Investigating short-term climate forecasts with surrogate modelling

Andrei Alexandru, Zainab Imam Attahiru, Marwan Salam, Karolis Spukas

1 Introduction

The motivation for climate modeling stems from an interest in better understanding and predicting climate behavior. Climate models are mathematical representations of major climate system components and their interactions. Global Climate Models (GCMs) track physical atmospheric and oceanic processes: capturing the flow of air and water, and the transfer of heat. A more recent subset of GCMs is Earth System Models (ESMs) which additionally capture biogeochemical cycles and their interactions with the governing climate. They do not treat atmospheric composition as fixed, but rather, simulate its characteristics over time, including the effect of changing climate conditions and human activity.

The following question inevitably poses itself: if these simulations give outputs consistent with observations, and can forecast reliably with a high degree of specificity, where's the catch?

In fact, a great limitation of these systems is their computationally expensive and time-consuming nature. And naturally, the more robust the model, the more expensive it will be and longer it will take to run.

Herein lies the increased attention paid to surrogate modeling (or emulation) for climate science. Using statistical processes like Artificial Neural Networks (ANNs) or Gaussian processes (GPs), these surrogate models can learn to map inputs to their respective simulation output. Once these models are fit, one can query them instead of the original simulation, thereby obtaining results more quickly and cheaply – and if modelled correctly, with more or less equivalent precision.

Inspired by the prospect of quickly and cheaply forecasting climate variables by means of a surrogate model, we turned to existing research to better shape our project.

2 Related Work

2.1 Reference Paper

The starting point we chose is [Web+20]. In this paper, the authors introduce a Convolutional Neural Network (CNN) architecture as a surrogate for monthly precipitation output from the 1pctCO2 run (the CO2 concentration increases by 1% per year) simulated by the second-generation Canadian Earth System Model (CanESM2). Their motivation for using CNNs is their demonstrated ability to understand image content, which they view as a reliable indicator for the spatial component involved in precipitation forecasting. Further, they use a sliding window of 5 years, to capture long and short-term trends. Their CNN is trained on historical data from 1850 to 1947 and validated on 1948 - 1968. It is tested against the simulator's predictions for 1969 - 1989. Their model achieves low error on the test set with a mean precipitation difference of -0.237 mm d^{-1} and mean percent error of -13.22%.

Noticeably, however, their model under predicts near the "equator, in the mid-latitude storm tracks, and in areas associated with monsoon precipitation." [Web+20] All of these areas experience extreme precipitation events and the MSE loss function they use does not capture outliers, thereby hindering their model's ability to properly forecast in these regions.

To that effect, we decided to use this paper as a motivation for further investigating surrogate models for the task of predicting monthly mean precipitation. We chose to use a Gaussian Process emulator given the low dimensionality of the problem, and the Gaussian Process' ability to quantify uncertainty.

2.2 Wider Research

Gaussian processes don't seem to be widely applied to precipitation modelling. This is perhaps due to high spatial and temporal variability of precipitation, causing statistical methods to fail at high frequency scales. The temporal aggregation of precipitation across seasonal, daily, monthly, and regional scales results in a distribution that is not normal, making it harder to calculate estimates. However, attempts have been made that validate the feasibility of using GPs to fit precipitation data. [Wat+21] demonstrate the robustness of their open-source emulation framework by modelling precipitation using 1-month daily simulated data from a cloud resolving climate model. Although they used random forests to implement this example, the predictors and methodology are based on and earlier paper that used Gaussian process to emulate clouds and cloud-related processes [Gla+19]. The data pre-processing techniques applied in both papers involve dimensionality reduction, thus introducing uncertainty into the emulation. One of the papers we surveyed deals with uncertainty in precipitation estimates but in the context of the output of climate models rather than emulation. Using a region exhibiting precipitation bias that is not corrected by downscaled climate models, the authors applied a non-linear bias correction to improve the precipitation estimates using observed data [Ban+19].

A common trend in these papers is the need for expert domain knowledge of the physical processes that affect precipitation in specific climates and ecosystems. The data pre-processing and model choices appear to be informed by this knowledge. In the *ESEm* [Wat+21] paper, hours not associated with the formation of shallow convective clouds were filtered as studies have shown that they have less effect on precipitation within the emulated region. Of course, not all attempts to emulate precipitation are grounded in an in-depth understanding of the complex climate interaction. [OL18] propose the use of echo state networks (ESNs), multi-gene genetic programming models (MGGP), and support vector regression (SVR) to forecast rainfall. Despite using observed data from over the span of a 43 year time period and applying a handful of data pre-processing (wavelet transform amongst others), the extrapolation limit of the best model, ESN was 6 months. MGGP fared much worse, although this could be due to the inability of empirical mathematical models to adequately characterise the complexity of rainfall. The reason for the ESNs better performance is most likely due to its recurrent neural network architecture.Rainfall forecasting uses LSTM architecture to a great extent, as it is a time-series problem and LSTMs capture these sequential interactions well. For instance, [Zha+20] use correlation analysis between rainfall and control forecast meteorological factors to select eight amongst them. They divide samples into four categories using K-means clustering, which in turn are combined to feed the LSTM-based model to make its predictions. Further, statistical downscaling methods provide highly accurate precipitation estimates, much like [Wan+20]), who develop an RNN model that performs statistical downscaling on temperature and precipitation to improve the accuracy of hydrological models. [Cas+14]) introduced an approach to emulate temperature and precipitation using a statistical model fit to a small set of training runs. They express both of these variables as functions of "the past trajectory of atmospheric CO2 concentrations', to which the statistical model is fit. When they compare their emulator against the CCSM3 model on precipitation in the equatorial Pacific west region, their pointwise 95% confidence bounds are very narrow, suggesting that the emulator can capture the mean trend with very high precision. This earlier study is interesting because it couples emulation with uncertainty quantification, in the form of confidence bands for the estimated regression, something the neural network models do not capture.

A reaffirming literature on the plausibility of our investigation is the use convolutional Gaussian processes to extrapolate temperature and precipitation on a seasonal scale [WZV21].

3 Simulation data

We consider simulations for three different variables under different scenarios to see the potential for a climate forecasting emulator. We would like to investigate the changes in variables in recent years, some of which will be due to climate change. The full list of variables of the simulator used in the simulations can be found at this website[1].

3.1 Historic monthly-mean precipitation

In [Web+20], a convolutional neural network (CNN) is used for short-term precipitation forecasting. In line with this, we first consider emulating historic precipitation (years 1850-2014). We visualise simulated precipitation over the last 100 years (1914-2014) as time series data for a specific location within the UK in Figure 1 (left). At first glance, we cannot see any long term trend in the data and the mean seems stable. There also does not seem to be a particularly clear seasonal trend and the data is prone to outliers. Note that different locations around the world have different conditions, however we have not noticed huge differences between various locations within various climate zones.

3.2 Historic monthly-mean surface snow thickness

Next, we consider emulating historic surface snow thickness again for years 1850–2014. The graph can be found in Figure 1 (center). This time, there is a clear seasonal pattern, however there is no long term pattern - data points seem to be centered around mean. This is quite unexpected, however could be explained by the fact that the data only lasts until 2014.

3.3 Future monthly-mean near-surface air temperature

Lastly, we also consider emulating future surface air temperature predictions under emissions-driven future scenario simulation for years 2015-2100 (RCP8.5 based on SSP5). Data for this period can be found in Figure 1 (right). Like surface snow thickness, there is a clear seasonal trend. Unlike the previous variables, however, we can clearly see that there is a slight long term trend as the near surface temperature is expected to increase over the next century due to climate change.

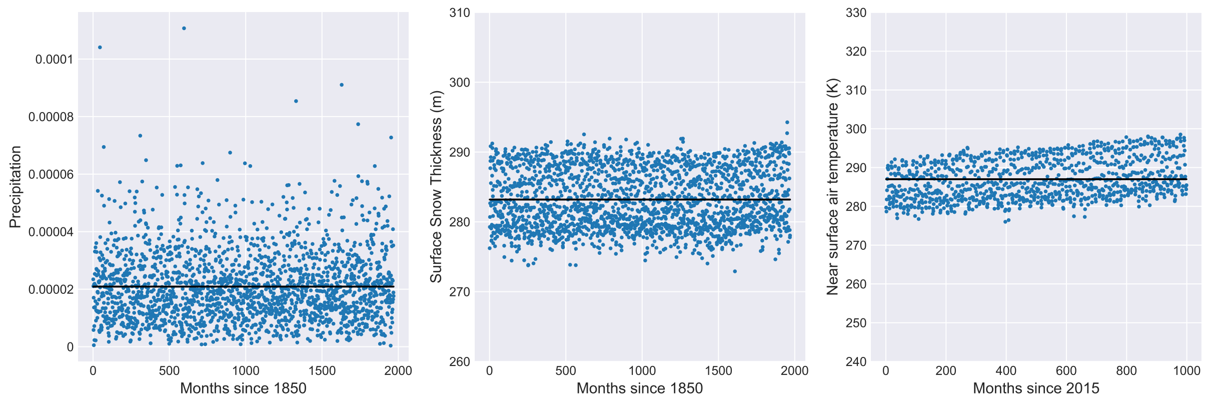


Figure 1: **Left:** simulated monthly mean precipitation over years 1850-2014 for location (51.875,0.9375) (UK). **Center:** simulated monthly mean surface snow thickness over years 1850-2014 for location (89.375,0.9375) (Antarctica). **Right:** simulated monthly mean near surface air temperature over years 2015-2100 for location (51.875,0.9375) (UK).

[1]`https://pcmdi.llnl.gov/mips/cmip3/variableList.html`

4 Extrapolation using Gaussian processes

Our approach to use Gaussian processes for time series forecasting and, more broadly, extrapolation of data was inspired by the ideas presented in Duvenaud's PhD thesis [Duv14] and Rasmussen and Williams' book *"Gaussian Processes for Machine Learning"* [Ras04]. Specifically, both works showed how Gaussian processes can be used to extrapolate the CO_2 level in the atmosphere over the last several decades. Similarly to their approach, we want to train a GP regression model on simulated data and extrapolate the underlying trend into the future where we do not have any data points.

4.1 Method

We assume a zero-mean Gaussian process. That is, we subtract mean of the data from each data point and then fit the GP on the residuals.

The main task is to design an appropriate kernel for our Gaussian process. Similarly to neural network architectures, there is no silver bullet choice of kernel – it is down to modeller's choice. Since we are dealing with climate data, it is obvious that we may need to model various short term trends (e.g. due to seasonality) and long term trends (e.g. due to climate change). Therefore, simply using stationary kernels, which quickly revert back to the mean, is not promising. For this reason, drawing inspiration from [Duv14; Ras04], we consider the following "building-block" kernels which are readily available in `GPy`:

- *RBF or Squared Exponential kernel* (RBF) - standard kernel for modelling smooth functions.

$$K_{RBF}(\mathbf{x}, \mathbf{x}') = \exp\Big(-\frac{\|\mathbf{x} - \mathbf{x}'^2\|}{2\sigma^2}\Big)$$

- *Periodic kernel* (PERIODIC) - used to capture periodicity in data (e.g. sin wave).

$$K_{Per}(\mathbf{x}, \mathbf{x}') = \sigma^2 \exp\Big(-\frac{2\sin^2(\pi|\mathbf{x} - \mathbf{x}'|/p}{\ell^2}\Big)$$

- *Rational Quadratic kernel* (RQ) - gives functions of varying smoothness; multiple RBF kernels combined.

$$K_{RQ}(\mathbf{x}, \mathbf{x}') = \sigma^2\Big(1 + \frac{(\mathbf{x} - \mathbf{x}')^2}{2\alpha\ell^2}\Big)^{-\alpha}$$

- *Linear kernel* (LINEAR) - used to draw linear functions, e.g. when there is a clear linear trend. Can be combined to obtain higher order polynomials.

$$K_{Lin}(\mathbf{x}, \mathbf{x}') = \sigma_b^2 + \sigma_v^2(\mathbf{x} - c)(\mathbf{x}' - c)$$

- *White noise* (WN) - generates independent and identically distributed noise.

$$K_{WN}(\mathbf{x}, \mathbf{x}') = \sigma^2 I_n$$

Combining these kernels (adding, multiplying and scaling) allows us to obtain more complex kernels and model various trends in the data, e.g. short term/long term periodicity. For example, we will consider the following combinations of kernels:

- RBF $\times$ PERIODIC - gives functions that are nearly periodic.
- RBF $\times$ LINEAR - can capture various trends (increasing/decreasing smooth functions).
- RBF $\times$ WN - gives varying noise levels

The choice of best kernel is non-trivial. In this report, we construct kernels manually. This, of course, is tedious and prone to errors. As Duvenaud put it, *"even for experts, choosing the kernel in GP regression remains something of a black art"*. Therefore, we also tried implementing automatic methods, e.g. Duvenaud's model search procedure used to build kernels using greedy search. However, we were unable to obtain good results on our data, mainly because of the difficulty of initialising a kernel with good hyperparameters. We address this by running hyperparameter optimisation.

4.2 Optimisation

During training, selecting a suitable kernel has the highest impact on the quality of the model fit. However, the choice of optimiser hyperparameters also has an outsized impact on the model error. A naïve solution is to choose these empirically, based on their performance on experiments that were carried out. But this is somewhat arbitrary, and we would like a principled way to make the decision.

Hyperparameter optimisation is a set of techniques concerned with finding the best combination of parameters for a given model. For example, in grid search, the modeller specifies a set of values that each parameter can take on. The search exhaustively covers all combinations, and each model is evaluated according to some metric. One issue with grid search is that depending on the number of hyperparameters to optimise and on the number of possible values each can take, the computational cost could make the search intractable. For example, one of the kernels used for this project, which was a combination of RBF, cosine and rational quadratic kernels had at least 8 hyperparameters. If each of these parameters were to take on 4 values, there would be $8^4 = 4,096$ combinations to try. Even if the model could be fit and evaluated in 30 seconds, it would still take 34 hours to optimise the hyperparameters in this way.

We instead optimise the hyperparameters of our kernels using Bayesian optimisation. The premise is relatively simple: consider a function that takes as arguments values for each hyperparameter, and returns the error of a model initialised using those hyperparameters. This is a function like any other, and we can perform Bayesian optimisation on it using another GP. The advantage here is that the candidate hyperparameters are chosen according to some acquisition function. In our case, we used lower confidence bounds for GPs[SLA12], but we achieved similar results with Expected Improvement (EI)[JSW98].

Early attempts used the `optimize()` function built into the `emukit` package[Pal+19], and largely delivered good results. However, when we could not get it to work reliably with some of the kernels we tried, we turned to the Bayesian optimiser in the `GPyOpt` Python library [aut16]. The latter gives the user freedom to specify most important choices about the optimisation itself: the model used to optimise (usually a GP, but we also tried a sparse GP with good results), the acquisition function, and other parameters. We built a wrapper around the optimiser so we could run experiments easily from the command line, including a more verbose log and a timer.

Optimisation was run for a maximum of 10, 50 or 100 iterations on various combinations of acquisition function, model type and domains for the kernels we had previously selected. Due to computational constraints, most of the optimisation experiments were on a window of 100 months (vs. the total of 1,980 monthly readings at each coordinate point), with 80/20 train-test splits. Depending on the number of iterations, size of the dataset and number of hyperparameters of the kernel, the optimisation experiments took anywhere from 5 seconds to several hours to run on an 8-core Intel i9 CPU, with no GPU acceleration.

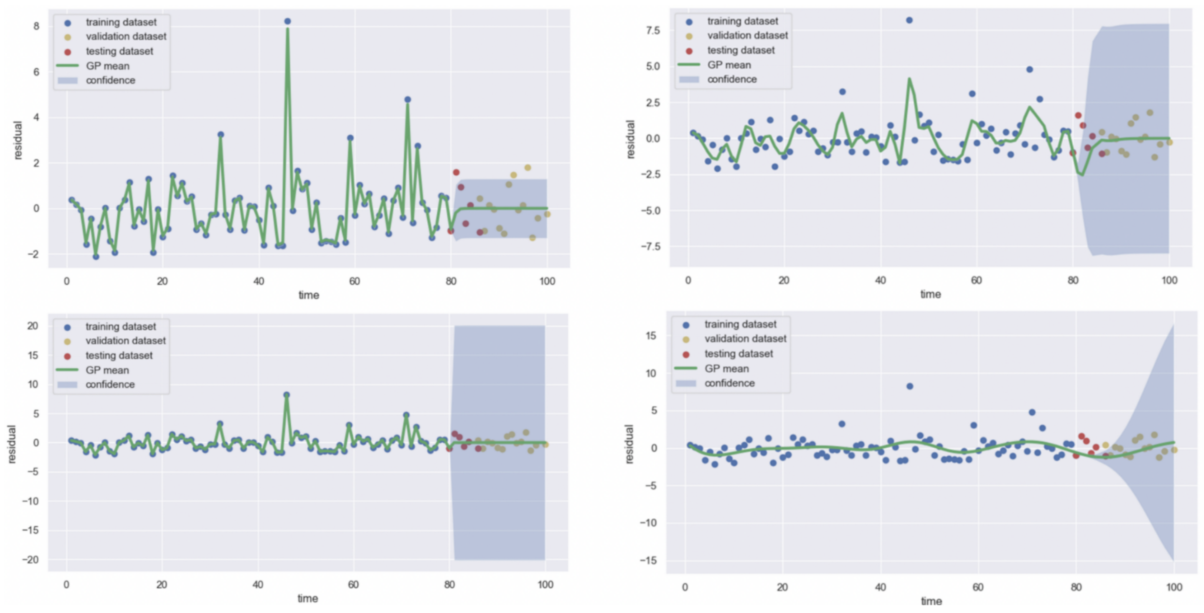


Figure 2: Results of Bayesian hyperparameter optimisation on the surrogate model. The top row and the left-most plot on the second row correspond to optimisation with smaller domains for the length-scale parameter. The fourth plot corresponds to a larger length scale domain, and a relatively small variance.

Using the in-house optimiser, we noticed that the resulting optimal values for the hyperparameters were highly sensitive to the domain over which we optimised. For length scale and variance in particular, the only constraint is that the values are positive. In theory, they can be arbitrarily high. In practice, we observed a number of interesting effects:

1. Small values for the length scale lead to more irregular functions, i.e. that can vary more quickly. This is good for interpolating the data points in the training set, but the extrapolation performance is poor – the model exhibits reversion to the mean. In Figure 2, plots 1, 2 and 3 belong to models where the hyperparameter optimisation was constrained to a small domain for the length scale: $(0, 5)$, $(0, 1)$ and $(0, 1)$, respectively.

2. Conversely, large values of the length scale result in more smooth functions. These don't fit the training data exactly, but the resulting extrapolation is more meaningful, at least for a reasonable amount of months into the future. Plot 4 in Figure 2 corresponds to a model whose length scales were optimised inside the domain $(0, 30)$.

3. Large values of the variance seem to correlate with higher uncertainty in predictions on the test set. However, the effect variance has on the quality of the predictions is affected by the choice of length scale: in the fourth plot above the uncertainty of predictions increases as one tries to predict farther into the future, but this happens gradually. In plot 3, the small length scale results in good interpolation, but as soon as we step into the test data the model reverts to the mean and has very high uncertainty in its predictions.

To try and decouple these domain effects from the skill of the modeller, we implemented grid search over the domains. In practice, the grid search was over the end values of intervals of real numbers that always begin at 0. The grid was on a logarithmic scale, and due to computational limits we only generated a small amount of possible values ranging $[10^{-3}, 10^{3}]$. A resulting domain is e.g. $(0, 0.1)$.

4.3 Sensitivity analysis

We attempted an uncertainty analysis on one of our target variables to determine the influence of the input variables on the output (precipitation). The goal was to gain some insight on the internal mapping of input to output, hopefully helping with choosing an appropriate kernel. Additionally, we hoped to attain some justification for our earlier intuition that sampling from a single spatial point and then interpolating and extrapolating across the time dimension is an adequate methodology for fitting a Gaussian process for climate data.

Assuming a non-correlated relationship between input variables, we restricted the model's input size to 10,000 points within the Tropics (23°N - 23°S) over a time period of 115 years (1850 - 1965). This decision is driven by the periodicity of the data within that time period, the high variance in precipitation in the Tropics, and the size of the dataset. We used the sensitivity module in the `emukit` library to compute the main and total effects of the inputs. The module carries out sampling using a Monte Carlo technique which works well with our assumption of uncertainty in the input data[KO00]. The module's method for computing the Sobol indices and total effects is based on [Sal02]. The GP model used for the sensitivity analysis was initially trained using no kernel, a reflection of our ignorance of the underlying function and later on using a composite kernel that worked well with interpolating and extrapolating over a single geographical point.

Time seemed to have more of an effect on the output, somewhat validating our initial intuition that the data could be modelled as a temporal series. However, the analysis appears to be dependent on our perception of the underlying system defined by the kernel. Unfortunately, uncertainty analysis did not proffer any solutions that could help us design better kernels.

5 Results

In this section, we present our emulator results. We firstly focus on extrapolating time series data at specific locations and then try to extend this to the whole region by incorporating geospatial data.

5.1 Results for precipitation at single locations

Unfortunately, we are not able to extrapolate precipitation data. Some of the example models and their extrapolations can be seen in Figure 2. Ultimately, we are able to fit training data with various combinations of periodic kernels, however extrapolation does not give fruitful results. We tried a variety of kernel combinations and hyperparameter initialisations, yet were unable to find any meaningful trends using our kernels of choice.

5.2 Results for monthly-mean surface snow thickness at single locations

As mentioned, monthly-mean surface snow thickness data shows seasonal pattern without any visible long term trend. Therefore, we use the following kernel to capture irregular seasonal periodicity:

$$\text{RBF} \times (\text{PERIODIC} + \text{RQ})$$

This is a simple model that extends the periodic kernel (PERIODIC) with some short term irregularities (RQ). RBF $\times$ PERIODIC captures the irregular seasonal periodicity (more snow in winter, less in summer). RBF $\times$ RQ, on the other hand, captures small deviations from the mean. Emulation of a specific point in Antarctica (where snow thickness is never zero) can be seen in Figure 3 below:

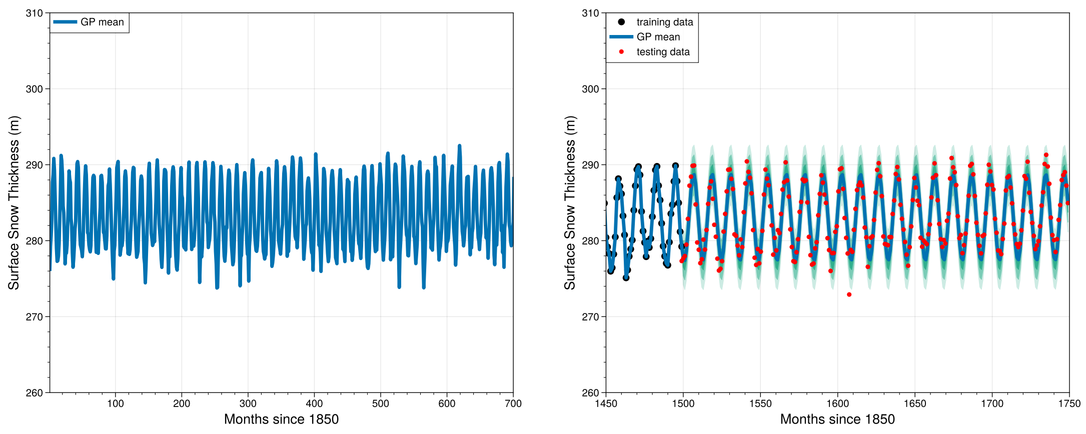


Figure 3: Monthly-mean surface snow thickness forecast. Blue: GP prediction mean. Green: 1,2 and 3 standard deviations. **Left:** training data fit for years 1850-1908. **Right:** forecast for years 1975-1995.

Unfortunately, after optimisation we do not observe RQ having much impact on the emulator. Note that this kernel may not give good results in other regions, e.g. where it snows for only one or two months a year. However, here we assume regions where snow is permanent or nearly permanent.

5.3 Results for monthly-mean air temperature prediction at single locations

As already mentioned, monthly mean future air temperature is expected to increase due to climate change. For this task, our best performing kernel is:

$$(\text{LINEAR} \times \text{RBF}) + \text{RBF} \times (\text{PERIODIC} + \text{RQ})$$

The LINEAR $\times$ RBF term models the long term trend of increasing mean. The second term is similar to the kernel described for surface snow thickness experiment. Surface air temperature prediction for a specific location can be seen in Figure 4.

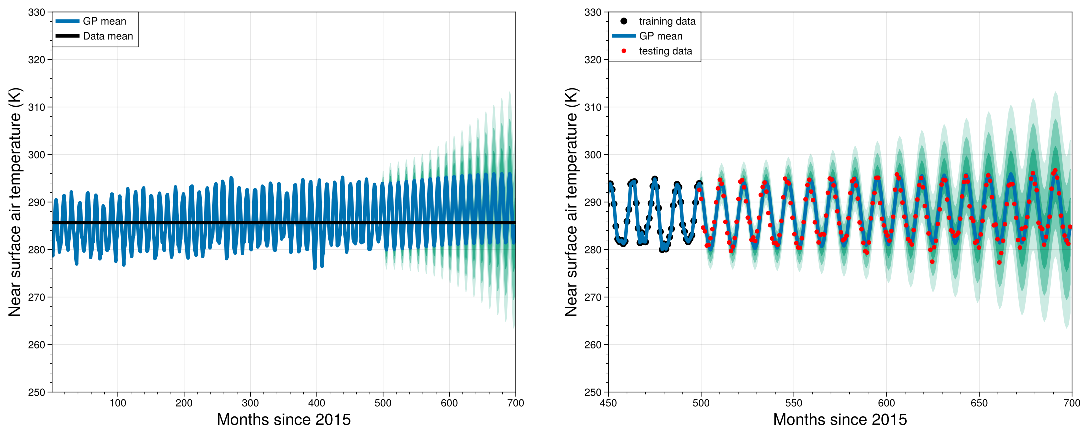


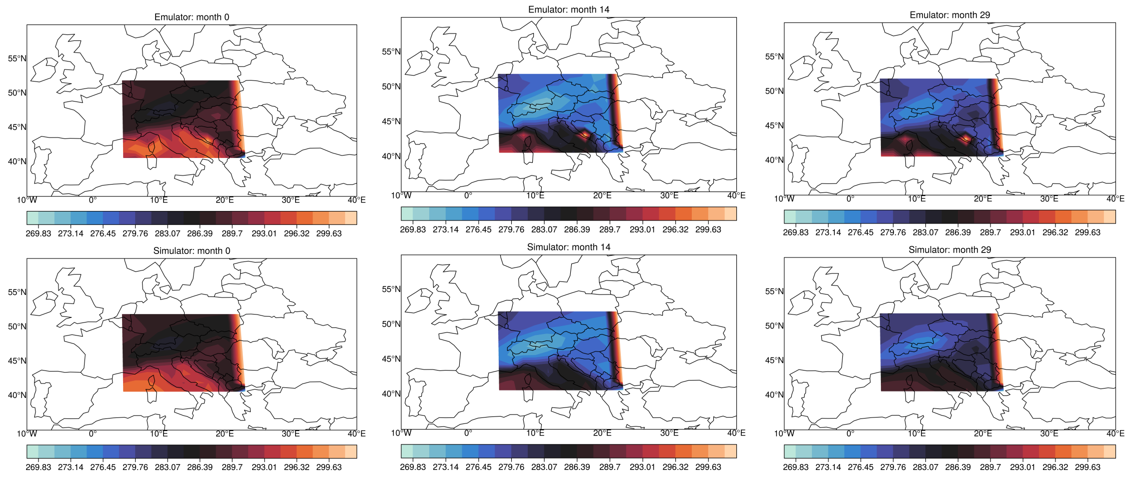
Figure 4: Monthly mean near surface air temperature forecast. Blue: GP prediction mean. Green: 1, 2 and 3 standard deviations. **Left:** training data fit for years 2015-2056 (months 0-500) and test data fit for years 2056-2073 (months 500-700). **Right:** zoomed-in forecast for years 2056-2073 (months 500-700).

We are able to fit the training data well and extrapolate into the future quite accurately. Our model captures both seasonal patterns (lower average temperature during winter months and higher during summer months) and slightly increasing mean over years. Especially accurate are the first 10 years. As expected, forecasts further than that have more uncertainty, however most of data points still fall within 1 standard deviation from the mean.

5.4 Results for monthly-mean air temperature prediction at specific geographical region

We present our kernel mean prediction for monthly-mean air temperature prediction in Central Europe (100 locations). Three different months from test data are shown in Figure 5. Note that the 0th month here corresponds to the first month of year 2056 where predictions start.

The mean prediction captures the trend well for majority of locations, although there are some points where our emulator clearly cannot extrapolate well. Looking at the fitted graph of each such location, we notice that the optimizer simply cannot find globally optimal hyperparameters and there is no decent extrapolation happening. For example, in Figure 5 (b) and (c), you can see locations in Balkans and Sardinia where the model does not extrapolate at all and gives high temperature.



(a) (b) (c)

Figure 5: Predicted monthly-mean air temperature for various locations in Central Europe at different test months by our emulator versus simulator output. Values in Kelvin (K). 100-month long GIF can be found in the code repository[Ale+22]

6 Discussion

6.1 Computational gains and their limits

In climate science, the advantages of running a surrogate over a climate model are largely related to the gain in speed. Climate models in general are computationally intensive, and the increase in compute availability in the last few years has only partially mitigated the issue. Due to the complexity of Earth's climate, some key processes, like cloud formation, can only be resolved by high-resolution models. Newer models aren't necessarily faster: as there is a finite computational budget, a choice has to be made between faster and more fine-grained experiments (and perhaps ones which take into account physical processes which were previously parameterised)[WBC09].

Earth System Models (ESMs) are large global climate models that cover many interconnecting physical processes. Their strength lies in their capability to model the interactions between different systems by being coupled to other 'specialist' models. Because of their capacity to integrate many of these complex processes, ESMs are some of the most skilful models in use today. On the other hand, due to their size, ESMs like the UKESM[Sel+19] require dedicated high-performance computing clusters and can take several days to produce data that spans a few years. These models also generate on the order of petabytes of data, which introduces challenges related to storage, access and maintenance of the outputs.

Because surrogate models are faster, a reasonably accurate model can be used to perform experiments that are intractable to run on the climate model directly. Using emulators, it's possible to reason about counterfactuals – what would happen had a certain quantity or phenomenon been different at some point in the past. The emulator can be retrained as many times as needed, and to varying degrees of fidelity. We chose monthly atmospheric data as a middle ground between the low temporal resolution yearly measurements and very fine-grained daily and intra-day simulations. In some of our experiments, the fidelity of the emulator had a direct impact on the visibility of multi-year trends.

At the same time, Gaussian process emulators themselves are constrained by the size of the dataset, and may require dimensionality reduction to be feasible. For example, in some applications, instead of using all outputs from a climate model, the emulator is fit on a mean of the measurements[McN+20]. Recent approaches to scalable GPs have improved the original $\mathcal{O}(n^3)$ time complexity, making this approach more applicable to domains with large data[Liu+20].

6.2 Using Gaussian processes for extrapolation

Unfortunately, Gaussian processes have quite a few limitations when it comes to extrapolating time series data. The major obstacle is finding suitable kernels. In our experience, even the automatic kernel building approach needed some manual tuning. Moreover, we struggled with hyperparameter initialisation, as it was one of the major factors for hyperparameter optimisation success. Lastly, it is not feasible to extrapolate into the distant future, especially for data we considered. Due to feedback loops, long-term climate trends may not be straightforward to anticipate.

We were unable to extrapolate for noisy data that does not show clear short term or long term patterns, for example monthly mean precipitation. State-of-the-art deep learning models may be more suitable for this task. For example, the paper [Web+20] that used CNN were able to at least somewhat extrapolate. It is important to mention, however, that Gaussian processes let us quantify uncertainty, which standard deep-learning methods lack.

References

[Web+20] Theodore Weber et al. "Deep learning for creating surrogate models of precipitation in Earth system models". In: *Atmospheric Chemistry and Physics* 20.4 (2020), pp. 2303–2317.

[Wat+21] D. Watson-Parris et al. "Model calibration using ESEm v1.1.0 – an open, scalable Earth system emulator". In: *Geoscientific Model Development* 14.12 (2021), pp. 7659–7672. DOI: 10.5194/gmd-14-7659-2021. URL: https://gmd.copernicus.org/articles/14/7659/2021/.

[Gla+19] F. Glassmeier et al. "An emulator approach to stratocumulus susceptibility". In: *Atmospheric Chemistry and Physics* 19.15 (2019), pp. 10191–10203. DOI: 10.5194/acp-19-10191-2019. URL: https://acp.copernicus.org/articles/19/10191/2019/.

[Ban+19] Daniel Bannister et al. "Bias Correction of High-Resolution Regional Climate Model Precipitation Output Gives the Best Estimates of Precipitation in Himalayan Catchments". In: *Journal of geophysical research.* 124.24 (2019), pp. 14220–14239. ISSN: 2169-897X.

[OL18] Qi Ouyang and Wenxi Lu. "Monthly Rainfall Forecasting Using Echo State Networks Coupled with Data Preprocessing Methods". In: *Water Resources Management* 32.2 (2018), pp. 659–674. DOI: 10.1007/s11269-017-1832-1. URL: https://doi.org/10.1007/s11269-017-1832-1.

[Zha+20] Chang-Jiang Zhang et al. "Correction model for rainfall forecasts using the LSTM with multiple meteorological factors". In: *Meteorological Applications* 27.1 (2020), e1852.

[Wan+20] Qingrui Wang et al. "Sequence-based statistical downscaling and its application to hydrologic simulations based on machine learning and big data". In: *Journal of Hydrology* 586 (2020), p. 124875.

[Cas+14] Stefano Castruccio et al. "Statistical emulation of climate model projections based on precomputed GCM runs". In: *Journal of Climate* 27.5 (2014), pp. 1829–1844.

[WZV21] Chao Wang, Wei Zhang, and Gabriele Villarini. "On the use of convolutional Gaussian processes to improve the seasonal forecasting of precipitation and temperature". In: *Journal of Hydrology* 593 (2021), p. 125862. ISSN: 0022-1694. DOI: https://doi.org/10.1016/j.jhydrol.2020.125862. URL: https://www.sciencedirect.com/science/article/pii/S0022169420313238.

[Duv14] David Kristjanson Duvenaud. "Automatic model construction with Gaussian processes". In: 2014.

[Ras04] Carl Edward Rasmussen. "Gaussian Processes in Machine Learning". In: *Advanced Lectures on Machine Learning: ML Summer Schools 2003, Canberra, Australia, February 2 - 14, 2003, Tübingen, Germany, August 4 - 16, 2003, Revised Lectures.* Ed. by Olivier Bousquet, Ulrike von Luxburg, and Gunnar Rätsch. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 63–71. ISBN: 978-3-540-28650-9. DOI: 10.1007/978-3-540-28650-9_4. URL: https://doi.org/10.1007/978-3-540-28650-9_4.

[SLA12] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. "Practical bayesian optimization of machine learning algorithms". In: *Advances in neural information processing systems* 25 (2012).

[JSW98] Donald R Jones, Matthias Schonlau, and William J Welch. "Efficient global optimization of expensive black-box functions". In: *Journal of Global optimization* 13.4 (1998), pp. 455–492.

[Pal+19] Andrei Paleyes et al. "Emulation of physical processes with Emukit". In: *Second Workshop on Machine Learning and the Physical Sciences, NeurIPS*. 2019.

[aut16] The GPyOpt authors. *GPyOpt: A Bayesian Optimization framework in python.* http://github.com/SheffieldML/GPyOpt. 2016.

[KO00] M. C. Kennedy and A. O'Hagan. "Predicting the Output from a Complex Computer Code When Fast Approximations Are Available". In: *Biometrika* 87.1 (2000), pp. 1–13. ISSN: 00063444. URL: http://www.jstor.org/stable/2673557.

[Sal02] Andrea Saltelli. "Making best use of model evaluations to compute sensitivity indices". In: *Computer Physics Communications* 145.2 (2002), pp. 280–297. ISSN: 0010-4655. DOI: https://doi.org/10.1016/S0010-4655(02)00280-1. URL: https://www.sciencedirect.com/science/article/pii/S0010465502002801.

[Ale+22] Andrei Alexandru et al. *Investigating short term climate forecasts with surrogate modelling.* `https://github.com/inwaves/climate-surrogate-model`. Version 1.0. Jan. 2022.

[WBC09] Warren M Washington, Lawrence Buja, and Anthony Craig. "The computational future for climate and Earth system models: on the path to petaflop and beyond". In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 367.1890 (2009), pp. 833–846.

[Sel+19] Alistair A. Sellar et al. "UKESM1: Description and Evaluation of the U.K. Earth System Model". In: *Journal of Advances in Modeling Earth Systems* 11.12 (2019), pp. 4513–4558. DOI: `https://doi.org/10.1029/2019MS001739`. eprint: `https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/2019MS001739`. URL: `https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2019MS001739`.

[McN+20] D. McNeall et al. "Correcting a bias in a climate model with an augmented emulator". In: *Geoscientific Model Development* 13.5 (2020), pp. 2487–2509. DOI: `10.5194/gmd-13-2487-2020`. URL: `https://gmd.copernicus.org/articles/13/2487/2020/`.

[Liu+20] Haitao Liu et al. "When Gaussian Process Meets Big Data: A Review of Scalable GPs". In: *IEEE Transactions on Neural Networks and Learning Systems* 31.11 (2020), pp. 4405–4423. DOI: `10.1109/TNNLS.2019.2957109`.