Supplementary Material: Climate Adaptation-Aware Flood Prediction for Coastal Cities Using Deep Learning

S1 Hydrodynamic model data generation

This section provides additional detail regarding the hydrodynamic model simulations performed to generate training data for both the Abu Dhabi and San Francisco Bay shorelines. For simulating the tidal dynamics of the considered region under different coastline protection scenarios, we utilized Delft3D ((Deltares)), a hydrodynamic model that dynamically solves the Reynolds-Averaged Navier-Stokes differential equations, i.e., it is a physics-based numerical model that considers the time-varying forces due to hydrostatic pressures (such as SLR), tidal forcing, meteorological stresses, bottom seabed friction, and river inflows over a finite-element computational grid (down to 30 m in horizontal resolution) spread over variable bathymetry. For any point in the grid, Delft3D will provide time series outputs at 30-minute intervals of water levels and local water circulation velocities throughout the specified simulation period. Importantly, the Delft3D model can handle computational grid cells that alternate between dry and wet states (Barnard et al., 2014).

The Delft3D model was validated (i.e., compared with observed tidal water levels) by running the simulator over a 3-month period between 1 January and 31 March 2017 (without wind forcing) and computing the root mean squared error between the model outputs of hourly water levels at 194 locations throughout the gulf and hourly tidal gauge water level data obtained from the TPXO8 Ocean Atlas for the same period (https://www.tpxo.net/global/tpxo8-atlas). The model was calibrated by adjusting the bottom Manning's roughness coefficient to obtain the lowest error between the predicted water levels the tidal gauge data at located throughout the gulf. Further details concerning the validation of the hydrodynamic model can be found in (Chow and Sun, 2022).

To account for storm surges and wave activity on the Abu Dhabi shoreline (which can be significant especially during prolonged Shamal events), the validated Delft3D model was rerun with wind forcing from the ERA5 database, and the results were fed to an additional spectral wave model, Simulating Waves Nearshore (SWAN) (Delft University of Technology, 2022), which allows capturing wind-wave generation, wave diffraction, amplification and refraction of water surface waves as they approach the shoreline. The SWAN model was applied at a scale of about 100 km along the shoreline to about 50 km offshore under the same forcing from the ERA5 database. Finally, where the waves are in contact with the coastline, the SWAN-computed significant wave heights and the local shoreline slope were used to compute the run-up elevations along the coastline where the waves hit the shore.

Owing to the location of San Francisco Bay as an inland bay, the 30 individual OLU shorelines are sheltered from the storm surges by the exterior Californian coastline, with mean significant wave heights within the Bay at about 0.07-0.2 m

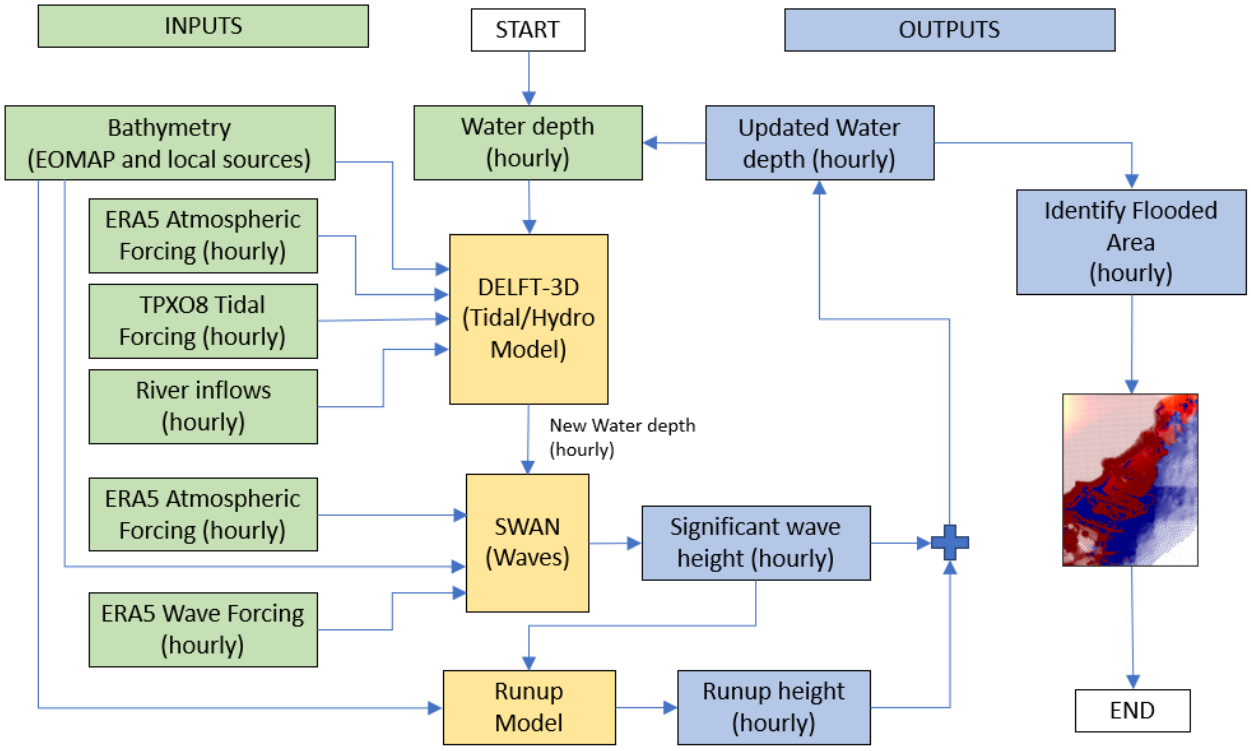

Figure S1. Schematic of hydrodynamic model elements used for the Abu Dhabi scenarios. Green elements denote input parameters; Yellow are the constituent sub-models; Blue are model outputs. Running one cycle of the above model will generate an hourly update of the water depth throughout the UAE coastline.

(https://www.usgs.gov/data/modeled-surface-waves-winds-south-san-francisco-bay), in contrast to about 2.0-3.0 m at Point Reyes located on the California coast outside the Bay. As such, SWAN was not employed for San Francisco Bay, and Delft3D alone was used for the prediction of the coastal flooding events due to SLR, as shown in Fig. S2. Further details about the San Francisco and Abu Dhabi hydrodynamic models are found in (Sun et al., 2020) and (Chow and Sun, 2022), respectively.

S2 Data Preprocessing

This section details the data preprocessing strategy employed to transform raw hydrological simulation data into tensors for training the proposed DL model. The raw data from the Delft3D simulator included inland inundation coordinates x and y, along with the corresponding peak water level (PWL) values for each coordinate pair (x, y). The filenames for each case reflect the underlying OLU protection scenarios represented as binary digits, followed by an underscore and the SLR depth. These binary sequences encode the protection status of OLUs, where '1' indicates a protected OLU and '0' indicates an unprotected OLU. Figure S3 shows randomly selected scenarios for both AD and the SF Bay Area. In the AD scenario (Figure

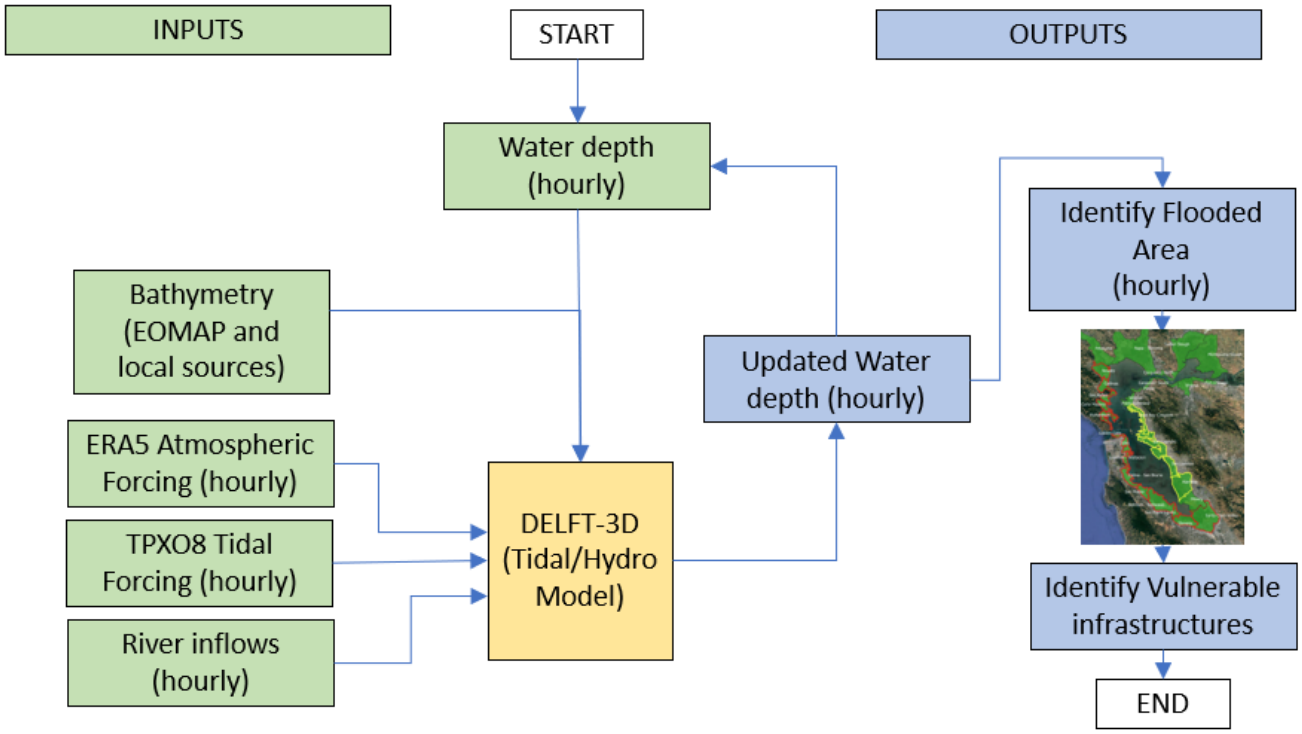

Figure S2. Schematic of hydrodynamic model elements used for the San Francisco scenarios. Green elements denote input parameters; Yellow are the constituent sub-models; Blue are model outputs. Running one cycle of the above model will generate an hourly update of the water depth throughout the shoreline of San Francisco Bay.

S3(a)), the 17 OLUs are represented by *"1010101010101010_0.5"*, where every alternate OLU, starting from the first, is protected while the others remain unprotected. For the SF Bay Area (Figure S3(b)), the 30 OLUs are represented by the binary sequence *"111000111000111000111000111000_1.0"*, where sets of three consecutive OLUs alternate between protected and unprotected status, with a 1.0 m SLR scenario applied. Additionally, the figures depict an inland flood map, allowing a detailed analysis of flood risks under these protection scenarios. The raw datasets were further processed using the following steps to generate a uniform grid representation suitable for training.

S2.1 Coordinate Transformation

The first step in preprocessing involves transforming the original coordinates to a common coordinate system, which is essential for accurate spatial analysis. Specifically, we transformed the coordinates (x, y) of the inundation points from WGS 84 / UTM zone 10N to a standardized latitude and longitude system.

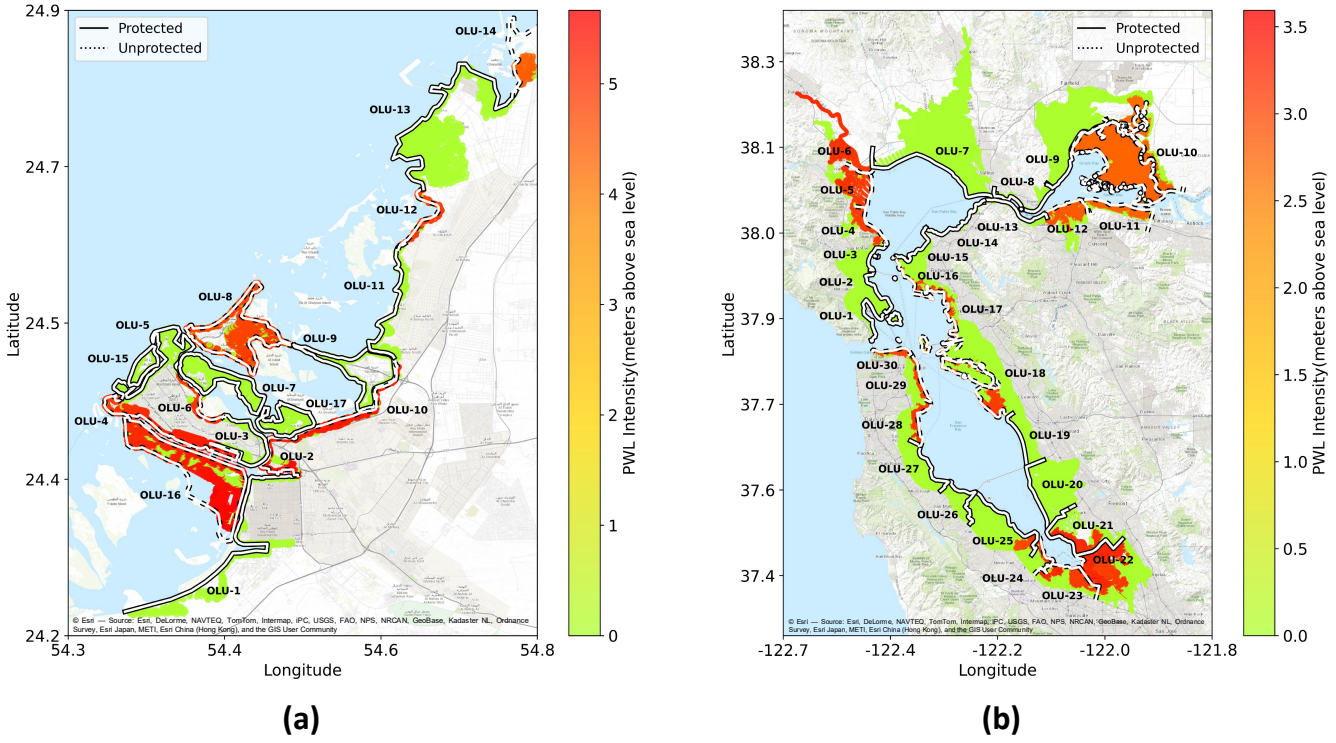

Figure S3. Effects of adopted shoreline protection strategies on inundation for (a) Abu Dhabi and (b) San Francisco Bay Area. In AD, alternate OLUs are protected under a 0.5 m SLR scenario (10101010101010101_0.5). For SF, sets of three consecutive OLUs alternate between protected and unprotected under a 1.0 m SLR scenario (11100011100011100011100011000_1.0). Here, '1' represents protected OLUs, while '0' denotes unprotected OLUs.

S2.2 Grid Mapping and Generation

Next, we mapped the coordinates (x, y) of each inundation point and their corresponding PWL values onto a regular grid to make the dataset compatible with the proposed DL framework. Since CNNs require input data in grid format for effective application of convolution operations, we first generated a grid and then transformed the remaining samples in the datasets accordingly. The grid is defined over the spatial extent of the dataset, with a resolution of $N \times N$. In this research, we set $N = 1024$ to provide an input of 1024×1024 to the proposed DL-based flood prediction model. The grid points ξ and η are computed, as expressed in Eq. (1-2):

$$\xi = \left\{ x_{\min}, x_{\min} + \frac{x_{\max} - x_{\min}}{N-1}, \ldots, x_{\max} \right\} \tag{1}$$

$$\eta = \left\{ y_{\min}, y_{\min} + \frac{y_{\max} - y_{\min}}{N-1}, \ldots, y_{\max} \right\} \tag{2}$$

where $x_{\min}, x_{\max}, y_{\min}, y_{\max}$ represent the minimum and maximum x and y values of the inundation points in the dataset. These values define the spatial boundaries for the grid, which serves as the foundational structure upon which the dataset of inundation points is projected. Next, we mapped each original inundation point (x_i, y_i) to the nearest grid cell (i', j'), as expressed in Eq. (3-4):

$$i' = \max\left(0, \min\left(N-1, \left\lfloor \frac{x_i - x_{\min}}{x_{\max} - x_{\min}} \cdot (N-1) \right\rfloor\right)\right) \tag{3}$$

$$j' = \max\left(0, \min\left(N-1, \left\lfloor \frac{y_i - y_{\min}}{y_{\max} - y_{\min}} \cdot (N-1) \right\rfloor\right)\right) \tag{4}$$

This ensures that each inundation point is mapped to the closest grid cell, maintaining spatial consistency and ensuring that all points are correctly positioned within the defined grid space. Conflicts can occur during grid mapping when multiple inundation points are allocated to one grid cell, due to its finite resolution and data point density. To preserve spatial integrity and achieve a one-to-one correspondence between inundation points and grid cells, we adopted the following conflict resolution strategy. Initially, we identified all grid cells that had multiple inundation points mapped to them. Let $\mathcal{C}$ denote the set of conflicting grid cells, where each grid cell $(i', j') \in \mathcal{C}$ has multiple inundation points $\{(x_k, y_k)\}_{k=1}^{n}$ mapped to it. To resolve these conflicts, we reassigned the conflicting points to the nearest available grid cell. This was done by first identifying a subset of grid cells located within a specified distance from the conflicted point. From this subset, we then calculated the Manhattan distance ($d_m = |i'_{new} - i'| + |j'_{new} - j'|$) between each candidate cell and the conflicted point, and selected the grid cell with the smallest distance. By systematically resolving these conflicts, we ensured that each inundation point was uniquely assigned to a distinct grid cell, resulting in a bijective mapping $\mathcal{M} : (x_i, y_i) \to (i', j')$. The grid structure was then used to process the remaining samples and generate training and evaluation data for the proposed DL model.

S2.3 OLU-Based Classification

In the next step, we classified each inundation point based on its proximity to protected and unprotected OLUs to incorporate the influence of shoreline protection measures into the model. As mentioned earlier in Sec S2, each protection scenario is represented by a binary string $s = s_1 s_2 \dots s_K$, where K is the total number of OLUs, and each bit $s_k \in \{0, 1\}$ indicates the protection status of the k-th OLU (0 for unprotected, 1 for protected). For each inundation point (x_i, y_i), we determined its classification value c_i based on the minimum great-circle distances to the protected and unprotected OLUs. The great-circle distance between an inundation point and a point on an OLU boundary (either protected or unprotected) on the Earth's surface was calculated using the Haversine formula. Specifically, x_i and y_i represent the latitude and longitude of the point of inundation, while x_j and y_j represent the latitude and longitude of a point on the boundary of the OLU. The Haversine formula calculates the distance d_{ij}, as expressed in Eq. (5):

$$d_{ij} = 2R \cdot \arcsin\left(\sqrt{\sin^2\left(\frac{\Delta x_{ij}}{2}\right) + \cos(x_i)\cos(x_j)\sin^2\left(\frac{\Delta y_{ij}}{2}\right)}\right), \tag{5}$$

where $\Delta x_{ij} = x_j - x_i$, $\Delta y_{ij} = y_j - y_i$, and $R = 6,367$ km is the Earth's approximate radius. The minimum distances are determined by iterating over all points defining the OLU boundaries, allowing us to find the closest OLU boundary (either protected or unprotected) for each inundation point, as expressed in Eq. (6-7):

$$d_{\text{prot}_i} = \min_{j \in \mathcal{P}} d_{ij}, \tag{6}$$

$$d_{\text{unprot}_i} = \min_{j \in \mathcal{U}} d_{ij}, \tag{7}$$

where $\mathcal{P}$ and $\mathcal{U}$ are the sets of points on the boundaries of protected and unprotected OLUs, respectively. Lastly, the classification value c_i is assigned to each inundation point based on the distances calculated, as expressed in Eq. (8):

$$c_i = \begin{cases} 1, & \text{if } d_{\text{unprot}} \leq d_{\text{prot}}, \\ -1, & \text{if } d_{\text{prot}} < d_{\text{unprot}}. \end{cases} \tag{8}$$

A higher classification value (1) indicates that the inundation point is closer to an unprotected area. In scenarios where one of the distances is undefined (e.g., if there are no protected or unprotected areas in the scenario), the classification is determined based solely on the available distance. This allowed us to encode the binary input, providing the DL model with essential contextual information about shoreline protection, which can significantly influence inundation patterns and PWL values.

S2.4 Training Data Generation

With the inundation points mapped to the grid and classified, the input and output matrices are generated to train and evaluate the DL model. The input matrix $\mathbf{I} \in \mathbb{R}^{N \times N}$ represents the spatial distribution of protected and unprotected areas, while the output matrix $\mathbf{O} \in \mathbb{R}^{N \times N}$ contains the PWL values associated with each grid cell. For each grid cell (i', j'), the input and output matrices are generated by assigning the values, as expressed in Eq. (9):

$$(I_{i'j'}, O_{i'j'}) = \begin{cases} (c_i, z_i), & \text{if an inundation point is mapped to } (i', j'), \\ (0, 0), & \text{otherwise.} \end{cases} \tag{9}$$

here, c_i is the classification value of the inundation point mapped to the grid cell (i', j'), and z_i is the PWL value of the inundation point corresponding to grid cell (i', j'). In cases where no inundation point was mapped to a particular grid cell, both the input and output values were set to zero. Figure S4 shows randomly selected input and output matrices from both AD and SF datasets. The systematic data preprocessing pipeline ensured that the matrices accurately represented the spatial distribution of the data and were properly formatted, facilitating effective training and evaluation of the proposed DL-based flood prediction model.

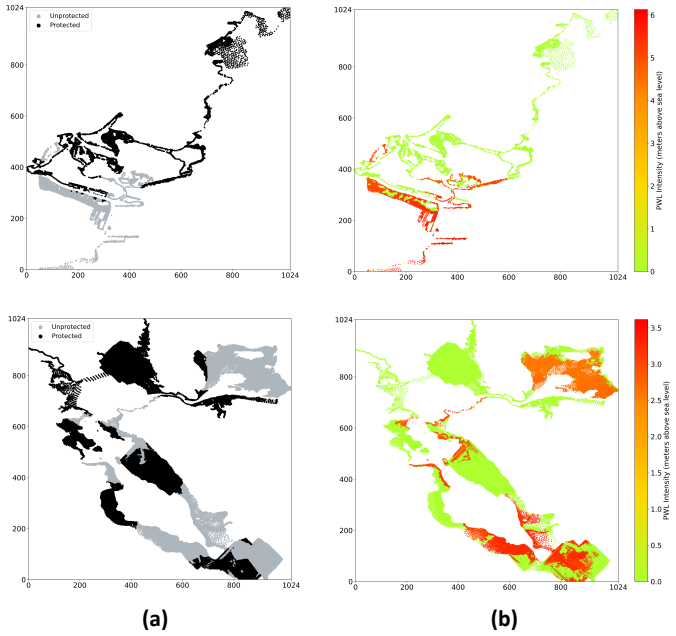

Figure S4. Input and output representations for training the proposed deep learning model, with the top row corresponding to AD and the bottom row to the SF Bay Area. (a) Input images represent inundation points categorized as protected or unprotected based on on their proximity to the nearest OLU. (b) Output images depict inundation values at each point (ground-truth data for training).

S3 Ablation Study

In this section, we detail the ablation experiments carried out to optimize the CASPIAN-v2 architecture, including the evaluation of loss functions, the MARX blocks, the SEE block and the integration of SLR. These experiments were designed to evaluate the impact of each component on model accuracy, generalization, and computational efficiency.

S3.1 Impact of Loss Functions on Model Performance

We first report the effect of various loss functions on the performance of the proposed flood prediction model. For the experiments, we considered the CASPIAN-v2$_{\text{BASE}}$ model consisting of an encoder and decoder depth of 4, integrated with 4 MARX blocks in the bottleneck and 4 SEE blocks between each encoder and decoder layer. The SLR scalar integer was incorporated at the output of the bottleneck before feeding features into the decoder. To ensure consistency across experiments, the initial weights were fixed and loaded for each run.

We evaluated the model using different loss functions, including the Huber (Huber, 1992) loss with 3 fixed delta values and 3 dynamic delta ranges, the Quantile (Koenker and Bassett Jr, 1978) loss with τ values of 0.25, 0.50, and 0.75, the Log-Cosh (Saleh and Saleh, 2022) loss, and a Hybrid loss function that is a weighted linear combination of 3 loss functions. The evaluation was conducted on the validation set using three metrics: ARMSE, AMAE, and avg. R2 Score. Table S1 summarizes

the performance of the CASPIAN-v2$_{\text{BASE}}$ model with different loss functions. As shown in Table S1, the choice of loss function significantly affects the model's performance. The CASPIAN-v2$_{\text{BASE}}$ model using the Hybrid loss function achieves the best overall performance, with the lowest ARMSE of 0.3136, the lowest AMAE of 0.0465, and the highest avg. R2 Score of 0.9356. This indicates that the Hybrid loss function effectively captures the nuances of the data, leading to more accurate predictions.

Among the Huber loss configurations with fixed delta values, increasing the delta from 0.25 to 0.75 improves performance. Specifically, the model with 0.75 delta value achieves an ARMSE of 0.3482, an AMAE of 0.0508, and avg. R2 Score of 0.9236, outperforming the configurations with lower delta values by approximately 13.61% in ARMSE. This suggests that a higher delta value in the Huber loss function allows the model to handle larger errors more effectively. For the Huber loss with dynamic delta ranges, the performance remains relatively consistent, with some variations. The model with a delta range of 0.3 to 0.7 achieves the best results with an ARMSE of 0.3282, an AMAE of 0.0494, and an avg. R2 Score of 0.9261, surpassing the fixed delta results. This indicates that a dynamic delta may be more beneficial for this application.

The Quantile loss functions perform notably worse than the other loss functions, especially at $\tau = 0.25$, where the ARMSE is 0.8519, and the avg. R2 Score drops significantly to 0.4120. This suggests that the Quantile loss alone may not be suitable for this particular task or requires further tuning. At $\tau = 0.50$, the Quantile loss achieves better results but still lags behind other loss functions, with an ARMSE of 0.4285.

The Log-Cosh loss function results in an ARMSE of 0.3442 and an AMAE of 0.0525, which is competitive with the Huber loss configurations. This indicates that while Log-Cosh is effective, combining it with other loss functions in a hybrid approach yields better results. Overall, the Hybrid loss function, combining the Huber, Quantile, and Log-Cosh in a weighted approach, leads to superior performance, reducing the ARMSE by approximately 4.45% compared to the next best configuration (Huber - 0.75). The higher avg. R2 Score indicates better explanatory power of the model when using the Hybrid loss function.

S3.2 Impact of MARX Block on Model Performance

In this experiment, we examine the effect of varying the number of MARX blocks in the bottleneck of the CASPIAN-v2 model on flood inundation prediction accuracy. The MARX blocks are critical components designed to capture multi-attention features and enhance the model's learning capacity. We tested configurations with 2, 4, 6, and 8 MARX blocks, denoted as CASPIAN-v2$_{\text{MARX}\times n}$, where n represents the number of MARX blocks.

The base model architecture remains consistent with the previous experiment, comprising an encoder and decoder depth of 4 layers and 4 SEE blocks between each encoder and decoder layer. The SLR scalar integer is incorporated at the output of the bottleneck before the features are fed into the decoder. To ensure a fair comparison, we employed the Hybrid loss function identified as the best-performing loss function from the previous experiment.

As shown in Table S2, the number of MARX blocks has a significant impact on the model's performance. The CASPIAN-v2$_{\text{MARX}\times 4}$ configuration achieves the best overall performance, with the lowest ARMSE of 0.3136, the lowest AMAE of 0.0465, and the highest avg. R2 Score of 0.9356. This indicates that using 4 MARX blocks in the bottleneck effectively balances model complexity and learning capacity, leading to more accurate predictions.

Table S1. Impact of different loss functions on the CASPIAN-v2 performance.

Config	ARMSE ↓	AMAE ↓	Avg. R2 Score ↑	Model Size ↓
CASPIAN-v2$_{\text{BASE}}$ (Huber* - 0.25)	0.3976	0.0628	0.9129	1731257
CASPIAN-v2$_{\text{BASE}}$ (Huber* - 0.50)	0.3470	0.0534	0.9249	1731257
CASPIAN-v2$_{\text{BASE}}$ (Huber* - 0.75)	0.3435	0.0508	0.9236	1731257
CASPIAN-v2$_{\text{BASE}}$ (Huber$^{\wedge}$ 0.4 to 0.6)	0.3482	0.0527	0.9242	1731257
CASPIAN-v2$_{\text{BASE}}$ (Huber$^{\wedge}$ 0.3 to 0.7)	0.3282	0.0494	0.9261	1731257
CASPIAN-v2$_{\text{BASE}}$ (Huber$^{\wedge}$ 0.2 to 0.8)	0.3450	0.0520	0.9114	1731257
CASPIAN-v2$_{\text{BASE}}$ (Quantile* - 0.25)	0.8519	0.3213	0.4120	1731257
CASPIAN-v2$_{\text{BASE}}$ (Quantile* - 0.50)	0.4285	0.0681	0.8865	1731257
CASPIAN-v2$_{\text{BASE}}$ (Quantile* - 0.75)	0.5399	0.1034	0.8187	1731257
CASPIAN-v2$_{\text{BASE}}$ (Log-Cosh)	0.3442	0.0525	0.9251	1731257
CASPIAN-v2$_{\text{BASE}}$ (Hybrid$^{\#}$)	0.3136	0.0465	0.9356	1731257

* Fixed delta or τ value, $^{\wedge}$ Dynamic delta range, $^{\#}$ With 0.3 Huber, 0.5 Log-Cosh, and 0.2 Quantile loss weightages. The delta for Huber loss was set dynamically between 0.3 and 0.7, and τ for Quantile loss was fixed at 0.5.

When using only 2 MARX blocks (CASPIAN-v2$_{\text{MARX}\times 2}$), the model shows a decrease in performance, with an ARMSE of 0.3249 and an avg. R2 Score of 0.9213. This suggests that with fewer MARX blocks, the model may not capture sufficient multi-scale features, limiting its ability to generalize well on the validation set.

Increasing the number of MARX blocks to 6 (CASPIAN-v2$_{\text{MARX}\times 6}$) slightly improves the AMAE to 0.0469 compared to the 2-block configuration but does not surpass the performance of the 4-block model. The ARMSE remains higher at 0.3195, and the avg. R2 Score decreases to 0.9313. This indicates that adding more MARX blocks beyond a certain point does not necessarily lead to better performance and may introduce redundancy or overfitting.

Further increasing the number of MARX blocks to 8 (CASPIAN-v2$_{\text{MARX}\times 8}$) results in a noticeable decline in performance. The ARMSE increases to 0.3346, the AMAE rises to 0.0526, and the avg. R2 Score drops to 0.9184. Additionally, the model size increases significantly to 1.93 million parameters, which may lead to increased computational cost and longer training times without corresponding performance gains.

The trend observed suggests that there is an optimal number of MARX blocks that maximizes the model's ability to learn complex patterns without overfitting. The CASPIAN-v2$_{\text{MARX}\times 4}$ model strikes this balance effectively. By capturing sufficient multi-scale features through 4 MARX blocks, the model can generalize better and produce more accurate flood inundation predictions.

Table S2. Impact of different number of MARX blocks on the CASPIAN-v2 performance.

Config	ARMSE ↓	AMAE ↓	Avg. R2 Score ↑	Model Size ↓
CASPIAN-v2$_{\text{MARX}\times 2}$	0.3249	0.0505	0.9213	1630459
CASPIAN-v2$_{\text{MARX}\times 4}$	0.3136	0.0465	0.9356	1731257
CASPIAN-v2$_{\text{MARX}\times 6}$	0.3195	0.0469	0.9313	1832055
CASPIAN-v2$_{\text{MARX}\times 8}$	0.3346	0.0526	0.9184	1932853

S3.3 Impact of SEE Block on Model Performance

In this experiment, we assess the effect of varying the number of SEE blocks on the performance of the CASPIAN-v2 model for flood inundation prediction. The SEE blocks are designed to integrate the SLR features between the encoder and decoder layers, contributing to the model's ability to capture SLR-induced details in flood maps. We tested configurations with 1, 2, 3, and 4 SEE blocks, denoted as CASPIAN-v2$_{\text{SEE}\times n}$, where n indicates the number of SEE blocks used.

Table S3. Impact of different number of SEE blocks on the CASPIAN-v2 performance.

Config	ARMSE ↓	AMAE ↓	Avg. R2 Score ↑	Model Size ↓
CASPIAN-v2$_{\text{SEE}\times 1}$	0.3212	0.0485	0.9281	405938
CASPIAN-v2$_{\text{SEE}\times 2}$	0.3196	0.0479	0.9311	472927
CASPIAN-v2$_{\text{SEE}\times 3}$	0.3172	0.0475	0.9315	727308
CASPIAN-v2$_{\text{SEE}\times 4}$	0.3136	0.0465	0.9356	1731257

The base model architecture is consistent with the previous experiments, featuring an encoder and decoder depth of 4 layers and 4 MARX blocks in the bottleneck, as determined optimal from prior analysis. The SLR scalar integer is integrated at the output of the bottleneck before feeding the features into the decoder. We utilized the Hybrid loss function, identified as the best-performing loss function in earlier experiments.

As depicted in Table S3, increasing the number of SEE blocks positively impacts the model's performance up to a certain point. The CASPIAN-v2$_{\text{SEE}\times 4}$ configuration demonstrates the best overall performance, achieving the lowest ARMSE of 0.3136, the lowest AMAE of 0.0465, and the highest avg. R2 Score of 0.9356. This suggests that utilizing four SEE blocks enhances the model's ability to capture spatial edge features effectively, leading to more precise flood inundation predictions.

Starting with a single SEE block (CASPIAN-v2$_{\text{SEE}\times 1}$), the model achieves an ARMSE of 0.3212 and an avg. R2 Score of 0.9281. While this configuration performs reasonably well, it lacks the detailed spatial feature enhancement provided by additional SEE blocks. Adding more blocks incrementally improves performance; with two SEE blocks (CASPIAN-v2$_{\text{SEE}\times 2}$), the ARMSE reduces to 0.3196 and the avg. R2 Score increases to 0.9311. Similarly, three SEE blocks (CASPIAN-v2$_{\text{SEE}\times 3}$) result in an ARMSE of 0.3172 and an avg. R2 Score of 0.9315. The most significant improvement is observed with four SEE

blocks, where the avg. R2 Score rises to 0.9356, reflecting a stronger correlation between predicted and actual flood inundation levels.

However, while performance improves with additional SEE blocks, the gain diminishes beyond two blocks. Moreover, the number of parameters grows significantly, from 0.41 million parameters for CASPIAN-v2$_{\text{SEE}\times 1}$ to 1.73 million parameters for CASPIAN-v2$_{\text{SEE}\times 4}$. This increase leads to higher computational costs and longer training times, making the model less suitable for real-time applications.

Given the relatively modest performance gains compared to the steep increase in model size, we propose using a single SEE block in the final model to strike a balance between performance and efficiency. This minimalistic design ensures the model remains lightweight and suitable for real-time flood prediction tasks, aligning with the research's goal of developing a computationally efficient solution. Future work may explore alternative strategies, such as model pruning or efficient network designs, to further optimize performance without significantly increasing complexity.

S3.4 SLR Integration at Different Network Stages

In this experiment, we investigate the effect of integrating the SLR scalar integer at various stages within the CASPIAN-v2 model architecture on flood inundation prediction accuracy. The SLR scalar is a critical input representing different sea-level rise scenarios, and understanding the optimal point of integration within the network can enhance the model's predictive capabilities. We explored integrating the SLR scalar at three key stages of the network: at the end of the bottleneck (BN), at the end of the last FR block in decoder stage (FR), and before the final output layer (End). Additionally, we examined combinations of these integration points with the SEE blocks to assess their combined effect on performance.

Table S4. Performance evaluation of CASPIAN-v2 with integrating SLR at different stages.

Config	ARMSE ↓	AMAE ↓	Avg. R2 Score ↑	Model Size ↓
CASPIAN-v2$_{\text{SLR}\rightarrow\text{BN}}$	0.3212	0.0485	0.9281	405938
CASPIAN-v2$_{\text{SLR}\rightarrow\text{BN+SEE}}$	0.3196	0.0465	0.9313	405942
CASPIAN-v2$_{\text{SLR}\rightarrow\text{FR}}$	0.3475	0.0597	0.9169	382950
CASPIAN-v2$_{\text{SLR}\rightarrow\text{SEE+FR}}$	0.3259	0.0511	0.9170	383022
CASPIAN-v2$_{\text{SLR}\rightarrow\text{End}}$	0.3199	0.0515	0.9306	383020
CASPIAN-v2$_{\text{SLR}\rightarrow\text{SEE+End}}$	0.3126	0.0459	0.9320	383022

For consistency, all configurations employed 4 MARX blocks and utilized the Hybrid loss function identified as optimal in previous experiments. We used only 1 SEE block in these experiments, as prior analysis indicated that increasing the number of SEE blocks did not significantly enhance performance but did increase the number of parameters.

The results in Table S4 indicate that the point at which the SLR scalar is integrated into the network significantly impacts the model's performance. Integrating SLR at the end of the bottleneck layer (CASPIAN-v2$_{\text{SLR}\rightarrow\text{BN}}$) achieves an ARMSE of

0.3212 and an AMAE of 0.0485. Incorporating SEE blocks in this configuration (CASPIAN-v2$_{\text{SLR}\rightarrow\text{BN+SEE}}$) slightly improves performance, reducing ARMSE to 0.3196 and AMAE to 0.0465, and increasing the avg. R2 Score to 0.9313.

When integrating SLR at the end of the last FR block (CASPIAN-v2$_{\text{SLR}\rightarrow\text{FR}}$), the model shows the poorest performance with an ARMSE of 0.3475 and an AMAE of 0.0597. Adding SEE blocks to this configuration (CASPIAN-v2$_{\text{SLR}\rightarrow\text{SEE+US}}$) improves the ARMSE to 0.3259 and AMAE to 0.0511, but the performance remains inferior compared to other configurations.

Integrating SLR before the final output layer (CASPIAN-v2$_{\text{SLR}\rightarrow\text{End}}$) achieves an ARMSE of 0.3199 and an AMAE of 0.0515. The best performance is observed when combining SEE blocks with this integration point (CASPIAN-v2$_{\text{SLR}\rightarrow\text{SEE+End}}$), achieving the lowest ARMSE of 0.3126, the lowest AMAE of 0.0459, and the highest avg. R2 Score of 0.9320.

These findings suggest that integrating the SLR scalar before the final output layer, especially in combination with SEE blocks, allows the model to more effectively leverage the SLR information for accurate flood inundation predictions. The model sizes remain relatively consistent across configurations, with minor differences due to the number of parameters introduced by the SLR integration and SEE blocks. The best-performing model (CASPIAN-v2$_{\text{SLR}\rightarrow\text{SEE+End}}$) has a model size of 0.38 million parameters, which is efficient given its superior performance.

S4 Holdout Dataset

To thoroughly evaluate the ability of our model to handle challenging conditions, we curated a specialized holdout set for both the AD and SF regions. The scenarios below were chosen based on the spatial configuration and proximity of the OLUs to ensure diverse yet demanding test cases for the model. Tables S5 and S6 list all holdout scenarios for the AD and SF regions, respectively.

Table S5. Holdout set scenarios for the AD region. `1` indicates a protected OLU, while `0` denotes an unprotected OLU.

AD Scenarios			
00000001111110000	00000001111111000	00000011111000000	00000011111100000
00000111100000111	00000111110000011	00000111111100000	00001110000111000
00001111000011110	00001111111110000	00011000110001100	00011100011100011
00011111111111000	00110011001100110	00111111111111100	01010101010101010
10101010101010101	11000000000000011	11001100110011001	11100000000000111
11100011100011100	11100111001110011	11110000000001111	11110000111100001
11110001111000111	11111000000011111	11111000001111100	11111000011111000
11111100000011111	11111100000111111	11111110000000111	11111110000001111

S5 Generalizability Dataset

We further evaluated generalizability of the CASPIAN-v2 model for unseen SLR conditions of 0.5 m and 1.5 m. This dataset contains 32 protection scenarios for the SF region: 30 scenarios each with exactly one protected OLU (and the remaining

Table S6. Holdout set scenarios for the SF region. `1` indicates a protected OLU, while `0` denotes an unprotected OLU.

SF Scenarios		
000001111111111111100000000000	000011111111100001111111010101	000111000111000111000111000111
001000000110101010111001011111	001001100011010100010000110101	001100110011001100110011001100
001100111100101000111010000010	001101101111101000010100001001	001111111010111000001001001100
010011000111110100101010000000	010100001011101110100101100001	010101010101010101010101010101
010101100100010101111100010000	011000001010000011110001111000	011000111000100011000001110010
011000111100001111001101001110	011010010000101000111110110100	011011000000011111011000100101
011100010110000001100011001011	011101000010011011111110001010	011101011000011111110101011001
011110110100101000001001101110	011111000111000101011010001001	100000000010100101001101111111
100010101001111100110000100100	100100000111111000001001001110	100101011111010111011101001100
101001101011010011000100100110	101010000010001101100100001010	101010101010101010101010101010
101011000111111100110100001100	101100111110011101010100111101	101110001110001001111001001001
110000101001111101011001111101	110001101101101111101101000110	110011001100110011001100110011
110111100111100111000010100001	110111110011100101000010001100	111000111000111000111000111000
111001111010010011111010110010	111100100010111111101100110100	111101111001000101111101100011
111110001000010010110111011100	111111100000000000001111111111	111111100011111000011110000000
111111110000000011111111100000		

235 unprotected), a completely unprotected scenario, and a fully protected scenario. Table S7 lists these configurations in binary form, where `0` and `1` denote unprotected and protected OLUs, respectively.

Table S7. Generalizability set scenarios for the SF region under 0.5 m and 1.5 m SLR. Each row contains binary strings of length 30, with `1` indicating a protected OLU and `0` indicating an unprotected OLU.

SLR Generalizability Scenarios (SF)		
000000000000000000000000000000	000000000000000000000000000001	000000000000000000000000000010
000000000000000000000000000100	000000000000000000000000001000	000000000000000000000000010000
000000000000000000000000100000	000000000000000000000001000000	000000000000000000000010000000
000000000000000000000100000000	000000000000000000001000000000	000000000000000000010000000000
000000000000000000100000000000	000000000000000001000000000000	000000000000000010000000000000
000000000000000100000000000000	000000000000001000000000000000	000000000000010000000000000000
000000000000010000000000000000	000000000001000000000000000000	000000000010000000000000000000
000000001000000000000000000000	000000010000000000000000000000	000000100000000000000000000000
000001000000000000000000000000	000001000000000000000000000000	000010000000000000000000000000
000100000000000000000000000000	001000000000000000000000000000	010000000000000000000000000000
100000000000000000000000000000	111111111111111111111111111111	

References

Barnard, P. L., van Ormondt, M., Erikson, L. H., Eshleman, J., Hapke, C., Ruggiero, P., Adams, P. N., and Foxgrover, A. C.: Development of the Coastal Storm Modeling System (CoSMoS) for predicting the impact of storms on high-energy, active-margin coasts, Natural hazards, 74, 1095–1125, 2014.

Chow, A. C. and Sun, J.: Combining Sea level rise inundation impacts, tidal flooding and extreme wind events along the Abu Dhabi coastline, Hydrology, 9, 143, 2022.

Delft University of Technology: Simulating Waves Nearshore (SWAN), https://swanmodel.sourceforge.io, 2022.

Deltares: Delft3d, https://oss.deltares.nl/web/delft3d.

Huber, P. J.: Robust estimation of a location parameter, in: Breakthroughs in statistics: Methodology and distribution, pp. 492–518, Springer, 1992.

Koenker, R. and Bassett Jr, G.: Regression quantiles, Econometrica: journal of the Econometric Society, pp. 33–50, 1978.

Saleh, R. A. and Saleh, A.: Statistical properties of the log-cosh loss function used in machine learning, arXiv preprint arXiv:2208.04564, 2022.

Sun, J., Chow, A. C., and Madanat, S. M.: Multimodal transportation system protection against sea level rise, Transportation Research Part D: Transport and Environment, 88, 102 568, 2020.