Power Ensemble Aggregation for Improved Extreme Event AI Prediction

Julien Collard
Ecole Polytechnique
Rte de Saclay, 91120 Palaiseau, France
julien.collard@polytechnique.org

Pierre Gentine
LEAP, Columbia University
2276 12th ave, New York, NY 10027, USA
pg2328@columbia.edu

Tian Zheng
Department of Statistics, Columbia University
1255 Amsterdam Avenue, New York, NY 10027, USA

Abstract

This paper addresses the critical challenge of improving predictions of climate extreme events, specifically heat waves, using machine learning methods. Our work is framed as a classification problem in which we try to predict whether surface air temperature will exceed its q-th local quantile within a specified timeframe. Our key finding is that aggregating ensemble predictions using a power mean significantly enhances the classifier's performance. By making a machine-learning based weather forecasting model generative and applying this non-linear aggregation method, we achieve better accuracy in predicting extreme heat events than with the typical mean prediction from the same model. Our power aggregation method shows promise and adaptability, as its optimal performance varies with the quantile threshold chosen, demonstrating increased effectiveness for higher extremes prediction.

1 Introduction

Numerical weather prediction (NWP) has recently made significant progress with the development of deep learning models [1, 2, 3]. With the emergence of new architectures and the use of massive datasets [4] and well-defined benchmarks [5], these models have been able to predict weather variables more and more accurately and now usually outperform physically based models [5]. However, predicting extreme events remains a challenge, as they are rare by definition and follow different patterns than the rest of the data. [6, 7] For those events machine-learning based models are still struggling as recently shown by Zhang et al. [8] and Sun et al. [9]. Therefore, predicting extremes with ML requires specialized techniques and is an active field of research [10, 11, 12].

In this study, we investigate an adaptive ensemble aggregation method to predict extreme events applicable to any generative model [13, 14]. This method, based on the power mean [15, 16] was previously tested in different fields [17]. We applied it to extreme surface air temperature classification, but this method could be adapted to any extreme prediction problem. For our experiments, we used a custom NWP generative model inspired by Weyn et al. [18] and Lopez-Gomez et al. [12] based on convolutions on a cubic sphere grid and made generative using Perlin noise [19] for input perturbation.

2 Methodology

2.1 Data

For this study, we used the ERA5 reanalysis data from the European Centre for Medium-Range Weather Forecasts (ECMWF) [4] for training and evaluation. We downloaded data resampled at 1.5° spatial resolution, covering 1990-2020 and at 6-hour temporal resolution using the WeatherBench2 [5] cloud-based datasets. We chose to resample the data to a daily temporal resolution to remove the diurnal cycle and focus on large temporal scale extremes.

The full list of variables downloaded for this study and the levels at which they were sampled are presented in appendix in Table 1. This choice of weather variables to predict surface air temperature was inspired by the work of Lopez-Gomez et al. [12].

Following the work of Weyn et al. [18], we regridded the data to a 6x48x48 gnomonic equiangular cube sphere grid (appendix A) to avoid pole singularities and allow the use of the CubeSphere Convolutional Neural Networks they introduced.

2.2 Defining extreme events

To define extremes, we used local climatologies, i.e. weather variable distributions defined for each location and time of the year.

For a given threshold $0.5 \leq q \leq 1$, we define an extreme heat event as the exceedance of the q-th local quantile of the surface air temperature. This local definition is essential as we want to capture extreme events happening at any location and time of the year. Without it, we would be limited to capturing global extremes in hot regions like the Sahara desert.

Assuming that surface air temperatures nearly follow a Gaussian distribution, a temperature is considered q-extreme if its local anomaly x is such that $\Phi(x) \geq q$ with Φ the cumulative distribution function of the standard normal distribution $\mathcal{N}(0, 1)$. Local anomalies are computed by subtracting local climatology mean to temperatures and dividing by local climatology standard deviation.

2.3 Classification problem

In this study, we consider extreme events' detection as a classification problem: knowing the state of the atmosphere on a given day d and its past few days, we want to predict whether the surface air temperature at any location will exceed its q-th local quantile on day $d + \Delta d$. For a given quantile q to be exceeded, on a given day d, a given time lead Δd and a given location, this becomes a binary classification problem. We denote x as the true local anomaly of the surface air temperature at time $d + \Delta d$ and at this location. The ground truth label y is then defined as $y = \mathbb{1}_{\Phi(x) \geq q} \in \{0, 1\}$.

We expect models to predict a score $\hat{s} \in [0, 1]$ that reflects how confident they are that the surface air temperature will exceed its q-th local quantile. Such a model is called a "score-based classifier". For real applications, a threshold τ must then be chosen to convert the score into a binary prediction $\hat{y} = \mathbb{1}_{\hat{s} \geq \tau}$.

To evaluate the performance of the heat-wave classifiers, we used the area under the receiver operating characteristic curve (AUC), a well-established metric for binary classification problems. It is defined as the area under the curve of the true positive rate (TPR) as a function of the false positive rate (FPR) for different thresholds. This metric measures the model's ability to distinguish between regular and extreme events.

2.4 Aggregation methods

Given the previous classification problem, we must now define how to get a $[0, 1]$ score from a model that outputs continuous weather variables, as a single prediction or as an ensemble of predictions.

When the model is deterministic (single prediction), from the single predicted local real-valued anomaly $\hat{x}$, a natural score $\hat{s}$ can be computed as $\hat{s} = \Phi(\hat{x})$. Given previous considerations, $\hat{s}$ should represent the predicted quantile and we can expect $\tau = q$ to be a good threshold.

However, when the model is generative and outputs an ensemble of local anomalies $\{\hat{x}_i\}_{1\leq i\leq n}$, the score $\hat{s}$ is not as straightforward to compute. A simple method is to compute the mean of the local anomalies and then apply the same procedure, yet this can be suboptimal as we will discuss later.

In this study, we investigate the use of a less trivial aggregation method: the power mean. As we are trying to predict extreme events, which are rare by definition, we might want to give more "weight" to members predicting an extreme. The most radical way to do that would be to use the maximum over the members as the actual prediction. However, when generating many members, this method would create too many False Positives. We therefore used a power mean aggregation [15, 16] as a compromise between the average and the maximum methods.

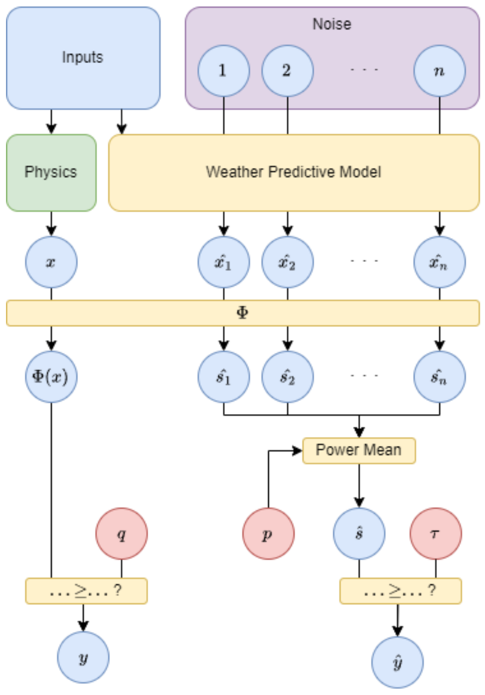


Figure 1: Aggregation method flowchart

Let's denote n the number of members generated, $p \geq 1$ a power exponent and $\{\hat{x}_i\}_{1\leq i\leq n}$ the local anomalies predictions. For each member of the ensemble, we compute a score $\hat{s}_i = \Phi(\hat{x}_i)$. The final score $\hat{s}$ is then computed as:

$$\hat{s} = \left(\frac{1}{n}\sum_{i=1}^{n} \hat{s}_i^p\right)^{1/p} \tag{1}$$

This method is illustrated in Figure 1. Note that the mean is applied to member scores $\hat{s}_i$ and not directly to anomalies $\hat{x}_i$ because this operation requires positive numbers.

Previous studies already investigated the use of the power mean for ensemble prediction aggregation in various application contexts [17] . They demonstrated that the arithmetic mean is not always the best choice and that the optimal value of p can vary depending on the application.

2.5 Model architecture

In order to test our aggregation method, we needed a generative model that could predict and output an ensemble of local anomalies. We chose to use a U-Net Convolutional Neural Network-like model inspired by the work of Weyn et al. [18] and Lopez-Gomez et al. [12]. Details on the model architecture are given in appendix B. The inputs of our model are atmospheric and ground variables at days d, $d-1$,... $d-3$, and the outputs are surface air temperature local anomalies at days $d+1$, $d+2$, $\ldots$, $d+12$.

To make our model generative, we followed the work of Bi et al. [1] and integrated Perlin noise [19] into the input of the deterministic baseline. For a given member of the ensemble, noise is randomly generated at different space scales and combined with the input through convolutions (more details in appendix B.2). For the results presented in this paper, the model generated $n = 50$ ensemble members.

The model was trained on data from 1990 to 2010 using a Continuous Ranked Probability Score (CRPS) [14] loss function. For more details on the training procedure, see appendix B.4.

3 Results

3.1 Optimizing the power exponent

To classify heat waves with a generative model, we introduced a power exponent p as a hyperparameter

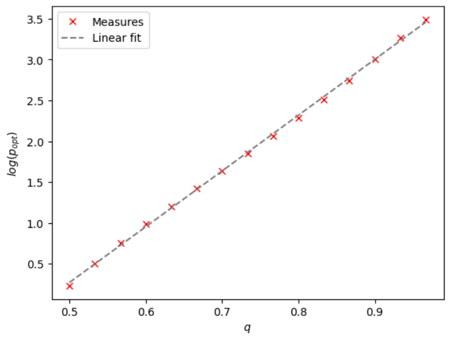


Figure 2: $log(p_{opt}) = f(q)$

of the aggregation method. For a given quantile threshold q, we computed the AUC for different values of p on the validation dataset (2010-2015). The forecast lead time was fixed to 7 days for this experiment. As conjectured, this function reaches a maximum at a given value of $p_{opt} \geq 1$ and exceeds the AUC of the mean prediction method.

When computing the optimal power exponent p_{opt} for different quantile thresholds, we found that this quantity increases almost perfectly exponentially with q as shown in Figure 2. This exponential scaling provides a simple estimate that can be used to predict the optimal power exponent for any given quantile threshold. Also, it emphasizes the adaptiveness of this method in its tuning when it comes to predicting extreme events, see appendix C for more details.

3.2 Final performance

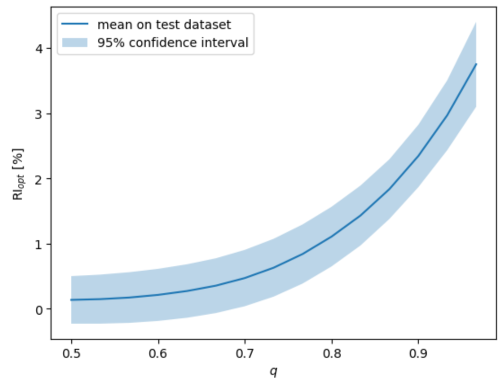


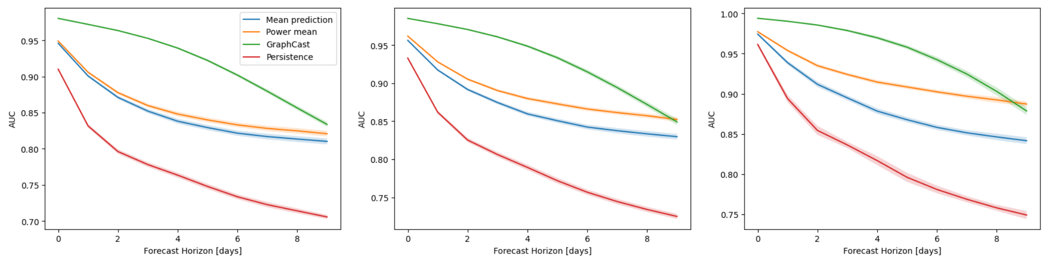
Figure 3: $\mathrm{RI}_{opt} = f(q)$ on test dataset

In the test data set (2015-2018), we computed the AUC for 7-day-ahead predictions, for different quantiles, and for both mean and power mean aggregation methods. To measure the effectiveness of our method, we define for each quantile q the relative improvement as:

$$\mathrm{RI}_{opt} = 100 \times \frac{\mathrm{AUC}_{p_{opt}} - \mathrm{AUC}_{\text{mean pred}}}{\mathrm{AUC}_{\text{mean pred}}} \tag{2}$$

The results presented in Figure 3 show that the power mean aggregation method outperforms the mean prediction method for all quantiles and its effectiveness increases with the quantile threshold considered.

Finally, we compared the performance of our method to a state-of-the-art AI weather model: GraphCast from DeepMind (deterministic) as a reference [3]. We kept the power exponents optimized on 7-day-ahead predictions but used them to compute the AUC for other forecast lead times. We compared 4 models: the mean prediction method (ensemble predictions), the optimized power mean method (ensemble predictions), GraphCast (single prediction), and the persistence model (single prediction) as a baseline. The results calculated for the year 2018 are presented in Figure 4.



(a) $q = 0.8$ (b) $q = 0.9$ (c) $q = 0.98$

Figure 4: AUC as a function of the forecast lead time for different quantiles

Again, the results show that the power mean aggregation method outperforms the mean prediction method for all quantiles and forecast lead times, confirming the relevance of this method. Also, they suggest that the relative improvement increases with the forecast lead time in addition to the quantile threshold. Moreover, for high quantiles and time leads, the power mean method applied to our simple generative model even outperforms GraphCast, illustrating once again the effectiveness of this method for extreme events prediction.

4 Conclusion

In this paper, we investigated the use of the power mean as an adaptive ensemble aggregation method for extreme event classification. The results confirmed the relevance of such an approach and showed that its effectiveness increases with the level of extreme intensity considered.

For future work, it would be interesting to apply this method to more complex and effective baseline models. This approach could also be generalized to any kind of weather variable to predict other extremes like droughts, heavy rains, storms…

Impact Statement

Extreme weather events, such as heat waves, have dramatic impacts including direct human losses, health issues, and material damages, imposing enormous costs on society. With climate change, these events are increasing in both frequency and intensity as stated by the Intergovernmental Panel on Climate Change (IPCC) in the 11th chapter of their 6th assessment report. [20]

In this context, predicting extreme events is more important than ever. Accurate predictions can be used at various scales to trigger preparation or emergency plans, allowing society to anticipate better, be more resilient, and reduce the impact of extreme events.

The method we investigated in this study shows real promise, significantly improving the performance of our model, and easily transferable and applicable to another model. An increase in the accuracy of extreme event classification could help society better anticipate heat waves and other meteorological disasters, reducing their negative impact, such as mortality rates.

Reproducibility

The data used in this study can be directly downloaded from the WeatherBench2 website.[5]

The appendix details the data used and the model architecture implemented for reproducibility purposes.

Experiments were conducted on a single GPU of 16GB. Storing the data and the models requires about 20 GB of disk space. Training the model takes a few hours on a single GPU. Testing a model to reproduce the results presented in this paper takes less than an hour on a single GPU.

Limitations

This study has several limitations that should be considered:

- **Simplified definition of extreme events:** The definition of extreme events used in this study is quite simple and does not capture all the complexity of real extreme events. The work of Zscheischler et al. [6] illustrates this complexity and the need for more advanced definitions.
- **Univariate problems:** Many extreme events are multivariate and depend on the interaction of several variables. Working with such problems requires more sophisticated climatology models, like copula-based distributions. The work presented in this paper only focuses on a single variable: surface air temperature.
- **Static climatology:** The anomalies considered in this study to define extremes and train models are based on a static climatology that does not take into account climate change. This is a major limitation as climate change is expected to significantly impact extreme events. For future work, it would be interesting to consider a dynamic climatology that evolves with time.
- **Metric choice:** The AUC metric used in this study is a general metric for binary classification problems but is not necessarily the best choice. To measure the performance of extreme events predictors from a more application-based perspective, socio-economic considerations could be integrated.

References

[1] Kaifeng Bi, Lingxi Xie, Hengheng Zhang, Xin Chen, Xiaotao Gu, and Qi Tian. Pangu-Weather: A 3D High-Resolution Model for Fast and Accurate Global Weather Forecast, November 2022.

[2] Jaideep Pathak, Shashank Subramanian, Peter Harrington, Sanjeev Raja, Ashesh Chattopadhyay, Morteza Mardani, Thorsten Kurth, David Hall, Zongyi Li, Kamyar Azizzadenesheli, Pedram Hassanzadeh, Karthik Kashinath, and Animashree Anandkumar. FourCastNet: A Global Data-driven High-resolution Weather Model using Adaptive Fourier Neural Operators, February 2022.

[3] Remi Lam, Alvaro Sanchez-Gonzalez, Matthew Willson, Peter Wirnsberger, Meire Fortunato, Ferran Alet, Suman Ravuri, Timo Ewalds, Zach Eaton-Rosen, Weihua Hu, Alexander Merose, Stephan Hoyer, George Holland, Oriol Vinyals, Jacklynn Stott, Alexander Pritzel, Shakir Mohamed, and Peter Battaglia. GraphCast: Learning skillful medium-range global weather forecasting, August 2023.

[4] Hans Hersbach, Bill Bell, Paul Berrisford, Shoji Hirahara, András Horányi, Joaquín Muñoz-Sabater, Julien Nicolas, Carole Peubey, Raluca Radu, Dinand Schepers, Adrian Simmons, Cornel Soci, Saleh Abdalla, Xavier Abellan, Gianpaolo Balsamo, Peter Bechtold, Gionata Biavati, Jean Bidlot, Massimo Bonavita, Giovanna De Chiara, Per Dahlgren, Dick Dee, Michail Diamantakis, Rossana Dragani, Johannes Flemming, Richard Forbes, Manuel Fuentes, Alan Geer, Leo Haimberger, Sean Healy, Robin J. Hogan, Elías Hólm, Marta Janisková, Sarah Keeley, Patrick Laloyaux, Philippe Lopez, Cristina Lupu, Gabor Radnoti, Patricia de Rosnay, Iryna Rozum, Freja Vamborg, Sebastien Villaume, and Jean-Noël Thépaut. The ERA5 global reanalysis. *Quarterly Journal of the Royal Meteorological Society*, 146(730):1999–2049, 2020. ISSN 1477-870X. doi: 10.1002/qj.3803.

[5] Stephan Rasp, Stephan Hoyer, Alexander Merose, Ian Langmore, Peter Battaglia, Tyler Russel, Alvaro Sanchez-Gonzalez, Vivian Yang, Rob Carver, Shreya Agrawal, Matthew Chantry, Zied Ben Bouallegue, Peter Dueben, Carla Bromberg, Jared Sisk, Luke Barrington, Aaron Bell, and Fei Sha. WeatherBench 2: A benchmark for the next generation of data-driven global weather models, January 2024.

[6] Jakob Zscheischler, Olivia Martius, Seth Westra, Emanuele Bevacqua, Colin Raymond, Radley M. Horton, Bart van den Hurk, Amir AghaKouchak, Aglaé Jézéquel, Miguel D. Mahecha, Douglas Maraun, Alexandre M. Ramos, Nina N. Ridder, Wim Thiery, and Edoardo Vignotto. A typology of compound weather and climate events. *Nature Reviews Earth & Environment*, 1(7):333–347, July 2020. ISSN 2662-138X. doi: 10.1038/s43017-020-0060-z.

[7] A.C. Davison and R. Huser. Statistics of Extremes. *Annual Review of Statistics and Its Application*, 2(1):203–235, April 2015. ISSN 2326-8298, 2326-831X. doi: 10.1146/annurev-statistics-010814-020133.

[8] Zhongwei Zhang, Erich Fischer, Jakob Zscheischler, and Sebastian Engelke. Numerical models outperform ai weather forecasts of record-breaking extremes, 2025.

[9] Y. Qiang Sun, Pedram Hassanzadeh, Mohsen Zand, Ashesh Chattopadhyay, Jonathan Weare, and Dorian S. Abbot. Can ai weather models predict out-of-distribution gray swan tropical cyclones? *Proceedings of the National Academy of Sciences*, 122(21), May 2025. ISSN 1091-6490. doi: 10.1073/pnas.2420914122. URL `http://dx.doi.org/10.1073/pnas.2420914122`.

[10] Wei Fang, Qiongying Xue, Liang Shen, and Victor S. Sheng. Survey on the Application of Deep Learning in Extreme Weather Prediction. *Atmosphere*, 12(6):661, June 2021. ISSN 2073-4433. doi: 10.3390/atmos12060661.

[11] David Richardson L. Magnusson. Verification of extreme weather events: Discrete predictands. https://www.ecmwf.int/en/elibrary/75517-verification-extreme-weather-events-discrete-predictands, 201409/2014.

[12] Ignacio Lopez-Gomez, Amy McGovern, Shreya Agrawal, and Jason Hickey. Global Extreme Heat Forecasting Using Neural Weather Models. *Artificial Intelligence for the Earth Systems*, 2 (1):e220035, January 2023. ISSN 2769-7525. doi: 10.1175/AIES-D-22-0035.1.

[13] Jieyu Chen, Tim Janke, Florian Steinke, and Sebastian Lerch. Generative machine learning methods for multivariate ensemble post-processing. *The Annals of Applied Statistics*, 18(1), March 2024. ISSN 1932-6157. doi: 10.1214/23-AOAS1784.

[14] Jonathan Berrisch and Florian Ziel. CRPS Learning. *Journal of Econometrics*, 237(2):105221, December 2023. ISSN 03044076. doi: 10.1016/j.jeconom.2021.11.008.

[15] David W. Cantrell and Eric W. Weisstein. Power Mean. https://mathworld.wolfram.com/PowerMean.html.

[16] Jonathan M. Borwein and Peter B. Borwein. *Pi and the AGM: A Study in Analytic Number Theory and Computational Complexity*. Wiley-Interscience, New York, July 1998. ISBN 978-0-471-31515-5.

[17] Mohammed Falih Hassan, Ikhlas Abdel-Qader, and Bradley Bazuin. A new method for ensemble combination based on adaptive decision making. *Knowledge-Based Systems*, 233: 107544, December 2021. ISSN 0950-7051. doi: 10.1016/j.knosys.2021.107544.

[18] Jonathan A. Weyn, Dale R. Durran, and Rich Caruana. Improving Data-Driven Global Weather Prediction Using Deep Convolutional Neural Networks on a Cubed Sphere. *Journal of Advances in Modeling Earth Systems*, 12(9):e2020MS002109, 2020. ISSN 1942-2466. doi: 10.1029/2020MS002109.

[19] Ken Perlin. An image synthesizer. *SIGGRAPH Comput. Graph.*, 19(3):287–296, July 1985. ISSN 0097-8930. doi: 10.1145/325165.325247.

[20] Intergovernmental Panel On Climate Change (Ipcc). *Climate Change 2021 – The Physical Science Basis: Working Group I Contribution to the Sixth Assessment Report of the Intergovernmental Panel on Climate Change*. Cambridge University Press, 1 edition, July 2023. ISBN 978-1-00-915789-6. doi: 10.1017/9781009157896.

[21] Paul A. Ullrich and Mark A. Taylor. Arbitrary-Order Conservative and Consistent Remapping and a Theory of Linear Maps: Part I. *Monthly Weather Review*, 143(6):2419–2440, June 2015. ISSN 0027-0644, 1520-0493. doi: 10.1175/MWR-D-14-00343.1.

[22] Paul A. Ullrich, Dharshi Devendran, and Hans Johansen. Arbitrary-Order Conservative and Consistent Remapping and a Theory of Linear Maps: Part II. *Monthly Weather Review*, 144(4): 1529–1549, April 2016. ISSN 0027-0644, 1520-0493. doi: 10.1175/MWR-D-15-0301.1.

[23] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation, May 2015.

[24] Ken Perlin. MAKING NOISE, December 1999.

[25] Pierre Vigier. perlin-jax, July 2024. URL `https://github.com/puffy310/perlin-jax`.

[26] Jonathan Heek, Anselm Levskaya, Avital Oliver, Marvin Ritter, Bertrand Rondepierre, Andreas Steiner, and Marc van Zee. Flax: A neural network library and ecosystem for JAX, 2023. URL `http://github.com/google/flax`.

[27] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL `http://github.com/google/jax`.

[28] Fisher Yu and Vladlen Koltun. Multi-Scale Context Aggregation by Dilated Convolutions, April 2016.

[29] Tilmann Gneiting and Adrian E Raftery. Strictly Proper Scoring Rules, Prediction, and Estimation. *Journal of the American Statistical Association*, 102(477):359–378, March 2007. ISSN 0162-1459, 1537-274X. doi: 10.1198/016214506000001437.

A Data and grid system

The data we downloaded for this study included solar irradiance, atmospheric variables such as geopotential and wind components at different pressure levels, and ground variables such as surface air temperature and soil water content. This choice of data to predict surface air temperature was inspired by the work of Lopez-Gomez et al. [12]. The full list of variables and the levels at which they were sampled are presented in Table 1.

Variable	**Units**	**Levels**
Surface air temperature	K	2 m above surface
Geopotential	m	500, 700 hPa
Wind zonal component	$m \cdot s^{-1}$	500, 700 hPa
Wind meridional component	$m \cdot s^{-1}$	500, 700 hPa
Wind speed	$m \cdot s^{-1}$	500, 700 hPa
Wind vorticity	s^{-1}	500, 700 hPa
Specific humidity	$kg \cdot kg^{-1}$	500, 700 hPa
Incoming solar radiation	$W \cdot m^{-2}$	top of the atmosphere
Soil water content	$m^3 \cdot m^{-3}$	0-7 cm depth

Table 1: Variables

Additionally, we also used three static fields:

- Latitude: $[-90, 90]$ degrees
- Longitude: $[0, 360]$ degrees
- Land-sea mask: $[0, 1]$ scalar field indicating the proportion of land in each grid cell.

The data were downloaded from the WeatherBench2 dataset [5] and are originally from the ERA5 reanalysis data from the European Centre for Medium-Range Weather Forecasts (ECMWF) [4].

Following the work of Weyn et al. [18], we regridded the data to a 6x48x48 gnomonic equiangular cubic sphere grid. This grid system has a few advantages:

1. It avoids pole singularities, which are problematic for many grid systems.
2. It gives a more uniform representation of the Earth's surface. With the latitude-longitude grid, data for the poles is over-represented, this is not the case with the cube sphere grid.
3. It allows the use of a special case of Convolutional Neural Networks (CNNs) for spatial data.

The cube sphere grid is based on a 3D cube where each face is a square of 48x48 pixels. In this grid system, spatial coordinates are the face number $f \in \{1, ..., 6\}$ and the pixel heights $h \in \{1, ..., 48\}$ and width $w \in \{1, ..., 48\}$ on the face. The grid is illustrated in Figure 5.

To regrid the data from the latitude-longitude grid to the cubic sphere grid, we used the "TempestRemap" software [21, 22]. It concretely generates a weight matrix that allows us to interpolate the data from one grid to the other. We also used k-nearest neighbors interpolation with $k = 4$ for faster and simpler regridding when necessary.

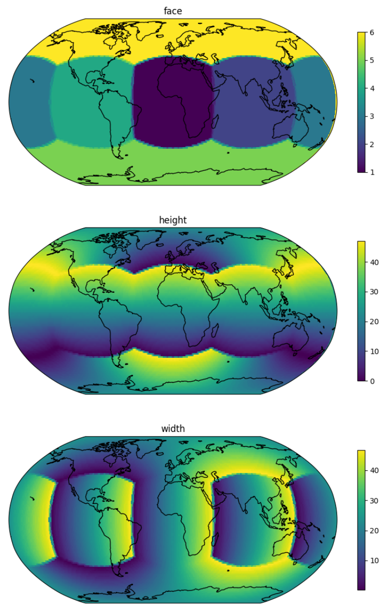
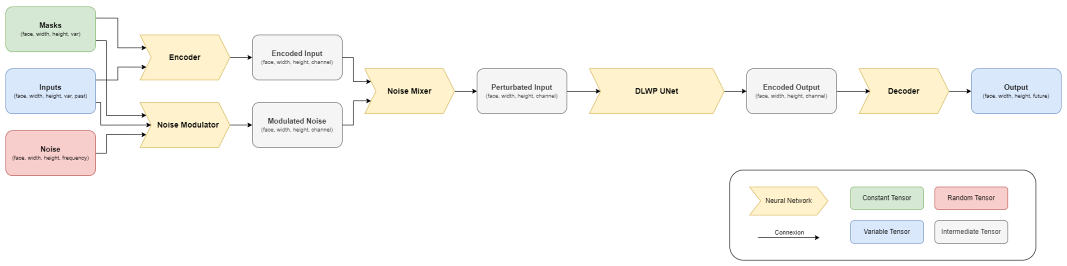


Figure 5: Cube Sphere Grid

B Model architecture

To experiment with the power mean aggregation method, we developed a simple generative model that could output an ensemble of local anomalies. Our implementation mainly follows the work of Weyn et al. [18] and Lopez-Gomez et al. [12]. Here is an overview of the model architecture, with more details in the following sections.



Masks
(face, width, height, var)
Inputs
(face, width, height, var, past)
Noise
(face, width, height, frequency)
Encoder
Noise Modulator
Encoded Input
(face, width, height, channel)
Modulated Noise
(face, width, height, channel)
Noise Mixer
Perturbated Input
(face, width, height, channel)
DLWP UNet
Encoded Output
(face, width, height, channel)
Decoder
Output
(face, width, height, future)
Neural Network
Constant Tensor
Random Tensor
Connexion
Variable Tensor
Intermediate Tensor

Figure 6: Model architecture overview

B.1 Deterministic baseline

For the needs of our study, we needed a relatively light ML model that could output local anomalies of surface air temperature at different forecast lead times (1 to 12 days). We chose to use a Convolutional Neural Network like model inspired by the work of Weyn et al. [18] and modified by Lopez-Gomez et al. [12]. Their models are based on the U-Net architecture [23] and adapted to the cube sphere grid (see appendix A).

Their main idea is to apply convolutions on each face of the cube sphere grid using shared kernel weights for equatorial and polar faces. This way, the model can learn different physics for poles and equator faces while maintaining a reasonable number of parameters.

In the U-Net part of the model, as the data goes down on resolution, the convolutions represent larger scales interactions and transport. The skipped connections, characteristic of the U-Net architecture, allow the model to maintain a high resolution representation of the data while still learning the large scale features.

The inputs of our models are atmospheric and ground variables at day d, $d-1$, ... $d-3$. They are scaled using the global climatology mean and standard deviation. This global scaling ensure that gradients are not distorted as they play a crucial role in the physics. As we want to predict the surface air temperature local anomaly, this variable is also given as input to the model. Given that the physics depends on the latitude, longitude and is significantly different on continents and oceans, we also give these three spatial masks as input to the model. Finally, the model outputs surface air temperature local anomalies at day $d+1$, $d+2$, ... , $d+12$.

B.2 Generative adaptation

To investigate ensemble aggregation methods, we needed a generative model that could output an ensemble of local anomalies. For that, we followed the work of Bi et al. [1] and added Perlin noise to the input of the deterministic baseline. We adapted the generation of this kind of noise to the cube sphere and modified it to better capture extremes. Also, we did not simply add the noise to the input but modulated it using the model itself.

Perlin noise is a type of gradient noise developed by Ken Perlin and introduced in 1985 in a paper called “An Image Synthesizer” [19]. On the contrary to white noise (pixel-wise Gaussian Noise), As shown in Figure 7, Perlin noise is smooth, continuous and has a spatial coherence that makes it suitable for generating natural-looking textures, or in our use case, weather variable fields.

To ensure continuity across the globe, we generated 3D Perlin Noise on the $[0,1]^3$ cube and selected the 2D slice corresponding to the Earth surface. An example of Perlin Noise generated on the globe is shown in Figure 8.

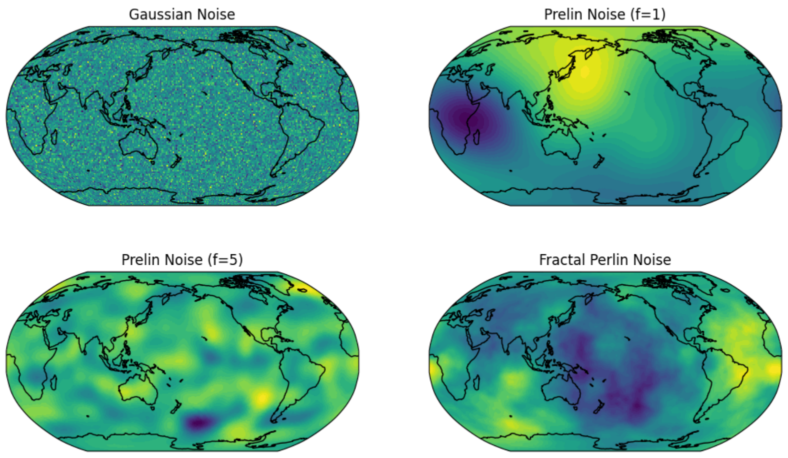


Figure 7: Gaussian Noise, Perlin Noises for different frequencies and Fractal Perlin Noise

The general approach to generating Perlin noise is detailed in Ken Perlin's 1999 talk [24] and illustrated on this website: `https://en.wikipedia.org/wiki/Perlin_noise#Algorithm_detail`.

The algorithm we concretely used was adapted from the work of Vigier [25]. Our contribution was to randomize the gradient vectors amplitude with a log-normal distribution to better capture extremes. Indeed, default Perlin Noise is bounded between -1 and 1, but for our use case, we wanted a noise that could go beyond these bounds.

To represent different scales of randomness, we generated Perlin noise at different frequencies and combined them to create a sort of fractal noise using a convolutional layer as an amplitude modulator. This noise modulator layer uses the same input as the deterministic model and learns how to generate a spatially coherent noise adapted to the data. Finally the noise is mixed into the input using a simple 1x1 convolutional layer as illustrated in Figure 6.

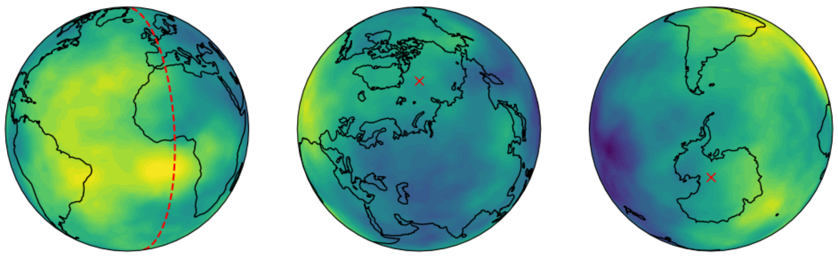


Figure 8: Generated Fractal Perlin Noise and its continuity on the globe

B.3 Model implementation

Our model is implemented in "Flax" [26] as a ML high-level API based on "JAX" [27], a numerical computing library. Following the work of Lopez-Gomez et al. [12], the different types of layers used in the model are:

- **Conv**: 2D dilated convolutions layer [28] with a kernel size of 3x3 or 1x1, a stride of 1 and a specified dilation factor. Dilation is used to increase the receptive field of the model without increasing the number of parameters. Depending on the kernel size and dilation factor, padding is added to maintain the same spatial resolution with data from other faces of the cube sphere grid. The notation Conv(32, 3x3, 1) means a convolutional layer with 32 filters, a kernel size of 3x3, and a dilation of 1.
- **PreLU**: Parametric ReLU activation functions with shared parameters along all dimensions except the channel axis.

- **MaxPooling**: 2x2 max pooling layers.
- **Reshaping**: Layer reshaping a tensor to a given shape.
- **Concat**: Concatenation of two tensors along the channel axis.
- **Product**: Element-wise product of two tensors.
- **Sum**: Sum of a tensor along a given axis.

The different blocks of our model, introduced in Figure 6, are made of the following layers:

Layer	Output shape
Input	(6,48,48,27)
Conv(32,3x3,1)	(6,48,48,32)
PreLU	(6,48,48,32)
Conv(32,3x3,1)	(6,48,48,32)
PreLU	(6,48,48,32)
Output	(6,48,48,32)

Table 2: Encoder

Layer	Output shape	Ref
Input	(6,48,48,27)	
Conv(64,3x3,1)	(6,48,48,64)	
PreLU	(6,48,48,64)	
Reshaping	(6,48,48,16,4)	[1]
Perlin Noise	(6,48,48,16,4)	[2]
Product [1] x [2]	(6,48,48,16,4)	
Sum [last axis]	(6,48,48,16)	
Output	(6,48,48,16)	

Table 3: Noise Modulator

Layer	Output shape
Input A	(6,48,48,32)
Input B	(6,48,48,16)
Concat [A] & [B]	(6,48,48,48)
Conv(32,1x1,1)	(6,48,48,32)
PreLU	(6,48,48,32)
Output	(6,48,48,32)

Table 4: Noise Mixer

Layer	Output shape
Input	(6,48,48,32)
Conv(12,1x1,1)	(6,48,48,12)
Output	(6,48,48,12)

Table 5: Decoder

Layer	Output shape	Ref
Input	(6,48,48,32)	[1]
MaxPooling	(6,24,24,64)	
Conv(64,3x3,1)	(6,24,24,64)	
PreLU	(6,24,24,64)	
Conv(64,3x3,2)	(6,24,24,64)	
PreLU	(6,24,24,64)	[2]
MaxPooling	(6,12,12,64)	
Conv(128,3x3,1)	(6,12,12,128)	
PreLU	(6,12,12,128)	
Conv(128,3x3,4)	(6,12,12,128)	
PreLU	(6,12,12,128)	
Conv(128,3x3,8)	(6,12,12,53)	
PreLU	(6,12,12,64)	[3]
Up-Sampling [3]	(6,24,24,64)	[4]
Concat [2] & [4]	(6,24,24,128)	
Conv(64,3x3,1)	(6,24,24,64)	
PreLU	(6,24,24,64)	
Conv(64,3x3,2)	(6,24,24,64)	
PreLU	(6,24,24,64)	
Conv(32,3x3,4)	(6,24,24,32)	
PreLU	(6,24,24,32)	[5]
Up-Sampling [4]	(6,48,48,32)	[6]
Concat [1] & [6]	(6,48,48,64)	
Conv(32,3x3,1)	(6,48,48,32)	
PreLU	(6,48,48,32)	
Conv(32,3x3,2)	(6,48,48,32)	
PreLU	(6,48,48,32)	
Conv(32,3x3,1)	(6,48,48,32)	
PreLU	(6,48,48,32)	
Output	(6,48,48,32)	

Table 6: DLWP U-Net

Our model contains approximately 1.2 million learnable parameters. In comparison, GraphCast [3] has around 36.7 million parameters.

B.4 Training procedure

To train our model to predict coherent ensembles of local anomalies, we used a probabilistic loss function: the Continuous Ranked Probability Score (CRPS) loss function. The CRPS is a proper scoring rule for probabilistic forecasts defined as:

$$\text{CRPS} = \int_{-\infty}^{+\infty} (\text{P}(\hat{x} \leq x) - \mathbb{1}_{x \leq x_i})^2 dx \tag{3}$$

where $\hat{x}$ is the predicted local anomaly seen as a random variable, x is the true local anomaly and $\mathbb{1}$ is the indicator function. Numerically, we approximated the loss as:

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} \left[\frac{1}{n} \sum_{j=1}^{n} |\hat{x}_{i,j} - x_i| - \frac{1}{2n^2} \sum_{j=1}^{n} \sum_{k=1} |\hat{x}_{i,j} - \hat{x}_{i,k}| \right] \tag{4}$$

where $1 \leq i \leq N$ represents a given time, location and forecast lead time and $1 \leq j \leq n$ the member index in the ensemble (n is the ensemble size). x_i is the true local anomaly and $\hat{x}_{i,j}$ are the predicted anomalies. This approximation comes from the fact that the CRPS can be expressed as:

$$\text{CRPS} = \mathbb{E}\left[|\hat{X} - x|\right] - \frac{1}{2}\mathbb{E}\left[|\hat{X} - \hat{X}'|\right] \tag{5}$$

where $\hat{X}$ and $\hat{X}'$ are two independent random variables with the same distribution as $\hat{x}$, according to the work of Gneiting and Raftery [29].

The model was trained using the Adam optimizer with a learning rate of 10^{-3} until the loss on the validation dataset stopped decreasing. The training was done on a single GPU with 16GB of memory and took a few hours.

C Results details

C.1 Overall predictive performances

A common way to evaluate the performance of a weather predictive model is to compute the Root Mean Squared Error (RMSE) between the predicted and true temperatures. For a given forecast lead time, we computed the RMSE across the whole test dataset as:

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{x}_i - x_i)^2} \tag{6}$$

where N is the number of members ($\simeq 365 \times 6 \times 48 \times 48 \simeq 5 \cdot 10^6$), $\hat{x}_i$ are the predicted global anomalies and x_i the true global anomalies. The lower the RMSE, the better the model.

We computed the RMSE of temperature global anomalies for different forecast lead times and different models:

- Persistence: a simple model that predicts the temperature at time $d + k$ to be the same as at time d.
- Climatology: always predicts the local mean temperature at any time of the year and location.
- Our model: the generative model we introduced that outputs an ensemble of local anomalies. The mean prediction is considered for this RMSE computation.
- GraphCast: GraphCast model [3] that outputs a single prediction.

Results are presented in Figure 9.

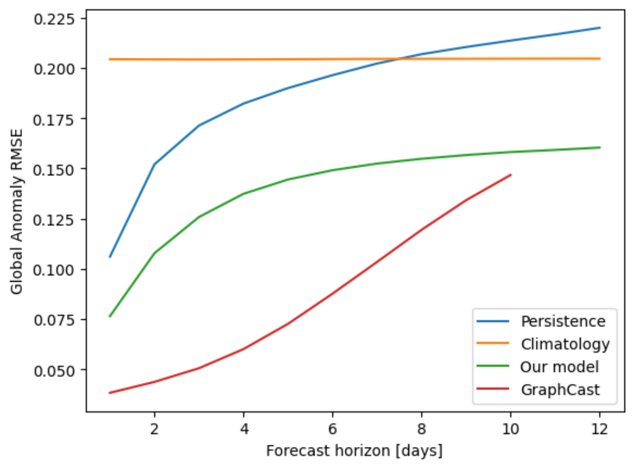


Figure 9: Global anomaly RMSE on test dataset

As expected, our generative model performs better than the persistence baseline and remained predictive even up to 12 days ahead, as its RMSE stays below the climatology reference. Also, and not surprisingly, its RMSE was always higher than GraphCast's, a more complex model well-suited for short-term weather forecasting.

C.2 Optimal power exponent

To find the optimal power exponent p_{opt} for a given quantile threshold q, we computed the AUC for different values of p on the validation dataset (2010-2015). As mentioned in the main text, we fixed the forecast lead time to 7 days for simplicity.

We defined the optimal power exponent p_{opt} as the value of p that maximizes the AUC and the relative improvement as:

$$\text{RI} = 100 \times \frac{\text{AUC}_p - \text{AUC}_{\text{mean pred}}}{\text{AUC}_{\text{mean pred}}}$$

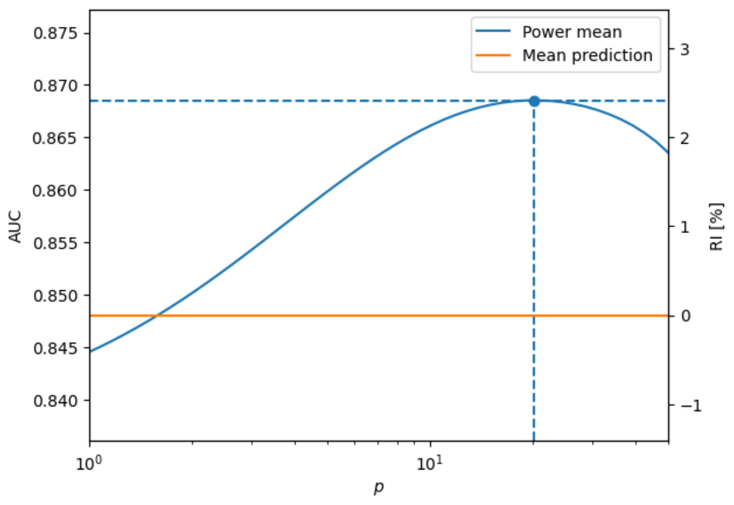


Figure 10: AUC $= f(p)$ for $q = 0.9$

For the curve presented in Figure 10 ($q = 0.9$), the optimal power exponent is $p_{opt} \simeq 18.3$ and the relative improvement is $\text{RI}_{opt} \simeq 2.67\%$. We can also see that the AUC for $p = 1$ is not the same as the AUC for the mean prediction method, because of the non-linearity of the normal distribution quantile function Φ.

Repeating this process for different quantiles, we obtained the following results:

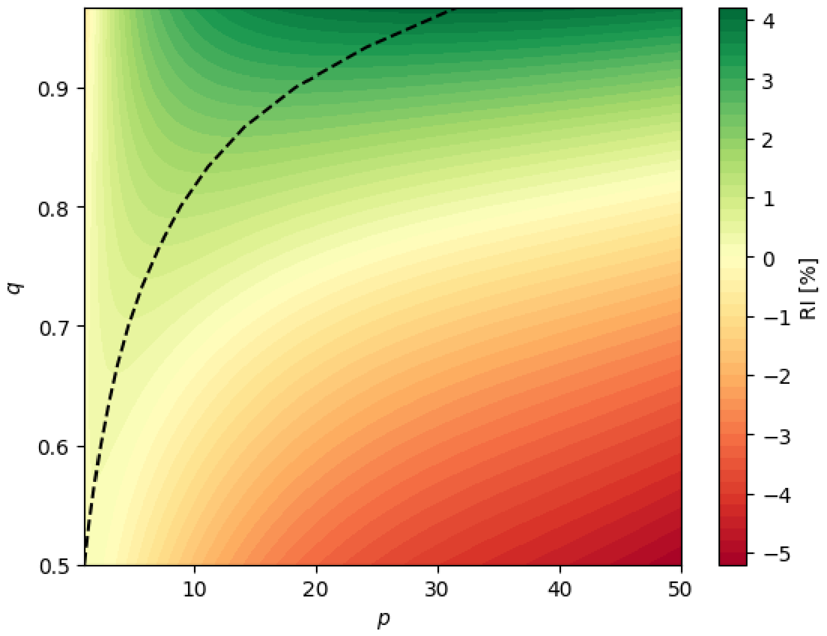


Figure 11: RI $= f(p, q)$

From this plot, we can see that the optimal power exponent p_{opt} seems to be exponentially increasing with the quantile threshold q. This assumption is confirmed by the linear regression presented in the main section in Figure 2. It also suggests that the relative improvement RI increases with the quantile threshold q, the result also confirmed on the test dataset as shown in Figure 3.