Contents lists available at ScienceDirect

Environmental Modelling and Software

journal homepage: www.elsevier.com/locate/envsoft

Coastal morphodynamic emulator for early warning short-term forecasts

Willian Weber de Melo[a,*], José Pinho[a], Isabel Iglesias[b]

[a] Centre of Territory, Environment and Construction (CTAC), School of Engineering, University of Minho, Campus de Azurém, Guimarães, 4800-058, Portugal
[b] Interdisciplinary Centre of Marine and Environmental Research (CIIMAR/CIIMAR), University of Porto, Terminal de Cruzeiros do Porto de Leixões, Av. General Norton de Matos, Matosinhos, 4450-208, Portugal

ARTICLE INFO

Handling Editor: Daniel P Ames

Original content: Deep learning model for XBeach morphodynamic emulation (Original data)

Keywords:
Numerical model emulator
XBeach
Deep learning
Tensorflow
Morphodynamics
Hydrodynamics

ABSTRACT

The use of numerical models to anticipate the effects of floods and storms in coastal regions is essential to mitigate the damages of these natural disasters. However, local studies require high spatial and temporal resolution numerical models, limiting their use due to the involved high computational costs. This constraint becomes even more critical when these models are used for real-time monitoring and warning systems. Therefore, the objective of this paper was to reduce the computational time of coastal morphodynamic models simulations by implementing a deep learning emulator. The emulator performance was evaluated using different scenarios run with the XBeach software, which considered different grid resolutions and the effects of a storm event in the morphodynamic patterns around a breakwater and a groin. The morphodynamic simulation time was reduced by 23%, and it was identified that the major restriction to reducing the computational cost was the hydrodynamic numerical model simulation.

1. Introduction

Forecasting the impacts of extreme events, such as droughts, floods, and storms, is fundamental to identify threatened areas and to allow early actions to increase resilience and reduce the eventual related damages. Anticipating extreme events becomes even more relevant in low-lying coastal areas, which present high risks due to the combination of meteo-oceanic events, climate trends and human activities. In open ocean coastal areas, cities and other settlements are more vulnerable to coastal flooding, whilst pluvial floods are one of the major threats in deltas and estuaries (Pörtner et al., 2022). Moreover, these areas shelter approximately 11% of the global population, which lives within 10 m above mean sea level (Haasnoot et al., 2021; Pörtner et al., 2022; Vousdoukas et al., 2020).

To anticipate and mitigate extreme events impacts, the implementation of forecasting systems must be addressed. These systems are usually implemented based on numerical models, such as XBeach and Delft3D (D3D) (Deltares, 2018; Roelvink et al., 2009), which had demonstrated to be accurate in predicting hydro-morphodynamic (HMD) variables under extreme events scenarios (Ferreira et al., 2019; Iglesias et al., 2022; Simmons and Splinter, 2022; Vousdoukas et al., 2012).

However, these numerical models demand considerable simulation computational time, mainly when applied to high spatial and/or temporal resolution domains and when different modules are used in a forecast simulation (*e.g.*, hydrodynamics, waves, winds, sediments, and morphology). However, the use of high-resolution models is crucial to correctly simulate hydrodynamic and morphodynamic patterns at coastal stretches. Moreover, real-time monitoring and warning systems require computationally efficient numerical models to optimize the prediction of future trends and to allow a faster response by competent authorities. To improve the computing efficiency of numerical models like XBeach, Rautenbach et al. (2022) compared the performance of the software in hydrodynamics and wave dynamics simulations using a CPU and a GPU, finding that even a desktop grade GPU can compete with the computational efficiency of high-performance computing CPU facilities. However, there are still few studies about the computational efficiency of HMD numerical models and ways to improve it.

An alternative to reduce the computational costs of HMD numerical models is the use of emulators, which are based on the application of statistical techniques that surrogate the numerical model. These are artificial intelligence-derived methods, namely machine learning (ML) and deep learning (DL), which implementation for surrogating morphodynamic numerical models had been approached by other authors.

* Corresponding author.
E-mail address: id9257@alunos.uminho.pt (W. Weber de Melo).

https://doi.org/10.1016/j.envsoft.2023.105729
Received 2 May 2023; Accepted 16 May 2023
Available online 17 May 2023

Poelhekke et al. (2016) developed a XBeach emulator to predict hazards at Faro beach, Anderson et al. (2021) used several numerical models and a Gaussian Process Regression (GPR) emulator to predict wave runup and overwash depths and Gharagozlou et al. (2022) developed a GPR-based emulator for post-storm subaerial beach and dune profile shapes. However, such works developed emulators for specific punctual locations within the domain. Hence, this work intends to fill this gap solving the emulators constraint in morphodynamic variables forecasting at full 2DH domains.

In a previous work of the authors, a DL model with convolutional layers, was implemented using simulation results obtained with D3D software (Weber de Melo et al. (2022). The developed technique used images of a D3D numerical model hydrodynamic variables as input and was able to forecast the long-term morphodynamic evolution of an intertidal shoal estuary, demonstrating that it was possible to reduce the total simulation time by surrogating the morphodynamic module of the numerical model for an emulator. However, short-term morphodynamics in beaches are more complex than long-term estuarine morphodynamics. Estuarine morphodynamics main drivers are the water currents, which varies according to tides and freshwater flows. On the contrary, beach morphodynamics results from the complex interactions between winds, non-linear waves processes, tides, water currents and return flows, and the processes involved in coastal morphodynamics are dominated by complex onshore and offshore transport. Thus, the numerical model selected in Weber de Melo et al. (2022) is not well-suited to model sand beaches bar behaviour because the processes of dune erosion and offshore sediment transport by the return flow are not included in the numerical code. On the other hand, XBeach software has all necessary processes to model bar behaviour (Trouw et al., 2012). These authors also note that the default values for wave-related bed load and suspended load factors in D3D are too high and, this way, the physics of the model are likely flawed.

In this context, this paper has the objective to improve the DL model architecture by Weber de Melo et al. (2022) and test if this improved architecture could reproduce the morphodynamic results of an experiment run with the XBeach model considering an extreme event scenario. The objective is to demonstrate if the new emulator, that successfully surrogated the D3D morphodynamic module for an estuarine environment, could also be applied to XBeach simulations of non-linear wave dominated coastal stretches. Two simulation sets were used to evaluate the XBeach emulator. The first set used a simplified linear bathymetry to assess if a change in the up-sampling operation in the neural network could reduce the error in the validation dataset and to determine the total simulation time reduction. The second simulation set tested if the emulator could reproduce the morphological evolution of a more complex coastal stretch involving a sandy beach with different coastal defence structures, including a submerged breakwater and a groin.

2. Methodology

The methodology applied in this study was adapted and improved from the previous work of Weber de Melo et al. (2022), which designed and implemented an HMD numerical model emulator to forecast the long-term morphological evolution of an estuarine area. However, as an already mentioned in the introduction, the processes that affect the morphodynamics in estuaries are different from those that occur in coastal areas. In the first case, the main morphological evolution driver is the water currents generated by river flows and tides. In the second case, the waves, tides and winds are the main drivers that affect the coastal morphological evolution. Due to the complexity of the morphological processes in coastal regions and the need to introduce other non-linear variables besides the water current, this work developed a new methodology to reduce the computational costs of a XBeach model to emulate the short-term morphological evolution of coastal stretches.

2.1. *XBeach model*

XBeach is an open-source numerical model that simulates HMD processes in sandy coastal areas by solving 2DH equations for wave propagation, flow, sediment transport and bottom changes (Roelvink et al., 2009). This software solves the shallow water equation for low-frequency waves and average flows considering the depth-averaged generalized lagrangian mean formulation (Roelvink et al., 2009). The sediment transport equation was solved accordingly to the Van-Thiel-Van Rijn formulation (van Rijn, 2007; van Thiel de Vries, 2009).

XBeach-based methodologies had already been demonstrated to be capable of achieving reliable results, having been applied to study the uncertainties in run-up predictions on natural beaches (e.g., Rutten et al., 2021), to predict extreme offshore directed sediment transport (Suzuki and Cox, 2021) and to model the morphological response of a sandy barrier island during hurricane conditions (Smallegan et al., 2016). However, XBeach simulations require intensive computational resources, mainly when the domain has a high spatial resolution, restraining its application in real-time early warning forecasting and decision support systems (Ferreira et al., 2019; Gharagozlou et al., 2022; Poelhekke et al., 2016).

In this study, two different simulation sets were defined to evaluate and optimize the performance of the emulator. The spatial domain of the first set was a simplified beach with a linear bathymetry (Fig. 1a). The focus of this first set was to assess the up-sampling operation in the architecture of the emulator to reduce the error in the forecasts. In the second set, the capacity of the emulator in reproducing the morphodynamics was evaluated in a more complex geometrical domain involving two different coastal defence structures, a submerged breakwater (Fig. 1b) and a groin (Fig. 1c).

2.1.1. *Coastal domains characteristics*

Three different coastal domains were defined to analyse the performance of the emulator (Fig. 1). The first domain consists of a beach with a 2 km cross shore width and 4 km alongshore length discretized with a 100×200 grid with 20 m cells size resolution. The bathymetry varied from 0 m to -20 m and the topography from 0 m to 10 m. The topo-bathymetry is homogeneous in the north-south direction. In the west-east direction, it varied linearly (Fig. 1a).

The second and third domains used a 335×375 grid with 5 m cells size resolution, resulting in a coastal domain of 1.68 km cross-shore width and 1.88 km alongshore length. The bathymetry varied between 0 m and -12 m at the offshore boundary, and the topography from 0 m to 2 m at the inland boundary (Fig. 1b and c).

2.1.2. *Numerical models initial and boundary conditions*

The XBeach simulations run in the surfbeat mode, in which the short-wave motion is solved using the wave action equation (Roelvink et al., 2009). For the first set of simulations that used the linear bathymetry, the initial water level was assumed equal to 1.6 m at the offshore boundary and 1.3 m at the landward boundary, while in the second set of simulations, which included domains with coastal defence structures, the initial water level was assumed equal to the mean sea level (0 m MSL).

The Joint North Sea Wave Project (JONSWAP) spectral wave model was selected to force the models to represent the occurrence of an extreme event. The significant wave height used in simulation set 1 was 6 m, intending to simulate a highly energetic storm to force intense morphological changes during the simulation. The significant wave height (*Hs*) was 3 m for the simulation set 2, which is a representative value, based on the historical wave measurements, of a typical storm in Portugal (C.A. Oliveira et al., 2020). For this geographical location, the average peak periods (*Tp*) vary between 5 s and 12 s and the mean *Hs* usually varies between 1.5 m and 2 m, but values between 3 and 6 m are recurrent (Vieira et al., 2020; Viitak et al., 2021).

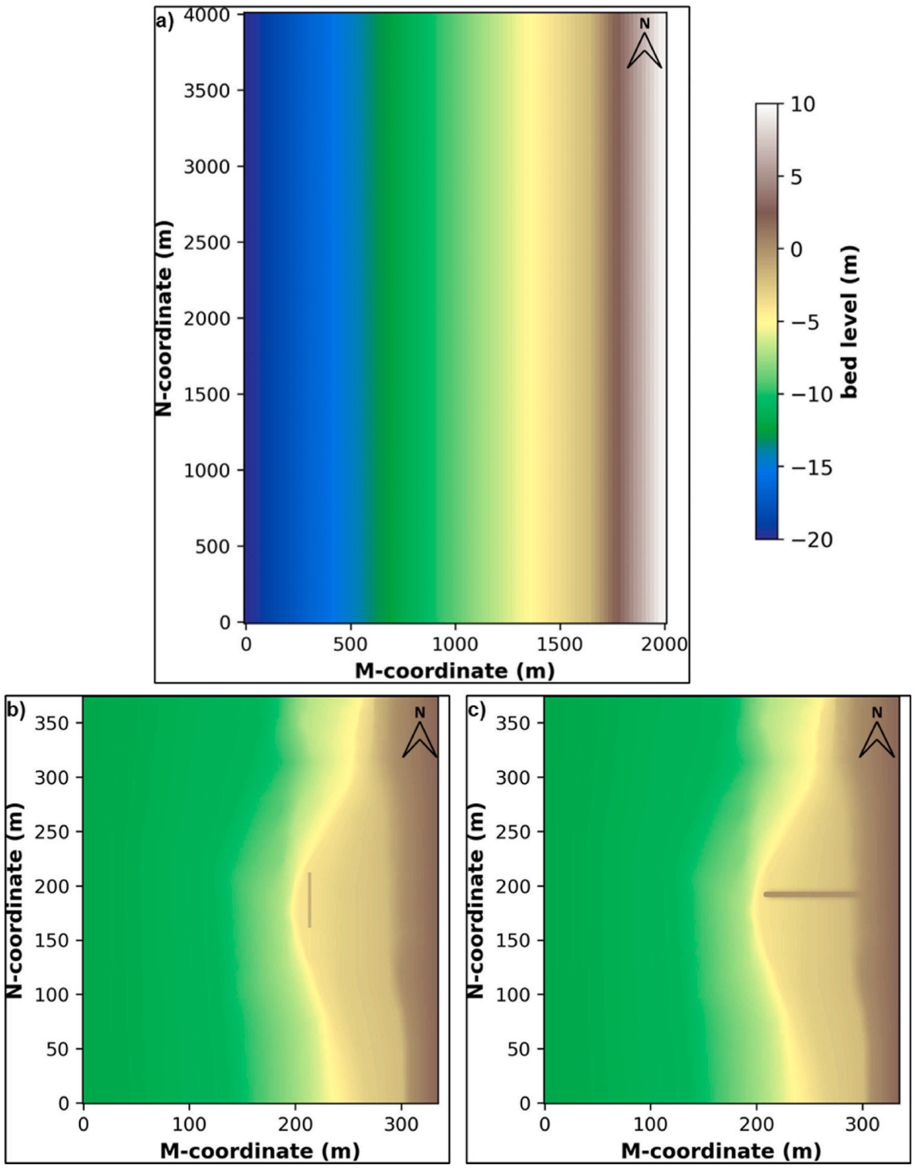

Fig. 1. Topo-bathymetry considered for each XBeach domains: a) Linear topo-bathymetry domain; b) Breakwater domain; c) Groin domain.

The hydrodynamic simulation period was 6 h for both simulation sets, although the sediment transport and morphology modules started with half hour lag in the simulation set 1 and with one-day lag in simulation set 2 to avoid numerical inconsistencies. Additionally, a morphological acceleration factor (MORFAC) of 20 was adopted in simulation set 2 to represent the morphological evolution along 5 days. Furthermore, the adopted critical avalanching slope above water was 0.15 and underwater was 1. The summary of the boundary conditions used for each simulation set are presented in Table 1, where the *mainang* parameter is the main wave angle in the nautical convention, *s* is the directional spreading coefficient, *gammajsp* is the peak enhancement factor and *fnyq* is the highest frequency of the JONSWAP spectrum.

2.1.3. Simulation scenarios

As already briefly explained in the introduction, each simulation set was elaborated to accomplish different objectives. The first set had the goal to study up-sampling operation to improve the performance of the emulator. The second set assessed the capacity of the DL model to emulate more complex coastal environments under the influence of coastal defence structures and assessed different combinations of inputs and outputs during the training of the network. Scenarios 1 to 4 in

Table 1
Domain characteristics and boundary conditions of simulation sets.

Simulation set	1	2
Objective	Evaluate up-sampling operation and simulation computational time	Evaluate the emulator performance with different scenarios
Domain	2 km × 4 km grid with 20 m resolution	1.68 km × 1.88 km grid with 5 m resolution
Bathymetry	Linear bathymetry	Simplified typical northern Portugal coastal bathymetry
Morfac	1	20
Boundary conditions (JONSWAP wave model)		
Hs (m)	6	3
Tp (s)	15	15
mainang (°)	300	315
s	20	20
gammajsp	3.3	3.3
fnyq (Hz)	1	1

With these conditions, the total simulation time was 28 min for the first domain, 65 min for the groin domain and 67 min for the breakwater domain, using a 6 cores processor with a 2.2 GHz base frequency.

simulation set 2 assessed the capacity of the emulator in reproducing different morphodynamic variables, the current timestep bed level change (CTBLC) or the cumulative erosion and sedimentation (CES). Scenarios 5 to 8 were defined to evaluate the possibility of implementing a single emulator for different domains. The models were trained using the data of both domains, but its performance was assessed considering the results of each domain individually.

The characteristics of the scenarios are synthesized in Table 2. The simulation set 1 presents the up-sampling functions that were tested. *ConvT* refers to the transpose convolution layer, followed by the activation function that was used in that layer. At the simulation set 2, the acronyms refer to the training and testing conditions. The first acronym refers to the domains of the data used to train the network. BW is for breakwater; G is for groin and BWG for both domains. The second acronym refers to the XBeach variable used as output by the emulator (CES or CTBLC). The third acronym, presented in scenarios 5 to 8, refers to the domain wherein results were used to assess the performance of the emulator. This last acronym was only necessary to differentiate the validating conditions of the models that were trained with the data of both domains; hence, it was not necessary in scenarios 1 to 4.

Additionally, the sensitivity of the emulator to the numerical model grid resolution was assessed. For this test, the set 1 was selected using the same model domain and the same topo-bathymetry but with two different grid resolutions: 20 m × 20 m grid resolution (original domain) and 40 m × 40 m grid resolution (a quarter of the original resolution). These models, which will receive the acronyms OR, for the highest resolution domain, and LR, for the lowest resolution domain, were used to assess if the emulator could learn using the coarser resolution hydrodynamic input to predict the morphodynamic output of the finer domain without losing accuracy.

2.2. Implementation of the morphodynamic emulator

The DL model was implemented in the Tensor Flow framework (Abadi et al., 2016), in Python programming language version 3.9.7 using a Spyder environment version 5.2.2. Details about the implementation of the emulator are presented in the following sub-sections.

2.2.1. Input and output images processing

The DL network used the Generalized Lagrangian Mean velocity (GLMV) magnitude and the bottom shear stress (BSS) as inputs. The output in simulation set 1 was CTBLC. In simulation set 2, the CES result was additionally used. To generate the image datasets with these variables, a Python script was programmed to ensure that all the images presented the same resolution and colour scale characteristics. The axes of the images were turned off and the margins were set to 0 to ensure that all the pixels of the images coincided with the numerical model domain.

The *netcdf4* library (Whitaker et al., 2020) was used to read the XBeach results and the Matplotlib library was used to plot the results (Hunter, 2007). The images were created with the *imshow* function and were exported with 300 dpi, resulting in a resolution of 452 × 900 pixels for simulation set 1 and 540 × 600 pixels for simulation set 2. The images were plotted in grayscale, and the limit values of the colour scales were determined using a programmed routine that iterated through all the variables' datasets to find the maximum and minimum values for each variable at all run time steps. Brighter tones indicate positive values (sedimentation) while darker tones indicate negative values (erosion) or zero, in the case of GLMV and BSS. Examples of these images are presented in Fig. 2.

The exporting loop was divided into 3 parts: firstly, it read the results of the ith time step and created an image with the pre-defined configurations. Secondly, it updated the image name according to the respective time step, to avoid overwriting the results. Thirdly, the image was exported to a folder in which all the images were stored according to each variable. It resulted in 54 and 150 images per variable for the simulation set 1 and 2, respectively. The scenario that evaluated the emulator with different input and output grid resolutions used 109 images per variable, maintaining the same training/test ratio. This increase in the number of images was necessary to reduce the emulator mean error.

For the simulation set 1, only the results of the CTBLC were used to evaluate the performance of the model. For the set 2, the CES results were also used to train the network and to evaluate if the emulator could properly forecast the morphological evolution of the domains. The inputs were the same in this second case, the unique difference was the output variable used for training the network.

2.2.2. Network architecture

A U-net and a recursively deconvolutional branched network are the basis for the emulator architecture, in which each branch mapped the image features of one specific resolution (Ronneberger et al., 2015; Santhanam et al., 2016). This configuration, which was also used (by Weber de Melo et al. (2022), is presented in Fig. 3. The hyperparameters, namely the activation functions, number of filters, kernel size, strides and dilation rate of the convolutional layers remained the same.

The DL model was implemented using the Functional API of the TensorFlow framework, due to the complexity of the chosen architecture. The network initially mapped the features of the input data and reduced the footprint of the data using convolutional operations with strides set to 2, to half the resolution of the layer inputs. After each stride convolution, a concatenation layer merged the layers' outputs. After that, up-sampling layers restored the original resolution of the input data, and 3 other convolutions were performed. The activation function was the ReLU in all convolutional layers except for the last, which used a hyperbolic tangent function (tanh).

Besides, a deconvolution layer, also known as transposed convolution (Dumoulin and Visin, 2016), replaced the up-sampling operation (green arrows in Fig. 3). This change increases the number of trainable parameters in the architecture of the network, allowing the improvement of the emulator performance. The ReLU and tanh activation functions were assessed during this step.

The main difference between these two operations is how they increase the resolution of the images. The up-sampling layer increases the number and/or columns of the input data. The values of the new elements will depend on the interpolation method of the layer. The deconvolutions, on the other hand, consists of a convolution with a filter derived from the transposition and inversion of the tensor resulting from the original convolution filter (Aggarwal, 2018). This type of layer presents the same properties, parameters and hyperparameters of a convolution layer, however, the output has more dimensions than the input.

In the last two layers, a ReLU operation with a maximum value equal to 1 limited the values of the output image between 0 and 1, and a rescaling operation set the output values in a 0–255 scale, which is the same scale as the output images.

Table 2
Testing conditions of the DL models. In simulation set 1, the up-sampling layer was evaluated whilst simulation set 2 assessed different combinations of inputs and outputs.

Simulation set		1	2
Scenarios	1	Up-sampling layer	BW - CES
	2	ConvT, ReLU	BW - CTBLC
	3	ConvT, tanh	G - CES
	4	–	G - CTBLC
	5	–	BWG - CES - BW
	6	–	BWG - CTBLC - BW
	7	–	BWG - CES -G
	8	–	BWG - CTBLC - G

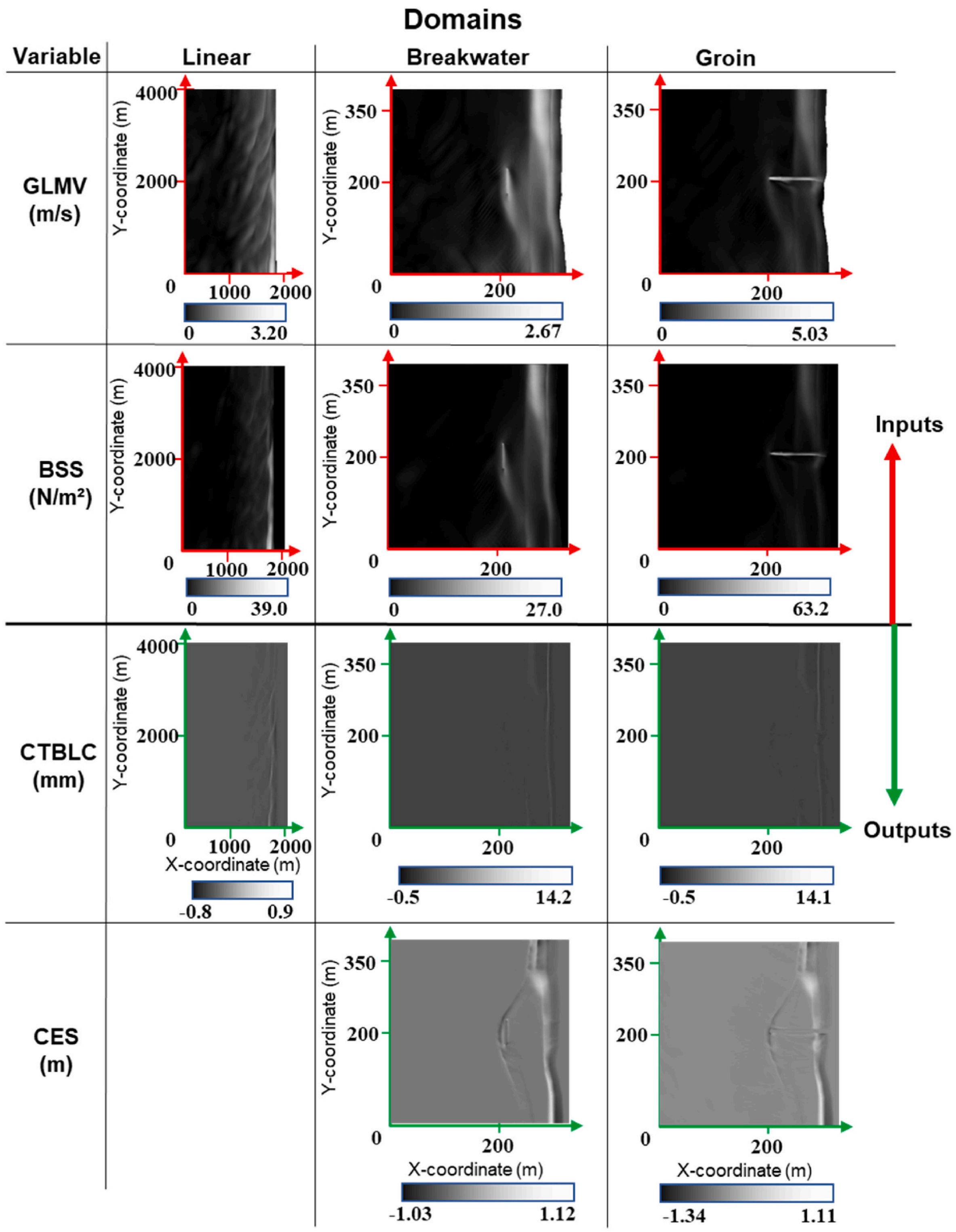

Fig. 2. Examples of images of the hydrodynamic model results used as input (GLMV and BSS) and of the morphodynamic model results used as output (CTBLC and CES) by the deep learning model. The bound values used to plot the images are in the grayscale colour bar above each image. The DL models used the first two variables as input and at least one of the last two variables as output.

2.2.3. *Training and validation*

The networks were trained using the RMSprop optimizer with a learning rate of 0.005 and a limit of 300 epochs. Early stopping was configured to interrupt the training if the performance of the emulator stabilized after 15 consecutive epochs. The loss function was the mean squared error (MSE), and the root mean squared error (RMSE) was used to monitor the training progress. 80% of the images were used for training and 20% for validating the model.

Each training epoch lasted approximately 20 s for simulation set 1 and 35 s for set 2. The training was performed in a CPU Intel Core i7-8750H 2.20 GHz, 16 Gb RAM and a graphics card Nvidia GeForce GTX 1060 with 6 Gb, using parallel computing aided by the CUDA

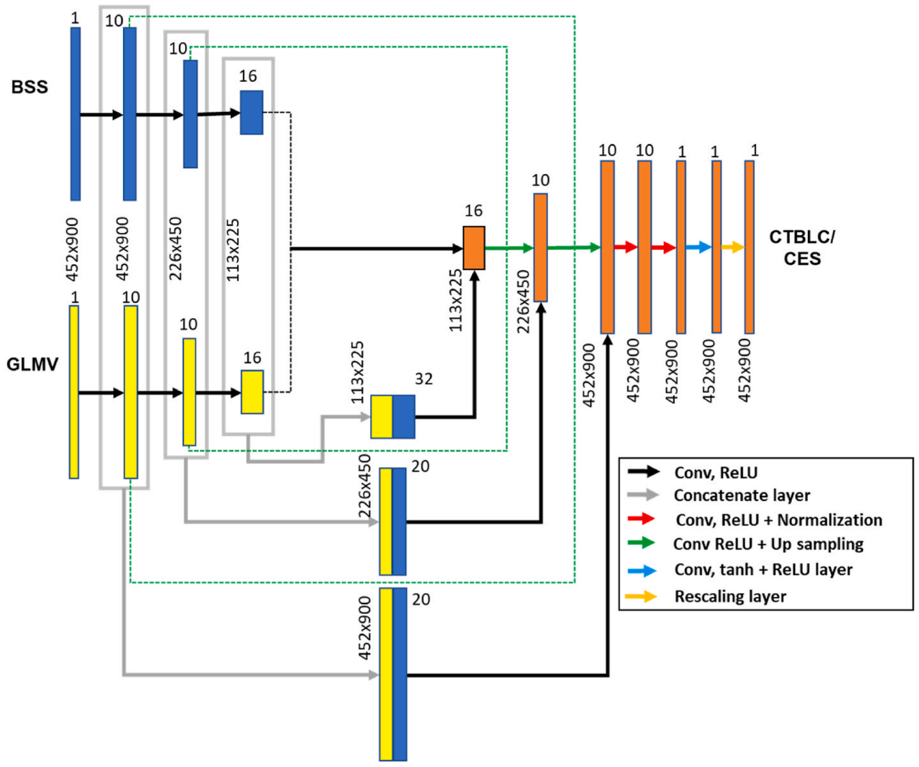

Fig. 3. DL model architecture. The rectangles represent the resolution of the image in each network layer. The numbers above each layer are the number of filters, and each coloured arrow represents one different operation performed in the network.

toolkit v. 11.2 (Harish and Narayanan, 2007).

The model architecture with the best performance obtained with the simulation set 1 was used in the simulation set 2 scenarios. For scenarios 1 to 4 the models were trained for each domain and for each output variable (CES and CTBLC), resulting in 4 emulators. In scenarios 5 to 8, the models were trained using the data of both domains at once, resulting in two emulators. These last scenarios verified the possibility of using only one model to predict the morphological evolution of different domains. Additionally, in this second case, the models would be capable of identifying the domain according to the input data.

2.2.4. Performance analysis

To compare the results of the numerical models and the results of the emulator, the root mean squared error (RMSE) of the model in the validation dataset was computed for each time step:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(Em - Nm)^2} \tag{1}$$

where *n* is the number of grid points, *Em* is the result of the emulator and *Nm* is the result of the numerical model. Also. the mean error was determined to identify the location of the highest errors in the domain:

$$ME = \sqrt{\frac{1}{m}\sum_{i=1}^{n}(Em - Nm)} \tag{2}$$

where *m* is the number of time steps of the validation dataset.

Besides, the kernel-density estimation (KDE) of the errors in the domains was computed, considering all the time steps. The KDE method estimates the probability density function of a non-parametric dataset (Chen, 2017). This last metric is relevant to successfully analyse the error distribution in all the domains to identify areas in which the emulator performance is better.

3. Results and discussion

3.1. Simulation set 1

Simulation set 1 had the objective of evaluating the influence of the up-sampling operation in the architecture of the emulator. The RMSE in scenario 1 was 0.0241 mm, 0.0247 mm, and 0.0183 mm for scenarios 1, 2 and 3, respectively.

All the models could achieve reasonable results, however, the use of the tanh activation function in the transposed convolution layer (Scenario 3) resulted in the emulator with the best performance. It probably occurred due to the increase in the number of trainable parameters in the network, granting a better approximation to the numerical model results. Additionally, it did not meaningfully affect the computational time required by the emulator to process the validation dataset. Therefore, the architecture used in scenario 3 was used to evaluate the computational performance in simulation set 2 scenarios.

3.2. Computational performance

The DL models needed 5 s to generate the morphodynamics results referred to the whole dataset (training and validation). However, it is also necessary to run the hydrodynamic module of the numerical model to generate the input data required by the emulator.

Therefore, to correctly estimate the computational time required for the DL model, it was necessary to run the numerical model with the sediment transport and morphology modules deactivated. This comparison was conducted using the simulation set 1 model, due to its simplified bathymetry and, consequently, hydrodynamics patterns. This model run a 6 h simulation, with morfac set to 0 and with the sediment transport and morphology modules deactivated.

With both morphodynamics modules activated, 28 min were necessary to run the simulation. With only the hydrodynamics module activated, the simulation took 23 min. Additionally, 15 s were necessary to generate 108 input images (54 per variable), and 5 s to run the DL model with this data, representing a reduction of 23% in the total time necessary to forecast the morphodynamic evolution of the domain, considering the predictions for the whole simulation period. Therefore, the main limiter to reduce the morphodynamics forecast computational time is the hydrodynamic simulation.

The emulator that used different input and output resolutions had the best performance gain. The hydrodynamic simulation lasted 3.3 min, resulting in a time reduction of 87% to generate the morphodynamics results. As the major restriction to reduce the computational time was the hydrodynamic simulation, reducing the grid resolution to train the emulator could represent a significant performance gain.

Additionally, it must be highlighted that it would be necessary to update the numerical model bathymetry before generating the hydrodynamics results of the next forecast window. The bathymetry variation directly affects the hydrodynamics variables, consequently, update it during this type of simulation is of utmost importance to correctly forecast the HMD variables and reduce the uncertainties of the results.

Looking forward to optimizing the generation of the hydrodynamics results, three approaches could be considered. Firstly, the XBeach simulation time presents a high dependence on the wave velocities that must satisfy the CFL condition (Courant et al., 1928; Roelvink et al., 2009; Schneider et al., 2013). The simplest method to ensure this condition and maintain the grid resolution is to set the smallest time step for the entire simulation, which is determined according to the highest wave speed (Gnedin et al., 2018). Although, this approach could increase the total simulation time, especially if the objective of the model is to simulate extreme events, which would result in higher wave velocities and demand a reduction in the simulation time step to satisfy the CFL condition.

Alternatively, the reduction of the grid resolution would allow higher time steps, reducing the total simulation time and satisfying the CFL condition. However, the focus of this methodology was to reduce the computational costs of high-resolution numerical models during short-term events, hence, this hypothesis was not considered in this work.

The last option would be to optimize the input image generation loop. In this study, a loop was used to generate the images of each variable. In each step of the loop, the image referring to the *i*th time step was created and exported. To reduce this time, a single loop for all variables was performed, which partially reduced the redundancy in the code, instead of using one loop for each variable, optimizing this process. In despite of that, the total time required to generate the hydrodynamics images was 15 s. Hence, an optimization in this process would not significantly reduce the total time required.

Additionally, an important result achieved, if compared with previous works, is the reduction in the number of images necessary to train the emulator. In Weber de Melo et al. (Weber de Melo et al., 2022), 1095 images were generated to train and test the emulator, whereas herein only 162 images were needed in the simulation set 1 to achieve good results. This demonstrate that using AI models do not necessarily require a huge database for training and testing, which agrees with data-centric AI concepts that affirms that the quality of the data is far more important than its quantity (Sambasivan et al., 2021). The XBeach database required 20.61 Mb of disk space, while the D3D needed 137.1 Mb, representing a reduction of 85% in the disk space usage. The emulator with different input/output resolutions needed 40.8 Mb, representing 70% less disk space.

3.3. *Influence of the grid resolution*

The difference between the emulators trained with inputs of different resolution and the numerical model output for the test dataset are presented in Fig. 4. The mean error of both emulators was negligible, but the mean error of the LR emulator was one order higher than the OR emulator. The maximum error occurred in the OR emulator results was

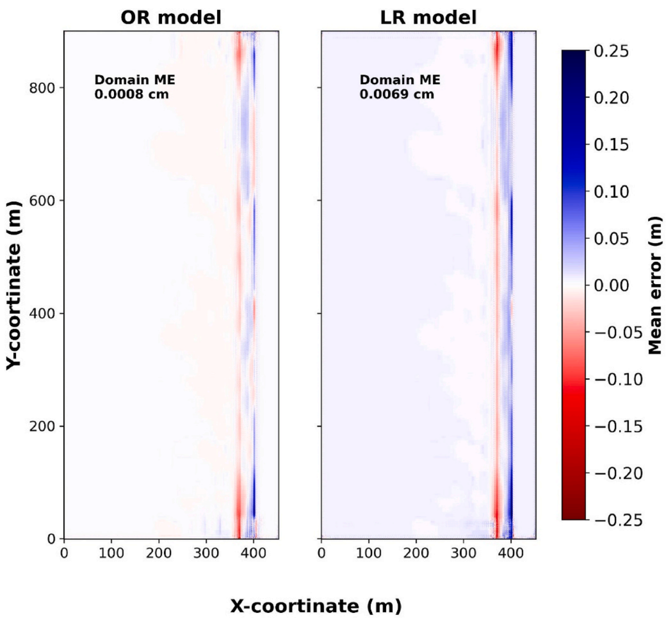

Fig. 4. Comparison between emulators using inputs with different grid resolution.

21 cm and in the LR emulator was 26 cm. These values are considerably higher than the mean error, however they are restricted to specific areas of the domain, particularly near the domain boundaries, which normally fall out of the region of interest when using numerical modelling approaches.

Fig. 5 shows the RMSE of the OR and LR emulators in the test dataset. It can be observed that the performance of both models is remarkably similar in all points and the maximum mean error was below 5 cm in the OR model and below 8 cm in the LR. Besides, both error curves follow a similar pattern, having the same behaviour at timesteps 14 and 17, for example.

3.4. *Simulation set 2*

Fig. 6 presents a comparison between the output in the last time step of the XBeach model with the results obtained with the emulator using the CES images (scenarios 1 and 3). The emulator could reproduce the main patterns of the morphological evolution of the domains with a remarkable resemblance. The erosion and accretion bars can be observed in the southern area of the beach in both images, as well as a larger accretion area around the coordinate (300, 300). This pattern results from the 315° wave angle of the wave spectrum applied at the boundaries of the XBeach model, which created this accretion spot upstream of the structures. However, the emulator underestimated the

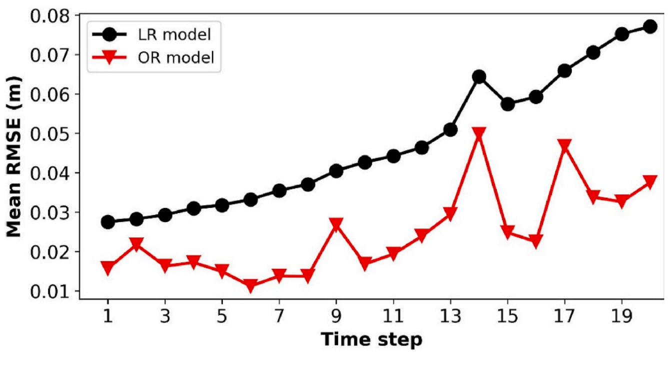

Fig. 5. Emulators RMSE in the test dataset.

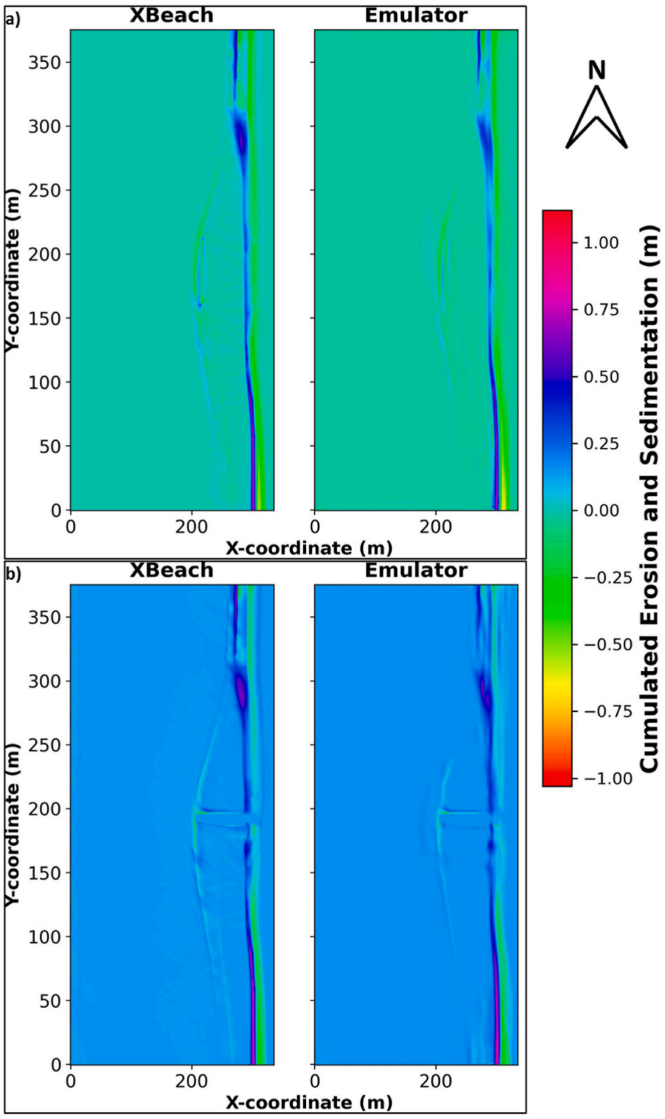

Fig. 6. Comparison between the CES simulated by the XBeach model and the emulator in the last simulation time step. a) Breakwater domain; b) Groin domain.

accretion area near this coordinate, which is evident due to the thinner blue bar in the breakwater domain and the reduced quantity of purple pixels in the groin domain.

This result demonstrates that the use of CES as output instead of CTBLC did not significantly affect the performance of the emulator. The emulator accurately simulated the main morphodynamic patterns of both study areas, independently of the selected domain.

Comparing the results of both selected coastal structures, the output of the breakwater emulator is more similar to the numerical model result than the output of the groin emulator. This could be related with the uniformity of the breakwater influence along the beach when compared to the groin. The groin breaks the linearity of the erosion/accretion bars, creating an area with a more complex morphodynamic pattern. Hence, more training images would be necessary to improve the precision of the groin emulator.

Additionally, the emulator underestimated the erosion that occurred near the groin, in the centre of the domain. It can be observed that the numerical model simulated a larger erosion area in the north-south direction, whilst this pattern was not accurately forecasted by the emulator, mainly at the north of the breakwater. In the case of the breakwater emulator, it underestimated the accretion that occurred in the east face.

The performance of the emulator could be improved by increasing the size and variability of the training dataset variables. The use of more images would give more information about the output pattern expected in that area of the domain, increasing its performance. Moreover, the error in the areas wherein the morphodynamic variation is more uniform was lower than the error in areas with higher variability, namely near the breakwater and the groin. However, it must be stressed that the emulator could still identify that the morphological change in the groin was zero, probably because this pattern was more common in all training time steps.

To assess the error variance of the emulators in all the validation timesteps, the RMSE was computed between the DL model output and the numerical model result for each pixel of the image. The mean value of the RMSE was then determined by dividing the RMSE for the validation time steps. Fig. 7 presents these results, separated by the output variable used to train the DL models.

The results demonstrated that the models trained in only one domain had a better performance than the models trained in both domains. It probably befell because the input variables values were affected by the groin and the breakwater only in the area around these interventions. In the other parts of the domain, the input values were more resemblant due to the boundary conditions being the same. Hence, the network could learn the CES patterns for both domains, having similar errors as demonstrated in Fig. 7 – a.

In Fig. 7 – b, the error of scenarios 6 and 8 was quite different when compared to the models trained with CES data. It probably happened because the CTBLC patterns are more similar between the domains, given that these results show only the areas in which the bed level was altered in the last time step. This fact could explain why the performance

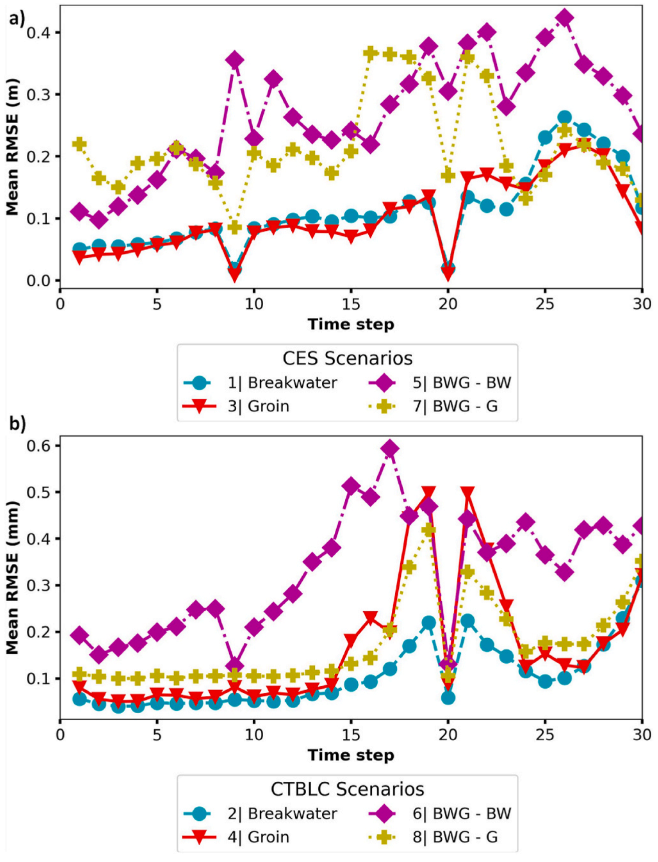

Fig. 7. Models mean RMSE at each validating time step. a) Error (m) of the models trained in CES data; b) Errors (mm) of the models trained in CTBLC data.

of scenario 8 was closer to scenarios 2 and 4, a different pattern when analyzing the CES results. Despite that, the performance of the models was better when the training conditions were from only one domain. It probably occurred owing to a possible ambiguity in the input and output data since most of the pixel values in the input images were not affected by the domain.

Besides, the BWG emulator had to decide which domain to prioritize during the training, resulting in a better adjustment to the groin domain instead of the breakwater. This could explain why the error in scenario 8 was lower than the error in scenario 6. However, the performance in scenario 8 overcame scenario 4 between time steps 15 and 23, indicating that the DL model parameters were more suitable for those inputs. This would allow the use of ensemble techniques (Biolchi et al., 2022; Iglesias et al., 2022) to create a unique and more robust model that could merge individual models trained in specific conditions, allowing the model to select a determined input-output path according to the available data.

Fig. 8 presents the mean error obtained in the simulation set 2 scenarios. This result allows to identify the average location of the major errors in the numerical model domains. The whiter colors represent areas where the error was near 0, red colors represent the emulator underestimation and blue colors represent overestimation. Scenarios 2, 3 and 5 had an average overestimation of the numerical model results, whilst the other scenarios presented an average underestimation.

Fig. 8 – a shows that, in all scenarios, the DL models underestimated the values of an erosion area that occurred near the coordinates (280, 350). This probably occurred owing to the proximity to the boundary of the numerical model, resulting in a higher uncertainty for that area. Moreover, the model also underestimated the accretion area at coordinates (300, 300). Despite that, all the models had a reasonable agreement between the numerical model results and the data model outputs.

It can be observed in Fig. 8 – b that the highest errors in scenarios 2 and 6 occurred near the beach, where the sediment transport is more intense, although is in this same area where the lowest errors in scenarios 4 and 8 are found. This demonstrates that the DL model trained in both domains (scenarios 6 and 8) adjusted its parameters to better represent only one domain, which was the one with lower errors. This same behaviour can be observed in scenarios 5 and 7, in which the error in scenario 7 was much lower than the error in scenario 5.

Lastly, the error distribution was plotted (Fig. 9), which allows for assessing the variability of the errors, considering all the time steps of

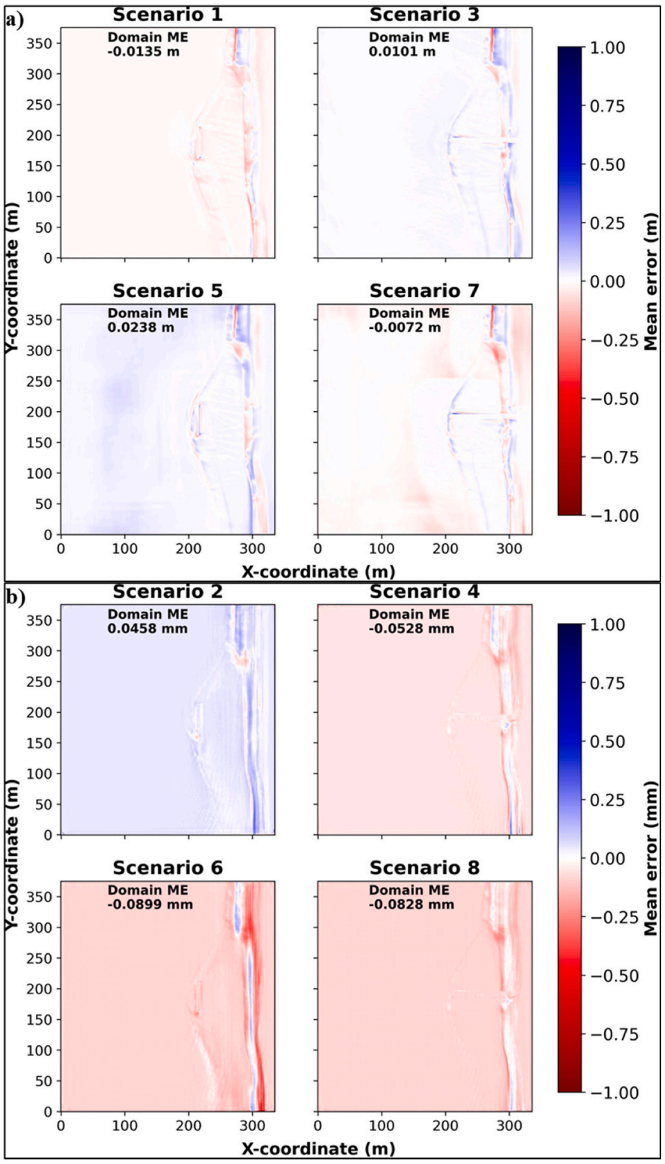

Fig. 8. Mean error of the DL models in the domains. Red colors represent emulator underestimation and blue colors overestimation. White colors indicate a perfect agreement between the numerical model and the emulator. a) Error of the models (m) in forecasting the CES; b) Error of the models (mm) in forecasting CTBLC.

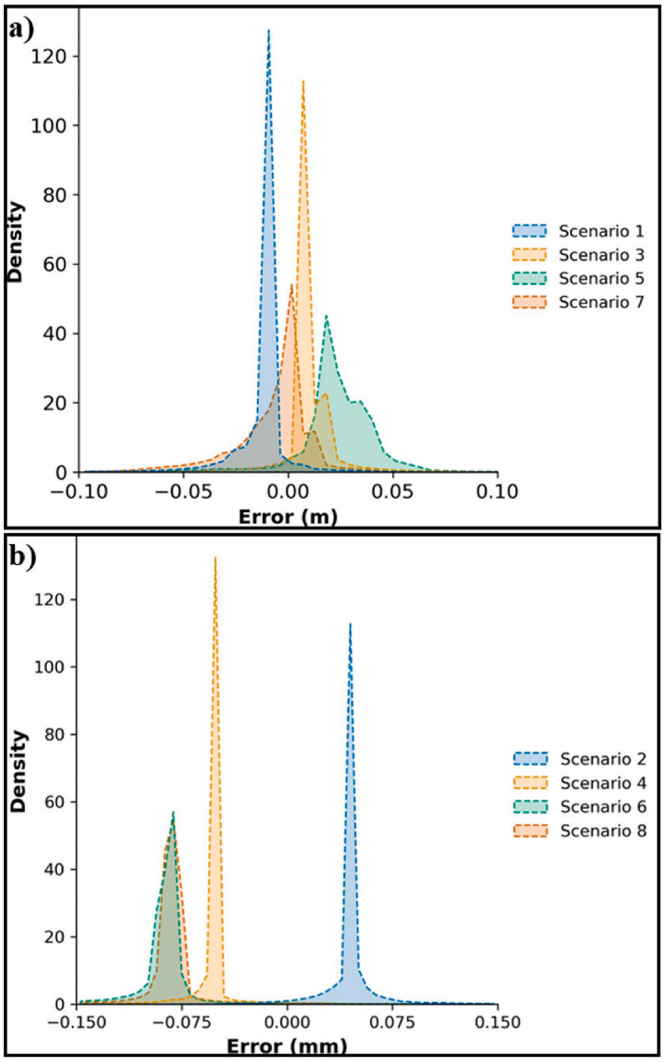

Fig. 9. DL model error distribution. a) Distribution of the CEM errors; b) Distribution of the CTBLC errors.

the validation dataset. The peak values in the distributions are approximately the mean errors presented in Fig. 8. The models trained in single domain data had a lower variability in the errors, regardless of the output variable. Additionally, the emulators presented acceptable results for all the domain areas, which is indicated by the error low variability.

Therefore, DL emulators could successfully surrogate morphological modules of numerical modelling tools as the XBeach without significant changes in the result. The errors achieved by the DL models represented 1.1% of the maximum CES values and 0.6% of the maximum CTBLC values, reinforcing the similarity between the results of the emulator and the XBeach models.

3.5. Future of HMD modelling

The capacity of the emulator in reproducing the main morphodynamic patterns of coastal beaches and the reduction of computational resources requirements demonstrate the potential of data-driven models, staring to modify the logic behind environmental modelling. Numerical models can achieve reliable results by solving deterministic equations, although these equations simplify some aspects of the reality. For instance, the bottom friction coefficient is usually simplified to a uniform value for all the computational domain, although it is known that this coefficient varies according to the characteristics of the sediments, bottom structure, depth and vegetation. These details can be inserted into numerical models, but the efforts in their implementation would sharply rose.

Data-driven models, conversely, intrinsically consider all the domain features, as it will be reflected in the training data. It is also important to stress that the data characteristics determines the validation interval in which data models can be applied. For example, for the implementation of a model to predict extreme events effects it is not expected that a data model trained in average conditions will have satisfactory performance, as well as a numerical model calibrated in average conditions will not either. In this context, a change of paradigm may be interesting for future modelling methodologies. Instead of creating more complex models that can consider all sources of uncertainty, efforts could be directed to improve the quality, variability, and frequency of data, thereby reducing the uncertainties of forecasts to the characteristics of the training data and to the errors in the measurements.

Regarding the computational resources required for modelling, the use of parallel computing and high-performance computers would reduce the simulation computational time. However, these types of machines, and their necessary infrastructure, have high associated costs, limiting the application of real time forecasting platforms to those who can afford them. Therefore, the development of methodologies that does not depend on super computers is a promising path to guarantee a wider application of early warning systems.

4. Conclusions

This study applied an emulator to a hydrodynamic coastal numerical model using a deep learning approach implemented in Python and the Tensor Flow framework. Images of the hydrodynamic variables, bed shear stress and GLM velocity, simulated with a XBeach model, were used as input of the DL models that forecasted the morphological evolution for three different domains.

In simulation set 1, a DL architecture based on the previous work of Weber de Melo et al. (2022) had a satisfactory performance for the XBeach models, demonstrating the suitability of the purposed methodology to reduce the computational time of morphodynamic numerical models. There is no reason to think that the developed emulator cannot be used with other hydro-morphodynamic software appropriate to simulate coastal stretches.

Additionally, the performance of the emulator was improved by the replacement of the up-sampling operation in the neural network architecture by transposed convolutions. This procedure increased the number of trainable parameters in the network, allowing the reduction of the error between the numerical model results and the DL model outputs without significantly affecting the computational time required by the emulator to forecast the validation dataset. Furthermore, the tanh activation function resulted in a lower error when compared to the ReLU activation function.

Regarding simulation set 2, it was demonstrated that the emulator could reproduce the numerical model morphodynamic results in coastal domains under the influence of a submerged breakwater or a groin. The errors obtained were two orders lower than the maximum and minimum values of the output variables.

Furthermore, the emulator accurately reproduced both CES and CTBLC results, indicating that the DL model can be easily adapted to forecast any variable of interest if there is enough data to train the network. To determine the size of the training dataset, it is necessary to assess the results in the validation dataset. The use of few data can result in models that can barely forecast the main patterns of the output variable, whilst the use of excessive data can drastically increase the time required for training the network. It is recommended that the training data includes enough variability to correctly represent the application range of the emulator. Moreover, the image database used in this paper requires 85% less disk space than in Weber de Melo et al. (2022), demonstrating that the methodology can achieve reliable results with fewer data.

It can be also concluded that it is possible to train one emulator with data from different domains, however, some precautions are necessary, namely identifying the possibility of a domain prioritization by the DL model. The errors increased in three of the scenarios in which the emulators were trained with both datasets (scenarios 5, 6 and 7) when compared to the emulators trained with a single one. Scenario 8 model results overcame scenario 4 in several time steps, although its error was slightly worse in the others time steps.

Regarding the computational costs of the emulator, it was achieved a time reduction of 23% in simulating the morphodynamic evolution using numerical models with the same input and output grid resolution, whilst a time reduction of 87% was obtained using the emulator with low grid resolution inputs. The main time-consuming task in this methodology was the generation of hydrodynamics results by the numerical model, which limited the improvement of the proposed methodology. Hence, the performance gain with the emulator application would be higher for long-term hydrodynamic simulations due to the lower wave velocities magnitudes, which allows simulations with higher time steps. Although the error in the LR emulator had slightly increased, its results were still similar to the numerical model. Additionally, the use of images as input and output allows the potential application of this methodology to any other numerical model, as mentioned before.

Reducing the total computational time required for simulating the short-term morphological evolution of a coastal area can be important to facilitate the application of XBeach models to real-time early warning and forecasting systems. This would allow to optimize the response of the authorities during extreme events, and to increase the resilience of coastal communities.

Finally, this study demonstrated that the use of deep learning techniques can reduce the computational costs of numerical models, allowing their implementation in real-time forecasting and warning systems. The emulators accurately reproduced the CES and CTBLC results obtained with the XBeach software, which is one of the most advanced coastal morphodynamics simulation model available, achieving an error two orders lower than the morphodynamics variables range values.

Software and data availability

- Name of the Software: XBeach version 1.23.

Developers: Deltares/XBeach Open-Source Community; First year available: 2009; Cost: Free; Software availability: :https://download.deltares.nl/en/download/xbeach-open-source/; Program size: 330.97 MB.

- The deep learning-based emulator used for surrogating the XBeach morphodynamic module was implemented in Python language (version 3.9) based on TensorFlow library. The authors used a Windows 11 Home OS environment, CPU Intel(R) Core (TM) i7-8750H 2.20 GHz, RAM 16 GB, GPU Nvidia GeForce GTX 1060.

The architecture of the model is available at:http://www.hydroshare.org/resource/b4ae97df748842a1800816b32a3d640 b.

Author contributions

W.M., J.P. and I.I. designed the research; W.M. performed the research; W.M., J.P. and I.I. analyzed the data; W.M. and J.P. implemented the models; J.P. and I·I proofread the article.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Deep learning model for XBeach morphodynamic emulation (Original data) (HydroShare)

Acknowledgements

This research was supported by the Doctoral Grant SFRH/BD/151383/2021 financed by the Portuguese Foundation for Science and Technology (FCT), and with funds from the Ministry of Science, Technology and Higher Education, under the MIT Portugal Program. I. Iglesias also acknowledge the FCT financing through the CEEC program (2022.07420. CEECIND).

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., et al., 2016. TensorFlow: large-scale machine learning on heterogeneous distributed systems. http://arxiv.org/abs/1603.04467.

Aggarwal, C.C., 2018. Neural networks and deep learning. In: Artificial Intelligence. Springer International Publishing. https://doi.org/10.1007/978-3-319-94463-0.

Anderson, D.L., Ruggiero, P., Mendez, F.J., Barnard, P.L., Erikson, L.H., O'Neill, A.C., Merrifield, M., Rueda, A., Cagigal, L., Marra, J., 2021. Projecting climate dependent coastal flood risk with a hybrid statistical dynamical model. Earth's Future 9 (12). https://doi.org/10.1029/2021EF002285.

Biolchi, L.G., Unguendoli, S., Bressan, L., Giambastiani, B.M.S., Valentini, A., 2022. Ensemble technique application to an XBeach-based coastal early warning system for the northwest adriatic sea (Emilia-Romagna region, Italy). Coast. Eng. 173 https://doi.org/10.1016/j.coastaleng.2022.104081.

Chen, Y.-C., 2017. A tutorial on kernel density estimation and recent advances. Biostat. Epidemiol. 1 (1), 161–187. https://doi.org/10.1080/24709360.2017.1396742.

Courant, R., Friedrichs, K., Lewy, H., 1928. Über die partiellen Differenzengleichungen der mathematischen Physik. Math. Ann. 100 (1), 32–74. https://doi.org/10.1007/BF01448839.

Deltares, 2018. Delft3D-Flow: Simulation of Multi-Dimensional Hydrodynamic Flows and Transport Phenomena, Including Sediments - User Manual.

Dumoulin, V., Visin, F., 2016. A Guide to Convolution Arithmetic for Deep Learning. http://arxiv.org/abs/1603.07285.

Ferreira, Ó., Plomaritis, T.A., Costas, S., 2019. Effectiveness assessment of risk reduction measures at coastal areas using a decision support system: findings from Emma storm. Sci. Total Environ. 657, 124–135. https://doi.org/10.1016/j.scitotenv.2018.11.478.

Gharagozlou, A., Anderson, D.L., Gorski, J.F., Dietrich, J.C., 2022. Emulator for eroded beach and dune profiles due to storms. J. Geophys. Res.: Earth Surf. 127 (8) https://doi.org/10.1029/2022JF006620.

Gnedin, N.Y., Semenov, V.A., Kravtsov, A.v., 2018. Enforcing the Courant–Friedrichs–Lewy condition in explicitly conservative local time stepping schemes. J. Comput. Phys. 359, 93–105. https://doi.org/10.1016/j.jcp.2018.01.008.

Haasnoot, M., Winter, G., Brown, S., Dawson, R.J., Ward, P.J., Eilander, D., 2021. Long-term sea-level rise necessitates a commitment to adaptation: a first order assessment. Clim. Risk Manag. 34 https://doi.org/10.1016/j.crm.2021.100355.

Harish, P., Narayanan, P.J., 2007. Accelerating large graph algorithms on the GPU using CUDA. High. Perform. Comput. HiPC 2007 4873 LNCS, 197–208. https://doi.org/10.1007/978-3-540-77220-0_21. Springer Berlin Heidelberg.

Hunter, J.D., 2007. Matplotlib: a 2D graphics environment. Computing in Science \& Engineering 9 (3), 90–95. https://doi.org/10.1109/MCSE.2007.55.

Iglesias, I., Pinho, J.L., Avilez-Valente, P., Melo, W., Bio, A., Gomes, A., Vieira, J., Bastos, L., Veloso-Gomes, F., 2022. Improving estuarine hydrodynamic forecasts through numerical model ensembles. Frontiers in Marine Science 9. https://doi.org/10.3389/fmars.2022.812255.

Oliveira, T.C.A., Cagnin, E., Silva, P.A., 2020. Wind-waves in the coast of mainland Portugal induced by post-tropical storms. Ocean Engineering 217. https://doi.org/10.1016/j.oceaneng.2020.108020.

Poelhekke, L., Jäger, W.S., van Dongeren, A., Plomaritis, T.A., McCall, R., Ferreira, Ó., 2016. Predicting coastal hazards for sandy coasts with a Bayesian Network. Coastal Engineering 118, 21–34. https://doi.org/10.1016/j.coastaleng.2016.08.011.

Pörtner, H.-O., Roberts, D.C., Tignor, M., Poloczanska, E.S., Mintenbeck, K., Alegría, A., Craig, M., Langsdorf, S., Löschke, S., Möller, V., Okem, A., Rama, B., 2022. IPCC, 2022: climate change 2022: impacts, adaptation and vulnerability. Contribution of working group II to the sixth assessment report of the intergovernmental panel on climate change. https://doi.org/10.1017/9781009325844.

Rautenbach, C., Trenham, C., Benn, D., Hoeke, R., Bosserelle, C., 2022. Computing efficiency of XBeach hydro- and wave dynamics on Graphics Processing Units (GPUs). Environmental Modelling & Software 157, 105532. https://doi.org/10.1016/j.envsoft.2022.105532.

Roelvink, D., Reniers, A., van Dongeren, A., van Thiel de Vries, J., McCall, R., Lescinski, J., 2009. Modelling storm impacts on beaches, dunes and barrier islands. Coastal Engineering 56 (11–12), 1133–1152. https://doi.org/10.1016/j.coastaleng.2009.08.006.

Ronneberger, O., Fischer, P., Brox, T., 2015. U-net: convolutional networks for biomedical image segmentation. In: Navab, N., Hornegger, J., Wells, W.M., Frangi, A.F. (Eds.), Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 9351. Springer International Publishing, pp. 234–241. https://doi.org/10.1007/978-3-319-24574-4_28. Issue Cvd.

Rutten, J., Torres-Freyermuth, A., Puleo, J.A., 2021. Uncertainty in runup predictions on natural beaches using XBeach nonhydrostatic. Coastal Engineering 166 (February), 103869. https://doi.org/10.1016/j.coastaleng.2021.103869.

Sambasivan, N., Kapania, S., Highfll, H., 2021. Everyone wants to do the model work, not the data work: data cascades in high-stakes ai. In: Conference on Human Factors in Computing Systems - Proceedings. https://doi.org/10.1145/3411764.3445518.

Santhanam, V., Morariu, V.I., Davis, L.S., 2016. Generalized deep image to image regression. In: Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, 2017-Janua, pp. 5395–5405. https://doi.org/10.1109/CVPR.2017.573.

Schneider, K., Kolomenskiy, D., Deriaz, E., 2013. Is the CFL condition sufficient? Some remarks. In: The Courant–Friedrichs–Lewy (CFL) Condition, pp. 139–146. https://doi.org/10.1007/978-0-8176-8394-8_9. Birkhäuser Boston.

Simmons, J.A., Splinter, K.D., 2022. A multi-model ensemble approach to coastal storm erosion prediction. Environmental Modelling and Software 150. https://doi.org/10.1016/j.envsoft.2022.105356.

Smallegan, S.M., Irish, J.L., Van Dongeren, A.R., Den Bieman, J.P., 2016. Morphological response of a sandy barrier island with a buried seawall during Hurricane Sandy. Coastal Engineering 110, 102–110. https://doi.org/10.1016/j.coastaleng.2016.01.005.

Suzuki, T., Cox, D.T., 2021. Evaluating XBeach performance for extreme offshore-directed sediment transport events on a dissipative beach. Coastal Engineering Journal 63 (4), 517–531. https://doi.org/10.1080/21664250.2021.1976452.

Trouw, K.J.M., Zimmermann, N., Mathys, M., Delgado, R., Roelvink, D., 2012. Numerical modelling of hydrodynamics and sediment transport in the surf zone: a sensitivity study with different types of numerical models. Coastal Engineering Proceedings 1 (33), 23. https://doi.org/10.9753/icce.v33.sediment.23.

van Rijn, L.C., 2007. Unified view of sediment transport by currents and waves. I: initiation of motion, bed roughness, and bed-load transport. Journal of Hydraulic Engineering 133 (6), 649–667. https://doi.org/10.1061/(ASCE)0733-9429(2007)133:6(649).

van Thiel de Vries, J.S.M., 2009. Dune Erosion during Storm Surges. Faculty of Civil Engineering and Geosciences - TU Delft.

Vieira, B.F.V., Pinho, J.L.S., Barros, J.A.O., Antunes do Carmo, J.S., 2020. Hydrodynamics and morphodynamics performance assessment of three coastal protection structures. Journal of Marine Science and Engineering 8 (3). https://doi.org/10.3390/jmse8030175.

Viitak, M., Avilez-Valente, P., Bio, A., Bastos, L., Iglesias, I., 2021. Evaluating wind datasets for wave hindcasting in the NW Iberian Peninsula coast. Journal of Operational Oceanography 14 (2), 152–165. https://doi.org/10.1080/1755876X.2020.1738121.

Vousdoukas, M.I., Ferreira, Ó., Almeida, L.P., Pacheco, A., 2012. Toward reliable storm-hazard forecasts: XBeach calibration and its potential application in an operational early-warning system. Ocean Dynamics 62 (7), 1001–1015. https://doi.org/10.1007/s10236-012-0544-6.

Vousdoukas, M.I., Ranasinghe, R., Mentaschi, L., Plomaritis, T.A., Athanasiou, P., Luijendijk, A., Feyen, L., 2020. Sandy coastlines under threat of erosion. Nature Climate Change. https://doi.org/10.1038/s41558-020-0697-0 (in press) (March).

Weber de Melo, W., Pinho, J., Iglesias, I., 2022. Emulating the estuarine morphology evolution using a deep convolutional neural network emulator based on hydrodynamic results of a numerical model. Journal of Hydroinformatics. https://doi.org/10.2166/hydro.2022.068.

Whitaker, J., Khrulev, C., Huard, D., Paulik, C., Hoyer, S., Pastewaka, L., Mohr, A., Marquardt, C., Counwenberg, B., Taves, M., Cuntz, M., Roet, S., Whitaker, J., Brett, M., Bohnet, M., Hetland, R., Korenčiak, M., Andrew, Barnam, Hamman, J., et al., 2020. Unidata/netcdf4-python: Version 1.5.5 Release. https://doi.org/10.5281/zenodo.4308773.