (2018).

Another potential path for further work would be to use GPytorch Gardner et al. (2018) to implement the Exact Deep Kernel Learning where and example can also be seen here EXACT DKL w/ KISS-GP. I had attempted to implement the Deep Kernel Learning Regression using a this layout but was unsuccessful due to the high dimensional output of the task.

5.5.2 XGBoost

The XGBoost technique can surpass the baseline model from ClimateBench. This paper proves that XGBoost is scalable for predicting a large and complex dataset like climate data. According to "XGBoost-based method for flash flood risk assessment", the XGBoost also proves its capability when it comes to predicting regional climate data (Ma et al. (2021)). The higher accuracy rates of XGBoost have made this model popular in recent years. We

believe that XGBoost has huge potential in the field of climate science for predicting other climate variables like humidity, snow, and wind. However, there is still space for improving the efficiency of XGBoost for large and complex datasets. For future work, we believe that we can dive deeper to improve the efficiency of XGBoost for complex datasets with multi-dimensions.

5.5.3 PINN

While our work with the PINN ultimately ended up with little improvement to the baseline CNN dictated in ClimateBench, there are still many other ways in which physical laws and constraints could be used to better improve a neural network approach to climate emulation. For our approach specifically, we incorporated the constants and equations set by FaIRv2.0 (Leach et al. (2021)), however, we were comparing our results to the results of the NorESM2 model, which FaIR had not calculated constants for in their paper. For our physical equations constants then, we used a simple average of all the constants that FaIR had calculated. If instead we could replicate FaIR's methodology to find constants specifically for the NorESM2 model, the PINN could theoretically do better. Besides that, looking for other differential equations that better show the relationship between precipitation and temperature with greenhouse gas emissions would also result in a better PINN.

6 Conclusion

To move forward from ClimateBench, we aim to extend the three baseline models Gaussian Process, Random Forest, and CNN to predict various climate data by given aerosol emission input data. The Deep Kernel Learning improves the performance of Gaussian processes by using the expressiveness of a Neural Network and combining it with the probabilistic modeling of a Gaussian Process. The XGBoost is a model that integrates the predictions from multiple decision trees like Random Forest, but it combines multiple machine learning algorithms to boost the decision trees. The PINN incorporates physical constraints brought on by known physical equations, thus contributing to the overall best performance among all models. By looking at the predictions, policymakers and the general public can assess which emission pathway is most suitable for achieving the goal of reducing global warming. Moreover, they also can learn about the potential of Deep Learning models when it comes to emulating the effect of climate change. In the future field of climate science, various techniques and algorithms of Deep Learning models are waiting for people to explore and research. Therefore, people can learn which emission path is most suitable for reducing the effect of climate change, and create a better environment for all.

References

Chen, Tianqi, and Carlos Guestrin. 2016. "XGBoost: A Scalable Tree Boosting System." In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM. [Link]

Gardner, Jacob R, Geoff Pleiss, David Bindel, Kilian Q Weinberger, and Andrew Gordon Wilson. 2018. "GPyTorch: Blackbox Matrix-Matrix Gaussian Process Inference with GPU Acceleration." In *Advances in Neural Information Processing Systems*.

Garnelo, Marta, Jonathan Schwarz, Dan Rosenbaum, Fabio Viola, Danilo J. Rezende, S. M. Ali Eslami, and Yee Whye Teh. 2018. "Neural Processes." *CoRR* abs/1807.01622 . [Link]

Görtler, Jochen, Rebecca Kehlbeck, and Oliver Deussen. 2019. "A visual exploration of gaussian processes." *Distill* 4(4), p. e17

Leach, Nicholas J., Stuart Jenkins, Zebedee Nicholls, Christopher J. Smith, John Lynch, Michelle Cain, Tristram Walsh, Bill Wu, Junichi Tsutsui, and Myles R. Allen. 2021. "Fairv2.0.0: A generalized impulse response model for climate uncertainty and future scenario exploration." *Geoscientific Model Development* 14(5), p. 3007–3036. [Link]

Liaw, Richard, Eric Liang, Robert Nishihara, Philipp Moritz, Joseph E Gonzalez, and Ion Stoica. 2018. "Tune: A Research Platform for Distributed Model Selection and Training." *arXiv preprint arXiv:1807.05118*

Ma, Meihong, Gang Zhao, Bingshun He, Qing Li, Haoyue Dong, Shenggang Wang, and Zhongliang Wang. 2021. "XGBoost-based method for flash flood risk assessment." *Journal of Hydrology* 598, p. 126382. [Link]

Matthews, Alexander G. de G., Mark van der Wilk, Tom Nickson, Keisuke. Fujii, Alexis Boukouvalas, Pablo León-Villagrá, Zoubin Ghahramani, and James Hensman. 2017. "GPflow: A Gaussian process library using TensorFlow." *Journal of Machine Learning Research* 18(40): 1–6. [Link]

Rasmussen, Carl Edward, Christopher KI Williams et al. 2006. *Gaussian processes for machine learning*. 1 Springer

Watson-Parris, Duncan, Yuhan Rao, Dirk Olivié, Øyvind Seland, Peer Nowack, Gustau Camps-Valls, Philip Stier, Shahine Bouabid, Maura Dewey, Emilie Fons et al. 2022. "ClimateBench v1. 0: A Benchmark for Data-Driven Climate Projections." *Journal of Advances in Modeling Earth Systems* 14(10), p. e2021MS002954

Wilson, Andrew Gordon, Zhiting Hu, Ruslan Salakhutdinov, and Eric P. Xing. 2016. "Deep Kernel Learning." In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*. Cadiz, Spain PMLR. [Link]

Appendices

A.1 Deep Kernel Learning Training Details

A.1.1 Deep Kernel Learning Structure

After data pre-processing, first we define the custom kernel function which will take in a 'feature_extractor' which is the output from the neural network.

```
class DeepKernel(gpflow.kernels.Kernel):
def __init__(self, feature_extractor, base_kernel, input_dim):
    super().__init__()
    self.feature_extractor = feature_extractor
    self.base_kernel = base_kernel
    self.input_dim = input_dim

def K(self, X, X2=None):
    # Transform X and X2 using the neural network
    X_transformed = self.feature_extractor(X)
    X2_transformed = self.feature_extractor(X2) if X2 is not None else X2
    # Compute the kernel using the transformed inputs
    return self.base_kernel(X_transformed, X2_transformed)

def K_diag(self, X):
    X_transformed = self.feature_extractor(X)
    return self.base_kernel.K_diag(X_transformed)
```

Then during the train, we initialize the 'DeepKernel' with an arbitrary kernel built in to GPFlow to guide the Deep Kernel Learning process in the right direction. Which is then connected to a GPR (Gaussian Process Regression) from GPFlow. For more information about GPFlow refer to Matthews et al. (2017).

A.1.2 Best Model Hyperparameters

Due to randomness there is the possibility of other hyper-parameter combinations being slightly better than the ones listed here, but overall these gave me the best results for each variable, as seen in A 2.

Table A 1: Deep Kernel Learning Hyper-parameters

Parameters	Temperature	Diurnal Temperature Range	Precipitation	90th Precipitation
Kernel	Matern52	Matern52	Matern52	Squared Exponential
Learning Rate	0.01	0.001	0.001	0.01
Input Dimensions	256	256	128	128
Output Dimensions	36	60	24	12
Activation	Sigmoid	Sigmoid	Sigmoid	Sigmoid
Dropout Probability	0.5	0.5	0.5	0.5
Dropout	False	False	False	True
Batch Normalization	False	True	False	False
Active Dimensions	1	2	2	1

A.2 Additional Figures

A.2.1 Deep Kernel Learning Geographic Mean Plot

In the plot A 1 shows the posterior mean from the Gaussian Process from the Deep Kernel learning model, Spatial variance by taking the mean of the outputs between years 2015-2100 comparing it to the NorESM2 Model and the Baseline Gaussian Process following our Climate Bench reproduction.

A.2.2 Deep Kernel Learning Time Series

In figure A 2 shows the plot for the global mean average timeseries, weighing the small grid sizes for the poles over the years 2015-2100. Comparing it with the NorESM2 Model and the Model from the reproduction.

A.2.3 Deep Kernel Learning Time Series with uncertainty

In figure A 3 we can see the timeseries plotted with the uncertainty that is obtained along with the Gaussian Process. One interesting to not about this, is that the uncertainty shows

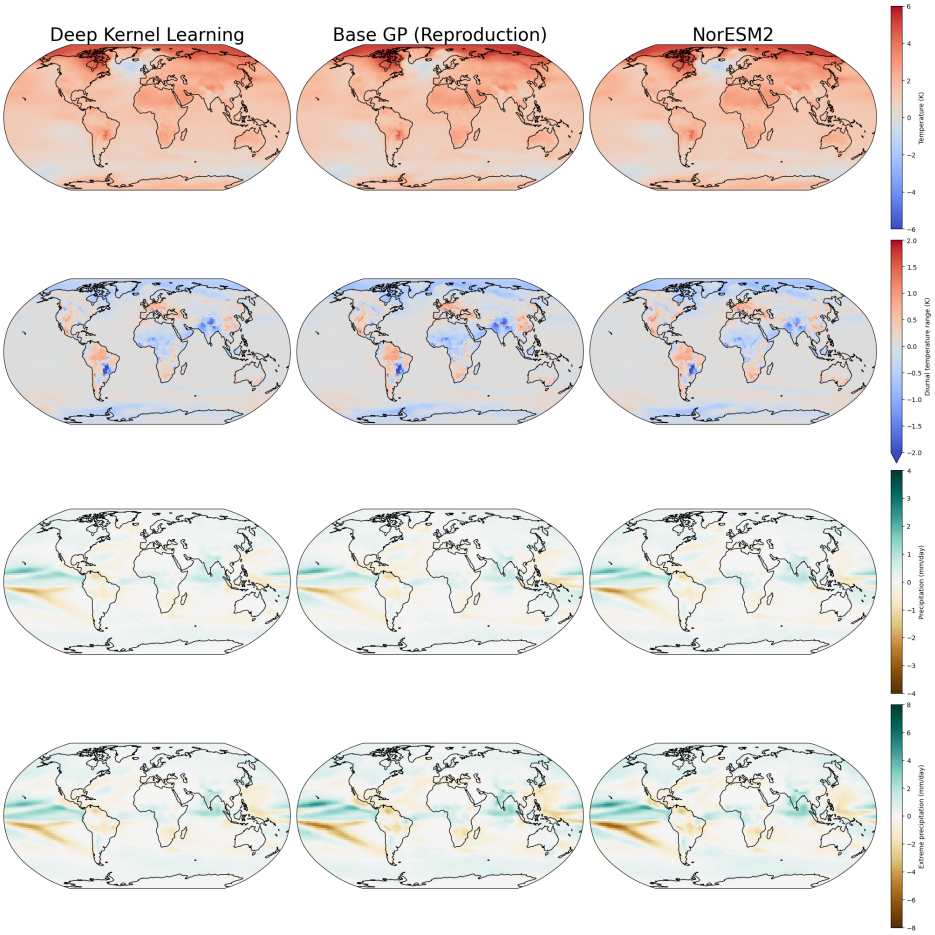

Figure A 1: Deep Kernel Learning Geo Plots Mean 2015-2100

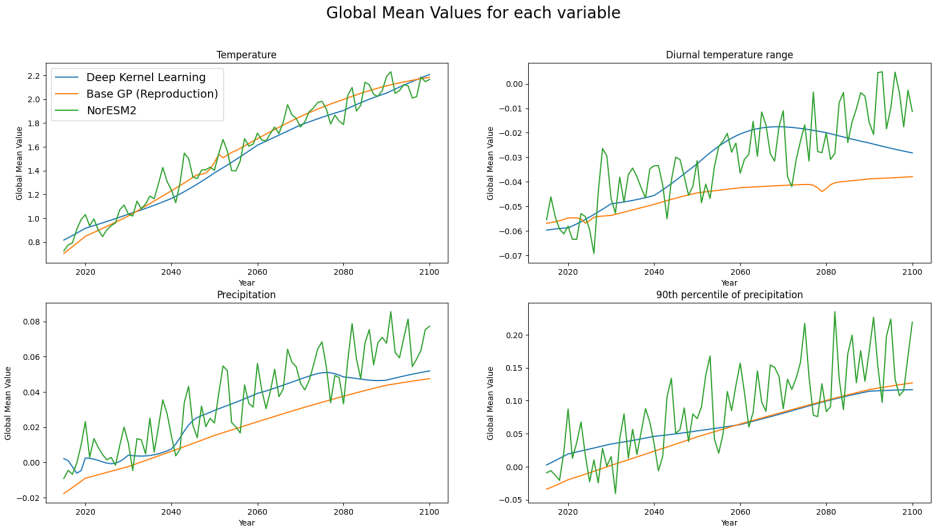

Figure A 2: Deep Kernel Learning Global Mean Time Series comparisons

that for the **Temperature** only the uncertain not only covers all of the NorESM2 variations but also has a relatively small variance and seems to capture the underlying patters really well. However, when looking at the other models, we can see that the variance still covers all of the NorESM2 models variations however, it is clear that not much can be interpreted from it as the variance seems to be very large making it redundant for interpretations.

A.3 XGBoost Randmized Search

```
param_grid1 = {
    'learning_rate': np.linspace(0.01, 0.2, 10),
    'n_estimators': [int(x) for x in np.linspace(start=100, stop=300, num=5)],
    'max_depth': [int(x) for x in np.linspace(5, 20, num=4)],
    'min_child_weight': [int(x) for x in np.linspace(1, 10, num=5)],
    'subsample': np.linspace(0.8, 1.0, 5),
    'colsample_bytree': np.linspace(0.4, 1.0, 5),
    'gamma': np.linspace(0, 0.5, 6),
}
random_search = RandomizedSearchCV(
    estimator=xgb_tas,
    param_distributions=param_grid1,
    n_iter=10,  # Number of parameter settings that are sampled
    scoring='neg_mean_squared_error',  # Use negative RMSE as scoring metric
    cv=5,  # Number of cross-validation folds
    verbose=2,  # Print progress information
    n_jobs=-1  # Use all available CPU cores
)
```

A.4 XGBoost Best Parameters

A.5 XGBoost Predicted Results

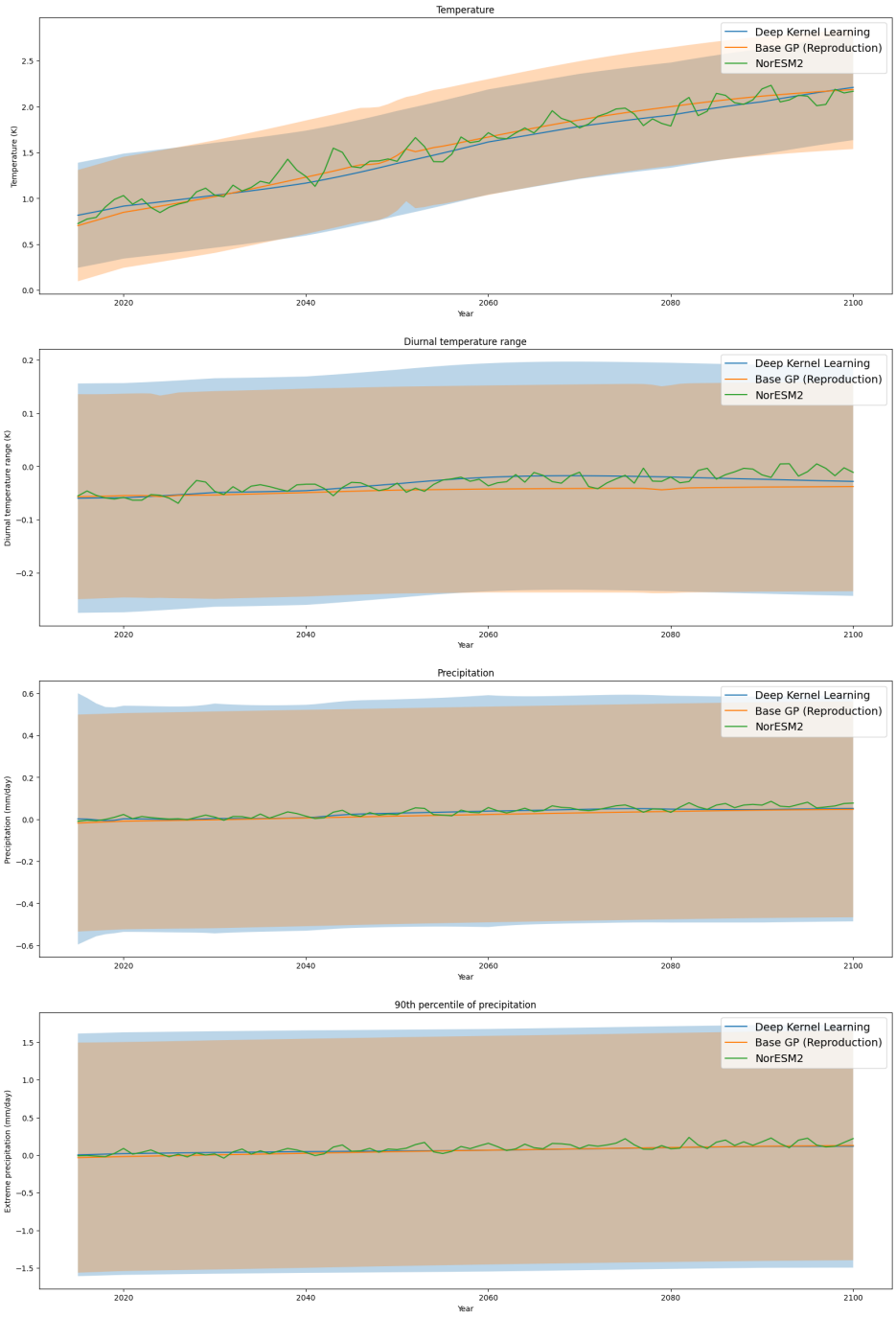

Figure A 3: Deep Kernel Learning Global Mean Time series with Uncertainty comparisons

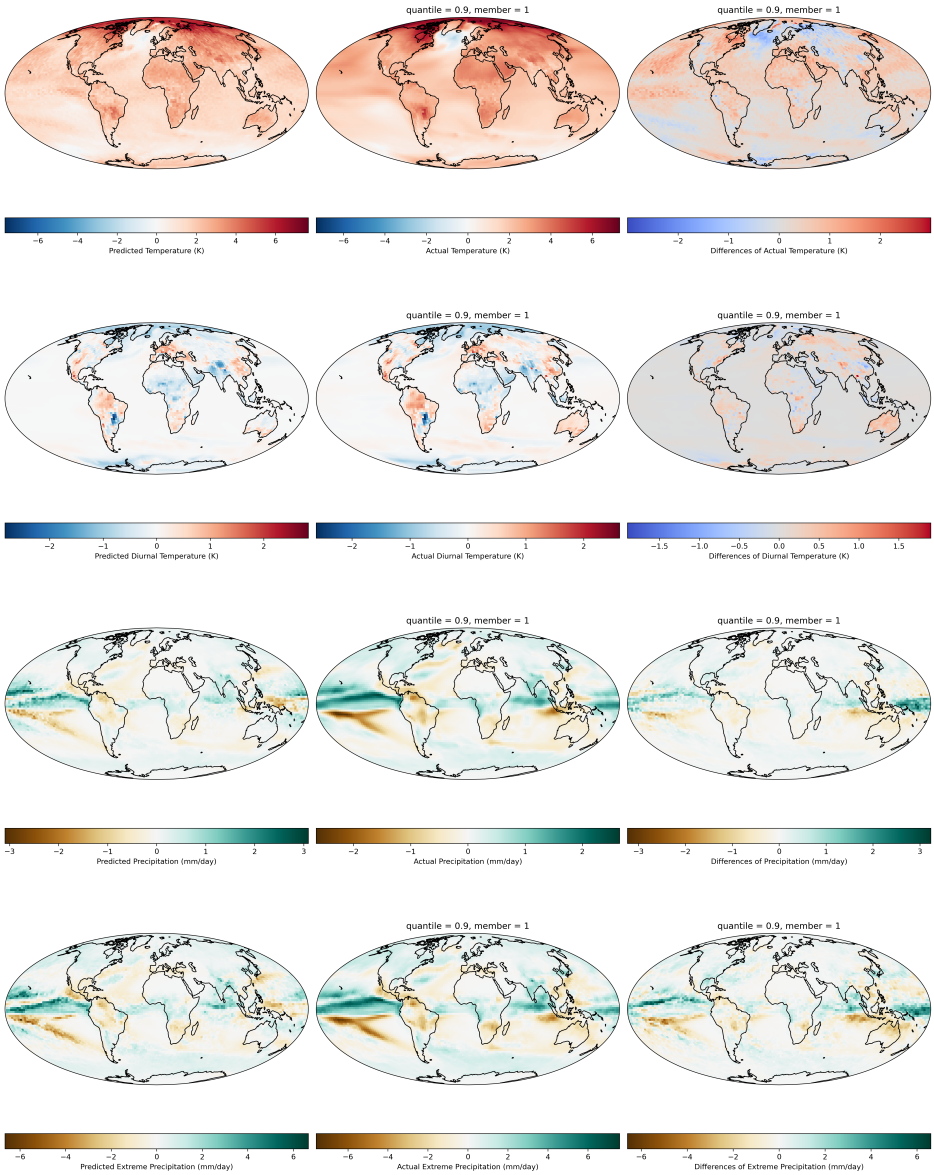

Figure A 4: XGBoost Predicted Results

Table A 2: XGBoost Best Parameters for Regressor

Parameters	Temperature	Diurnal Temperature Range	Precipitation	90th Precipitation
subsample	0.950	1.000	1.000	1.000
n_estimators	100	300	160	200
min_child_weight	7	5	7	10
max_depth	15	10	15	15
learning_rate	0.073	0.052	0.052	0.010
gamma	0.300	0.500	0.400	0.300
colsample_bytree	0.850	1.000	0.550	0.700