Neural Computing and Applications
https://doi.org/10.1007/s00521-023-08719-2

ORIGINAL ARTICLE

An advanced spatio-temporal convolutional recurrent neural network for storm surge predictions

Ehsan Adeli[1] · Luning Sun[2] · Jianxun Wang[2] · Alexandros A. Taflanidis[1]

Received: 28 December 2022 / Accepted: 31 May 2023

Abstract
In this research paper, we study the capability of artificial neural network models to emulate storm surge based on the storm track/size/intensity history, leveraging a database of synthetic storm simulations. Traditionally, computational fluid dynamics (CFD) solvers are employed to numerically solve the storm surge governing equations that correspond to expensive to evaluate partial differential equations (PDE). This study presents a neural network model that can predict storm surge, informed by a database of synthetic storm simulations. This model can serve as a fast and affordable emulator for the expensive CFD solvers creating the original database. The neural network model is trained with the storm track parameters used to drive the CFD solvers, and the output of the model is the time-series evolution of the predicted storm surge across multiple nodes within the spatial domain of interest. Once the model is trained, it can be deployed for further predictions based on new storm track inputs. The developed neural network model is a time-series model, composed of a long short-term memory (LSTM), a variation of recurrent neural network (RNN), further enriched with convolutional neural networks (CNNs). The convolutional neural network is employed to capture the correlation of data spatially (across the aforementioned nodes). Therefore, the temporal and spatial correlations of data are captured by the combination of the mentioned models, representing the ConvLSTM model. As the problem is a sequence to sequence time-series problem, an encoder–decoder ConvLSTM model is designed. Furthermore, the performance of the developed convolutional recurrent neural network model is improved by residual connection networks. Additional techniques are employed in the process of model training to enrich the model performance that the model can learn from the data in a more effective way. The performance of the developed model is compared with the results provided by a Gaussian process (GP) implementation, representing a state-of-the-art alternative for establishing time-series emulation of storm surge predictions. The results show that the proposed convolutional recurrent neural network outperforms the GP implementation for the examined synthetic storm database.

Keywords Advanced neural networks · Storm surge prediction · Recurrent neural networks · Convolutional neural networks

1 Introduction

Predicting future storm surge-related impact is receiving growing attention within the global scientific community, recognizing the widespread socio-economic implications of this natural hazard that need to be addressed within diverse prevention, mitigation, and post-disaster settings [1]. Efforts to provide enhanced decision support against these imminent dangers over the past couple of decades have focused, among other topics, on numerical advances for storm surge predictions, producing high-fidelity simulation models that permit a detailed representation of

✉ Ehsan Adeli
eadeli@nd.edu

[1] Department of Civil & Environmental Engineering & Earth Sciences, University of Notre Dame, Notre Dame, IN 46556, USA

[2] Computational Mechanics & Scientific AI Lab, Department of Aerospace and Mechanical Engineering, University of Notre Dame, Notre Dame, IN 46556, USA

Published online: 15 June 2023

hydrodynamic processes and therefore support high-accuracy forecasting. One such computational fluid dynamics (CFD) solver, utilized later in this paper, is ADCIRC [2], which is widely used [3] to simulate with high accuracy tidal circulation and storm surge propagation over large computational domains, and is, furthermore, typically coupled with appropriate models like SWAN [4] or STWAVE [5] to additionally incorporate wave action within the predictions. Unfortunately, the computational burden of such numerical models is large, requiring thousands of CPU hours for each simulation, something that limits their applicability for real-time surge forecasting (during landfalling events) or regional probabilistic flood studies. Due to this computational complexity, such models can be utilized to provide only a small number of high-fidelity, deterministic predictions, but cannot easily accommodate thousand-run storm ensembles, for example for examining the impact of forecast errors [6] in the predicted track during landfalling events. This dramatically limits their utility for decision makers either in emergency response management (during landfalling events) or regional planning (long-term projection of storm impact) settings.

To address these computational challenges associated with high-fidelity solvers, and offer an alternative approach for probabilistic storm forecasting and risk assessment applications, machine learning tools and surrogate models have attracted significant attention [7–15] for storm surge emulation. Based on databases of synthetic storm simulations, these approaches can provide fast-to-compute, data-driven approximations for the expected storm surge. They are capable of replacing, with a high level of accuracy, the high-fidelity numerical model used that created the original database, maintaining the detailed underlying representation of hydrodynamic processes [16], while offering substantial computational efficiency. The latter efficiency makes them highly appropriate for supporting probabilistic surge forecasting and coastal hazard estimation applications. As such, they can be leveraged to offer enhanced decision support for emergency response managers and regional planners [17, 18].

Among the different machine learning techniques that could be considered for this application, artificial neural networks has shown great promise [9, 11, 20–23]. This study extends past efforts in this domain by considering a neural network implementation for predicting the entire time-series evolution of the storm surge using as input the time-series evolution of the storm track (latitude and longitude of eye of storm), intensity (pressure at center of storm) and size (radius of maximum winds). Past studies have focused on prediction of peak surge only (as opposed to the time evolution of the surge) and/or used instantaneous characteristics of the storm features as inputs for establishing the machine learning predictions. Should be pointed out that focus on prediction of peak surge is common in most studies that have examined storm surge emulations, with very few establishing predictions for the entire evolution of the storm surge. This study considers simultaneously time-series properties for both the surge predictions as well as the storm feature evolution, addressing, additionally, the spatial character of the predictions. To accommodate this substantial extension, a time-series recurrent neural network model (RNN) is developed to predict the storm's behavior. The spatial correlation of data, i.e., the fact that the storm surge is estimated across multiple locations within the geographic domain of storm impact, is additionally considered by applying convolutional neural networks (CNNs). Ultimately, this allows both spatial and temporal correlations of data to be comprehensively captured by using a convolutional recurrent neural network model. The model input parameters to predict the storm surge are time series for the storm track, size and intensity, while the model output is the time series of the storm surge level for specified locations along the coast.

Mathematically speaking, the typical neural network model maps the input parameters (layer) $z_0 \in \mathbb{R}^{n_0}$ (the aforementioned four input parameters for our application) to the output $z_L \in \mathbb{R}^{n_L}$ (surge values across different nodes in the domain in our application). The layers between the input and output layers are the hidden layers z_l, where $l = 1, ..., L$. Two adjacent layers are connected through the formulation below.

$$z_l = \mathcal{F}(\boldsymbol{W}_l^T z_{l-1} + \boldsymbol{b}_l) \tag{1}$$

In Eq. 1, $\boldsymbol{W}$ and $\boldsymbol{b}$ represents the model parameters, weight matrix and bias vector, respectively, and $\mathcal{F}$ denotes the activation function. After the model is trained, the model parameters are determined, and the output surge prediction can be rapidly computed from the given input parameters. This forward computation that involves only matrix multiplications has negligible computational burden compared to the original high-fidelity, CFD simulation. The model's performance is improved through developing the applied neural network models. Additionally, various other techniques to improve the model's efficiency are considered in the training process. The results are compared to the results computed by a Gaussian process implementation [10], which represents a state-of-the art alternative emulation technique for predicting the time-series evolution of the storm surge.

The main novelty of this manuscript from the storm surge application perspective is that it establishes neural network-based spatio-temporal predictions across a large geographic domain. Previous studies, as presented earlier, have either focused on peak surge predictions (not addressing time evolution of the surge), or have considered time-series surge predictions for a moderate only number of spatial nodes. For artificial neural network applications, this has allowed these past previous studies to consider independent formulations across the different nodes, with no requirement on the trained network to describe additionally the spatial variability of the surge. Additional novelties from the NN perspective include the development and training a time-series neural network model which considers the spatial and temporal correlation of the data established through a unique combination of different models, cells, and layers to address unique challenges (detailed in Sect. 3) of the spatio-temporal storm surge emulation problem.

In Sect. 2, we discuss the problem formulation and the synthetic simulation data for training and testing of the machine learning models. Section 3 describes the machine learning methods and how the models are trained with the provided data. Then, results and comparison discussions are provided in Sect. 4. Finally, the conclusions are given in Sect. 5.

2 Storm surge prediction problem characteristics

The devastating flooding effects of numerous storms in the past two decades, such as hurricane Katrina and superstorm Sandy, have incentivized researchers to establish high-accuracy models to predict storm surge impact on coastal regions. These efforts have produced numerous advanced numerical CFD solvers [2, 3, 16, 24] used by various actors for emergency response management or regional planning. These solvers simulate the storm surge by solving the shallow water wave equations given the initial and boundary conditions. The simulation is driven by the atmospheric pressure and wind velocity that describes the time evolution of the hurricane vortex. This wind and velocity input can be derived through information for the storm track (location of center of rotation and forward speed of the vortex), size and intensity [25, 26], with intensity described by the wind speed or the pressure loss between the center and the far-away ambient conditions, and size described by the distance between the center and the location of maximum wind speeds. Interested readers can found additional information for hurricane physics and modeling in [27]. These numerical tools can be ultimately used to accommodate deterministic and probabilistic approaches for establishing storm surge predictions [6, 28–32].

As discussed in the introduction, the aforementioned models provide high-accuracy estimates (empowered by high-resolution spatial grids), but entail a very large computational cost that posed a great challenge for their widespread use, especially in the context of probabilistic assessments for real-time forecasting applications. To overcome this challenge, machine learning techniques can be developed that leverage precomputed datasets of synthetic hurricane simulations, providing information for storm parameters, paths and surge responses. Within this setting, the unknown functional relationship between inputs (hurricane parameters) and responses (storm surge) can be approximated by some type of regression, response surface or non-parametric emulation model. Specifically, this study focuses on artificial neural network (ANNs) implementation. Substantial research efforts have already been made to consider ANN applications within storm surge emulation setting.

Lee et al. [33–35] conducted research on shallow networks with a limited number of neurons to predict the storm surge for a few typhoons impacting Taiwan. A similar study with almost a similar size of networks has been carried out by De Oliveira et al. [36] for the southeast coastal region of Brazil. Another study to reduce the uncertainty of storm surge prediction for Venice, Italy, is conducted by Bajo et al. [37], again using shallow neural networks. It should be pointed out that in the mentioned studies, the neural network models are trained with very few storms, limiting predictive potential of the network and ability to establish in-depth learning from the data. To improve the model's performance, Kim et al. [9] has used a bigger set of data established by using the ADCIRC model for the New Orleans region, and trained a shallow network which is tested on the historical hurricane Katrina. Note that the model they applied was not a time-series neural network model, and therefore, the temporal correlation of data was not properly leveraged within the model development. Several other similar studies have been carried out by Hashemi et al. [38], Kim et al. [39], Chao et al. [40] and Das et al. [41] for other geographical regions, using larger training datasets (with larger number of storms) to train neural network models. However, these efforts did not, once again, consider the temporal correlation of the storm data.

More recently, a number of studies have employed time-series models to predict storm surge based on the time evolution of the storm input parameters, in all cases

utilizing a small number of storm simulations. Alemany et al. [42] has employed a recurrent neural network (RNN) to predict the storm surge when it gets close to the beach based on the very initial part of the surge. Igarashi et al. [43] has also employed a standard recurrent neural network by utilizing a database of about 150 storms to estimate the surge for future storm events. Furthermore, Chen et al. [44] have applied a standard modification of the time-series model called long short-term memory (LSTM) model and trained it with a database of twelve storms.

In most of the aforementioned studies, the models are trained with a limited number of storm observations. Also in all these studies, the number of grid points for which the surge is predicted is relatively small. This is accomplished either by examining a small geographic region only, or by establishing some type of clustering approach, to reduce the original grid to a smaller number of representative points. Moreover, the standard sequence neural network models are mostly used as a black box in these studies, and no development and further investigation is applied to the standard time-series models, to accommodate some of the unique features of the storm surge emulation problem.

This study, extends these past efforts and considers a neural network implementation for predicting the time-series evolution of the storm surge across a geographic domain including a large number of save points (SPs), utilizing a database with a large number of storm surge simulations. The database is part of the Army Corps of Engineers Coastal Hazard System [18] and corresponds to synthetic storms simulations for the greater Coastal Texas region with a total of 4800 SPs, also shown in Fig. 1. The database was developed for a regional flood study and consists of storms selected based on a variation of the joint-probability-method optimal sampling (JPM-OS), to populate the input domain of plausible future storms. JPM-OS [REF] resembles a Bayesian quadrature numerical scheme for selecting storm samples, and, therefore, yields datasets that deviate from traditional space-filling sampling schemes (prioritize coverage of probability space). The high-fidelity numerical model utilized for predicting storm surge for creating the database was ADCIRC [19]. Five hundred storms will be used for calibration of the neural network emulator, and an additional eight storms will be used as test sample for its validation. The input for the synthetic storm simulations corresponds to: the latitude and longitude of the storm center (storm track parameters), the central pressure deficit (storm intensity parameter) and the radius of max winds (storm size parameter). The time evolution for all these four parameters is utilized as input to the neural network. Note that some recent studies have considered some additional, derived parameters for describing the neural network input, namely the forward speed and the track heading [20], but these correspond to redundant storm characteristics if time evolution of the storm features is examined (instead of instantaneous features) and contribute to an over-parameterization of the database. As such, the input is represented by only four storm parameters. The predicted output corresponds to the storm surge across the 4800 SPs. This creates a sequence-to-sequence prediction problem, with both the input and the output of the neural network corresponding to sequences. Such sequence prediction problems are widely acknowledged to be exceptionally challenging.

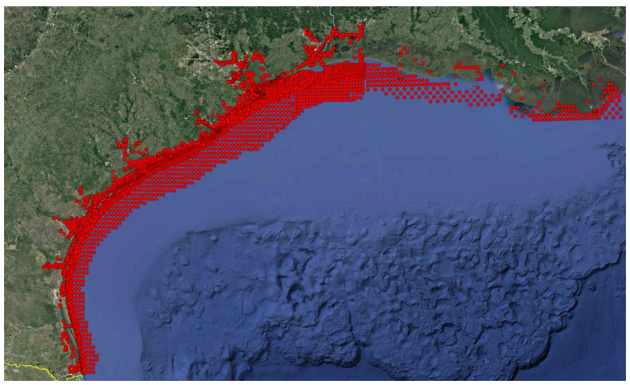

Fig. 1 Grid of save points within the region of study

For both the input and the output, 125 time steps are utilized, extending from the time each storm is a couple thousands of kilometers before making landfall, to a few hundred kilometers after making landfall. This range is chosen to encompass the time instances the maximum surge manifests across the entire geographic domain of interest. Note that the selection of range and time-stepping based on distance to landfall has been shown very recently to provide benefits for surge emulation applications [46]. Synchronization of the time series is established with respect to the landfall for each storm, as done in past studies [9]. This landfall corresponds roughly to step 90. Figures 2 and 3 show variation of the four input parameters for a typical storm and the variation of the surge for different nodes for the same storm, respectively. It is evident from this figure that the size and intensity of the synthetic storms remain practically unchanged before the storm makes landfall. This is common characteristic of many synthetic storm databases and creates some challenges for the neural network application as will be detailed later.

3 Neural network methods

3.1 Convolutional long short-term memory

Long short-term memory (LSTM) [47] is a class of recurrent neural networks (RNNs) [48], capable of

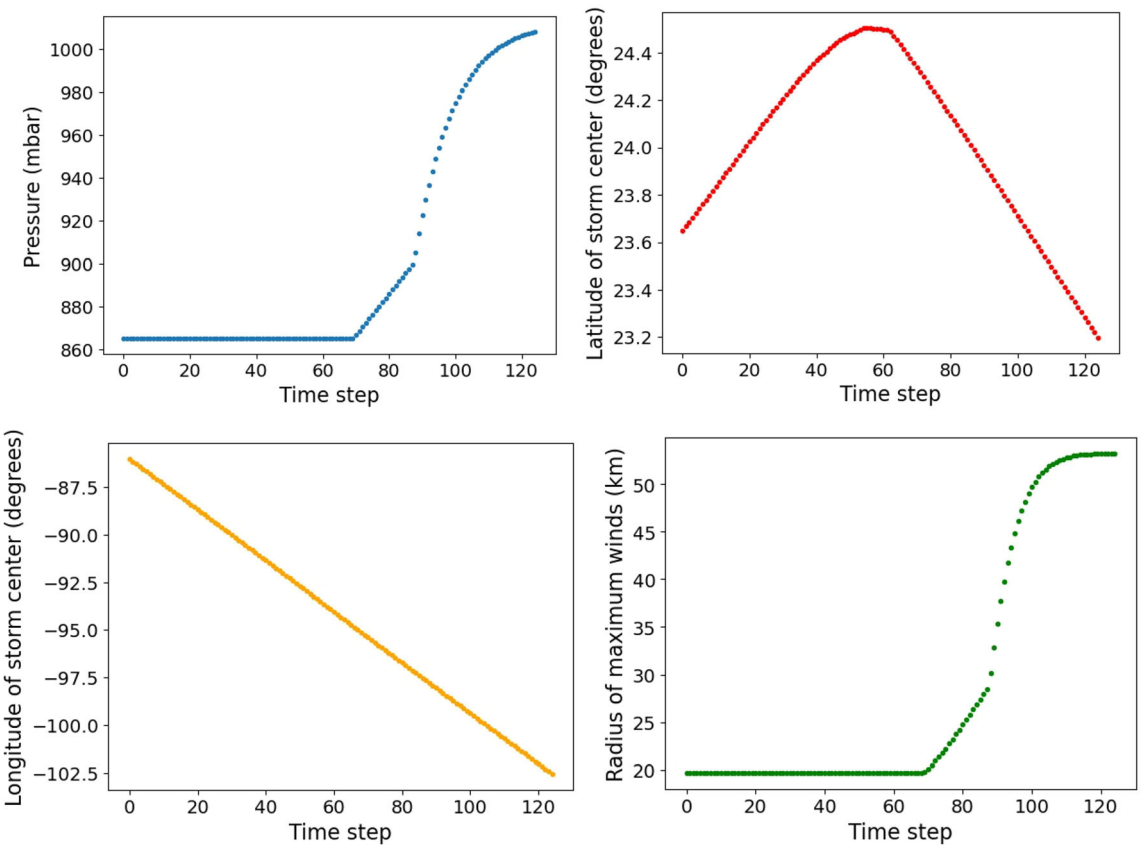

Fig. 2 Input parameters for a typical database storm

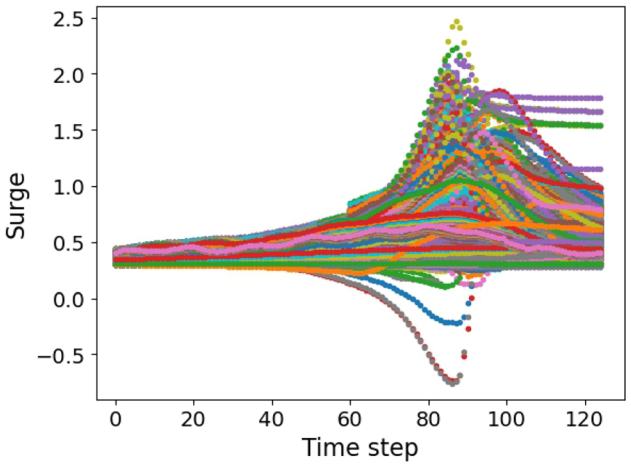

Fig. 3 Surge output for different SPs for a typical database storm

addressing the gradient vanishing problem for long-term temporal dependencies encountered in RNNs. LSTMs have a memory cell that can maintain information in memory for a long period of time, and also gates that allow for better control over the gradient flow by forgetting, updating, and outputting part of the needed information. These gates represent the essential component for enabling better preservation of long-term time dependencies. To consider a simultaneous spatial and temporal learning framework, an extension of LSTMs named ConvLSTMs [49] is employed, representing a convolutional neural networks (CNNs) extension of LSTMs. Specifically convolutional layers are employed instead of the fully connected NNs (dense layers) in gated operations because of their better representational capability of spatial connections. Thus, the applied ConvLSTM extended form of the long short-term memory (LSTM) represents a spatio-temporal formulation, specifically developed for the purpose of sequence-to-sequence learnings. Figure 4 demonstrates a typical graphic of ConvLSTMs.

In Fig. 4, $\boldsymbol{X}_{\mathrm{t}}$ stands for the input tensor. The hidden state and cell state are indicated by $\boldsymbol{h}_{\mathrm{t}}$ and $\boldsymbol{C}_{\mathrm{t}}$, respectively, and are updated at each time t. The ConvLSTM cell consists of four gate variables facilitating input-to-state transition and state-to-state transition. The forget gate and input gate are indicated with $\{\boldsymbol{f}_{\mathrm{t}}\}$ and $\{\boldsymbol{i}_{\mathrm{t}}\}$, respectively, at time t. The other two gates, an internal cell and an output gate, are also denoted with $\{\tilde{\boldsymbol{C}}_{\mathrm{t}}\}$ and $\{\boldsymbol{o}_{\mathrm{t}}\}$, respectively.

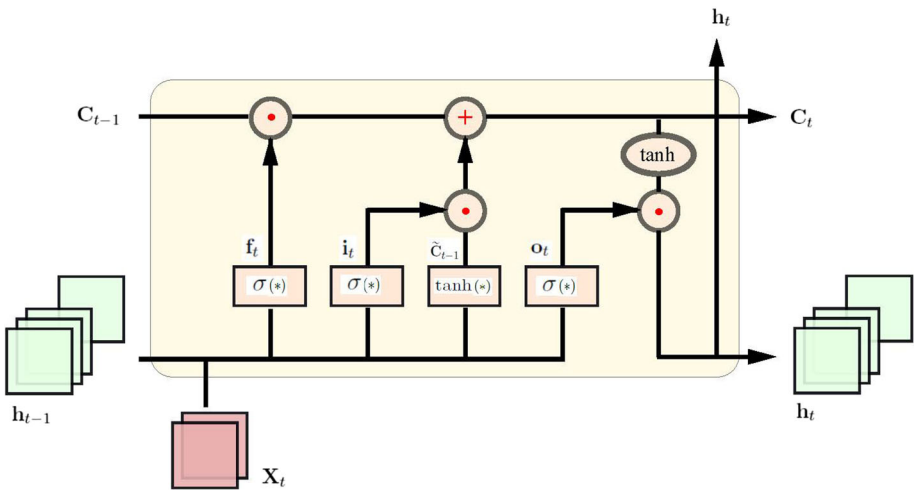

Fig. 4 Single ConvLSTM cell at time t

The sigmoid activation function $\sigma(\cdot)$ is used for the gates, leading to the mapping of the outputs to values between 0 and 1. Therefore, the forget gate layer adaptively clears the memory information in the cell state $\{\boldsymbol{C}_{t-1}\}$. The memory stored in the cell state originates from the cooperation between the input gate layer and the internal cell state, where the internal cell state is a new cell candidate created from the hyperbolic tangent activation layer (i.e., $\tanh(\cdot)$), and the input gate layer decides the information propagating into the cell state. Lastly, the output gate layer filters and regulates the cell state for the final output variable/hidden state. The updating ConvLSTM is governed by the mathematical formulations which are described in Eq. 2.

$$\boldsymbol{i}_t = \sigma(\boldsymbol{W}_i * [\boldsymbol{X}_t, \boldsymbol{h}_{t-1}] + \boldsymbol{b}_i) \tag{2a}$$

$$\boldsymbol{f}_t = \sigma(\boldsymbol{W}_f * [\boldsymbol{X}_t, \boldsymbol{h}_{t-1}] + \boldsymbol{b}_f) \tag{2b}$$

$$\tilde{\boldsymbol{C}}_{t-1} = \tanh(\boldsymbol{W}_c * [\boldsymbol{X}_t, \boldsymbol{h}_{t-1}] + \boldsymbol{b}_c) \tag{2c}$$

$$\boldsymbol{C}_{t-1} = \boldsymbol{f}_t \odot \boldsymbol{C}_{t-1} + \boldsymbol{i}_t \odot \tilde{\boldsymbol{C}}_{t-1} \tag{2d}$$

$$\boldsymbol{o}_t = \sigma(\boldsymbol{W}_o * [\boldsymbol{X}_t, \boldsymbol{h}_{t-1}] + \boldsymbol{b}_o) \tag{2e}$$

$$\boldsymbol{h}_t = \boldsymbol{o}_t \odot \tanh(\boldsymbol{C}_t) \tag{2f}$$

In Eq. 2, $*$ indicates the convolutional operation and $\odot$ denotes the Hadamard product. Also, $\{\boldsymbol{W}_i, \boldsymbol{W}_f, \boldsymbol{W}_c, \boldsymbol{W}_o\}$ are the weight parameters of the model for the corresponding filters where $\{\boldsymbol{b}_i, \boldsymbol{b}_f, \boldsymbol{b}_c, \boldsymbol{b}_o\}$ represent bias vectors.

3.2 Additional techniques

The input and label data for a typical storm are shown earlier in Sect. 2 in Figs. 2 and 3. As discussed earlier, Fig. 2 clearly shows that some of the key input data (size and intensity of storm) do not vary substantially before the storms makes landfall, creating significant challenges for effective training. A standard ConvLSTM model is not able to learn from such data as it needs to establish a one-to-one learning [45]. Initial attempts to train such a standard ConvLSTM model to predict the upcoming storm surge based on the desired inputs were unsuccessful. Therefore, a few techniques were developed, discussed below, to adapt the model to the specifics of the provides datasets.

To accommodate the need to establish a sequence-to-sequence prediction model [45], an encoder–decoder, a popular approach of organizing recurrent neural networks for sequence-to-sequence prediction applications, is used. Encoder–decoder models are very capable with the sequential data [45, 50]. With a finely tuned LSTM layer, we can make a whole network perform appropriately with the sequential information of the data by making the network memorize the sequence. The encoder–decoder modeling involves two recurrent neural networks: one for reading the input sequence, called the encoder and a second to decode the encoded source sequence into the target sequence, decoding the fixed-length vector and outputting the predicted sequence, called the decoder. Here, our original model is combined with the encoder–decoder network model to build a high-performance model for the desired sequential data. The encoder–decoder structure is an end-to-end training; therefore, we do not need to explicitly train a latent representation model. This represents a key component of the proposed solution to the modeling process, because it is a dimension reduction technique [51] and therefore can capture the feature of latent space very well. Furthermore, it is very convenient and has already demonstrated effectiveness in many convolutional neural network-based models [52–54].

Inspired by the forward Euler scheme, a global residual connection is also designed. The residual connection is between the input state variable $\boldsymbol{u}_{\mathrm{i}}$ and the output variable $\boldsymbol{u}_{\mathrm{i+1}}$. The learning process at time instant $\mathrm{t_i}$ is formulated as $\boldsymbol{u}_{\mathrm{i+1}} = \boldsymbol{u}_{\mathrm{i}} + \delta \mathrm{t} \cdot \mathcal{NN}[\boldsymbol{u}; \boldsymbol{\theta}]$, where $\mathcal{NN}$ denotes the trained network operator and t is the time interval. Based on this formulation, the output state variable $\boldsymbol{u}_{\mathrm{i+1}}$ at time instant $\mathrm{t_i}$ switches into the input variable at t_{i+1}. These residual connection networks represent the second technique employed to improve the performance of the developed model.

The other technique we leverage is pixel shuffle [55], which is an upsampling strategy. Pixel shuffle maintains satisfactory reconstruction accuracy in image and video super-resolution tasks without high computational and memory costs [55]. In comparison with deconvolution [56], which always needs more layers to reach the expected resolution, pixel shuffle has lower computational complexity. Beyond that, another advantage of pixel shuffle is that it introduces fewer checkerboard artifacts compared with deconvolution [57]. The final developed model structure, incorporating all aforementioned advances, is shown in Fig. 5 where PS stands for pixel shuffle.

As shown in Fig. 5, the initial state variable indicated with $\boldsymbol{u}$ is provided to the encoder block with three convolutional layers indicated by light yellow color. The ConvLSTM cell is then fed with the encoder data output. Also, the hidden state and cell state, indicated with $\boldsymbol{h}_{\mathrm{t}}$ and $\boldsymbol{C}_{\mathrm{t}}$, respectively, are provided to the ConvLSTM cell of the next state. The output of the ConvLSTM is deconvoluted through the Pixel Shuffle layer and finally the data are passed through a dense layer in the decoder block. The output of the encoder, ConvLSTM cell, and the decoder, which is our trained network operator $\mathcal{NN}$ for this certain state, is finally multiplied by a constant, δt, and is summed up with the initial state $\boldsymbol{u}_0$ and labeled as the next state variable, $\boldsymbol{u}_1$. This process continues for all the states from the first time state to the final time state $\boldsymbol{u}_{\mathrm{T}}$ at t denoted in Fig. 5. The outcome of our neural network model is then compared with the true values and a minimization problem is solved in each epoch, before a same process is applied in the next epoch.

3.3 Training process

Considering the special shape of inputs shown for one storm in Fig. 2, i.e., the fact that, as stressed earlier, certain inputs are not changing very much over time, special attention needs to be given to data standardization. We first standardize these four inputs separately, one by one, using $\mathrm{x}' = \frac{\mathrm{x}-\mu}{\sigma}$ where μ and σ are the mean and standard deviation, respectively. The label data (the outputs), shown in Fig. 3, are also normalized in a way that they centralized around zero by means of the hyperbolic function. The model experiences a faster convergence for such normalized data.

Before discovering the data through the ConvLSTM cell, we pass the input data through the encoder to study the entire sequence of the data. The encoder contains three convolutional layers where the ReLU activation function is employed for these layers. The kernel size, padding size, and stride size for these three layers are 4×4, 1×1, and 2×2, respectively. These three layers' input and output channels are receptively 2 and 16, 16 and 32, and 32 and 64. Right before the encoder with these three convolutional layers, three linear layers are designed, fed with four inputs, and outputted the same dimension of label data, 4800 elements. For these linear layers, the hyperbolic activation function, tanh, is used. Once the data are provided in the latent space, a ConvLSTM cell is employed

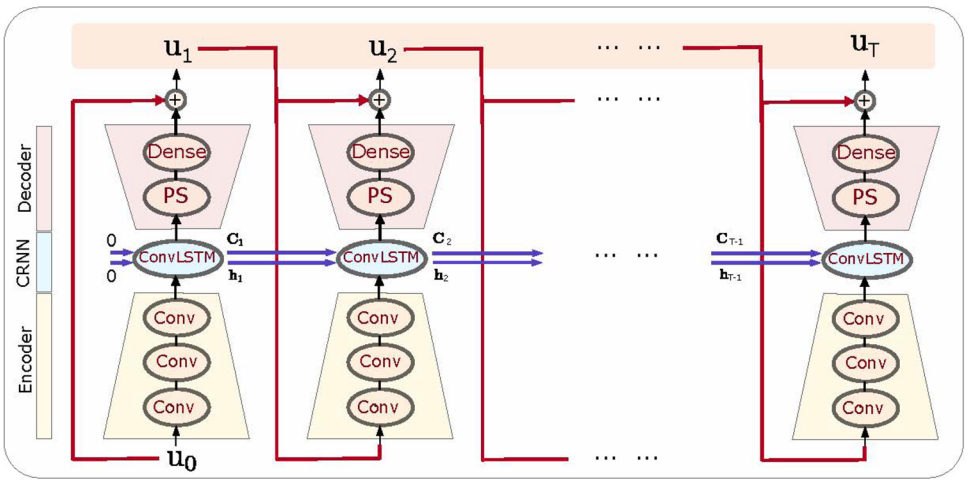

Fig. 5 Developed convolutional recurrent neural network (CRNN) structure

Table 1 Convolutional recurrent neural network model architecture

Cell	Layer	Filter/Upscale factor	Output
Input			[100, 1, 4]
	Dense		[100, 1, 40]
	Dense		[100, 1, 400]
	Dense		[100, 1, 4800]
	Reshape		[100, 1, 120, 40]
Encoder	Convolutional	[4, 4, 16]	[100, 16, 60, 20]
	Convolutional	[4, 4, 32]	[100, 32, 30, 10]
	Convolutional	[4, 4, 64]	[100, 64, 15, 5]
ConvLSTM	ConvLSTM	[5, 5, 64]	[100, 64, 15, 5]
Decoder	Pixel Shuffle	[8]	[100, 1, 120, 40]
	Reshape		[100, 1, 4800]
	Dense		[100, 1, 4800]
Output			[100, 1, 4800]

where the kernel size, padding size, and stride size are 5×5, 2×2, and 1×1, respectively. The input and output channels for the ConvLSTM cell are both 64. Note that model training in the latent space, where the ConvLSTM layer is the optimal space to train the model. The only two layers in the decoder are upsampling through pixel shuffle explained in Sect. 3.2 and a final linear layer that outputs the same shape of data as label data. It should be pointed out that the pixel shuffling decreases the channel size from 64 to 1, as a pixel shuffle layer with an upscale factor 8 is applied. It also increases the height and width of data by 8. Table 1 summarizes all the employed layers, filter sizes, and outputs of each layer separately. The training is carried out with a batch size of 100. Therefore, each batch contains 100 storm data, and the steps above are repeated for all 125 time steps.

The model is trained with 35,000 epochs, and the learning rate is selected as $1e-4$. The L_2 norm loss function is minimized over the epochs by the mini-batch gradient descent method as follows:

$$\mathcal{L}(\boldsymbol{W},\boldsymbol{b}) = ||\mathcal{C}(t,\boldsymbol{x},\boldsymbol{\theta}) - \boldsymbol{z}_L(t,\boldsymbol{x},\boldsymbol{\theta};\boldsymbol{W},\boldsymbol{b})||_{\mathrm{L}_2(\Omega)} \tag{3a}$$

$$\boldsymbol{W}^*,\boldsymbol{b}^* = \arg\min_{\boldsymbol{W},\boldsymbol{b}} \mathcal{L}(\boldsymbol{W},\boldsymbol{b}) \tag{3b}$$

In Eq. 3, $\mathcal{L}(\cdot)$ stands for the loss function and the L_2 norm is indicated with $||\cdot||_{\mathrm{L}_2(\Omega)}$. The CFD solution (storm surge database predictions) is denoted with $\mathcal{C}(t,\boldsymbol{x},\boldsymbol{\theta})$ and $\boldsymbol{W}^*,\boldsymbol{b}^*$ represent the (sub)optimal neural network parameters, the weights and biases obtained from the optimization problem.

The hyper-parameters are set initially randomly. Many models with different sets of hyper-parameters are run in parallel to find the models whose loss values converge over epochs. Once the trainable models with specified hyper-parameters are determined, the hyper-parameters are evaluated in random search [58] to find the optimal set of hyper-parameters. The loss function over epoch numbers for the optimal model trained on the studied storm datasets is shown in Fig. 6.

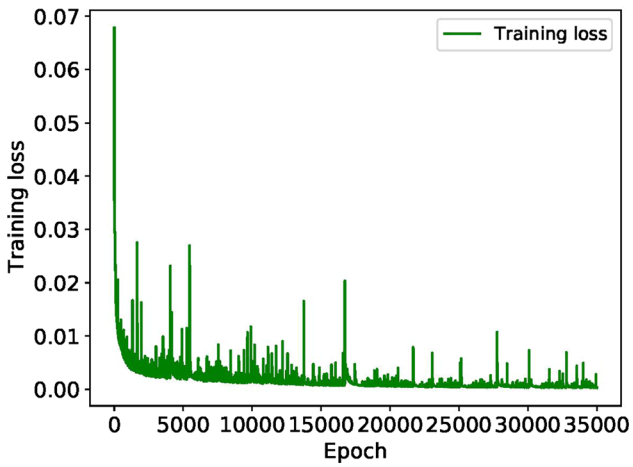

Fig. 6 Loss function over epochs

As Fig. 6 shows, the loss error decreases continuously over epochs from about $7e-2$ to $3e-4$, which shows a significant reduction. Once the model is trained, it is ready to predict the storm surges for new storms with the provided input values. The next section assesses the train model evaluation and the predicted storm surge's accuracy.

Before concluding this section, we would like to point out that the proposed neural network model was derived after examining different alternative ones. These are reviewed in Table 2. Effort initiated by examining simpler models and as such models were not able to learn from the data and their trends, more complicated models were investigated leading eventually to the proposed model, which was demonstrated to that it could accurately learn

Table 2 Comparison of the applied neural network model structures

Model	Layer	Temporal correlation	Spatial correlation	Training result
MLP	Dense	No	No	Failed
MLP	Dense/CNN	No	Yes	Failed
MLP/LSTM	Dense	Yes	No	Failed
MLP/LSTM	Dense/CNN	Yes	Yes	Failed
MLP/ConvLSTM	Dense/CNN	Yes	Yes	Failed
CRNN	Dense/CNN	Yes	Yes	Succeed

from the data. Improvements across the models were established by considering greater sophistication in the way correlation of data spatially and temporally is treated. Therefore, we tried different model types and layer types, and finally, the encoder–decoder modeling was the key solution of our modeling process. This led eventually to the developed convolutional recurrent neural network (CRNN) structure which is a combination of multilayer perceptron (MLP), ConvLSTM, residual connections, and encoder–decoder neural network modeling. Note that in Table 2, the models that are not able to train from the data, and their insight are indicated with "Failed," while the only model which was able to learn well from the provided data and captured their correlation both spatially and temporally is denoted with "Succeed."

Once we reached to our final developed convolutional recurrent neural network model (CRNN) structure, the hyperparameter tuning was carried out. The average RMSEs and MAEs were compared and the best model was chosen for further investigation. The hyperparameter tuning summary is shown in Table 3.

In Table 3, n, m, and q are set of numbers $\{2, 3, 4\}$ and the developed models are numbered from one to four based on their layers' activation functions as different types of activation functions and are used for our four sub-layers including: (a) pre-encoder layers, (b) encoder layers, (c) main-cell layer (ConvLSTM), and (d) decoder layers. Many models and sub-models are trained based on the introduced four models with defined activation layers and different hyperparameters, as shown in Table 3. For each main model, the smallest relative error to the other models is reported and compared. The best hyperparameter setting which was a sub-model of CRNN 1 with the layers' structures and hyperparameters mentioned in Table 3 and with other hyperparameters described in detail in the beginning of Sect. 3.3 is chosen based on the reported error.

4 Model evaluation

Eight synthetic storms within the original database, not utilized in the training phase, are now used to validate the performance of the developed convolutional recurrent neural network model. The predicted surges are compared to the label test data, corresponding to the simulated surge for the same SPs and time steps utilized in the model development. An alternative surrogate model implementation is also considered in this section, a Gaussian process emulator. Approach utilizes a simplified parameterization of the storm input, using instantaneous storm features close to landfall to characterize each storm and considers independent predictions for the surge for each SP or time step, using principal component analysis to incorporate spatio-temporal correlation features in the surge output predictions. Further details for this formulation are discussed in [10]. The root-mean-square errors (RMSE), defined in Eq. 4, and mean absolute error, defined in Eq. 5, of the test set are reported in Tables 4 and 5, respectively, separately for each of the eight storms. Note that the L_1 norm in Eq. 5 is indicated with $||\cdot||_{L_1(\Omega)}$.

$$RMSE = ||\mathcal{C}(t, \boldsymbol{x}, \boldsymbol{\theta}) - \boldsymbol{z}_L(t, \boldsymbol{x}, \boldsymbol{\theta})||_{L_2(\Omega)} \tag{4}$$

$$MAE = ||\mathcal{C}(t, \boldsymbol{x}, \boldsymbol{\theta}) - \boldsymbol{z}_L(t, \boldsymbol{x}, \boldsymbol{\theta})||_{L_1(\Omega)} \tag{5}$$

The comparisons in Tables 4 and 5 across the RMSE and MAE of the predictions by the developed neural network model, and the Gaussian process show that the neural network model offers greater accuracy storm surge predictions than the Gaussian process for all of the storm datasets. The average RMSE and MAE of the predictions for these eight test samples is $5.312e-2$ and $3.812e-2$, respectively, and for Gaussian process is $8.793e-2$ and $5.946e-2$, respectively, which shows improvement by about $\%50$.

Table 3 Hyperparameter tuning

Model	Layers	Activation function	Learning rate	Batch size	Filter size	Residual connection constant	Epoch	Best relative error
CRNN 1	n*FC/p*CNN/ConvLSTM/PC-q*FC	Tanh/ReLu/ReLu/Tanh	0.01/0.001	50/100/250	3*3, 4*4, 5*5	0.01/0.001	20/35/50k	1.00
CRNN 2	n*FC/p*CNN/ConvLSTM/PC-q*FC	ReLu/ReLu/ReLu/Tanh	0.01/0.001	50/100/250	3*3, 4*4, 5*5	0.01/0.001	20/35/50k	1.67
CRNN 3	n*FC/p*CNN/ConvLSTM/PC-q*FC	Tanh/Tanh/ReLu/Tanh	0.01/0.001	50/100/250	3*3, 4*4, 5*5	0.01/0.001	20/35/50k	1.38
CRNN 4	n*FC/p*CNN/ConvLSTM/PC-q*FC	ReLu/Tanh/ReLu/Tanh	0.01/0.001	50/100/250	3*3, 4*4, 5*5	0.01/0.001	20/35/50k	1.26

Furthermore, the RMSE and MAE for the times corresponding to the ten percent highest storm surge, H(1/10), are calculated and presented in Tables 6 and 7, respectively. These metrics examine accuracy over the larger surge values for each time series and therefore provide key validation information for assessing emulator accuracy, emphasizing the performance close to the peak surge, whose prediction is highly relevant in surge forecasting.

As it is shown in Tables 6 and 7 by comparing the RMSE and MAE of the predictions by the developed neural network model and the Gaussian Process for every single storm, it can be inferred that the neural network model offers greater accuracy storm surge predictions than the Gaussian process for all of the storm datasets. The average RMSE and MAE of the predictions for these eight tests is $7.884e-2$ and $7.385e-2$, respectively, and for Gaussian process is $1.534e-1$ and $1.426e-1$, respectively, which shows at least a two-times less error in total.

The training parameters, the computational time needed, and the computational complexity of these two approaches are also compared and summarized in Table 8.

As shown in Table 8, substantially larger number of parameter are used in our developed neural network compared to the GP, something that explains the differences in computational complexity in training (calibration) and testing (predictions) for the neural network, as well as the higher predictive accuracy it enjoys. The training time is not a significant concern for us, as we have ample resources and time to train our model. However, test time is of utmost importance since we are constrained to predicting storm surges within a few hours after the storm information is reported until it reaches the coast. With a test time of only 318 s by our developed model, which is considerably short compared to the time permitted, there was no cause for concern in this aspect. Moreover, the accuracy that we obtained from the developed model is of greater importance to us.

For one of the test datasets, we further look at true values and predictions provided by the developed neural network model and the Gaussian process method in Fig. 7. In this figure, the line $x = y$ is also plotted, reflecting perfect correlation between the predicted and true values.

It is evident from Fig. 7 that the neural network model enjoys substantial higher correlation between predictions and true responses, offering additional validation of the high accuracy trends reported earlier in Tables 6 and 7.

Moreover, to further confirm our observations, we compare the predictions by the developed neural network model and the Gaussian Process to the true storm surges for the eight test datasets for a specific sample location (point)

Table 4 RMSE of test datasets

	Test 1	Test 2	Test 3	Test 4	Test 5	Test 6	Test 7	Test 8
CRNN	1.089e − 2	2.799e − 2	6.914e − 2	2.749e − 2	4.351e − 2	6.677e − 2	4.946e − 2	1.302e − 1
GP	2.111e − 2	8.912e − 2	1.339e − 1	1.523e − 1	5.178e − 2	9.386e − 2	4.663e − 2	1.148e − 1

Table 5 MAE of test datasets

	Test 1	Test 2	Test 3	Test 4	Test 5	Test 6	Test 7	Test 8
CRNN	8.700e − 3	1.930e − 2	5.540e − 2	2.170e − 2	2.840e − 2	4.750e − 2	3.410e − 2	8.990e − 2
GP	1.730e − 2	7.030e − 2	6.240e − 2	1.115e − 2	5.020e − 2	5.480e − 2	3.120e − 2	7.800e − 2

Table 6 RMSE of H(1/10) test datasets

	Test 1	Test 2	Test 3	Test 4	Test 5	Test 6	Test 7	Test 8
CRNN	1.250e − 2	4.430e − 2	7.495e − 2	3.070e − 2	8.070e − 2	1.091e − 1	4.060e − 2	2.379e − 1
GP	2.300e − 2	2.966e − 1	1.834e − 1	2.847e − 1	9.590e − 2	1.295e − 1	5.000e − 2	1.643e − 1

Table 7 MAE of H(1/10) test datasets

	Test 1	Test 2	Test 3	Test 4	Test 5	Test 6	Test 7	Test 8
CRNN	1.150e − 2	4.060e − 2	6.910e − 2	2.840e − 2	7.780e − 2	1.018e − 1	3.700e − 2	2.246e − 1
GP	2.040e − 2	2.875e − 1	1.636e − 1	2.730e − 1	9.220e − 2	1.169e − 1	3.980e − 2	1.473e − 1

Table 8 Developed convolutional recurrent neural network vs Gaussian process

	Training parameters	Training time (s)	Test time (s)	Time complexity
CRNN	899,853	274683	318	$\mathcal{O}(n)$
GP	154	1800	1	$\mathcal{O}(n^3)$

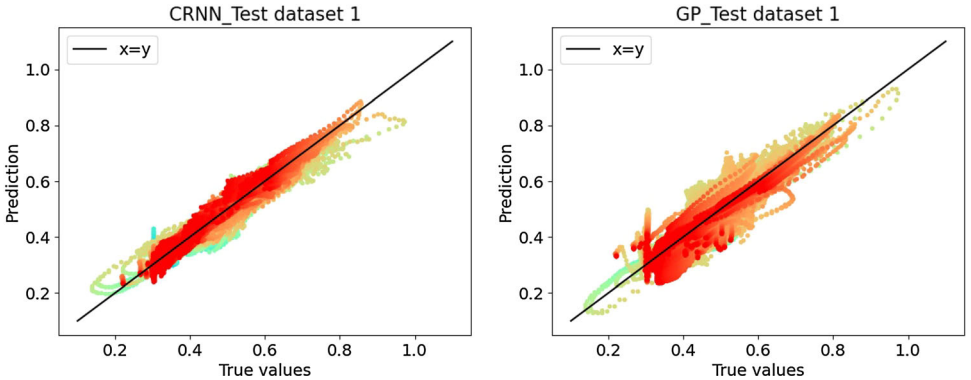

Fig. 7 Storm surge prediction vs. true test data provided by CRNN and GP

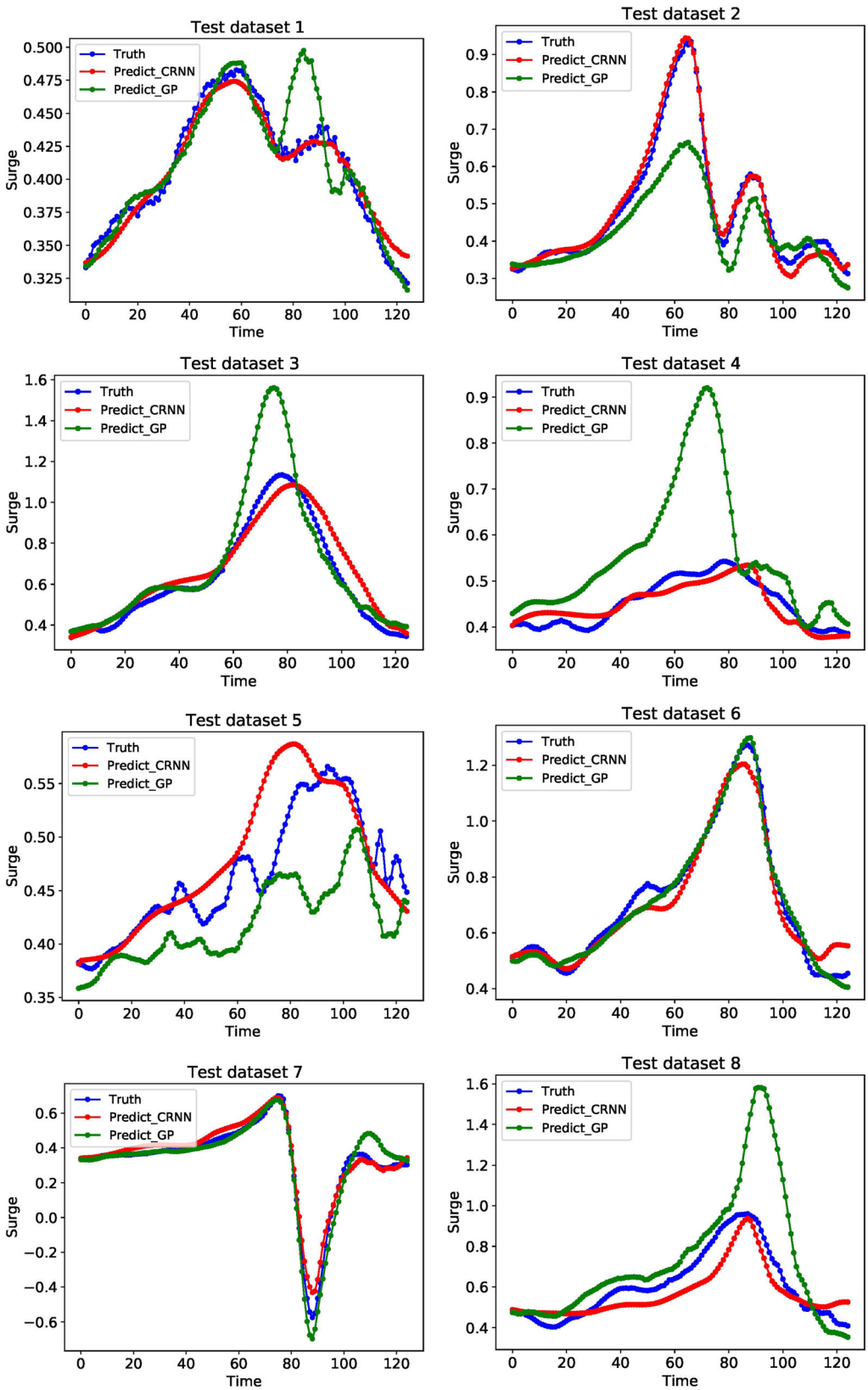

Fig. 8 Storm surge predictions for one grid SP in different test storms for a grid point in the coast middle layers

in the coast in Fig. 8. The specific location corresponds to a middle layer in the ordered response for the grid points.

As shown in Fig. 8, the storm surge predictions computed by the developed convolutional recurrent neural network in all the studied tests are very close to the true values for the entire time interval. The Gaussian process provides less accurate storm surge predictions as the predictions are generally further away from the true surge values. However, the Gaussian process has partially learned the data trend, and its surge predictions can somewhat mimic the surge true values' trend. Two more SPs from very early layers and end layers of the response grid are chosen, and the predictions provided by the two approaches are compared with the true values in Appendix 1. The results reported in the Appendix follow the same trends discussed above allowing a generalization of the observations.

5 Conclusions

This study examined the development of a neural network for emulating time-series surge predictions using a database of synthetic storm simulations. The developed convolutional recurrent neural network model is enriched by an encoder–decoder model, so that the developed model takes the entire sequence of the data into account. Therefore, the entire storm surge can be predicted based on the storm-driven parameters' complete history. The encoder–decoder add-on ultimately makes the developed neural network model a sequence-to-sequence (seq2seq) storm surge forecast model. Also, the model's performance is improved by incorporating a residual connection network. Several additional techniques are applied in the training process to further improve predictive accuracy. Overall, the spatial and temporal correlations of the data are captured by employing convolutional neural network layers and the recurrent neural network, respectively, through a ConvLSTM cell. The ConvLSTM cell is trained on the data provided in the latent space right between encoder and decoder cells, accommodating better learning for the ConvLSTM cell. In contrast to previous storm surge prediction studies, where machine learning methods were predominantly used as black boxes and surge for a few representative stations was only predicted, the aforementioned formulation allows us to predict surge for all the save points within the domain of interest by establishing problem-specific advances for the neural network implementation. Furthermore, through these formulations, the correlations of data both spatially and temporally are learned by the model to enhance prediction accuracy, something that further improves upon past efforts. The main novelty of this manuscript from the storm surge application perspective is that it establishes spatio-temporal predictions across a large geographic domain. Previous studies, as presented earlier, have either focused on peak surge predictions (not addressing the time evolution of the surge) or have considered time-series surge predictions for a moderate only number of spatial nodes. For artificial neural network applications, this has allowed previous studies to consider independent formulations across the different nodes, with no requirement on the trained network to describe the spatial variability of the surge additionally. The evaluation of the trained model on test datasets show that the model can accurately predict the storm surge. The develop model can, ultimately, accommodate fast predictions for the time-series surge evolution, driven by track/-size/intensity storm input features, and can be used to support efficient risk assessment and emergency response management operations.

Appendix: Test grids

In this section, the comparison of the prediction by the developed convolutional recurrent neural network and Gaussian process is shown in Figs. 9 and 10. Figure 9 shows the comparison for a grid from the early layers of the coast, and Fig. 10 shows a grid from the last layer of grids in the coast. As it is mentioned in Sect. 4, the results by convolutional recurrent neural network model are much better and more accurate than the Gaussian process.

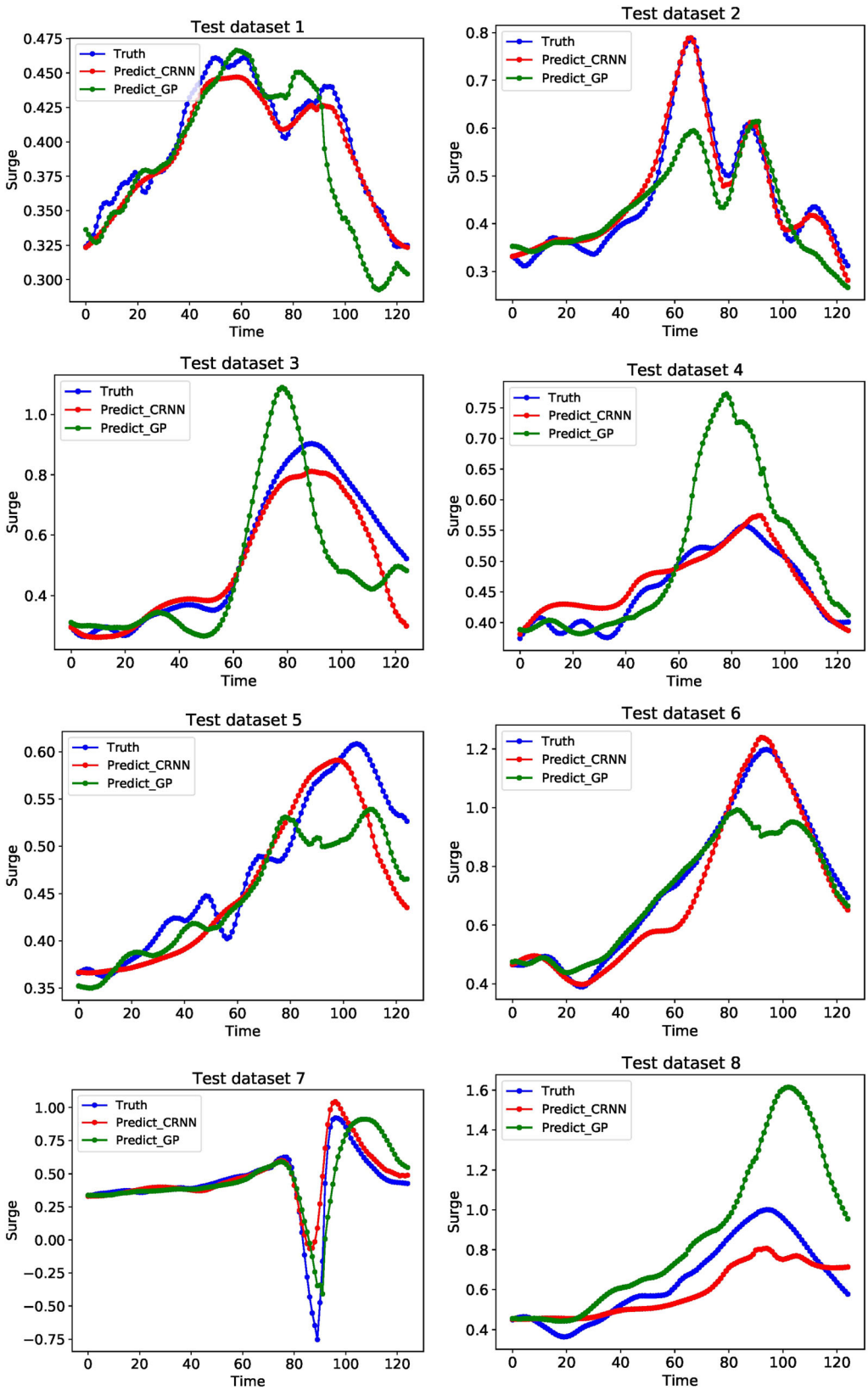

Fig. 9 Storm surge predictions for one grid SP in different test storms for a grid point in the coast front layers

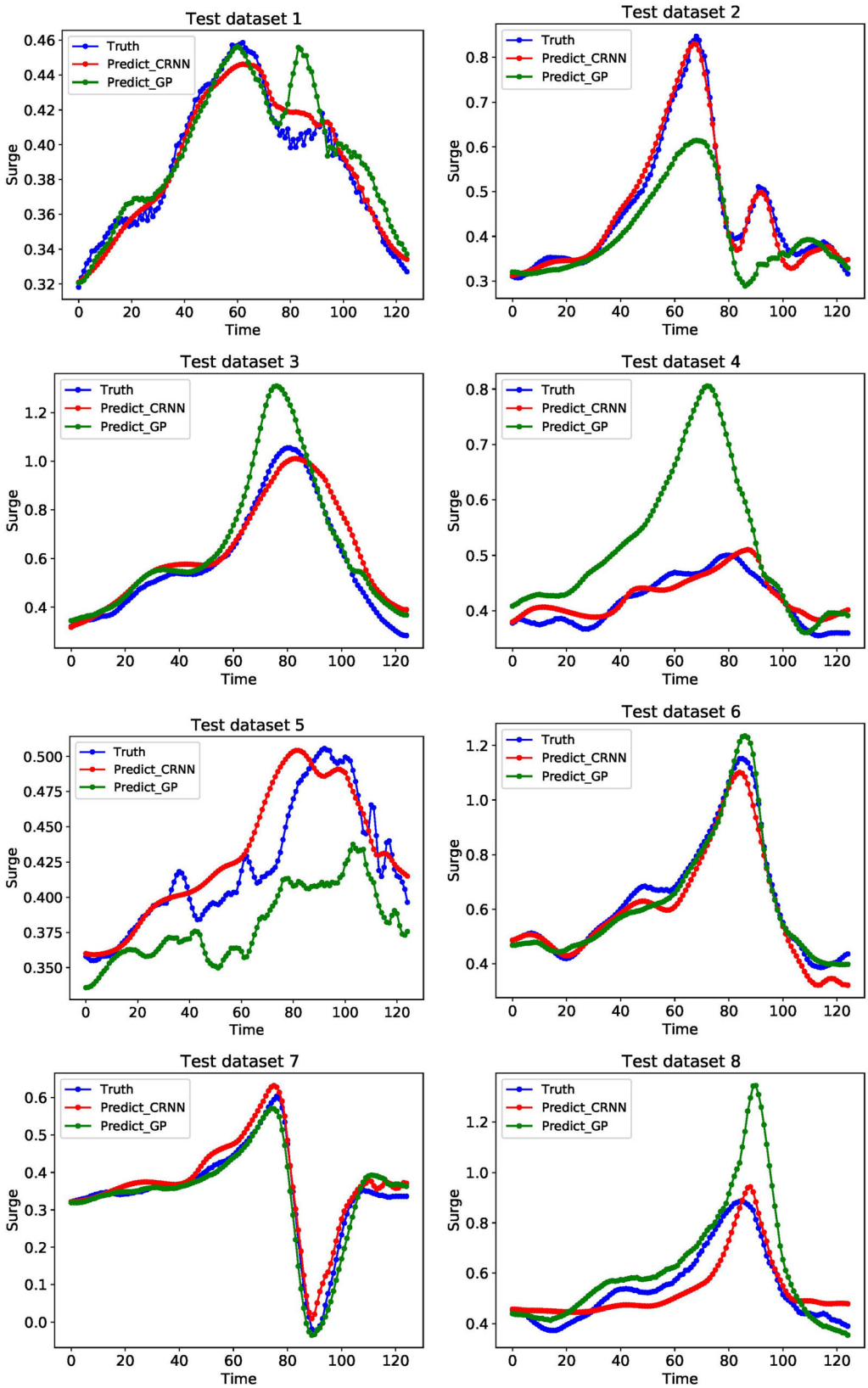

Fig. 10 Storm surge predictions for one grid SP in different test storms for a grid point in the coast back layers

Acknowledgements Authors would like to thank the Army Corp of Engineers, Coastal Hydraulics Laboratory of the Engineering Research and Development Center for providing access to the storm surge data used in the illustrative case study, through the coastal hazards system (https://chs.erdc.dren.mil/).

References

1. Hallegatte S, Patmore N, Mestre O, Dumas P, Corfee-Morlot J, Herweijer C, Muir-Wood R (2008) Assessing climate change impacts, sea level rise and storm surge risk in port cities: a case study on Copenhagen. OECD Environment Working Papers(3), 0_1,
2. Luettich RA, Westerink JJ, Scheffner NW (1992) ADCIRC: An advanced three-dimensional circulation model for shelves, coasts, and estuaries. Report 1. Theory and methodology of ADCIRC-2DDI and ADCIRC-3DL. Dredging Research Program Technical Report DRP-92-6, U.S Army Engineers Waterways Experiment Station, Vicksburg, MS,
3. Westerink J, Luettich R, Feyen J, Atkinson J, Dawson C, Roberts H, Powell M, Dunion J, Kubatko E, Pourtaheri H (2008) A basin- to channel-scale unstructured grid hurricane storm surge model applied to Southern Louisiana. Mon Weather Rev 136:833–864
4. Booij N, Holthuijsen LH, Ris RC (1996) The SWAN wave model for shallow water. 25th International Conference on Coastal Engineering, Orlando, FL, 668-676,
5. Smith J. M, Sherlock AR, Resio DT (2001) STWAVE: Steady-state spectral wave model user's manual for STWAVE, Version 3.0. DTIC Document,
6. Kyprioti AP, Adeli E, Taflanidis AA, Westerink JJ, Tolman HL (2021) Probabilistic storm surge estimation for landfalling hurricanes: advancements in computational efficiency using quasi-Monte Carlo techniques. J Marine Sci Eng 9(12):1322. https://doi.org/10.3390/jmse9121322
7. Irish JL, Resio DT, Cialone MA (2009) A surge response function approach to coastal hazard assessment. Part 2: Quantification of spatial attributes of response functions. Natural hazards 51(1):183–205
8. Jia G, Taflanidis A (2013) Kriging metamodeling for approximation of high-dimensional wave and surge responses in real-time storm/hurricane risk assessment. Comput Methods Appl Mech Eng 261:24–38
9. Kim S, Melby J, Nadal-Caraballo NC, Ratcli J (2015) A time-dependent surrogate model for storm surge prediction based on an artificial neural network using high-fidelity synthetic hurricane modeling. Nat Hazards 76(1):565–585
10. Jia G, Taflanidis A, Nadal-Caraballo N, Melby J, Kennedy A, Smith J (2016) Surrogate modeling for peak or time-dependent storm surge prediction over an extended coastal region using an existing database of synthetic storms. Nat Hazards 81:909–938
11. Al Kajbaf A, Bensi M (2020) Application of surrogate models in estimation of storm surge: A comparative assessment. Applied Soft Computing, 106184,
12. Contento A, Xu H, Gardoni P (2020) Probabilistic formulation for storm surge predictions. Struct Infrastruct Eng 16(4):547–566
13. Traffic Flow Prediction for Urban Road Sections Based on Time Series Analysis and LSTM-BILSTM Method. IEEE Transactions on Intelligent Transportation Systems. Volume 23(6), 5615-5624, 2022
14. Chen X et al (2020) Sensing Data Supported Traffic Flow Prediction via Denoising Schemes and ANN: a comparison. IEEE Sens J 20(23):14317–14328
15. Xiao G et al. Exploring influence mechanism of bikesharing on the use of public transportation - a case of Shanghai. https://doi.org/10.1080/19427867.2022.209328
16. Resio D, Westerink J (2008) Modeling the physics of storm surges. Physics Today,
17. Kijewski-Correa T, Taflanidis A, Vardeman C, Sweet J, Zhang J, Snaiki R, Wu T, Silver Z, Kennedy A (2020) Geospatial environments for hurricane risk assessment: applications to situational awareness and resilience planning in New Jersey. Front Built Environ 6:549106
18. Nadal-Caraballo NC, Campbell MO, Gonzalez VM, Torres MJ, Melby MJ, Taflanidis AA (2020) Coastal hazards system: a probabilistic coastal hazard analysis framework. J Coastal Res 95(sp1):1211–1216
19. Toro G, Resio D, Divoky D, Niedoroda A, Reed C (2010) Efficient joint-probability methods for hurricane surge frequency analysis. Ocean Eng 37(1):125–134
20. Lee J-W, Irish JL, Bensi MT, Marcy DC (2021) Rapid prediction of peak storm surge from tropical cyclone track time series using machine learning. Coast Eng 170:104024
21. RamosValle AN, Curchitser EN, Bruyere CL, McOwen S (2021) Implementation of an Artificial Neural Network for Storm Surge Forecasting. Journal of Geophysical Research: Atmospheres 126(13):
22. Kim R, So C, Jeong M, Lee S, Kim J, Kang J (2019) HATS: A Hierarchical Graph Attention Network for Stock Movement Prediction. arXiv preprint arXiv:1908.07999,
23. Namdari A, Durrani T (2021) HATS: A Multilayer Feedforward Perception Model in Neural Networks for Predicting Stock Market Short-term Trends. Operations Research Forum volume 2, Article number: 38,
24. Jelesnianski C. P, Chen J, Shaffer W. A (1992) SLOSH: Sea, lake, and overland surges from hurricanes. NOAA Technical Report, NWS 48. US Department of Commerce, National Oceanic and Atmospheric Administration,
25. Holland GJ (2008) A revised hurricane pressure-wind model. Mon Weather Rev 136(9):3432–3445
26. Holland GJ, Belanger JI, Fritz A (2010) A revised model for radial profiles of hurricane winds. Mon Weather Rev 138(12):4393–4401
27. Marks FD (2003) Hurricanes. Handbook of Weather, Climate, and Water: Dynamics, Climate, Physical Meteorology, Weather Systems, and Measurements 641–675
28. Di Liberto T, Colle BA, Georgas N, Blumberg AF, Taylor AA (2011) Verification of a multimodel storm surge ensemble around New York City and Long Island for the cool season. Weather Forecast 26(6):922–939
29. Dresback K, Fleming J, Blanton B, Kaiser C, Gourley J, Tromble E, Kolar R, Hong Y, Cooten S, Vergara H, Flamig Z, Lander H, Kelleher K, Nemunaitis-Monroe K (2013) Skill assessment of a real-time forecast system utilizing a coupled hydrologic and coastal hydrodynamic model during Hurricane Irene. Cont Shelf Res 71:78–94
30. Davis JR, Paramygin VA, Forrest D, Sheng YP (2010) Toward the probabilistic simulation of storm surge and inundation in a limited-resource environment. Mon Wea Rev 138(7):
31. Bernier NB, Thompson KR (2015) Deterministic and ensemble storm surge prediction for Atlantic Canada with lead times of hours to ten days. Ocean Model 86:114–127
32. Taylor AA, Glahn B(2008) Probabilistic guidance for hurricane storm surge. 19th Conference on probability and statistics, ,
33. Lee T (2006) Neural network prediction of a storm surge. Ocean Eng 33:483–494
34. Lee T (2008) Back-propagation neural network for the prediction of the short-term storm surge in Taichung harbor, Taiwan. Eng Appl Artif Intell 21:63–72

35. Lee T (2009) Predictions of typhoon storm surge in Taiwan using artificial neural networks. Adv Eng Softw 40:1200–1206
36. De Oliveira M, Ebecken F, De Oliveira F, de Azevedo Santos I (2009) Neural network model to predict a storm surge. J Appl Meteorol Climatol 48 (1), 143-155,
37. Bajo M, Umgiesser G (2010) Storm surge forecast through a combination of dynamic and neural network models. Ocean Model 33(1):1–9
38. Hashemi M, Spaulding M, Shaw A, Farhadi H, Lewis M (2016) An efficient artificial intelligence model for prediction of tropical storm surge. Nat Hazards 82(1):471–491
39. Kim S, Pan S, Mase H (2019) Artificial neural network-based storm surge forecast model: Practical application to Sakai Minato Japan. Appl Ocean Res 91:101871
40. Chao W, Young C, Hsu T, Liu W, Liu C (2020) Long-lead-time prediction of storm surge using artificial neural networks and effective typhoon parameters: revisit and deeper insight. Water 2020 12(9):2394
41. Das H, Jung H, Ebersole B, Wamsley T, Whalin R (2011) An efficient storm surge forecasting tool for coastal Mississippi. Coastal Eng Proceed 1(32):21
42. Alemany S, Beltran J, Perez A, Ganzfried S Predicting Hurricane Trajectories using a Recurrent Neural Network. arXiv:1802.02548v3
43. Chen K, Kuang C, Wang L, Chen K, Han X, Fan J (2022) Storm surge prediction based on long short-term memory neural network in the east china sea. Appl. Sci. 2022, 12(1):181
44. Igarashi Y, Tajima Y (2021) Application of recurrent neural network for prediction of the time-varying storm surge. Coast Eng J 63(1):68–82
45. Sutskever I, Vinyals O, Le Q (2014) Sequence to sequence learning with neural networks, arXiv preprint arXiv:1409.3215,
46. Kyprioti A, Irwin C, Taflanidis A, Nadal-Caraballo N, Yawn M, Aucoin L (2023) Spatio-temporal storm surge emulation using Gaussian Process techniques. Coast Eng 180:104231
47. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780
48. Dupond S (2019) A thorough review on the current advance of neural network structures. Annu Rev Control 14:200–230
49. Shi X, Chen Z, Wang H, Yeung D-Y, Wong W-K, Woo W-C (2015) Convolutional lstm network: a machine learning approach for precipitation nowcasting. Adv Neural Inf Process Syst 28:802–810
50. Graves A (2013) Generating sequences with recurrent neural networks, arXiv preprint arXiv:1308.0850,
51. van der Maaten L, Postma E, van den Herik H (2009) Dimensionality reduction: a comparative review. Tilburg University, Tech. rep.
52. Bengio Y (2009) Learning Deep Architectures for AI. Found Trends Mach Learn 2(1):1–127. https://doi.org/10.1561/2200000006
53. Krizhevsky A, Sutskever I, Hinton G (2012) Imagenet classification with deep convolutional neural networks, in: Advances in neural information processing systems,pages 1097-1105,
54. Han G, Sun L, Wang J (2021) PhyGeoNet: Physics-informed geometry-adaptive convolutional neural networks for solving parameterized steady-state PDEs on irregular domain. J Comput Phys 428,
55. Shi W, Caballero J, Huszar F, Totz J, Aitken A. P, Bishop R, Rueckert D, Wang Z (2016) Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network, In: Proceedings of the IEEE conference on computer vision and pattern recognition, 1874-1883,
56. Shan Q, Li Z, Jia J, Tang C (2008) Fast image/video upsampling. ACM Trans Graph (TOG) 27(5):1–7
57. Odena A, Dumoulin V, Olah C (2016) Deconvolution and checkerboard artifacts. Distill 1(10):e3
58. Bergstra J, Bengio Y (2012) Random search for hyper-parameter optimization. J Mach Learn Res 13:281–305