We thank the reviewer for the careful and constructive evaluation of our manuscript. We greatly appreciate the insightful comments, which have helped us improve the clarity, methodological transparency, and contextualization of our work. Below we provide a detailed, point-by-point response to each comment. In the remainder, reviewer comments are reported in bold-italics, followed by our responses. All changes are incorporated in the revised manuscript, with explicit references to sections, figures, and supplementary material.

During the preparation of the revised manuscript, we identified an implementation issue in the inference pipeline: predictor variables in the testing and case-study datasets were inadvertently standardized using statistics computed from the testing set itself rather than the scaler fitted on the training data. This was inconsistent with the preprocessing applied during model training. The pipeline has now been corrected so that all predictors in the testing set and in the November 2022 inference experiment are standardized using the scaler derived from the training data.

We are also grateful to the reviewer for several comments that prompted us to re-examine and further improve the implementation of the recurrent architectures. During the revision process, we refined the recurrent models by fully adopting a sequence-to-sequence formulation combined with Truncated Backpropagation Through Time (TBPTT), thereby improving the consistency with which temporal dependencies are represented during both training and inference. The revised manuscript now includes a detailed description of this implementation in Section 2.4. In addition, several reviewer comments motivated new sensitivity analyses and methodological clarifications, which have strengthened both the robustness and transparency of the study.

After implementing these corrections, we recomputed the full set of experiments, including the 40 runs for each emulator architecture and the November 2022 case study. The figures and quantitative results have been updated accordingly. While quantitative differences are observed, the overall performance patterns and qualitative behavior of the models remain consistent with the original submission, and the main conclusions of the manuscript are unchanged.

Overall, these revisions improve the methodological consistency of the study while preserving the central findings, namely that the interaction between loss-function design and model architecture plays a central role in accurately reproducing extreme storm surge events.

1. The temporal prediction strategy is not mentioned in the manuscript. It is not stated whether the rolling window approach or a one-shot strategy was applied, along with the related input window and prediction window sizes. This is important for understanding the predictive capability of the model in practice.

We thank the reviewer for this important point regarding the temporal prediction strategy adopted by the recurrent architectures. In the original manuscript, the implementation details of the temporal formulation were not sufficiently described. To address this, we substantially

expanded Section 2.4 to clarify both the adopted recurrent training strategy and the additional sensitivity experiments performed during the revision process. In the revised manuscript, we now explicitly describe that the recurrent architectures are trained using a full-sequence formulation combined with Truncated Backpropagation Through Time (TBPTT). The following can now be found in the revised manuscript (L190-204): *The recurrent architectures (RNN and LSTM families) are trained on the full time series as a single continuous sequence, enabling the exploitation of temporal dependencies through their internal state dynamics. To make this approach computationally feasible for long records, training is carried out using truncated backpropagation through time (TBPTT). In this framework, the full sequence is partitioned into shorter temporal segments (chunks), which are processed sequentially during training. In this study, a chunk length of 1024 time steps (hours) is adopted, corresponding to approximately 43 days. This temporal extent is well beyond the characteristic duration of storm surge events, providing sufficient context for the model to capture physically relevant dependencies.*

Crucially, the hidden state of the recurrent network is propagated across consecutive segments, allowing the model to retain information from previous time steps, while gradients are truncated at segment boundaries to control memory usage and ensure numerical stability. This strategy enables the model to approximate full-sequence learning while avoiding the prohibitive computational cost of backpropagating through the entire time series at once. As a result, temporal dependencies are not explicitly encoded in the input feature space (i.e. no lookback window is constructed), but are instead learned implicitly through the evolution of the hidden state. This formulation allows recurrent models to exploit temporal context without requiring manual construction of lagged predictors.

Additionally, following the reviewer's comment and to ensure that the temporal capabilities of the recurrent architectures were being properly exploited, we implemented and evaluated an explicit sliding-window (lookback window) formulation. Several lookback lengths (3, 12, 24, and 36 hours) were tested and compared against the TBPTT approach using the LSTM-MSE emulator as baseline. The corresponding methodology and results are now presented in Section 2.4 and Table S1. The additional experiments showed that, for the present study area and predictor configuration, the full-sequence + TBPTT formulation generally outperformed the sliding-window approach, particularly in terms of bias and overall error characteristics for extreme surge events. Nevertheless, we also clarify that this conclusion may depend on the study area and predictor configuration, and therefore should not be interpreted as universally favouring TBPTT over sliding-window approaches in all coastal environments. The following can now be found in the revised manuscript (L206-231): *To further assess the adequacy of the recurrent formulation and to explicitly evaluate the role of temporal context, an additional set of experiments was conducted using a sliding window approach. In this formulation, the input at time t consists of a fixed-length window of past predictors, i.e. $\{x(t-L+1), \ldots, x(t)\}$, where L denotes the look-back length. This approach provides an explicit representation of temporal memory in the input space, in contrast to the implicit memory learned through the hidden state in the TBPTT formulation. The comparison between both approaches was designed to ensure*

that the use of recurrent architectures is being properly exploited and that the adopted TBPTT strategy does not artificially constrain the temporal dependencies learned by the model.

The analysis was performed using the LSTM-MSE emulator as a baseline, following Hermans et al. (2025), and results were averaged across locations for surge peaks exceeding the 99th percentile. Each emulator configuration was trained ten times using different random initializations, and the reported performance corresponds to the best run, selected based on the linear fit slope closest to unity on the validation set. The validation dataset is described in Section 2.6. To assess the role of explicitly defined temporal memory, different look-back windows were tested (3, 12, 24, and 36 hours), and the corresponding performance metrics are summarized in Table S1.

Overall, the full-time series + TBPTT approach consistently outperforms the sliding window formulation in our experiment, particularly in terms of bias and overall error characteristics. The TBPTT model exhibits a lower bias (-0.058) compared to all sliding window configurations (ranging from -0.083 to -0.102), indicating a reduced systematic underestimation of extreme surge values. While Pearson correlation and RMSE remain broadly comparable across approaches, the sliding window configurations tend to exhibit larger errors and a more pronounced bias, especially for both shorter and longer look-back windows. Furthermore, increasing the look-back length does not lead to a consistent improvement in performance. Although intermediate windows (e.g., 24 hours) yield results closer to the TBPTT formulation, longer windows (e.g., 36 hours) generally degrade performance. This suggests that explicitly increasing the input memory does not enhance predictive skill in the study area and may instead introduce noise or reduce model robustness. Nevertheless, these findings are likely conditioned by the characteristics of the study area and the predictor configuration adopted here, and do not necessarily imply that the TBPTT formulation will systematically outperform sliding-window approaches in other coastal environments or surge regimes.

2. The hyperparameter tuning procedure is not presented. Ranges of the varied parameters along with the respective range evaluation metrics should be provided. Using MSE loss during hyperparameter tuning further enhances the arguments in favor of the MADc² - based models, however this choice and the respective limitations (as for instance, using MADc² loss for tuning might probably further improve the models) should be mentioned.

We thank you the reviewer for this comment. To ensure full reproducibility and clarify the extent of the hyperparameter optimization, the descriptions have been expanded to document all relevant hyperparameters, including the following:

- Section 2.4 L233-238: *The architectural hyperparameters of the neural network emulators were selected through an exploratory grid search designed to balance emulator complexity, robustness, and computational efficiency. For the MLP architecture, different configurations of hidden layers were tested, including one- and two-layer networks with 60 and 120 neurons per layer. For the recurrent architectures (RNN and*

LSTM), the number of hidden units was varied between 30, 60, and 120. In addition, multiple random weight initializations were considered for each configuration to account for stochastic variability in gradient-based optimization.

- Section 2.5 L240-251: *All emulators were trained using the Adam optimizer with a fixed learning rate of* $1x10^{-3}$. *An exception was made for the RNN-based emulators, for which a reduced learning rate of* $5x10^{-4}$ *was adopted to ensure stable training and mitigate gradient explosion effects inherent to simple (non-gated) recurrent neural network architectures. All remaining training parameters were kept fixed across architectures to ensure a consistent comparison and to isolate the impact of architectural choices and loss-function design. In particular, all emulators were trained using full-batch gradient descent, with each epoch processing the entire training dataset, while the optimizer type, learning rate, and input normalization were held constant across experiments. Hyperparameter selection for each model was based on validation performance using the slope of the linear fit between predicted and observed storm surge, as this metric directly reflects the ability of the emulator to reproduce the amplitude of extreme events. The configurations yielding the most robust validation performance across random initializations were retained for the full training experiments reported in this study. No additional regularization (e.g., dropout or weight decay) was applied, as preliminary tests did not indicate overfitting under the adopted data partitioning and full-batch training regime.*
- Section 2.5 L315-322: *The hyperparameter exploration described above was performed independently of the loss function. Once the optimal architectural configuration for each emulator type was identified based on validation slope performance, the same architecture was trained using both MSE and MADc² loss functions. This ensured that performance differences between MSE- and MADc²-trained models reflect solely the influence of the loss formulation, rather than architecture-specific tuning differences. We note that hyperparameter selection was based on the validation slope criterion rather than directly minimizing either MSE or MADc². While it is conceivable that tuning architectures explicitly under MADc² could further enhance extreme-event performance, adopting a common validation-based selection strategy across both loss functions ensures a controlled and comparable experimental framework.*
- Section 2.6 L359-363: *For the neural network emulators, each run consisted of 800 training epochs, while the MLR emulator was trained for 10000 iterations to ensure convergence. These values were determined through preliminary convergence tests, in which training was extended until improvements in validation performance became negligible and performance metrics reached a stable plateau. The combination of multiple runs and sufficiently long training ensures both reproducibility and reliable estimation of emulator skill.*

3. There is in inconsistency regarding the choice of the "best model". In L185, it is stated that "This slope was therefore used to identify the best-performing model run", with the slope also

being mentioned as the criterion for chosing a "best model" in the violin plot captions. However, in L240, it is stated that: "each emulator architecture was trained and evaluated on the testing set 40 times. For each run and location, statistical metrics were computed separately and then averaged across Punta della Salute and Trieste, yielding one set of values per run". So In the evaluations in sections 3.3 and 3.4, what model was used? The one determined by best slope during validation, or the average per location?

We thank the reviewer for pointing out the need for clarification. The relevant paragraph has been revised (L353–357) to explicitly state the following in the revised manuscript: *Gradient-based training methods introduce randomness into the optimization process, so the final parameter set is not identical across different training runs. For this reason, each ML emulator was trained 40 times, with each run corresponding to a unique random initialization of model weights. The number of repeated runs was selected empirically by progressively increasing the number of realizations and verifying that the distributions of performance metrics stabilized, indicating robust sampling of initialization-induced variability.*

Regarding the selection of the best model, the revised manuscript now states the following (L365-377): *Model selection was based on validation performance, using a validation-based model selection strategy, a widely adopted approach for identifying optimal configurations during training (Bishop, 2006; Goodfellow et al., 2016). Several performance metrics were tested on the validation set, including both general metrics over the entire period and specialized ones computed over storm surge events exceeding the 99th percentile. The 99th percentile was selected as a compromise between focusing on rare, high-impact events and retaining a sufficiently large sample to ensure statistical robustness, allowing us to characterize extreme conditions while avoiding the instability associated with very small sample sizes at higher percentiles (Wahl et al., 2017). Among the tested metrics, the slope of the linear fit between emulator output and observations emerged as the most effective criterion, as it directly quantifies the ability of the models to reproduce the dynamic range and intensity of extreme surges while avoiding systematic over- or underestimation. While recent studies have shown that data-driven models may exhibit reduced skill at even more extreme percentiles (e.g., above the 99.9th percentile), a systematic evaluation at such thresholds is beyond the scope of the present work due to the limited number of events. Nevertheless, the behavior of the models in the upper tail of the distribution is discussed in Section 4 (Figure 12), providing additional qualitative insight into performance under the most extreme conditions.*

For detailed diagnostic analyses (scatter plots, case studies, CDF comparisons in Sections 3.3–3.4), a single run was selected based on the validation-based slope criterion (slope closest to 1). The following can now be found in the revised manuscript (L430-437): *As noted earlier, each emulator architecture was trained 40 times using different random initializations. Performance metrics were computed separately for each run and location, and subsequently averaged across Punta della Salute and Trieste, yielding one set of values per run. Violin plots in Section 3 display the full distribution of these 40 realizations. For diagnostic analyses, scatter plots, cumulative distribution comparisons, and case studies (Sections 3.3–3.4), a single run was selected based on the validation-based slope criterion described in Section 2.6, i.e., the run*

with slope closest to 1. This ensures consistency between model selection and detailed performance assessment. Analyses were conducted on both the full dataset and on surge peaks above the 99th percentile of the predictand distribution. The full dataset was assessed using RMSE, MADp, and MADc, while surge peaks were evaluated with bias, MADp, and MADc.

This ensures methodological consistency between model selection and detailed performance assessment.

4: In L107: "The first seven principal components (PCs) of each predictor were used independently, without merging into a multivariate series". What is the difference? Isn't the input a 2D array, with one dimension being the time and the other the different features? Some schematic on the feature dimensionality might improve the presentation. Also in L109, more information should be provided for the MLR performance tests (e.g., on the whole test set? Best model?)

We have clarified this point in Section 2.3, for which we added the following in the revised manuscript (L150-168): *The spatial domain for predictor extraction was selected via performance testing with multivariate linear regression (MLR) emulators and is shown in Figure 1a and in the Supplementary Information (Figure S1). MLR was used at this stage as a computationally efficient baseline, allowing systematic exploration of multiple domain configurations while preserving physical interpretability of the predictors. Since most predictors are spatial fields and the target is time-series regression, dimensionality reduction was applied using Principal Component Analysis (PCA). The first seven principal components (PCs) of each predictor were retained and used independently, without merging into a multivariate series. This number was selected based on MLR performance tests and represents a compromise between information retention and dimensionality reduction. Tidal data were excluded from this process and used directly as single hourly time series. The selected EOFs for each predictor are shown in Figures S2–S5 and the explained variance of each PC is shown in Figure S6. Since four spatial predictors were considered (sea surface height, wind stress components, and mean sea level pressure), this results in 28 PCA-derived features per time step. These features, together with the tidal time series, form the final input vector at each time step. The resulting predictor matrix therefore has dimensions* $[N_{time}, N_{features}]$, *where* $N_{features}$ *corresponds to the total number of retained PCs plus tides.*

Predictor extraction and PCA were performed separately for each location using the corresponding spatial domain (Figure 1a). Although the same types of basin-scale variables (sea surface height, mean sea level pressure, wind stress components, and tides) were used, the resulting predictor matrices are location-specific, reflecting differences in spatial footprint and local coastal dynamics. For reproducibility and clarity, Table 1 summarizes the predictor variables used in the ML emulators, their data sources, preprocessing steps, and resulting feature dimensionality.

Additionally, we added Table 1 with the list of predictors, data source, preprocessing carried out, and the number of features per location.

5. In L145, the synthetic case study is unclear, and the caption of the respective Figure (S7) is probably wrong.

We have expanded the description of the synthetic experiment in Section 2.4 to clearly describe it. The following can now be found in the revised manuscript (L280-300): *Building on this definition, the adoption of MADc² warrants a brief discussion from both theoretical and practical perspectives. Here, we focus on its definition as the square of MADc and on its convexity, i.e., the existence of a single global minimum, which underpins the robustness of gradient-based optimization algorithms. MADc² is preferred over MADc because it is differentiable at the theoretical minimum (MADc² = 0), whereas MADc is not. This differentiability enables smoother convergence in optimization routines that rely on gradient information.*

From a theoretical standpoint, MADc² is convex: for an ideal model that perfectly reproduces the observations, both the MAD and the MADp vanish. The loss function therefore reaches its unique global minimum at this point of perfect agreement, with no other minima admissible. In practice, however, convexity may be locally attenuated if the parameter values that minimize MAD differ from those minimizing MADp. In such cases, compensation effects between the two components can generate shallow valleys or low-gradient regions in parameter space, potentially slowing convergence.

To illustrate the optimization properties of MADc², we constructed a synthetic case study based on a simple linear regression model of the form $\hat{y} = Wx + B$, where W and B denote single weight and bias parameter. Synthetic data were generated using known coefficients ($W = 2, B = 1$), and the MADc² loss was evaluated over a grid of parameter values in the vicinity of the true solution. Figure S7 shows the resulting two-dimensional loss surface as a function of W and B, with colors representing $log_{10}(MADc^2)$. The surface exhibits a single well-defined minimum located near the true coefficients and a smooth diagonal valley structure, with no evidence of multiple local minima in the explored region. While this simplified experiment does not constitute a formal proof of convexity in high-dimensional neural network parameter spaces, it provides a practical illustration that MADc² yields a smooth and well-behaved optimization landscape in a representative regression setting, supporting the stability of gradient-based training.

Additionally, the caption of Figure S7 has been corrected and now properly describes the MADc² loss surface and its logarithmic scaling.

6. Around L260, it is mentioned that: "the MADc²-trained emulators exhibit performance comparable to SHYFEM-MPI, with a significant portion of their distributions outperforming the benchmark (Figure 2b)". This is quite an overstatement, as only a portion of the low tails are below the benchmark lines. In Figure 2, indices a-c are also missing in the graphic.

We agree with the reviewer and have revised the complete description of Figure 2 to read (L451-465): *The RMSE distributions across the 40 training experiments indicate that emulators trained with the MSE loss generally achieve lower RMSE values than their MADc²-trained counterparts in the testing set (Figure 2a). Among them, the MLR-MSE emulator ranks as the top performer in terms of RMSE. While several MADc²-based emulators also surpass the SHYFEM-MPI benchmark on this metric, their distributions tend to be broader, reflecting increased variability across runs.*

Regarding MADp, the effect of the MADc² loss depends on the emulator architecture (Figure 2b). The clearest improvements are observed for the MLR and MLP emulators, whose MADp distributions shift toward lower values relative to their MSE-trained counterparts and below the SHYFEM-MPI benchmark. In contrast, recurrent architectures show smaller and less systematic differences between loss functions, with substantial overlap between the corresponding distributions.

For MADc, the MLR-MADc² and MLP-MADc² emulators again exhibit the clearest improvements relative to their MSE-trained counterparts (Figure 2c). For recurrent architectures, however, the effect of MADc² is less pronounced, with only slight improvements observed for the selected LSTM-based runs (red stars). Overall, these results indicate that the impact of MADc² on full time-series performance is architecture-dependent, improving percentile-based metrics for some emulators while providing more limited benefits for others.

Additionally,panel indices (a–c) have also been added to Figure 2.

7. In Figure 2c, it is shown that all MADc² models appear to have greater variability than their MSE counterparts when evaluated with the MADc metric. An explanation to that would be interesting for interpreting the behavior of MADc² as a loss metric.

We have added a detailed explanation in Section 3.1, clarifying that the increased variability arises from the composite nature of MADc², which jointly minimizes pointwise deviations and percentile-based discrepancies. The following can now be found in the revised manuscript (L467-473): *The increased variability observed in MADc²-trained emulators, particularly for recurrent architectures (Figure 2c), likely reflects the multi-objective nature of the loss function. By jointly penalizing pointwise deviations and discrepancies in the distributional structure, MADc² introduces a more complex optimization landscape in which different parameter configurations can achieve comparable loss values through different trade-offs between components. This can increase sensitivity to initialization and training dynamics, especially in*

recurrent models, leading to a broader spread of outcomes. Conversely, the MSE loss defines a single quadratic objective, which tends to produce more stable solutions but does not explicitly enforce agreement in the distributional characteristics of the target signal.

8: In L116, is this a temporal permutation (i.e., are the values in each timeseries suffled in order)?

R: We clarified in Section 2.3 that permutation was performed independently at each time step, disrupting temporal alignment between feature and target. The description of the permutation analysis now reads (L173-181): *To assess the relative influence of the input features on emulator predictions, a permutation importance analysis was conducted. The permutation was performed independently at each time step, thereby disrupting the temporal alignment between the feature and the target variable. This approach isolates the instantaneous contribution of each predictor to the regression task and is consistent with the contemporaneous formulation of the emulators. The trained emulator then generates predictions on the permuted dataset, and the resulting mean absolute deviation (MAD) is compared with the baseline MAD obtained from the original inputs. The increase in MAD reflects the feature's importance, with larger increases indicating a stronger influence on predictions. To mitigate randomness, each permutation was repeated 10 times, and the final importance score was computed as the average increase in MAD. This procedure offers a model-agnostic and interpretable assessment of feature relevance, enabling the identification of the most influential predictors driving the emulator's performance.*

9: Around L285 and in Figure 4: Is this the improvement compared to the MSE models or the numerical model baseline? Probably the first, but it could be better to clarify. In the text, the discussion reads quite cumbersome with all the percentages included. It would be better to keep only the take-home message of these statistics. In the figure caption, the term "variability" is a bit misleading. Probably the term "deviation" would be more appropriate (?). The authors could also state the formula to clarify.

We clarified in the caption of Figure 4 that the percentage deviation is computed relative to the MSE-trained counterpart:

$$Metric_{MSE} - Metric_{MADc^2} / Metric_{MSE} \times 100$$

The Results section has been revised to reduce excessive listing of percentages and instead emphasize the main takeaway, reading as follows (L496-507): *Figure 4 shows the mean percentage change in performance for surge peaks above the 99th percentile when using the MADc^2 loss instead of MSE, based on the selected runs. The MLR emulator exhibits the most pronounced and consistent improvements across nearly all metrics, particularly for bias, MADp, and MADc, where error reductions exceed 40–50%. These gains indicate a substantial enhancement in the representation of the distributional structure and magnitude of extreme surge events when training with MADc^2. The MLP emulator also benefits from the MADc^2 loss,*

showing moderate but systematic improvements across most metrics, especially for bias-related and percentile-based measures.

Among recurrent architectures, the response to MADc² training is more variable. The RNNh and LSTMh emulators show consistent positive changes across most metrics, with particularly noticeable improvements in bias, MADp, and MADc for LSTMh. In contrast, the standard RNN emulator exhibits only limited gains and slight degradations in some distribution-based metrics, indicating a less stable optimization behaviour. The standard LSTM emulator presents a mixed response, with improvements in SLF but small degradations in RMSE, MAD, and bias.

10. In L415-L420: The PIT histogram could explained a bit further, particularly on what is the meaning of the x-axis, for less informed readers.

The PIT description has been expanded (Section 4) to explicitly define the x-axis as the cumulative probability (percentile position) of the observed value within the emulator's empirical distribution. We also added an explanation of how deviations from uniformity (U-shaped or dome-shaped histograms) reflect under- or over-dispersion. In the revised manuscript, its description reads as follows (L669-682): *The benefit of this formulation is further evidenced by the Probability Integral Transform (PIT) histograms (Figure 9), shown here for the MLR emulators. For each time step, the PIT value corresponds to the cumulative probability of the observed surge under the empirical distribution of emulator predictions. In practice, this is computed as the percentile position of the observed value within the distribution defined by the emulator outputs. The x-axis of Fig. 9 therefore represents these PIT values, which range between 0 and 1.*

PIT histograms provide a direct diagnostic of predictive calibration: if the emulator reproduces the full distribution of observed values without systematic bias, the PIT values should follow a uniform distribution on [0,1]. Deviations from uniformity indicate calibration errors. For example, U-shaped histograms suggest underdispersion (extremes occurring more frequently than predicted), whereas dome-shaped histograms indicate overdispersion. While the MLR-MSE emulator shows visible departures from uniformity (Figures 9a and 9c), particularly under-representing the lowest and highest percentiles, the MLR-MADc² emulator produces a distribution much closer to uniform (Figures 9b and 9d). This result indicates improved distributional calibration and confirms that MADc² enhances not only pointwise accuracy at extremes but also the overall statistical consistency of the predicted surge distribution.

11. A few figure captions include typos: in Figure 3, "bias" should be used instead of "RMSE". In Figure 11, "LSTMh" should be used instead of "MADc²". In Figure 12, (a), (b) is for Trieste and (c), (d) is for Punta della Salute.

R: We thank the reviewer for identifying these issues. All caption typos have been corrected.

12. The limitations of this study should be more thoroughly discussed. While this is a very interesting case study to benchmark the MADc² loss metric that the authors previously assessed (and its advantages over MSE) in a few ML architectures, it should be clarified that factors such as the limited number of stations, the use of only a few time segments for testing (and potentially the prediction horizon - not sure if applicable) that were used in this assessment limit the scope of this study from extracting further conclusions on the overall performance of ML emulators in real time scenarios.

We thank the reviewer for this comment. In the revised manuscript, a dedicated limitations paragraph has been added in Section 4, explicitly addressing:

- The restriction to two stations.
- The use of a single temporal split.
- The instantaneous regression formulation.
- PCA-based dimensionality reduction.
- Lack of real-time operational evaluation.

The mentioned paragraph is the following (L603-610): *As with any data-driven modeling study, several limitations should be acknowledged. First, the analysis is restricted to two tide-gauge locations in the northern Adriatic Sea, which limits the direct generalization of the results to coastal environments with different dynamical regimes. Second, model evaluation relies on a single temporal split between training, validation, and testing periods rather than a full cross-validation framework, which may introduce sampling dependence despite the representativeness of the extreme events in the testing set. Finally, the emulator configurations considered here represent a specific design choice in terms of predictor representation, model architecture, and operational integration. These aspects influence model behavior and therefore require careful interpretation. The following sections discuss these methodological considerations in more detail and outline directions for future research.*

We agree that these aspects constrain the generalizability of conclusions and have clearly acknowledged them in the revised manuscript.