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.

Main comments

1. According to the authors, tailoring their loss function to the extremes is a key contribution of the manuscript (L411). Recent studies have also shown the benefits of adapting the loss function to predict extremes, using either density-based weights (Hermans et al., 2025) or quantile loss (Longo et al., 2025). As the MADc2 loss function is a similar intervention, the better performance obtained is not super surprising. It would therefore be very helpful if the authors could also test other loss functions that address the data imbalance, so that in addition to confirming previous

findings, they could also draw conclusions about which loss functions are most effective in their case.

We thank the reviewer for this comment. In the revised manuscript, we clarify the developmental timeline of the MADc^2 loss function, adding the dedicated section 2.5 Loss functions, which contains the following (L253-313): *As loss functions, both the Mean Squared Error (MSE) and MADc^2 were applied across all models. The MADc^2 loss function is derived from the corrected mean absolute deviation (MADc), introduced in Campos-Caba et al. (2024). MADc combines the standard Mean Absolute Deviation (MAD) with a percentile-based component (MADp), which measures the average absolute difference between simulated and observed values across percentiles of the observed distribution, thereby quantifying how well the empirical distribution is reproduced. MADc is defined as the sum of MAD and MADp, capturing both overall magnitude errors and discrepancies in distributional shape. MADc^2 is then defined as the square of MADc. This formulation is preferred over MADc as a loss function because it is differentiable at the theoretical minimum (unlike MADc, which is not) enabling smoother convergence in gradient-based optimization. Formally, MADc^2 is defined as:*

$$MADc^2 = \left(\overline{|S - O|} + MADp\right)^2 \qquad (1)$$

where S represents the simulation value and O the observation value. The full formulation of MADp, MADc, and MADc² as performance metrics is provided in Section 2.7.

The percentile component of the MADc^2 loss was computed using a non-uniform set of percentiles, with increased sampling density near the tails of the distribution in order to enhance sensitivity to extreme events. Specifically, additional percentiles were introduced in the upper and lower tails (e.g., 0.95, 0.98, 0.99, 0.995, and 0.999), while maintaining coarser sampling across the central portion of the distribution. This weighting strategy was designed to place greater emphasis on accurately reproducing rare surge levels without neglecting the overall distributional structure.

For recurrent architectures trained using the TBPTT approach, the percentile-based MADp component was not estimated independently within each chunk. Instead, a rolling buffer strategy was adopted, whereby the percentile distributions were computed using both the current chunk and a detached history of predictions and targets from previous chunks within the same epoch. This approach provides a more representative estimate of the global distribution while preserving the computational efficiency and memory constraints of TBPTT. Several buffer lengths were tested, and a size of 100000 samples provided the best compromise between training stability and extreme-event representation.

Building on this definition, the adoption of MADc^2 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. MADc^2 is preferred over MADc because it is differentiable at the theoretical minimum (MADc^2 = 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.

Recent studies have explored alternative strategies to address data imbalance and improve the representation of extremes in data-driven storm surge modelling, including density-based weighting schemes (Hermans et al., 2025) and quantile-based loss functions (Longo et al., 2025). These approaches share the common objective of increasing sensitivity to the upper tail of the distribution, albeit through different formulations.

In this study, we focus on MADc² as a representative and conceptually simple loss function that directly targets both overall error magnitude and distributional consistency through its combined MAD and percentile-based components. A systematic intercomparison of multiple extreme-aware loss functions is beyond the scope of the present work and would require additional design choices (e.g., quantile selection or weighting strategies) that may strongly influence results. Nevertheless, the convergence of findings across recent studies suggests that explicitly incorporating sensitivity to extremes into the training objective, rather than the specific mathematical form of the loss, is the key factor in improving emulator performance for storm surge extremes.

Additionally, we expanded the Discussion (Section 4, L688-699) to position our contribution within the recent literature on extreme-aware loss functions, including density-weighted losses (Hermans et al., 2025) and quantile-based losses (Longo et al., 2025), adding the following: *Recent advances in data-driven storm surge modeling have increasingly emphasized the importance of tailoring the training objective to the representation of extremes. For example,*

Hermans et al. (2025) demonstrated that density-based weighting schemes can substantially improve the prediction of high-percentile surges, while (Longo et al., 2025) explored quantile-based loss functions to explicitly target the upper tail of the distribution. Although these approaches differ in formulation, they share a common principle: standard loss functions such as MSE tend to underweight rare but impactful events, leading to systematic underestimation of extremes. The results presented here are fully consistent with this emerging body of work and provide complementary evidence that explicitly embedding sensitivity to extremes within the loss function is a key driver of improved performance. In this context, MADc² offers a conceptually simple and physically interpretable alternative, directly linking pointwise errors and distributional discrepancies, while avoiding the need for additional design choices such as quantile selection or density weighting. Our findings therefore reinforce the growing consensus that the choice of loss function is at least as important as model architecture for extreme-event emulation, and show that even simple models can achieve state-of-the-art performance when trained with objectives explicitly designed for extremes.

We emphasize that MADc² was developed independently and prior to the publication of Hermans et al. (2025), and that the convergence of findings across studies supports the broader conclusion that explicitly incorporating sensitivity to extremes in the training objective is a key driver of improved performance.

A systematic intercomparison of alternative loss functions is acknowledged as a valuable future research direction but is beyond the scope of the present study.

2. The authors find that their multi-linear regression model performs similarly to some of their neural networks. However, due to several methodological aspects is not fully clear whether the neural networks were appropriately optimized to support this conclusion.

We appreciate this comment. We have expanded the following section to provide clearer descriptions of the hyperparameter exploration:

- Sections 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: *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.*

We further added the following caveat in the Conclusions (Section 5, L827-833) noting that our findings regarding the relative importance of loss function versus model architecture are conditional on the PCA-based predictor representation and the spatial scale considered: *At the same time, we note that this conclusion is conditional on the spatial scale and predictor representation adopted in this study. The use of PCA-based dimensionality reduction limits the explicit exploitation of spatial structures in the predictors. Previous studies have shown that convolutional layers can provide substantial benefits when larger spatial predictor domains are considered and when spatial patterns play a dominant role in surge generation (e.g., Tiggeloven et al., 2021; Hermans et al., 2025). Therefore, while our results demonstrate that loss-function design can outweigh architectural complexity, future work could explore more complex architectures in combination with extreme-aware loss functions, particularly when larger spatial domains or richer spatial representations are considered.*

2a. The authors applied PCA to reduce the dimensions of the input data and did not consider convolutional layers in their architectures. Tiggeloven et al. (2021) and Hermans et al. (2025), however, found that for larger prediction regions, convolutional layers are beneficial for the prediction of (extreme) storm surges. Adding a clear caveat to the conclusions in L519- 524 to reflect on this would be helpful.

The following caveat has been added to the Conclusions (Section 5, L827-833) explicitly acknowledging that convolutional layers have been shown to improve performance when larger spatial predictor domains are considered (Tiggeloven et al., 2021; Hermans et al., 2025): *At the same time, we note that this conclusion is conditional on the spatial scale and predictor representation adopted in this study. The use of PCA-based dimensionality reduction limits the explicit exploitation of spatial structures in the predictors. Previous studies have shown that convolutional layers can provide substantial benefits when larger spatial predictor domains are considered and when spatial patterns play a dominant role in surge generation (e.g., Tiggeloven et al., 2021; Hermans et al., 2025). Therefore, while our results demonstrate that loss-function design can outweigh architectural complexity, future work could explore more complex architectures in combination with extreme-aware loss functions, particularly when larger spatial domains or richer spatial representations are considered.*

2b. The authors did not specify the number of preceding timesteps at which predictor data was used for the LSTMs. If no look-back window was used, this would likely lead to a worse (relative) performance.

We have clarified in Section 2.4 that temporal dependencies in recurrent emulators (RNN and LSTM) are handled internally through the recurrent cell state and gating mechanisms, rather than through an explicit look-back window. Specifically, the following can be found on L190-204 of the revised manuscript*: 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 capabilities of the recurrent architectures were properly exploited, we explored the sliding-window (lookback window) approach. A description of the implementation and the main results is provided in Section 2.4, 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.

2c. The authors optimized several of their input settings for all model architectures based on tests with only the MLR model (L104-108). Could the authors motivate whether these settings are also expected to work best for their other architectures, and if not, reflect on this in the discussion?

We now clarify in Section 2.3 that MLR was used as a computationally efficient and interpretable baseline for selecting the predictor domain and number of retained principal components. The following can be found on L150-153 of the revised version of the manuscript: *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.*

Additionally, In Section 4 (L635-645), we explicitly acknowledge that other architectures may benefit from alternative configurations and that architecture-specific optimization represents a natural extension of this work: A*nother important consideration in this study concerns the use of PCA for reducing the spatial dimensionality of the predictors. While PCA is effective in retaining the components that explain most of the variance, it applies a linear transformation to the input data, which may affect the emulator's ability to capture nonlinear patterns or interactions that are potentially relevant for storm surge dynamics. For this process, the selection of the predictor spatial domain and the number of retained principal components was guided by performance tests conducted with the MLR emulator. This choice was motivated by the computational efficiency and transparency of linear models, which enable rapid exploration of multiple configurations. While this approach provides a consistent and physically interpretable baseline across all architectures, it does not guarantee that the selected domain size or number of PCs is optimal for more complex nonlinear models. In particular, architectures such as LSTMs or hybrid networks may, in principle, benefit from alternative dimensionality-reduction strategies or from retaining a larger number of components. Exploring architecture-specific optimization of the predictor representation represents a natural extension of this work and will be addressed in future studies.*

2d. Hyperparameter tuning is mentioned only briefly. Could the authors add additional detail on what parameters they used (not just the depth of the network, but also the other learning parameters) and to what extent these parameters were optimized? These are important details to place their results into context.

We thank 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 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 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. The authors used sea surface height simulations from a high-resolution ocean reanalysis (Med-MFC) as a predictor variable for their data-driven models. I do not understand the rationale for this: if used in a forecasting or longer-term prediction setting, this means that to obtain the performance reported by the authors, a high-resolution hydrodynamic model will be necessary as well, which defeats the purpose of the data-driven models. Furthermore, unless I misunderstand, they use part of what they want to predict (surge) as input (sea surface height), meaning that performance of the data-driven models will be enhanced by construction. Could the authors please explain their methods in this regard and discuss how this affects their conclusions in L465-472? How would the data-driven models perform using only atmospheric data as predictors, as previous studies did?

We thank the reviewer for this important comment. We have expanded Section 2.3 regarding the use of sea surface height from Med-MFC as predictor. The following can be found in the revised manuscript (L140-148): *The inclusion of basin-scale sea surface height (SSH) from the Med-MFC reanalysis reflects a deliberate choice aligned with a statistical downscaling framework, rather than an attempt to replace physics-based models. In coastal downscaling, large-scale ocean models routinely provide SSH boundary conditions that are subsequently refined by higher-resolution regional or coastal models. In this study, the ML emulators are designed to perform an analogous task: they take available basin-scale SSH fields together with atmospheric predictors and statistically refine them to tide gauge-scale storm surge signals. In this sense, the ML emulators function as data-driven coastal correctors of basin-scale output rather than standalone substitutes for ocean circulation models. This approach differs from purely atmospheric-driven time-series forecasting and instead mirrors established dynamical downscaling chains, focusing ML capacity on resolving coastal-scale processes that coarse models cannot explicitly capture.*

We also added a dedicated discussion (Section 4, L612–620) to clarify that our approach is framed as a statistical downscaling paradigm rather than a standalone forecasting system. In this context, basin-scale sea surface height (SSH) from Med-MFC is used as a large-scale predictor that is subsequently refined to the local coastal response, in analogy with traditional dynamical downscaling chains. We explicitly discuss both the advantages of this approach and its dependency on the availability and quality of the parent ocean model: *An important methodological choice in this study is the use of basin-scale sea surface height from Med-MFC as a predictor for the ML emulators. While several data-driven storm surge studies rely exclusively on atmospheric forcing, our approach is intentionally framed as statistical downscaling rather than standalone time-series downscaling. By leveraging basin-scale SSH, freely and consistently available through Copernicus services, the emulator refines the large-scale ocean signal to the local coastal response, in direct analogy with the role played by high-resolution dynamical models in traditional downscaling chains. This design choice is not*

circular, as the emulator does not attempt to reproduce the basin-scale solution, but instead learns the systematic transformations required to resolve local surge dynamics that are unresolved at coarse resolution. At the same time, this choice implies that emulator performance is conditional on the availability and quality of the parent ocean model, a limitation that we explicitly acknowledge.

In response to the reviewer's suggestion, we have additionally conducted new experiments using only atmospheric predictors (mean sea level pressure and wind stress components), excluding basin-scale SSH and tide. These results are now presented and discussed in Section 4, L622-633: *To further assess the operational independence of the proposed framework, additional experiments were conducted using only atmospheric predictors (mean sea level pressure and wind stress components), excluding basin-scale SSH and tide, while maintaining the same configurations described in Sections 2.4 and 2.6. These atmospheric-only configurations represent fully standalone forecasting-style emulators. As expected, overall performance metrics degrade relative to SSH-informed models, reflecting the loss of explicit basin-scale ocean state information. Pearson correlation and slope values decrease moderately, and error metrics increase across both locations (Figure S9). However, the degradation is not drastic, and the atmospheric-only emulators retain substantial skill in reproducing surge variability. Notably, the percentile structure and the scaling of higher surge values remain reasonably well captured, indicating that atmospheric forcing alone contains significant predictive information for extreme events. These results suggest that while basin-scale SSH provides valuable additional skill, particularly for overall variance reproduction, purely atmospheric-driven emulators remain viable alternatives in contexts where ocean model output is unavailable. The comparison highlights a clear trade-off between physical completeness and operational independence.*

As expected, the performance of the atmospheric-only approach decreases relative to SSH-informed emulators, particularly in terms of overall variance reproduction. However, the degradation is moderate, and the atmospheric-only emulators retain substantial skill in capturing surge variability and extremes. This demonstrates that atmospheric forcing alone contains significant predictive information, while also highlighting a clear trade-off between physical completeness (SSH-informed models) and operational independence (atmospheric-only models).

These additional analyses strengthen the interpretation of our results and clarify the role of SSH within the proposed framework.

4. L540: The authors did not make their code and data available, so their results are currently irreproducible and cannot be reviewed. This needs to be published in an appropriate repository.

A Data availability section has been added (Section 6), stating that all data and code supporting the results will be made publicly available in an appropriate repository to ensure full reproducibility.

Other comments

L35: Calafat et al. (2022) did not attribute the trends they found to climate change, so I suggest using a different reference here.

The sentence has been revised in the Introduction (Section 1) to avoid explicit attribution to climate change and to include additional references (IPCC, 2021) where appropriate. The following now can be found in the revised manuscript (L35-36): *Accurate storm surge prediction is vital for coastal risk management, particularly in light of observed increases in extreme sea-level events and projected future changes associated with sea-level rise and climate change (Calafat et al., 2022; IPCC, 2021).*

L40: After 'widely' I would expect a few more examples, such as Tadesse et al. (2020) and references therein.

Additional references, including Tadesse et al. (2020), have been added in Section 1 to support the statement. The following now can be found in the revised manuscript (L59-60): *Regression-based ML emulators, such as Multivariate Linear Regression (MLR), have been widely used to predict storm surge levels (Cid et al., 2018; Mi Zhang et al., 2019; Tadesse et al., 2020; Harter et al., 2024).*

L48-49: Yes, but they also showed that convolutional layers did help to increase performance when using larger predictor regions. Please also see Hermans et al. (2025).

Paragraph was edited to acknowledge the benefits of convolutional layers on LSTM architecture. The reference of Hermans et al. (2025) was also added to support the statement. The following now can be found in the revised manuscript (L67-73): *Tiggeloven et al. (2021) tested CNN-LSTM hybrid emulators but ultimately found that standard LSTM models outperformed more complex architectures. However, they also showed that convolutional layers can improve performance when larger spatial predictor domains are considered, a result further supported by more recent studies (e.g., Hermans et al., 2025). Further enhancements include Gated Recurrent Units (GRUs) with physics-informed loss functions (Feng & Xu, 2024) and transformer-based models (Rus et al., 2023b), which extend emulator capabilities in capturing both spatial and temporal surge patterns. Ensemble techniques and bias correction approaches have also contributed to improved predictive skill and interpretability (Giaremis et al., 2024; Sun & Pan, 2023).*

L61: As only two locations are considered, point (2) is not really addressed in this manuscript either.

We now acknowledge this limitation, adding the following on L90-91: *While the analysis is limited to two locations, it provides insight into site-dependent performance and contributes to the broader understanding of emulator behavior across coastal environments.*

Additionally, we added the following on Section 4 (Discussion), L603-605: *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*.

L104: Please consider showing the spatial domain of predictors in Figure 1.

Figure 1 has been updated to explicitly show the spatial domain of the predictors, derived directly from the predictor grid used in the ML emulators.

Section 2.4: Details about the architecture, number of layers, units, other hyperparameters (such as dropout rate, learning rate, etc.) and their optimization are absent -> please provide for reproducibility.

As replied on comment 2d, we have further extended the hyperparameter tuning of the architectures to ensure reproducibility.

L163: The test split only covers 3 years -> are the extremes during this period representative of the overall distribution? Could they be easier or more difficult to predict than extremes in other 3-year periods by chance? Did the authors test different splits or split ratios?

Section 2.6 has been expanded and Supplementary Figure S8 is now explicitly discussed to demonstrate that the testing extremes are representative of the overall distribution. The limitation of a fixed split is acknowledged. The following can now be found in the revised manuscript (L338-424): *Following the approach of Gholamy et al. (2018), 80% of the available data (28 years) was allocated for training, while the remaining 20% (6 years) was evenly split into validation and testing sets (3 years each). The specific years assigned to each set were as follows: training: 1987–1992 and 1997–2018; validation: 1993, 1994, and 2019; testing: 1995, 1996, and 2020. This temporal partitioning was designed to ensure that the distributions of observed storm surge values were comparable across sets, particularly between training and testing, in line with the recommendations of Uçar et al. (2020).*

Figure S8 illustrates this comparison: panels (a) and (c) show that the training sets for Punta della Salute and Trieste include the highest observed percentiles, ensuring that the models are exposed to extreme events during training. Panels (b) and (d) demonstrate that the testing sets

also sample the upper tail of the distribution, with percentile–percentile relationships closely aligned with those of the training and validation sets. This indicates that the extreme values present in the testing period are representative of the broader distribution and are neither systematically easier nor more difficult than those observed in the remaining record. While the testing period is necessarily limited to three years, the explicit comparison of percentile distributions across subsets provides confidence that the selected split offers a robust and unbiased evaluation of model performance on extreme storm surge events.

L173-174: Please explain how these settings were determined.

Additional information is provided on the following L294-304 of the revised manuscript: *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.*

L181: Why specifically was the 99th percentile used? The results of Harter et al. (2024) and Hermans et al. (2025) suggest that for even more extreme values, the performance of data-driven models falls off compared to a high-resolution hydrodynamic model. It would be interesting to check whether this is also the case here.

We have clarified in the manuscript why the 99th percentile was selected, emphasizing its balance between extremeness and statistical robustness. We also acknowledge that model performance at higher percentiles may differ, and we discuss the upper-tail behavior qualitatively elsewhere in the manuscript. Section 2.5 (L309-318) now clarifies this choice, with the following: *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)*.

We also acknowledge the emulator performance at higher percentiles may differ and discuss upper-tail behavior qualitatively in Section 4, L735-753: *MADc^2 training improves the ability of the emulators to reproduce the upper tail of the observed surge distribution, particularly at the highest percentiles (Figure 12). In Trieste, several MSE-trained emulators underestimate the most extreme surge levels (Figure 12a), whereas the MADc^2-trained emulators generally provide a closer agreement with the empirical distribution of the tide gauge data (Figure 12b). At Punta della Salute, the differences between MSE- and MADc^2-trained emulators are more moderate above the 99th percentile (Figures 12c–d), with both groups of models capturing the general distributional behaviour. However, when focusing on the most extreme events beyond the 99.9th percentile (Figures 12e–f), the advantages of MADc^2 training become more evident.*

Several MADc²-trained emulators reduce the underestimation observed in the MSE-based models and provide a closer representation of the most extreme surge levels. Nevertheless, some MADc²-trained emulators also exhibit a tendency to overestimate the highest surge values, particularly between the 99.9th and 99.98^{t}h percentiles.

Overall, the results indicate that MADc²-trained emulators match or even surpass the performance of SHYFEM-MPI, a high-resolution dynamical model specifically developed for storm surge prediction in the region, when the most extreme surges are considered. For instance, the distributions of bias, MADp, and MADc for MLR-MADc² for surge peaks above the 99th percentile shown in Figure 3 indicate that a considerable portion of the 40 runs performed of this emulator outperforms SHYFEM-MPI across the mentioned metrics. Similarly, most of the distributions for LSTMh-MADc² demonstrate better performance than the SHYFEM-MPI benchmark for the same evaluation criteria, including the selected runs based on the adopted validation-based model selection strategy. These results confirm that ML emulators, when trained with appropriate loss functions, can serve as efficient and accurate alternatives to physics-based models, particularly in ensemble forecasting or early warning applications.

Section 4 (Discussion): comparison with and discussion in the context of previous literature on data-driven modeling of storm surges (the ones mentioned above and several other relevant studies) is largely missing; I would encourage the authors to describe how their results fit into the bigger picture and help advance the field.

We have extended the Discussion (Section 4) of the revised manuscript describing how the implementation of the MADc² loss function fits into recent advances of ML emulators optimized for extreme events. The following can now be found in the revised manuscript (L688-699: *Recent advances in data-driven storm surge modeling have increasingly emphasized the importance of tailoring the training objective to the representation of extremes. For example, Hermans et al. (2025) demonstrated that density-based weighting schemes can substantially improve the prediction of high-percentile surges, while (Longo et al., 2025) explored quantile-based loss functions to explicitly target the upper tail of the distribution. Although these approaches differ in formulation, they share a common principle: standard loss functions such as MSE tend to underweight rare but impactful events, leading to systematic underestimation of extremes. The results presented here are fully consistent with this emerging body of work and provide complementary evidence that explicitly embedding sensitivity to extremes within the loss function is a key driver of improved performance. In this context, MADc² offers a conceptually simple and physically interpretable alternative, directly linking pointwise errors and distributional discrepancies, while avoiding the need for additional design choices such as quantile selection or density weighting. Our findings therefore reinforce the growing consensus that the choice of loss function is at least as important as model architecture for extreme-event emulation, and show that even simple models can achieve state-of-the-art performance when trained with objectives explicitly designed for extremes.*

General: please consider using perceptually uniform colormaps.

All figures have been updated to replace rainbow colormaps with perceptually uniform alternatives (e.g., viridis). Specifically, the following figures were edited: Fig. 7, Fig. 8, Fig. 10, Fig. 11, and Fig. S9.