Deep Learning on the Sphere for Multi-model Ensembling of Significant Wave Height

Andrea Littardi[1,2], *Anders Hildeman*[3], and *Mihalis A. Nicolaou*[2]

[1]Offshore Monitoring Ltd., Cyprus
[2]Computation-based Science and Technology Research Center, The Cyprus Institute, Cyprus
[3]Dept. of Space, Earth and Environment, Chalmers University of Technology, Sweden

***Abstract*—When working with geophysical variables on a global scale, a solution for processing data on the surface of a sphere is needed. At the same time, region-specific dynamics that deviate from the general behavior across the globe also need to be accounted for. Addressing these two necessities, we propose the first Deep Learning approach for multi-model ensembling that operates directly on the sphere. Our methodology allows to progressively allocate region-specific model complexity, guided by the clustering of the model forecasting errors. We evaluate our proposed method on a multi-model ensembling application of significant wave height, where the proposed method is shown to outperform 2D CNNs with less than half the parameters needed, while producing comparable results to models with more than 10 times the number of parameters.**

***Index Terms*—Multi-model ensembling, Geoscience, Deep learning, Clustering.**

I. Introduction

Human activities are significantly affected by meteorological and climatological events, making modelling and processing geospatial data an area of great scientific interest, with a far-reaching impact in a wide range of applications [1, 2, 3].

In this work we address the problem of improving the forecast of geophysical variables (e.g., climatological or meteorological) through multi-model ensembling (MME), a process that combines information extracted from multiple forecasts, i.e. the *forecast ensemble*, in order to alleviate individual biases and uncertainties in the forecasts [4]. While the simplest way to combine multiple forecasts may rely on taking the average of the ensemble at each location, this process does not address errors that can be systematic and where some forecasts models may be more reliable than others at specific locations. MME was first proposed in [5], where linear regression was utilized for medium range weather, seasonal climate and hurricane forecasts. Locally-weighted learning algorithms were also introduced in [6], in order to alleviate issues that arise from misalignment of forecast locations in different models.

The rising impact of Deep Learning in earth observation data [7] has since led to several methods being proposed that utilize Deep Learning. Such methods benefit from desirable properties of Convolutional Neural Networks (CNNs), such as capturing non-linear patterns irrespective of their location in a shift-invariant manner, while leveraging information from neighbouring locations. Typically CNN-based models outperform their linear counterparts in this task. For instance, the applicability of CNNs to MME in the context of improving precipitation forecasts resolution on a regional scale was studied in [8, 9] and for medium range wind forecasts at a global scale in [6].

Undoubtedly, the adoption of Deep Learning has brought on significant progress in MME. However, straightforwardly adopting CNN-based architectures leaves two main problems unresolved, which serve as the primary motivation for this paper. Firstly, the distortions introduced when working with data on a manifold such as the surface of a sphere directly undermine the assumption of equal distance and direction between neighbouring pixels, lying at the base of CNNs which assume an underlying Euclidean space. Secondly, the presence of location-specific dynamics makes the problem at hand non-stationary over the spatial domain, thus limiting the value of the weight-sharing property of CNNs, and failing to capture patterns that may arise in a region-specific manner.

To tackle these problems, in this paper we propose a novel deep learning architecture for MME that operates directly on the sphere using mesh convolutions, greatly reducing the number of trainable parameters and thus the resources required to train the models. Furthermore, the proposed method is able to capture region-specific patterns by progressively increasing the complexity (depth) of the network in high-loss regions. This only requires fine-tuning the additional layers and keeps the model efficient, without resorting to solutions that would lead to a vast increase of trainable parameters. In summary, the contributions of this work are listed in what follows:

- We present, to the best of our knowledge, the first Deep Learning approach for multi-model ensembling that operates directly on the sphere, avoiding the issues introduced by the equirectangular projection to the plane.
- To accommodate for local or region-specific phenomena, the proposed method facilitates progressively growing the complexity of the network to model local patterns that do not generalize across locations.
- With a set of extensive experiments on MME for predicting significant wave height, we show that (1) our method outperforms convolutional neural networks by 23% while requiring less than half the trainable parameters, while (2) the proposed method produces comparable results with the best compared method while needing only a small fraction of the number of parameters ($\approx 7\%$).

II. Related Work

A primary concern originates from when signals on the surface of a sphere are mapped to pixels on a plane as the spatial proportions cannot be retained — resulting in deformations when moving away from the equator and becoming severe in the polar regions. Solutions proposed in literature include sampling schemes that correct for spherical distortions [10, 11], but leave over-representation of pixels around the poles unaddressed - leading to evaluation issues. A different approach lies in directly applying convolutions on manifolds, such as a mesh [12, 13, 14]. In [14], a parametrized differential operator (MeshConv) is introduced, that replaces the convolutional kernel in CNNs. The operator can represent the spatial dependencies between nearby locations, and can be defined on arbitrary Riemannian manifolds such as spheres — in contrast to traditional CNNs. In this work, we will utilize this MeshConv architecture since it allows for efficient models on the sphere with significantly reduced number of parameters while at the same time solving problems arising from the orientation of convolutional filters on a spherical surface.

Another limitation of CNNs when dealing with geophysical variables lies in the manifestation of location-dependent patterns. A solution lies in introducing locally-connected layers [8] — however this solution does not consider filters that can be shared across regions, and would greatly increase the number of parameters proportionally to the number of locations considered. Allocating model complexity in an automatic manner has been explored in several works, for example by employing attention mechanisms [15, 16, 17, 18]. In more detail, in [18] an auxiliary module classifies different regions of the input image in order to apply a specific filter to each of them. However, their model implies stationarity in the distribution of classes over the spatial domain. For this reason, we are taking a different approach, associating each class with a region of the world using a residual-based angular-penalty clustering approach. Furthermore, we allow the classes to share much of the network complexity, making the model cheap in terms of trainable parameters.

III. Methodology

Basic Architecture. Our method builds on the MeshConv operator. Following [14] we replace convolutional kernels with parametrized differential operators of varying orders, that can represent spatial dependencies between nearby locations as

$$\mathcal{R} = w_0 I + w_1 \nabla_{lat} + w_2 \nabla_{lon} + w_3 \nabla^2, \tag{1}$$

where ∇_{lat} denotes the directional derivative in the north-south direction, ∇_{lon} the directional derivative in the west-east direction, and ∇^2 the Laplace-Beltrami operator, leading to only 4 parameters, w_i, per layer. Furthermore, we employ an encoder-decoder based architecture, using residual blocks [14] that include the operators in Eq. 1, as well as batch normalization layers, residual connections, and ReLU activation functions. We discretize the spherical domain by employing the icosahedral spherical mesh (icosphere) [19], a discrete approximation of the sphere denoted by the set of vertices $\mathcal{V}$ and edges $\mathcal{E}$.

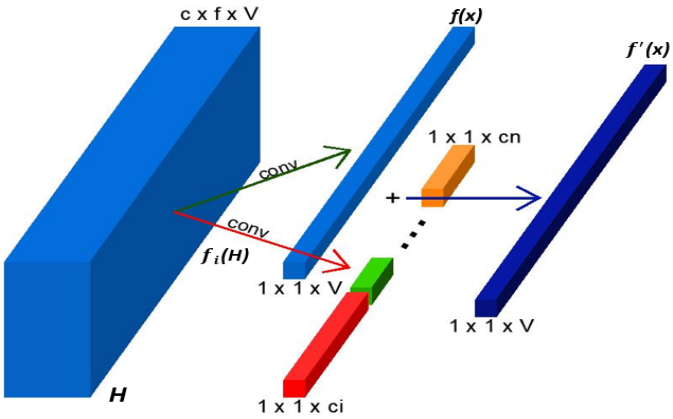

Fig. 1: Locally increasing model complexity: light blue blocks are tensors in baseline architecture. Red arrow indicates convolutions added for each cluster, and is applied on the output of the last hidden layer $\mathbf{H}$ (ci and cn indicate the number of vertices in cluster i and n). Activations arising from the new layer are added to the original output tensor (blue arrow).

Hence, our input (and output) data is first projected from the 2d lat-lon grid onto the vertices of the icosphere. We subsequently train the architecture using a suitable loss function $\mathcal{L}$, which in our case is the Mean Squared Error (MSE).

We denote this base architecture with $\boldsymbol{f}$, having weights $\mathbf{w}$ and taking as input a matrix $\mathbf{X} \in \mathbb{R}^{m \times |\mathcal{V}|}$, where m is the number of input channels corresponding to forecasts from each model. The output of $\boldsymbol{f}$ is a vector of predicted values, $\hat{\mathbf{y}}$, of size $|\mathcal{V}|$, i.e., the number of vertices. Likewise, we denote the true values for each vertex as $\mathbf{y}$. The superscripts *train* and *val* are used to refer to the training and validation datasets.

Residual-based Thresholding. We start by inspecting the residuals on the validation dataset given some vertex-wise error metric, $L_v := L(\hat{\mathbf{y}}_v^{val}, \mathbf{y}_v^{val})$. In this way we can identify locations where errors tend to be large. We define a threshold value T on the vertex-wise loss and denote the set of thresholded vertices with $\mathcal{V}^*$, i.e.,

$$\mathcal{V}^* = \{v : L_v > T, v \in \mathcal{V}\}. \tag{2}$$

Angular-penalty Clustering. For spatial data, and geophysical variables in particular, it is generally the case that nearby locations exhibit higher dependencies than distant ones. This results in high-loss vertices generally appearing in geographically clustered regions, as we can see from Figure 2 from the example below. The high loss in the clustered regions may be because of two reasons: either the forecasts do not hold the information necessary for good predictive performance (inherently hard to predict), or the model and data is not complex enough to capture patterns occurring in the region (model misspecification). The first case leaves no room for improvement. In the latter case, one would want to enhance the model capacity for the specific clusters of problematic regions. As the geography remains constant over time, we expect the problematic regions to remain the same

also for future, unseen, data. Hence, we want to separate $\mathcal{V}^*$ into n spatially local clusters $\{\mathcal{C}_i\}_{i=1}^n$. In order to find these cluster groups, we start with picking a random vertex from $\mathcal{V}^*$ and assigning that one as the master vertex of group 1, denoted as p_1. We then choose a second vertex, p_2, as the one maximizing the angular distance to p_1, i.e.,

$$p_2 = \arg\max_{p_2 \in \mathcal{V}^*} d(p_1, p_2). \tag{3}$$

Vertex p_2 is assigned to be the the master vertex of the second group. All vertices in $\mathcal{V}^*$ can now be classified as being part of the group for which they have the smallest angular distance to, i.e.,

$$\mathcal{C}_i := \{v : d(v, p_i) \leq d(v, p_j), \forall j \in \{1, \ldots, n\} \setminus \{i\}, v \in \mathcal{V}^*\}. \tag{4}$$

In the presence of a tie, the vertex is assigend to the cluster group with the smallest index number.

The centroids of each group, $\{c_i\}_{i=1}^n$, are then computed (the average location of the vertices belonging to the same group projected onto the surface of the globe), and vertices are classified again to minimize the angular distance to the centroids instead of master vertices. From here on, the procedure of adding groups is iterated until all vertices in any group are no further away from the group centroid than a given angular distance θ, i.e, until

$$\max_{i \in \{1,\ldots,n\}} (\max_{v \in \mathcal{C}_i}(d(v, c_i))) \leq \theta. \tag{5}$$

We thus acquire n cluster groups, each one associated with a mutually exclusive set of vertices $\{\mathcal{C}_i\}_{i=1}^n$. Figure 2c shows an example of the classification of $\mathcal{V}^*$ with angular constraint θ set to 20 degrees.

Learning Region-specific Patterns. Having identified clusters that describes regions of interest, we adapt the architecture of the network in order to improve the prediction over each cluster. This is achieved by introducing an additional MeshConv layer, f_i, with weights $\boldsymbol{w_{ci}}$ for each cluster i, and adding the sum of f_i to the output of $f(\boldsymbol{X})$ for each cluster group. f_i outputs a vector of size $|\mathcal{V}|$ which elements are zero for vertices that are not part of $\mathcal{C}_i$. Hence, the final prediction over the whole set of vertices, $\mathcal{V}$, are given by the network,

$$f'(\boldsymbol{X}, \{\mathcal{C}_i\}_{i=1}^n) = f(\boldsymbol{X}) + \sum_{i=1}^{n} [\mathbb{I}(k \in \mathcal{C}_i) f_i(\boldsymbol{H})_k]_{k=1}^n. \tag{6}$$

Here, $\mathbb{I}$ is the indicator function, taking the value of 1 if the statement in its argument is true and 0 otherwise. The tensor, $\mathbf{H}$, is the values at the last hidden layer of the f network. The index k denotes a specific element of a vector and the notation $[\cdot]_{k=1}^{|\mathcal{V}|}$ defines a vector by the values of its $|\mathcal{V}|$ elements. In this way, the new auxiliary network is designed to correct the residual of the baseline network to compensate for regional biases. Figure 1 shows a graphical example of the resulting architecture. Finally, having formulated the cluster-specific architectures, we can train the additional (cluster specific) weights $\boldsymbol{w_c}$, freezing the baseline network weights $\boldsymbol{w}$.

(a) MSE. (b) Thresholded (c) Clusters

Fig. 2: Geographical regions associated with large validation errors and the acquired clusters.

Without loss of generality, in the proposed method we defined the region-specific auxiliary architecture of f_i to be a single MeshConv operation in order to correct the residual from the prediction produced by f. However, to increase the receptive field and add further model complexity, multiple layers and activation functions could be stacked also to the region-specific architectures, $\{f_i\}_{i=1}^n$. When θ tends to zero we obtain a solution where each vertex in $\mathcal{V}^*$ becomes a cluster - similarly to employing locally connected layers [8] with no weight sharing.

IV. Experiments

In this section we describe the data used, introduce the parameterization of the models and present the results.

Data. Significant wave height is a parameter representing a summary statistic of the sea elevation at a point in time and space. As a proxy for the true significant wave height we are using the "Significant Height of Combined Wind Waves and Swell" variable from the ECMWF ERA-5 reanalysis dataset [20]. Reanalysis datasets are generally considered to be an accurate representation of the studied variables, being models assimilated with both satellite and in-situ observations.

As the input, to predict the ground truth from, we are using forecasts from three forecasting models:

- SWH from "Global Ocean Waves Analysis and Forecast" provided by CMEMS [21].
- VHM0 from "Global GWAM" provided by DWD [22].
- HTSGW from "GFS version 16" provided by NCEP [23].

We use data from February to September 2021, taken with a temporal frequency of 3 hours. The data is separated into a training set (first 70% of time instances), validation set (70%-85%), and test set (85%-100%). We use data with a forecast lead time of five days (the time interval between when the forecast was produced and the time being forecasted), in order to have relevant disagreement between the three input forecasts.

Global data are usually stored in a geographical coordinate system on a 2D grid with latitude and longitude on its axes. Before being passed to the model, we first project the data onto the surface of the icosphere using nearest neighbour interpolation. We use a level-7 icosphere [14] (icosahedron after 7 subsequent edge splits), resulting in 163'842 vertex locations.

Experimental setting. We are comparing 6 alternative models. First, we compute the ensemble mean, that is, the

TABLE I: Absolute and relative performance of compared methods in terms of MSE.

Method	Parameters	all locations	Cluster locations	Non-Cluster locations
Ensemble mean	0 *0 %*	15.5176 *13073 %*	87.8057 *12414 %*	14.7876 *13109 %*
2D convolutional NN	55031 *16.8 %*	0.1494 *125.9 %*	0.8887 *125.6 %*	0.142 *125.9 %*
MeshConv 8 channels	21553 *6.6 %*	0.1222 *102.9 %*	0.7475 *105.7 %*	0.1158 *102.7 %*
MeshConv 32 channels	328129 ***100 %***	0.1187 ***100 %***	0.7073 ***100 %***	0.1128 ***100 %***
Local Clusters $\theta = 20°$	21949 *6.7 %*	0.1215 *102.4 %*	0.6838 ***96.7 %***	0.1158 *102.7 %*
Local Clusters $\theta = 5°$	23104 *7 %*	0.1215 *102.4 %*	0.6807 ***96.2 %***	0.1158 *102.7 %*

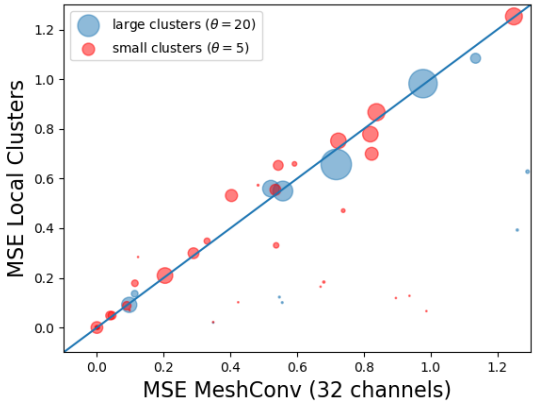

Fig. 3: Error improvement over cluster regions. Each circle identifies a cluster. Circle size is proportional to the number of vertices in the cluster.

simple per-vertex average of the three input models. Second, we compare with the 2D CNN presented in [6]. This is based on dilated convolutions and skip connections. We then utilize two different configurations of the architecture presented in [14]. A "light" version, using 8 convolutional channels at each hidden layer. A deeper version, using 32 hidden channels, to use all the available memory of a 32 GB GPU. Both architectures start from a icosphere resolution of level 7 and we use level 4 as the lowest representation in the encoder. We then compare 2 alternatives of our proposed method. In this case we are using the "light" MeshConv architecture, adding an extra output layer for each cluster, having 33 additional parameters (4 features x 8 channels + 1 bias), whose output gets added to the original prediction as shown in Figure 1. We group the same 1'638 vertices (1% of 163'842) with an angular constraint, θ, equal to 20°, resulting in 12 clusters shown in Figure 2c, and then equal to 5°, resulting in 38 smaller clusters. It is worth noting that, while all tests have been performed using a single processor on a GPU Tesla V100 32BG, this architecture is highly parallelizable, making it possible to train region specific architectures on different hardwares.

Results. Table I shows the comparison of the MSEs over the test dataset for the 6 models. Percentage values are relative to the MeshConv "deep" model, the one with the highest number of parameters. Notice, the number of parameters, given in table I is a measure of complexity for the models. Column "all locations" shows the average loss across all locations (except land). "Cluster locations" refers to the set of problematic locations for which the extra convolutions were activated, and "Non-Cluster locations" the result on all remaining ones. We can see that all methods requiring training greatly outperform the Ensemble mean. Also, the MeshConv models, processing data directly on the sphere, perform better than the 2D-plane alternative. The values in bold show how the "light" networks with specific cluster-allocated convolutions performs better than the deeper model on the clustered regions, while having much fewer parameters (21'949 and 23'104 parameters respectively, compared to 328'129 parameters). Adding a single extra convolutional layer over the 12 cluster regions reduced the error by 8.5% from 0.7475 to 0.6838. The loss remains unchanged by construction for the remaining locations. Figure 3 shows a breakdown of the improvement of the loss for each individual cluster. On the horizontal axis we show the MSE for the model with the most parameters, the MeshConv with 32 channels. On the vertical axis, are the losses for the locally clustered models. In blue are the 12 clusters using $\theta = 20$, in red the 38 ones using $\theta = 5$. We can see how points are mostly below the line, with smaller clusters witnessing the largest performance improvement. In a few instances, the deeper non-local model performs better. This can be either due to the inability of the lighter one to capture general patterns occurring at those locations (underfitting), or to the added local convolutions not being able to generalize to the test dataset over those locations, more likely in the case of the smallest clusters (overfitting).

V. Conclusion

In this work we present the first Deep Learning approach for MME that operates directly on the sphere while progressively increasing the complexity of the network to model regional patterns. By addressing core limitations of 2D CNNs in this problem domain the proposed method outperforms 2D CNNs, while needing less than half the number of parameters. Furthermore, by focusing model capacity on problematic regions, the results are comparable to those of a global MeshConv network with 10 times as many parameters.

Acknowledgment

We wish to thank the Ecosail project for partially funding this research. The project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 820593. This work was supported by a grant from The Cyprus Institute on Cyclone under project ID p055. ERA-5 [20] was downloaded from the Copernicus Climate Change Service (C3S) Climate Data Store. This study has been conducted using E.U. Copernicus Marine Service Information.

References

[1] M. Dell, B. F. Jones, and B. A. Olken, "What do we learn from the weather? the new climate-economy literature," *Journal of Economic Literature*, vol. 52, no. 3, pp. 740–98, 2014.

[2] D. Diaz and F. Moore, "Quantifying the economic risks of climate change," *Nature Climate Change*, vol. 7, no. 11, pp. 774–782, 2017.

[3] P. De Girolamo, M. Di Risio, G. M. Beltrami, G. Bellotti, and D. Pasquali, "The use of wave forecasts for maritime activities safety assessment," *Applied Ocean Research*, vol. 62, pp. 18–26, 2017.

[4] C. Tebaldi and R. Knutti, "The use of the multi-model ensemble in probabilistic climate projections," *Philosophical transactions of the royal society A: mathematical, physical and engineering sciences*, vol. 365, no. 1857, pp. 2053–2075, 2007.

[5] T. Krishnamurti, C. M. Kishtawal, T. E. LaRow, D. R. Bachiochi, Z. Zhang, C. E. Williford, S. Gadgil, and S. Surendran, "Improved weather and seasonal climate forecasts from multimodel superensemble," *Science*, vol. 285, no. 5433, pp. 1548–1550, 1999.

[6] E. Larsson, A. Littardi, M. Nicolaou, L. Eriksson, and W. Qazi, "Multimodel superensembling using convolutional neural networks," 2020.

[7] M. Reichstein, G. Camps-Valls, B. Stevens, M. Jung, J. Denzler, N. Carvalhais, *et al.*, "Deep learning and process understanding for data-driven earth system science," *Nature*, vol. 566, no. 7743, pp. 195–204, 2019.

[8] E. R. Rodrigues, I. Oliveira, R. Cunha, and M. Netto, "Deepdownscale: a deep learning strategy for high-resolution weather forecast," in *2018 IEEE 14th International Conference on e-Science (e-Science)*, pp. 415–422, IEEE, 2018.

[9] C. Chaudhuri and C. Robertson, "Cligan: A structurally sensitive convolutional neural network model for statistical downscaling of precipitation from multi-model ensembles," *Water*, vol. 12, no. 12, p. 3353, 2020.

[10] M. Eder, T. Price, T. Vu, A. Bapat, and J.-M. Frahm, "Mapped convolutions," *arXiv preprint arXiv:1906.11096*, 2019.

[11] B. Coors, A. P. Condurache, and A. Geiger, "Spherenet: Learning spherical representations for detection and classification in omnidirectional images," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 518–533, 2018.

[12] T. Cohen, M. Weiler, B. Kicanaoglu, and M. Welling, "Gauge equivariant convolutional networks and the icosahedral cnn," in *International Conference on Machine Learning*, pp. 1321–1330, PMLR, 2019.

[13] Y. Lee, J. Jeong, J. Yun, W. Cho, and K.-J. Yoon, "Spherephd: Applying cnns on a spherical polyhedron representation of 360deg images," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9181–9189, 2019.

[14] C. Jiang, J. Huang, K. Kashinath, P. Marcus, M. Niessner, *et al.*, "Spherical cnns on unstructured grids," *arXiv preprint arXiv:1901.02039*, 2019.

[15] X. Jia, B. De Brabandere, T. Tuytelaars, and L. V. Gool, "Dynamic filter networks," *Advances in neural information processing systems*, vol. 29, pp. 667–675, 2016.

[16] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei, "Deformable convolutional networks," in *Proceedings of the IEEE international conference on computer vision*, pp. 764–773, 2017.

[17] M. Jaderberg, K. Simonyan, A. Zisserman, *et al.*, "Spatial transformer networks," *Advances in neural information processing systems*, vol. 28, pp. 2017–2025, 2015.

[18] J. Chen, X. Wang, Z. Guo, X. Zhang, and J. Sun, "Dynamic region-aware convolution," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8064–8073, 2021.

[19] J. R. Baumgardner and P. O. Frederickson, "Icosahedral discretization of the two-sphere," *SIAM Journal on Numerical Analysis*, vol. 22, no. 6, pp. 1107–1115, 1985.

[20] H. Hersbach, B. Bell, P. Berrisford, G. Biavati, A. Horányi, J. Muñoz Sabater, J. Nicolas, C. Peubey, R. Radu, I. Rozum, *et al.*, "Era5 hourly data on single levels from 1979 to present," *Copernicus Climate Change Service (C3S) Climate Data Store (CDS)*, vol. 10, 2018.

[21] "Copernicus marine service, product identifier: Global_analysis_forecast_wav_001_027."

[22] "Deutscher wetterdienst, gwam."

[23] "National center for environmental prediction, gefs - wave - global."